Relay Protocol

Learn about the Relay Protocol

Introduction

Relay Protocol is a cross-chain payments system that connects users to relayers willing to perform onchain actions on their behalf. Relay's design minimizes gas costs, enables rapid chain expansion, and maximizes capital efficiency, all while remaining open and trustless. Such a system is particularly well suited for low-value, high frequency cross-chain actions such as those expected in a world of chain abstraction.

Background: Cross-Chain Relaying

Relay is similar to other "intent-based" protocols like Across, where liquidity providers known asRelayers use their own capital to "fast fill" user requests to move between chains, and then rebalance using slow, expensive bridges under the hood.

This design has two key advantages over typical bridging:

- Speed
-
    - Because relayers are fronting their own capital, they can take on confirmation risk and fill optimistically, without waiting for global consensus
- Cost
-
    - Because rebalancing is done infrequently and in batches, the security cost of using bridges is amortized across many users

The typical flow for intent-based protocols is as follows:

1. Userescrows
2. funds on the origin chain
3. Relayerfills
4. the order on the destination chain
5. Relayer latersettles
6. the order on the origin, to claimpayment

Each of these steps has a cost, which different protocols minimize in different ways. Across, which pioneered this design, reduces the cost of settlement with two optimizations:

- Instead of proving every fill with an oracle or cross-chain message, settlement is done optimistically, by assuming it is correct, and having a challenge period
- Settlement is done in large batches across the whole protocol, every 4 hours

As a result, the vast majority of the remaining cost comes from theDeposit (~77,000 gas) andFill (~120,000 gas). This is whereRelay is able to achieve significant improvements:

Relay's Design

Relay shares the same core components as as other escrow systems: escrow, fulfillment, settlement, and payment. However, it changes who, how and where they are executed, to improve efficiency.

Settlement Chain

The key difference is that instead of users depositing funds into escrow on every chain that they want to pay from, relayers deposit into escrow on a single dedicated "Settlement Chain".

A number of significant benefits flow from this change:

- Gas efficiency

-
    - Because users aren't the ones escrowing funds, they can send payment directly to the relayer as an efficient raw transfer (only 21,000 gas)
- Deployment efficiency
-
    - Because all escrow and settlement happens on a single dedicated chain, supporting new origin / destination chains is much simpler
- Capital efficiency
-
    - Because this settlement chain can be a low-cost L2, orders can be settled immediately, preventing relayer capital from being locked up for hours
- Fee efficiency
-
    - Because relayers hold ETH on this low-cost L2, granular fees can be paid out with near-zero overhead

Each of these benefits is discussed more below.

Direct Transfers

In Relay, the flow becomes:

1. Relayersescrow
2. funds on the "Settlement Chain"
3. User sendspayment
4. directly to the relayer wallet on the origin
5. Relayerfills
6. directly to the user wallet on the destination
7. Relayer latersettles
8. , freeing their escrow to be used again in a future order

The biggest unlock from this is that because the relayer's escrowed funds act as a bond, the user and relayer can safely make direct transfers to each other, reducing the deposit and fill to highly efficient direct transfers (~21,000 gas each). This makes it up to 5x cheaper than alternative relaying protocols.

This can make a huge difference to the sort of use-cases that are possible to implement cross-chain, especially when transaction values are low (e.g. 1 NFT mints).

Intuitively, it might seem insecure to send funds directly to the relayer, but the risk and UX is similar to the typical flow:

- User is transferring funds into a state where they no longer control them
- The full value of their transaction is locked in escrow as collateral
- In the event of the failure, user has a mechanism to get refunded, after a delay

The main downside of direct transfers is that there is increased risk of user error. However, this can be counteracted with robust client-side validation. For many applications, the dramatically lower costs are worth the trade-offs. For others, there is always the option to use onchain validation on top of the base protocol.

Single Contract

Interoperability protocols are often slow in expanding to new chains for a number of reasons:

- Global consensus must be achieved amongst a set of validators or token holders to begin supporting a chain
- Canonical smart contracts need to be deployed and managed on every supported chain by the core team
- Liquidity pools need sufficient depth to offer good prices

As a result, most protocols are not well suited to supporting the long tail of chains in a world of abundant L2s. Relay was designed with this in mind.

Because settlement happens on a single dedicated chain, and asset transfers happen through direct transfers, it means there are no smart contracts required on the origin and destination chains. All it takes is a single relayer that is willing to offer service on that chain, and it can be supported. No permission or deployments are required, while still being able to tap into the security that the protocol provides.

Immediate Settlement

Another advantage of having all settlement happen on a single chain is that you can choose a chain with optimal properties. On other protocols, where settlement happens on all chains, including high-security chains like Ethereum, batch settlement is critical to keep costs low. However, batch settlement has downsides:

- Capital inefficiency - Relayer funds are tied up in between batches (4 hours for Across)
- Inflexibility - All orders must use the same settlement mechanism

For Relay, settlement is done on a low-cost L2 with a trust-minimized bridge to Ethereum, making it viable to settle every order individually and immediately. This adds some costs, but very minor, and unlocks massive capital efficiency and flexibility gains.

Consider the case where orders are settled every 2 minutes, which is comparable to other interoperability protocols that do message validation like Wormhole, LayerZero, etc. This is around 240 times quicker than Across. Once you factor in that only half the relayer's capital can be used for filling (because the rest is used as escrow), it works out around 120x more capital efficient. This means Relayer can either earn more or charge less for the same amount of capital.

One other nice thing about individual settlement is that it doesn't require global consensus or third party validation. You just need the two parties involved in the transaction, the user and the relayer, to agree that the order was filled. This allows even more aggressive settlement time. In Relay, actors will be incentivized (through reputation, bond penalties, etc) to quickly "self-settle" orders and only rely on 3rd party validation to settle disputes. This can result in even greater capital efficiency for relayers.

Relay ETH

One major factor that needs to be considered when striving to reduce costs is the efficient collection of fees. It's quite common for fees to be collected by various actors in the order pipeline, beyond just the relayer:

- Protocol fees
- Interface / wallet fees
- Aggregator API fees
- Referrer fees

In other protocols, these fees are collected on either the origin or destination chain. This can add considerable gas overhead, which is either passed onto the user in the form of increased costs, or forces participants to find other mechanisms to earn revenue.

Relay solves this problem by collecting fees on the low cost settlement chain. This means that even when a transaction is occurring on expensive chains like Ethereum, granular fees can be collected at near zero cost.

The way it works is that fees are paid out of the relayer's escrowed balance. For example, consider a scenario where the user is bridging 1 ETH, the relayer is charging 0.001, and other actors charging 0.002:

1. User pays 1.003 to relayer on origin
2. Relayer pays 1 to user on destination
3. 0.002 in fees taken out of relayer balance on settlement chain

In effect, the ETH that lives on the settlement chain has multiple purposes. It acts as a bond to secure transactions on other chains, but also can be used for low cost payments between different actors. Other use cases for Relay ETH include:

- Users can hold it, and use it as a low-cost cross-chain spending balance
- Unspent gas can be rebated as Relay ETH, removing the need for relayers to factor destination gas fluctuations in their quotes

Conclusion

The key theme of the Relay Protocol is the pursuit of maximum efficiency. This is because exciting new use cases only become possible once cost, latency and friction are reduced to an absolute minimum. Even as blockspace becomes abundant, designing a system that is maximally efficient remains important, because it allows savings to be passed on to users, relayers and apps.

Future Work

This overview was designed to give a high level summary, and did not go into technical detail or discuss how adversarial

scenarios are handled. These will be addressed in a future whitepaper.

Additionally, this overview focused on instant bridging and cross-chain execution, which are the first use-cases being productized. However, Relay is effectively a generalized protocol for executing “non-atomic swaps”. While that makes it particularly well-suited to cross-chain actions, it can also have interesting applications for same-chain actions, which will be expanded upon at a later date. Deep Linking The Reservoir Relayer twitter Powered by Mintlify