Team Information

- Team Name: The Interoperable Foundation
- Team Members:
- Jinsuk (@jinsuk)
- Sarah (@Shana)
- Tedi (@tedinho)
- Jinsuk (@jinsuk)
- Sarah (@Shana)
- Tedi (@tedinho)

Project Repository

- GitHub Link:
- op-intent service: GitHub - flashbots/optimism at feat/interop_builder
- op-geth with preconfirmation: GitHub - avalonche/op-geth at interop-preconf
- optimism-package: GitHub - ethpandaops/optimism-package: A kurtosis package for deploying an optimistic rollup
- op-intent service: GitHub - flashbots/optimism at feat/interop_builder
- op-geth with preconfirmation: GitHub - avalonche/op-geth at interop-preconf
- optimism-package: GitHub - ethpandaops/optimism-package: A kurtosis package for deploying an optimistic rollup

Project Goal

- Project Name: Optimism Interop Builder
- Brief Description:
- What is the primary goal of your project?
- Ensure synchronous inclusion of transactions across more than two rollups
- Design an RPC method and data structure to express user intent at the op-intent backend
- Block building according to the interop spec
- Wanted to make an atonic transaction possible across multiple OP rollups. Developed coordinator services across four different sequencers.
- Coordinator sends a transactions to the preconfirmation RPC endpoint which returns the log index to be parsed and used for the executing message
- Accepts a graph bundle for sequential execution across multiple rollups
- Accepts a graph bundle for sequential execution across multiple rollups
- Ensure synchronous inclusion of transactions across more than two rollups
- Design an RPC method and data structure to express user intent at the op-intent backend
- Block building according to the interop spec
- Wanted to make an atonic transaction possible across multiple OP rollups. Developed coordinator services across four different sequencers.
- Coordinator sends a transactions to the preconfirmation RPC endpoint which returns the log index to be parsed and used for the executing message
- Accepts a graph bundle for sequential execution across multiple rollups
- Accepts a graph bundle for sequential execution across multiple rollups

- What are the non-goals?
- Atomic inclusion of interop messages.
- Handling safety of interop messages. We assumes that the rollups fully trust each other.
- Considering reorg scenario. We work with the happy path only
- Account abstraction or EIP 3074 AUTH/AUTHCALL enabled contract to facilitate signing of messages in the DAG
- Atomic inclusion of interop messages.
- Handling safety of interop messages. We assumes that the rollups fully trust each other.
- Considering reorg scenario. We work with the happy path only
- Account abstraction or EIP 3074 AUTH/AUTHCALL enabled contract to facilitate signing of messages in the DAG
- What is the primary goal of your project?
- Ensure synchronous inclusion of transactions across more than two rollups
- Design an RPC method and data structure to express user intent at the op-intent backend
- Block building according to the interop spec
- Wanted to make an atonic transaction possible across multiple OP rollups. Developed coordinator services across four different sequencers.
- Coordinator sends a transactions to the preconfirmation RPC endpoint which returns the log index to be parsed and used for the executing message
- Accepts a graph bundle for sequential execution across multiple rollups
- Accepts a graph bundle for sequential execution across multiple rollups
- Ensure synchronous inclusion of transactions across more than two rollups
- Design an RPC method and data structure to express user intent at the op-intent backend
- Block building according to the interop spec
- Wanted to make an atonic transaction possible across multiple OP rollups. Developed coordinator services across four different sequencers.
- Coordinator sends a transactions to the preconfirmation RPC endpoint which returns the log index to be parsed and used for the executing message
- Accepts a graph bundle for sequential execution across multiple rollups
- Accepts a graph bundle for sequential execution across multiple rollups
- What are the non-goals?
- Atomic inclusion of interop messages.
- Handling safety of interop messages. We assumes that the rollups fully trust each other.
- Considering reorg scenario. We work with the happy path only
- Account abstraction or EIP 3074 AUTH/AUTHCALL enabled contract to facilitate signing of messages in the DAG
- Atomic inclusion of interop messages.
- Handling safety of interop messages. We assumes that the rollups fully trust each other.
- Considering reorg scenario. We work with the happy path only
- Account abstraction or EIP 3074 AUTH/AUTHCALL enabled contract to facilitate signing of messages in the DAG

Challenges

- Challenges Faced:

- Biggest pain point was that you need a log index from the initiating message to put into the executing message
- If the destination block has earlier timestamp than the source block, synchronous inclusion is impossible due to timestamp invariant
- Adding chains to dependency set was not trivial because a transaction to addDepndency

method in SystemConfig contract had to be proxied via a Safe contract.

- Biggest pain point was that you need a log index from the initiating message to put into the executing message
- If the destination block has earlier timestamp than the source block, synchronous inclusion is impossible due to timestamp invariant
- Adding chains to dependency set was not trivial because a transaction to addDepndency

method in SystemConfig contract had to be proxied via a Safe contract.

Current State

- Project Status at the End of Hackathon:
- What features are complete?
- op-intent:

multi chain intent coordinator service * eth_sendGraphBundle

method to include multi chain bundle encoded in a DAG format.

- eth_sendGraphBundle

method to include multi chain bundle encoded in a DAG format.

- op-intent:

multi chain intent coordinator service * eth_sendGraphBundle

method to include multi chain bundle encoded in a DAG format.

- eth_sendGraphBundle

method to include multi chain bundle encoded in a DAG format.

- What features are complete?
- op-intent:

multi chain intent coordinator service * eth_sendGraphBundle

method to include multi chain bundle encoded in a DAG format.

- eth_sendGraphBundle

method to include multi chain bundle encoded in a DAG format.

- op-intent:

multi chain intent coordinator service * eth_sendGraphBundle

method to include multi chain bundle encoded in a DAG format.

- eth_sendGraphBundle

method to include multi chain bundle encoded in a DAG format.

[

image

2412×906 208 KB

](https://collective.flashbots.net/uploads/default/original/2X/e/e40f22c76e4b97472932fa1b26bfaacae83a68e1.png)

type Message struct { ChainId *big.Int* *json:"chainId"* *Data hexutil.Bytes* *json:"data"* *Target common.Address* *json:"target"* *Value*

big.Int json:"value" }

```
type GraphBundle struct {
    Root  Message       `json:"root"`
    Nodes []GraphBundle `json:"nodes"`
}
```

- op-geth with preconfirmation
- eth_sendInteropBundle

RPC method * inserts bundles two blocks ahead

- can be replaced by a bundle with higher gas fees
- returns the logs emitted by transaction.
- inserts bundles two blocks ahead
- can be replaced by a bundle with higher gas fees
- returns the logs emitted by transaction.
- eth_sendInteropBundle

RPC method * inserts bundles two blocks ahead

- can be replaced by a bundle with higher gas fees
- returns the logs emitted by transaction.
- inserts bundles two blocks ahead
- can be replaced by a bundle with higher gas fees
- returns the logs emitted by transaction.
- What is still a work in progress?
- Canceling the preconfirmed bundles
- Canceling the preconfirmed bundles
- Are there any known bugs or issues?
- The synchronous transaction is impossible when destination block’s timestamp is lower than source block’s timestamp. If op labs want to pursue this path seriously, they need to rethink their spec on the timestamp invariant
- The synchronous transaction is impossible when destination block’s timestamp is lower than source block’s timestamp. If op labs want to pursue this path seriously, they need to rethink their spec on the timestamp invariant

How to Run

- Prerequisites (e.g., software versions, dependencies)
- kurtosis with optimism package
- GitHub - tedim52/optimism-package: A kurtosis package for deploying an optimistic rollup
- GitHub - tedim52/optimism-package: A kurtosis package for deploying an optimistic rollup
- kurtosis with optimism package
- GitHub - tedim52/optimism-package: A kurtosis package for deploying an optimistic rollup
- GitHub - tedim52/optimism-package: A kurtosis package for deploying an optimistic rollup
- Step-by-step installation and setup guide
- build docker image for op-geth with preconfirmation: GitHub - avalonche/op-geth at interop-preconf
- run kurtosis run github.com/jinmel/optimism-package --args-file network-params.yml --enclave interop

network-params.yml

l1: participants: - el_type: geth network_params: preset: minimal l2s: - name: "rollup_two" participants: - el_type: op-geth el_image: jinmel/op-geth:preconf additional_services: - blockscout network_params: preset: minimal network_id: "22" seconds_per_slot: 6 - name: "rollup_one" participants: - el_type: op-geth el_image: jinmel/op-geth:preconf network_params: preset: minimal network_id: "21" seconds_per_slot: 6 additional_services: - blockscout

- clone GitHub - flashbots/optimism at feat/interop_builder
- check the rpc address of each rollups by kurtosis enclave inspect interop
- cd optimism && cd op-intent
- make
- ./bin/op-intent --rpc.addr 0.0.0.0 --rpc.port 8545 --l2-eth-rpc 21=${RPC1} --l2-eth-rpc 22=${RPC2}
- check the rpc address of each rollups by kurtosis enclave inspect interop
- cd optimism && cd op-intent
- make
- ./bin/op-intent --rpc.addr 0.0.0.0 --rpc.port 8545 --l2-eth-rpc 21=${RPC1} --l2-eth-rpc 22=${RPC2}
- clone GitHub - jinmel/crucible: crucible nyc hackathon
- run.sh
- python main.py --abi ./contracts/out/A.sol/UniswapInterop.json
- run.sh
- python main.py --abi ./contracts/out/A.sol/UniswapInterop.json
- build docker image for op-geth with preconfirmation: GitHub - avalonche/op-geth at interop-preconf
- run kurtosis run github.com/jinmel/optimism-package --args-file network-params.yml --enclave interop
- clone GitHub - flashbots/optimism at feat/interop_builder
- check the rpc address of each rollups by kurtosis enclave inspect interop
- cd optimism && cd op-intent
- make
- ./bin/op-intent --rpc.addr 0.0.0.0 --rpc.port 8545 --l2-eth-rpc 21=${RPC1} --l2-eth-rpc 22=${RPC2}
- check the rpc address of each rollups by kurtosis enclave inspect interop
- cd optimism && cd op-intent
- make
- ./bin/op-intent --rpc.addr 0.0.0.0 --rpc.port 8545 --l2-eth-rpc 21=${RPC1} --l2-eth-rpc 22=${RPC2}
- clone GitHub - jinmel/crucible: crucible nyc hackathon
- run.sh
- python main.py --abi ./contracts/out/A.sol/UniswapInterop.json
- run.sh
- python main.py --abi ./contracts/out/A.sol/UniswapInterop.json

Additional Notes

- Future plans for the project
- Atomic inclusion of transactions

- Handling large DAG (graph with more than 20 nodes across 4 rollups) inclusion by op-intent
- Atomic inclusion of transactions
- Handling large DAG (graph with more than 20 nodes across 4 rollups) inclusion by op-intent
- Acknowledgments
- kurtosis team for providing us a easy way to launch multiple l2 devnets on our local machine
- kurtosis team for providing us a easy way to launch multiple l2 devnets on our local machine

Questions:

- You are using the OP interop spec?
- Is it atomic (i.e. all or nothing) or is it synchronous?
- technically synchronous
- technically synchronous