LiftChain: A Scalable Multi-Stage NFT Transaction Protocol

Hari Kishore Chaparala
Computer Science
University of California, Irvine
Irvine, USA
hchapara@uci.edu

Sai Vineeth Doddala
Computer Science
University of California, Irvine
Irvine, USA
sdoddala@uci.edu

Ahmad Showail
Computer Engineering
Taibah University
Madinah, Saudi Arabia
ashowail@taibahu.edu.sa

Abhishek Singh
Computer Science
University of California, Irvine
Irvine, USA
abhishas@uci.edu

Samaa Gazzaz
Computer Science
University of California, Santa Cruz
Santa Cruz, USA
sgazzaz@ucsc.edu

Faisal Nawab
Computer Science
University of California, Irvine
Irvine, USA
nawabf@uci.edu

2022 IEEE International Conference on Blockchain (Blockchain) | 978-1-6654-6104-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/BLOCKCHAIN55522.2022.00057

***Abstract*—The Non-fungible token (NFT) market has been booming with some reports estimating the surge in the market value to over 80 Billion USD in 2025. With this surge, comes the challenge of scaling NFT transactions and providing low latency responses to end-users. Exclusive layer-1 blockchains like Bitcoin and Ethereum which focus more on security and traceability are not suitable for high throughput NFT transactions of low value due to high gas fees. Layer-2 solutions can scale but have a trade-off of losing some strong decentralization and security guarantees offered by the Mainnet. In this paper, we propose LiftChain, a multi-stage NFT transaction protocol that can scale for high volume NFT transactions and at the same time inherit the security guarantees from Mainnet. LiftChain differs from existing NFT transaction protocols in that it allows multiple NFT transactions in the off-chain before becoming consistent with the on-chain. LiftChain also uses batching for gas fee savings and better bandwidth utilization. Our evaluations show that LiftChain provides comparable performance to baseline off-chain and with batching we see more than a 5-fold improvement in gas fee savings.**

***Index Terms*—Blockchain, NFT, off-chain, Multi-Stage transactions**

I. Introduction

NFTs have been gaining immense popularity. The NFT market cap is now more than $7 billion [22], the sales have grown over 100 times from 2020 to 2021, and some popular collections have traded a volume worth over $2 billion [21]. With organizations like Meta [19], Microsoft [20], and other video game industries investing in Metaverse in which Web3 can play a major role, this trend could further increase. This popularity surge begs the question of whether we can scale NFT transactions to a point where NFTs can be seamlessly integrated with high user volume applications like video games, medical records, and voting, among others.

Popular NFT marketplaces like OpenSea using Wyvern protocol [23] are able to transfer the NFT ownership state quickly off-chain but cannot support similar performance for multiple consecutive NFT transactions. Lazy minting [24] can reduce gas fee payments from original creators but is still an issue for resellers. In addition, sellers have to go through the process of listing and selling their tokens and incur the gas fee upon accepting offers [25] [26]. This approach is not scalable in applications like online gaming where assets can change hands quickly. Moreover, it is inefficient in terms of gas fee management. At the time of writing this paper, the Ethereum gas fee is around $20 [27] [28] and during some peak times, it can be more than $200. Currently, it's up to the users to choose when to perform the NFT transactions. But there can be a smarter way to run on-chain transactions to save gas fees. Moreover, transaction failures on the on-chain also incur gas fees [26]. There should be a mechanism that performs sanity checks to avoid failing transactions to prevent gas fee wastage. OpenSea has an integration with Polygon [33] which is a Layer 2 scalability solution. With this, the scalability issue can be largely addressed but at the cost of strong security and decentralization guarantees of the Mainnet.

In this paper, we propose LiftChain, a multi-stage transaction protocol for scaling NFT transactions. To ensure that the transactions occur with low latency in an NFT marketplace for a better end-user experience, our model proposes an off-chain marketplace where transactions initially occur with low latency and the final stage of our multi-stage model interacts with on-chain to make the off-chain marketplace eventually consistent with the correct state. Although our model is specific to the NFT marketplace, it can be adapted to any kind of high-volume transactions on blockchain that requires fast responses, given that the applications relying on the transactions can tolerate temporary inconsistencies.

With LiftChain, applications using NFT transactions enjoy the off-chain level performance and it also opens doors for industries that have been so far set back due to scalability issues in adopting NFTs. LiftChain can be used to design

978-1-6654-6104-7/22/$31.00 ©2022 IEEE
DOI 10.1109/Blockchain55522.2022.00057

application-specific marketplaces, thus reducing reliance on external marketplaces which typically take some commission on every transaction. In addition, LiftChain comes with final section batching that can help reduce gas fee costs and improve bandwidth utilization as shown in Section IV-D. LiftChain also has a greedy mechanism, described in Section IV-B, to prevent unnecessary gas fee wastage by proactive ownership checks. LiftChain protocol is adequately lightweight as it is designed to be pluggable with several existing Blockchain scalability solutions [7]. This is an inherent property of its Multi-stage transaction model which we will see in Section IV of the paper. With the model of eventual consistency between off-chain and on-chain, security can be seen as an issue. We show in Section III-C that a malicious actor does not have any incentive in taking advantage of this protocol. In LiftChain, we perform the initial transaction section on off-chain nodes and provide an instantaneous response to the user. Asynchronously, we trigger these transactions on the on-chain and share the results with the off-chain. If there is any discrepancy in the results, we provide updated information to the user with an apology concluding the final section of the transaction.

This paper makes the following contributions:

- Analysis of scalability issues with current NFT transaction protocols for applications with high throughput microtransactions.
- Design of LiftChain protocol (Section III) using Multi-Stage Invariant Confluence with Apologies (MS-IA) [2] that supports any number of transaction inconsistencies at any given time between on-chain and off-chain. This is in contrast with current NFT marketplaces that require a transaction to be synchronized with the on-chain before performing another transaction on a given NFT.
- We provide proof of correctness and security model of our protocol (Sections III-C & IV).
- We prototype and evaluate LiftChain (Section V) and show that LiftChain offers comparable performance to baseline off-chain and more than 5-fold improvement in gas fee savings.

II. Related Work

Current on-chain solutions like sharding [7] can help with scaling to some extent but for real-time high throughput NFT transactions, it still falls behind and comes at the cost of making changes to the Mainnet. Off-Chain solutions don't require changes to the main blockchain protocol and communicate with Mainnet for eventual consistency. Layer-2 based off-chain solutions typically involve transactions initially processed off-chain and then later submitted to the Mainnet. Some implementations are Lightning Network [32], Rollups, State channels, and Sidechains [7]. While these solutions significantly increase the transaction speed and throughput, they come with challenges including transaction ordering, wait times, performing intensive validity proofs, open participation, and trade-off of some decentralization and strong security guarantees provided by Mainnet [9]. Another scaling platform popularly used for NFTs is Polygon [33]. It is based on a Plasma chain, a separate child blockchain that is anchored to the Mainnet and has its own block validation mechanism. It also offers high throughput transactions and reportedly reached over 7200 tps in its testnet. However, this is also susceptible to Block holding attacks, mass exit issues, and less decentralization [9]. OpeanSea using Wyvern protocol [23] can ensure atomic transactions between a seller and buyer, thus giving instant NFT access to the buyer off-chain. But this is limited to a single transaction on an NFT at a given time and there is also a gas fee wastage when reselling an NFT. This closely mimics Multi-Stage Serializability (MS-SR) property [2]. LiftChain on the other hand uses a variation of MS-IA [2] which offers a safety guarantee even with a chain of NFT transactions that are completed off-chain and are still pending on-chain. This is especially helpful for applications like in-game purchases or microtransactions.

LiftChain uses a high availability and eventual consistency model where we aim to address the availability problem by responding to clients quickly using the initial section of the transaction and a correction is sent to clients in the final section. This is suited for an asymmetric environment such as off-chain and on-chain communication. One of the recent works which uses multi-stage transactions in a similar setting is You've got a Friend in ME [36] where the authors apply multi-stage transactions for stock price prediction. Also, using batching and greedy pre-final section verifications, we improve gas fee savings and bandwidth utilization. Hence, our approach offers reasonable accuracy and performance guarantees. Additionally, LiftChain can be integrated with the aforementioned scalability solutions.

III. LiftChain Overview

In this section, we present the overview of LiftChain and design considerations. We also discuss some optimizations for better performance and gas fee savings.

A. System Model

We define clients or users as agents who send and receive NFTs. For the design, we consider a single edge node interacting with multiple clients and a Blockchain. However, this can be extended to multiple edge nodes having different data partitions with a distributed consensus. Also, we define an NFT transaction as the transfer of NFT ownership to another client or receiver. Minting can be modeled as transferring ownership from a null address to a client and the same protocol can be used with slight modifications.

LiftChain is best suited for NFTs that don't mutate the state of the external application relying on them. Suppose a gaming application uses some blocks of land as NFTs which can be used for construction. A momentary inconsistency in the NFT ownership could lead to irreversible state changes in the external application. However, we argue that the majority of such inconsistencies happen due to malicious sellers and in Section III-C, we show that a seller has no incentive to perform such attacks compromising the application state inconsistency.

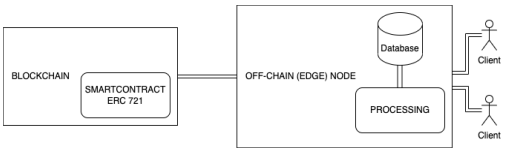

Fig. 1: LiftChain high level architecture

The off-chain node consists of a data store holding the NFT ownership information and transnational data. The processing unit is for performing LiftChain transactions and responding to the clients with the updated state. The high-level architecture of the model is shown in Figure 1 and the workflow is discussed in the next subsection.

B. Workflow

Every transaction in LiftChain is divided into initial and final sections or stages. When a user triggers an NFT transaction on the off-chain node, it checks whether the triggered user is the owner of the NFT. This initial verification is discussed in Section IV-C. Upon successful verification, we execute the initial section and update ownership in the data store. The response is immediately sent to the client once the initial section of the transaction is complete. Next, the final section is initiated by running the transaction on-chain. If the transaction in the on-chain fails, we send an apology to the client and update the ownership in the data store. This concludes the final section of the transaction.

C. Security

We can relate our off-chain nodes to an NFT marketplace. In LiftChain, an NFT seller can be malicious and may try to transfer an NFT without actually being the owner, due to the temporary inconsistency between the on-chain and off-chain state. In fact, initiating a transfer outside the marketplace can lead to the original buyer temporarily losing in-application access to the NFT. This can be prevented by providing guidelines to application users to perform all transactions within the marketplace. In addition, some cryptocurrency can be withheld from the sellers during the transactions to prevent such double selling attacks. Due to failing on-chain transactions also costing gas fees, the attacker has no incentive in trying to double sell. In addition, the exchange protocol should be either atomic or prevent the seller from pulling the funds before the NFT transfer. Similarly, a malicious buyer can be thwarted by locking price funds in the exchange smart contract before initiating the NFT transfer. The application implementing LiftChain can also ensure that the transaction happens in an atomic way using existing solutions [23]. Moreover, the application can revert the ownership state in the off-chain in case the on-chain transaction is taking too long. Once the transaction is successful, the off-chain state can be updated again.

NFT transactions typically involve the buyer paying cryptocurrency in exchange for an NFT. It is recommended to not initiate the NFT transfer before receiving the payment to protect against malicious buyers. Some marketplaces like OpenSea which uses Wyvern protocol, support an atomic transfer of both assets in the transaction. Here a purchase involves sending the NFT bid price and NFT to a trusted smart contract. Then upon authorization using proxy wallets from both parties, the swap happens to change the ownership state atomically. This atomic transaction or swap can be used in LiftChain. This implementation is orthogonal to our protocol and to keep LiftChain lightweight and flexible to support other exchange protocols, we mainly focus our work on the performance aspects and gas fee savings. We show that it is feasible to obtain off-chain level performance and at the same time utilize the inherent security properties of the blockchain. LiftChain tries to give the best possible initial transaction section commit time at the cost of future corrections. So this protocol is well-suited for applications that are tolerant to temporary state inconsistency with Mainnet.

D. Safety

It is important to ensure that our off-chain is eventually consistent with the on-chain data. With just one transaction, safety can be ensured by locking data items involved in the transaction. However, by splitting the transaction into two parts, we are introducing more complexity. We have to account for concurrent transactions within each section and also transactions happening between any transaction's initial and final sections. Also, we can't assume any traditional concurrency control mechanisms to ensure the serializability of transactions with each section as used in [2]. This is because the transactions in on-chain are not guaranteed to be ordered. Hence, we need to come up with a mechanism to ensure the serializability of transactions performed on-chain. This can be either done at the off-chain level or in the smart contract. For simplicity of implementation, we proposed an MS-IA based algorithm in off-chain that ensures safety.

E. Model

1) Client interface: Our model starts with clients initially requesting the off-chain node to transfer an NFT. Let this be denoted by $\langle client\_id, NFT\_id, receiver\_id \rangle$. The response is of two types, (1) the initial section commit response $\langle transaction\_information, status, optional\{newstate\} \rangle$. It also mentions that the response is not final and the state may change, and (2) an apology $\langle transaction\_information, apology, failurereason, optional\{newstate\} \rangle$. The client should only be able to transact an NFT if she is the owner according to the off-chain data store. The client is also provided an option to refresh the ownership data.

2) Definitions:

- We define a transaction with identifier a on NFT id x as $t_a(x)$
- The initial section of a transaction is denoted by t^i and the final section by t^f.
- Two transactions t_a and t_b are said to be conflicting if they operate on the same NFT. In such a case, we need

to serialize them in each section in the same order they are received.
- Retracting a transaction should involve sending a notification to the users that the transaction had failed and freeing any withheld cryptocurrency or assets as part of the exchange.

3) On-chain processing: Our Smart contract residing on the blockchain is responsible for executing the batch of transactions. The off-chain can periodically query the status or use oracles [34] for this purpose. We use the ER-721 standard [12] with modifications to support batch processing and update correct NFT ownership information in the smart contract in case of failed transactions.

IV. LiftChain Protocol Design

In this section, we discuss the LiftChain implementation details. At the heart of the protocol is our concurrency control algorithm based on MS-IA. First, we explore MS-SR which closely mimics existing Decentralized NFT exchanges.

A. Multi-Stage Serializability (MS-SR)

MS-SR gives us the safety principles of serializability. Using MS-SR, we can achieve (1) the final section of a transaction t_a^f commits after the initial section t_a^i (2) two conflicting transactions maintain the same order in both initial and final sections (3) and the final section of a transaction must complete before the initial section of a later transaction.

In MS-SR, we first acquire the lock on the NFT state in the off-chain data store and perform the initial commit. Next, we start the final section which runs the transaction on-chain. To ensure that no other conflicting transaction starts, we hold the lock till the end of the current transaction. This approach is clearly not suitable for applications with high number of conflicting transactions.

B. Multi-Stage Invariant Confluence with Apologies (MS-IA)

Here we discuss a modified MS-IA [2] approach based on invariant confluence [14] and apologies [15]. In MS-IA, the initial section is a guess sent to the client and when the final section executes, an apology will be sent to the client if there was an inconsistency. The execution of the final section is responsible for reconciling the inconsistencies caused by initial sections across conflicting transactions. With invariant confluence, we try to reconcile in the final section by making sure that application-level invariants are preserved. We follow a similar approach of apply-then-check where we first apply the initial section commits and then later merge or reconcile them in the final section execution. The final section should (1) make minimum possible retractions ensuring safety (2) prioritize preserving valid transactions. Figure 2 shows the workflow for MS-IA-based LiftChain.

NFT transactions have the below invariants:

- Two clients cannot hold the same NFT at any given time
- Each NFT is uniquely determined by an identifier

In addition to conflicting transactions being serialized in the same order in each section, we also need the property where the final section of the transaction executes after its initial section. Algorithm 1 ensures both these properties.

Algorithm 1 MS-IA for NFT transactions in LiftChain

```
nft_id ← get_nft_id(t_a^i)
if acquirelock(nft_id) then
  execute(t_a^i)
else
  abort
end if
Initial Commit
releaselocks(nft_id)
BEGIN execute(t_a^f)
    t_a^f ← fetch_transaction() // fetches the latest transaction
from the priority queue.
    nft_id ← get_nft_id(t_a^f)
    acquire_final_section_lock(nft_id)
    response ← run_transaction_on_blockchain(t_a^f)
    if response is SUCCESS then
      commit(t_a^f)
    else
      owner ← get_nft_owner(response)
      nft_id = get_nft_id(t_a^f)
      retract(t_a^i)
      Update ownership of nft_id in the off-chain datastore
      for MS-IA
    end if
    release_final_section_lock(nft_id)
END execute(t_a^f)
Final Commit
```

The final section of our MS-IA follows the pattern of a producer-consumer model with a thread-safe queue. Once the initial commit is done, we push the transaction to the queue. As the initial section is synchronized using a lock, the order of the conflicting transactions is also preserved in the final sections. This measure is taken because Blockchain doesn't guarantee the order of transactions. It is also noteworthy that we keep a separate NFT data store for MS-IA's final section. With this isolation, we only update the original data store during the transaction when the off-chain should become consistent with the on-chain. Also note that in the final section, if a transaction fails, using the new owner information we skip the subsequent final sections and retract their initial sections until an owner matches. This saves gas fees as the transactions are anyways going to fail. It is also possible that the NFT ownership changes on-chain while processing the final transactions making any of the aborted transactions valid. We see this as a rare case and still retract greedily prioritizing gas fee savings.

In the next two theorems, we show that the above MS-IA algorithm preserves the NFT transaction invariants and ensures a minimum number of retractions.

Theorem 1. *A transaction t_i retracting its initial section is sufficient to ensure consistency and preserve invariants in NFT*

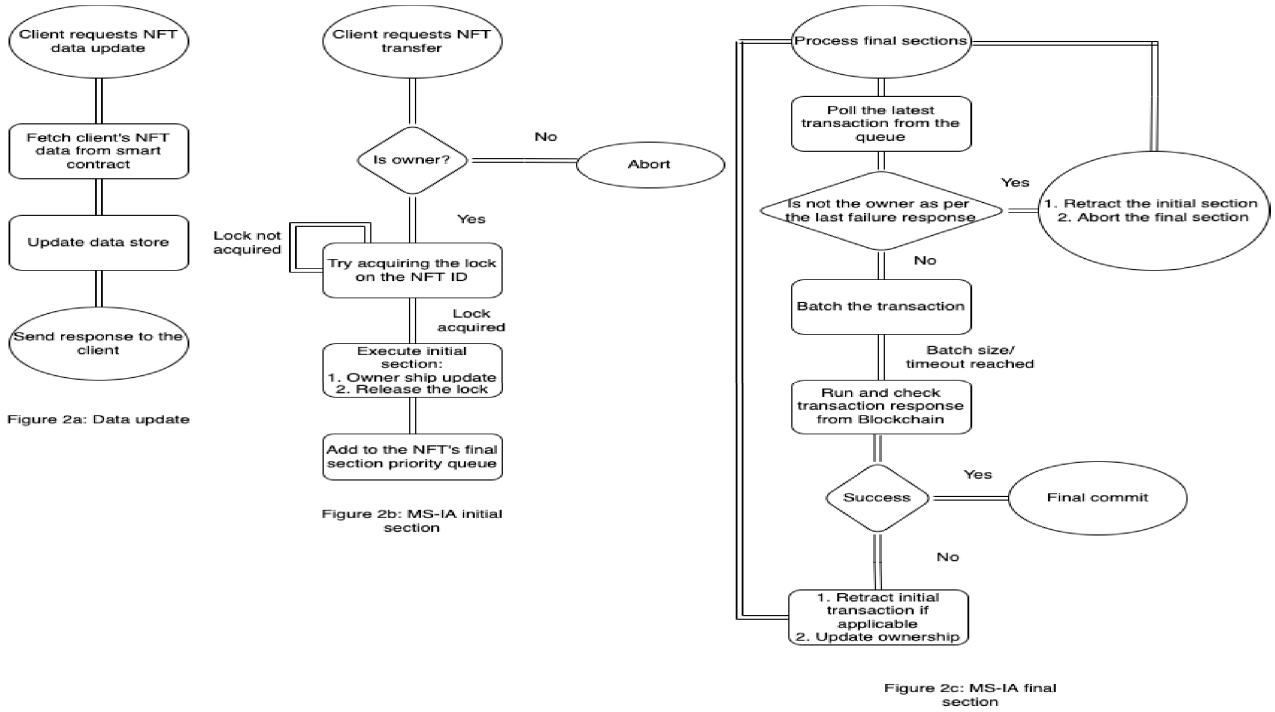

Fig. 2: Lift Chain transaction flow with data update control in Figure 2a and MS-IA initial and final sections in 2b and 2c respectively

transactions.

Proof. Due to locking in both initial and final sections, there can be only one owner for a given NFT at any time. With ownership update at each failed on-chain transaction and serialization of the final sections using queuing, any subsequent conflicting transactions either fail or succeed if the client is the actual owner. Other conflicting transactions' final sections will correct their corresponding initial sections, hence requiring no other coordination during apologies to maintain consistency. □

Theorem 2. *The execution function of MS-IA minimizes the number of retractable transactions in off-chain.*

Proof. Case 1: Let $t_1, t_2, t_3, ..., t_n$ be non conflicting transactions. As they operate on different NFTs, any failure in the on-chain will only retract its corresponding transaction and does not affect other transactions.

Case 2: Let $t_1, t_2, ..., t_n$ be all conflicting transactions. As per our MS-IA algorithm, the final sections are all executed sequentially in the same order as the initial sections. Say if transaction t_i fails, we retract its initial section and update the ownership (updated owner = O) in the MS-IA datastore for the final section. Using this updated owner information, we skip the subsequent final sections and retract their initial sections until the sender of $t_k \neq O$. We then execute the final section of t_k and proceed with the same steps. This ensures that we only retract the transactions based on the updated ownership instead of failing all transactions from t_{i+1} to t_n.

Also, if we assume that a transaction t_x where $i < x < k$, does not need to be retracted, it implies that the sender of t_x is O for it to be successful. But as per our above statement, the first transaction where the sender is O is t_k. However, $t_k \neq t_x$. So this is a contradiction and we have all transactions t_x where $i < x < k$ retracted. □

C. Initial Verification

The initial section first checks if the NFT transaction is initiated by the owner. This can be done either by (1) checking the off-chain data store (2) querying the smart contract (3) a hybrid approach by restricting the queries to the smart contract only when the client has been idle for some time. Reads from the smart contract can take a long time up to several ms as shown in our evaluations. However, if the transaction fails on-chain, the client incurs a gas fee. With the hybrid approach, LiftChain can provide a balance between latency and gas fee savings.

D. Batching

Instead of making requests to the on-chain smart contract for every transaction, we batch requests ensuring that a batch does not contain any conflicting transactions. This is because blockchain does not guarantee transaction ordering. Batching helps with efficient bandwidth utilization and scaling the system as there can only be a limited number of open

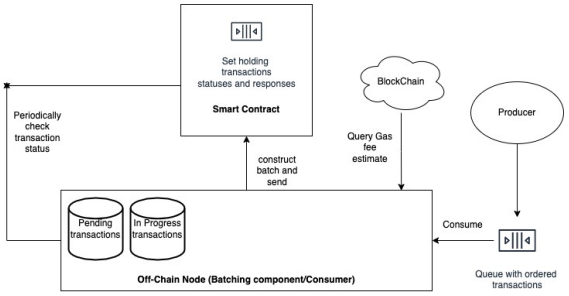

Fig. 3: Batching in LiftChain

connections due to resource constraints. Using batching, we run transactions during non-peak hours and spread them in order to reduce the overall gas fee. This strategy can be prioritized for low-value NFTs.

Our batching strategy is based on a producer-consumer model as shown in Figure 3. The initial section of MS-IA acts as a producer and adds transactions to a thread-safe queue. The final section polls from the queue and adds to the current batch to the In-Process set IP. If the transaction is already in the IP, we place it in a Pending set P. We trigger batch processing whenever the queue size crosses a threshold or within some predefined time period to ensure progress. Once the batch is ready, we send it to the on-chain and terminate the connection to save network resources. LiftChain maintains a data structure in the smart contract with transaction statuses which the off-chain node periodically queries and updates its IP to complete final sections. In addition, we also periodically query for gas fee estimates in blockchain for batching NFT transactions.

V. Evaluation

In this section, we evaluate LiftChain performance when compared to on-chain and off-chain baselines. We also evaluate MS-IA used in LiftChain and MS-SR which mimics the serialization done in most existing decentralized exchanges. We perform the following three experiments to evaluate our claims.

- Compare LiftChain response times with off-chain and on-chain response times.
- Compare the performance of MS-SR and MS-IA.
- Evaluate the gas fees and communication time with on-chain and show how batching can improve bandwidth utilization and gas fee savings.

A. Experiment Setup

We are using Ropsten Testnet Network [35] for evaluation. Ropsten is a test network for Ethereum Mainnet that allows blockchain development testing before deployment on the main Ethereum network. At the time of experimentation, the average gas burn for the Ropsten network is around 25 Gwei per Block. Block-time is around 30 seconds and the network has an average of 250k transactions per day. The off-chain node has an 8-core processor and 8 GB RAM. The client interacts with LiftChain using a command-line interface.

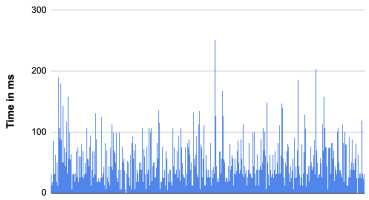

Fig. 4: 10000 conflicting transactions with 5% failures

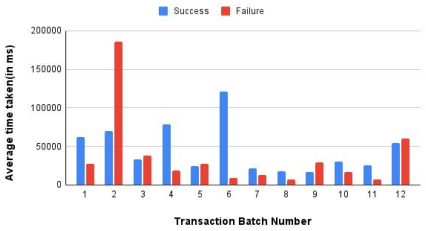

Fig. 5: On-chain transaction time for success and failure cases

Following cases are used to test the inconsistencies between on-chain and off-chain.

- The client performs a transaction directly on-chain.
- The client places several concurrent requests in off-chain.
- We only test for cases where a transaction is successful off-chain but fails on-chain. If a transaction fails in the off-chain but is supposed to be successful, we ask clients to update the NFT balance sheet off-chain by using an API call (See Figure 2a).

B. Experimental Results

1) Comparison with off-chain and on-chain: In this experiment, we evaluate transactions on LiftChain and compare them with direct on-chain and off-chain transactions. We will see that LiftChain transactions have similar performance as the off-chain baseline.

Figure 4 shows baseline off-chain execution times for conflicting transactions with a 5% transaction failure rate. From Figure 5, we have the success and failure transaction time in the on-chain ranging from 7 seconds to 150 seconds with an average of 25 seconds. This is again dependent on the gas price and how busy the network can be. Table I shows LiftChain initial section transaction times for different settings in MS-IA. In the last row, we find that with failure transactions in conflicting and non-conflicting cases the time taken is similar ($\sim$6 ms). This indicates that failure responses are fast as there is no data store update and queuing for final sections. From Figure 4 and Table I, it is clear the MS-IA initial client response times are comparable to baseline off-chain performance.

2) Comparing MS-SR and MS-IA: This evaluation compares LiftChain with MS-SR based transaction protocol as MS-SR mimics most existing decentralized exchanges with one level of inconsistency with on-chain.

	Number of transactions	Time in seconds
Conflicting with 5% failures	10	2.20
	1000	9.53
	10000	75.41
	40000	279.23
Conflicting with 5% failures	**Number of transactions**	**95th Percentile response time in ms**
	1000	37.59
	10000	43.63
	40000	37.83
Non-conflicting with 5% failures	**Number of transactions**	**95th Percentile response time in ms**
	1000	43.94
	10000	37.41
1000 failure transactions	**Transaction type**	**95th Percentile response time in ms**
	Conflicting	6.01
	Non-conflicting	6.00

TABLE I: LiftChain initial section response time to client

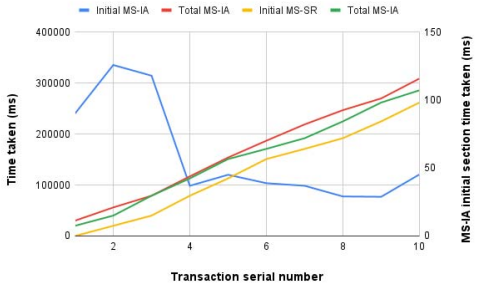

Fig. 6: MS-SR vs LiftChain MS-IA (Initial section time and Total time)

Figure 6 shows a breakup of initial and final section execution times in LiftChain with 10 conflicting transactions. The initial section of the MS-IA takes around 70 ms on average. The total time of MS-IA per transaction keeps increasing from ∼30 seconds to 300 seconds. On-chain execution time is on average 30 seconds. This contributes to the majority of the latency but the user gets responses within 70 ms after the initial section. The increase in the total time can be attributed to the synchronous behavior of the final sections. In MS-SR, we see that although the first initial section happens within a few ms, subsequent initial section times follow the total time curve with ∼28 seconds difference. The main performance bottleneck in MS-SR is that the locks are held till the final section commits. MS-IA has a small overhead of an additional queuing logic for the final sections but it is relatively small compared to the lock contention of the MS-SR.

3) Gas fee and bandwidth utilization using batching in LiftChain: As part of this experiment, we evaluate the gas fee savings with LiftChain. Figure 7 shows the communication time between the off-chain and on-chain which is around 50 ms. This is part of the final section and should not negatively impact the user experience. On the other hand, the network and CPU resources used by off-chain can be efficiently managed by batching the final sections. This will help in reducing

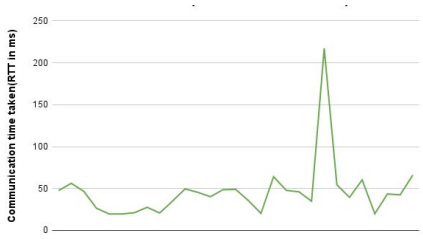

Fig. 7: Communication time taken (Off-chain to On-chain)

On-chain transaction settings	Transaction fee (in Ether)	Average time taken on On-chain(in sec)
Max priority: 50 Gwei	0.0024631	58.40
Max priority: 500 Gwei	0.024631	22.21
Max priority: 5000 Gwei	0.24631	20.89
No constraints(Success)	0.0004967	46.45
No constraints(Failures)	0.000255972	27.57

TABLE II: Transaction fee and transaction time

bandwidth utilization and network connections overhead. Also, since the number of concurrent links that can be open at a time is limited, this significantly helps with scaling. Table II shows the gas fee and transaction times with different settings. It is clear that the transaction time decreases as we are willing to spend more fees but saturates after a certain point. These differences might be more granular when we work on Mainnet. As per [18] and [30], the gas fee varies significantly throughout the day. So batching NFT transactions and spreading them to run on the Mainnet when the fee estimate is low can improve the gas fees savings by more than 5 fold.

VI. Conclusion

In this paper, we highlighted current issues with the scalability of NFT transactions, especially for real-time applications like online gaming which require low latency microtransactions. We proposed LiftChain, a multi-stage transaction model that guarantees quick responses comparable to a baseline off-chain. In our model, all transactions eventually run in the on-chain, thus inhering its decentralization and safety features. LiftChain also comes with security measures that ensure that malicious users have no incentive in misusing the protocol. LiftChain uses MS-IA for transactional safety and we proposed an MS-IA algorithm that retracts the minimum possible transactions upon failures. We compared LiftChain with MS-SR based concurrency control that imitates several existing exchange protocols and showed that our MS-IA based mechanism gives significant performance improvement and is well suited for real-time applications integrating with NFT marketplaces.

Our future works include integrating LiftChain with other blockchain scalability solutions [7]. We also plan to build a decentralized exchange leveraging the eventual consistency model of LiftChain to minimize the number of on-chain transactions required for asset exchanges. In addition, LifChain batching can use machine learning techniques for improvising gas fee savings and dynamic batch sizes.

VII. Acknowledgement

This research is supported in part by the NSF under grant CNS-1815212.

References

[1] "Ethereum Development Documentation," ethereum.org. [Online]. Available: https://ethereum.org/en/developers/docs/. [Accessed: 29-Jan-2022].

[2] S. Gazzaz, V. Chakraborty, F. Nawab, "Croesus: Multi-Stage Processing and Transactions for Video-Analytics in Edge-Cloud Systems," arXiv preprint arXiv:2201.00063, 2021. [Accessed: 29-Jan-2022].

[3] Q. Wang, R. Li, Q. Wang, and S. Chen, "Non-fungible token (NFT): Overview, evaluation, opportunities and challenges," arXiv preprint arXiv:2105.07447, 2021. [Accessed: 29-Jan-2022].

[4] L. Ante, "The non-fungible token (NFT) market and its relationship with Bitcoin and Ethereum," SSRN, 08-Jun-2021. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3861106. [Accessed: 01-Feb-2022].

[5] D. Abadi, O. Arden, F. Nawab, and M. Shadmon, "Anylog: a grand unification of the internet of things," Conference on Innovative Data Systems Research (CIDR '20), 2020.

[6] M. Alaslani, F. Nawab, and B. Shihada, "Blockchain in iot systems: End-to-end delay evaluation," IEEE Internet of Things Journal, vol. 6, no. 5, pp. 8332–8344, 2019.

[7] "Scaling Ethereum". ethereum.org. [Online]. Available: https://ethereum.org/en/developers/docs/scaling/. [Accessed: 12-Feb-2022].

[8] B. N. Musungate, B. Candan, U. C. Cabuk, and G. Dalkilic, "Sidechains: Highlights and challenges," in Proc. Innov. Intell. Syst. Appl. Conf. (ASYU), Oct. 2019, pp. 1–5.

[9] C. Sguanci, R. Spatafora, and A. M. Vergani. Layer 2 Blockchain Scaling: a Survey. arXiv preprint arXiv:2107.10881, 2021.

[10] C. Xu, C. Zhang, J. Xu, and J. Pei, "SlimChain: Scaling blockchain transactions through off-chain storage and parallel processing," Proceedings of the VLDB Endowment, pp. 2314–2326, 2021.

[11] K. B. Muthe, K. Sharma, and K. E. N. Sri, "A blockchain based decentralized computing and NFT infrastructure for game networks". In 2020 Second International Conference on Blockchain Computing and Applications (BCCA) (2020, November) (pp. 73-77). IEEE.

[12] "ERC 721 specification" ERC. [Online]. Available: http://erc721.org/. [Accessed: 26-Feb-2022].

[13] P. A. Bernstein, V. Hadzilacos, and N. Goodman, "Concurrency Control and Recovery in Database Systems". Addison-Wesley, 1987.

[14] P. Bailis, A. Fekete, M. J. Franklin, A. Ghodsi, J. M. Hellerstein, and I. Stoica, "Coordination avoidance in database systems," Proceedings of the VLDB Endowment, vol. 8, no. 3, pp. 185–196, 2014.

[15] P. Helland and D. Campbell, "Building on quicksand," in CIDR 2009, Fourth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 4-7, 2009, Online Proceedings. www.cidrdb.org, 2009. [Online]. Available: http://www-db.cs.wisc.edu/cidr/cidr2009/Paper_133.pdf

[16] W. Canny, "Jefferies sees the NFT market reaching more than $80B in value by 2025," CoinDesk Latest Headlines RSS, 20-Jan-2022. [Online]. Available: https://www.coindesk.com/business/2022/01/20/jefferies-sees-the-nft-market-reaching-more-than-80-billion-in-value-by-2025. [Accessed: 26-Feb-2022].

[17] M. Nadini, L. Alessandretti, F. Di Giacinto, M. Martino, L. M. Aiello, and A. Baronchelli, "Mapping the NFT revolution: Market trends, Trade Networks, and visual features," Scientific Reports, vol. 11, no. 1, 2021.

[18] "Anyblock Analytics," AnyBlock Dashboards. [Online]. Available: https://dashboards.anyblock.tools/ethereum/ethereum/ropsten/network-metrics/ [Accessed: 17-Mar-2022].

[19] "Welcome to meta: Meta," Welcome to Meta — Meta. [Online]. Available: https://about.facebook.com/meta/. [Accessed: 20-Mar-2022].

[20] "The metaverse is coming. here are the cornerstones for securing it.," The Official Microsoft Blog, 29-Mar-2022. [Online]. Available: https://blogs.microsoft.com/blog/2022/03/28/the-metaverse-is-coming-here-are-the-cornerstones-for-securing-it/. [Accessed: 30-Mar-2022].

[21] "Market tracker — NFT sales and trends — nonfungible.com." [Online]. Available: https://nonfungible.com/market-tracker. [Accessed: 30-Mar-2022].

[22] "Top nft floor price by trading volume," CoinGecko. [Online]. Available: https://www.coingecko.com/en/nft. [Accessed: 30-Mar-2022].

[23] "Wyvern protocol," Wyvern Protocol. [Online]. Available: https://wyvernprotocol.com/. [Accessed: 30-Mar-2022].

[24] "Can I list an item without paying to 'mint' it? – opensea." [Online]. Available: https://support.opensea.io/hc/en-us/articles/1500003076601-Can-I-list-an-item-without-paying-to-mint-it-. [Accessed: 30-Mar-2022].

[25] "Who pays the gas fees when using Ethereum on OpenSea?" [Online]. Available: https://support.opensea.io/hc/en-us/articles/360061699514-Who-pays-the-gas-fees-when-using-Ethereum-on-OpenSea-. [Accessed: 30-Mar-2022].

[26] "What are gas fees on Ethereum? – opensea." [Online]. Available: https://support.opensea.io/hc/en-us/articles/1500006315941-What-are-gas-fees-on-Ethereum. [Accessed: 30-Mar-2022].

[27] "Get estimation of confirmation time," Etherscan. [Online]. Available: https://docs.etherscan.io/api-endpoints/gas-tracker. [Accessed: 30-Mar-2022].

[28] "OpenSea Activity," OpenSea. [Online]. Available: https://opensea.io/activity. [Accessed: 30-Mar-2022].

[29] "UNISWAP," Uniswap Protocol. [Online]. Available: https://uniswap.org/. [Accessed: 31-Mar-2022].

[30] "Ethereum Gas Price Charts" ethereumprice, 12-Mar-2021. [Online]. Available: https://ethereumprice.org/gas/. [Accessed: 02-Apr-2022].

[31] M. T. Najaran and C. Krasic, "Scaling online games with adaptive interest management in the cloud," 2010 9th Annual Workshop on Network and Systems Support for Games, 2010, pp. 1-6, doi: 10.1109/NETGAMES.2010.5680282.

[32] Y. Guo, J. Tong and C. Feng, "A Measurement Study of Bitcoin Lightning Network," 2019 IEEE International Conference on Blockchain (Blockchain), 2019, pp. 202-211, doi: 10.1109/Blockchain.2019.00034.

[33] "Bring the world to ethereum," Polygon. [Online]. Available: https://polygon.technology/. [Accessed: 02-Apr-2022].

[34] "Oracles," ethereum.org. [Online]. Available: https://ethereum.org/en/developers/docs/oracles/. [Accessed: 02-Apr-2022].

[35] "TESTNET Ropsten (ETH) Blockchain Explorer." [Online]. Available: https://ropsten.etherscan.io/. [Accessed: 05-Apr-2022].

[36] Z. Karl, H. Freedman, A. Showail, A. Singh, S. Gazzaz, and F. Nawab, "You've Got a Friend in ME (Mobile Edge): Blockchain Processing with Cloud Node Backup," in First International Symposium on Recent Advances of Blockchain Evolution: Architecture, Intelligence, Incentive, and Applications (BlockchainEvo 2022), 2022.