Garou: An Efficient and Secure Off-Blockchain Multi-Party Payment Hub

Yongjie Ye, Zhifeng Ren, Xiapu Luo, Jingjing Zhang, and Weigang Wu, *Member, IEEE*

***Abstract*—To mitigate the scalability problem of decentralized cryptocurrencies such as Bitcoin and Ethereum, the payment channel, which allows two parties to perform secure coin transfers without involving the blockchain, has been proposed. The payment channel increases the transaction throughput of two parties to a level that is only limited by their network bandwidth. Recent proposals focus on extending the two-party payment channel to the N-party payment hub. Unfortunately, none of them can achieve efficiency, flexibility in the absence of a trusted third-party. In this paper, we propose Garou, a secure N-party payment hub that allows multiple parties to perform secure off-chain coin transfers. Except in the case of disputes, participants within the payment hub can make concurrent and direct coin transfers with each other without the involvement of the blockchain or any third-party intermediaries. This allows Garou to achieve both high-performance and flexibility. If all participants keep online, Garou guarantees all honest users' balance security against strong adversarial capabilities. To demonstrate the feasibility of the Garou protocol, we develop a proof of concept prototype for the Ethereum network. Experimental evaluations show that, with 300 participants and without disputes, that the maximum transaction throughput of Garou is 20× higher than that of state-of-the-art payment hubs.**

***Index Terms*—Off-chain protocol, payment hub, smart contract, scalability, concurrency.**

I. Introduction

SINCE the advent of Bitcoin [1] in 2008, decentralized cryptocurrencies have gained great popularity over the last ten years. The combination of consensus algorithm (e.g., POW [2] and POS [3]) and tamper-resistant hash-linked chain of blocks, i.e., blockchain, allows secure coin transfers between mutually untrusting peers across the world. Inspired by Bitcoin, other cryptocurrencies supporting the execution of Turing-complete smart contracts have emerged. These cryptocurrencies allow coins to be managed by user-defined smart contracts of arbitrary complexity. Among them, the most prominent one is Ethereum [4], which uses Solidity as the programming language for its smart contracts.

However, the major inherent limitation of these cryptocurrencies lies in their performance. For example, Bitcoin can only support up to 6∼7 transactions per second while Ethereum supports up to 20 transactions per second [5]. The global nature of consensus algorithms makes it impossible to grain a magnitude growth of throughput with simple re-parameterization [6]. Such a low transaction throughput is far from enough to support the widespread use of decentralized cryptocurrencies, let alone the execution of complex smart contracts. In contrast, Visa processes up to 47,000 transactions per second [7].

To mitigate the scalability problem of decentralized cryptocurrencies, the payment channel [8], [9], which allows two parties to perform secure off-chain micropayments, has been proposed. One extension of the two-party payment channel is the payment network, which uses the payment channels as building blocks to construct a linked off-chain overlay. The payment network uses the *Hashed Time-Lock Contract* (HTLC) transactions to route payments across multiple intermediary payment channels with atomicity. However, HTLC transactions require that every intermediary payment channel locks a portion of its available balance capacity until the payment is settled, which may lead to a deadlock in a concurrent situation [10]. Additionally, to increase the flexibility of payment channels, channel factories [11], [12] can make coins transfer among multiple parties without opening direct payment channels.

Recently efforts have been made to extend the two-party payment channel to N-party payment hub, which is organized in a star-like topology and allows multiple parties to perform secure off-chain coin transfers. Compared with other off-chain scalability proposals such as channels and sidechains, payment hub has higher the transaction flexibility, lower consensus overhead and high-performance.

TumbleBit [13] is a Bitcoin-compatible anonymous payment channel hub (PCH), which guarantees the untraceability of transactions between its payer to its payee. Tairi *et al.* [14] propose anonymous atomic locks (A^2L), which improves the interoperability and efficiency of TumbleBit [13] while preserving the security and privacy. On the basis of the state channel [15], Dziembowski *et al.* [16] propose Perun, a payment channel hub that is maintained by one intermediary Ingrid among multiple parties.

Manuscript received December 29, 2020; revised May 23, 2021 and July 25, 2021; accepted July 27, 2021. Date of publication August 2, 2021; date of current version December 9, 2021. This research is partially supported by the National Natural Science Foundation of China (U1711263, U1811461, U1801266), and Guangdong Provincial Natural Science Foundation of China (2018B030312002). The associate editor coordinating the review of this article and approving it for publication was A. Veneris. *(Corresponding author: Weigang Wu.)*

Yongjie Ye was with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510275, China (e-mail: yeyj6@mail2.sysu.edu.cn).

Zhifeng Ren, Jingjing Zhang, and Weigang Wu are with the School of Computer Science and Engineering, and Guangdong Province Key Lab of Big Data Analysis and Processing, Sun Yat-sen University, Guangzhou 510275, China (e-mail: renzhf@mail2.sysu.edu.cn; zhangjj43@mail2.sysu.edu.cn; wuweig@mail.sysu.edu.cn).

Xiapu Luo is with the Department of Computing, Hong Kong Polytechnic University, Hong Kong (e-mail: csxluo@comp.polyu.edu.hk).

Digital Object Identifier 10.1109/TNSM.2021.3101808

1932-4537 © 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Wuhan University. Downloaded on September 09,2024 at 06:51:45 UTC from IEEE Xplore. Restrictions apply.

Apart from the payment hubs mentioned above, Commit-chains are specifically proposed for the off-chain payment scenario to optimize transaction protocol. In contrast to the three-state model of payment channel, commit-chains keep an always ongoing state once launched and achieve periodic on-chain checkpoint commitments. The operator is responsible for facilitating the execution of transactions and user's enrollment and withdrawal.

NOCUST [17] is a new specification for the N-party payment hub with a high degree of flexibility, which allows its participants to transfer all their allocated funds to any other member. Plasma [18] is a high-level specification of a UTXO-based commit-chain, which is maintained by central operators on the top of an account-based blockchain.

In summary, extending the two-party payment channel to the N-party payment hub is a nontrivial task. None of the existing proposals can achieve both efficiency, flexibility, and untrusty.

A. This Work

In this paper, we propose a high-performance, flexible and untrusted N-party payment hub called Garou. It allows multiple parties to make concurrent and flexible off-chain coin transfers with each other without compromising security. Specifically, we adopt an automatic leader election mechanism to eliminate the reliance on any trusted third-party, which also can prevent the leader from launching a collusion attack. The separation of spending and receiving allows Garou to achieve high concurrency. In addition, the escrow-free design allows participants to spend their money flexibly. Finally, Garou allows its participants to join in and withdraw from the payment hub at any time. The main features of Garou are listed as follows.

- *Efficiency:* Garou is designed for high-performance off-chain coin transfers. Concurrent transactions are allowed among participants of the payment hub. Except in the case of disputes, the off-chain transaction of Garou does not require the involvement of the blockchain, and therefore Garou can significantly increase the performance and scalability of the blockchain systems. Our evaluation results show that the maximum transaction throughput of Garou is 20 $\times$ higher than that of state-of-the-art payment hubs.
- *Flexibility:* Garou exhibits a high level of transaction flexibility, which allows its participants to freely transfer all their available funds to any other member of the payment hub. It does not require any intermediary nodes to route coin transfers or escrow their coins before transactions. Besides, Garou allows its participants to join in or withdraw from the payment hub at any time.
- *Security:* Garou guarantees that an honest party will not bear any financial losses despite strong adversarial capabilities. Using the on-chain smart contract as a dispute resolver, an honest party can always maintain its balance security even when all other members of the payment hub are malicious. Additionally, Garou does not rely on any third-party operators and therefore can effectively avoid the single points of failure.

B. Organization of the Paper

In Section II, we provide the necessary background and review the state-of-the-art payment hub designs. In Section III, we present the system and threat models, and then give a overview of the Garou protocol. In Section IV, we present the detailed design of the Garou protocol. The security of Garou is analyzed in Section V. In Section VI, we provide our proof-of-concept implementation and the evaluation results based on the Ethereum network. Finally, we conclude this paper in Section VII.

II. Background and Related Works

Recently many attempts have been made to improve the scalability of blockchains, such as alternative consensus mechanisms [19]–[21], usage of trusted execution environment [22], [23], blockchain sharding [24]–[27], rollups [28], payment channel and payment network [8], [9], [29], etc.

The sharding technique splits the blockchain into multiple shards, and each shard forms a consensus group and maintains its fraction of nodes or transactions. This could improve the performance of one blockchain system to thousands of transactions per second and reduces confirmation time to a few seconds [27]. Moreover, sharding does not sacrifice the transaction traceability, i.e., all transactions still need to be committed to the blockchain. However, sharding makes the blockchain much more complex, and many challenging issues must be carefully studied, such as cross-shard atomicity [30], transaction placement [27], single-shard takeover attack [31], etc.

Rollups [28] is a special off-chain scaling solution, which stores compressed transaction data on the blockchain to achieve data availability. In Optimistic rollups [32], contract keeps track of its entire history of state roots and the hash of each batch. If anyone discovers that one batch has an incorrect post-state root, it can publish a proof to chain, proving that the batch was computed incorrectly. In ZK rollups [33], every batch includes a cryptographic proof called ZK-SNARK [34], which proves that the post-state root is the correct result of executing the batch. No matter how large the computation, the proof can be very quickly verified on-chain. Although rollups can improve the scalability of the blockchain, the transaction data needs to be stored on the root chain, which is an obvious bottleneck of performance.

Another approach to improving the scalability of cryptocurrencies is the off-chain payment channel/network. At the cost of transaction traceability, the off-chain techniques can achieve substantial performance improvement and lower storage burden than the sharding techniques.

Our work focuses on the design of payment hub, which is an extension of off-chain payment channel technique.

A. Channels

1) Payment Channels: Payment channels [8], [9], [35] allow two parties to perform secure coin transfers privately without involving the blockchain, yet still maintaining balance security of honest parties. The payment channel can significantly increase the transaction throughput of two parties to a

Authorized licensed use limited to: Wuhan University. Downloaded on September 09,2024 at 06:51:45 UTC from IEEE Xplore. Restrictions apply.

TABLE I
COMPARISON OF PAYMENT HUB DESIGNS

	Channel Hub	**Commit-chain**	**Garou**
Topology	Star	Star	Dynamic Star
Lifecycle	3-phase	Periodic on-chain commit	Periodic off-chain checkpoint
Third Party	✓	✓	✗
TX Flexibility	✗	✓	✓
TX concurrency	✗	✗	✓
TX Finality	Instant	Delayed	Delayed
Collateral Allocation	$O(n)$ on-chain	$O(1)$ on-chain	$O(n)$ on-chain

level that only limited by their network bandwidth, as well as reducing the storage burden for the blockchain system. The Lightning Network [9] resorts to punishment to prevent malicious nodes from rolling back.

2) Payment Networks: Using the two-party payment channel as building blocks, we could construct a linked payment channel network (PCN) [9], [29], [36], [37], where nodes can utilize multiple intermediary payment channels to establish a multi-hop path and route coins to others. To achieve the atomicity, the Lightning Network [9] adopts a technique called *Hashed Time-Lock Contract* (HTLC) transaction. In addition to atomicity, some other factors of the payment network have also been concerned, such as privacy [38], concurrency [10], and routing efficiency [39], [40], [41].

3) Channel Factories: Burchert *et al.* [11] propose the concept of a channel factory for Bitcoin. In a channel factory, n parties lock coins in an n-party deposit by signing the funding transaction, which is then re-allocated to a set of pair-wise payment channels. This re-allocation of pair-wise channels can be built using Duplex Micropayment Channels (DMC) [8], while Pedrosa *et al.* [12] replace DMC with Lightning channels.

B. Payment Hubs

In this paper, the term "payment hub" refers to an infrastructure that allows multiple parties to make off-chain coin transfers with each other. The payment hub is usually maintained by one single party that acts as an intermediary for facilitating transactions.

In the following, we review existing works on the payment hub including channel hubs and commit-chains, and compare them with our proposed one. The differences between them are shown in Table I.

1) Payment Channel Hub: TumbleBit [13] is an anonymous payment hub designed for the Bitcoin network. It uses cryptographic techniques to ensure the privacy and anonymity of its participants, i.e., no one could link a payment from its payer to its payee. TumbleBit requires that each participant open a directed payment channel with the untrusted intermediary, which is called Tumbler. In contrast, our proposed Garou does not require any intermediary nodes to route coin transfers or escrow their coins before transactions.

Tairi *et al.* [14] propose a cryptographic PCH realization, the core of which lies in a novel cryptographic primitive called anonymous atomic locks. A^{2}L improves the interoperability and efficiency of TumbleBit [13] while preserving the security and privacy. However, the collusion between the Tumbler and the payee might happen in both TumbleBit and A^{2}L.

Perun [16] is a virtual payment channel built on top of existing ledger channels to enabling efficient off-chain transactions between two parties that are connected to one intermediary Ingrid. To connect users that do not have a joint open channel, existing channels can be composed to form so-called virtual channels. However, any state transition of Perun must be explicitly agreed by all participants, which leads to poor performance in the payment scenario. In contrast, our proposed payment hub is dedicated to off-chain coin transfers.

2) Commit-Chain: Commit-chains have a network topology similar to the payment channel hub and also serve a similar purpose. However, Commit-chains specifically provide transaction protocol optimized for the off-chain payment scenario.

Plasma [18] is a high-level specification of a UTXO-based commit-chain, which is maintained by central operators on the top of an account-based blockchain. The recipient of Plasma needs to verify the entire transaction history for a complete coin transfer, which results in a linear growing of the data storage and an expensive computation cost. Additionally, since the operators are untrusted third parties, Plasma needs to perform on-chain commitments frequently to prevent a double-spends attack. This leads to a low transaction throughput and a high overhead of transaction fees on the Ethereum platform.

NOCUST [17] is an N-party payment hub with a high degree of flexibility, which allows the allocated funds of a party to be totally paid to any other member of the payment hub. A NOCUST payment hub consists of two fundamental components: an on-chain verifier contract and an off-chain operator server. The on-chain verifier contract serves as a trusted financial custodian which manages the deposits of all participants and acts as a dispute resolver. The off-chain operator server executes every coin transfer and periodically commits the balance information of all participants (encoded into a merklelized interval tree) to the on-chain verifier contract to keep consistency.

The major drawback is that the execution of the NOCUST payment hub relies on a third-party operator server, exposing the system to the single point of failure. The operator server is also required to make costly on-chain data commitments periodically, which de-incentivizes its election, causes substantial overhead to the payment hub, and significantly limits the scalability of the blockchain system. Moreover, NOCUST does not allow its participants to initiate any other transfers until the current one is done, which limits the concurrency of the payment hub. In contrast, our proposed protocol does not rely on any third-party operators. All operations can be done internally. Garou also allows concurrent coin transfers between participants and therefore Garou can achieve high-performance.

III. SYSTEM OVERVIEW

In this section, we present the system, network and threat models, and then give a brief overview of the Garou protocol.

Authorized licensed use limited to: Wuhan University. Downloaded on September 09,2024 at 06:51:45 UTC from IEEE Xplore. Restrictions apply.

A. System Model

1) Blockchain and Smart Contract: In this paper, we model the blockchain as a trusted and immutable ledger that records all valid on-chain transactions. Those transactions will be globally available for all participants after being confirmed on the blockchain, usually on an average block time *T*. In a typical blockchain network, every participant is uniquely identified by its account address, which is usually hash-derived from a public key.

In addition to regular coin transfers between participants, many blockchain systems also support the execution of arbitrary complex smart contracts. Our proposed protocol requires a smart contract execution environment such as Ethereum [42]. Smart contracts allow money to be fully controlled by its program code. Once a smart contract is deployed on the blockchain, its public functions can be invoked by anyone via an on-chain transaction, which is processed by the miners. Transaction fees have to be paid to incentivize miners to process the transaction and execute the smart contracts. In addition, the same piece of smart contract code can be deployed multiple times. Each contract instance has its own globally unique identifier (known as *contract account* in the Ethereum network), and different contract instances are independent of each other.

2) Network Model: Given that every participant in the blockchain network is uniquely identified, we assume an underlying communication network that all participants of the payment hub can communicate directly with each other. The communication between participants is authenticated and confidential, e.g., through TLS. We further assume that the communication channels between participants are synchronous, i.e., messages sent between honest participants arrive within a maximum delay Δ. Finally, we assume that the connections between any two uncompromising participants and the communication with the blockchain network are uncorrupted. We note that all these above assumptions are reasonable and held in almost all blockchain systems.

B. Threat Model

Similar to other off-chain protocols, we assume the presence of an irrational adversary that can behave arbitrarily at any given time. The irrational adversarial may sustain financial losses in order to cause honest participants to lose some or all of their funds. Besides, it may take control of the leader, some or even all but one honest participant. The internal states, communication channels, and identities of the corrupted participants are fully exposed to the adversary. In addition, the adversary may send arbitrary messages on behalf of the corrupted party, launch a denial of service attack, etc. We assume that the communication channels among honest participants and their integrity of identities cannot be corrupted by the adversary. We also assume that the communication with the blockchain network are uncorrupted because nodes can participate in the blockchain network through any miner. Furthermore, we assume that the adversary is computationally bounded, i.e., corrupting cryptographic primitives is unfeasible for an adversary.

TABLE II
INFORMATION MAINTAINED BY LEADER

Notation	Description
$\mathcal{B}_i(e)$	Initial balance of $\mathcal{P}_i$ of epoch e
$\mathcal{S}_i(e)$	Total amount successfully sent by $\mathcal{P}_i$ during epoch e
$\tau_i(e)$	Total amount tried to be sent by $\mathcal{P}_i$ during epoch e
$\mathcal{R}_i(e)$	Total amount received by $\mathcal{P}_i$ during epoch e
$\mathcal{E}(e)$	Enrollment set of epoch e
$\mathcal{W}(e)$	Withdrawal set of epoch e

C. Overview

We now describe the key ideas of Garou and give a high-level overview on how Garou achieve both high-performance, flexibility, untrusty, and security.

Garou is composed of two fundamental building blocks: an on-chain smart contract and an off-chain transaction protocol. The on-chain smart contract manages the deposits of all participants of the payment hub and acts as a dispute resolver. It guarantees that the latest epoch state agreed by all participants is always accepted. Backed by the on-chain smart contract, an honest party will not bear any financial losses even when all other participants of the payment hub are malicious.

The off-chain transaction protocol is designed for concurrent and flexible off-chain coin transfers and consists of multiple epochs, each of which is divided into three phases: the leader election, transacting, and consensus phase. (1) For each epoch, a leader is elected out of all participants to drive the off-chain execution of the protocol. We adopt an automatic leader election mechanism to eliminate additional communication overhead while avoiding the single point of failure caused by a fixed leader. (2) During the transacting phase, participants of the payment hub can freely make coin transfers with each other. Garou does not require any intermediary nodes to route coin transfers or escrow extra coins before transactions, and therefore Garou exhibits a high degree of flexibility. Moreover, the spending and receiving of coins are separated, i.e., participants cannot spend more than their initial balance of the current epoch, and its receiving coins cannot be spent until the next epoch. This simplifies the verification and allows participants to perform concurrent transactions. (3) Finally, during the consensus phase, all participants collaborate to reach a consensus about the final state of the current epoch, which acts as a checkpoint for the off-chain execution of the Garou protocol. In case of disputes, an honest participant could commit the latest epoch state to the on-chain smart contract to enforce its balance security.

IV. GAROU PROTOCOL

A. Off-Chain Transaction Protocol

In this paper, we denote the set of participants as $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$, whereby all participants are ordered according to the time of joining.

1) Leader Election Phase: At the beginning of each epoch, a leader needs to be elected out of all participants to drive the off-chain execution. Specifically, the leader is responsible for:

- maintaining information as shown in Table II;

Authorized licensed use limited to: Wuhan University. Downloaded on September 09,2024 at 06:51:45 UTC from IEEE Xplore. Restrictions apply.

- issuing transaction id for every off-chain transaction in the current epoch;
- early verification of every transaction to prevent malicious participant from double-spending;
- driving all participants of the payment hub to collaborate to reach a consensus on the epoch state during the consensus phase;
- answering the epoch state challenge in case of disputes (discussed in Section IV-B).

To avoid the single point of failures caused by a fixed and malicious leader, we adopt a leader election mechanism to automatically elect a leader according to the balance information of the current epoch. More formally, the leader of epoch e is elected by:

$$l = Bcrypt\left(\bigoplus_{i=1}^{n} \mathcal{B}_i(e), salt, cost\right) \mod n \tag{1}$$

where *Bcrypt* [43] is a slow hash function, $\bigoplus$ denotes the XOR operation, $\mathcal{B}_i(e)$ denotes the initial balance of participant $\mathcal{P}_i$ in epoch e (as shown in Table II), *salt* is a 16-bytes random number, *cost* is a security parameter that defines the number of iterations. By this way, participant $\mathcal{P}_l$ will automatically become the leader of epoch e.

Bcrypt [43] is a password-hashing function based on the Blowfish cipher [44]. Besides incorporating a salt to protect against rainbow table attacks, Bcrypt is adaptive over time. The iteration count can be increased to make it slower, so it remains resistant to brute-force search attacks even with increasing computation power. By adjusting the security parameters of Bcrypt, we can control the calculation time of each hash. For participants, the election does not take much time. However, the attacker will spend much longer time than the epoch duration to make brute force attempts, which will prevent the leader from launching a collusion attack.

Garou requires that all participants should maintain knowledge about the initial balance set $\{\mathcal{B}_i(e)\}$ for every epoch, and therefore they can determine the leader by themselves without any additional communication or synchronization overhead, i.e., the leader election can be done at negligible cost.

2) Transacting Phase: During the transacting phase, participants within the payment hub are free to make coin transfers with each other, and all messages are conducted asynchronously.

We use the following tuple to denote an off-chain coin transfer:

$$tx = (e, txId, sender, receiver, x, y) \tag{2}$$

where *txId* denotes the transaction id issued by the leader of epoch e, x denotes the total amount of coins being spent by the sender $\mathcal{P}_i$, y denotes the amount being received by $\mathcal{P}_j$. The difference between x and y is the transaction fee for the leader (which must be zero or more). The transaction id is used for ordering transactions in the consensus phase.

As outlined in Fig. 1, to start an off-chain coin transfer, the sender $\mathcal{P}_i$ first requests a transaction id from the leader. When the leader receives the request for a new transaction id, it should check if the sender is trying to over-spend its balance.

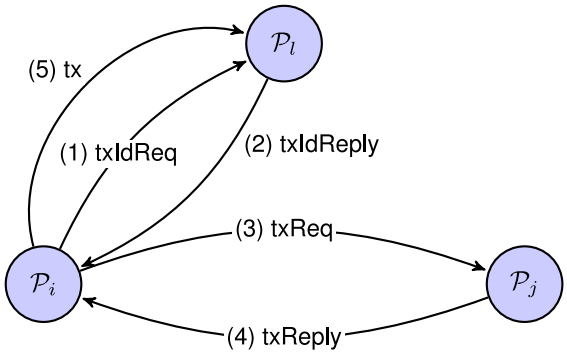


Fig. 1. Message Flow of an Off-Chain Transaction. $\mathcal{P}_l$ denotes the leader of the current epoch; $\mathcal{P}_i$ denotes the sender and $\mathcal{P}_j$ is the receiver of the transaction.

More formally, the leader should ensure the following holds:

$$x \leq B_i(e) - \tau_i(e) \tag{3}$$

If this is the case, the leader should approve this transaction, update the amount of coins that tried to be sent by $\mathcal{P}_i$ so that $\tau_i'(e) = \tau_i(e) + x$, and then reply with a monotonically increasing transaction id signed by the leader itself. After that, $\mathcal{P}_i$ sends a transaction request to the receiver $\mathcal{P}_j$, asking for his approval. When receiving the transaction request, $\mathcal{P}_j$ should verify the leader's signatures on the transaction id and then reply to $\mathcal{P}_i$ with its own signatures on the transaction request. Finally, $\mathcal{P}_i$ forwards the completed transaction request to the leader. And the leader will update its information table so that $\mathcal{S}_i'(e) = \mathcal{S}_i(e) + x$, $\mathcal{R}_j'(e) = \mathcal{R}_j(e) + y$ and $\mathcal{R}_l'(e) = \mathcal{R}_l(e) + x - y$.

Note that the off-chain transaction does not require any intermediaries to route or escrow their coins before transactions. The sender $\mathcal{P}_i$ could spend all its available balance according to its own will. From this perspective, Garou exhibits a high level of flexibility. From the equation (3), we also notice that when doing the early verification of a transaction, the leader does not take into account the amount of coins received by $\mathcal{P}_i$ in the epoch e, i.e., $\mathcal{R}_i(e)$. That is because of the separation of spending and receiving. Since an off-chain transaction cannot be completely confirmed before the consensus phase, spending those unconfirmed coins immediately in current epoch would increase the risk of double-spending. Therefore, Garou does not allow coins received in the current epoch to be spent until the next epoch. This limitation significantly simplifies the verification and enables us to perform concurrent transactions. As a result, $\mathcal{P}_i$ can initiate another transaction before he sends the result of the current transaction to the leader, as long as he does not try to spend more than his initial balance of the current epoch.

3) Consensus Phase: In the consensus phase, all participants should collaborate to reach a consensus about the final state of the current epoch, denoted as $State(e)$.

The epoch state $State(e)$ contains the following information:

$$State(e) = \big(e, \mathcal{B}'(e), \mathcal{M}(e), \mathcal{E}(e), \mathcal{W}(e)\big) \tag{4}$$

where $\mathcal{B}'(e)$ denotes the final balance set of all participants of the payment hub. $\mathcal{B}'(e)$ is equal to the initial balance set

Authorized licensed use limited to: Wuhan University. Downloaded on September 09,2024 at 06:51:45 UTC from IEEE Xplore. Restrictions apply.

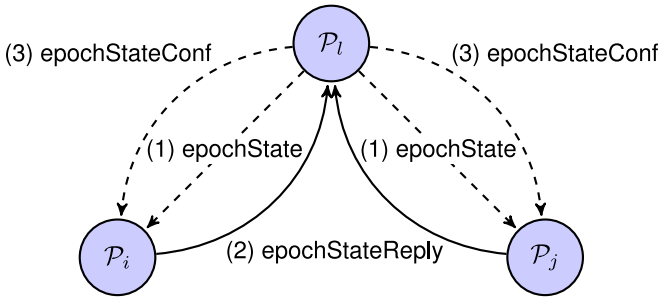


Fig. 2. Communication of the Consensus Phase. The solid line indicates unicast and the dashed line indicates broadcast.

of next epoch, namely $\mathcal{B}(e+1)$, and it is calculated by:

$$\mathcal{B}'_i(e) = \mathcal{B}_i(e) - \mathcal{S}_i(e) + \mathcal{R}_i(e), i \in \{1, \ldots, n\} \quad (5)$$

The transaction merkle tree root $\mathcal{M}(e)$ [45] is constructed for each participant $\mathcal{P}_i$ using all its involved transactions in epoch *e*, including both the spending and receiving transactions.

As shown in Fig. 2, the consensus phase is driven by the leader, who calculates the epoch state *State*$(e+1)$, collects participants' signatures on *State*$(e+1)$, and finally broadcasts the epoch state confirmation to all participants. When participant $\mathcal{P}_i$ receives the epoch state from the leader, it should verify its own balance $\mathcal{B}_i(e+1)$, the proper construction of its transaction merkle tree root $\mathcal{M}_i(e)$, and ensure that $\sum_{k=1}^{n} \mathcal{B}_k(e+1)$ equals to the total deposit of the payment hub. Each participant then replies to the leader with its signatures on the epoch state *State*$(e+1)$. Once all signatures are collected by the leader, an epoch state confirmation containing both *State*$(e+1)$ and all signatures will be broadcast to all participants to finish the current epoch. After that, participants of the payment hub can proceed to the next epoch.

If some participants have not received the epoch state checkpoint containing all the participants' signatures before the predefined timeout *T*, then any honest participant can open an on-chain epoch state challenge, which will be explained below.

B. On-Chain Dispute Resolver

As mentioned earlier, the on-chain smart contract is responsible for managing the deposits of all participants and acts as a dispute resolver for the payment hub. We now discuss how the on-chain smart contract resolves disputes.

Based on the system model described in Section III, our protocol treats the following scenarios as abnormal situations:

a) The leader is malicious. In the consensus phase, it tries to tamper with the final balance set $\mathcal{B}'(e)$ of the epoch state *State*(e).
b) The leader is malicious. In the consensus phase, it intentionally withholds the epoch state containing the signatures of all members to some participants, making it impossible for these participants to know the latest epoch state information.
c) The partial nodes (including the leader) are offline and fail to reach an agreement on the current epoch state information in the consensus phase.

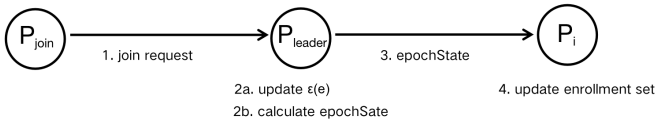


Fig. 3. The flowchart of participants' enrollment. $\mathcal{P}_{join}$ denotes the new participant of the current epoch.

d) Some participants are malicious. In the off-chain transacting phase, they try to launch a double-spending attack.

The above abnormal situations lead to the same result, that is, some participants have not received the epoch state checkpoint containing all the participants' signatures before the predefined timeout *T*. Therefore, if one of the above situation occurs, any honest participant can open an on-chain epoch state challenge.

The epoch state challenge enforces the latest epoch state agreed by all participants of the payment hub. The epoch state with the largest epoch id *e* is exactly the latest one. When a participant does not receive the epoch state confirmation upon timeout during the consensus phase, it will invoke the on-chain smart contract to open up an epoch state challenge. The answer to that state challenge, which consists of the latest epoch state and signatures from all participants, should be committed to the contract by the leader within a predefined timeout *T*. If the epoch state challenge is fulfilled, participants could proceed to the next epoch of off-chain execution. Otherwise, the payment hub should roll back to the last epoch state that is agreed upon by all its participants. In this case, participants can choose to proceed to the next epoch of the enforced one, or withdraw from the payment hub.

C. Dynamic Enrollment and Withdrawal

All Garou instance addresses and user account addresses are public on the Ethereum blockchain, and users can freely choose to join. Garou allows its participants to join in and withdraw from the payment hub at any time. Each participant can join in multiple Garou instances at the same time.

To join an existing payment hub (as shown in Fig. 3), (1) the new participant should first invoke the on-chain smart contract instance along with its initial deposit and a collateral. The collateral is set according to the gas cost of calling the dispute resolver, which will be presented in Section VI. The collateral is for everyone to share the cost evenly in the event of a dispute. Participants that deplete collateral will be forced to withdraw from the current protocol instance, and each participant needs to ensure that its own collateral is sufficient for the normal transaction. (2) Then in the next consensus phase, the leader should add the new participant into the enrollment set $\mathcal{E}(e)$ and send it to all participants along with the new epoch state, as shown in equation (4). The enrollment set $\mathcal{E}(e)$ contains those newly joined participants and their initial deposit on the payment hub. (3) After reaching a consensus, the newly joined participant can then perform off-chain transactions with each other in the next epoch.

Similar to the joining process, we use the withdrawal set $\mathcal{W}(e)$ as an off-chain synchronization tool (as shown in

Authorized licensed use limited to: Wuhan University. Downloaded on September 09,2024 at 06:51:45 UTC from IEEE Xplore. Restrictions apply.

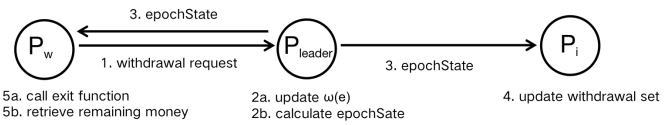


Fig. 4. The flowchart of participants' withdrawal. $\mathcal{P}_w$ denotes the participants who exit from the Garou of the current epoch.

Fig. 4). (1) To withdraw from a payment hub, the participant should first send a withdrawal request to the leader. (2) When the leader receives that withdrawal request, he adds the participant into the withdrawal set and broadcasts it to all participants along with the new epoch state. The withdrawal set $\mathcal{W}(e)$ contains those participants that are intended to withdraw and their final balance of epoch *e*. (3) After reaching a consensus, the participant intended to withdraw from the payment hub is not allowed to make any coin transfers with others. (4) The participant can now safely perform the exit process to reclaim its balance.

The exit process requires two on-chain transactions, one for calling the public exit function and one for retrieving the remaining funds. The two transactions should be separated by at least one confirmation time *T*, to prevent malicious nodes from the forced exit of the Garou instance without an agreement.

V. Security Analysis

In this section, we analyze liveness and the security properties of Garou Protocol. Garou guarantees that an honest participant following the prescribed protocol will not bear any financial losses despite strong adversarial capabilities, i.e., even when all other participants of the payment hub are malicious. The balance security of honest participants is enforced by the on-chain smart contract.

A. Liveness

Since in each epoch all participants should confirm the epoch state $State(e+1)$, Garou requires that all participants of the payment hub should always keep online, which is the same online requirement as NOCUST and Plasma. A malicious participant could launch denial-of-service attacks to hinder the proceeding of the Garou protocol.

More specifically, a malicious participant could (1) refuse to reply its signature on the epoch state $State(e+1)$ to the leader in the consensus phase, or (2) refuse to forward the completed transaction to the leader in the transacting phase. A malicious leader could (3) refuse to construct and broadcast the epoch state $State(e+1)$ in the consensus phase, or (4) refuse to approve the transaction id in the transacting phase. All the above cases will block the proceeding of the Garou protocol.

For the case (1), the epoch state $State(e+1)$ is correctly constructed but remains incomplete due to a lack of signatures of offline participants. In this case, the leader should commit the incomplete epoch state $State'(e+1)$ to the on-chain smart contract to enforce signatures. When an incomplete epoch state is submitted to the on-chain contract, participants that have not yet provided its signature need to submit their signature on $State'(e+1)$ to the on-chain contract within a maximum timeout *T*. Otherwise, those uncooperative participants will be evicted out by the on-chain dispute resolver.

For the case (2), in order to check whether there are completed transactions that are refused to forward, the leader should find out these participants through the missing transaction id, and then request the associated transaction senders and receivers to resend the 5th final message within a maximum timeout *T*. If both the sender and receiver do not reply the right 5th message, this specific transaction will be abandoned in the current epoch.

For the case (3), since only the leader maintains all the transaction information of the current epoch, it's infeasible to construct the correct epoch state $State(e+1)$ in the presence of a malicious leader. In this case, other participants should open up an epoch state challenge to roll back to the previous state.

For the case (4), it's quite trivial since one could wait until the next epoch to reapply for a transaction id. The automatic leader election mechanism can reduce the risk of a malicious leader and prevent the malicious leader from dominating the payment hub for a long time.

B. Safety

1) Balance Security: Garou guarantees that an honest participant will not bear any financial losses even when all other participants are malicious. Since the adversary cannot corrupt the identity of the honest participant, it is unfeasible to fabricate spending transactions in the name of honest participants. Therefore the only way to cause a financial loss on an honest participant is to tamper the balance on the incomplete epoch state. However, an honest participant always maintains knowledge about all its involved transactions and therefore is able to determine its final balance of the current epoch. When receiving a new epoch state from the leader in the consensus phase, an honest participant should verify the right amount of its balance and the right amount of total balance of the payment hub. Otherwise, an honest participant should refuse to reply with its signature on the epoch state that may lead to its financial loss. This will effectively prevent the payment hub from reaching a consensus, and the result is an on-chain epoch state challenge, which enforces the leader to provide a properly constructed epoch state.

2) Transaction Confirmation: Garou guarantees that if the leader is honest, all valid transactions of an honest participant are expected to be confirmed at the end of the current epoch. However, in the presence of a malicious leader, there is no guarantee that transactions will eventually be confirmed. A valid transaction consists of a transaction id signed by the leader, the right amount of coins, and signatures from both the sender and receiver. When constructing and verifying the transaction merkle tree root $\mathcal{M}(e)$, one should always ensure that all of its involved transactions, including both spending and receiving, are included and correctly ordered by the transaction id. Given an ordered set of transactions as its leaves, the constructed merkle tree root will be unique. In this way, a properly constructed merkle tree can ensure that all valid

Authorized licensed use limited to: Wuhan University. Downloaded on September 09,2024 at 06:51:45 UTC from IEEE Xplore. Restrictions apply.

transactions of an honest participant will be eventually confirmed. However, a malicious leader may choose to omit some of the valid transactions when constructing the merkle tree root. In this case, an honest participant should refuse to reply with its signature on the epoch state and resort to the on-chain smart contract to enforce a properly constructed epoch state.

In fact, the confirmation time of an off-chain transaction is exactly equal to the epoch duration, which is the same as NOCUST. And for this reason, Garou does not allow those unconfirmed coins to be spent until the next epoch.

3) Double Spending: The support for concurrent transactions will lead to a situation where a participant cannot accurately know the available balance of other participants. Therefore a malicious participant could have a chance to launch a double-spending attack by spending more than its available balance. The problem is quite trivial if the leader is honest. When issuing the transaction id, the leader should verify that a participant $\mathcal{P}_i$ should not spend more than its initial balance $\mathcal{B}_i(e)$ in the current epoch. If someone is trying to over-spend its balance, an honest leader should refuse to reply with transaction id, which effectively prevents the malicious participant from double-spending. However, if the leader happens to be malicious, the situation becomes more complicated. In this case, a malicious participant will manage to get a transaction id from the leader and construct a spending transaction that overspends its balance to an honest receiver. However, an over-spending transaction will eventually cause the total balance of the payment hub to deviate from its correct value b, more specifically, $\sum_{i=1}^{n} \mathcal{B}_i(e) > b$. Therefore, an honest participant should verify the right amount of the payment hub and ensure that $\sum_{i=1}^{n} \mathcal{B}_i(e) \equiv b$ holds during the consensus phase. Otherwise, it should refuse to provide its signature on the epoch state and open an on-chain state challenge to enforce its balance security. In the worst case, the payment hub should roll back to the last epoch state that is agreed upon by all its participants.

VI. Implementation

To evaluate the effectiveness and performance of our design, we implement the Garou protocol for the Ethereum network. Then we measure the execution cost, evaluate the off-chain performance of the Garou protocol, and compare it with the state-of-the-art payment hubs.

A. Execution Cost

Ethereum uses gas to measure the amount of computational and storage resources cost by the execution of smart contracts. The gas cost of the dispute resolver contract can be calculated using smart contract development tools, such as the official Ethereum Remix. Every instruction executed by the *Ethereum Virtual Machine* (EVM) costs a certain amount of gas, e.g., verifying a user's signature costs 3000 gas units [42]. Table III shows the execution costs and off-chain message complexity of each operation of the Garou protocol. Apart from the joining and withdrawal process, if all participants are honest and cooperative, the off-chain coin transfers and consensus do not require any involvements of the blockchain, resulting in zero on-chain transaction and gas cost. Every off-chain coin transfer costs a constant five messages, and the message complexity of the consensus phase is linear to the number of participants. In the presence of malicious parties, no matter a malicious leader or ordinary participants, $1+m$ on-chain transactions are needed to resolve disputes: one for opening an epoch state challenge, and m for each malicious party answering the state challenge. The gas cost of answering an epoch state challenge mainly consists of verifying participants' signatures on the epoch state, and the cost used to commit the epoch state data to the smart contract.

TABLE III
Execution Cost and Message Complexity of Garou

Operation	On-Chain Tx	Gas Cost	Off-Chain Msg
Join	1	43211	0
Withdraw	2	100166	0
Transfer	0	0	5
Consensus (optimistic)	0	0	$3n$
Consensus (pessimistic)	$1+m$	$10962n$	-

n: the number of participants of payment hub
m: the number of malicous participants

In Ethereum, the gas limit refers to the upper bound of gas that can be consumed by executing a smart contract. As more participants join in the payment hub, more information and signatures will need to be collected and verified by the on-chain smart contract when resolving disputes. Therefore, the number of participants of a payment hub cannot grow infinitely. In our implementation, the cost of resolving disputes increases by approximately 10962 gas units per node (as shown in Table III), and smart contracts have a block gas limit of about 3.14 million, leading to the maximum number of participants being approximately 300 for each payment hub. However, as mentioned in Section III-A1, the blockchain network allows multiple independent contract instances to be running at the same time. Therefore we could scale out the off-chain system by instantiating multiple payment hubs.

B. Performance Evaluation

We now evaluate the off-chain transaction performance of the Garou protocol and compare it with NOCUST. We implement Garou's participants and leader code in Golang 1.13.4, and also implement Dispute Resolver in Solidity 0.4.24. Our evaluations are run on a machine with Intel Core i5-8400 2.80GHz CPU and 32G RAM. We deploy at most 300 nodes to simulate an Off-Chain network. The bandwidth of all connections between participants is set to 20Mbps and a latency of 100 $\pm$ 20ms is imposed on all communication links. Our subsequent evaluations are conducted on one single payment hub, i.e., we open a payment hub instance, let participants join in the payment hub, and then measure the off-chain transaction throughput and latency of that payment hub instance.

To effectively measure the maximum off-chain performance, we need to assume that all participants of the payment hub are

Authorized licensed use limited to: Wuhan University. Downloaded on September 09,2024 at 06:51:45 UTC from IEEE Xplore. Restrictions apply.

honest and cooperative. Otherwise, in the presence of malicious or unresponsive participants, the honest one should pause the off-chain processes and invoke the on-chain smart contract, which will lead to an unpredictable delay, depending on the behavior of malicious participants and the delay of the blockchain networks. Since there is no realistic off-chain transaction dataset, and the available on-chain transaction datasets cannot meet our concurrency requirement. Therefore we use randomly generated transaction dataset in our evaluation.

1) Transaction Throughput: Firstly, we measure the maximum transaction throughput under different combinations of the number of nodes and epoch duration. Specifically, we vary the number of nodes from 10 to 300. In addition, the epoch duration is closely related to the confirmation time of an off-chain transaction for both Garou and NOCUST. We do not expect a long confirmation time and therefore vary the epoch duration from 2 seconds to 60 seconds. Fig. 5 summaries the maximum off-chain transaction throughput of Garou and NOCUST respectively. We do not measure the transaction throughput of the multi-party virtual state channel [46] since its upper limit is quite predictable. It requires that any state update should be explicitly agreed upon by all participants. Therefore, given that the network latency is about 100 ms, its theoretical transaction throughput cannot exceed 10.

From the results of our evaluation, we notice that the off-chain transaction throughput of Garou is much higher than NOCUST. Garou obtains quite high transaction throughput when the epoch duration is set to 10 seconds. In this case, Garou exhibits nearly 20 $\times$ the maximum throughput of NOCUST. However, the transaction throughput of these two payment hubs demonstrates different growth trends. Garou's transaction throughput increases with the length of the epoch duration and does not increase with the number of participants, but it is just the opposite for NOCUST. This can be explained as below.

Garou's transaction throughput is mainly governed by the leader because every off-chain coin transfer requires a transaction id issued by the leader. Therefore, the increase in the number of participants will not lead to an increase in transaction throughput. On the contrary, the shortening of epoch duration will cause the leader to spend a larger proportion of time in the consensus phase, and therefore reduces the overall transaction throughput of the payment hub. This is quite obvious when the epoch duration is relatively short, e.g., 2s. On the contrary, NOCUST does not allow both the sender and receiver to engage in any other transfers until the current one is settled, which significantly limits the concurrency and the transaction throughput of the payment hub. As a result, the maximum transaction throughput of NOCUST is much worse than Garou. The only way to increase the concurrency of NOCUST, as shown in Fig. 5(b), is letting more pair of nodes to make transfers simultaneously. However, as mentioned above, the number of participants cannot be increased infinitely due to the gas limit.

As shown in Fig. 5(a), the off-chain transaction throughput of Garou protocol increases significantly when the epoch duration is between 2 and 10 seconds. When the epoch duration exceeds 10 seconds, the growth of transaction throughput

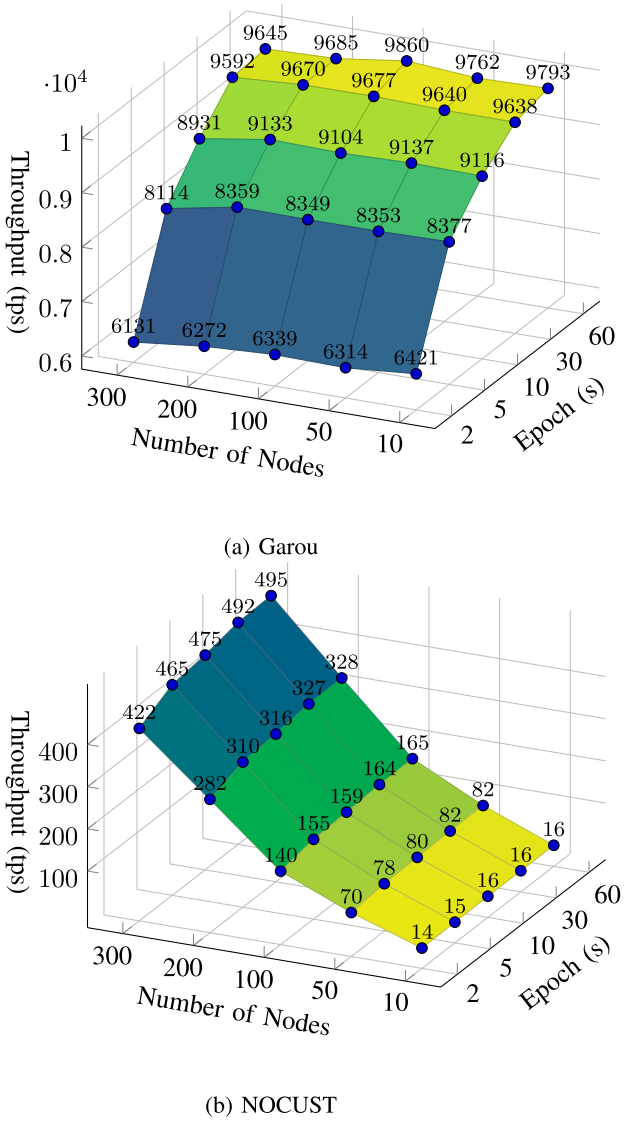


Fig. 5. Impact of different number of nodes and epoch duration on maximum off-chain transaction throughput.

slows down. Moreover, a longer epoch duration means longer transaction confirmation time. Therefore, to achieve a balance between transaction throughput and confirmation time, we set the epoch duration to 10 seconds in the following evaluations.

2) Transaction Execution Time: Fig. 6 shows the average off-chain transaction execution time of both Garou and NOCUST. Transaction execution time refers to the time interval between sender's sending a txidReq message and the leader's final reception for an off-chain transaction in the transacting phase. As expected, the concurrency limitation of NOCUST leads to its smooth latency curve. In contrast, the transaction execution time of Garou remains stable at about 500ms when the number of participants does not exceed 100. This is because a fund transaction under this agreement requires five rounds of communication to complete. When the average network delay is 100ms, the average transaction delay of the protocol is about 500ms. When it does exceed 100, the transaction execution time starts to increase gradually. The

Authorized licensed use limited to: Wuhan University. Downloaded on September 09,2024 at 06:51:45 UTC from IEEE Xplore. Restrictions apply.

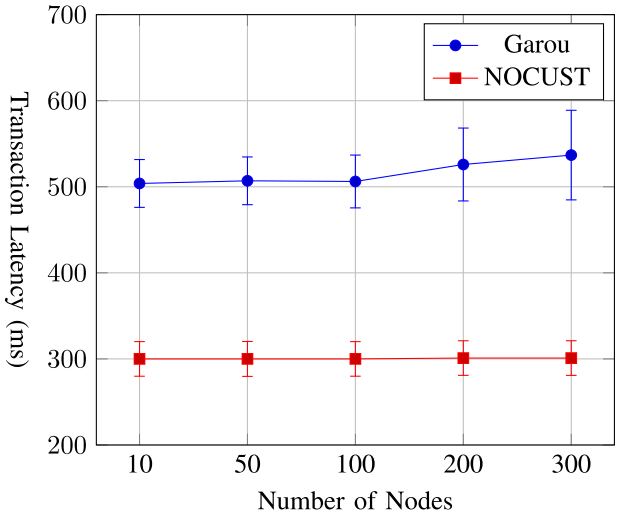


Fig. 6. Average transaction execution time.

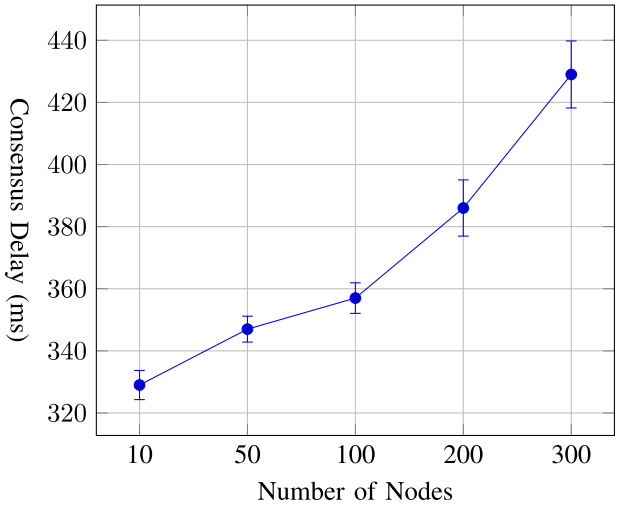


Fig. 7. Consensus Delay of Garou.

main reason is the queuing phenomenon that occurs when the master node issues transaction ids for the fund transactions of member nodes. Nevertheless, even when the number of participants reaches the upper limit of 300, the average transaction execution time still does not exceed 1 second, which is quite acceptable. We also notice that the average transaction execution time of Garou is higher than NOCUST. This is because Garou's off-chain transactions require more communication between the leader and participants to prevent malicious nodes from double-spending in concurrent scenarios. At the cost in transaction execution time, Garou obtains an order of magnitude improvement in throughput.

3) Consensus Delay: Fig. 7 shows the consensus delay of Garou protocol, i.e., the time spent in the consensus phase for each epoch. Since NOCUST does not have a consensus phase, we only test the consensus delay of our proposed protocol. During the consensus phase, all participants of the payment hub are not allowed to make any coin transfers until entering the next epoch. Similar to transaction execution time, the consensus delay starts to increase significantly when the number of participants exceeds 100. This is simply because the leader needs to communicate with all participants of the payment hub and collect their signatures in the consensus phase.

4) Summary of Results: In summary, the performance evaluations show that Garou has achieved a substantial improvement in transaction throughput than the state-of-the-art payment hubs. More precisely, the maximum transaction throughput of Garou is 20× higher than NOCUST. Roughly, Garou's high performance mainly benefits from its support for concurrent off-chain transactions. Moreover, when the number of participants is within 100 and the epoch duration is set to 10 seconds, Garou can achieve a balance in both transaction throughput, latency, and consensus delay. Since there is no limitation on the number of Garou instances, our proposed protocol could dramatically improve the performance and scalability of the blockchain systems.

VII. Conclusion

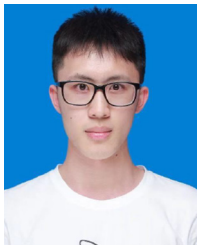
Currently, the off-blockchain payment system is still far from maturity and exhibits many challenges in terms of performance, scalability, privacy, and security. In this paper, we present Garou, a high-performance and secure off-chain N-party payment hub that allows multiple parties to perform concurrent and flexible off-chain coin transfers with each other without compromising security. The support for concurrent transaction significantly increases the performance of the payment hub and allows Garou to achieve up to 20× the maximum transaction throughput of the state-of-the-art payment hubs. Transactions within the payment hub can be executed directly between participants without the need of any third-party operators. Garou also guarantees the balance security of honest participants despite strong adversarial capabilities. Overall, our proposed Garou protocol achieves a balance in both performance, flexibility, and security and therefore serves as an enrichment to the off-blockchain payment systems.

References

[1] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: https://bitcoin.org/bitcoin.pdf
[2] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail," in *Proc. 12th Annu. Int. Cryptol. Conf. Adv. Cryptol. (CRYPTO)*, 1992, pp. 139–147.
[3] S. King and S. Nadal. (2012). *PPCoin: Peer-to-Peer CryptoCurrency With Proof-of-Stake*. [Online]. Available: https://cdn.bitturk.com/whitepaper/ppc.pdf
[4] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum, Zug, Switzerland, Yellow Paper, 2014.
[5] K. Croman *et al.*, "On scaling decentralized blockchains," in *Proc. Int. Conf. Financ. Cryptography Data Security*, 2016, pp. 106–125.
[6] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2016, pp. 3–16.
[7] M. Trillo. (2013). *Stress Test Prepares Visanet for the Most Wonderful Time of the Year (2013)*. [Online]. Available: https://www.visa.com/blogarchives/us/2013/10/10/stress-test-prepares-visanet-for-the-most-wonderful-time-of-the-year/index.html
[8] C. Decker and R. Wattenhofer, "A fast and scalable payment network with bitcoin duplex micropayment channels," in *Proc. 17th Int. Symp. Stabilization Safety Security Distribut. Syst.*, vol. 9212, 2015, pp. 3–18.
[9] J. Poon and T. Dryja. (2016). *The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments*. [Online]. Available: https://www.bitcoinlightning.com/wp-content/uploads/2018/03/lightning-network-paper.pdf

Authorized licensed use limited to: Wuhan University. Downloaded on September 09,2024 at 06:51:45 UTC from IEEE Xplore. Restrictions apply.

[10] G. Malavolta, P. Moreno-Sanchez, A. Kate, M. Maffei, and S. Ravi, "Concurrency and privacy with payment-channel networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2017, pp. 455–471.
[11] C. Burchert, C. Decker, and R. Wattenhofer, "Scalable funding of bitcoin micropayment channel networks," *Roy. Soc. Open Sci.*, vol. 5, no. 8, 2018, Art. no. 180089.
[12] A. R. Pedrosa, M. Potop-Butucaru, and S. Tucci-Piergiovanni, "Scalable lightning factories for bitcoin," in *Proc. 34th ACM/SIGAPP Symp. Appl. Comput.*, 2019, pp. 302–309.
[13] H. Ethan, F. Baldimtsi, L. Alshenibr, A. Scafuro, and S. Goldberg, "TumbleBit: An untrusted tumbler for bitcoin-compatible anonymous payments," in *Proc. Netw. Distrib. Syst. Security Symp. (NDSS)*, 2017, pp. 1–9.
[14] E. Tairi, P. Moreno-Sanchez, and M. Maffei, "A2L: Anonymous atomic locks for scalability in payment channel hubs," Cryptology ePrint Archive, Rep. 2019/589, 2019. [Online]. Available: https://eprint.iacr.org/2019/589
[15] S. Dziembowski, S. Faust, and K. Hostáková, "General state channel networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security (CCS)*, 2018, pp. 949–966. [Online]. Available: http://doi.acm.org/10.1145/3243734.3243856
[16] S. Dziembowski, L. Eckey, S. Faust, and D. Malinowski, "PERUN: Virtual payment hubs over cryptocurrencies," in *Proc. IEEE Symp. Security Privacy (SP)*, 2019, pp. 106–123.
[17] R. Khalil and A. Gervais, "Nocust—A non-custodial 2nd-layer financial intermediary," in *Proc. IACR Cryptol. ePrint Archive*, 2018, p. 642.
[18] J. Poon and V. Buterin, "Plasma: Scalable autonomous smart contracts," White Paper, pp. 1–47, 2017. [Online]. Available: https://plasma.io/plasma.pdf
[19] I. Eyal, A. E. Gencer, E. G. Sirer, and R. van Renesse, "Bitcoin-NG: A scalable blockchain protocol," in *Proc. USENIX*, 2016, pp. 45–59.
[20] L. Luu, V. Narayanan, K. Baweja, C. Zheng, S. Gilbert, and P. Saxena, "SCP: A computationally-scalable Byzantine consensus protocol for blockchains," in *Proc. IACR Cryptol. ePrint Archive*, 2015, p. 1168.
[21] R. Pass and E. Shi, "Hybrid consensus: Efficient consensus in the permissionless model," in *Proc. Int. Conf. Distrib. Comput.*, vol. 91, 2017, p. 16.
[22] J. Lind, I. Eyal, F. Kelbert, O. Naor, P. Pietzuch, and E. G. Sirer, "Teechain: Scalable blockchain payments using trusted execution environments," 2017. [Online]. Available: arXiv:1707.05454
[23] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, "Town crier: An authenticated data feed for smart contracts," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2016, pp. 270–282.
[24] M. Zamani, M. Movahedi, and M. Raykova, "Rapidchain: Scaling blockchain via full sharding," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security (CCS)*, 2018, pp. 931–948.
[25] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2016, pp. 17–30.
[26] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "OmniLedger: A secure, scale-out, decentralized ledger via sharding," in *Proc. IEEE Symp. Security Privacy (SP)*, 2018, pp. 583–598.
[27] L. N. Nguyen, T. D. Nguyen, T. N. Dinh, and M. T. Thai, "OptChain: Optimal transactions placement for scalable blockchain sharding," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2019, pp. 525–535.
[28] V. Buterin. *On-Chain Scaling to Potentially 500 TX/SEC Through Mass TX Validation*. Accessed: Jul. 23, 2021. [Online]. Available: https://ethresear.ch/t/on-chain-scaling-to-potentially-500-tx-sec-through-mass-tx-validation/3477
[29] P. McCorry, M. Möser, S. F. Shahandasti, and F. Hao, "Towards bitcoin payment networks," in *Proc. Aust. Conf. Inf. Security Privacy*, 2016, pp. 57–76.
[30] J. Wang and H. Wang, "Monoxide: Scale out blockchains with asynchronous consensus zones," in *Proc. 16th {USENIX} Symp. Netw. Syst. Design Implement. (NSDI)*, 2019, pp. 95–112.
[31] A. Chauhan, O. P. Malviya, M. Verma, and T. S. Mor, "Blockchain and scalability," in *Proc. IEEE Int. Conf. Softw. Qual. Rel. Security Companion (QRS-C)*, 2018, pp. 122–128.
[32] K. Floersch. (2019). *Ethereum Smart Contracts in L2: Optimistic Rollup*. [Online]. Available: https://medium.com/plasma-group/ethereum-smart-contracts-in-l2-optimistic-rollup-2c1cef2ec537
[33] A. Gluchowski. (2019). *ZK Rollup: Scaling With Zero-Knowledge Proofs*. Accessed: Jul. 23, 2021. [Online]. Available: https://pandax-statics.oss-cn-shenzhen.aliyuncs.com/statics/1221235526992813.pdf
[34] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, "Succinct non-interactive zero knowledge for a von neumann architecture," in *Proc. 23rd {USENIX} Security Symp. ({USENIX} Security)*, 2014, pp. 781–796.
[35] C. Decker, R. Russell, and O. Osuntokun, "eltoo: A simple layer2 protocol for bitcoin," White Paper, 2018. [Online]. Available: https://blockstream.com/eltoo.pdf
[36] J. Khamis and O. Rottenstreich, "Demand-aware channel topologies for off-chain payments," in *Proc. IEEE Int. Conf. Commun. Syst. Netw. (COMSNETS)*, 2021, pp. 272–280.
[37] J. Khamis, S. Schmid, and O. Rottenstreich, "Demand matrix optimization for offchain payments in blockchain," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency (ICBC)*, 2021, pp. 1–9.
[38] G. Malavolta, P. Moreno-Sanchez, C. Schneidewind, A. Kate, and M. Maffei, "Anonymous multi-hop locks for blockchain scalability and interoperability," in *Proc. NDSS*, 2019, pp. 1–30.
[39] P. Prihodko, S. Zhigulin, M. Sahno, A. Ostrovskiy, and O. Osuntokun. (2016). *Flare: An Approach to Routing in Lightning Network*. [Online]. Available: https://bitfury.com/content/downloads/whitepaper_flare_an_approach_to_routing_in_lightning_network_7_7_2016.pdf
[40] G. Malavolta, P. Moreno-Sanchez, A. Kate, and M. Maffei, "Silentwhispers: Enforcing security and privacy in decentralized credit networks," in *Proc. NDSS*, 2017.
[41] S. Roos, P. Moreno-Sanchez, A. Kate, and I. Goldberg, "Settling payments fast and private: Efficient decentralized routing for path-based transactions," in *Proc. Netw. Distrib. Syst. Security Symp.*, 2018, pp. 66–92.
[42] V. Buterin *et al.* (2014). *Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform*. [Online]. Available: https://ethereum.org/en/whitepaper/
[43] N. Provos and D. Mazieres, "A future-adaptable password scheme," in *Proc. USENIX Ann. Tech. Conf.*, 1999, pp. 81–91.
[44] B. Schneier, "Description of a new variable-length key, 64-bit block cipher (blowfish)," in *Proc. Int. Workshop Fast Softw. Encrypt.*, 1993, pp. 191–204.
[45] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Proc. Conf. Theory Appl. Cryptograph. Techn.*, 1987, pp. 369–378.
[46] S. Dziembowski, L. Eckey, S. Faust, J. Hesse, and K. Hostáková, "Multi-party virtual state channels," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2019, pp. 625–656.



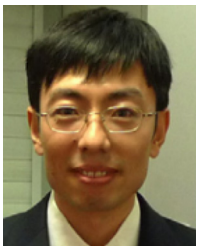
Yongjie Ye received the M.Sc. degree from the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China, in 2020. He is currently with Alibaba. His research interests include distributed systems and cloud computing systems.

Zhifeng Ren received the B.E. degree from the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China, in 2019, where he is currently pursuing the M.Sc. degree. His research interests include distributed systems and blockchain.

Authorized licensed use limited to: Wuhan University. Downloaded on September 09,2024 at 06:51:45 UTC from IEEE Xplore. Restrictions apply.

Xiapu Luo received the Ph.D. degree in computer science from the Hong Kong Polytechnic University in 2007, and was a Postdoctoral Research Fellow with the Georgia Institute of Technology. He is currently an Associate Professor with the Department of Computing and an Associate Researcher with the Shenzhen Research Institute, the Hong Kong Polytechnic University. His current research focuses on smart phone security and privacy, network security and privacy, and Internet measurement.

Jingjing Zhang received the M.Sc. degree from the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China, in 2017, where she is currently pursuing the Ph.D. degree. Her research interests include distributed systems and blockchain.



Weigang Wu (Member, IEEE) received the B.Sc. and M.Sc. degrees from Xi'an Jiaotong University, China, in 1998 and 2003, respectively, and the Ph.D. degree in computer science from Hong Kong Polytechnic University in 2007. He is currently a Full Professor with the School of Computer Science and Engineering, Sun Yat-sen University, China. He has published more than 100 papers in major conferences and journals. His research interests include distributed systems, cloud computing and big data processing. He has served as an organizing/program committee member for many international conferences.

Authorized licensed use limited to: Wuhan University. Downloaded on September 09,2024 at 06:51:45 UTC from IEEE Xplore. Restrictions apply.