Benchmarking Blockchain Bridge Aggregators

Shankar Subramanian* André Augusto† Rafael Belchior†‡ André Vasconcelos† Miguel Correia†
*University of Massachusetts, Amherst †INESC-ID, Instituto Superior Técnico, Universidade de Lisboa ‡Blockdaemon

***Abstract*—Blockchain aggregators play an instrumental role in the evolution of blockchain technology, serving as pivotal enablers of interoperability, efficiency, and user accessibility in an increasingly decentralized digital world. However, the literature on this emerging technology is scarce and is not systematized, making it harder for practitioners and researchers to understand the field. In this paper, we systematize bridge aggregators, a type of blockchain aggregators. We present an exhaustive analysis of a diverse array of token and message aggregators, each distinguished by its unique architecture. Our research delves into critical aspects of these aggregators, encompassing their functionality, security measures, pricing models, and latency. This research aims to provide readers, users, and developers with insightful and actionable information, facilitating informed navigation through the complex landscape of blockchain aggregators. We explore our findings and compare them with our intuitive expectations. We show that there is a value in centralizing token aggregators. Message aggregators are found to be more powerful but less efficient in transaction cost and latency. Finally, we propose a set of future research directions for practitioners.**

***Index Terms*—Blockchain, DLT, Cross-chain bridge, Interoperability, Aggregators**

I. Introduction

Efforts to address blockchain interoperability [1], [2], such as cross-chain bridges, Decentralized Exchanges (DEXes) [3], [4], off-chain API-based protocols, and on-chain oracle services [5], [6], fall short in providing an integrated experience to users, as there is a lack of abstractions that unify disparate protocols and blockchains. Decentralized Finance (DeFi) is currently a highly competitive field, with token values fluctuating in seconds, so users trying to maximize their trade values-aggregators need to move fast. Otherwise, they may get undercut by competitors or exploited by Miner Extractable Value (MEV) bots [7], [8]. With the wide variety of DeFi protocols deployed in a substantial amount of blockchains, user resources get fragmented across these platforms. The unification of fragmented resources is nothing new. The early stages of the internet had a similar problem with e-commerce. Internet aggregators, such as Amazon, Booking, and Uber, consolidated these scattered resources, thereby enhancing user experience and becoming indispensable in everyday life. There are even aggregators that operate at the protocol level, e.g., the Dynamic Host Configuration Protocol (DHCP) [9].

Recognizing the significance of such aggregators in streamlining and enhancing user and developer experiences, this paper analyzes similar aggregation structures within the blockchain ecosystem used as an approach to the interoperability problem. This article focuses specifically on bridge aggregators, exploring their potential to revolutionize the blockchain landscape [1]. Aggregators can substantially augment the functionality and accessibility of blockchain technologies for both seasoned and novice users. A vital attribute of an aggregator is its capacity to conduct routing analyses on behalf of the user, thereby optimizing multi-protocol or chain transactions based on various parameters such as gas costs, trade values, security, and transaction speed.

A. Problem

The contemporary landscape of blockchain technology is marked by significant fragmentation and a lack of comprehensive abstraction layers. Users must individually establish connections and networks, adding to the already extensive list of prerequisites for effective blockchain interaction, not to mention the necessity to abide by local laws and regulations [10]. Aggregators present a viable solution to the interoperability challenge by unifying fragmented resources, thereby improving user experience and facilitating smoother blockchain interactions. Aggregators improve liveness by way of redundancy through fragmented resources.

However, the practical and theoretical analysis of cross-chain aggregators is a cumbersome task for developers, and even more so for users of such protocols, given the multitude of protocols that require thorough and strategic examination. This complexity diverts user resources from other critical activities, such as conducting security reviews, developing new features, or expediting product launches.

B. Contributions

The primary objective of this paper is to empirically analyze bridge aggregator archetypes. As bridge aggregators are in their infancy [11], there is a lack of literature on the subject, but also several aggregator archetypes to investigate.

Our study dissects the various elements that compose an aggregator and evaluates them. We examine the diverse architectures and design choices, highlighting the different trade-offs. This research provides valuable insights to prospective users who intend to utilize these protocols and future researchers and developers who will design and implement cross-chain solutions. We consider different parameters such as system architecture, supported features, openness, decentralization, latency for completion, total cost, fallback model, and ease of use. We also suggest use cases for each type of aggregator. In addition to this theoretical analysis, we rigorously benchmark token aggregators, making our artifacts open-source. The data and code are available on GitHub, accessible at the following URL: https://github.com/hyperledger-labs/benchmarking-cross-chain-bridges/tree/main.

II. Overview of Aggregators

We define an aggregator as a mechanism that allows users to interact with one or more services by abstracting the individual protocol handlers. We classify aggregators into three types, in which each one allows:

1) users to interact with different blockchains by reading data and issuing transactions.
2) smart contracts to interact with smart contracts on other blockchains.
3) for multi-protocol DeFi.

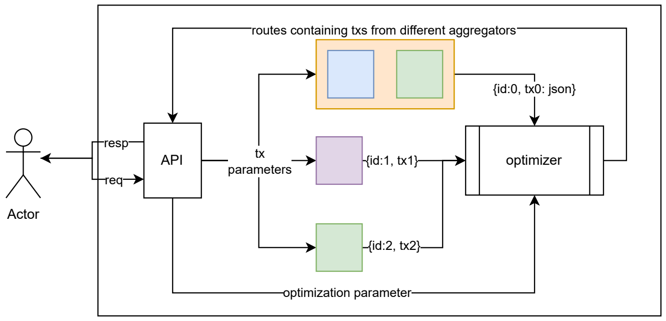


Fig. 1: Aggregator Black Box

The interaction flow of an aggregator is depicted in Figure 1. Different protocols are represented by pink, green, and blue rectangles, and different blockchains by the orange rectangle. Any protocol that provides a similar service can be plugged into this architecture, even if hosted on different blockchains.

The aggregator feeds the transaction parameters into the protocols. The routes generated by these protocols are then ordered based on the optimization parameters (transaction speed, token value, or security) and returned to the user. The blockchain aggregators involved with on-chain state change are called Bridge Aggregators and are the focus of this paper, as we dive into their types and functionalities in the following section.

A. System Model and Actors

We classify bridge aggregators based on their architectures, composed of their user interaction model and the open-source nature of the code base. An aggregator generally lies within a spectrum between open-source and user-run to proprietary and API-based.

Bridge aggregators can function either as Token or Message Aggregators. Token aggregators perform token swapping and bridging across protocols (e.g., 0x), DEXes (e.g., Uniswap), and bridges (e.g., Jumper). These aggregators tend to be to some extent closed-source, to protect their proprietary routing algorithms. On the other hand, message aggregators allow for arbitrary cross-chain transactions to be sent across chains and are more functional due to their generalized message-passing capabilities.

B. Bridge Aggregator Architecture

We present a generic architecture of a bridge aggregator in Figure 2. It contains components from the API Based Aggregators (in blue), Open Sourced Model Aggregators (in red), and the Cross-Chain Protocol Agents (in yellow). This section provides an overview of an aggregator transaction and an in-depth explanation of its components, starting from the top-left of the figure:

User: Takes care of user interaction with the aggregator. SDKs translate user input into an aggregator-specific language and query the aggregator for routes.

Routing API: Decomposes user input into parameters and feeds them to an algorithm to generate routes.

Source Chains 1 & 2: Contains smart contracts with which the user interacts. There are router contracts that emit events about the transaction state for cross-chain protocol agents.

Cross-chain Protocol Agents (Phase 1): In the first phase, agents (also called relayers) listen to events emitted by the router contract on the source chain. If there exist multiple relayers, these run a consensus algorithm to decide on the validity of the transaction. Relayers are informed of an incomplete source transaction that is yet to have a transaction on the destination chain. After consensus, the transaction is finalized and distributed to the blockchain nodes' mempool. Transaction builders create the destination transactions along with validity proofs and submit them to the destination chain.

Cross-chain Protocol Agents (Phase 2): In the second phase, the transaction on the destination chain is executed on the router contract. Similarly to Phase 1, relayers listen for events on the destination network, and update the global state to reflect the transaction data.

Destination/Target Chain: The transaction on the destination chain interacts with the target address and events are logged.

User run Cross-chain Protocol: Here, the routing software is open-sourced and run by the user. The user acts as the validators and relayers, and issue transactions on both networks.

C. Parts of an Aggregator Transaction

We separate the parts involved in an aggregator transaction into components and parameters. Components are the data generated by aggregators and users, such as routes and queries. Parameters are factors that influence generated data.

1) Components: Components are non-standard data generated by aggregators, users, and the benchmarking tool:

1) *User Quote Query:* Contains user-generated transaction parameters such as tokens, value, chains, and payload.
2) *Aggregator Quote:* The quote object returned by an aggregator which contains the aggregator route.
3) *Aggregator Route:* Contains the transaction path and information about the protocols.
4) *APIReport:* The standardized report generated by the benchmarker. It contains information on the networks involved, the time of creation, the source, and destination tokens, the aggregator used, the protocol the aggregator uses, and the value of tokens traded.

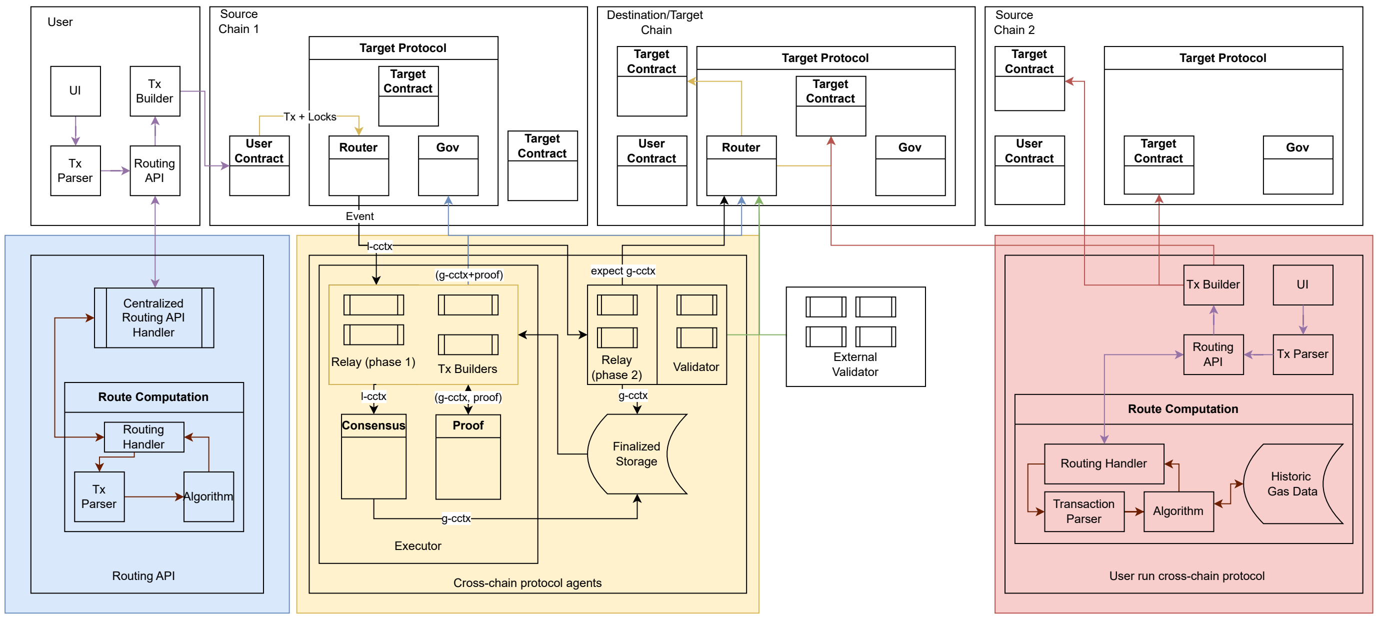


Fig. 2: Generic architecture of a bridge aggregator

2) Parameters: Parameters are factors that influence the components. In this section, we explain what they are:

- *Fees:* Aggregators may charge a fee in addition to the protocol fee, such as the withholder and bridging.
- *Latency:* There are two primary latencies and the aggregator API latencies.
- *User Interface:* Protocols may have a UI. Using such a UI can incur a fee, such as Uniswap [12]. Protocols may also have a transaction explorer.
- *Customizability:* There may be additional services provided by protocols that allow for the customization of transactions.

D. Proprietary Backend Model – API Based Aggregators

These aggregators offer their services through APIs. They tend to have proprietary routing algorithms, and users may have to create an account to use them. The functionality tends to be limited due to a lack of generalizability. They almost always perform cross-chain transactions.

E. Open Source Model

The users have all the components required to perform a transaction by themselves. As the source code is open-source, organizations can modify the code tailored to their requirements. These protocols tend to be single-chain. The routing algorithm relies on mathematical models and network statistics, as there is no centralized source for processed data. This architecture does not have a router contract or consensus layers for a single client, however, they can be extended to work for n-client architectures.

F. Cross-chain Protocol Agents

Allow for cross-chain messaging by use of validators and relayers. Aggregators may run solo relayers, for more efficiency, but reduced decentralization. If there are several relayers, there is a consensus phase that involves a mempool.

III. Instantiations of Aggregators

In this section, we classify a set of aggregators based on their archetypes and look at a few aggregators of each type. We analyzed the most up to date version of each protocol at time of writing this paper, in December 2023. The architectures that we came up with are: Centralized Token Bridge Aggregators; Decentralized Liquidity Aggregators; On-chain Pool Based Token Swap Aggregators, and Message Aggregators. In-depth expositions into the aggregators can be found in the extended version of our paper [13].

A. Centralized Token Bridges:

These aggregators do not execute trades. They compute the routes that redirect users to trade on a protocol. They only support mainnet trades. Aggregators classified in this category:

1) LiFi Jumper v1 – Multichain DeFi Protocol Aggregator: Jumper supports bridge, swap, and execution of cross-chain contract calls [14]. The routing API has a powerful rate limiter and there were several instances of protocol failures. They use multiple protocols.

2) Socket Bungee v2 – Multichain DeFi Protocol Aggregator: In addition to swap and bridge, it provides a service called Refuel that allows users to also get native tokens on the destination network they are swapping to [15]. The only supported protocols are 0x and 1inch. [16]. It has a Data Layer that is pending developer documentation that aims to achieve chain abstraction.

3) XY Finance – Multichain Pool based aggregator: Uses a 2-tier structure comprising XSwap (aggregator) and YPool (liquidity pool) [17]. The pool maintains liquidity thresholds that are shared between all chains. The fee calculation is unclear from the documentation, as it contradicts the results from the API and dApp.

B. Decentralized Liquidity Aggregators:

These aggregators execute trades by clients that provide liquidity to match user trade requests. Users do not submit on-chain transactions when interacting with these aggregators apart from enabling token allowance. This allows users to circumvent MEV bots and send gasless transactions.

1) CoW Swap v1 – Permissionless off-chain protocol that Aggregates Trades on single-chain: CoW Swap uses Batch Auctioning to establish token prices. Transactions are submitted to settlers that complete partial or entire amounts of the trade [18]. Swapping is only supported on the same chain.

2) 0x – Liquidity aggregator between Makers and Takers: 0x tries to match liquidity between those that provide it (Makers) and those that consume it (Takers) [19]. Trades start off-chain where a Maker and Taker arrive at a deal, then submit the signed deal to the 0x smart contract that performs the trade between the on-chain wallets of the Maker and Taker.

C. On-chain Pool Based Token Swap Aggregators:

When interacting with aggregators of this class, users may generate routes locally and then send transactions to the blockchains. This allows for 100% availability if the token pools on-chain have tokens. They are also the quickest to complete, as they only involve a single transaction. They usually support only same-chain trades. Aggregators:

1) Uniswap – Decentralized AMM and off-chain trading protocol: Despite the debate about the open sourcedness of the Uniswap code due to the licensing [20], Uniswap does not contain proprietary code. It offers two aggregative services: AlphaRouter and UniversalRouter. AlphaRouter allows for multiple hops between Uniswap pools to try to find the best route. UniversalRouter also aggregates trades across protocols [21]. While using Uniswap on-chain does not incur a fee, using the Uniswap Trading Application on the website does charge a 0.15% fee [22]. Users can avoid the "UI fee" by interacting with the protocol through an SDK.

2) 1inch – Decentralized AMM and P2P trading: The 1inch aggregator protocols provide users with multi-protocol aggregation and P2P aggregation where users directly trade ERC20 [23]. Users can take part in a cowswap-like auction trading by being resolvers or price aggregators. They have different services that offer user protection from front-run attacks and sandwich attacks.

D. Message Aggregators

Message aggregators aggregate blockchains to send messages across. They come in different flavors – some are block header verifiers, some are decentralized, and some are centralized for the benefit of efficiency. They can execute generic blockchain transactions and are not restricted in functionality. These aggregators support testnets and mainnets.

1) Hyperlane v2 – Modular Interoperability Protocol: Hyperlane is developer-focused, comes with prebuilt modules, and is customizable [24]. The on-chain customizability allows developers to create their own security modules. The protocol fee is collected in native tokens after the destination transaction is sent. Cross-chain transactions are quick due to the centralized backend, with stages of the transaction being constant time [1]. Transactions are created from by users.

2) CCIP – Chainlink's cross-chain messaging protocol: CCIP has a decentralized backend which contributes to slower transactions due to the consideration of block finality on the source chain [25]. Users can pay the fee as Native or Link. The fee is paid when the source transaction is created. Transactions are issued through smart contracts.

3) Hashi v1 – Redundant Array of Independent Oracles: Inspired by RAID, this protocol collects block headers from different sources and validates them on-chain at each destination network [26]. Using this protocol is expensive and slow due to the amount of on-chain computation, the tradeoff gives the highest security among the discussed aggregators. Developers are required to validate transactions within the validated blocks. This protocol is under development and the documentation is not complete [27].

E. Aggregator Summary Framework

Our benchmark framework overviews the Aggregator components II-C1 and parameters II-C2 for each of the aggregators benchmarked III:

- *Type:* The main type of aggregator — token or messaging
- *Design Model:* The user interaction model and the aggregator backend architecture
- *No Setup:* Configuration required to use the aggregator, such as deploying aggregator-specific contracts, sending transactions to join a validator list
- *Block Explorer:* To help users visualize transactions
- *Functionality:* The core component of the architecture that settles the transactions
- *Optimize Transaction On:* The parameters that the user can choose to optimize on
- *Customizability:* The changes to the base configuration such as implementing user-specific rules
- *Open Source:* We consider the open-sourcedness of smart contracts, backend code, and documentation.
- *Target Audience:* The end user.

IV. Benchmarks

In order to analyze bridge aggregators, we designed a benchmarking tool [28] that generates routes for aggregators, implements contracts, executes routes, and executes cross-chain transactions. We break down the analysis of the aggregators into theoretical analysis and practical tests obtained from benchmarking experiments, explained in Section II-C1.

[1]http://tinyurl.com/hyperlane-tx

	LiFi	Socket	XY	CoW	0x	Uniswap	1inch	Hyperlane	CCIP	Hashi
Type	Token Bridge, Token Pool	Token Bridge, Token Pool	Token Bridge, Token Pool	Liquidity Aggregator	Liquidity Aggregator	On-Chain Pool Aggregator	On-Chain Pool Aggregator	Token, Messaging	Token Messaging	Messaging
Design Model	API Relays	API Relays	API Pools	Off-chain Auctions	Off-chain Auctions	On-chain Token Pools	On-chain Token Pools	On-chain Centralized Oracles	On-chain Decentralized Oracles	On-chain 3rd Party Oracles
No Setup	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗
Block Explorer	✓	✓	✗	✓	✓	✗	✗	✓	✓	✗
Core Architecture	Asset Router	Asset Router	Private Settlement Chain	Off-chain Auctions	Off-chain Auctions	Interactable Pools, Auctions	Interactable Pools, Auctions	User transactions initiated	Smart contract initiated	Block Validation Framework
Optimize Transaction On	Speed, Value	UX, Value	Multichain Token, Value	Token Value	Token Value	Token Value	Token Value	Developer Choice	Predefined	Security
Customizability	✗	✗	✗	◑	◑	◑	◑	●	●	●
Open Sourcedness	◑	◑	◑	●	●	●	●	●	●	●
Target Audience	Traders and Bridgers	Traders and Bridgers	Traders and Bridgers	Traders	Traders	Traders	Traders	dApps	Protocol Devs, Bridges	High-Security Protocols, DAOs

TABLE I: Summary of Token Aggregator Frameworks

A. *Setup*

The experiments were executed on a machine with 8 cores@1.9 GHz base clock CPU and 32 GB RAM. The storage used was 29 MB for 360 runs that generated 2250 aggregator-specific quotes and APIReports. The Coingecko pricing for 360 quotes was 1.6 MB. We used network pairs that were supported by most aggregators. For token aggregators, we used Ethereum to Ethereum/Polygon. For message aggregators, we chose Sepolia and Polygon Mumbai, as they all supported cross-chain. RPC URLs were provided by Alchemy. We used the SDK or API provided by each aggregator, and Coingecko[2] pricing API to collect token price data.

B. *Benchmark Process*

We collected results over a two-day window, polling a new route every 20 minutes. This gave us 360 batches of token aggregator quotes. The collection window was from 2023-12-08, 18:33 UTC to 2023-12-11, 14:02 UTC. A single batch ran for about 41 seconds.

Message aggregators were statically benchmarked, where we only logged the reports generated by the scripts on a single run. This was possible due to the metrics generated by the message aggregators, such as the deployment cost, message cost, and gas fee, being dependent on mathematical equations.

To benchmark message aggregators, we created a simple number storage script that could work with an aggregator. At the time of benchmarking, the following were the token prices: 1 DAI / 0.9985 USD; 1 ETH / 2,234.26 USD; 1 MATIC / 0.8556 USD. / Although the codebase is configured to execute trades and monitor the latency, we were unable to execute token aggregator quotes as they did not support cross-chain trades on testnets. We believe that testing latencies is more a DEX benchmark, and does not truly isolate the aggregator which is what we are trying to evaluate.

C. *Hypotheses*

Before benchmarking, the following were our hypotheses:

[2]https://www.coingecko.com/api

1) API-based aggregators have tighter bounds on the relation between aggregator quote value and the actual value of a token, quoted by coin gecko.
2) Open source aggregators have higher variance in the token pricing and more latency.
3) The fee involved is a function of the gas price on the destination network.
4) Gas estimates to be a function of the network they are executing on.
5) DEXs are more secure than CEXs due to their decentralized nature.

V. Results

In this section, we present the benchmarking results generated by the benchmarking tool in the experiments.

A. *Fees and Gas Price:*

We present aggregator net fees collected over constant parameters, including network conditions. Figure 3 depicts net fee charged (in blue), against the source and destination chain gas prices (in grey). We also present Table II with the measurements of the mean and variance of the net fee. For message aggregators we list the operations performed, the gas price of the network, the gas used, and the amount in USD to create a standard comparison across aggregators. We did the above for both versions of CCIP in Table III, Hyperlane in Table IV, and Hashi in Table V.

Takeaway 1: XY and Socket do not charge a fee (Table II).

Takeaway 2: Ignoring gas fee when computing net fee, only the destination chain gas price affects the net fee. (cf. Figure 3 - (b), (c)) show the blue lines (aggregator fee) tightly following the destination network gas price.

Takeaway 3: When accounting for the gas fee, the source chain also has an impact, although minor (cf. Figure 3 - (a), (b), (c)) show the aggregator fee move in a similar pattern to the source chain gas price.

Takeaway 4: Cross-chain transactions incur more fee than same chain transactions (see Table II).

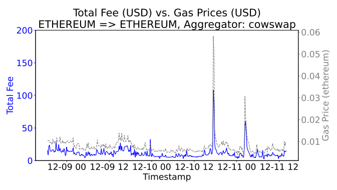
Takeaway 5: Table II shows that Open Source Protocols (CoW Swap and Uniswap) have the best variance-mean ratio.

Aggregator	Source Chain	Dest Chain	Net Fee (σ)	Net Fee (μ)
CoW Swap	Ethereum	Ethereum	8.88	11.30
LiFi	Ethereum	Ethereum	9.85	28.80
LiFi	Ethereum	Polygon	15.69	40.90
Socket	Ethereum	Ethereum	7.03	12.61
Socket	Ethereum	Polygon	6.08	11.32
Uniswap	Ethereum	Ethereum	0.98	1.38
XY	Ethereum	Ethereum	0.00	0.09
XY	Ethereum	Polygon	0.60	1.63

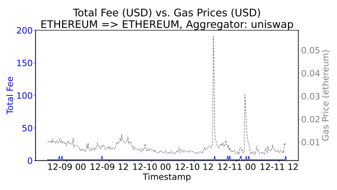
TABLE II: Average net fees for each aggregator, source-chain, and dest-chain combination.

Operation	Network	Gas Price	Tx Gas	Amount
Message Fee (ETH)	Sepolia	126.512107387	29081	8.220
Message Fee (LINK)	Sepolia	126.512107387	47732	13.491
Deploy Sender	Sepolia	126.512107387	864548	244.373
Deploy Counter	Mumbai	3.000000032	498019	0.001
Send Txs (ETH)	Sepolia	126.533233854	230984	65.301
Send Txs (LINK)	Sepolia	126.533233854	237100	67.030

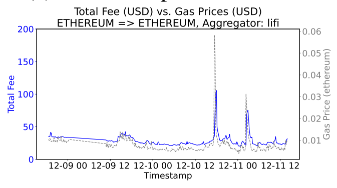
TABLE III: CCIP v1.2.0 Cost to Use. Amounts are in USD and Gas Prices are in gwei.



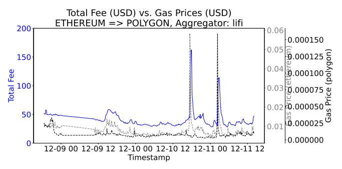
(a) CoW Swap - Same-chain



(b) Uniswap - Same-chain



(c) LiFi - Same-chain



(d) LiFi - Cross-chain

Fig. 3: Net Fee

Takeaway 6: Chainlink CCIP and Hyperlane use transparent and deterministic protocol fee computation. The data used in the computation is updated by a selected committee [3] [29].

Takeaway 7: Hashi sources block headers from several independent oracles [30].

[3]http://tinyurl.com/ccip-evm2evmonramp

Operation	Network	Gas Price	Tx Gas	Amount
Deploy Counter Tx	Mumbai	3.000000032	222699	0.001
Send Tx to Counter	Goerli	3.000000206	74685	0.501
Send Tx to Counter	Sepolia	113.57621251	67081	17.022
Interchain Gas Paymaster Fee [1]	Sepolia	113.57621251	224289	56.913

[1] Retrieved Tx Gas from a ETHEREUM to POLYGON transaction on the Hyperlane Explorer due to testnet transactions not having fees

TABLE IV: Hyperlane v2 Cost to Use. Amounts are in USD and Gas Prices are in gwei.

Operation	Network	Gas Price	Tx Gas	Amount
Deploy Yaho	Goerli	3.000000804	1004278	6.731
Deploy Yaru	Gnosis	16.52392343	953260	0.016
Deploy AMBRelay	Goerli	3.00000081	480194	3.218
Deploy AMBAdapater	Gnosis	15.962067614	1082831	0.017
Deploy Counter	Gnosis	15.962067614	212871	0.003

TABLE V: Hashi Cost to Use. Amounts are in USD and Gas Prices are in gwei.

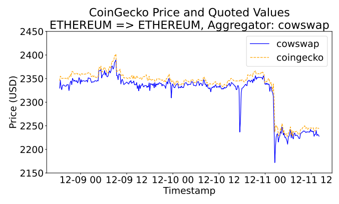
Takeaway 8: Hyperlane does not charge a fee on testnets. We used an identical mainnet transaction to get the fee charged. (cf. Table IV).

Takeaway 9: Contract deployment is a large part of the cost in CCIP and Hyperlane. CCIP requires a sender contract that Hyperlane does not (cf. Table III and Table IV).

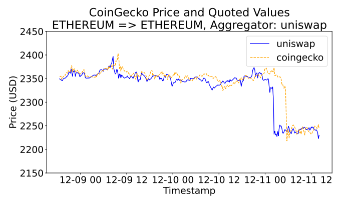
Takeaway 10: Sending CCIP transactions with native tokens is cheaper than with LINK (cf. Table III).

Takeaway 11: CCIP estimates computation and fees on-chain while Hyperlane does not [29].

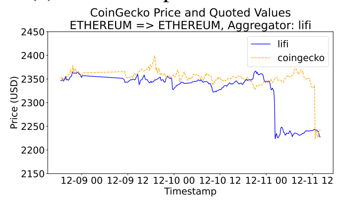
B. Aggregator Quote vs. Coingecko



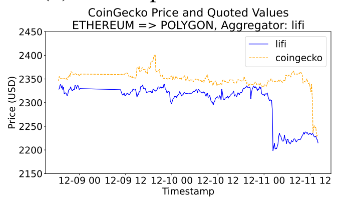
(a) CoW Swap - Same-chain



(b) Uniswap - Same-chain



(c) LiFi - Same-chain



(d) LiFi - Cross-chain

Fig. 4: Aggregator Quote vs Coingecko Quote

We use Coingecko API as a common source to compare all the aggregator quotes, due to its reputation within the community. Not needing the creation of accounts or API keys was a key decision point. The aggregators were instructed to maximize trade value. Figure 4 shows the movement of the aggregator quote price (in blue) against the Coingecko price (in yellow). In Table VI, we compare the deviation between the aggregators and Coingecko's values.

Takeaway 12: Table VI shows that API-based aggregator quotes tend to match Coingecko Quotes and have higher variance.

Takeaway 13: There is more variance in the quotes when the aggregator quotes under Coingecko than when they quote over Coingecko pricing Table VI.

Takeaway 14: Table VI shows that API aggregators may not have 100% availability and can go down, or not produce a route when there exists a route.

Name	Source Chain	Destination Chain	Over (μ)	Over (σ)	Under (μ)	Under (σ)	Diff (μ)	Diff (σ)	Count
CoW Swap	Ethereum	Ethereum	0.00	0.00	11.35	9.20	-11.35	9.20	360
LiFi	Ethereum	Ethereum	13.53	11.93	35.20	43.28	-29.59	43.77	304
LiFi	Ethereum	Polygon	0.00	0.00	48.11	39.94	-48.11	39.94	305
Socket	Ethereum	Ethereum	5.63	4.20	7.13	11.05	-1.16	10.66	359
Socket	Ethereum	Polygon	6.25	5.01	7.86	12.04	-0.81	11.61	358
Uniswap	Ethereum	Ethereum	6.88	5.65	16.71	27.57	-9.45	25.58	341
XY	Ethereum	Ethereum	5.97	4.23	6.34	8.61	1.36	8.63	360
XY	Ethereum	Polygon	7.02	4.85	7.31	12.21	2.49	10.35	359

TABLE VI: Price differences of Aggregator Quote (USD) vs Coingecko (USD)

Takeaway 15: Figure 4 shows that CoW Swap was unaffected by the "12-11 00" price drop across all the API aggregators and Uniswap.

C. Latency

We measure the mean and standard deviation of the latency of an aggregator quote.

Takeaway 16: API latency of cross-chain transactions is about 2x to 10x longer than same chain trades (cf. Table VII – LiFi, Socket, and LiFi).

Takeaway 17: Variance in cross-chain transactions is about 2x that of the same chain (cf. Table VII — LiFi, and Socket).

Takeaway 18: API aggregators tend to have lower variances in latency than the Open Source Model Aggregators (cf. Table VII – CoW Swap and Uniswap).

Takeaway 19: We order each aggregator latency-wise from the fastest to slowest: API-based aggregators (same-chain) < CoW Swap < Uniswap < API based aggregators (cross-chain) (cf. Table VII).

VI. Discussion

A. Token aggregators

Our hypotheses (1,2, and 5) presented in Section IV-C are based on the centralization gains experienced by API Aggregators due to their architectures. Hypothesis (3,4) were based on the involvement of aggregators in cross-chain messages by spending their own tokens on gas fees. We expected most aggregators to pick the same protocol, the one that provides maximal trade value, which was not the case. The trade value generally moves with Coingecko's USD quote of ETH (Figure 4), with a bias of quoting lower. Cross-chain transactions use gas paid by the protocol, which explains Takeaways 2 and 3.

Open-source aggregators simply optimize and match the given inputs. The generated trade values can slip, as seen in Table VI. However, Cow Swap provided the best trade values across all aggregators.

Takeaway 15 shows that aggregators have recovery strategies to correct errors when protocols produce incorrect quotes. It also appears that many protocols were linked to the pricing quoted by Uniswap, inferrable from the Uniswap slippage impacting quotes from all aggregators except CoW Swap. As aggregator quotes vary drastically from Coingecko quotes, we recommend having different thresholds to prevent being affected by slippage when using automated trading.

B. Message Aggregators

From Takeaway 5, deterministic fee computation is strongly influenced by the message length and networks involved[4]. Additionally, Takeaway 9 shows that deploying Contracts on the networks seems to influence the cost to use a protocol the most, with CCIP requiring a sender and receiver contract, while Hyperlane and Hashi only require one. This is due to the difference in architectures where CCIP can be automated on-chain to send cross-chain messages and is more of a protocol than Hyperlane which is an application.

C. Security

Aggregators allow for more attack vectors than standalone protocols due to infinite allowances, change in msg.sender, etc.

Token Allowance exploits are the most common attacks. Consider the following scenario – an aggregator generates a route on protocol P with token allowance value greater than the trade value to prevent future allowance updating transactions, a common practice to save gas. As aggregators often provide users with unsigned transactions to approve allowance, users tend to execute them and not know the value being allowed, which could be infinite. Now, when P or any aggregator that contains a route to interacting with P is hacked, the attacker can drain that wallet. This is what happened with Socket [31]. As Socket was centralized and pausable, the developers quickly paused trading on the socket contract. The attack was from a prototype and smart contract deployed on mainnet. Lifi Jumper also faced an exploit due to an unchecked external call that led to an approval exploit [32].

Additionally, protocols should no longer rely on `msg.sender` as a proof of validity because the protocol relayer and the routing contracts are consistent across all destination chain transactions. This means that should a user approve a protocol router, then anyone can send an arbitrary cross-chain transaction to that destination chain executing a trade from the spender to the hacker. This would always pass as the `msg.sender` would be either the approved protocol relayer or routing contract, while the `tx.origin` would be the relayer.

In protocols that support arbitrary function calls (i.e. when there is no required receive function enforced by the protocol), `msg.sender` assertions on protocol relayers need to be

[4]http://tinyurl.com/ccip-priceregistry

Name	Source Chain	Destination Chain	Over (μ)	Over (σ)	Under (μ)	Under (σ)	Latency (μ)	Latency (σ)
CoW Swap	Ethereum	Ethereum	0.00	0.00	2568.08	1160.18	2568.08	1160.18
LiFi	Ethereum	Ethereum	1094.74	251.72	1118.77	342.22	1116.00	333.14
LiFi	Ethereum	Polygon	0.00	0.00	2793.89	623.90	2793.89	623.90
Socket	Ethereum	Ethereum	532.46	383.97	499.43	363.57	514.89	373.62
Socket	Ethereum	Polygon	6774.20	2398.20	6716.69	1833.03	6745.45	2134.59
Uniswap	Ethereum	Ethereum	4247.02	1309.14	4327.64	934.18	4302.82	1064.46
XY	Ethereum	Ethereum	2837.70	711.98	2789.60	650.32	2819.66	689.90
XY	Ethereum	Polygon	5698.48	1727.13	5710.53	1978.43	5698.14	1809.40

TABLE VII: Latency vs Quote Difference

avoided. Otherwise, attackers can call any function on a contract, which gets executed as the protocol relayer.

Phishing attacks are a problem with decentralized exchanges like the one on Uniswap [33]. Cowswap got hit with a solver exploit, but as they do not use approvals, user funds were not in danger [34].

D. Summary

Aggregators may not select the most optimal protocol to trade on. API aggregators do not have 100% availability as seen from the generated report count, and they may use the same protocol causing another centralization issue on data source. API-based aggregators have recovery strategies to counter these vulnerabilities. Most aggregators quote a value lower than Coingecko's quote and the usual range is within +/-10%. The same protocol on different networks may have different token pricing. Latencies follow the expectations across different networks and aggregators, with different aggregators using different security defenses such as block finality on the source network.

While neither major archetype is free from exploits, it can be safe to say that decentralized ones are more secure due to shorter and independent transaction chains as compared to the ones at centralized archetypes where in addition to single point failures, there are longer transaction chains with more components allowing for increased number of attack points.

VII. Future Research Directions

This paper provides the first comprehensive benchmark on bridge aggregators. We compile a list of promising future research directions:

- Extend the benchmarking framework to more aggregators and parameters, helping practitioners choose the best aggregator for their needs in the industry and academia. Similarly to [35], a framework for selecting an aggregator should be provided. We should thus test our hypothesis on production chains (mainnets).
- Research on security and privacy in cross-chain technology is still scarce, but recent developments are promising [10]. Exploring privacy mechanisms for bridge aggregators and performing a comprehensive security analysis of specific solutions using varied techniques such as cross-chain models [36] are needed.
- Enterprises need connectivity to blockchains to satisfy their needs for modernizing their financial services. For this, enterprise-grade infrastructure providers play a pivotal role [37]. However, aggregators are still not ready to be adopted by companies needing to abide by different laws and regulations. Research to understand enterprise needs in terms of organizational and legal interoperability [38] is required to create a well-defined list of functional and non-functional requirements for aggregators.

VIII. Conclusion

Aggregators are here to stay, as part of the ever-evolving mission to make blockchain interoperability more user-friendly. In this study, we performed the first benchmarking study of bridge aggregators, showcasing a holistic analysis of the evolving landscape.

Most of the expectations that came into the project were consistent with the results. API aggregators usually had the lowest latencies, but had more downtime, as expected from centralized services. The decentralized and open-source models had higher latencies, but the lowest downtime. Aggregators also tended to quote values under those of coingecko quotes. This is expected, as the protocols with which they interact have varying degrees of slippage on varied quote values. The open-source aggregators had a lesser variance in their price quotes, which was surprising. Message aggregators tend to be larger code chunks that take longer periods to execute while being more expensive in terms of gas and fees. This is consistent with the expectation of a generalized messaging service compared to that of a highly engineered single-task service like a token aggregator.

The benchmarking results help academics and practitioners alike to systematically reason about aggregators. We provide future research directions, including cross-chain privacy and studies on the feasibility of enterprise-grade aggregators.

Acknowledgments

This work was supported by Hyperledger through the Hyperledger Mentorships program. Rafael was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UIDB/50021/2020 (INESC-ID) and 2020.06837.BD. This work was developed within the scope of the project nr. 51 “BLOCKCHAIN.PT - Agenda Descentralizar Portugal com Blockchain”, financed by European Funds, namely “Recovery and Resilience Plan - Component 5: Agendas Mobilizadoras para a Inovação Empresarial”, included in the NextGenerationEU funding program.

REFERENCES

[1] R. Belchior, A. Vasconcelos, S. Guerreiro, and M. Correia, "A survey on blockchain interoperability: Past, present, and future trends," *ACM Computing Surveys*, vol. 54, no. 8, p. 1–41, May 2021.

[2] R. Belchior, J. Süßenguth, Q. Feng, T. Hardjono, A. Vasconcelos, and M. Correia, "A brief history of blockchain interoperability," *Communications of the ACM*, 2024, accepted for publication.

[3] S. Werner, D. Perez, L. Gudgeon, A. Klages-Mundt, D. Harz, and W. Knottenbelt, "Sok: Decentralized finance (defi)," in *Proceedings of the 4th ACM Conference on Advances in Financial Technologies*, 2022, pp. 30–46.

[4] Y. Chen and C. Bellavitis, "Blockchain disruption and decentralized finance: The rise of decentralized business models," *Journal of Business Venturing Insights*, vol. 13, p. e00151, 2020.

[5] G. Caldarelli and J. Ellul, "The blockchain oracle problem in decentralized finance—a multivocal approach," *Applied Sciences*, vol. 11, no. 16, p. 7572, 2021.

[6] G. Caldarelli, "Before ethereum. the origin and evolution of blockchain oracles," *IEEE Access*, vol. 11, pp. 50 899–50 917, 2023.

[7] K. Qin, L. Zhou, and A. Gervais, "Quantifying blockchain extractable value: How dark is the forest?" in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 198–214.

[8] P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels, "Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 910–927.

[9] R. Droms, "RFC2131: Dynamic host configuration protocol," 1997.

[10] A. Augusto, R. Belchior, M. Correia, A. Vasconcelos, L. Zhang, and T. Hardjono, "Sok: Security and privacy of blockchain interoperability," in *2024 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, may 2024, pp. 234–234. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SP54263.2024.00182

[11] [Online]. Available: https://blog.li.fi/what-are-blockchain-bridges-and-how-can-we-classify-them-560dc6ec05fa

[12] [Online]. Available: https://uniswap.org/

[13] S. Subramanian, A. Augusto, and R. Belchior, *Benchmarking Bridge Aggregators*, Jan. 2024. [Online]. Available: https://www.techrxiv.org/users/687326/articles/697988-benchmarking-bridge-aggregators?commit=27d477ddc8a93be4726ebc8903395ecdc8789054

[14] [Online]. Available: https://docs.li.fi/

[15] [Online]. Available: https://docs.socket.tech/socket-overview/what-is-socket

[16] [Online]. Available: https://github.com/SocketDotTech

[17] [Online]. Available: https://docs.xy.finance/

[18] [Online]. Available: https://docs.cow.fi/

[19] [Online]. Available: https://0x.org/docs/introduction/introduction-to-0x

[20] [Online]. Available: https://github.com/Uniswap/v4-core/pull/259

[21] [Online]. Available: https://docs.uniswap.org/contracts/universal-router/overview

[22] [Online]. Available: https://support.uniswap.org/hc/en-us/articles/20131678274957-What-are-Uniswap-Labs-fees-

[23] [Online]. Available: https://portal.1inch.dev/documentation/swap/quick-start

[24] [Online]. Available: https://github.com/hyperlane-xyz

[25] "Chainlinkccip." [Online]. Available: https://docs.chain.link/ccip/concepts

[26] [Online]. Available: https://hashi-doc.gitbook.io/hashi/v0.1/introduction

[27] [Online]. Available: https://github.com/gnosis/hashi

[28] [Online]. Available: https://github.com/hyperledger-labs/benchmarking-cross-chain-bridges

[29] "Interchain gas payment | hyperlane v3 docs." [Online]. Available: https://v3.hyperlane.xyz/docs/reference/hooks/interchain-gas

[30] "Introduction | gnosis hashi 2023," Dec. 2023. [Online]. Available: https://docs.gnosischain.com/bridges/hashi/

[31] [Online]. Available: https://sockettech.notion.site/Socket-Incident-Report-16-Jan-9aba3bbf08814fc49e4f2ffb58284912

[32] [Online]. Available: https://blog.li.fi/20th-march-the-exploit-e9e1c5c03eb9

[33] [Online]. Available: https://www.coindesk.com/tech/2022/07/12/uniswap-user-loses-8m-worth-of-ether-in-phishing-attack/

[34] [Online]. Available: https://blog.cow.fi/cow-swap-solver-exploit-post-mortem-07-02-2023-2faa9f918e29

[35] R. Belchior, L. Riley, T. Hardjono, A. Vasconcelos, and M. Correia, "Do you need a distributed ledger technology interoperability solution?" *Distrib. Ledger Technol.*, vol. 2, no. 1, mar 2023. [Online]. Available: https://doi.org/10.1145/3564532

[36] R. Belchior, P. Somogyvari, J. Pfannschmidt, A. Vasconcelos, and M. Correia, "Hephaestus: Modeling, analysis, and performance evaluation of cross-chain transactions," *IEEE Transactions on Reliability*, p. 1–15, 2023.

[37] [Online]. Available: https://www.blockdaemon.com/

[38] R. Belchior, J. Süßenguth, Q. Feng, T. Hardjono, A. Vasconcelos, and M. Correia, "A brief history of blockchain interoperability," 09 2023.