OClient: On-board client clustering to bridge the interaction gap across multi-fabric adaptive ecosystem

Ankur Lohachab, Saurabh Garg, Jianjing Guo, and Muhammad Bilal Amin

Abstract—The current blockchain paradigm is being hailed as an unquenchable goal to develop Hyperledger Fabric (HF)-based applications, resulting in a set of substantial enterprises shifting their networking operations, as intriguing consortia to plug the permissioned-nature of blockchain-based computing. While this practice endeavors to fundamentally revolutionize industries, one of the unavoidable problems that emerge is a lack of communication among autonomous HF-based networks, which can hinder the ongoing research landscape and consequently, have detrimental effects on its adoption. Towards this, blockchain interoperability (BI) has been heralded as the underlying concept by which communication can be derived through locating an inter-connection bridge. Considering BI's significant traction, this paper introduces a new methodological approach to deal with the disparate nature of HF-based networks, termed OClient, for embracing two different interoperabilities (inter-network and inter-chain). This approach strategically tunes the inner working of clients by providing the global information of HF network topology that extends their conventional role as a matching service provider for facilitating opportunistic communication, in contrast to the existing solutions where typically third-parties have been exploited as the intermediaries. The paper streamlines the discussion by putting forth a pragmatic multi-HF (MHF) network scenario to formulate the disconnectivity problem, followed by constructing interaction-path-based graphs and conducting rigorous simulations. More importantly, this paper manifests experimental examples and corresponding results, providing insights across various dimensions of client-level interoperability so as to understand the impact on performance for combinations of inter-network and inter-chain scenarios. Succinctly, the results demonstrate that OClient is close to the performance of autonomous HF networks, which makes it a useful approach. Besides, this paper strives to understand how the adopted failure strategy to maintain ledger consistency contributes to performance and finally, concludes by anticipating significant future research directions towards HF interoperability.

Index Terms—Blockchain, Blockchain Interoperability, Hyperledger Fabric, Autonomous Vehicles.

1 INTRODUCTION

THE interoperability among blockchain systems has emerged as the primary domain of research in the blockchain development landscape. One of the reasons behind this is that Blockchain Interoperability (BI) is widely considered as a fundamental challenge in different perspectives (i.e., public, private, and permissioned) to implement blockchain systems [1]. Beyond this, one can anticipate that applications developed based on different underlying blockchain systems, ushering in the era of Distributed Ledger Technology (DLT), will assuredly confront a disconnectivity problem. Quite surprisingly, at first glance, the state-of-the-art BI solutions seem to address the challenge; however, practically they are introducing more complexity in the way of defining or determining BI. It is broadly attributed to two reasons: firstly, the window of interoperability solutions is relatively small because of multifaceted challenges, such as different ledger implementations, governance models, incentive mechanisms, and hashing techniques. Secondly, the existing solutions firmly lack the notion of decentralized (e.g., relatively less autocratic) BI.

However, amid the BI research, businesses creating Applications of Value (AoV) typically form consortia, such as We.Trade, BankChain, Marco Polo [2], for facilitating blockchain features (e.g., transparency, immutability, security) to the stakeholders. It advocates that obtaining “private communication” among the network’s stakeholders, specifically from the business’s perspective, is of pre-eminent importance, as it determines that the underlying network can be provisioned for confidential transactions. In such a scenario, Hyperledger Fabric (HF) emerges as a natural solution, as the notion of ‘channel’ (i.e., private subnet) in HF provides flexibility, where member stakeholders (e.g., organizations) can define the channel by calling the configuration system chaincode. For example, our recent work [3] manifests that different stakeholders can effectively utilize the capabilities of HF in a Peer-to-Peer (P2P) energy trading scenario. Moreover, real-world applications augmented by DLT Labs, Honeywell Aerospace, Walmart, Sony Global Education’s [4] popularly foster various enterprises (i.e., specifically that were skeptical about blockchain earlier) to safeguard their businesses’ interests in the course of the right-to-be-transparent by encompassing HF-based blockchain.

Despite its large-scale adoption, the fundamental question remains the same; *Does HF-based blockchain supports BI?* We argue this question through exemplifying a Multi-HF (MHF) scenario (depicted in Figure 1). Suppose End-Customers (EC) own Intelligent and Autonomous (I&V)

- Ankur Lohachab, Saurabh Garg, Jianjing Guo, and Muhammad Bilal Amin are with School of Information and Communication Technology, University of Tasmania, Sandy Bay Campus, Hobart, Australia, 7005. Corresponding author e-mail: ankur.lohachab@utas.edu.au

Manuscript received September 10, 2021; revised xx.

vehicles that face a hazardous situation (e.g., foggy and slippery road), resulting in an unfortunate accident (more likely due to no line-of-sight). This is where MHF kicks in: EC acquaints the liable Insurance Company (IC) regarding the incident. Now, IC being a client of HF 1 interacts with Manufacturing Company (MC) through Software Development Kit (SDK). From there, let's say MC application requires to interact with Servicing Company (SC) (e.g., operated on HF 2) and Procurement Company (PC) (e.g., operated on HF 3) networks to acquire service details and warranty/components' specifications, respectively. It is to be noted that a given MHF network scenario may contain any number of HFs, where each HF can consist of any number of channels and peer nodes. From BI viewpoint, in this scenario any EC should be able to accomplish (or meet) the following services (or properties): (i) link assets (i.e., one can create an asset on one HF network and transfer or swap on other HF networks), (ii) atomicity (i.e., cross-chain transactions should be considered as a single action), (iii) verifiability (i.e., should be able to verify on HF 2 that asset A_i with value V_j on HF 1 exists). However, the existing HF working mechanism fails short to suffice most of the mentioned services, which results in lacking a functional approach to adopt BI. In other words, can the MC application be able to span/interact with the SC and PC HF networks. Currently, there are no such guidelines for allowing such a scenario (i.e., that enable interaction among MHF).

Figure 1: The disconnectivity problem (i.e., (MHF Network)-BI Problem) is manifested through an example practical scenario

Thus, towards manifesting BI for the MHF network environment, this paper introduces a collaborative approach termed as OClient, which not only addresses the above ser-

vices from the HF perspective, but employs the existing HF in a manner so that no additional development is needed. This simply means that OClient mechanism is implemented in a way that does not conflict with the cloverleaf of the underlying principles of HF and BI. *Can BI be purposively designed and constructed in HF? If yes, then how?* To answer this, consider clients as the architects of the bridge that enable BI. One can argue that: *why only client coordination; Is there no better way to enable BI among HF-based systems?* Interestingly, besides the reason that it is simpler and requires minimal modification to the applications that are implemented on top of different HF versions, it is noted that an alternative to the client applications that can enable interoperability is peer nodes. However, the preliminary informal investigation suggests that it may be one of the important actors, but within its direct and obvious practical capacity. First of all, finding relevant peer nodes in itself is a sophisticated task (as there are different roles of peer nodes), and if somehow the peer node type is determined, another drawback may be derived from its processing capabilities (e.g., peer pod processing limitations) and other factors (e.g., data residency rules). Thus, this may make the hypothesis of choosing peer nodes a complex approach.

Figure 2: An abstract view of the work structure in the present context

Because of such potential complexities, this paper reasons that peer nodes may not be an appropriate foundation, either for achieving our goals or working towards introducing an easy-to-adopt approach with better features in a practical scenario. To shed more light on choosing clients as “matching service”, let's address one more question, which will circumvent a certain degree of subjectivity while underlies our argument. Why the selection of clients is not relevant? Here, this paper goes one level back to revisit the argument that going from client-level interoperability to other HF components cooperation assumption is a considerable leap in bypassing complexity, while clinging “interaction” among fragmented HF networks. Furthermore, we reinterpret one of the design goals described in [5], which primarily focus on *understandability*. The result of this is that concerning practical systems, it is not significant how sophisticated the approach should be, but could we present an approach that is much understandable and easier to implement, is of much significance. Therefore, at this point, conceptually, this paper does not encounter an intuitive drawback for not selecting clients as the key ingredients of our approach design. Thus, if one wants to argue about the importance or feasibility of OClient, this paper sets to conduct an in-depth investigation to sketch various relevant research questions that need to be addressed at a range of points in design-level of the proposed approach. The reasoning and opportunities that this paper puts forward while addressing such questions, create an initial research frame that not only provides convincing arguments from

an experimental evaluation, but will implicitly open to serendipity. In context of such reasons, overall, this paper aims to practically test the research hypothesis feasibility and how much the underlying environment gets affected by which OClient is adopted (e.g., what it offers and how it behaves). It makes the underlying methodology categorized into three broader stages; a high-level representation is illustrated in Figure 2. Nonetheless, the following sub-section outlines the contributions, a bit pointed, but still broadly characterizes the insights provided in this paper.

Major Contributions To the best of our knowledge, this paper is the first attempt to address the (MHF Network)-BI Problem using an approach that exclusively utilizes one of the HF's components to construct a scalable and upgradable bridge from HF network-to-HF network and HF channel-to-HF channel. Within this approach, the key contributions presented in this paper are summarized below -

- It introduces the notion of MHF interoperability by refining the general BI problem within the mechanism to achieve inter-network and inter-chain interoperability in HF-based systems. Here, the objective is to outline unreachability stemmed from the disparate HF networks in both inter-network and inter-chain scenarios, which may exacerbate as a collection of the isolated archipelago of HF-based applications.
- We revisit the working mechanism and capabilities of the state-of-art BI techniques to uncover an underlying implementation approach that can be leveraged to derive the potential solution conforming to the MHF interoperability viewpoint. It typically mounts a rigorous underlying approach whose anticipated outcome can handle the HF's dynamics under heterogeneous network scenarios and the number of channels.
- This paper formalizes the (MHF Network)-BI Problem that beholds the subsumptions between theory and practicality, while formulating conceptual graphs in the sense of capturing the possible interaction pathways. Here, the objective is to provide graphical representations that can serve as generalized pathways through which SCs relate disparate HF networks by invoking each other.
- This paper provides a series of experimentation to investigate the impact on the correlated metrics, such as the number of networks/channels/clients on performance, where interoperability has been enabled by utilizing OClient. Moving a step forward, to demonstrate the OClient's pragmatic ability, these experiments are in-line with the autonomous vehicle use case, which incorporates the sophisticated interactions among stakeholders under the diverse conditions of inter-network and inter-chain HF interoperability.
- This paper's expository failure rate arguments assist in identifying how much ledger-consistency maintaining mechanism contributes to affecting performance index, and are there elusive ledger inconsistencies induced even after employing this mechanism. It will provide a plausible basis for estimating the consistency-induced delay, given the different

degrees of failure rates. At the same time, this analytics can extend towards a realistic evaluation of consistency correctness, which may be of particular importance when it comes to characterizing the atomicity, in case, where an MHF transaction "in transit" cannot be completed and its initial operations cannot be undone (or irreversible).

Approach and Paper Organization To aim for the inclusion of the these contributions along with other derived benefits, the rest of the paper is organized in the following way. It begins with Section 2, briefly describing a few recent works that drive forward the main underlying directions of BI. Section 3 illuminates how the (MHF Network)-BI Problem has been modeled, stating the assumptions that underlie the definitions, which are referred to formulate the overall problem and determine the research roadmap. Section 3 also discusses the interaction path possibilities for this problem using a prescribed connected graph, which holds for the general objective of devising dependencies that exist in fragmented HF network instances. Section 4 presents the working mechanism of the proposed approach, OClient, and also identifies the underlying key design principles that certain insights to characterize requirements for OClient. Thenceforth, Section 5 manifests the case study design that is accountable for the experimental evaluation, under which the instances of higher interest based on inter-network and inter-chain HF interoperability, diversify the hardheaded evaluations. Subsequently, Section 5 describes the considered failure patterns that can potentially bring arbitrary deterioration in the ledger consistency. Section 6 outlines the underlying experimental methodology, and then, presents the obtained results under numerous experimentation settings by exerting the autonomous vehicle use case as an underlying example. Moreover, this section discusses the reasons that may be responsible for some of the achieved results, followed by contemplating the application of the proposed approach through its positioning in contrast to few other similar works. Finally, in Section 7, the paper summarizes the insights to provide a sneak view of the anticipated future research directions that can be carried out towards HF interoperability. Finally, Section 8 concludes the paper.

2 BACKGROUND AND RELATED WORK

Based on the gist of the design goal of state-of-the-art works to accomplish BI up to a certain level, these works can be classified under four primary implementation ways: (i) Notary, (ii) Hashed Time-Lock Contracts (HTLCs) (iii) Sidechain/Relay, and (iv) Cross-chain Smart-Contract (CMC). Thus, this section outlines several approaches based on the analysis of their adopted ways to solve the BI problem.

2.1 Notary-based solutions

Conceptually, the notion of "notarization" refers to a condition where usually a trusted entity (aka notary) can monitor multiple participated blockchains, and consequently, assists one blockchain in confirming/validating that a specific desired task has occurred on another blockchain. Furthermore,

conventionally, notarization is categorized in two modes: (i) centralized (i.e., notary operates as representative), and (ii) decentralized (notary provides matching services) [6]. Commonly, it is believed that “notarization” increases dependability, thereby fostering centralization in both modes. While, it is also anticipated that if the scope of the notary’s quorum will increase, it could reduce the centralization. To do so, a naive approach would be quorum periodic updating, which implies the quorum size should comprise at least Q number of nodes (i.e., minimum threshold) at a given point in time, where Q can be defined by the stakeholders depending on the requirement to balance out the security and performance. Alongside the threshold, this approach can allow more nodes to be further included in the quorum (this can be an optional parameter). It is to be noted that blockchain as a notary service within the context of various functionalities (e.g., in TLS) is different from the meaning that it has been used as a service to enable interoperability (despite being the same fundamental underlying approach). Example of such work, where blockchain-as-a-service has been leveraged to enable TLS is presented in [7], and in context of social media is presented in [8].

2.2 HTLCs-based solutions

One wants to securely exchange/swap an asset (commonly fungible) ‘X’ for another asset ‘Y’ in a “timed-release” and “P2P” approach, which seems to be the foremost idea behind the HTLCs. But the problem is, in such a task, who would initiate first without a predetermined assurance that nothing will go wrong. Of course, one can argue that why would anyone want to do that? The answer to these questions has been significantly studied in [9] [10]. In particular, in [10], the authors suggested that in such a scenario, trusted entity can play an important role. However, the current approaches based on HTLCs primarily focus on the decentralization aspect to achieve the goal of “exchange X for Y”, where a time-lock puzzle is constructed using an ordered transaction pair (e.g., commitment and redeem) and hashing techniques. For considering HTLCs-based solutions (or time-lock puzzles in general) to be valuable, “global time” transpired due to blockchain height should be taken into consideration. It means that in general, different blockchain systems incorporate distinct global time references, and to deal with it, it is assumed that solutions will apprehend time-relevant uncertainty (e.g., race conditions). For blockchain systems, the HTLCs had been first introduced concerning Bitcoin [11], and thereafter, this notion is widely adopted/explored at various levels. For example, Interledger [12] connects HTLCs in different blockchain systems through a network of connectors to foster decentralized exchange in BI. Although differing from conventional HTLCs, Wanchain [13] implements a “locked account” approach that introduces a new account (by breaking the private key and distributes it among multiple nodes) that will be locked until the transfer process is happening. Moreover, several techniques to improvise the centralized atomic exchange have been introduced, such as a new technique for hashing that achieves anonymous atomic swaps [14]. Another similar work, namely atomic swaption is proposed in [15], which extends the atomic

swaps, does not require oracles and introduces margin contracts to create and leverage short positions. In [16], the author’s proposed approach works as an extension towards HTLCs and bidirectional payment channels (e.g., Raiden network) to facilitate developers for building, executing, or even testing their micro crypto-funds.

2.3 Sidechain/Relay-based solutions

Conceptually, the notion of sidechain is no different from mainchain (sometimes referred to as blockchain or parent chain). However, to draw an analogy between them at the implementation level, consider that sidechain operates in an isolated way to primarily provide scalability and interoperability among multiple blockchains. This notion (aka pegged sidechain [17]) has been proposed to address the challenges (e.g., infrastructure fragmentation) confronted in alt chains (aka alternate blockchains), and till now, a significant progress has been made concerning sidechains. Considering the analysis carried in [18], there exist three primary ways/design choices (i.e., centralized two-way pegs, federated (aka multi-signature) two-way pegs, simplified payment verification) to implement sidechains. It is to be noted that currently, two-way peg is considered as the fundamental mechanism that technically underpins the sidechains, as it is a logical box responsible for transferring assets among blockchains. The main thrust of the centralized two-way pegged mechanisms sometimes becomes a point of concern due to its contradictory nature (i.e., centralization). Consequently, the federated two-way peg is introduced in [19] [20]. In [21], the authors formalize the sidechain system by introducing a dogmatic cryptographic definition that can apprehend the security for blockchains based on various consensus algorithms, such as proof-of-stake. In [22], the authors firstly discussed the blockchain technologies emphasizing the BI challenge, especially, which are related to ethereum private sidechains. Thenceforth, in [23], the authors proposed a cross-chain technique (specifically parameter passing method to link functions) that considers originating transaction and its corresponding subordinate transactions an atomic action (i.e., all are accepted or rejected). They also mentioned that how the concept of threshold signature can be utilized to prove values existing on one consortium sidechain to another. Moreover, they addressed the challenge of time-out synchronization (as mentioned earlier) through the notion of a global time-out by using coordination blockchain. There is another idea termed as “Relays” that is closely related to sidechains, as it depicts a mechanism for proving the presence of a state between the mainchain and sidechain. Such as in [24], the proposed scheme utilizes the validation-on-demand and on-chain SPV along with a sophisticated incentive approach to provide data verification across ethereum-based systems without requiring a trusted third entity. In [25], the authors introduced a trusted framework comprising relay services (i.e., fetch data to ensure authentication), system contracts (i.e., independent from the business logic of applications to assist in access control and identity management), and a communication protocol (i.e., buffer locates the blockchain address, target contracts, and corresponding functions) for facilitating data transfer between HF-based blockchain. It is

to be noted that the authors explicitly mentioned that the notion of "relay service" within their scope should not be misunderstood as a relay chain, as it functions as representative for each participated HF network in the interoperability message flow.

2.4 CMC-based solutions

By leveraging smart contracts, numerous blockchain systems demonstrate the viability of a range of applications either through prototypical implementation or real-world use cases. Thus, smart contracts being the logical center, now a new dimension to enable BI is being investigated. CMC primarily involves two smart contracts, let's say a contract (e.g., CAL) calls a function of the destination blockchain contract (e.g., RES) from the source blockchain and receives a response consisting of the desired results (e.g., return value). CMC as a significant research challenge in BI is firstly highlighted in [26], followed by the work done in [27], in which the authors presented a framework and provided reference implementation on Ethereum. In this framework, a voting mechanism (i.e., particularly penalizing) is proposed to address the intermediary (aka broker) behavior, where the intermediary is responsible for data transfer after the validator's validation. In a similar work [28], the proposed framework utilizes Remote Procedure Call (RPC) proxy and RPC server on the source and destination blockchain, respectively. In addition to RPC, it also requires clients (operated from off-chain) to pass the message in the request-response scenario. There exist various other works that directly/indirectly correspond to CMC, such as in [29] Sol2js, a kind of translator (also referred to as transpiler), translates commonly used language in Ethereum (i.e., solidity) and Hyperledger Fabric (i.e., javascript). In [30], smart contracts are utilized to determine data/file storage (i.e., healthcare data) and access control (e.g., patients records) using both ways of implementing ethereum (i.e., public and private). Recently, in [31], slightly different context than CMC, but within the scope of BI, the authors investigated the contributions in cross-chain technology. Based on their theoretical analysis, claimed that their results resolve differences depicted in literature.

3 THE PROBLEM

In this section, we start with the problem statement and assumptions, accompanied by the problem formulation of BI concerning the BHF network.

Definition 3.1 (MHF Network). *A network is said to be MHF if comprised of at least two autonomous HF networks. Formally, consider $MHF = \bigcup_{i=2}^n HF_i$, where n represents any possible number of HF.*

Definition 3.2 (Autonomous HF Network). *An autonomous HF is a connected modular framework that utilizes plug-and-play components to deliver applications that are presumed to be self-sufficient.*

Definition 3.3 (MHF Transaction). *A transaction proposed by applications running on different autonomous HF networks initializes and manages the ledger state that is scoped solely to chaincode accountable for its creation.*

Definition 3.4 (MHF-Ledger Consistency). *A ledger state is consistent if all inter-HF transactions get successfully executed and all corresponding intra-HF smart contracts update the ledger state in one-pass.*

3.1 Problem statement

- (MHF Network)-BI Problem: Consider an MHF transaction that is to be executed on MHF network, then how a chaincode on one HF network can invoke another chaincode running on a different HF network so that the ledger state consistency is maintained across MHF network and the complete task can be accomplished within the state-of-the-art HF?
- Roadmap of the solution: This paper converts the problem of BI concerning the MHF network in terms of investigating the ability to consider cross-chain smart contract interaction (i.e., invoking of chaincode on a HF network through chaincode on different network) as a single event (aka action). Since, the execution of a transaction is roughly distinguished into two forms: (i) a dependent transition relation, and (ii) an isolated transition relation. This paper is precisely interested in (i), as (ii) is an existing scenario, which only concerns the execution of a transaction in any autonomous HF.

3.2 Problem formulation

Lets recall the scenario discussed in Section 1 in a more generic way. Now, consider K autonomous HF networks denoted as $\{MHF_i\}_{i=1}^K$. For abstracting the involved operations (i.e., cross-chain smart contracts) of such dependent transition relation into an atomic action, the primitive notation can be used as follows: $v(a) \mapsto^c v(a)'$, which means that an action c , when implemented to the asset value $v(a)$, operates on it, exchanging it into $v(a)'$. For instance in an MHF network scenario, the application of operations can comprise reading or updating a variable, or putting/getting a value from the world state. Although it is to be noted that implementing an action c is arbitrarily complex, thus more generically, it can be said that not only operations, transactions (denoted as t) are too involved in affecting $v(a)$. Also, an atomic action is non-deterministic due to arbitrary failures of involved operations and transactions. This is denoted as $v(a) \mapsto_t^c v(a)'$. From these rules, it can be deduced that asset value changes by means of terminated sequences of cross-chain smart contracts. Under the new state $v(a)'$, it can be observed that calling out the MHF transaction (or multiple transactions composing MHF) is either accepted or rejected. In this way, we shall be able to address various questions: "Does the MHF Transaction behaves as is intended?"; "Does the chaincode fulfill the purpose in the MHF network?"; "Are the escrows reliable while accomplishing the atomic operations?"

Expected MHF interoperability workflow: Within one autonomous HF domain, it is well known that ledger-level consistency is achieved through the default invocation of chaincode (operating in an isolated secure container) or in another case where two or more chaincode get involved. Here, given the required permits, it is attainable that one

Figure 3: An example CcG diagram to delineate the invoking of smart contracts in MHF network through clients

chaincode can also invoke any other chaincode. What is noticeably considered the missing link to date is the model or approach for HF, aiming to achieve ledger-level consistency in an MHF scenario? Putting simply, it can determine whether an intra-HF transaction that manages the ledger state can be considered valid or not (e.g., such point allows reasoning whether intuitive outcomes are indeed achieved or not).

3.3 Learning the MHF network interactions

Graph Construction: Towards this end, for the generalization of the proposed approach, this paper constructs a graph called Chaincode Graph (CcG) of the MHF network. The context to construct such a graph is taken from the model discussed in [32], [33], [34], [35], specifically, from [34], where the authors presented a model to formally study transactions. Now, to construct CcG, consider vertex set V as a union of two mutually independent sets, i.e., $V = S \cup C$, where S depicts smart contracts (aka chaincode) and C represents clients. Edge set E includes outgoing and incoming edges, i.e., $E = E_O \cup E_I$, where an edge $e \in E_I$ represents invocation of smart contract, and $e \in E_O$ represents response from the smart contract through connecting smart contract s and client c (where $s \in S$ and $c \in C$) if and only if there exists a communication mechanism between them. Formally, the entities used to construct CcG are discussed as follows -

- **Smart Contract (SC).** Consider smart contracts to be a template (constrained by conditions) that is depicted by two types of nodes (i.e., $\odot$ and $\circ$), where both are initialised/invoked by transactions. Here, $\odot$ models a unique initialization SC that has only one incoming edge (i.e., $e \in E_I$ and it is represented by $\rightarrow$) (although not depicted in Figure 3, but it is assumed that there exists a smart contract that can be invoked exactly once), and has one or more outgoing edges (i.e., $e \in E_O$, represented by $\rightarrow$). $\circ$ models smart contracts that get invoked as a result of preceding smart contracts (e.g., outcome from $\odot$). This process can be cyclic as well.
- **Client.** Commonly, the client application (or simply client that is one of the actors in HF) is utilized/acts on behalf by/of end-users to submit a transaction proposal for getting an endorsed response by invoking chaincode through a subset of peer nodes. Given this role, $\square$ denotes the clients that might be concurrently running on multiple HF networks and also, the clients that might be running on dif-

ferent HF network from the invoked chain code. It is slightly different from the conventional client, where it preferably runs on the same network. One critical aspect that is quite beneficial in the MHF scenario is that the Client class design is stateful, and it maintains persistence via a plugin, namely, stateStore. Every SC of the MHF network is invoked by precisely one client. Moreover, since clients are involved in the execution of more than one SCs, thus after invoking $\odot$, its implementation is inevitably tailored to the assumption that preceded SC gets successfully executed.

- **Invoking Edge.** An invoking edge $e \in E_I$ represented as $\square \rightarrow \odot$ or $\square \rightarrow \circ$, carries chaincode function invocation (e.g., `Init`) by the clients to generate transactions for getting, putting, and deleting the world state value (e.g., for the purpose of changing/creating/transferring assets in the ledger). Putting simply, it connects clients and SC using an appropriate connection profile expressed in YAML syntax.
- **Response Edge.** A response edge $e \in E_O$ (or outgoing edge) represented by $\odot \rightarrow \square$ or $\circ \rightarrow \square$, carries a piece of information that tells clients to invoke another related smart contract (if any) only after the `commitTimeout` (e.g., 15 s). This particular transition is an outcome of a prior smart contract transaction (i.e., invocation resulting in asset key-value pair commit (e.g., `create/update/delete`)). For simplicity, there exists $e \in E_O$, if at least the invocation is successful (commonly invocation gets failed only when `--package-id` is set wrong during chaincode definition approval).
- **Failure Edge.** In CcG, smart contracts rely on each other to perform MHF transactions. Therefore, initialization SC must discover whether the subsequent SCs work as intended or if any of the SCs fails, as the failure of even one SC in MHF results in MHF transaction failure. In the latter case (i.e., MHF transaction failure), the point at which failure occurs should inform the preceding smart contracts so that ledger consistency can be maintained. To this aim, consider a failure edge $e \in E_F$ represented by $\odot \dashrightarrow \square$ or $\circ \dashrightarrow \square$, carries a piece of information (i.e., failure message) to inform the corresponding client. Once the client receives the failure message, it automatically passes it onto the previous SCs. It is to be noted that because of the HF's im-

mutability feature, it will be impossible to revert the data; however, our proposed approach ensures that the client initiates a new transaction to ensure ledger consistency. In the proposed approach, SCs failures are taken into consideration; however, within one HF, transaction failure is already a point of concern, as transactions can fail due to various reasons (e.g., concurrent execution of conflicting transactions).

Having these entities described, now let's look at the CcG workflow, in particular, what are the possible pathways in which smart contracts and clients can interact. Concerning this, Figure 3 depicts three possible graphs determining the workflow of the smart contract and client interaction in the MHF network. Although this is a utopian scenario, still the following perspicacity can be realized through building these graphs -

- **Insight 1:** Consider a graph $SCMC = (S, C, E)$, where S is the set of smart contracts, C is a set of Clients providing matching service to create connection among smart contracts, and an edge $e = (c_i, s_j) \mid e \in E$ indicates that the caller $c_i \in C$ invokes the callee $s_j \in S$. Thus, one possible scenario is that if there exists a unique node s_j , then there can exist a directed path that begins from s_j and ending at $S \setminus \{s_j\}$ nodes. An analogy to this specific graph workflow is *single-input multiple-output*. Intuitively, this is depicted in Figure 3(a) via two instances of $SCMC$. Here, at a higher level, it can be seen that an *initialization* SC interacts with two SCs (in general, there may be more than two). Moreover, it can be observed that in both the instances, number of SCs are the same. Now, let's say that the leaf contracts are denoted by S_2 and S_3 , and the root contract is denoted by S_1 . Here, it can be said that execution of S_2 and S_3 is dependent on S_1 , and consequently, S_1 can have two possible exits (e.g., process termination). But following only one of the exits from S_1 must not satisfy the atomic condition. Thus in this scenario, atomicity will only get fulfilled if and only if S_1 exits from both (i.e., S_2 and S_3).
- **Insight 2:** Consider another graph $SCSC$ such that there exists only one exit path from a unique $s \in S$. Then for every s_j invocation, there must exist a successful commit of s_{j-1} . An analogy to this specific graph workflow is *single-input single-output*. It simply means that in this graph, there exists an ordered pair (e_i, e_j) of sequence edges (where $e_i \in E_I$ and $e_j \in E_O$), which implies sequence of dependent invoking-response smart contracts. Certainly, from Figure 3(b), it can be observed that there exists only one exit path; however, handling of one smart contract has proven to be straightforward in such a graph. And it must follow three primary conditions: (i) must acknowledge the dependency existence, (ii) must comprehend the correct sequence, and (iii) must differentiate within the given sequence. Here, first and third condition satisfy logical integrity of smart contracts, and the second condition somehow ensures atomic execution.
- **Insight 3:** Suppose now CcG be an MCSC, then

it can be parameterized by multiple response edges followed by a single response edge. Given such a condition that is eventually associated with a single exit, roughly it can be said that the response edge is dependent on the combination of the corresponding preceding response edges. To this intention, recall an edge $e \in E$ indicating the connection between caller and callee, therefore in MCSC, the possible scenario will be that if there exists a unique exit node s_j , then there may exist a directed path that begins from $S \setminus \{s_j\}$ and end at s_j . For example, one such graph can be observed in Figure 3(c). An analogy to this specific graph workflow is *multiple-input single-output*. It is to be noted that here the meaning of input-output should not be interpreted as the input-output of the *MHF transaction* (this is applicable to the preceding discussions as well).

4 OCLIENT: THE CLIENT CLUSTER APPROACH

In this section, we describe *OClient*, which is further organized into two sub-sections. Firstly, the design principles through which our approach accomplishes various requirements have been discussed. Thenceforth, the next sub-section incorporates a description to explain our overall approach's working mechanism from the perspective of interaction between different modules. Figure 4 sketches the schematic representation of the OClient approach (layers and their interfaces).

4.1 Design requirements and principles

The proposed on-board client cluster approach is primarily adapted from three fundamental concepts: (i) matching services, (ii) clustering, and (iii) opportunistic computing. One can wonder about the applicability of such concepts within paving the promenade for BI research. This paper applies these concepts to fulfill following three requirements, while proposing a new approach: (i) simple (i.e., ease to adopt), (ii) decentering (i.e., less autocracy of escrows), and (iii) flexibility (i.e., client participation and adaptability).

The first requirement describes the primary objective of OClient, which is to provide a simple approach to stakeholders, i.e., with minimal changes to the existing working mechanism of HF. The embodiment behind this is that clients (aka submitting clients) are already an implicit actor of the HF (e.g., on-board clients submit the first invocation-transaction). Thus, this paper puts forward the notion of defining a new role for the clients as escrows, which will require a minimalistic change in the codebase (e.g., in the `gateway.connect()` class) of the state-of-the-art HF-based projects. In this new role, on-board clients can read/write data from the ledger in the MHF scenario by invoking SC functions through sending Transaction Requests (TR). Besides, in OClient, the on-board clients are configured with appropriate connection profiles, so that it can distinguish between the different SCs and HF networks in order to establish a connection with the target channel of the specified HF network. Here, consider an "on-board client" that follows the basic six steps (e.g., selecting identity) while submitting an invoke transaction (i.e., sending PROPOSE

message). Given this role, it can be clearly said that it somehow overcomes the obstacle of BI in HF by enabling efficient matching service among SCs belonging to different HF networks.

In OClient, decentering the centralization is the second main objective, as in the conventional notary-based approaches, making the notary a predominant power obstructs its widescale adoption (it is one of the leading problems, but maybe there are others as well). Concerning such a challenge, this paper revisits the notion of clusters from the viewpoint of clients. Under this proposition, it is comprehended that it enables a way to associate those clients who relatively share a homogeneous task. For example, recall the scenario discussed in Section 1. Now, suppose 5 clients conform as a starting point from IC at HF 1, and work together as a bridge to perform an MHF transaction. In general, client cluster represents a valued group of on-board clients in which each client application defines its role as escrow to enable matching service. In other words, it can be said that a higher degree of clusters can be recognized as a tool for decentering. In this way, it introduces a decentering of the escrows, whilst allowing BI for the MHF network as a whole.

Figure 4: Simplified OClient approach: A schematic representation to illustrate interaction among modules of OClient topology

Of notable importance, finally, the third, flexibility is reflected through the concept of clients, “the bridge enablers”, which participate in the source and target HF network only when it shares a common interest. Otherwise, they can utterly circumscribe to their previously defined role. In the disconnected MHF networks, the intermittent but collaborative actions lead to the execution of the MHF transaction through invoking SCs, as they opportunistically exchange relevant information. This adaptive role, albeit opportunistically, opens a new way of implementing interoperability in the MHF network scenario, which hitherto has not being leveraged. Indeed, it might seem that this role is ‘on the fly’ at the defining level; however, practically it is not, since the interoperability specifications (i.e., in the connection profile) should be defined before the application

loads the configuration file (i.e., when clients are off-board). Otherwise, even opportunistically, they cannot act as the bridge enablers at the later stage.

4.2 OClient working mechanism

The functionality of the Client layer is broadly categorized as two-fold: (i) Inter-Chain Interoperability (i.e., enables interaction between two or more chaincodes that are implemented on different channels within one blockchain), and (ii) Inter-Network Interoperability (i.e., enables interaction between two or more different Hyperledger fabric networks). To circumvent disconnectivity problem existing within different channels and networks, the interaction (assuming that MHF network is already set up and running) among underlying modules of OClient are as follows -

- 1) The client layer being the application backend will serve as an entry point, where a client cluster submits an invocation TR to orderers (i.e., ordering service), and for knowing the updated status of the TR, the client cluster continues to be connected to an API provided by SDK. Here, the client cluster (as mentioned earlier) is composed of clients that collaboratively work together to implement a single task (i.e., MHF transaction). Thus, the aggregated client clusters that are independent of each other’s accounts are termed as Client Layer.
- 2) The TX proposal is then constructed using Node API and thereafter, the SDK packages the same into the buffer over gRPC. Moreover, SDK also produces a unique signature corresponding to the TX proposal by utilizing the user’s cryptographic credentials. It is to be noted that various processes in OClient are similar to the working mechanism of the HF (e.g., verification of the constructed signature by the endorsing peers, in case of updating the ledger submission of the TX to the orderer nodes). Therefore, here, we are not describing such processes in detail.
- 3) Unconventional from the existing clients (as pointed out previously), the design goal of the client layer is to provide communication in the MHF network through transmitting relevant information from a source HF to destination HF in the presence of all the intermediate HFs. On this extended role, one arguably can question: *Is it a simple pursuit?* If yes, then, *How the client layer can transverse the disconnectivity boundaries of the different HF networks?* As it can be observed, the first question is reminiscent of our first requirement and its corresponding principle. Thus, it tends to stress “how?”. To better understand this “how”, the argument is elaborated in the subsequent two steps.
- 4) For establishing the connection with network nodes, the application is boot up with information of a set of trusted peer nodes (i.e., those are defined as EXTERNAL_ENDPOINT), among which, it utilizes one candidate that is in the corresponding organization. Whenever needed, the SDK can get the relevant static information (e.g., IP addresses, TLS certificates) of other peer network nodes dynami-

cally from the discovery service (i.e., running on the peers) by issuing a configuration query.

- 5) Once the event (e.g., TX is valid or invalid) is notified to the client application corresponding to its submitted TX proposal (e.g., the application gets automatically notified using submit transaction API). Accordingly, it will encapsulate the notified event as a new TX invocation proposal and sends the same to a different HF network or channel. At a high level, it can be said that the client application simply repeats the process from step 1.

5 CASE STUDY DESIGN AND IMPLEMENTATION

In the paper, we primarily study various smart contract interaction workflows by taking into account the three graphs (as discussed in Section 3.3) to investigate the inter-chain and inter-network interoperability across the prototypical scenario mentioned in Section 1. The case study design depicted in Figure 5 will diversify the evaluation of OClient's potential and address the question raised in this paper. In the essence of the OClient implementation, the atomicity of the MHF transaction should be ensured (illustrated in Figure 7), such that it can only be terminated either by reversing the effect of sub-transactions (i.e., in case of permanent or even transient failure of one or more sub-transactions) or after successfully committing the MHF transaction (i.e., in case when all sub-transactions get committed).

Figure 5: The evaluation strategy: Overview of the case study design for in-depth investigation of example scenario under OClient implementation

In the following discussion, we elucidate within the context of the prototypical scenario (discussed in Section 1) that how the OClient approach can be efficiently implemented in both cases (i.e., inter-chain and inter-network).

5.1 Divide SCs among different HF networks

For each way, as depicted in Figure 6, we constructed the prototypical scenario (as described in Section 1) under the assumption of CcG model. For example, recall the graph (i.e., SCSC) insight from the CcG model, where only after the preceding SC, current SC will get invoked (i.e., dependent invoking-response SC). Now, assume that three smart contracts are divided among three HF networks (i.e., one SC per HF network). Then, each SC will be responsible for the task it is attributed to and will follow the SCSC workflow, as illustrated through Figure 10a. We mark this as Scenario-1 (S_1) (illustrated in Figure 6(c)). Given this condition, let's say that manufacturing company is responsible for managing the autonomous vehicle's basic information, such as ID, type, color, and model through implementing smart contract (i.e., denoted as SC 1) on HF 1. Next, the

role of the servicing company is to maintain the smart contract (i.e., denoted as SC 2) that implements servicing operations (e.g., queryService, createService functions) for a general autonomous vehicle's network on HF 2. Lastly, the smart contract (i.e., denoted as SC 3) managed by the procurement company running on HF 3 is responsible for handling warranty-related enquires (e.g., by using createWarranty, query warrants functions). With these SCs called, in contrast to three network scenarios, concerning the MHF network scenario comprising two networks, SCs can be further decomposed into two different representations. From a practical perspective, it might be interesting how three different companies can maintain their corresponding SCs in two distinct HF networks. To this end, let's say in Scenario-2 (S_2) (illustrated in Figure 6(a)), one network with one channel is managing one SC, and the second network comprises two channels, each implementing one SC.

Figure 6: Different ways of smart contract interaction depicting Inter-network HF (i.e., MHF) scenario

It is a sophisticated case, as in this, there is a requirement to employ interaction between different standalone channels and HF networks resulting after reiterating both interoperability kinds, it is observed that it does not fully relate to neither of them. Thus, to capture its oblivious feature informally, it can be characterized as *hybrid MHF interoperability*. In the second representation, let's say Scenario-3 (S_3) (illustrated in Figure 6(b)), one blockchain is implementing one SC on one channel, and another blockchain is managing two SCs on one channel. In that case, it simply requires establishing inter-network interoperability.

5.2 Divide SCs among channels within an HF network

While different channels are acceptable among stakeholders (specifically among enterprises); however, considering the appropriate permissions required for invoking SC implemented on different channels might not be suitable from the accountability level. Considering this as a challenge, without losing simplicity and generality, OClient extends its services to support interactions among smart contracts that are implemented on different channels (i.e., inter-chain interoperability). The overall idea is similar to that in inter-network SC interaction; the only difference in this case is that the different companies act as the organization that implements SCs on distinct channels (illustrated in Figure 8) rather than the different networks. For given three SCs setting within one HF network, in our experiment the value of C_A is $1 \leq \Lambda \leq 3$, where C_A represents number of application channels. The connection between the scenario illustrated in Figure 10a and Figure 10b has been fruitful in conjecturing how OClient solves the disconnectivity problem in the prototypical example under inter-network and inter-chain HF networks. Having described the SCs maintained by the different companies implemented in different models/sub-models of inter-network and inter-chain HF ecosystems,

Figure 7: OClient-based SC interaction State transition diagram used for studying the atomicity and ledger consistency

we will now describe assumptions that are instantiated in the underlying prototypical scenario to make a realistic evaluation during the experimental analysis (illustrated in Figure 11). It is to be noted that the underlying assumptions are defined by the Boolean data type, as the interaction is assumed to be uncertain. This means that SCs can fail at any point of time during their implementation in both an inter-network (MHF) and inter-chain network scenario, and is therefore determined using a status parameter (i.e., success or failure). Moreover, these assumptions will assist in addressing the question, “How consistent is the OClient approach when one or more accountable SCs will fail?” Note: within the scope of this paper, the definition of consistency is:

Definition 5.1 (Consistency). *Consistency is the property of the underlying HF system that facilitates a degree of assurance that the status parameter of all participated SCs will operate in the way, as is intended, irrespective of the SCs execution success or failure.*

Figure 8: Different ways of smart contract interaction depicting Inter-chain HF scenario

Assumption 0 (A0): Let there be $v(a) \mapsto_t^c v(a)'$ as an atomic action, then the preferred specifications of $(n/2)$ SCs fail at an epoch δE , where n is the total number of involved SCs.

Under this hypothesis, the random sample approach has been opted since in this approach, during an epoch of SCs interaction, it is presumed that $(n/2)$ SCs fail randomly, which ensures a significant degree of randomness. This paper identifies such failures as *sporadic failure*.

Assumption 1 (A1): This formulation is circumscribed by the assumption that the number of failed SCs, denoted as f will be $f < n/2$.

In practice, however, considering that failures are unbalanced and arbitrary, it cannot be estimated that what will be the failure rate. However, for studying the multi-view nature of failures, it is reasonable to assume that the number of failed SCs will always be less than half of the total participating SCs. Observe that this hypothesis is different than (3), as in this, at least one SC will perpetually fail out of n SCs (termed as *inlier-sporadic failure*).

Assumption 2 (A2): In this, assume that if the number of failed SCs are denoted as f , then in a defined random setting $f > n/2$ percentage of nodes will always fail.

Figure 9: Variations in failure rate of the smart contract based on the underlying assumptions

It is a pragmatic and non-biased setting, where it assumes that the number of failure SCs is more than fifty percent of the total number of participating SCs. In this setting, there is a higher probability that the epoch time for which the interaction persists will always be less than δE . Such failures are identified as *wide-spread failure*.

Assumption 3 (A3): According to this assumption, f will become 0 when all SCs will fail, and 1 when all SCs will be executed as intended.

Under this hypothesis, there are only two possible outcomes for each transaction flow, often designated as upper bound and lower bound, where upper bound means the failures of all SCs (termed as *absolute failure*), and in case of lower bound, it signifies that there are no failed SCs (termed as *no-success failure*), which is the desired goal. Interestingly, in the upper bound, if SC_1 is failed to commit, then $(n-(n-1))$ dependent SCs will not get invoked. Even the world state corresponding to SC_1 will fail to retrieve/update data. It is to be noted that the failure rate may be subjected to various other variations or ranges; however, within our scope, such failures are termed as outliers.

6 IMPLEMENTATION AND EVALUATION

In this section, we first describe the simulation methodology for the OClient approach to investigate the performance of various parameters (illustrated in Figure 5) within the context of the prototypical scenario (Figure 1). Thenceforth, we demonstrate that what will be the impact of the assumptions on proposed approach consistency in practice by randomly selecting the transaction flows (based on the CcG graph), where SCs also interact (invoke each other) in a randomized way. Lastly, this section analyses the experimentation results to present important insights into the overall approach.

Figure 10: Examples of OClient approach with inter-network and inter-chain SCs interaction

6.1 Experimentation simulation methodology

In order to simulate the OClient approach more practically, and specifically, deploying the sophisticated multi-HF network scenario, this paper sets up the environment in federated research cloud (i.e., ARDC’s Nectar research cloud). More specifically, we employed instances with homogeneous hardware specification comprising 2 VCPUs, 4 GB RAM, 30GB disk space, and utilized homogeneous software specification consisting of Ubuntu 18.04 LTS (Bionic) amd64 as an operating system, Docker v19.03.6 as the container, Docker-Compose v1.17.1, Node v6.14.8, Python v 2.7.0, and Golang v1.14.6/amd64 as the preliminaries to run the HF network. This homogeneous configuration is highly relevant for correctly measuring various significant parameters that are important from an interoperability viewpoint (both inter-chain and inter-network), and it also ensures a higher degree of uniformity. Our proposed approach (i.e., OClient) is implemented under these specifications on the latest version of HF (v 2.2.0), which is equally compatible with earlier HF versions (ranging from 0.x.x to 1.x.x).

Opting to have multiple channels within one HF network, we assume that a peer node cannot join more than one application channel to reduce redundancy. Consequently, performance can also be improved upto some extent by following this approach of allocating channels (specifically, by permitting parallel endorsements, as an example). Furthermore, towards maintaining the balance among parameters that are significant for HF nodes, such as networking and computation, this paper comparatively advocates utilizing a small HF network that includes one orderer node and two organizations having one peer node each. To incorporate the prototypical scenario, three chaincode (aka smart contracts) are deployed (written in JavaScript and Golang), one for maintaining the autonomous vehicle information, the second for maintaining the servicing-related information, and the third is responsible for maintaining the warranty information. A naive potential deployment of these SCs under different networks and different channels is illustrated in Figures 6 and 8, respectively. Intuitively, this paper defines a connection profile for each interaction through which client applications can connect (as described earlier) and where registered users can participate using the certificates stored in the wallet.

6.2 Experimentation examples

This section provides a sneak viewpoint of one pathway of each graph depicting potential interaction among SCs using client-level interoperability. Perhaps, it is preferable to present this using examples implementing prototypical scenario, benefiting from the OClient approach. Although these examples can be understood as a general representation for each graph category that allows understanding the client-level interoperability; however, they are quasi-simultaneous, which implies that they are independent of various other possible implementations.

Example 1: The applications that may want to invoke two or more SCs after the commitment of one SC, e.g., SC maintained on manufacturing company wants to invoke SCs maintained on servicing company and procurement company, to perform query/update functions simultaneously. In this example, the client cluster must be able to determine the interaction path as specified in the SCMC graph. In addition, to perform the MHF transaction successfully, let’s say all the callee SCs (i.e., dependent on successful commit of the `initialization` SC) must have been invoked successfully, is the responsibility of the caller SC (i.e., `initialization` SC). Whilst the successful invocation of the callee SCs cannot endorse the MHF transaction success alone, it must update the `initialization` SC status parameter as (e.g., “SUCCESS” or “VALID”), which means that all the callee SCs function as intended have been notified.

Example 2: In light of the prototypical scenario, this example describes a plausible condition when two or more SC’s combined output invokes one distinct SC, which forms one of the three graphs of CcG (i.e., MCSC). The dominant use of such an example is where multiple SCs that are analogous as tightly coupled (across networks or channels), logically need to be instantiated as an `initialization` SC to forward a TX proposal to a disassociated specific SC. For example, MC’s SC needs to check whether the autonomous vehicle’s details are the same as recorded by a particular stakeholder, and simultaneously servicing company SC queries its world state to check whether the vehicle is still under warranty or not. In this case, the output of both SCs will be encapsulated to invoke SC maintained on the procurement company (e.g., update the warranty details of a specifically requested

component).

Example 3: The preceding two examples reflect either 1-C-N or N-C-1, where C denotes the client cluster(s) and N is the number of SCs; however, there is still one unconventional topology that needs to be considered in which interaction reveals a serially ordered SCs invoking sequences. The serial ordering assumes that for a particular MHF transaction, the path for the action sequence should be distinct from other MHF transactions. For example, SC on a manufacturing company in all likelihood can interact with SC on a procurement company; however, not directly, but it must be in a successive invoking fashion in proportion to their level of dependence (e.g., in our case study through SC on the servicing company, as illustrated in Figure 10).

6.3 Result-oriented performance evaluation of the case study

As OClient may have different performance results under real-world scenarios. Hence, to study how its practical implications affect the performance in such scenarios, we have conducted simulations under different configuration settings (e.g., different numbers of HF networks), where this comprehensive evaluation report can also contribute insights in terms of inter-chain/network metrics. Besides, underlying parameters (e.g., Batch size, TX arrival rate) will remain homogeneous for each HF network, as described in Section 6.1. Though the experiments are performed in a simulated environment; however, the stakeholders can interpret them as an effective proof-of-concept, as the proposed approach is implemented using cloud services rather than a local environment.

In order to provide an insight into the timing behavior of the atomic task, the execution time needs to be observed to find out whether the assigned task in the MHF environment takes longer or shorter time as compared to an autonomous HF network. Moreover, considering the research highlighted in [36], where knowledge for the execution time has been claimed to be of utmost importance while creating and verifying real-time tasks. Thus, this section compares the Average eXecution Time (AVXT) by presenting and analyzing from various viewpoints of the proposed OClient approach. One can argue, *why not maximum (i.e., worst) or minimum (i.e., best)?* The reason behind this is, while conducting several rounds of experimentation with the same underlying network setting, it has been observed that there is a minor variation in min or max execution time. Consequently, in contrast to an observational study, our experimental design chooses to determine AVXT (as there are no significant outliers as well), and it is obtained in two steps, as outlined below -

- 1) The bound (i.e., upper and lower) for each round depends on the duration (i.e., when a client cluster submits a TX proposal until the TX response is not being inspected by the application). It is to be noted that duration may differ, as in the case of ledger updation, the application needs to submit the inspected (or even uninspected) response to the orderer, and then the notification is received from the peer node. This bound is specified in a program (written in shell script) that simulates SCs

interaction based on the insights provided under CcG graph.

- 2) In the second step, after the calculated max and min execution has obtained, the AVXT is derived by taking an average of the higher bound for both max and min execution time of multiple conducted rounds, as we assume that the hardware and operating system behavior (such as timing behavior, DMA performance on the task) on the Nectar cloud are predictable.

6.3.1 Instantiating different number of HF networks

While increasing the number of different HF networks may be understood straightforward, identifying the practicality of these networks to provide insights for implementing more such systems is a sizable challenge. Towards this, a practically sound approach would be statistically evaluating the AVXT, it may also assist in inspecting the question that: *Do the different number of HF networks affect the series of practical examples?*

However, this is a possibility that the prevalence of the obfuscation due to several other HF system parameters may affect the correctness of this approach. To address this, we introduce minification (i.e., in terms of the number of HF networks during their evaluation) and other parameters are instrumented to be similar (i.e., not modified for each experimentation round). To determine the impression on the

Figure 11: Performance measurements depicting calculated AVXT to compare performance achieved in different examples, with variation in the number of HF networks and interaction paths (i.e., based on CcG) under similar SUT individual HF's condition

examples (described in Section 6.2), we run the experimental process for each of the three scenarios (i.e., S1, S2, and S3 as mentioned in Section 5.1). Overall, as can be observed from Figure 11 (snapshot of the log entries of execution time), for the number of HF networks that participated in the inter-network interoperability, the peak AVXT is approximately 7.669s, and the minimum AVXT is 3.103s. So far, from this observation, it can be said that it attains a reasonable range when compared with the performance of the AVXT for an individual HF network. Since, a single HF network with a known topology of nodes and probabilistic finality invoke.js script takes 2.5 s (with default configuration), which places

these results to match with our achieved results (as $2.5 * 3HF$ networks = 7.5 s). When this AVXT distribution is analyzed for each example based on the defined scenarios, there is no statistically considerable difference observed between E1 and E3. However, example 2 (E2) is subject to a less AVXT across scenarios, comparatively to both examples (i.e., E1 and E3). Given this peculiar reading of E2, let's find the factors (i.e., parameter value) that affect the AVXT distribution. The primary reason we found after an in-depth inspection is that the way of SC interaction is directly responsible for a consistent lower-AVXT pattern, which means that when the MHF transaction is executed in an MCSC fashion, the probability of AVXT lower range will be most likely.

Observation: The experimentation results illustrate the best-case AVXT and two other readings (with a small variation) achieve an approachable range compared to an autonomous HF network. Assuming that these statistics capture the near-practical scenario, it can be said that the introduced OClient approach, regardless of the number of different participating HF networks, achieves a consistent AVXT in inter-network HF interoperability, which manifests its feasibility. The second notable insight achieved from this experiment is that the reason for AVXT deviations under the same system settings (in particular, the number of HF networks) is marked by a new kind of effect (i.e., the way of SC interaction). In particular, this insight addresses the question: *Do the SC interaction flow and corresponding dependencies affect AVXT?*

Remark: Our analysis addresses stakeholder's concerns, such as *what will the trade-off between performance and the number of HF networks, and also if OClient will be opted for enabling inter-network HF interoperability, Will the performance degrade or improve?* To this end, we identified that OClient does not introduce any additional complexity (as it includes negligible compute-intensive logic), and hence, the resulting performance is not degraded. However, in comparison to individual HF network performance, it is not improved as well, which we believe is not a limitation (as this work does not concern the improvement of HF networks performance). Although, as future work, it can be extended to improve the communication delays caused by involved clients.

6.3.2 Instantiating different number of application channels

Are the questions addressed so far able to provide a well-argued opinion about OClient? Yes, but a bit pointed, as the goal of OClient is to implement inter-chain HF interoperability. Thus now, considering the specifications that are of particular interest to the inter-chain HF interoperability, this section carries an in-depth investigation of the achieved results for the defined scenarios (i.e., where C_A is $1 \leq \Lambda \leq 3$ as described in Section 5.2) and examples (as discussed in Section 6.2). More specifically, the considered specifications are similar to inter-network HF interoperability; however, in the sense, when an organization implements its corresponding SC on multiple fragmented application channels rather than different networks. Clearly, the AVXT results depicted in Figure 12 remind the AVXT achieved in Inter-network HF interoperability, as the comparison ratio (irrespective of the different channels and networks) closes the gap between the AVXT difference between them.

Figure 12: Performance measurements depicting calculated AVXT to compare performance achieved in different examples with variation in the number of channels and interaction paths (i.e., based on CcG) under similar SUT individual HF's condition

In a more general perspective, this investigation demonstrates that OClient facilitates a coordinated competency while walking between inter-network and inter-channel competency. Nonetheless, here, the discussion is about the OClient ability to achieve AVXT over a different range of channels. Hence, a more realistic question would be, *Is OClient's calculated performance parameter (i.e., AVXT) satisfies the expectations, in case, stakeholders desire to opt for different channels to deploy their smart contracts?* Elaborating on the results shown in Figure 12, the discussion is two-fold: both are correlated, however, arguably provide different insights. First, it suggests that the varying number of channels (i.e., Λ) participating in the inter-chain interoperability does not disrupt the performance of the proposed approach, which relatively addresses a broader concern of the stakeholders. It can also be anticipated that overprovisioning of different channels to enable a higher degree of privacy is a viable option for stakeholders, ultimately, justify the utilization of HF as a scalable blockchain platform. Second, attempting to architect the way of SC interaction (if possible) should be taken into account (it is analogous to inter-network AVXT performance), as AVXT will continue to get affected by different ways of SC's interaction or dependencies (e.g., E2 AVXT can be easily discerned as a best-effort from other two examples).

Observation: In essence, fragmentation in different channels is an HF interoperability problem. This paper identified a number of ways (CcG graph) in which SCs implemented on application channels can interact with each other. Moreover, we conducted minimalistic complex experimentation (assuming that there are various other viable conditions) to point out the OClient performance. Of course, the provided insights (previous subsection) typically address most of the pragmatic research questions. Yet, stakeholders may face a common challenge *"Is OClient a beneficial approach for enabling inter-chain interoperability?"* To this, there is no simple answer, as there might be different viewpoints that may be firmly considered by stakeholders across different application areas. Despite this potentially appeared perspective rift, experimentation results largely conclude by highlighting that OClient provides a "reasonable performance that does

not need a different mechanism for enabling inter-chain HF interoperability”.

Remark The observation above reveals that more sophisticated (and possibly diverse) approaches can be mapped on application channels while implementing SCs. However, one aspect of the investigation suggests that the interaction flow of invoking SC assigns varying performances consistently. Overall, this specific experimentation estimates the AVXT within Inter-chain HF interoperability, so that given a set of application channels and SCs along with their corresponding invoking flow, one can better determine the potential AVXT range based on their requirement.

6.3.3 Instantiating different number of clients in a client cluster

The OClient approach shown in Figure 5, can be evaluated under three parameters, two of which are already discussed. Now, to better study the decentralized perspective of matching service (i.e., escrows (in our case clients)), it is expected to investigate *Will varying number of clients in clusters affect the performance given the inter-network and inter-chain HF interoperability?* To investigate this, the experimental set-up has similar specifications corresponding to Sections 6.3.1 and 6.3.2 while mapping the different client numbers. This a-priori configuration model assumes that the number of clients can vary within one cluster, which can be further be elevated as heterogeneous clusters. The argument behind this is that there is a higher probability that clients who have some opportunities of sharing assets will be deployed in one cluster (can be referred to as a homogeneous cluster). More importantly, one may raise concerns about deploying a comparatively less number of clients (i.e., 1, 3, and 5 clients within one cluster). Facing such concerns, it can be said that here clients do not mean that the number of users is limited, as there can multiple users participating in the HF by registering on one client. Therefore, our proof-of-concept could be more appealing to stakeholders who wish to encourage users to participate with a steady and small number of clients (notably, the choice of client sets depends on each stakeholder’s perspective). As we can observe from Figures 13 and 14 that visualize the execution time of the test examples and scenarios in terms of varying number of clients. The interpretation from these results suggests that with the growing number of clients, OClient’s performance is not depreciated, as the observed continuous increase is much anticipated, thereby providing an approximate range for estimating the distribution of time taken by the clients. With this range-based execution time, it is noted that even the anticipated increase is practically a relative decrease, e.g., the observed increase is 23.3% when the client number increased from 1 to 3; however, when it gets increased from 3 to 5, the observed increase is 16.6%. Besides this, there is one more question often raised in the HF network performance, *how much execution time is taken by different transaction types?* It will not be surprising that this same question turns towards MHF transaction types. Hence, to address this question apriori, especially for such a scenario, where which transaction type can be prioritized to achieve the best service effort is the main motivation. Thus, during the experimentation carried out to investigate this, we single out one configuration with three different HF networks, as

it adequately provides a much-sophisticated environment to test the execution time for MHF read and write transactions. That is, the MHF read transactions are separated from MHF write transactions, and the metrics achieved are illustrated in Figure 15 for varying numbers of clients. In this way, it can be said that the scalability is also preserved, while studying MHF read/write transactions provides better assessment in a more decentralized environment. As it can be observed, there is no trade-off between client number and MHF transaction type (Figure 15); however, as expected, the execution time, where implementing an MHF transaction requires two read operations and one write operation, is less (middle plot of Figure 15), as compared to, where MHF transaction requires all write operations (left/right plots of Figure 15).

Observation: The insights achieved using OClient have already shown that it shares the common ground when it comes to enabling either inter-network or inter-chain HF interoperability. This persistence also appears to be exemplified in the varying number of clients experimentation, as the execution time incurred in both HF interoperability does not discard each other (i.e., the increase is synchronized), thereby confirming that OClient does not require additional computational or programming overload. That is, given the number of clients within a cluster, the execution time consistency for client tasks (i.e., matching services) holds for both HF interoperability. Interestingly, the histograms report the distinction between execution time obtained for different examples, which reaffirms the insight that SC’s interaction way is the most notable factor affecting the OClient’s performance. The middle plot of both figures identifies that the example where interaction structure is employed on the presumption of MCSC graph demonstrates a lower AVXT rather than the other two examples (left and right plot). Given the initial observation regarding MHF read/write transaction type, it can facilitate at various points (specifically, in a practical scenario) where a designer can select the pattern to execute a specific type of transaction for the purpose of reducing the overall execution time (i.e., if a particular transaction category relatively consumes more execution time). Furthermore, in a way, research questions such as *how complex their overall design can become and does it has the same implications as an autonomous HF network?*, can be addressed based on this experimentation observation within the context of implementing OClient.

Remark: The following observations show the effectiveness of OClient in terms of scalability, and from this decreasing viewpoint, although, stochastically, but it can be estimated that that OClient’s ability towards varying client numbers can provide a relatively higher probability of stable execution time under similar parameters of HF network.

6.4 Evaluating operational consistency under unreliable presumptions

This paper identifies one of the common problems with individual HF networks as well as with respect to interoperability scenario that the consistency has only being evaluated under the assumption of a fully reliable system (i.e., simply assuming that the underlying blockchain system or its components (e.g., smart contracts) never fail).

Figure 13: Clustering execution time of OClient based on the underlying scenarios (i.e., Inter-network) that is further categorized based on the examples and is compared against execution time of three different client configurations (i.e., 1, 3, and 5 clients)

Figure 14: Clustering execution time of OClient based on the underlying number of channels (i.e., Inter-chain) that is further categorized based on the examples and is compared against execution time of three different client configurations (i.e., 1, 3, and 5 clients)

Figure 15: Visualization of the impact of different MHF transaction types (i.e., write and read) on the performance of the different client configurations. As can be seen, MHF read transaction (i.e., E2) requires less execution time (less is better) for all of the client configurations

Therefore, there is a higher probability that the derived results from such assumptions demonstrate that the proposed approach ensures consistency. To address such a challenge, we identify an approach *Multilevel Failures* that can be one of the ways to comprehensively evaluate the proposed approach, in contrast to the no failure assumption. In this approach, it is presumed that consistency should be ensured, even if some of the system components are failing or complete the system (a variety of assumptions are described in Section 5.2). There are various steps at which these failure assumptions can be applied, since for triggering (i.e., invoke) a smart contract, a TX proposal is required. Thus, this paper considers the transaction as a first logical

break-point where the transaction flow is intended based on the assumption of CcG graph. While the transaction flows are defined for investigating these varying percentages of breakpoints, it is also assumed that randomization will be a more appropriate way. Therefore, on the client-side, we have conducted the experimentation in the sense that an MHF transaction can invoke a varying number of SCs with the randomized corresponding transaction flow.

Figure 16: The plot representing the percentage of considered transactions, grouped based on the different SFR range

Furthermore, for the virtue to reflect a near-practical scenario, the experimentation involves 124 different transaction types, along with 15 different transaction flows. Intending to investigate different failure assumptions, it is relevant to accurately capture what is the failure rate within the context

of HF interoperability? Concerning this case, if given the total number of involved SCs is represented as T_{SC} , and the number of SCs failed to function as intended is presented as E_{SC} , then the *smart contract failure rate* (SFR) = $(E_{SC} \div T_{SC}) \times 100$. It also appears to provide an estimate of the underlying network reliability (r), and at a higher level, it can be said that $SFR \propto \frac{1}{r}$. Besides, this reliability estimation is not simply dependent upon a single value of SFR, as it can be closely related to SFR intensity (i.e., percentage). Thus to track this variable SFR, the Mean Time Between Failure ($MTBF$) can also be considered as a parameter of estimation. Therefore, it can be defined as: $MTBF = \frac{1}{SFR}$. When the SFR drops to 0 on the axis of Figure 11 (i.e., A_3), then $MTBF$ will be $+\infty$, if the solid vertical line marked to rightmost of the same axis, then $MTBF$ will be 1. Similarly, the expected $MTBF$ in case of A_0 will be 2, in case of A_1 will be $+\infty < MTBF < 2$, and lastly, in case of A_2 will be $2 < MTBF < 1$. Now, the amount of transactions constituted for these SFR or $MTBF$ are pooled from the total number of transactions considered for this experimentation. Using this, we distribute 12% of transaction under A_3 (i.e., 0% SFR), 12% under A_3 (i.e., 100% SFR), 16% under A_0 , 28% under A_1 , and 32% under A_2 , as illustrated in Figure 16. Presumably, the failure constraints determined will minimally allow investigating a general ledger consistency under OClient. In particular, this will allow to a certain degree understanding of what amount of rollback transaction is required to achieve the ledger consistency requirement.

Figure 17: The plot comparing TP1-TP5 by evaluating rollback transactions required in the considered scope of SCs

One of the major overheads that may render in OClient (in case of intermittent or complete failures) is due to the TX rollback. Due to the implicit functionalities (e.g., immutability) of blockchain, currently, there are no other techniques (other than rollback transaction) that can reverse the action caused by TX write in the HF database. Though not an optimized method (yields redundancy), but still to avoid inconsistent database in case of aborted action, OClient considers a TX-rollback oriented approach (as discussed previously). Thus, to compare the amount of rollback TX generally required for distinguished transaction patterns (TP) (TP is labeled to represent assumptions (A0-A3)), this paper evaluates the rollback TX number corresponding to a maximum of 8 SCs and a minimum of 2 SCs. To better understand this TPs evaluation, intuitively, it will be worth

considering different scopes for the SCs. Accordingly, the scopes for which these TPs are evaluated are denoted as A, B, and C, in an increasing number of SCs. Figure 17 shows the number of rollback TXs elicited, where failure checkpoints are based on the different combinations of underlying TP's and SC's scope. Besides, this experimentation will address the research question: *How much rollback TXs required under the SCs scope and SFR assumptions to ensure consistency?*

Observation: The results analyzed to calculate rollback TXs intensity suggest that the value will remain as 0 when the underlying experimentation system observes $+\infty$ and 1 MTBF. Fig 17 shows that across different SC's scope, the rollback transaction does not vary and remains zero, in case if there is no failure or absolute failure. Note that the likelihood of rollback TX, in case of no failure, can easily be generalized; however, one may argue that if a complete failure has occurred, then why no rollback TX is expected. Recollect that for the CcG graph, if failure checkpoint (impending of its different insights) detects an error in the first SC invoke, then OClient will not require any rollback TX. It is the reason for no differences between the number of rollback TX for these two MTBFs, which is an interesting observation. In the extended analysis, it is noted that there is an arbitrary probability to make a generalization about rollback transaction numbers in other varying numbers of failure assumptions. One of the identified reasons causing uncertainties is the threshold value of participated SCs. Consequently, if the value is low under the same failure assumption, then the accumulated rollback transactions will remain low, and vice versa. For example, when MTBF ranges from $2 < MTBF < 1$, then the number of detected rollback transactions is progressing at a linear rate in the case of Scope C, as compared to other TPs (illustrated in Figure 17). It is also possible that even if the failure rate is higher and the SC scope is the same (e.g., TP4 and TP5 with the underlying Scope B), then it may persevere less number of rollback transactions. It depends on various reasons, such as sometimes the larger number of checkpoints may minimize the probability of the higher number of rollback transactions.

Remark: For a given failure rate or MTBF, consistency will continue to be determined through independent rollback transactions. This is even more applicable when this approach fits the requirement of ledger consistency according to the initial assumption of ensuring the desired level of 100%. However, during the experimentation, there is an assumption that the rollback transaction will not fail (as there is no such case encountered during the experiments). Although the probability is quite small, one can encounter that a newly created rollback transaction will abort an arbitrary number of times. In this case, ledger consistency will only be ensured by adopting the mechanism of rerun or iterations. Therefore, questions such as *how many iterations should be required? How much execution time should be expected?* are worthy for further investigation in such successive rollback scenarios.

6.5 Positioning OClient: Comparison with existing works

The stakeholders may find this OClient beneficial after studying the preliminary performance evaluation; however, let's take this evaluation discussion to another step where on one side, they might argue that *how relevant OClient is as compared with the existing approaches (or where it stands?)*. Putting simply, they might ask "What OClient is" and "What it is not?" Moreover, researchers may argue that —*Why only clients have been selected as the escrows while there are various other HF components?* Now, through the analyses provided in this paper, let's span these questions collaboratively, as it can be noted that they are intertwined at a conceptual level. To find the OClient position or relevance in contrast to the literature work or even ongoing projects is not that straightforward, as various parameters that have been formulated in our approach are hardly investigated in them. However, this does not make them significant or insignificant. Contemplating this, this paper suggests that the broader domain that yields to make the comparison ratio more consistent lies within "HF" and "BI".

There are various works that are trying to address the dimension of interoperability within the blockchain, as briefly highlighted in Section 2. However, if this dimension will be deduced to HF, and in particular, enabling interoperability among different HF networks, in that case, after studying most of the identified works, it can be said that there is no substantial correlation that prompts for describing comprehensive comparison between them and the proposed work. Meanwhile, during this comparison attempt, the work done in [25] can be somewhat directly compared with the proposed work. To make this comparison fair, it should be noted that the authors [25] have not provided a description regarding their implementation evaluation (both in terms of performance and correctness). However, understanding their work's relevance, this parameter is not tuned for comparison. Now, we consider their underlying architecture and message flow in contrast to our proposed approach and SC interaction flow (based on the CcG graph) as the fundamental parameters for comparison. Their proposed architecture introduces a relay service that is operated on three primary layers (i.e., technical, syntactic, and semantic). On the contrary, our approach introduces a new role for the already existing clients in the HF, which ensures more simplicity during the implementation. Specifically, while addressing fragmentation challenge, they assume that underlying participated networks will initialize three system contracts (Configuration Management, Exposure Control, and Data Acceptance) to implement networking rules using SC run-time environment (i.e., in a similar way as application contracts). In addition to this hypothesis, they also assume that participating networks must have the preliminary knowledge of identities and configuration of each other. In comparison, our proposed approach does not necessitate any such assumptions or even implementation of system contracts, as it defines the client role in a way (which is not a separate component of HF) that exploits the application-level contracts to enable interoperability. Furthermore, in this work, the clients should have the knowledge of the participating network; however, as future work, we observe

that this knowledge assumption can be minimized using a suitable approach (as discussed in Section 7).

Nonetheless, to sum up, there are other research efforts as well that are parallel/close to the proposed approach for questing towards interoperability, with the exception that their goal of interoperability focuses on other blockchain platforms. Table 1 provides an outline to compare critical features or capabilities enabled by such state-of-art works in contrast to the proposed approach. This comparison indeed does not aims to point out the drawbacks/limitation of the highlighted works; however, it endeavors to retrospect a question (as mentioned earlier) that what are the goals achieved by these works, and where the proposed OClient stands with respect to them?

7 DISCUSSION

In this paper, we are motivated to look into the possibility that can act as a *systematic approach for enabling interoperability among MHF network scenarios, transversing the existing disparate boundaries, performing smart contract interaction, and acting timely to stakeholder's requirements*. The use-case acting as a proof-of-concept benefiting from the proposed OClient approach, on a promising note, consequently will underpin a better provisioning of seamless SCs interaction on the move using on-board client clusters among various HF-based applications. Moreover, the proposed work marks as a stepping stone towards HF-interoperability. This paper believes that it is only one way to achieve the goal, as there may exist different ways to achieve the same. Thus, we briefly outline the following practical future directions that can be explored.

FD 1: *What should be the way forward to ensure atomicity if client clusters will not be considered on-board in each participating HF network?*

Based on this paper's suggestion, ideally, clients should be members of each participating HF network in the MHF scenario (i.e., on-board); however, it turns out that there can be a research possibility that if the public key is included in the transaction rather than the Membership Service Provider (MSP) operating on the ordering service verifying the signature, then the off-board client may submit the TX proposal using the private key (i.e., bypassing the MSP). Despite its simplicity, there may be unexpected security concerns with this approach. Thus it opens new challenges that can be investigated in the future. Looking forward, another pragmatic possibility is that within an MSP, appropriate rules can be defined in both local and channel MSP, as we can consider an MSP as a component that can turn trusted identities into the member of the HF. In other words, it addresses the fundamental question "*how HF network utilizes the chain of trust?*". In this way, if an off-board client can prove that it is a trusted entity, then even without becoming an actual member, practically it can access the services in the MHF scenario.

FD 2: *How the MHF transaction's successful commit guarantee can be improved. Or in other words, how the MHF network can be recovered, mainly in the case, where client clusters or even one client gets failed?*

From this paper, one of the perspicacity is that in both inter-network and inter-channel interoperability, transactions require a synergetic effort from each client belonging

Table 1: OClient Positioning: Overview of comparison among existing works

	Bifrost [37]	Abebe et al. [25]	Herdus [38]	OClient (Proposed approach)
Source/Destination Ledger	Source / Destination: Bitcoin, Stellar, Ethereum, EOS, IOTA, Sawtooth, Multichain	Source: permissioned blockchain / Destination: permissioned blockchain	Source: Any cryptocurrency / Destination: Ethereum	Source: HF / Destination: HF
Trust assumption (required)	Requires trust at three layers: First, underlying code and cryptographic properties of representative BC. Second, RPC client and server. Third, Notary	Trust in the relay service (i.e., derives from underlying consensus protocol)	Introduces an augmented information layer	No additional assumption
Core underlying approach	Notary	Shared network-neutral network	Notary	Hybrid (i.e., Notary and CMC)
Known for	Enabling storage and retrieval of data through primarily connecting seven different blockchains using the notary	Enables a generic trusted data exchange among fragmented networks	Cross-chain interoperability with better scalability, conformation time, and private key security	Enabling inter-network and inter-chain interoperability among HF
Implementation level	Using three primary components namely, The API, Adapters, and database	Relay service operates at three layers (i.e., exploits system contracts)	Proof-of-stake based blockchain can be operated as a sidechain (i.e., blocks-of-blocks architecture)	Clients
Technical compatibility (demonstrated)	Seven blockchains (as already mentioned in 1st row)	HF-to-HF networks	Commonly BTC-to-ETH, claimed any cryptocurrency-to-cryptocurrency using HER token	<ul style="list-style-type: none"> • HF-to-HF networks (Inter-network) • HF-to-HF channels (Inter-chain)

to a corresponding client cluster, to ameliorate the trade-off between consistency and TX commit successfully. However, one of the foremost challenges in this is the client's reliability, as in the case where a client responsible for the SC interaction, will not function as intended or miss the desirable timeout (e.g., batchTimeout, commitTimeout) or move out from the task. To this end, the proposed approach ensures reliability in the sense that using rollout TX (resubmitting a fresh TX proposal), the ledger will be returned to its previous state. However, currently, there is no mechanism to tolerate the client failure so that re-submissions of TX can be minimized (e.g., how to prevent clients from anticipated failures). To this future goal of preventing client failure, this work's preliminary research suggests that by harnessing the incentivize mechanism, the dynamics of the client's intentional back out can be handled. This option can be interpreted as - the more resilient the network is, the higher probability of the MHF transaction's commit success. More specifically, in the critical scenarios where transaction success is an utmost desirable requirement, incentivize-based client-oriented interoperability may be more appropriate.

FD 3: *With the ever-evolving networks, how can client participation as an interaction bridge, can be backed without any prior provisioning of the network configuration?*

In this work, we primarily investigated the overlooked possibility to enable MHF network interoperability (both inter-network and inter-chain) and considered the simplified version of the client participation. However, there is scope for more research work to explore the question of dynamic client participation. In particular, it is important to achieve in a situation where there is an inevitable demand for querying/updating a ledger on an HF network that is not yet been included in the MHF network. In this unanticipated scenario, by harnessing the proposed approach, the client that is already a member of the foreign network can easily bridge the interaction (i.e., it will be feasible to send a TX proposal to the foreign HF network), given a condition that somehow it can delineate a future work that off-board client can include in the ongoing SC interaction (i.e., MHF network).

Towards these three primary future research directions, several other questions remain unexplored in this work. In particular, their theoretical and pragmatic challenges can only be understood, only if they are investigated comprehensively.

8 CONCLUSIVE REMARKS

The hype of the nascent HF applications paradigm increased upon the realization of the myriad of features provided

by this enterprise-grade blockchain platform. This niche of HF use cases is in the spirit of the siloed environment; consequently, these extrapolated implementations will underpin major bumps in its research roadmap in the form of fragmented HF islands. For addressing such problematic repercussions, interoperability is identified as the pragmatic approach that presently and in the long-term unravels the possibility of the connected environment by enabling an interaction bridge.

In this paper, we proposed a CoA-based interoperability approach for the MHF scenario that endorses building blocks features, such as flexibility and no-additional trust assumptions. Unlike other existing works, it does not foster the provision of redesigning the state-of-art HF framework, as it determines a new role for one of the HF's central elements (i.e., clients). We have constructed a graph (i.e., CcG) to formally studying the possible transaction pathways to invoke SCs implemented in inter-network and inter-chain MHF scenarios. This formulation sets out to investigate (1) the impact of different numbers of SCs implemented, (2) the impact on the ledger consistency for varying failure rate, and (3) the impact of the different prototypical scenarios on the performance. Through different exemplar implementations, this paper demonstrated the proof-of-concept based on practical autonomous vehicles use case, where underlying companies implement their corresponding SCs on distinguished HF networks/channels. In the spirit of evaluating OClient's performance, the experiments adhere to delineate various failure assumptions, client numbers, HF network number, and channel number, shifting SCs implementation from different networks to channels on the source and target HF networks.

Based on the pragmatic analysis, the insights (in both inter-chain and inter-network HF interoperability) depict that AVXT (e.g., in case of the different number of clients, channels, and networks) has sustained a good balance, if compared to an individual HF network, consequently, leaving out a promising solution to address the performance issues among interoperability solutions. The in-depth evaluation through a series of appropriate examples sheds light on the argument of interaction-path considerations, which inscribes an initial attempt to practically assess the performance where MHF transactions path in its primal form is based on the CcG graph. And finally, it would be inapt to say that OClient has addressed every question concerning HF interoperability; however, it can be seen as a prefatory step towards introducing an approach that does not require an additional component to employ a meaningful interaction bridge among fragmented HF networks.

REFERENCES

- [1] Ankur Lohachab, Saurabh Garg, Byeong Kang, Muhammad Bilal Amin, Junmin Lee, Shiping Chen, and Xiwei Xu. Towards interconnected blockchains: A comprehensive review of the role of interoperability among disparate blockchains. *ACM Computing Surveys (CSUR)*, 54(7):1–39, 2021.
- [2] Shilpa Karkeraa. Use cases. In *Unlocking Blockchain on Azure*, pages 229–266. Springer, 2020.
- [3] Ankur Lohachab, Saurabh Garg, Byeong Ho Kang, and Muhammad Bilal Amin. Performance evaluation of hyperledger fabric-enabled framework for pervasive peer-to-peer energy trading in smart cyber-physical systems. *Future Generation Computer Systems*, 118:392–416, 2021.
- [4] Hyperledger. Case Studies. <https://www.hyperledger.org/learn/case-studies/>, 2021. [Online; accessed 13-June-2021].
- [5] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm (extended version), 2013.
- [6] Rafael Belchior, André Vasconcelos, Sérgio Guerreiro, and Miguel Correia. A survey on blockchain interoperability: Past, present, and future trends. *arXiv preprint arXiv:2005.14282*, 2020.
- [7] Pawel Szalachowski. Padv: A blockchain-based tls notary service. In *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 836–843. IEEE, 2019.
- [8] Gyuwon Song, Suhyun Kim, Haejin Hwang, and Kwanhoon Lee. Blockchain-based notarization for social media. In *2019 IEEE international conference on consumer electronics (icce)*, pages 1–2. IEEE, 2019.
- [9] Ron Rivest, Adi Shamir, and David Wagner. Time lock puzzles and timed release cryptography. Technical report, Technical report, MIT/LCS/TR-684, 1996.
- [10] Timothy May. Timed-Release Crypto. <http://cypherpunks.venona.com/date/1993/02/msg00129.html/>, 1993. [Online; accessed 17-June-2021].
- [11] Jiahua Xu, Damien Ackerer, and Alevtina Dubovitskaya. A game-theoretic analysis of cross-chain atomic swaps with htcs. *arXiv preprint arXiv:2011.11325*, 2020.
- [12] Stefan Thomas and Evan Schwartz. A protocol for interledger payments. URL <https://interledger.org/interledger.pdf>, 2015.
- [13] WANCHAIN. Building Super Financial Markets for the New Digital Economy, Whitepaper Version 0.9.1. <https://wanchain.org/files/Wanchain-Whitepaper-EN-version.pdf/>, 2017. [Online; accessed 20-June-2021].
- [14] Johan Kirsten and Hamun Davarpanah. Anonymous atomic swaps using homomorphic hashing. Available at SSRN 3235955, 2018.
- [15] James A Liu. Atomic swaptions: cryptocurrency derivatives. *arXiv preprint arXiv:1807.08644*, 2018.
- [16] Guy Zyskind, Can Kisagun, and Conner Fromknecht. Enigma catalyst: A machine-based investing platform and infrastructure for crypto-assets, 2018.
- [17] Adam Back, Matt Corallo, Luke Dashjr, Mark Friedenbach, Gregory Maxwell, Andrew Miller, Andrew Poelstra, Jorge Timón, and Pieter Wuille. Enabling blockchain innovations with pegged sidechains. URL: <http://www.opensciencereview.com/papers/123/enablingblockchain-innovations-with-pegged-sidechains>, 72, 2014.
- [18] Amritraj Singh, Kelly Click, Reza M Parizi, Qi Zhang, Ali Dehghantanha, and Kim-Kwang Raymond Choo. Sidechain technologies in blockchain networks: An examination and state-of-the-art review. *Journal of Network and Computer Applications*, 149:102471, 2020.
- [19] Johnny Dille, Andrew Poelstra, Jonathan Wilkins, Marta Piekarska, Ben Gorlick, and Mark Friedenbach. Strong federations: An interoperable blockchain solution to centralized third-party risks. *arXiv preprint arXiv:1612.05491*, 2016.
- [20] Liping Deng, Huan Chen, Jing Zeng, and Liang-Jie Zhang. Research on cross-chain technology based on sidechain and hash-locking. In *International Conference on Edge Computing*, pages 144–151. Springer, 2018.
- [21] Peter Gaži, Aggelos Kiayias, and Dionysis Zindros. Proof-of-stake sidechains. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 139–156. IEEE, 2019.
- [22] Sandra Johnson, Peter Robinson, and John Brainard. Sidechains and interoperability. *arXiv preprint arXiv:1903.04077*, 2019.
- [23] Peter Robinson, David Hyland-Wood, Roberto Saltini, Sandra Johnson, and John Brainard. Atomic crosschain transactions for ethereum private sidechains. *arXiv preprint arXiv:1904.12079*, 2019.
- [24] Philipp Frauenthaler, Marten Sigwart, Christof Spanring, and Stefan Schulte. Testimonium: A cost-efficient blockchain relay. *arXiv preprint arXiv:2002.12837*, 2020.
- [25] Ermyas Abebe, Dushyant Behl, Chander Govindarajan, Yining Hu, Dileban Karunamoorthy, Petr Novotny, Vinayaka Pandit, Venkatraman Ramakrishna, and Christian Vecchiola. Enabling enterprise blockchain interoperability with trusted data transfer (industry track). In *Proceedings of the 20th International Middleware Conference Industrial Track*, pages 29–35, 2019.
- [26] Stefan Schulte, Marten Sigwart, Philipp Frauenthaler, and Michael Borkowski. Towards blockchain interoperability. In *International Conference on Business Process Management*, pages 3–10. Springer, 2019.
- [27] Markus Nissl, Emanuel Sallinger, Stefan Schulte, and Michael Borkowski. Towards cross-blockchain smart contracts. *arXiv preprint arXiv:2010.07352*, 2020.
- [28] Marten Sigwart, Philipp Frauenthaler, Christof Spanring, and Stefan Schulte. Towards cross-blockchain smart contracts.
- [29] Muhammad Ahmad Zafar, Falak Sher, Muhammad Umar Janjua, and Salman Baset. Sol2js: translating solidity contracts into javascript for hyperledger fabric. In *Proceedings of the 2nd Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers*, pages 19–24, 2018.
- [30] Gaby G Dagher, Chandra L Adhikari, and Tyler Enderson. Towards secure interoperability between heterogeneous blockchains using smart contracts. In *Future Technologies Conference (FTC)*, pages 73–81, 2017.
- [31] Niclas Kannengießner, Michelle Pfister, Malte Greulich, Sebastian Lins, and Ali Sunyaev. Bridges between islands: Cross-chain technology for distributed ledger technology. In *Proceedings of the 53rd Hawaii International Conference on System Sciences*, 2020.
- [32] Tesnim Abdellatif and Kei-Leo Brousmiche. Formal verification of smart contracts based on users and blockchain behaviors models. In *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5. IEEE, 2018.
- [33] Nicola Atzei, Massimo Bartoletti, Stefano Lande, and Roberto Zunino. A formal model of bitcoin transactions. In *International Conference on Financial Cryptography and Data Security*, pages 541–560. Springer, 2018.
- [34] Christian Cachin, Angelo De Caro, Pedro Moreno-Sanchez, Björn Tackmann, and Marko Vukolic. The transaction graph for modeling blockchain semantics. *IACR Cryptol. ePrint Arch.*, 2017:1070, 2017.
- [35] Jeeta Ann Chacko, Ruben Mayer, and Hans-Arno Jacobsen. Why do my blockchain transactions fail? a study of hyperledger fabric (extended version). *arXiv preprint arXiv:2103.04681*, 2021.
- [36] Peter Puschner and Ch Koza. Calculating the maximum execution time of real-time programs. *Real-time systems*, 1(2):159–176, 1989.
- [37] Eder J Scheid, Timo Hegnauer, Bruno Rodrigues, and Burkhard Stiller. Bifröst: a modular blockchain interoperability api. In *2019 IEEE 44th Conference on Local Computer Networks (LCN)*, pages 332–339. IEEE, 2019.
- [38] Herdus. HERDIUS Next Generation Decentralized Blockchain Financial Infrastructure. <https://www.herdus.com/whitepaper/HerdusWhitepaper1.1.pdf>, 2017. [Online; accessed 14-July-2021].