Towards Connected Blockchains: An In-Depth Analysis and Improvement of Cross-Chain Communication Protocols

A Thesis submitted in fulfilment of the requirements for
the candidature of

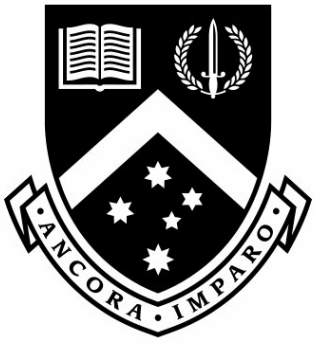
Doctor of Philosophy in Information Technology

By

João Otávio Massari Chervinski

Supervisor: Jiangshan Yu

Co-supervisors: Rafael Dowsley and Xiwei Xu



Department of Software Systems and Cybersecurity

Faculty of Information Technology

Monash University

September 14, 2023

Copyright Notice

I certify that I have made all reasonable efforts to secure copyright permissions for third-party content included in this thesis and have not knowingly added copyright content to my work without the owner's permission.

Abstract

During its early years, the promising potential of blockchains led industry and academia to rapidly invest in the research and development of the technology. However, due to the absence of standards for blockchain design, those efforts were largely uncoordinated, resulting in a landscape of isolated and fundamentally different systems. As the adoption of blockchain systems surged, the ability for them to exchange information was deemed crucial for the success of the technology. Yet, due to their fundamental differences, the transfer of information between blockchains, known as cross-chain communication, involves several challenges. Various strategies were proposed to enable cross-chain communication, but those proposals have unique requirements and objectives and are often mutually incompatible. This complicates the selection of suitable approaches to connect blockchains based on one's requirements. Furthermore, it delays the identification of issues that arise with broader adoption, such as vulnerabilities and design flaws.

This thesis bridges those gaps by systematically studying cross-chain communication protocols, their applications, performance and design limitations. In our first work, we categorize and analyze ten leading proposals to connect blockchains. This work provides an holistic view of the state of practice to help developers choose cross-chain communication strategies according to their needs. In our second work, we perform the first comprehensive performance evaluation of the Inter-blockchain Communication Protocol, a state-of-the-art proposal to transfer arbitrary data between blockchains. Our findings include bottlenecks, technical challenges and also reveal the inability of the protocol to scale given its current design. In our third work, we identify and explore several limitations in the design of cross-chain relayers, applications that are essential for the transport of data across major blockchain platforms such as Cosmos and Ethereum. We address those limitations in two ways: we propose a novel protocol to coordinate independent cross-chain data relayers; and we provide a detailed discussion on the trade-offs inherent to the design of cross-chain data relayers for permissionless settings.

List of Publications

The following works have been published and are included in the contents of this thesis:

- João Otávio Chervinski, Jiangshan Yu, Xiwei Xu. "***Characterizing Blockchain Interoperability Systems From an Architecture Perspective***". The 8th EAI Smart City 360° International Summit, 2022.

  Available at: https://doi.org/10.1007/978-3-031-06371-8_33.

- João Otávio Chervinski, Diego Kreutz, Xiwei Xu, Jiangshan Yu. "***Analyzing the Performance of the Inter-blockchain Communication Protocol***". 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 2023.

  Available at: https://doi.org/10.1109/DSN58367.2023.00026

- João Otávio Chervinski, Diego Kreutz, Jiangshan Yu. "***Towards Scalable Cross-Chain Messaging***". The 6th IEEE International Conference on Blockchain, 2023.

  Available at: https://arxiv.org/pdf/2310.10016.pdf

The following works were conducted and published during my candidature, but are outside of the scope of this thesis:

- João Otávio Chervinski, Diego Kreutz, Jiangshan Yu. "***Analysis of Transaction Flooding Attacks Against Monero***". The 3rd IEEE International Conference on Blockchain and Cryptocurrency, 2021.

  Available at: https://doi.org/10.1109/ICBC51069.2021.9461084

- Sina Aeeneh, João Otávio Chervinski, Jiangshan Yu, Nikola Zlatanov. "***New Attacks on the Untraceability of Transactions in Cryptonote-style Blockchains***". The 3rd IEEE International Conference on Blockchain and Cryptocurrency, 2021.

  Available at: https://doi.org/10.1109/ICBC51069.2021.9461130

Thesis including published works declaration

This thesis includes *3* original papers published in peer reviewed conferences. The core theme of the thesis is *Cross-chain communication*. The ideas, development and writing up of all the papers in the thesis were the principal responsibility of myself, the student, working within the *Faculty of Information Technology* under the supervision of *Jiangshan Yu*.

The inclusion of co-authors reflects the fact that the work came from active collaboration between researchers and acknowledges input into team-based research. In the case of *Chapters 3, 4 and 5* my contribution to the work involved the following:

Thesis Chapter	Publication Title	Status	Nature and % of student contribution	Co-author name(s), nature and % of contribution	Co-author(s), Monash student (YES/NO)
3	Characterizing Blockchain Interoperability Systems From An Architecture Perspective	Accepted	Literature review, framework design, system classification and analysis, writing, 85%	1) Jiangshan Yu (10%): framework design and input into manuscript 2) Xiwei Xu (5%): discussions and input into manuscript	None
4	Analyzing the Performance of the Inter-blockchain Communication Protocol	Accepted	Literature review, framework and tool design, tool implementation, experiments, data analysis, writing, disclosure of results to developers, 90%	1) Diego Kreutz (2.5%): discussions and input into manuscript 2) Xiwei Xu (2.5%): discussions 3) Jiangshan Yu (5%): discussions, framework design and input into manuscript	None
5	Towards Scalable Cross-chain Messaging	Accepted	Literature review, identification of open challenges, protocol design, trade-offs analysis, writing, 92.5%	1) Diego Kreutz (2.5%): discussions, and input into manuscript 2) Jiangshan Yu (5%): discussions, protocol design and input into manuscript	None

I have renumbered and renamed sections of published and submitted papers in order to generate a consistent presentation within the thesis. I have also made the following modifications to the contents of the papers included in this thesis:

- **Chapter 3 - Characterizing Blockchain Interoperability Systems From An Architecture Perspective**:

  1. Modified the contents of *Section 3.1 Introduction* to reduce content over-

lap with the remainder of the thesis.

2. Expanded upon the contents of *Section 3.4 Analysis of Blockchain Interoperability Solutions* to provide additional details and references.
3. Extended *Section 3.5 Related Work* to offer a more detailed discussion of the included literature.
4. The contents of *Section 3.6 Summary* were revised to present an overview of the work undertaken in this chapter. A detailed discussion on the significance of its contributions is provided in *Chapter 6 Conclusions*.

- **Chapter 4 - Analyzing the Performance of the Inter-blockchain Communication Protocol**:

1. Modified the contents of *Section 4.1 Introduction* to reduce content overlap with the remainder of the thesis.
2. Extended *Section 4.6 Related Work* to provide a more comprehensive review of related work on blockchain performance evaluation.
3. The contents of *Section 4.7 Summary* were revised to present an overview of the work undertaken in this chapter. A detailed discussion on the significance of its contributions is provided in *Chapter 6 Conclusions*.

- **Chapter 5 - Towards Scalable Cross-Chain Messaging**:

1. Modified the contents of *Section 5.1 Introduction* to reduce content overlap with the remainder of the thesis.
2. The contents of *Section 5.8 Summary* were revised to present an overview of the work undertaken in this chapter. A detailed discussion on the significance of its contributions is provided in *Chapter 6 Conclusions*.

I hereby declare that this thesis contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

Student name: João Otávio Massari Chervinski

Student Signature:

Date: September 14, 2023

I hereby certify that the above declaration correctly reflects the nature and extent of the student's and co-authors' contributions to this work. In instances where I am not the responsible author I have consulted with the responsible author to agree on the respective contributions of the authors.

Main supervisor name: Jiangshan Yu

Main supervisor signature:

Date: September 14, 2023

Acknowledgements

I would like to thank CSIRO for sponsoring me with a Data61 scholarship and enabling me to pursue this degree. I would also like to thank the Faculty of Information Technology for the additional support provided to students impacted by the COVID-19 pandemic.

For my supervisors, Jiangshan Yu and Xiwei Xu, I am very grateful. Not only they provided me with support and guidance throughout my candidature, they also believed in me even when I struggled to do so. They showed kindness and patience by giving me time to learn, grow and explore my academic interests.

I am fortunate to have Diego Kreutz as a mentor and as a friend. He went out of his way to provide great advice on both professional and personal matters throughout my PhD journey. He was always ready to listen and share his experience in a relatable manner. It was thanks to his words of encouragement that I was able to persist through the difficult moments of my candidature.

I will be forever grateful for my mom and the endless love and support she provided to me even from half the world away. No matter the time, she always promptly responded, listened and did her best to comfort me. She gave me a reason to fight for.

I am extremely thankful for the love, patience and understanding I received from my partner Jesslyn Wijaya. Together we celebrated my victories and navigated through the hard times. She is one of the pillars that sustained me during this journey. Thank you for being besides me in every step of the way.

Lastly, but not least, I would like to acknowledge all the friends and colleagues whose names are not explicitly mentioned herein. Every discussion I had and piece of advice I received, no matter how small, played a role in getting me where I am today. I am deeply grateful for all of you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Blockchain is a transformative technology characterized by the provision of decentralization, transparency and immutability. As a result, blockchain-based systems are seen as a promising solution to address outstanding challenges in different domains such as record-keeping, product provenance tracking, identity management, secure voting and finance [1]–[8]. In 2023 the global blockchain market was valued at USD 17.46 billion, and it is predicted to grow significantly as blockchains continue to bring value across diverse industries [9]. This growth is fueled in part by the technology's role in enabling over 9,000 decentralized digital currencies, which collectively account for a market capitalization of USD 1.05 trillion.[1]

Due to the promising potential of blockchains, both industry and academia invested significantly in the research and development of the technology following its launch in conjuction with Bitcoin [10] in 2009. As a result, various blockchain platforms and applications were proposed [11]. However, due of the absence of standards and guidelines for blockchain design at the time, those initial efforts were largely uncoordinated. This led to the creation of a fragmented landscape composed of systems that were not only isolated but also built upon fundamentally different principles [12], [13]. Yet, the technology continued to thrive.

As the years progressed and the technology became more mature, the adoption of blockchain-based systems surged. With broader adoption, the ability for blockchains to exchange information, commonly referred to as cross-chain communication, was deemed crucial for the long-term success of the technology [14], [15]. Connecting blockchains allows them to mutually exchange value and introduces possibilities such as transferring digital assets between different blockchains systems, deploying and synchronizing smart contracts across multiple blockchain networks, querying external systems and offloading computational tasks to improve blockchain performance [16], [17]. Furthermore, cross-chain communica-

[1]Up to date as of September 8, 2023. Retrieved from https://coinmarketcap.com/

tion is increasingly being recognized as a potential solution for the improvement of blockchain scalability [18]. However, despite the technology's positive prospects, achieving this integration involves significant challenges.

Due to the problem of fragmentation and the fundamental differences among blockchains, transferring information between them requires the integration of systems built upon different architectures, protocols, security guarantees and trust models [19]. Various approaches were proposed to address those challenges, but different solutions have different requirements, objectives and design trade-offs. For example, sidechains are constructions that enable assets to be transferred back and forth between a pair of blockchains, but must be designed on a case-by-case basis [20]–[24]. Blockchain notaries enable the verification of external events, but often introduce an additional layer of trust [25]. Atomic Cross-Chain Swaps [26]–[30] achieve cross-chain communication by coordinating the atomic execution of transactions across separate blockchains, but lack the expressive power to support all but the simplest types of operations [31].

This variety of solutions, built based on different designs, technologies and objectives, exacerbates the problem of blockchain fragmentation. Choosing the most appropriate strategy to connect different blockchains is a complex task. This is especially true given the fast-changing nature of the field and the scarcity of up-to-date documentation [32]. Furthermore, there is a lack of widely accepted standards for cross-chain communication and, despite ongoing work on the integration of distinct protocols [33], [34], most existing solutions are unable to understand and interpret external data. As a result, adoption is divided across multiple proposals, delaying the discovery of design and implementation issues that would otherwise become apparent through widespread use. Identifying and addressing those problems at the earliest opportunity is critical, given that vulnerabilities in cross-chain communication systems can have cascading impacts across multiple platforms and applications [35].

1.1 Problem Statement

A wide variety of strategies have been proposed to enable blockchains to work together. However, there is no clear guidance on how to select a suitable approach

from the existing solutions, that is, an approach that best aligns with one's requirements and use-cases. Such guidance is essential to prevent resources from being spent on the development and integration of systems that either exceed or fail to meet the necessary requirements. For example, consider a company that wishes to utilize blockchains to engage in business with another company. If the business arrangement requires numerous small payments over an extended period, an approach optimized for small, low fee transactions is more appropriate than one that is designed for general purpose data transfers. While prior works have provided extensive formalization [36], [37], [31], [38] and classification of those systems [25], [36], [32], there is a lack of comparative evaluations on the advantages and disadvantages of different interoperability system architectures. This leads to the following problem:

- **Problem 1:** *There is a gap in the literature concerning the evaluation of the features, advantages and limitations of different blockchain interoperability system architectures.*

The work conducted towards addressing Problem 1 informed the second stage of our study. Existing approaches to blockchain interoperability lack support to a common and widely adopted messaging format. Moreover, those approaches are often limited to token transfers. Our analysis indicates that in order for blockchains to fully benefit from interoperability, there is a need for a strategy capable of transferring arbitrary data between heterogeneous blockchains. One proposal to address this problem stands out, the Inter-blockchain Communication Protocol (IBC). IBC is an end-to-end protocol that supports the transfer of arbitrary data, thus enabling the execution of custom cross-chain logic. IBC is currently being used to connect over 60 blockchains and has facilitated the transfer of over USD 30.3 billion worth of assets in 2022 [39].

Despite the growing popularity of IBC, there is a lack of empirical studies focused on its performance and limitations, as well as those of other cross-chain protocols in general. Existing literature shows that performance evaluations of blockchain systems can reveal problems related to their design and implementation [40]–[54], [44]. In addition, a comprehensive system analysis can also identify

security vulnerabilities and assist in the mitigation of attacks [55]–[57]. This is especially important in light of recent events, where the exploitation of vulnerabilities in cross-chain systems led to the theft of USD 2.5 billion [58]. As the need for cross-chain communication grows, existing solutions must be thoroughly evaluated to ensure they continue to work as intended while meeting the increasing demand. This results in the second problem approached in this thesis:

- **Problem 2:** *There is a lack of empirical studies on the performance, limitations and security of cross-chain communication protocols.*

Informed by our investigations into Problem 2, we have discovered several limitations in the design of permissionless relayers, off-chain applications that are essential for the transport of data in cross-chain protocols such as IBC [35]. Relayers are also widely used for the provision of cross-chain services in platforms such as Ethereum and Polkadot. Due to the lack of empirical research on cross-chain communication protocols, those problems have not been previously explored. Addressing these challenges and improving the current design of permissionless relayers has the potential to enhance the performance and reliability of cross-chain communication across several blockchain platforms. Based on these observations, we identify the third problem investigated in this thesis:

- **Problem 3:** *There are unexplored design limitations that adversely impact the performance and reliability of permissionless cross-chain relayers.*

1.2 Research Objectives

Based on the problems we have identified, we define three key research objectives for this thesis:

- **Research Objective 1:** *Conduct an in-depth analysis of different cross-chain communication system architectures to identify their unique features, advantages and disadvantages, thus providing an holistic view to assist in the selection of suitable cross-chain communication solutions.*

- **Research Objective 2:** *Conduct a comprehensive evaluation of the Inter-blockchain Communication Protocol, focusing on its performance, design shortcomings and deployment challenges, thereby identifying potential improvements and contributing to the advancement of the state of the art of cross-chain communication.*

- **Research Objective 3:** *Investigate the design limitations of permissionless cross-chain relayers, identify their impact on performance and security, and propose mechanisms to improve upon the current models, therefore contributing to the enhancement of protocols that rely on relayers for the transport of data.*

1.3 Contributions

In this thesis we contribute to the practice and theory of cross-chain communication by categorizing, evaluating and improving proposals to transfer information between blockchains. First, we classify ten leading blockchain interoperability solutions into four distinct categories based on their system architecture (detailed in *Chapter 3*). Following, we propose a framework to evaluate existing solutions according to five distinct architectural aspects. We utilize our framework to analyze blockchain interoperability solutions belonging to each of the four proposed categories and present a comparative analysis of their advantages and disadvantages.

We continue the work of this thesis by presenting the first comprehensive empirical evaluation of the Inter-blockchain Communication protocol, one of the leading proposals to facilitate the transfer of arbitrary data between blockchains (detailed in *Chapter 4*). Through substantial engineering efforts, the IBC protocol has been successfully implemented in diverse blockchain platforms, such as Penumbra, Hyperledger Fabric and Corda. However, the protocol is still primarily used in production by its proponent platform, the Cosmos Network, due to its native support to the protocol. In this light, we use IBC to perform cross-chain transactions between two Cosmos Gaia blockchains and conduct several experiments to measure the protocol's performance and scalability. We elaborate on our choice of Gaia blockchains and the generalizability of our results in *Chapter 4*. Through a detailed analysis of the experimental results, we observe that the protocol is unable to scale due to design constraints. Moreover, we explore practical challenges that

add complexity to the deployment and utilization of IBC.

Lastly, we build upon the results from our evaluation of the IBC protocol and explore the limitations of permissionless cross-chain data relayers, which are employed by various cross-chain protocols for the transport of information (detailed in *Chapter 5*). We identify three previously unexplored design challenges that adversely impact system performance, scalability and security. We take the first step towards addressing the identified performance and scalability issues by proposing a novel protocol to coordinate independent relayers. In addition, we provide a detailed discussion concerning the design trade-offs that impact different aspects of cross-chain relayers, including performance, reliability, task allocation and incentive mechanisms.

Chapter 2

Literature Review

Due to the emerging nature of the research field under investigation in this thesis, there is a lack of consistency in terms and vocabulary across different works [18]. The terms *blockchain interoperability* and *cross-chain communication* are often used interchangeably, but can be used to represent different concepts. Blockchain interoperability is a broader term that represents the ability of blockchains to work together, including technical, legal and organizational aspects [59]. In contrast, cross-chain communication concentrates on the technical aspects and mechanisms through which different blockchains can exchange information. Given these distinctions, we organize this literature review into two parts: the first part presents works that explore blockchain interoperability as a broader concept, the second part focuses on works that approach the topic strictly from a technical or theoretical perspective.

This chapter presents a brief overview of the existing literature for two primary reasons. First, *Chapter 3* presents an in-depth analysis and discussion of existing proposals to connect blockchains. Second, *Chapters* 3, 4 and 5 each include a *Related Work* section, which discusses literature that is closely related to their respective subjects. Thereby, we limit this chapter to the discussion of literature that is not reviewed in the chapters that follow.

2.1 Blockchain interoperability

Kannengiesser et al. [60] review proposals to connect blockchains and provides a synthesis of the properties and characteristics of blockchain interoperability solutions. The authors classify blockchain interoperability solutions into four categories according to their communication patterns and discuss use cases for blockchain interoperability technologies in general.

Belchior et al. [18] present a detailed discussion on the evolution of blockchain systems. The authors examine the role of blockchain interoperability in the de-

velopment of the technology and emphasize the importance of this research field for addressing outstanding challenges such as scalability. This work discusses the distinct layers that compose blockchain interoperability solutions, including regulatory and governance aspects.

Schulte et al. [12] examine the problem of blockchain fragmentation and elaborate on the critical role of interoperability in enabling an open landscape where users can interact across different platforms. The authors discuss the state of the art in blockchain interoperability and outline research directions that must be pursued in order to advance the field.

Lohachab et al. [61] presents a systematic review of blockchain interoperability and its different aspects, requirements and implementations. The authors propose a taxonomy for blockchain research and a multi-layer architecture to achieve interoperability among heterogeneous blockchains. This work also provides insights into state-of-the-art projects to enable blockchain interoperability and puts forward directions for future research.

Belchior et al. [59] propose an interoperability assessment framework for blockchains, allowing the comparison of potentiality, compatibility and performance among solutions. The authors study the main interoperability layers present in traditional models for computer systems interoperability and present a generic framework for the design of blockchain interoperability mechanisms.

2.2 Cross-chain communication

Zamyatin et al. [36] provide a systematic analysis of cross-chain communication protocols and formalize the problem of cross-chain communication [36]. The authors draw a comparison between the process of establishing cross-chain communication and the Fair Exchange problem, arguing that cross-chain communication is impossible without the help of a trusted third party. This work also introduces a framework for the evaluation of cross-chain communication protocols based on their trust models and design rationale.

Herlihy et al. [31] proposes and formalizes cross-chain deals, an abstraction to enable the creation of complex distributed deals across different blockchains in adversarial settings. The proposed notion improves upon classical cross-chain co-

ordination protocols by allowing flexibility of outcomes and adding the possibility of arbitrage. The authors describe novel safety and liveness properties that are required to conduct cross-chain deals in adversarial settings. Furthermore, they propose two alternative protocols for the implementation of cross-chain deals, the first is based on synchronous communication, and the second is based on semi-synchronous communication.

Zamyatin et al. [62] introduce XCLAIM, the first generic framework for achieving trustless cross-chain asset exchanges. The authors leverage publicly verifiable smart contracts to eliminate the need for trusted third-parties and employ relays to transport state information between blockchains. An instantiation of the proposed framework is implemented and evaluated to compare its performance against other cross-chain communication protocols.

Lafourcade and Lombard-Platet [37] provide a formalization of the problem of transferring information between separate blockchains. Based on their theoretical analysis, the authors argue that cross-chain communication is impossible according to the classical definition of blockchains and can only become possible by relaxing existing definitions.

Chapter 3

Characterizing Blockchain Interoperability Systems from an Architecture Perspective

Various strategies were proposed to enable interoperability among different blockchain systems. Yet, the process of choosing an approach that aligns with one's requirements and use-cases is made complex by this diversity. This challenge is further exacerbated by the fact that, despite ongoing work to integrate existing interoperability solutions [33], [34], they are still largely incompatible with one another. Specifically, this means that they lack the mutual support for a common data format, making them unable to understand and interpret data generated by each other. This raises the question: *how can one choose the most appropriate strategy for connecting fundamentally different blockchains, especially when the available solutions themselves are incompatible?*

To address *Research Objective 1*, this chapter categorizes blockchain interoperability solutions into four distinct categories based on their system architecture. It then proposes an interoperability solution evaluation framework and applies it to present a comparative analysis of solutions belonging to each of these categories.

3.1 Introduction

Interoperability has been deemed critical for the advancement of blockchain technology [63], [15], [14]. Consequently, the interest in this area of research is growing rapidly. When blockchains are interoperable, they become capable of mutually exchanging value, not only in the form of token transactions, but also through external data queries, offloading of tasks to improve system performance, and the ability to combine functionalities from multiple platforms into a single application, a concept called composability. For example, a user can leverage interoperability to request the execution of a transaction in an external blockchain without needing to create a new wallet and swap tokens back and forth through

an exchange. This streamlines the operation and eliminates the often substantial fees associated with the use of exchanges. Another example of a current real-world application of interoperability is shared security, available to blockchains in the Cosmos Ecosystem. This feature allows blockchains to borrow validator votes and security from each other, increasing their resiliency against attacks.

Multiple academic proposals [23], [64], [20], [38], [62], [65] and industry projects [66], [67], [24], [68]–[70] have been proposed to address this challenge. However, differences in security mechanisms, protocols and scripting capabilities can all pose challenges for approaches that attempt to enable cross-chain communication. For that reason, many interoperability solutions are developed in an ad-hoc manner and target specific use-cases [25], [36].

While existing research provides comprehensive review on the existing systems [25], [36], [63], a holistic view from the architecture perspective remains unclear. Such a big picture is important to aid developers in the process of understanding the different characteristics of the existing solutions and selecting the ones which meet their needs. In this work, we bridge this gap by characterizing blockchain interoperability solutions from an architecture perspective.

The main contributions of this work are summarized as follows:

- ***Contribution 1*** *(Section 3.2)*: We classify ten leading proposals to enable blockchain interoperability into four distinct categories based on the characteristics of their architecture.

- ***Contribution 2*** *(Section 3.3)*: We propose an evaluation framework to assist developers in selecting an interoperability solution from existing systems that aligns with their specific requirements and use-cases. Our framework is comprised by five different criteria concerning the features that enable blockchain interoperability.

- ***Contribution 3*** *(Section 3.4)*: We provide a comparative analysis between the proposed interoperability system categories and highlight the advantages and disadvantages inherent to the systems that belong in each of them.

Our framework provides design guidance to assist developers in selecting interop-

erability solutions that best align with their deployment requirements. Our analysis provides support in the assessment of the benefits and drawbacks of interoperability system categories and individual systems. Throughout this chapter we refer to blockchain interoperability and cross-chain communication interchangeably.

This work is organised as follows. Section 3.2 characterizes the architectures of cross-chain communication systems. Section 3.3 presents our proposed evaluation framework and Section 3.4 presents a detailed analysis of the system categories and the ten leading systems. Finally, Section 3.5 presents related work and Section 3.6 presents a summary of this chapter's work.

3.2 Blockchain Interoperability Categories

In this section we propose four categories to characterize and distinguish blockchain interoperability systems. We define blockchain interoperability as *the ability for different blockchains to interact via a semantically compatible format*, where semantically compatible means the data format can be understood and interpreted in the same manner by the systems that are interacting.

The proposed categories were derived from the analysis and comparison of both the architecture and functionality provided by ten leading blockchain interoperability systems. The systems we selected are either from academia [38], [64], [23], [69], [65] and known by the research community or well-known industrial projects [66], [67], [24], [70], [71]. We have selected those systems as they provide a diverse representation of the state-of-the-art approaches to enable interoperation between blockchains. A detailed analysis of each category and its corresponding systems is provided in Section 3.4. Through our analysis of the selected systems we identified four distinct categories of blockchain interoperability systems:

Blockchain ecosystems manage networks of independent blockchains that are interconnected through a central chain or hub that forwards information. Ecosystems enable native cross-chain communication between blockchains inside the network and can also allow communication with external blockchains through bridging systems. Blockchain ecosystems have their own native tokens which can be used for stake-based consensus and for voting in feature proposals. Those systems can be either permissioned or permissionless. *Cosmos* [66] and *Polkadot* [72] are

classified in this category.

Relay-Based systems enable a blockchain (called the destination chain) to fetch and validate data coming from a source blockchain. The destination blockchain needs to implement the consensus verification mechanism of the source chain in order to verify the validity of the information received through the relay. A single relay system allows the destination chain to verify events that took place in a source chain, however the contrary is not necessarily true and may require two relays, one in each chain. This category of system includes *BTC Relay* [71], *ETH Relay* [69] and *Proof-of-Burn (PoB)* [64].

Sidechain-based systems are constructions that connect two blockchains and allow them to transfer assets back and forth between each other via a process called a two-way peg [20]. Sidechain-based systems aim to enable users of a blockchain to transfer their assets to another blockchain, called a sidechain, and use those assets to access features and applications not available in the source blockchain. The systems included in this category are *Liquid Network* [24], *Proof-of-Stake sidechains* [23] and *Proof-of-Work sidechains* [64].

Peer-based Systems enable users to exchange assets across distinct blockchains by participating in deals with other users. Peer-based systems exchange information by leveraging protocols that create agreements based on hashed timelock contracts (HTLCs). The systems that belong to this category are *Atomic Cross-Chain Swaps (ACCSs)* [38] and *Interledger* [70].

3.3 Evaluation Framework

In this section, we propose an evaluation framework composed of five criteria to understand and analyze the architecture of blockchain interoperability systems. We have specifically selected those five criteria as they are key components in the process of transporting and validating data, which is essential for establishing cross-chain communication. Given our focus on the mechanics and capabilities of information transfer between blockchains, we limit the scope of our work by not including security as a separate criterion. This allows our framework to focus on the functional and operational aspects of interoperability and avoid an overly broad range of criteria. The proposed framework provides a concise overview of

cross-chain communication capabilities of the selected systems, evaluating them based on the following criteria:

3.3.1 Direction of communication

Direction of communication defines in which direction blockchain interoperability systems are able to transfer information. It can be either unidirectional or bidirectional. Systems with unidirectional data transfer only allow the destination chain to fetch data from the source chain, whereas systems with bidirectional data transfer enables both destination chain and source chain to send and receive data from each other.

3.3.2 Communication channel

The communication channel refers to the medium used to send the information required to execute the cross-chain operation. Six out of the ten interoperability solutions analyzed in this chapter lack support for arbitrary data transfers, therefore, for consistency, we evaluate communication strategies based on token transfer operations. We divide cross-chain operations into four phases: *setup*, *value transfer*, *claim transfer* and *settlement*.

The *setup* phase is used to exchange information regarding the structure of the deal such as intention to transfer, amount, addresses, conditions and time constraints. Those details should be analyzed when deciding whether to follow through with the deal or to abort the operation.

If participants agree on the parameters of the deal they move on to the *value transfer* phase. In this phase transactions are released to the counterparties or published in the blockchain. This may occur between multiple pairs of users (or hops) in scenarios where multiple participants route payments between blockchains along the way to a destination address.

After the involved parties execute their part of the deal, they enter the *claim transfer* phase, where a verifiable claim must be relayed to the counterparty in order to prove that an agreed upon action was performed and registered in the blockchain.

Finally, during the *settlement* phase, verifiable claims generated in the previous steps are validated and the final transaction in the cross-chain operation is committed to the blockchain, updating the state of the ledger. This may take from a few seconds to hours, depending on the confirmation latency of the involved blockchains. For each phase, one of the following channels can be used:

• *On-chain:* When a phase of the cross-chain operation takes place inside the blockchain, it is considered an on-chain communication.

• *Off-chain channels:* When a phase of the cross-chain operation takes place in a medium predefined by the protocol but is not recorded on the blockchain, we say it requires off-chain channels.

• *Out-of-bound channels:* When a phase of the cross-chain operation requires a communication channel that is not defined by the protocol (such as messaging apps, email, phone calls or in-person), it is considered an out-of-bound channel.

3.3.3 Type of Exchanged Information

The type of exchanged information defines the type of data that can be sent across chains, including digital tokens and arbitrary data.

• *Digital token:* Digital tokens represent cryptocurrencies and are commonly represented as a key-value pair, where the key is the token address and the value is the amount of token balance (i.e., the number of tokens contained in that address) like in the UTXO model [10] or account model [73].

• *Arbitrary data:* Arbitrary data allows non-token data, such as the blockchain state, to be exchanged across chains. This is particularly useful for non-cryptocurrency blockchains, such as Hyperledger Fabric [74].

3.3.4 Verifiable Claims

Verifiable claims define how a blockchain system proves to another blockchain that an event, e.g, the acceptance of a transaction, has taken place. Events must be verified before committing operations to the blockchain in order to prevent malicious behavior. The verification of claims, as formally defined in a recent work [75], is a challenge for cross-chain communication mechanisms. A system can prove that

data has been added to the chain through the following types of claims:

• *Validator signature:* A validator or set of validators are trusted to provide a proof about the internal state of a (commonly permissioned) blockchain. The proof could be in the form of a threshold number of signatures (or an aggregated signature to reduce the proof size) from the set of validators [75]–[77].

• *SPV proof:* Simplified Payment Verification (SPV) enables light clients (such as mobile wallets) to verify that a transaction has been included in the blockchain without downloading the entire blockchain history.

• *Pre-commitment:* A pre-commitment is a cryptographic construction which guarantees that a certain action, such as releasing a transaction, is going to be carried out once a condition is fulfilled. Pre-commitments are used to create agreements which prove that transactions are going to be released when all participants of the cross-chain operation commit to executing the deal.

3.3.5 Validator

A validator is an entity or group of entities that participate in the blockchain's consensus process, where they are responsible for verifying, validating, and adding transactions to the blockchain. In the context of blockchain interoperability, validators specifically verify the validity of a verifiable claim generated in another blockchain. The way in which validators operate may vary according to the consensus algorithm used by the blockchains in which they participate. They may have elevated privileges, in which case the security of the system relies on the integrity of the set of validators, or may be selected in a trustless manner, leading consensus to be achieved when a majority of the blockchain users agree with the state changes proposed by the validator.

• *Permissionless validator:* We consider validators to be permissionless if they are (possibly weighted) randomly chosen from the set of blockchain participants. They are permissionless as any blockchain user can join the validation process. Systems that employ permissionless validators [10], [78], [73], [79] incentivize users to take part in the validation process through financial compensation.

• *Permissioned validator:* We consider validators to be permissioned if they are

chosen from a fixed and pre-determined subset of blockchain participants. Unlike in permissionless systems, not every blockchain user can be a validator in a permissioned system.

3.4 Analysis of Blockchain Interoperability Solutions

Distinct blockchain interoperability systems achieve their objectives by employing cross-chain communication mechanisms that differ in their levels of complexity. For that reason, distinct systems may present different strengths and weakness when compared to each other. In this section we classify ten leading blockchain interoperability systems (as presented in Table 3.1) according to our proposed evaluation framework. We also analyze the advantages and disadvantages of each proposed category using attributes based on those defined by the ISO/IEC 25010 software quality model [80]. Lastly, we provide a comparative analysis between the systems in each category. Rather than discussing the same set of properties for each analyzed system, we discuss only the unique properties that distinguish the systems apart from each other.

Table 3.1: Summary and classification of blockchain interoperability solutions.

System attributes		Blockchain Ecosystems		Relay-based systems			Sidechain-based systems			Peer-based systems	
		Cosmos	Polkadot	BTC Relay	ETH Relay	Proof-of-Burn	Liquid Network	PoS Sidechains	PoW Sidechains	Atomic Swaps	Interledger
Direction of communication	Unidirectional			X	X	X					
	Bidirectional	X	X				X	X	X	X	X
Communication Channel*	Setup	●	●	●	●	●	◐	●	●	○	◐
	Value transfer	●	●	●	●	●	●	●	●	●	○
	Claim transfer	●	●	○	○	○	◐	●	○	○	○
	Settlement	●	●	●	●	●	●	●	●	●	●
Verifiable Claims	Validator signature		X				X	X			
	SPV proof	X		X	X	X			X		
	Pre-commitment									X	X
Validator	Permissionless		X	X	X			X	X		
	Permissioned						X				
	Blockchain dependent	X				X				X	X
Type of exchanged information	Digital tokens	X	X	X	X	X	X	X	X	X	X
	Arbitrary data	X	X					X	X		

* On-Chain: ●; Off-Chain: ◐; Out-of-Bounds: ○.

While certain well-known interoperability solutions like XCLAIM [62] and cross-chain deals [31] are innovative and important, they significantly differ from the core characteristics of other popular systems. To maintain a coherent analysis, we focus on evaluating and discussing systems composed by attributes that are widespread among interoperability solutions. Therefore we do not include those systems in our classification and evaluation. This approach allows our analysis

and discussions to be applicable to the systems that are currently most prevalent in the blockchain interoperability landscape while keeping our evaluation framework concise.

3.4.1 Blockchain Ecosystems

Blockchain ecosystems (illustrated in Figure 3.1) are networks that host independent interconnected blockchains. Ecosystems are organized in a topology in which multiple blockchains are connected to one central blockchain. The central blockchain serves as a connector that forwards information across connected blockchains (a.k.a. internal blockchains). Internal blockchains can also establish communication with external chains, i.e., those not deployed inside the ecosystem.

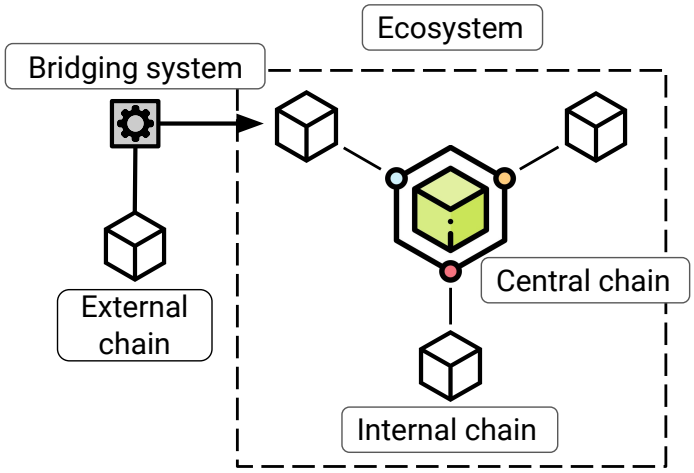


Figure 3.1: Architectural overview of blockchain ecosystems.

However, this requires the development of bridging systems to adapt the information coming from external chains in such a way that it can be understood by the internal chains.

- **Strengths of blockchain ecosystems:**

Compatibility (Interoperability): Internal chains can participate in cross-chain operations with other internal chains connected to the central chain without requiring adaptations or additional configuration.

• **Weaknesses of blockchain ecosystems:**

Portability (Installability): Developing and deploying a blockchain inside an ecosystem is hard and time consuming. Blockchains need to be compatible with the requirements of the ecosystem, e.g, a specific consensus engine. In order to deploy blockchains inside an ecosystem as an internal chain, aspects such as block proposal strategy, validation incentives and feature support might need to reconsidered.

Portability (Adaptability): Blockchains deployed inside an ecosystem only natively suppport communication with other internal chains. If communication with an external chain is needed, a bridging system is required.

Cosmos is an ecosystem of interconnected independent blockchains. In Cosmos, there are zones and hubs. Zones are regular blockchains but can also choose to become hubs, which are a special type of blockchains that connects zones together. Zones can communicate with a theoretically unlimited number of other zones as long as they are connected to a common hub. The Cosmos ecosystem is permissionless, allowing anyone to deploy a blockchain in the network. Zones are powered by Tendermint BFT [1]. Cosmos has a native token called ATOM, which is used to secure the Cosmos Hub through a staking-based validator selection.

Polkadot is a network of interconnected blockchains. In the network there are multiple parachains and one relay-chain. Parachains are independent blockchains that run in parallel and the relay-chain is a central chain that validates blocks proposed by the parachains and coordinates the entire network. The relay-chain provides a shared view of every parachain's state to the ecosystem. Polkadot has a native token that is used for staking, called the DOT. Staking DOTs is a requirement for being a relay-chain validator and is also required for joining the auction process that leases parachain slots in the network. Polkadot provides a protocol called Cross-Chain Message Passing (XCMP) to enable the transfer of assets and arbitrary data between parachains. Parachains need to open one channel for sending messages and another one for receiving messages for each parachain they want to communicate with. Those channels require funding using DOT tokens, which

[1] https://tendermint.com/core/

are returned when the channels are closed[2].

Analysis of blockchain ecosystems. The two well-known systems in this category are Cosmos and Polkadot. Both systems aim to reduce the difficulty and time required to develop and deploy blockchains inside their networks by providing frameworks for blockchain development. Cosmos offers the Cosmos SDK and Polkadot offers the Substrate framework. Both are modular frameworks that provide developers with pre-built modules that can be applied to blockchains, eliminating the need for designing basic functionalities such as governance, staking and token distribution. However, the utilization of these frameworks also introduces trade-offs. For instance, they may limit the flexibility of individual blockchain implementations by constraining developers to the pre-defined modules and architectural choices of the frameworks.

Polkadot offers a limited number of 100 slots in which parachains can be deployed[3]. Consequently, users are required to participate in an auction process to earn the right to deploy their blockchain inside the ecosystem. This makes it unlikely for small companies and blockchain projects to secure a slot in the network due to the requirement of competing in an auction against well-funded companies and projects. Cosmos in comparison, offers unlimited slots, allowing anyone to create a zone in the network without any barriers to entry.

When it comes to block production, Cosmos zones are responsible for selecting their own set of validators and achieving block finality through Tendermint consensus. Hubs are not responsible for validating transactions issued in the zones connected to it [66]. Parachains, on the other hand, are responsible for defining their own block production strategy, which is executed by participants called collators. However, they are unable to validate blocks on their own. Blocks proposed by collators must be approved by a global set of relay-chain validators that are responsible for guaranteeing the security of the entire ecosystem. Relay-chain validators are assigned to parachains in a rotating fashion and perform the validation of proposed blocks before adding them to the ecosystem's relay-chain. As both ecosystems rely on strategies that require a pre-defined set of validators to approve and

[2]https://wiki.polkadot.network/docs/en/learn-crosschain
[3]https://wiki.polkadot.network/docs/en/learn-auction

finalize blocks, they are incompatible with blockchains that employ Proof-of-Work (PoW) based consensus.

Validation is performed locally in Cosmos zones and globally in Polkadot's parachains, therefore their security assumptions differ. The Polkadot network is secured by a large amount of tokens staked by system wide relay-chain validators. In Cosmos, every zone is responsible for maintaining its own state and keeping it secure through their own individual staking process. This means that during cross-chain communication, Cosmos zones need to trust other zones they are interacting with. In Polkadot, parachain's states are shared across the entire ecosystem through the relay-chain, therefore, trust assumptions are on the relay-chain's validators rather than in the individual parachains. As a consequence of Cosmos' security model, zones with a small amount of staked tokens are more susceptible to attacks when compared to Polkadot's parachains, which are secured by large amounts of staked tokens. If an attacker is able to acquire the majority of the stake in a zone she can influence the outcome of governance proposals and selectively censor transactions.

3.4.2 Relay-based Systems

Relay-based systems (illustrated in Figure 3.2) enable information about state changes in one source blockchain to be relayed to and validated inside one destination blockchain using a smart contract. This information can be used for asset exchange and asset portability [25]. This process does not require the source chain to have any knowledge of the relay contract. The communication provided by relay systems is unidirectional, if a party needs to fetch information bidirectionally, two independent relay systems must be used.

- **Strengths of relay-based systems:**

Portability (Installability): Relays can be implemented in a destination chain without requiring the source chain to have any knowledge of it.

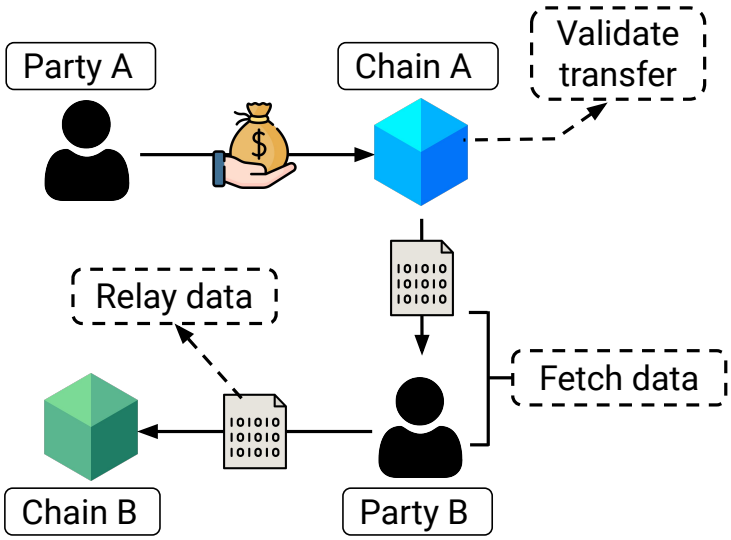


Figure 3.2: Overview of asset transfer using a relay-based system.

- **Weaknesses of relay-based systems:**

Performance efficiency (Resource utilization): As smart contracts are leveraged for transaction validation, the cost of verifying data using relay-based systems is proportional to the complexity of the verification algorithm.

Security (Integrity): Relay-based systems are susceptible to block reorganization attacks. Most relay solutions [69], [71], [81] only work with source chains that achieve probabilistic block finality and those chains may have their blocks reorganized after suffering an attack, invalidating a previously accepted transaction.

BTC Relay is a system that enables information to be relayed from the Bitcoin to the Ethereum blockchain and allows users to validate this information on Ethereum. BTC Relay receives and stores on the Ethereum blockchain the Bitcoin block information relayed by users. The system charges a fee to validate the received block information according to Bitcoin's consensus algorithm. BTC Relay allows Ethereum to verify and react to the state of Bitcoin's blockchain but not the other way around. No action needs to be taken the Bitcoin blockchain for block information to be relayed neither for it to be verified in Ethereum.

ETH Relay can be deployed in a destination chain to validate block information coming from a source chain in a cost-effective way. This system allows any pair of Ethereum-based blockchains to establish cross-chain communication but

only works unidirectionally.

ETH Relay uses an optimistic way of accepting block headers, accepting them without validation at first, but keeping them on a trial period in which users can dispute its validity. ETH Relay also enables efficient transaction verification by storing information about block branches and paths along with block headers, allowing the system to traverse blockchain data in an efficient way.

Proof-of-Burn is a mechanism that allows tokens to be transferred across blockchains by destroying them in a source chain, verifying the operation in a destination chain and then generating a representation of the destroyed assets.

To transfer asset using Proof-of-Burn, digital assets must first be "burned" in the source chain. In order to do that, users need to send the assets to a provably non-spendable wallet address causing them to be lost forever. To verify that the burning transaction has been committed in the source chain, a proof of inclusion has to be relayed to the destination chain. Such a proof can be constructed by using Non-Interactive Proofs of Proof-of-Work (NIPoPoW) [82], state digests [75] or AVL hash trees [83]. This proof needs to be verified in a smart contract prior to creating new assets. The burning procedure is unidirectional, can only be conducted once and is irreversible once the assets have been transferred to a burn (non-spendable) address.

Analysis of relay-based systems. The relay-based systems category includes BTC Relay, ETH Relay and Proof-of-Burn. Both BTC Relay and ETH Relay serve the same purpose, which is to allow a destination chain to verify state changes in a source chain. In terms of adaptability, however, BTC Relay is limited to verifying Bitcoin state changes inside the Ethereum blockchain, whereas ETH Relay can be applied to any pair of Ethereum-based blockchains. In addition, ETH Relay is more cost-effective than BTC Relay as it employs a block validation strategy that is optimized to consume less computational resources.

Proof-of-Burn on the other hand, was not idealized as a cross-chain communication system, but as a way to burn currency in exchange for participating in the block validation process. Burned assets are lost forever, potentially causing problems such as inflation due to reduction in asset availability if many burning

operations are executed. However, while not ideal for simple cross-chain asset transfers, Proof-of-Burn fits niche use-cases such as being used for destroying one type of token in order to transfer value to a different digital asset in a different blockchain [65]. This system should be used cautiously as destroying assets without having a guaranteed way of claiming them in another blockchain may lead to losing them forever. Additionally, if the burn operation is not set up in a secure and verifiable manner, burned assets may be claimed multiple times inside distinct blockchains as the claim remains unknown to other ledgers.

3.4.3 Sidechain-based Systems

A sidechain is a blockchain that is connected to another ledger referred to as the main chain through a mechanism that enables assets to be transferred back and forth between both blockchains. The mechanism which allows a sidechain and main chain to communicate bidirectionally is called a two-way peg, thus connected blockchains are also referred to as pegged chains. Sidechain systems must be adapted to a specific pair of blockchains. Figure 3.3 presents a generalization of the architecture and communication between pegged chains.

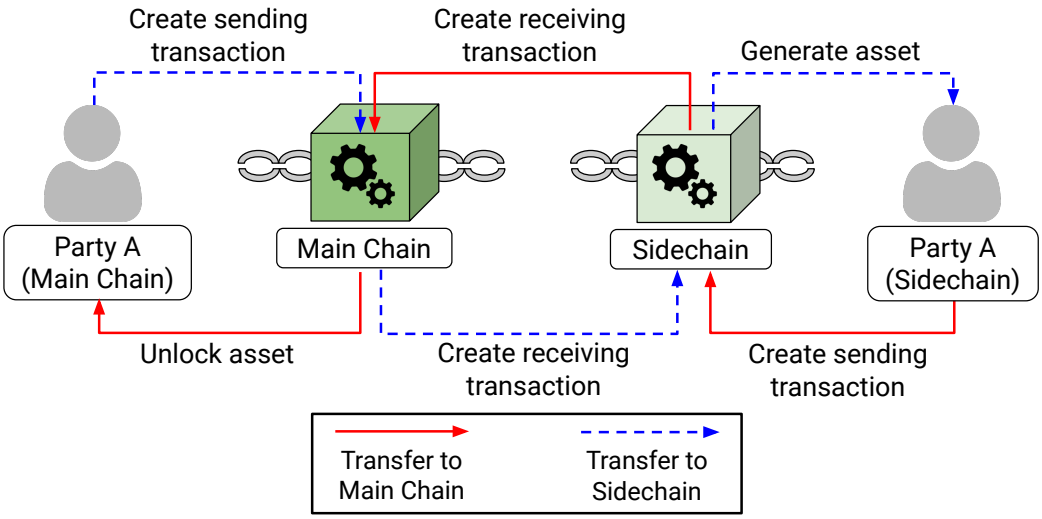


Figure 3.3: Transfer of assets between sidechains.

The relationship between a sidechain and a main chain can be either symmetric or asymmetric. When symmetric, both chains are able to operate independently and failure in one of them does not directly affect the other. Nodes in the main

chain do not have to be aware of the sidechain and vice-versa, unless they want to actively take part in cross-chain operations [64]. The asymmetric scheme consists of an independent main blockchain and a sidechain that relies on the main chain to operate. All the sidechain nodes are aware of the main chain, however, only main chain nodes who chose to support the sidechain are aware of it. When chains have an asymmetric relationship, failures on the main chain will have an impact on the sidechain, the opposite, however, is not true.

- **Strengths of sidechain-based systems:**

Maintainability (Modifiability): A benefit of sidechains is the possibility to extend blockchain functionality. Connecting a sidechain to a main chain that is deprived of certain features, e.g, transaction privacy, can enable users of said main chain to have access to additional desirable features without requiring changes in the main chain.

- **Weaknesses of sidechain-based systems:**

Portability (Installability): Sidechains systems must be developed for a specific pair of blockchains and therefore need to be designed to be compatible with both chains before deployment. Sidechains can also increase the complexity of a system in different levels [20].

Decentralization: Sidechains are susceptible to centralization of mining when one of the connected chains provides higher mining incentives than the other. This leads miners to work only for the most profitable chain and may slow down block validation in the other chain.

Liquid Network is a federated sidechain that allows assets to be moved from the Bitcoin blockchain and back to it. When assets are moved to the sidechain they fall under custody of a group of participants called functionaries [84]. Bitcoins (BTC) transferred to Liquid are represented as Liquid Bitcoins (L-BTC) inside the sidechain. For every existing amount of L-BTC there has to be a correspondent amount of BTC locked in the Bitcoin blockchain.

Proof-of-Stake Sidechains are sidechain constructions that enable cross-chain communication between Proof-of-Stake (PoS) blockchains. To transfer assets users must diffuse a message signaling their intention to transfer assets to the other chain in the source chain and a recovery message to reclaim its assets on the destination chain. The same operation can then be conducted in the sidechain to transfer the assets back to their source chain. Cross-chain proofs require a signature from a set of validators chosen through a leader election algorithm.

Proof-of-Work Sidechains are sidechain constructions that enable two independent Proof-of-Work based chains to establish cross-chain communication. To prove that an event took place in another chain, the proposed construction uses a cryptographic proof of state called Non-Interactive Proofs of Proof-of-Work [85]. The use of Proof-of-Work makes this system trustless, as users don't have to trust other peers, but rather validate transactions through a decentralized consensus algorithm. This system has the ability to transfer arbitrary data between chains in addition to transferring digital tokens.

Analysis of sidechain-based systems. The sidechain-based systems category main contains three cross-chain communication systems, namely Liquid Network, Proof-of-Stake sidechains and Proof-of-Work sidechains. Out of all three, the Liquid Network is the most limited in terms of adaptability, being restricted to cross-chain operations between the Bitcoin and the Liquid Network blockchains. Proof-of-Stake and Proof-of-Work sidechains are generic constructions and therefore can be incorporated to any Proof-of-stake and Proof-of-Work based blockchains respectively.

In regards to ease of use, the Liquid Network has an advantage over the other systems in this category. This is due to the fact that Liquid is a sidechain solution that has already been deployed, eliminating the need for development and deployment work. All one needs to start using Liquid is to transfer BTC to a Liquid Network peg-in address to receive an equivalent amount of L-BTC in the sidechain. The Liquid sidechain also provides features not available within the Bitcoin blockchain and makes them accessible via the use of BTC. Those features include fast transaction speed, enhanced transaction privacy and the ability for users to issue

their own tokens. However, Liquid's fast transaction speeds come at the cost of centralizing trust in a group of functionaries due to its federated model. This may be a security concern, as it introduces potential points of failure. Proof-of-Stake and Proof-of-Work sidechains, on the other hand, are academic proposals that require considerable effort to be developed into fully functioning sidechains, given that their objective is to provide generic constructions to enable secure cross-chain communication. However, despite not being ready-to-use systems like the Liquid Network, Proof-of-Stake and Proof-of-Work sidechains provide rigorously defined constructions that can be used as a reference for the development of cross-chain communication mechanisms from scratch while enabling developers to design the remaining sidechain components in a way that suits their needs.

In terms of consensus, Proof-of-Stake and Proof-of-Work sidechains were designed to work with blockchains that employ those specific consensus mechanisms and therefore are better tailored for a permissionless setting. The Liquid Network operates under a federated model and relies on a set of pre-defined validators called functionaries to secure the network. When users exchange their BTC for L-BTC, their BTC fall under control of Liquid's functionaries, requiring users to trust that the federation will keep their locked assets secure. The use of federations is a trade-off of decentralization for faster transaction processing speed.

3.4.4 Peer-based systems

Peer-based systems allow users to exchange assets directly with other peers rather than relying on a third party service. Figure 3.4 presents a generalization of the process of transferring assets using peer-based systems.

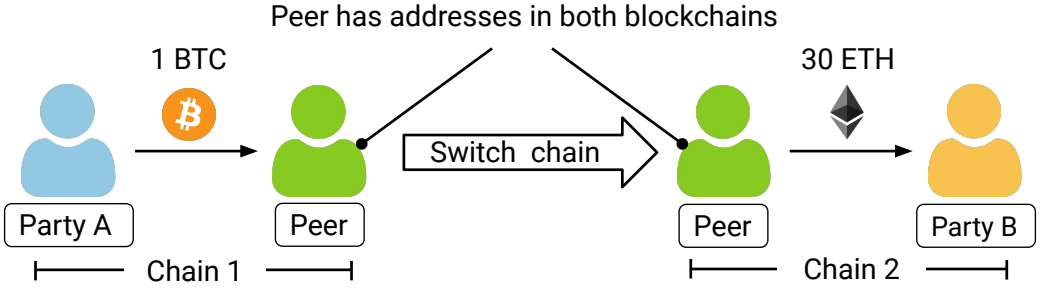


Figure 3.4: Process of transferring assets using peer-based systems.

User-to-user communication in peer-based systems takes place in communication channels outside of the blockchain environment. To enforce correctness and security during transfers, this category of systems relies on protocols that employ hashed timelock contracts and pre-commitments to coordinate operations across blockchains. Those systems allow two parties in any two different blockchains to engage in cross-chain asset transfers, as long as both blockchains support the scripting capabilities necessary to set up the coordination contracts by using hashed timelock contracts or agreements [86].

- **Strengths of peer-based systems:**

Portability (Installability): Transfers that leverage peers to forward assets across systems can be executed without the need to deploy complex systems. The logic required by exchange timelocked contracts is less complex than that required by other cross-chain communication approaches such as sidechains and relay contracts. Furthermore, implementations of Atomic Cross-chain Swaps are readily available for major blockchain platforms like Ethereum, Bitcoin and Litecoin [87], [88].

- **Weaknesses of peer-based systems:**

Performance efficiency (Time behaviour): Settlement of transactions using peer-based systems can have long delays that may last up to 48 hours [30]. Those are caused by the hashed timelock contracts (HTLCs) and the time required to allow parties to participate in or to abort deals.

Portability (Adaptability): Protocols that rely on timelocks to coordinate operations between blockchains lack the expressive power required to support complex cross-chain operations and deals [31]. Therefore, they are mostly limited to cross-chain asset transfers and payments.

Atomic Cross-Chain Swaps (ACCSs) allow users to participate in atomic asset exchanges between blockchains. A swap takes place between parties who hold assets in distinct blockchains, however, users must also have an account or address in the blockchain on which they wish to receive their payment. For example, if

Alice wants to exchange BTC for Ether and Bob wants to exchange Ether for BTC, both parties need one address in each participating blockchain to be able to execute a swap. To set up a deal using ACCSs, the participants must establish an out-of-bounds communication channel to agree on the details of the operation. The asset transfer protocol is coordinated using HTLCs, which carry out actions when the conditions agreed upon by the participants of the deal are met.

Interledger is a protocol that allows users in independent blockchains to exchange assets through a payment routing network composed of other users called connectors. Interledger is optimized for micropayments, making it similar to payment channels [89]. When a user wishes to transfer assets, the Interledger protocol is used to find a route of connected users that can form a path to forward the value packets hop by hop from the sender to the receiver, across different blockchains. It can be used by leveraging readily available implementations of different protocol components [90]–[93]. Users that provide services connecting other peers in the network can earn revenue by taking margins from the transfers they participate in. Whenever a user wishes to forward payments to a different ledger, it must use an Interledger Protocol Client [94].

Analysis of peer-based systems. The peer-based systems category includes Atomic Cross-chain Swaps and Interledger. Those systems require no development effort in order to be used for cross-chain communication. ACCSs are coordinated using HTLCs and while they require programming knowledge to set up, they are significantly less complex than full-fledged cross-chain communication systems. Interledger utilizes the Interledger Protocol to find routes of peers to connect the users that want to exchange payments. To use the protocol, users may leverage ledger plugins [95], [96] to communicate with connectors and find a cost-effective route to forward packets. Even though fairness issues have been identified in the Atomic Cross-chain Swap protocol [30], in general peer-based systems are secure and enforce correctness, meaning that users obtain what they expect.

Neither ACCSs or Interledger are tied to specific blockchains, however they require the ledgers involved in the cross-chain operation to support the creation of HTLCs. Interledger uses hashed timelock agreements (HTLAs), a generaliza-

tion of HTLCs that can be implemented in any type of blockchain. Additionaly, Interledger can also be used to exchange value with fiat money payment networks, a feature that is not supported by ACCSs.

Functional suitability however, is where the two systems differ. ACCSs only allow the execution of one cross-chain operation and need to be set up every time assets need to be transferred across blockchains. Additionally, operations using ACCSs take a long time (can last for 48 hours) and users need to stay active throughout the execution of the protocol in order to claim their assets. In contrast, Interledger is optimized for micropayments and allows users to open payments channels with each other and exchange value packets until one of the parties wishes to close the channel and update their balance in the blockchain. This characteristic makes Interledger more suitable for systems that need to perform a large volume of transactions over a long period of time. ACCs, on the other hand, are designed for deal one-time, definitive transactions which may be beneficial for one-off operations, but lack flexibility for ongoing interactions.

3.5 Related Work

In this chapter we have analyzed works that propose systems to achieve blockchain interoperability and have discussed them in detail. Therefore, we limit this related work discussion to related literature that approaches the problem of blockchain interoperability from a broader, more general perspective.

Buterin [25] presents a preliminary view of the blockchain interoperability field and discusses open challenges that must be addressed to advance the technology. The author classifies existing solutions into three distinct categories, analyzes their applications and provides a discussion on how to model cross-chain communication protocols in order to address failures. This work also provides a brief discussion on the expectations for the future of blockchain interoperability.

Belchior et al. [63] conducts an extensive literature review on blockchain interoperability. The authors systematically contact developers to obtain and present up-to-date information on grey literature. This work also proposes a framework to assess interoperability solutions based on evaluation criteria such as trust establishment and transaction settlement. By applying the proposed framework, the

authors classify studies on blockchain interoperability into three categories. Lastly, this work provides a discussion on the future of blockchain interoperability and presents open challenges within the field.

Koes and Poll [97] propose a set of properties to classify and evaluate blockchain interoperability solutions, including functionalities offered, regulation and third party dependency. Using the proposed properties, the authors provide a detailed comparison between the characteristics of Cosmos and Polkadot, two of the most popular proposals for blockchain interoperability.

3.6 Chapter Summary

As a result of the increasing interest in blockchains, the demand for enabling interoperability among them is rapidly increasing. This chapter provided a study on the architecture of blockchain interoperability systems with the aim of expanding existing knowledge and supporting practitioners and developers in the process of selecting strategies to connect blockchains based on existing systems. Additionally, it proposed an evaluation framework that can be applied to provide a streamlined way of assessing the capabilities of existing solutions. The proposed framework was utilized to provide an in-depth analysis of the features that characterize four distinct categories of blockchain interoperability systems and ten existing strategies divided across them. Each proposed category had their strengths and weaknesses outlined, offering valuable guidance for selecting suitable blockchain interoperability strategies according to specific use-cases.

Chapter 4

Analyzing the Performance of the Inter-blockchain Communication Protocol

Besides lacking the support for a common and widely adopted messaging format, most approaches to cross-chain communication are limited to the transfer of tokens. Blockchains can only fully benefit from cross-chain communication if they are capable of sending and receiving arbitrary data from other systems. One of the most popular proposals to address this challenge is the Inter-blockchain Communication Protocol (IBC), an end-to-end, blockchain-agnostic protocol that enables the transfer of custom payloads of data between blockchains. The support for arbitrary data allows not only for the transfer of different types of assets, but allows blockchains that implement IBC to even borrow security from other blockchains [98]. Despite its promising potential, the protocol is still under development and is subject to design and implementation issues. Therefore, it could significantly benefit from a comprehensive analysis of its current performance and limitations.

This chapter accomplishes *Research Objective 2* by presenting the first comprehensive evaluation of the performance of the Inter-blockchain Communication Protocol, a state-of-the-art proposal to transfer arbitrary data between blockchains. It presents a framework to guide the empirical evaluation of cross-chain communication protocols and applies it to implement a benchmarking tool for the IBC protocol. This tool is leveraged to analyze the protocol's throughput, latency and scalability, revealing several bottlenecks and design limitations that adversely impact cross-chain communication.

4.1 Introduction

While the interest in distributed ledger technology has increased, long-standing blockchain issues such as low throughput, high latency and trade-offs between security, scalability and decentralization [99]–[101] are still prevalent. For instance,

Bitcoin and Ethereum achieve a transaction throughput of roughly 7 and 15 transactions per second (tps) respectively, and are often compared to traditional payment systems such as Visa, which claims to be capable of processing 24,000 tps[1], to highlight the large gap between their performance.

Several works have evaluated and identified issues affecting the performance of both public and permissioned blockchains [40]–[54]. For instance, Hyperledger Fabric v0.6.0 is unable to scale beyond 16 nodes because of implementation issues in its consensus protocol [44]. Similarly, previous work shows that certain cryptography operations and REST API calls have led to transaction validation bottlenecks in Hyperledger Fabric v1.0 [52]. Beyond evaluating the system's performance, the authors propose optimizations that lead to a 16x increase in throughput. This highlights the importance of in-depth performance evaluations.

In modern blockchain applications, which increasingly require exchanging data between different blockchains (a.k.a. cross-chain communication) [102]–[105], bottlenecks may have a severe impact as they degrade the performance and increase waiting times for all blockchains participating in an operation. Cross-chain communication requires specific protocols to connect and reliably deliver data between blockchains, however, those protocols introduce additional points of failure and new bottlenecks, more so given that many are still under active development [63], [106].

Various protocols have been proposed to enable cross-chain communication [38], [64], [107], [69], [65]–[67], [24], [70], [71], [62], [20], [75], each with different trade-offs between security, performance and deployment complexity [21]. Relay protocols such as BTC Relay [71] and ETH Relay [69] employ smart contracts to verify transactions added to external blockchains, but work only one-way and are costly to maintain. Sidechains [20], [64], [107], [24] enable two blockchains to send assets back and forth via a two-way peg mechanism, but must be designed for a specific pair of blockchains. Blockchain ecosystems, such as Cosmos [66] and Polkadot [67], host independent blockchain applications and provide protocols to facilitate cross-chain communication, but impose constraints on blockchains to make them com-

[1] https://usa.visa.com/run-your-business/small-business-tools/retail.html

patible with the ecosystem.

Cross-chain operations differ from those executed in isolated blockchains in the following ways: they must undergo consensus to be recorded in all participating blockchains; they require a reliable communication channel through which data can be sent from one blockchain to another; they might be rejected by a participating blockchain, therefore they require atomicity in order to maintain a consistent state across ledgers [30], [108], [109]. Hence, cross-chain performance analysis requires an approach that is different from those proposed by existing evaluation frameworks for isolated blockchains.

Among proposals to enable cross-chain communication, Cosmos (USD 3 billion market cap) and Polkadot (USD 4.8 billion market cap)[2] have attracted significant attention in the past few years. Cosmos is a network of blockchains and decentralized applications built using its own SDK. Blockchains deployed in the Cosmos network can leverage a hub-and-spoke topology and the Inter-Blockchain Communication (IBC) protocol to exchange information. Binance Chain, Osmosis, Secret Network and Ethermint[3] are examples of chains deployed in the Cosmos Ecosystem. Similarly, Polkadot is a network of interconnected blockchains, called parachains, developed with the Substrate framework. Parachains can communicate through the XCMP (Cross-Chain Message Passing) protocol, built on top of the XCM (Cross-consensus Message) format. The Moonbeam network is currently deployed in the Polkadot ecosystem and a Chainlink implementation is currently under development[4]. But, despite their popularity, there are no performance evaluations of cross-chain communication using IBC or XCMP. This may be partly explained by fact that those protocols are under active development, which leads to breaking changes, outdated documentation and scarcity of information regarding setup and configuration.

In this work, we present the first performance analysis of the Inter-Blockchain Communication protocol paired with the Hermes Relayer for the transport of data. Those technologies are used to enable cross-chain communication in the Cosmos

[2]Up to date as of September 8, 2023. Retrieved from https://coinmarketcap.com/
[3]https://v1.cosmos.network/ecosystem/apps
[4]https://parachains.info/

Network. We leave the analysis of XCMP out of the scope of this work. This choice is motivated by the fact that XCMP is not yet implemented and is currently being replaced by a temporary, feature incomplete protocol, called Horizontal Relay-routed Message Passing (HRMP), which will be replaced in the near future [110]. The design of HRMP makes it significantly more resource demanding than XCMP, therefore, we deem that work on the analysis and improvement of the protocol would be more fruitful on its feature complete, long-term version. We leave such analysis as a future research direction.

IBC is an open source protocol designed to facilitate communication between heterogeneous blockchains. It defines a standard for constructing cross-chain messages and requires only the implementation of a minimal set of functions in the communicating blockchains. This gives IBC the potential to be used not only between blockchains deployed in the Cosmos ecosystem, but any blockchains that meet the requirements. In contrast, working implementations of IBC are already being used to connect over 70 homogeneous blockchains including the Cosmos Hub, Osmosis and the Juno Network[5]. Additionally, Polymer Labs and Hyperledger YUI Labs are working toward supporting IBC on existing blockchain platforms such as Ethereum, Hyperledger Fabric, Hyperledger Besu, Corda, Solana, Polygon and Fantom [111], [112].

Evaluating the performance of cross-chain communication protocols such as IBC poses additional challenges compared to isolated blockchains:

(i) The protocol is composed of multiple steps (*transfer, receive, acknowledge and timeout*) that must be recorded in the two communicating blockchains.

(ii) Blockchains are unable to communicate directly and require additional agents and/or protocols to deliver messages from one blockchain to another.

(iii) Operations may fail after having steps recorded in the blockchain. For instance, when a message is not delivered before timing out, intermediary steps recorded in transactions remain in the blockchain even though the cross-chain operation was not completed.

[5]https://hub.mintscan.io/chains/ibc-network

We emphasize that (i) increases the complexity of the experimental setup as multiple systems need to be deployed and connected to each other. Similarly, (ii) and (iii) increase analysis complexity by requiring the state of operations to be tracked across all of the systems involved in cross-chain communication.

In this work we make four major contributions:

- ***Contribution 1*** *(Section 4.3)*: We propose a novel framework for evaluating the performance of cross-chain communication protocols. As a first instantiation of our framework, we implement and make available an open source tool[6] to measure the performance of cross-chain communication between Cosmos blockchains using the IBC protocol. Our tool reduces the effort required to deploy and connect two Cosmos blockchains using the Hermes Relayer and provides seven configurable parameters that can be used to evaluate different blockchain configurations. Additionally, our tool generates execution reports to assist in performance evaluations for different setup configurations.

- ***Contribution 2*** *(Section 4.4)*[7]: We analyze the performance of cross-chain communications between Cosmos Gaia blockchains using the Inter-Blockchain Communication protocol and the Hermes Relayer[8]. Throughout our comprehensive throughput, latency and relayer scalability analysis we identify several surprising results and issues impairing performance. First, we show that using two relayers to relay for a single cross-chain channel decreases throughput by 33% compared to using a single relayer. Second, we show that the relayer application processes cross-chain transfers in batches, leading to high transfer completion latency, e.g., 455 seconds for 5,000 transfers. Third, we show that the main cross-chain communication bottleneck, which lies in the blockchain's RPC implementation, causes 69% of the time required to process cross-chain transfers to be spent on querying data from the blockchains.

- ***Contribution 3*** *(Section 4.4)*: We provide a comprehensive dataset that contais 158GB of execution logs collected throughout the execution of our experi-

[6] https://github.com/joaotav/ibc-benchmarking
[7] We have discussed our findings with the Hermes Relayer team from Informal Systems.
[8] https://github.com/informalsystems/hermes

ments [113]. This dataset serves two purposes. First, it provides assistance in the process of addressing the issues identified in our work. Second, it provides a useful resource for future studies on the performance of the IBC Protocol and cross-chain relayers.

- ***Contribution 4*** *(Section 4.5)*: We present five challenges faced during the deployment of the Cosmos Gaia blockchains and the Hermes Relayer. We discuss how those challenges can impact cross-chain performance and increase the difficulty of deploying and using the Inter-blockchain Communication Protocol.

4.2 The Cosmos Ecosystem

Cosmos is a network (also referred to as an ecosystem) of independent application-specific blockchains, called zones, interconnected through blockchains named hubs. Cosmos zones are powered by Tendermint BFT, a deterministic, partially synchronous consensus engine [114]. Hubs represent points of connection between zones, allowing the transfer of information. Different zones can communicate via the Inter-Blockchain Communication (IBC) protocol [115].

Cosmos provides a modular open-source framework, the Cosmos SDK, for the development of both permissioned and permissionless blockchains and applications. The Cosmos SDK enables developers to build on top of the Tendermint platform and aims to shorten development time by providing access to open-source composable modules that implement common blockchain functionalities such as token minting, token transfers, staking and punishment for misbehaviour.

4.2.1 Tendermint

Tendermint is a blockchain platform that provides a consensus engine called Tendermint Core and an interface called Application BlockChain Interface (ABCI). Tendermint Core consists of a Byzantine Fault Tolerant, deterministic and partially synchronous consensus engine. It also provides a peer-to-peer networking protocol which enables blockchain nodes to communicate. Tendermint's ABCI provides a generic interface that allows blockchain applications to interact with the consensus

engine provided by Tendermint Core.

Tendermint's consensus engine provides a protocol through which network participants called validators can cooperate to append information to the blockchain and update its state. The protocol can tolerate arbitrary behavior from up to one third of the network's validators while still guaranteeing consistency of state among honest nodes.

The protocol to achieve consensus is executed in rounds. In each round one participant from the validator set is elected as a proposer and gets to suggest a block of transactions for the current blockchain height. The remaining validators take part in two stages of voting, named pre-voting and pre-commiting, to decide whether to accept the proposed block or not. As soon as 2/3 of the validators cast their pre-vote for the same block, the protocol can receive pre-commits. If a block receives pre-commits from 2/3 of the validators before the end of the round, it is appended to the blockchain and the state of the ledger is updated based on its contents. If timeout is reached during a round and validators do not agree on a new block, the protocol advances to the next round and a new validator has the chance to propose a block.

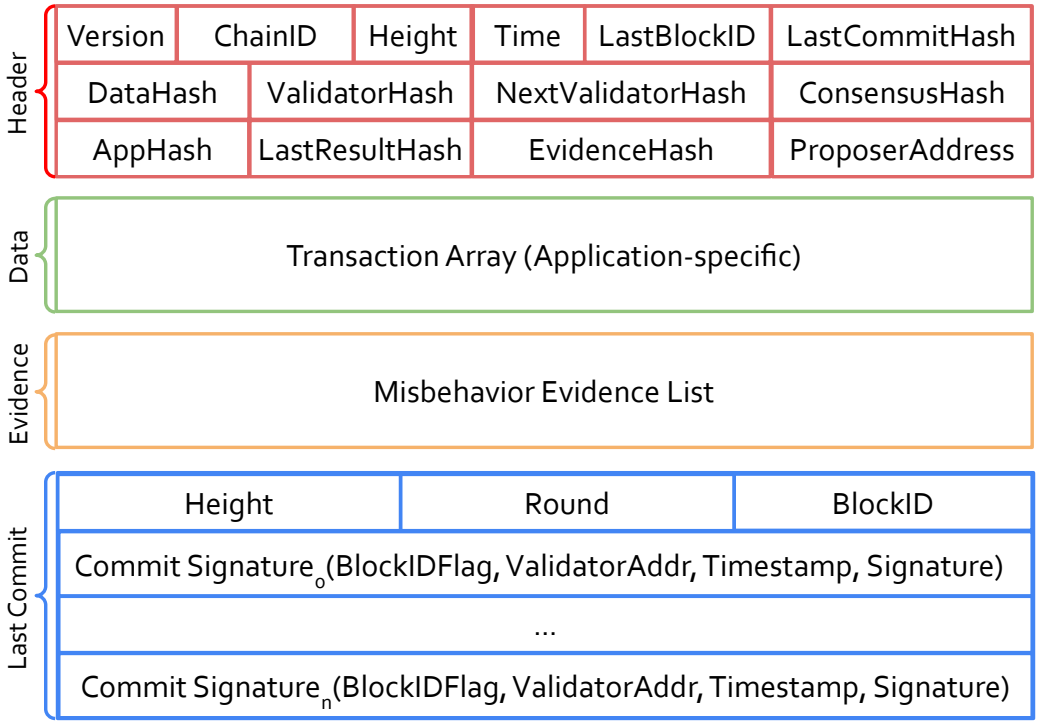


Figure 4.1: Structure of a Tendermint block.

We show the structure of a Tendermint block in Figure 4.1. A block is divided into four main fields, the *Header*, the *Data* field, the *Evidence* field and the *LastCommit* field. Except for the *Data* field, Tendermint performs validation on all the fields that compose a block. Transaction data is application-specific and unknown to Tendermint, hence it is a responsibility of the blockchain application to perform the validation of this data.

The *Header* contains information regarding the block and its position on the chain, as well as metadata related to consensus, validator set and the blockchain application. The *Data* field contains the set of transactions chosen by the block proposer and agreed upon by the consensus validators. Proofs of malicious activity carried out by members of the validator set are included in the *Evidence* field. Examples of malicious activity include the *lunatic validator* attack, in which more than 1/3 of a blockchain's validators vote for an invalid block and mislead clients into trusting this block, and the *equivocation* attack, in which validators vote for two different blocks within a single block generation round. When an attack is detected by a blockchain client, it collects the headers from the conflicting blocks and uses them as evidence to prove the occurrence of the attack. This piece of evidence can then be verified by full nodes and used by the blockchain application to punish the participants who took part in malicious activity. The *Evidence* field is empty in the absence of misbehavior.

Lastly, the *LastCommit* field contains the votes from the participants of the validator set. It contains data regarding block height, consensus round information, a *BlockID* of the corresponding block and an array of validator signatures. The signatures include the *BlockIDFlag*, indicating whether the validator voted for the block accepted by the majority of validators, *nil*, for a different block or did not cast a vote. Together with a vote, a validator's address, timestamp and commitment signature are included.

4.2.2 IBC Protocol

The Inter-Blockchain Communication protocol is an open-source, end-to-end, payload agnostic protocol that provides reliable and authenticated message pass-

ing between IBC modules residing in distinct blockchains [115]. Aside from a set of functions described in the interchain standard that describes host requirements (ICS 24)[9], no other prerequisites are imposed on the host blockchains.

Data transported across blockchain modules is opaque to the IBC protocol and only needs to be readable by the sending and receiving applications. This makes the protocol flexible and enables it to be applied for a variety of use cases such as fungible token transfers, deployment of multi-chain contracts, multi-chain account management and cross-chain data sharing.

The IBC protocol handles authentication, ordering and transport of data, but it does not have access to a transport layer to deliver data to other blockchains. Hence, in order to communicate using IBC, blockchains require access to external applications, called relayers, with access to network infrastructure and communication protocols such as TCP/IP.

Relayers are off-chain processes responsible for scanning blockchain data to identify messages that need to be delivered to another blockchain. When pending messages are found, the relayer retrieves them from the blockchain, transforms them into IBC datagrams and delivers them to their destination.

4.2.2.1 IBC Channel Setup

Blockchains can only communicate via IBC once a channel is established between them. Channels function as routes between communicating blockchain modules and, as per the IBC protocol's definitions, are responsible for providing ordering, exactly-once delivery and controlling the permissions that determine which modules are able to send and receive packets.

To open an IBC channel, a pair of communicating blockchains is required to first establish a *connection* through a handshaking process. To establish this connection, both blockchains must deploy a light client of the counterparty blockchain. Light clients are responsible for maintaining current information about the state of the counterparty blockchain. They achieve this by implementing an algorithm capable of receiving and verifying proofs that attest to the counterparty's latest

[9] https://github.com/cosmos/ibc/tree/master/spec/core/ics-024-host-requirements

consensus state. Light clients utilize relayers to periodically query their respective counterparty blockchain for state updates and may also receive these updates bundled together with new cross-chain operation requests. For conciseness, we implicitly assume that all cross-chain operation requests include state updates, which are verified by the respective light clients in the background. Once the light clients have been deployed, the blockchains can perform the handshaking process to open a *connection* and gain access to authorised communication.

After going through the handshaking process and establishing a *connection*, a pair of communicating blockchains can open a *channel* through which they can send and receive IBC packets. Channels can be either ordered, delivering packets in the order they are sent, or unordered, where packets are delivered in any order. Two blockchains can open multiple *channels* using a single *connection*.

4.2.2.2 The IBC Packet Life Cycle

IBC messages must be included in the blockchain before being sent through a cross-chain channel. The relayer application scans new blocks appended to the blockchain looking for transactions with IBC messages addressed to the channels it relays for. Upon finding pending messages, the relayer extracts their information from the blockchain, includes them in IBC packets with an up-to-date proof of the blockchain's state, and delivers them to the destination blockchain. A cross-chain transfer requires the inclusion of 3 messages in the communicating blockchains to be completed, namely, the transfer message (*MsgTransfer*), the receive message (*MsgRecvPacket*) and the acknowledgement message (*MsgAcknowledgement*). The transfer message requests a fungible token transfer and stores a proof of commitment to the packet data and packet timeout in the sending blockchain. Upon receiving a packet with a transfer request, the blockchain to which it is addressed verifies the packet commitment proof in the state of the sending chain and includes a transaction containing a receive message to accept it. Based on the receive message, an acknowledgement is sent back to the blockchain that initiated the transfer. Upon receiving the acknowledgement, the blockchain verifies that a proof of acknowledgement was stored in the receiving chain and includes a transaction with an ac-

knowledgement message. It is worth emphasizing that relayers must be connected to full nodes of both blockchains to execute queries for transaction data and relay packets.

We show the flow of a packet sent via the IBC protocol in Figure 4.2. Let blockchain$_A$ and blockchain$_B$ be a pair of blockchains. Blockchain$_A$ runs application$_A$ and implements a module that handles inter-blockchain communication, called IBC module$_A$. Similarly, blockchain$_B$ runs application$_B$ and implements IBC module$_B$.

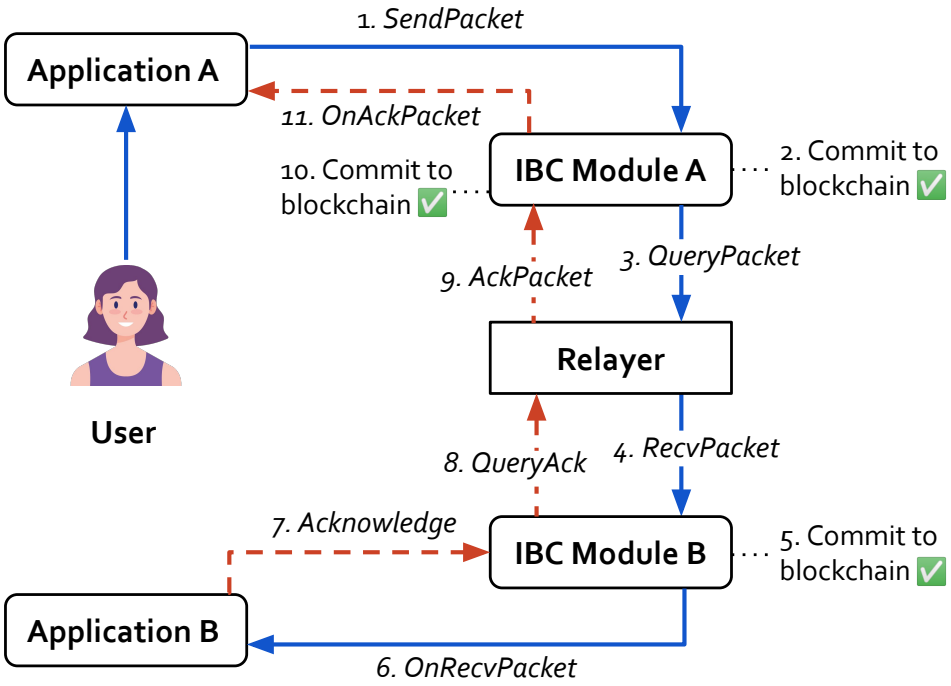


Figure 4.2: IBC protocol packet flow (sucessful).

Firstly, a user requests application$_A$ to send a packet to application$_B$. Application$_A$ then calls the *SendPacket* function implemented by IBC module$_A$ on the same blockchain. IBC module$_A$ stores a commitment to the outgoing packet and its timeout. This commitment is added to the state of the blockchain. Secondly, the relayer application queries the blockchain for pending messages and retrieves the information it needs to construct the corresponding IBC packet. Once the relayer has a packet ready for delivery, the receiving module on the destination blockchain, IBC module$_B$, triggers the *RecvPacket* function. Thirdly, the receiving module verifies the packet's timeout and the commitment in the source chain. If the data is successfully validated, the module routes the packet to application$_B$. Fourthly,

application$_B$ processes the operations contained in the packet's data and sends an acknowledgement back to IBC module$_B$. Fifthly, this module stores a commitment acknowledging that application$_B$ has received the packet from application$_A$. This commitment is stored in the state of the blockchain and is then queried by the relayer application. Next, the relayer builds an *AcknowledgePacket* and delivers it to IBC module$_A$ in the source blockchain. At this point, packet commitments stored by the module can be deleted since the packet has been successfully delivered.

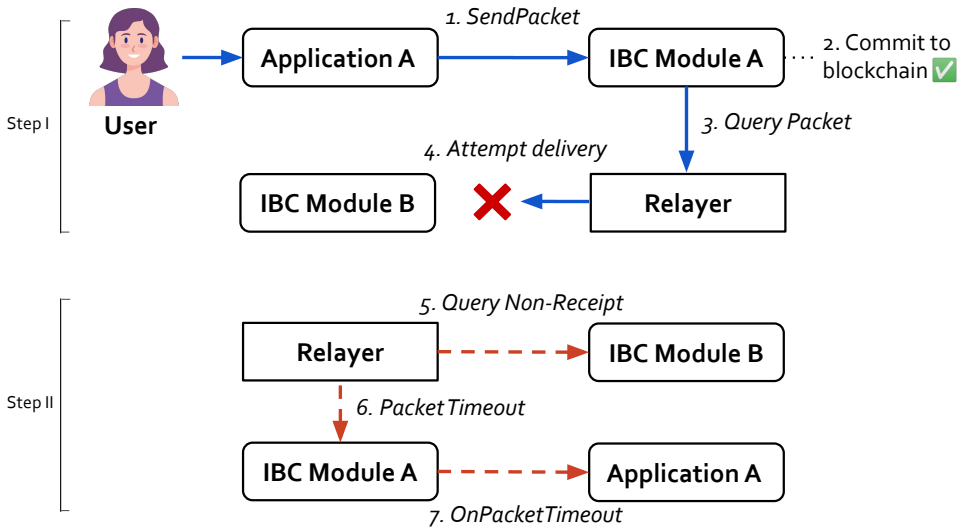


Figure 4.3: IBC Protocol packet flow (timeout).

Due to the nature of network asynchrony, packets may not be delivered before timing out. Timeouts can also be triggered when the receiving blockchain module does not accept the incoming packet. We illustrate the IBC packet with expired timeout in Figure 4.3. Application$_A$ calls the *SendPacket* function and IBC module$_A$ stores the packet commitment and timeout. However, it fails to be delivered to the destination module on blockchain$_B$ before it reaches the specified timeout. The relayer application identifies that the packet can no longer be accepted by the destination and sends a query to IBC module$_B$ asking for a proof that it has not been received. This information is sent back to IBC module$_A$ in the source blockchain and used to trigger the *OnPacketTimeout* function. This function implements the logic for undoing operations executed before packet commitment, such as unlocking assets that were previously held locked while the transfer request was pending.

4.2.3 Hermes Relayer

The Hermes Relayer is an open-source implementation of an IBC relayer written using the Rust programming language. It is one of the two IBC relayers that are currently under active development, the other being the Golang Relayer[10]. We chose to use the Hermes Relayer in our analysis of cross-chain communication for three reasons. First, it offers comprehensive documentation. Second, it provides extensive event logging capabilities, a feature that is essential for performance analysis. Lastly, at the time of writing, this relayer offers more features and is updated more often than the Golang Relayer.

In Figure 4.4 we present an overview of the Hermes Relayer's architecture. The Supervisor subscribes to events generated by blockchain full nodes. Each event is forwarded to the Packet Command Worker associated with its cross-chain channel. The Packet Command Worker schedules relayer tasks and assigns them to Packet Workers. Upon completing the assigned tasks, Packet Workers forward the resulting data to the Chain Endpoint, which interfaces directly with the destination blockchain through a full node. We refer the reader to the developer's overview [116] for an in-depth discussion of the Hermes Relayer's architecture.

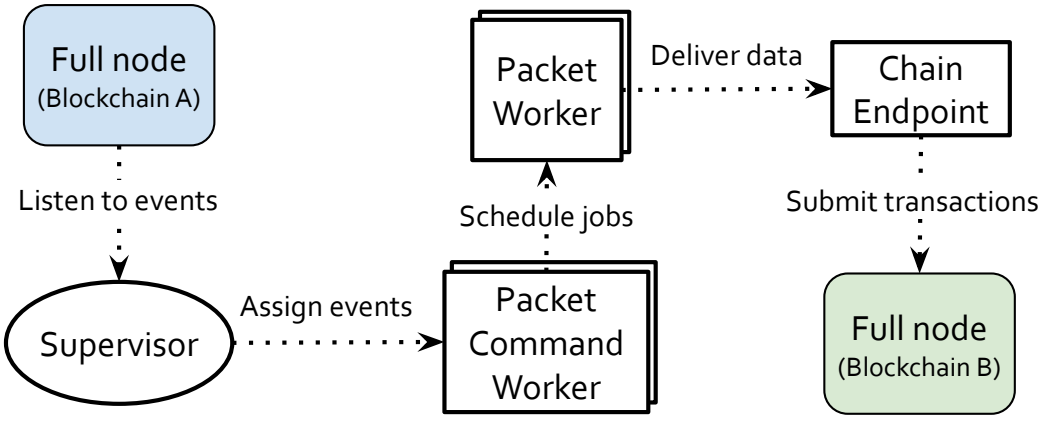


Figure 4.4: Architectural overview of the Hermes Relayer.

4.3 Experimental Methodology

We first introduce a novel framework for evaluating cross-chain communication protocols. We then provide a first instantiation of our framework as a tool for

[10]https://github.com/cosmos/ibc-go

evaluating the performance of the IBC protocol when used for cross-chain communication between two Cosmos Gaia blockchains. We opted for Gaia as the blockchain implementation for our experiments because it is readily available and officially maintained. In addition, it supports the deployment of private networks for testing purposes. Private test networks are essential for our experiments, as they provide the freedom to configure blockchain parameters and allow us to execute large amounts of transactions without the use of real-world assets. Gaia is built with the Cosmos SDK and leverages Tendermint Core for peer-to-peer networking and consensus. The same technology is used by over 70 blockchains currently deployed in the Cosmos Network. This allows our results to be generalized to interactions among other Tendermint-based blockchains in the Cosmos Network. Our tool provides assistance in experiment setup, execution, data collection and data analysis. For our experiments we set up a private testnet environment and use it to measure the performance of the IBC protocol under different scenarios.

4.3.1 Cross-chain Performance Evaluation Framework

We present our framework for evaluating the performance of cross-chain communication protocols in Figure 4.5. To the best of our knowledge, this is the first framework to guide empirical evaluation of cross-chain communication performance.

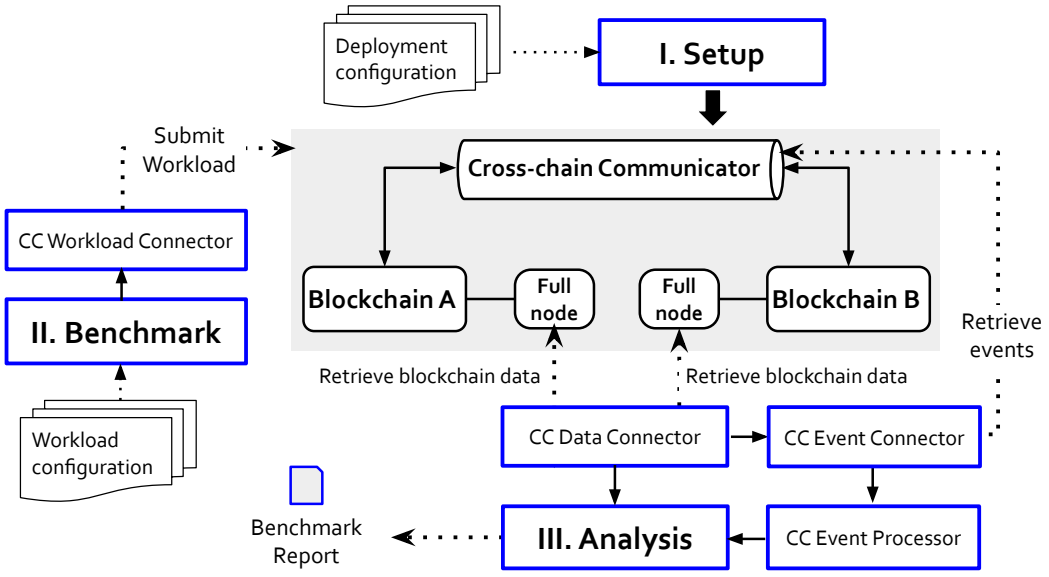


Figure 4.5: Cross-chain performance evaluation framework.

Our framework introduces four new key components, namely the *Cross-chain Communicator*, *Cross-chain Data Connector*, *Cross-chain Event Connector* and the *Cross-chain Event Processor*. The *Cross-chain Communicator* enables data transfer between blockchains; the *Cross-chain Data Connector* is required to retrieve data from different communicating blockchains concurrently; the *Cross-chain Event Connector* monitors and collects events associated with cross-chain communication, both from the communicating blockchains and from the *Cross-chain Communicator* component; and the *Cross-chain Event Processor* aggregates and interprets cross-chain communication events and their corresponding steps.

Our framework has three main modules, namely *Setup*, *Benchmark* and *Analysis*. The *Setup* module provides flexibility and automation for deploying blockchains with user-chosen parameters. This module is also responsible for configuring and deploying the *Cross-chain Communicator* (e.g, relayers for IBC). If evaluating cross-chain communications in an already deployed environment, the *Setup* can be skipped.

The *Benchmark* module allows users to evaluate cross-chain communication by executing workloads of different cross-chain operations (e.g, fungible and non-fungible token transfers and cross-chain data queries). It provides the *Cross-chain Workload Connector* component, which uses different resources (e.g., APIs, CLIs) to submit cross-chain operations directly to a blockchain or to an external application (e.g, the Hermes Relayer) that submits it to a blockchain.

The *Analysis* module processes blockchain and cross-chain communication data to generate performance metrics. This module provides the *Cross-chain Data Connector*, *Cross-chain Event Connector* and *Cross-chain Event Processor*. While in cross-chain communication protocols such as IBC a relayer fulfils the role of the *Cross-chain Communicator*, in Atomic Cross-chain Swaps this role is played by the users participating in the protocol. In the second case the *Cross-chain Event Connector* will retrieve events from the blockchain (e.g, transactions to claim assets from a contract) rather than from the relayer, using the *Cross-chain Data Connector* component.

4.3.2 Performance Analysis Tool

As a first instantiation of our framework, we implement a performance analysis tool to evaluate the IBC protocol. Our tool assists with the configuration and automated deployment of the test environment, workload execution and generation of performance metrics.

We implement our *Setup* module to deploy two Cosmos Gaia blockchains and connect them via a cross-chain channel established using the Hermes Relayer. The *Benchmark* module uses cross-chain fungible token transfers and instantiates a *Cross-chain Workload Connector* that binds the workload submission to the Hermes Relayer CLI. For the *Analysis* module, we implement an instance of the *Cross-chain Data Connector* which provides an interface to Tendermint RPC endpoints served by full nodes, allowing us to query data from the communicating blockchains. We further implement instances of the *Cross-chain Event Connector* and the *Cross-chain Event Processor* to collect and parse logs generated by the Hermes Relayer.

4.3.3 Deployment Configuration

We set up a private testnet environment composed of two Cosmos Gaia v7.0.3 blockchains, each maintained by five validator nodes. As we show later in Section 4.4, the number of consensus nodes has a negligible impact in our analysis as the performance bottlenecks lie in the relayer application and the blockchain's RPC server implementation.

We use Hermes Relayer 1.0.0 to connect the blockchains via an unordered cross-chain channel. The IBC protocol is implemented by application layer modules of the Gaia blockchains and by the Hermes Relayer. We employ five machines equipped with Intel i7-9700 3GHz processors, 16GB of 2666 MT/s RAM, and 7200RPM HDDs, running the Debian 11 "bullseye" operating system and connected to a local area network to run our experiments. We simulate wide area network conditions by enforcing a round-trip latency of 200 milliseconds between any pair of machines, similar to previous works [117]–[119]. Each machine hosts two full nodes, one validator of the source blockchain and one validator of the destination blockchain. The relayer application is executed in a machine together with

two validator nodes, but no more than one instance of the relayer application is executed in each machine. This setup allows the relayer to interact with the blockchain nodes via local endpoints and is recommended for use in production.

While our relayer setup is the same as that of a real application, the number of validators differs, i.e., 5 in our experiments and up to 128 in some blockchains. For a payload of 1024 bytes, consensus latency is approximately 25ms for 5 validators and 110ms for 128 validators [120]. However, we argue that this higher latency has an insignificant impact (approx. 1%) on the performance of cross-chain communications. For example, completing a single cross-chain transfer (requiring 3 blockchain transactions) takes 21 seconds on average in our experiment. When considering a real application with 128 validators, the added latency for each complete cross-chain transfer is approximately 255ms.

4.3.4 Experiment Settings

We configure the blockchains so that the time interval between the creation of two consecutive blocks is of at least 5 seconds. Blocks containing large amounts of transactions may increase the block interval beyond 5 seconds to allow time for the transactions to be processed.

We call *source* the blockchain that receives requests to initiate the transfer of tokens (*MsgTransfer*) to another blockchain (referred to as *destination* blockchain). Every transaction we submit to the source blockchain contains batches of 100 cross-chain transfer messages. This is the maximum number of messages per transaction allowed by the relayer application. We chose this number of messages for two reasons. First, this facilitates the submission of large amounts of transfers while reducing blockchain transaction processing overhead. Only one transaction has to be validated in order for all the messages within it to be processed. This allows us to include cross-chain transfer requests in the source blockchain at a faster rate, putting more stress on the relayer application. Second, it helps us in overcoming a limitation[11] caused by Cosmos blockchains, which restricts the number of transactions each user account (public key) can submit per block to 1. Cosmos' block-

[11] https://github.com/cosmos/cosmos-sdk/blob/274f389111c323c850c981c0de1b7b57eeb23912/x/auth/ante/sigverify.go#L219-L230

chains enforce transaction ordering through the use of account sequence numbers to prevent transaction replay attacks. In its current implementation, this mechanism causes users to have to wait for the confirmation of a submitted transaction before submitting another one with the next sequence number. This prevents us from using multiple transactions from the same user account to increase the number of cross-chain transfer requests within a single block. We use multiple user accounts to submit transactions containing 100 cross-chain transfer messages each to mitigate this problem.

When presenting our experiments, we refer to input rates in terms of requests per second and to throughput in terms of transfers per second. Those input rates correspond to the number of cross-chain transfers we submit per second to the source blockchain in the best case scenario, i.e., when one new block is produced each 5 seconds. If the interval is longer than that, the input rate will be affected. For instance, when the blockchain takes 10 seconds to produce a block, the number of transfers submitted per second is reduced by half. This happens due to the aforementioned limitation on transaction submission, causing the relayer to submit one transaction for each user and wait for its confirmation before submitting the next one. Therefore, a request rate of 1,000 transfers per second corresponds to a batch of 5,000 transfers being submitted every 5 seconds. To avoid ambiguity with the widespread abbreviation of transactions per second (TPS), for the remainder of this work we refer to TransFers Per Second as TFPS.

4.3.5 Evaluation Metrics

In this work we use the following performance metrics to evaluate cross-chain communications:

- **Throughput**: measures how many cross-chain transfers are completed per second. A transfer is considered completed only when all three required steps (*transfer, receive, acknowledge*) are correctly recorded in the blockchain.
- **Latency**: measures the amount of time required for an operation to be completed. As cross-chain operations usually require several seconds to be completed because of block intervals, we measure latency in seconds.

- **Relayer scalability:** measures the change in throughput and latency taking into account the number of concurrent relayers working for a cross-chain channel.

4.4 Performance Evaluation

In this section we assess the performance of the IBC protocol using the tool and metrics presented in Section 4.3. We analyze 158GB of data generated through 460 hours of experimentation. Based on our findings, we provide observations on how to improve the performance of cross-chain communication. In particular, we show that the main bottleneck lies in the blockchain's RPC server implementation. We also observe that when two relayers work concurrently for the same channel they are unable to relay in a coordinated manner due to their inability to communicate with each other. As a consequence, this reduces throughput and negatively impacts scalability in the presence of multiple relayers. Such discoveries are fundamental for the improvement of IBC and other cross-chain communication protocols that utilize relayers.

4.4.1 Throughput

We analyze the throughput of the different components used to achieve cross-chain communication, namely the Tendermint-based Cosmos Gaia blockchains and the Hermes Relayer. This allows us to identify performance bottlenecks and show how the peak throughput achieved by those components differ. Specifically, we show that including cross-chain transfer requests in the blockchain can be done at a rate over 10 times faster than the relayer application is able to complete them. This demonstrates that our experiments are stressing the relayer application rather than the blockchains.

Tendermint blockchain. We evaluate the throughput of the Tendermint blockchain by using the Hermes Relayer to submit cross-chain transfers (*MsgTransfer*) at input rates ranging from 250 to 14,000 requests per second (RPS) for 15 consecutive blocks. We set the minimum interval between the generation of two consecutive blocks to 5 seconds. In this setup we test only the blockchain's capacity to initiate

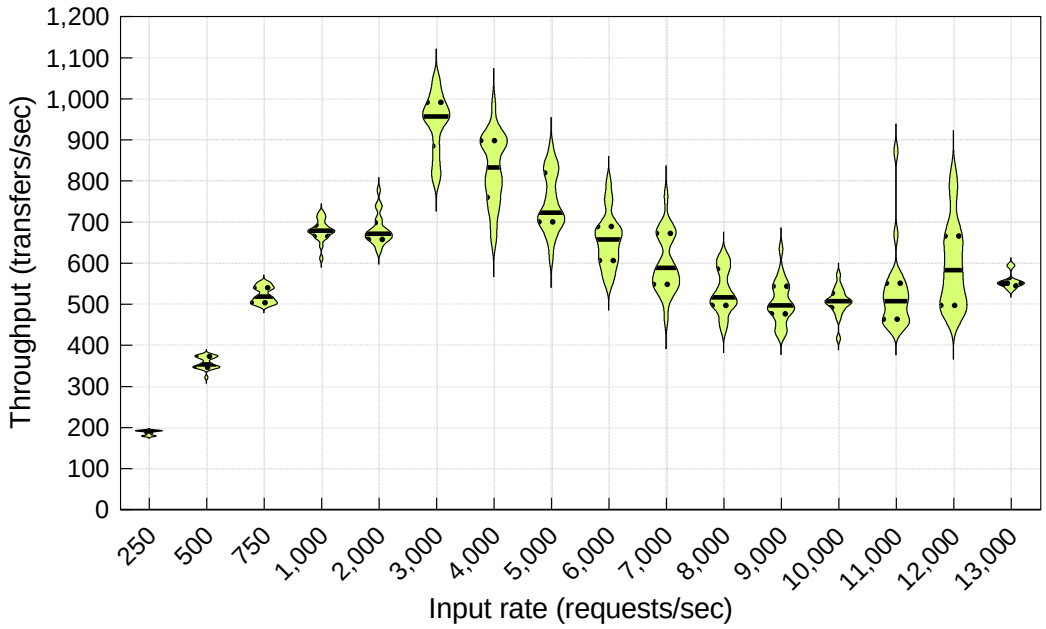


Figure 4.6: Throughput achieved by the Tendermint blockchain with 5 validator nodes, 5 seconds block intervals and a network latency of 200ms. Input rates range from 250 requests/sec to 13,000 requests/sec and each violin represents the distribution of the measurements obtained from 20 executions.

cross-chain operations by committing transfer messages. It is worth noting that at 14,000 RPS more than 90% of the requests to initiate a cross-chain transfer fail to be submitted to the blockchain, as we show in Table 4.1.

Figure 4.6 shows the throughput achieved by the blockchain with input rates ranging from 250 to 13,000 RPS. We omit results for 14,000 RPS as none of those experiments generated more than 5 consecutive blocks, leading to highly inconsistent results, e.g., throughputs of 10,373 TFPS, 605 TFPS and 0 TFPS. Violins in the plot represent the distribution of the measurements obtained from 20 executions using the same input rate. Inside each violin, the continuous line represents the median throughput and the dotted lines above and below it represent the upper and lower quartiles respectively. We begin by measuring the throughput achieved with 250 RPS, which yields roughly 200 TFPS. Throughput rises as we increase the input rate from 500 RPS (350 TFPS) to 3,000 RPS (961 TFPS), peaking at 961 transfer messages included in the blockchain per second. From 1,000 RPS to 2,000 RPS we observe a slight drop in throughput due to empty blocks, which appear in all experiments with input rates above 2,000 RPS. This is caused by an increase in the amount of processing required by the relayer as the number of requests to be sub-

mitted to the blockchain grows. Increased processing time causes the relayer to miss the transaction submission window, leading consensus validators to agree on an empty block.

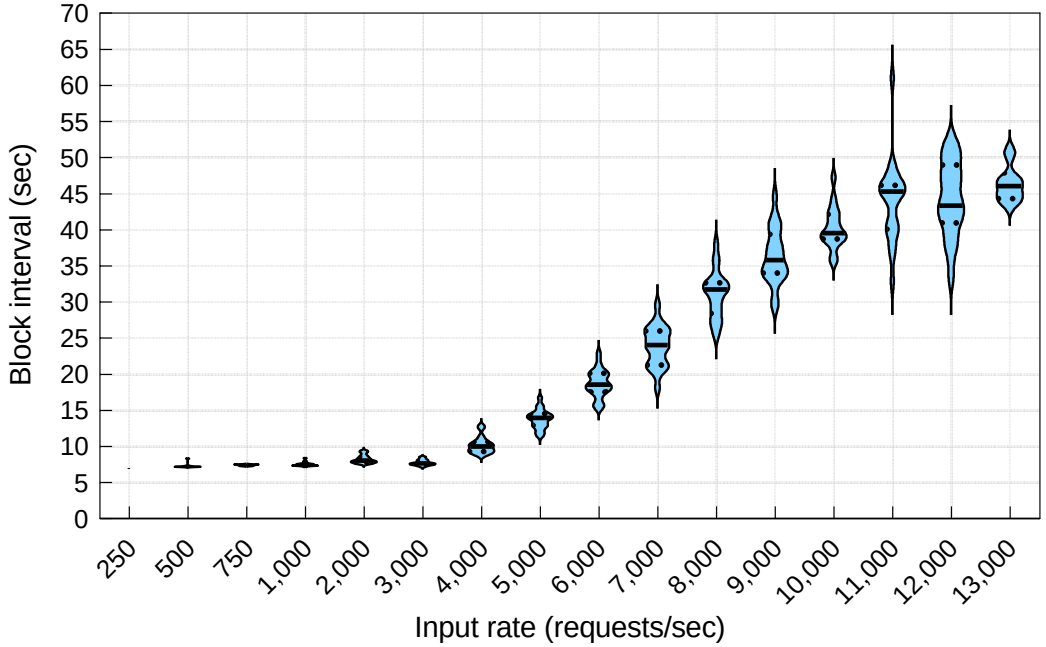


Figure 4.7: Average time interval between two consecutive blocks according to cross-chain transfer input rates ranging from 250 to 13,000 RPS.

For 3,000 RPS and beyond, the variance in measurements is more than twice that of smaller input rates. We also noticed an increase in block intervals as we continue increasing the rate of requests. This increase is shown in Figure 4.7. At this point, throughput becomes less consistent across each execution and decreases from 830 TFPS at 4,000 RPS to 499 TFPS at 9,000 RPS. From 10,000 RPS to 13,000 RPS measures become even less consistent due to the high amount of errors raised by the relayer application, namely *"account sequence mismatch"* and *"failed tx: no confirmation"*. The first accounts for 637,157 occurrences and 97.24% of the total errors, while the second represents 18,071 occurrences and 2.75% of the errors. The high incidence of *"account sequence mismatch"* errors is a consequence of the high volume of transactions being submitted. High transaction submission rates stresses the blockchain's RPC endpoint, which is also used to query for confirmed transactions. As a consequence, the relayer cannot reliably confirm submitted transactions and increment sequence numbers, leading transaction submission to fail.

We present the rate of failed transfers for each input rate and the amount of

requests made in Table 4.1. From 250 to 10,000 RPS the relayer is able to submit the majority of our requests (99%) to the blockchain. At 10,000 RPS, 80.17% of our cross-chain transfer requests reach the blockchain's transaction pool. From those, 98.3% are included in the chain. Higher request rates greatly decrease the number of transfers submitted and confirmed. At 14,000 RPS, only 8.5% of our requests make it through to the blockchain, and only 29.2% of those are committed by the validators.

Table 4.1: Execution summary for Tendermint throughput experiments.

Input rate (requests/sec)	*Requests made to Hermes*	*Submitted to blockchain*	*Committed (from submitted)*
250 to 9,000	18,750 to 675,000	>99%	>99%
10,000	750,000	601,300 (80.17%)	591,450 (98.3%)
11,000	825,000	319,152 (38.6%)	292,424 (91.6%)
12,000	900,000	160,343 (17.8%)	119,733 (74.6%)
13,000	975,000	100,688 (10.3%)	51,436 (51%)
14,000	1,050,000	90,000 (8.5%)	26,360 (29.2%)

Hermes Relayer. We measure cross-chain throughput by observing the number of cross-chain transfers that Hermes can send from the source to the destination blockchain (*transfer, receive and acknowledge*) within 50 consecutive blocks. We use input rates ranging from 20 RPS to 300 RPS and execute experiments using both one and two instances of the relayer to evaluate its ability to scale. In addition to executing experiments with 200ms network latency, we also performed measurements in a local area network with negligible latency (<0.5ms) to demonstrate how it impacts performance. Table 4.2 shows the size of the transactions used in our experiments.

Table 4.2: Summary of the sizes of transactions and messages used in the evaluation of the Hermes Relayer. Each transaction can only be composed by messages of the same type.

	MsgTransfer	***MsgRecvPacket***	***MsgAcknowledgement***
Transaction size (100 msgs)	18 kB	113 kB	127 kB
Single message size	183 bytes	1134 bytes	1274 bytes

The 100 messages used in our transactions consume an average of 3,669,161 gas

for transfers, 7,238,699 gas for receives and 3,107,462 gas for acknowledgements. Those requirements vary by at most 1%, 4.1% and 7.6% for transfer, receive and acknowledgement transactions across all of our experiments, respectively. In the Hermes Relayer's configuration, we set the gas price to 0.01 tokens per unit of gas. The average size of a transaction with 100 messages is 18 kB, 113 kB and 127 kB for *MsgTransfer*, *MsgRecvPacket* and *MsgAcknowledgement*, respectively.

Table 4.3: Evaluation parameters for experiments with 1 relayer.

Blocks	Block interval	No. transactions	Validators	Channels	Relayers	Latency
50	5 seconds	5,000 to 75,000	5 per blockchain	1	1	0.5ms and 200ms

Table 4.3 presents the parameters used to assess the cross-chain throughput with a single relayer. We evaluated performance at varying input rates by incrementally increasing the total number of transfer submissions in 5,000 step increments, while keeping the 50 blocks experiment duration constant. Our results are shown in Figure 4.8. Circles indicate measurements with 0.5ms and squares with 200ms. The error bands surrounding the data points represent the standard deviation of the measurements. Each point represents the average of measurements obtained from 20 executions using the same input rate. With 200ms latency, throughput is consistent across executions from 20 RPS (14 TFPS) to 120 RPS (72 TFPS). Relayer performance peaks at 140 RPS, being able to complete 90 cross-chain transfers per second, and drops with further increase in input rate.

Our findings show a consistency between the rate of requests and throughput for input rates 20, 40, 60, 80 and 100 TPS, with little variance between the executions within the same scenario. However, as we increase the input rate and approach the maximum throughput of the relayer application in our experiments, the difference between measurements increases. Peak throughput is observed with an input rate of 140 TPS, reaching 90 TFPS with 0.5ms and 80 TFPS with 200ms latency, a difference of 11%. Above 140 TPS, throughput gradually decreases as we increase input rate, dropping to 50 TFPS (200ms) and 56 TFPS (0.5ms) at 300 RPS.

We also repeated our experiments using two instances of Hermes instead of one. Our goal was to measure the system's throughput with multiple relayers working for a single cross-chain channel. Table 4.4 summarizes the parameters

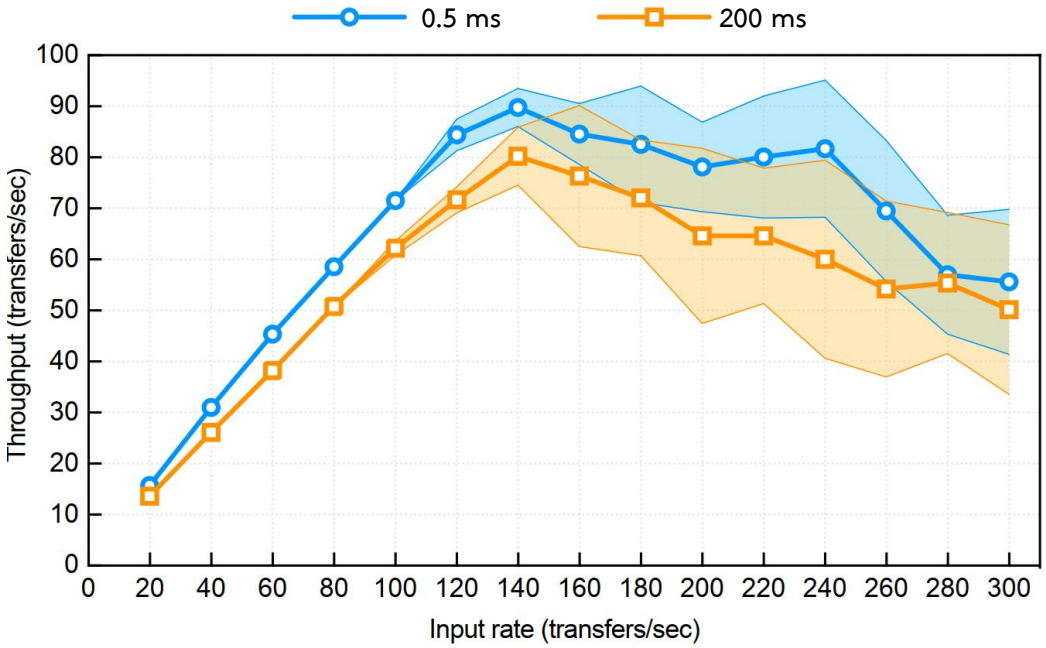


Figure 4.8: Cross-chain transfer throughput achieved by running one instance of the Hermes Relayer. Data points represent the average measurement obtained through 20 executions.

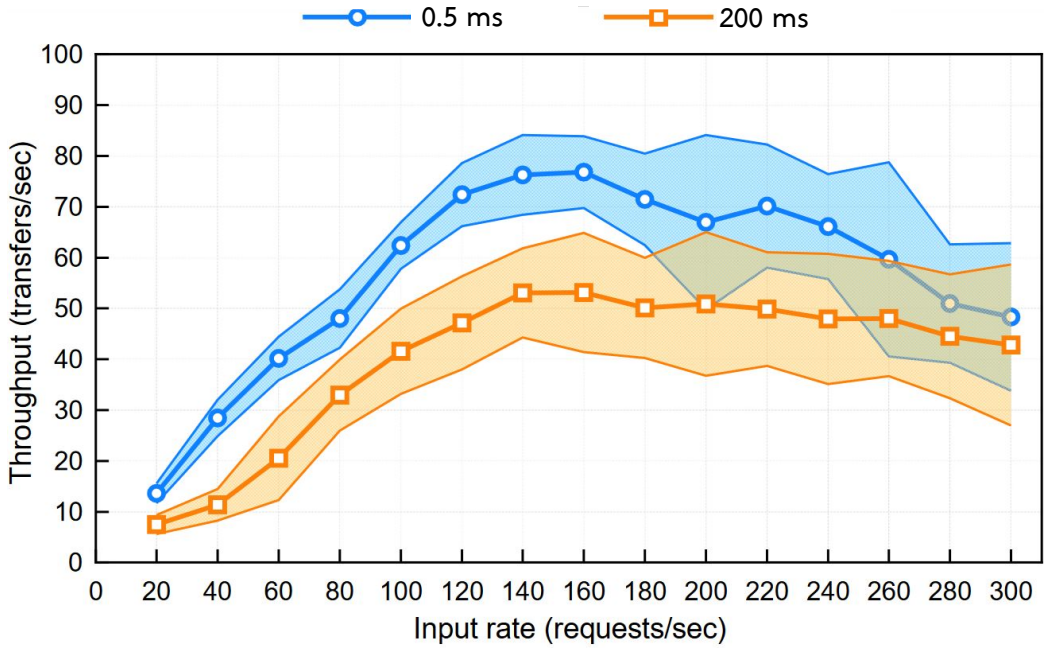


Figure 4.9: Cross-chain transfer throughput achieved by running two relayers for a single channel. Data points represent the average measurement obtained through 20 executions.

used to evaluate the cross-chain throughput achieved by 2 relayers in a single channel. Surprisingly, we got a maximum throughput of just 77 TFPS with 200ms and 53 TFPS with 0.5ms at an input rate of 160 RPS. Comparing these results with our setup using a single relayer, we got a peak performance 14% and 33% lower for

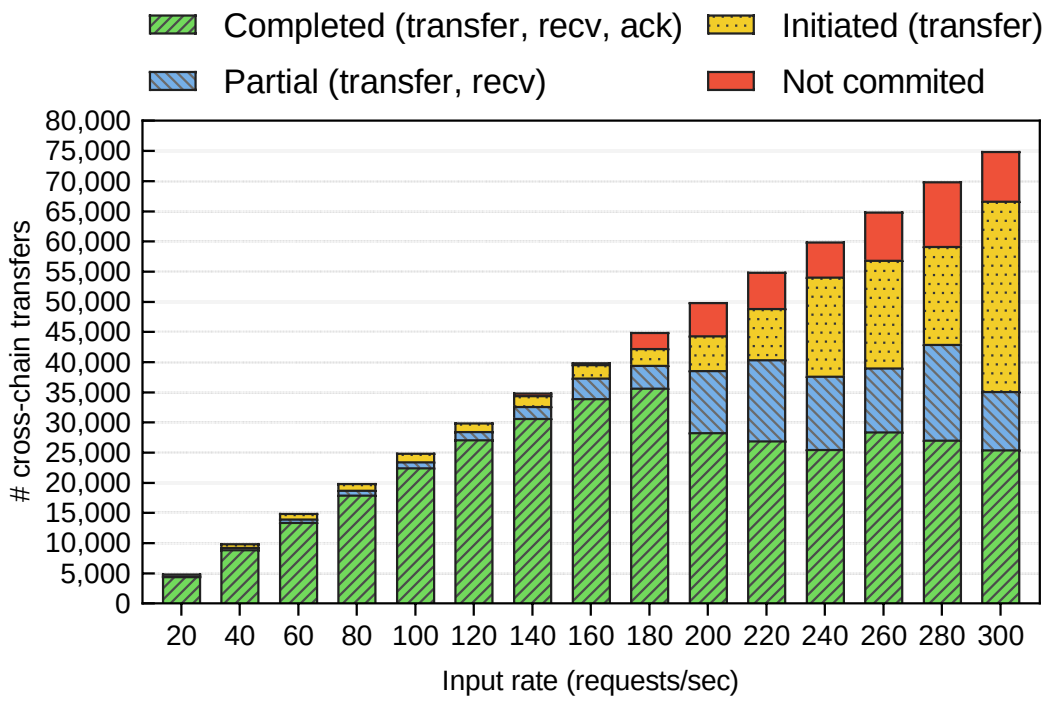


Figure 4.10: Average number of messages included in the source and destination blockchains within a 50 blocks interval using one relayer.

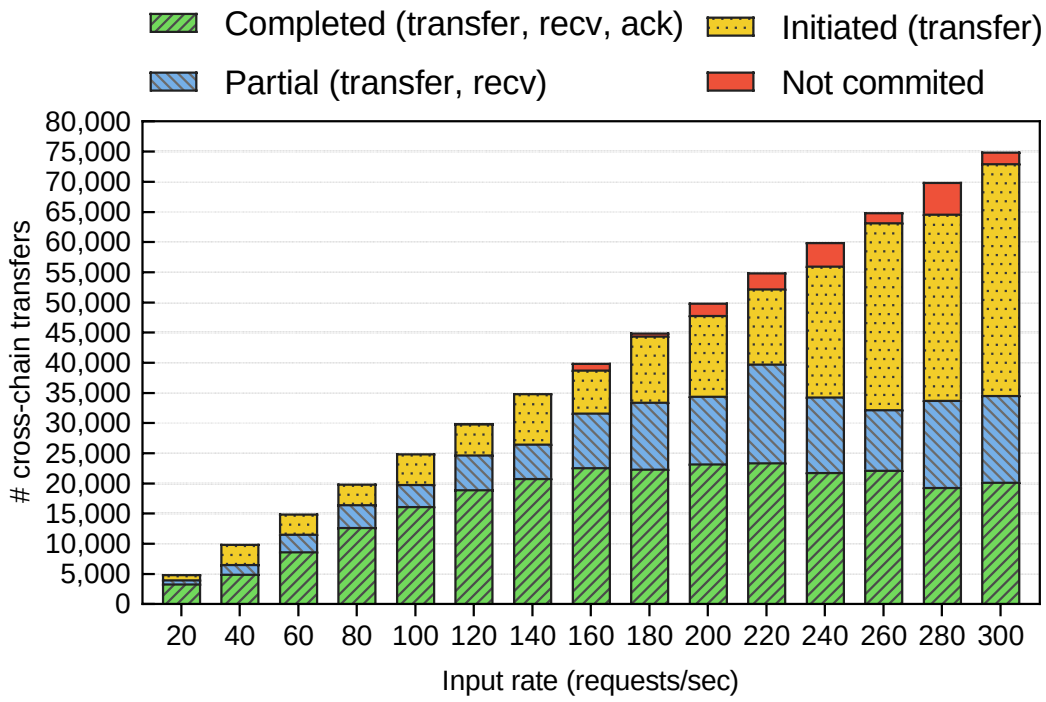


Figure 4.11: Average number of messages included in the source and destination blockchains within a 50 blocks interval using two relayers.

network latencies of 0.5ms and 200ms respectively. Upon investigating the logs generated by Hermes, we observed several *"packet messages are redundant"* errors, e.g., 23,020 times for 100 RPS. This happens when both relayers attempt to deliver

Table 4.4: Evaluation parameters for experiments with 2 relayers in a single channel.

Blocks	Block interval	No. transactions	Validators	Channels	Relayers	Latency
50	5 seconds	5,000 to 75,000	5 per blockchain	1	2	0.5ms and 200ms

the same messages to the blockchain, causing the first to succeed and the second to fail. This issue happens due to design choices of the relayer application, such as the inability of relayers to communicate and coordinate packet delivery. The Interchain Standard that defines Relayer Algorithms (ICS 18) [121] does not discuss scalability, a decision which can be detrimental to engineering and implementation. For instance, the standard neither considers nor discusses the use of multiple cross-chain channels for scalability. It also lacks a discussion on how to scale relayers within a single channel. We believe that basic scaling specifications would benefit the protocol if included in the standards. In short, this problem impairs throughput in setups with multiple relayers. Additionally, it also leads relayer operators to pay fees in an attempt to deliver redundant packets.

An alternative to increase cross-chain throughput would be to establish separate cross-chain channels for each relayer to relay on. However, while IBC allows the creation of multiple channels between a pair of blockchains, there are downsides to this approach. When tokens of the same type are sent through different channels, they are represented by different denominations (token type identifiers) in the destination blockchain. This causes the transferred tokens to be no longer interchangeable, making the use of multiple channels an unsuitable strategy for increasing performance. Additionally, while multiple uncoordinated relayers have a negative impact on throughput, they may maintain network liveness if a subset of them stops working.

Besides throughput, we have collected data regarding transfer completion status during experiments with 200ms latency. We measured the number of transfers that were completed (*transfer*, *receive*, *acknowledge*), partially completed (*transfer*,*receive*), only initiated (*transfer*) and those that were not committed to the source blockchain. We present the average message completion status for experiments with one relayer in Figure 4.10 and two relayers in Figure 4.11.

At the lowest input rate of 20 RPS we have submitted 5,000 transfers over a time frame of 50 blocks. At 300 RPS we have submitted 75,000 transfers over the same time frame. Up to 160 RPS, the majority of our transfer requests (>99.9%) were committed to the blockchain with both one and two relayers. With two relayers, however, a portion of the committed transfers are initiated, but fail to be delivered before the end of the experiment due to lower throughput caused by the previously discussed message redundancy errors. For 180 RPS and above, the percentage of completed transfers decreases with both one and two relayers. As we show in our latency analysis later in this section, messages contained within a block are processed sequentially, leading to longer confirmation times as we increase the number of transfers in a block. This leads to two observations in experiments with inputs rates of 180 RPS and above. First, transfers start getting completed, i.e, having the acknowledgement message included in the source chain, only several blocks after the experiments have been initiated. Second, transfers that are submitted several blocks after the start of the experiment were unable to be completed before the blockchain generated the 50th block and ended up either partially completed or only initiated. It is worth emphasizing that we measure the performance achieved by the relayer within a bounded time frame. If given sufficiently large timeouts and enough time after submission all valid transfers can be completed.

4.4.2 Latency

To measure the latency of cross-chain communication, we break down cross-chain transfers into 13 individual steps, as shown in Figure 4.12. A transfer operation starts with broadcast of the *transfer* message and ends with the confirmation of the *acknowledgement* message. Our goal here is to observe the time required by the Hermes Relayer to complete each step of a cross-chain transfer operation.

We submitted 5,000 cross-chain transfer requests within a single block and analyzed the logs generated by Hermes. This decision was based on a preliminary evaluation of the relayer's behavior under large transaction volumes, specifically processing 2,500, 5,000, 7,500, and 10,000 transfers within one block. Our analysis indicated that submissions exceeding 5,000 transfers either suffer a significant

delay before being added to the blockchain's transaction pool, causing them to be split across multiple blocks, or result in the relayer application emitting a *"Failed to collect events"* error because the amount of pending transactions is too large to be processed by Tendermint's WebSocket. We delve deeper into this issue in *Section* 4.5. Consequently, we selected 5,000 transfers for our experiment, as this number approaches the upper limit of what can be included in a single block and allows us to analyze how the relayer processes a large, single batch of transfers.

We show a detailed breakdown of the execution of the 5,000 transfers in Figure 4.12. We divide each of the three steps of a complete successful cross-chain transfer (*transfer, receive, acknowledge*) further into *build*, *broadcast*, *message extraction*, *message confirmation* and *data pull*. Steps are numbered according to the order in which they are executed by the relayer.

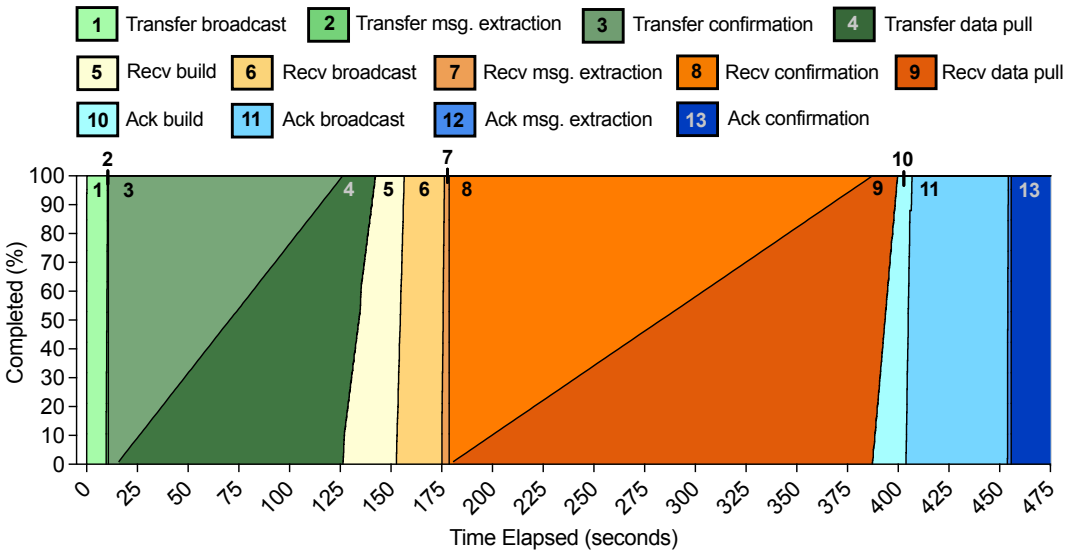


Figure 4.12: Breakdown of the operations executed to process of 5,000 cross-chain transfers submitted within 1 block.

A step starts being processed when the area marked by its number first appears in the figure. As operations get completed, the area moves towards the top. When a step succeeds another, the area of the step that is currently being processed is drawn on top of the previous one. For example, step 4 (*Transfer data pull*) starts being processed at 16 seconds, is 50% completed (2,500 out of 5,000) at 75 seconds and finishes when 126 seconds have elapsed. Steps such as *message extraction* and *message confirmation* are completed near instantly for all the requested transfers,

leading to a vertical line in the figure. Others, such as *build* require more processing and have their completion percentage increase gradually.

We start measuring the latency of operations from the moment *transfer* messages are broadcast. After a message is broadcast, the relayer extracts status information from the blockchain (*message extraction*), confirms that it has been committed (*confirmation*) and requests the data contained in it to the blockchain (*data pull*) to build the next message required by the protocol. Overall, steps required to process the *transfer* message consume 126 seconds or 27.6% of the total processing time, steps related to the *receive* message require 261 seconds (57.3%) and steps related to *acknowledgement* message are processed in 68 seconds (14.9%). The first transfer operation to be completed out of the 5,000 we submitted required 455 seconds from *transfer broadcast* to *acknowledgement confirmation*.

Blocks are handled sequentially by the relayer, leading it to process the first step for every *transfer* message within the block before moving on to the next. As a consequence, the relayer only processes the next step when the previous one has been completed for every transfer. This means that no transfers can be completed until the first 12 steps have been completed for the whole batch being processed.

We observe that the main bottleneck in the process of cross-chain communication is caused by the operations that retrieve data from the *transfer* and *receive* messages included in the blockchain, namely *transfer data pull* and *recv data pull*. The former corresponds to 24% (110 seconds) of the time spent processing the 5,000 cross-chain transfers. The latter corresponds to 45% (207 seconds). Together, those operations require 317 seconds to be completed, which is roughly 69% of the total processing time. For the duration of those operations, Hermes is using RPC to query blockchain nodes for data. The reason why the *transfer data pull* and *recv data pull* operations correspond to over two thirds of the the time required for cross-chain communication lies in Tendermint's RPC implementation. Tendermint is unable to process queries in parallel, requiring the relayer to wait while its requests for data are processed one by one. This leads to completion latencies in the order of several minutes.

We conducted experiments using different transaction submission strategies

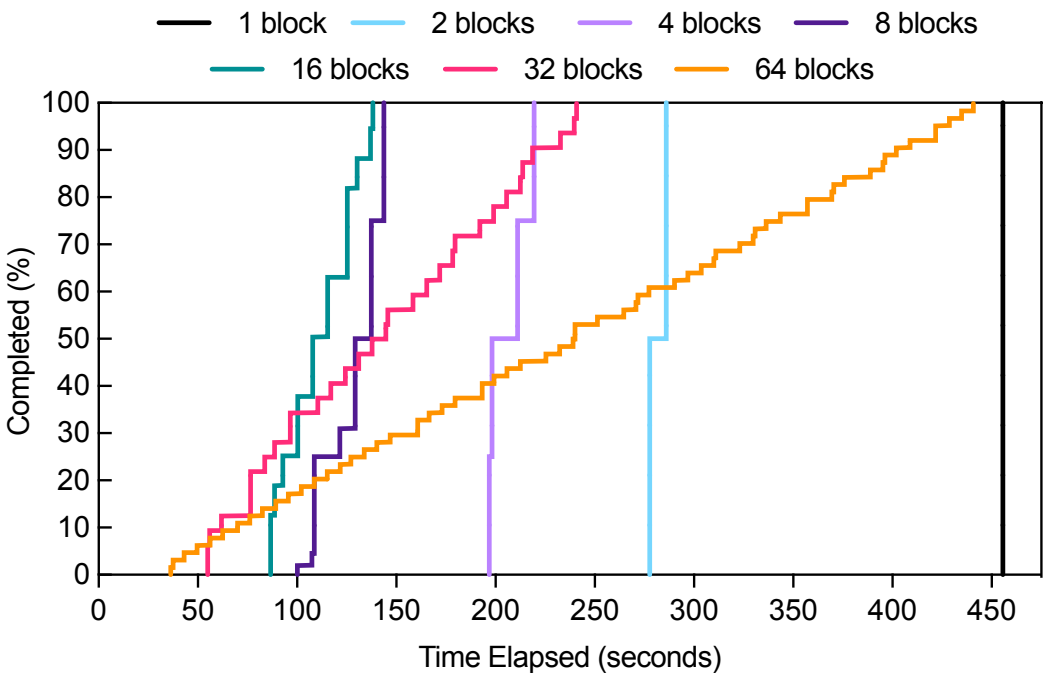


Figure 4.13: Completion of 5,000 cross-chain transfers over time based on seven distinct transaction submission strategies.

in an attempt to reduce the completion latency of cross-chain transfers. For this purpose, we evenly divided the total number of cross-chain transfer requests into different intervals of blocks, ranging from 1 to 64. Figure 4.13 shows how different submission strategies affect completion latency. Submitting 5,000 transfers within 1 block, shown in detail in Figure 4.12, leads to a 455 seconds completion latency for all 5,000 transfers. Dividing transfer submission into 2 blocks yields a 286 seconds completion latency, a 37% reduction. As we double the interval over which we submit transfers, completion latency is further reduced to 219 seconds (4 blocks), 143 seconds (8 blocks) and 138 seconds (16 blocks). We observe that spreading submissions further beyond this point increases completion latency, rather than reducing it further, to 240 seconds (32 blocks) and 441 seconds (64 blocks). This is a result of the accumulated period of waiting for each block interval time to elapse, which increases as we distribute the transfers over a larger number of blocks. In Figure 4.13, as we increase the submission interval from 1 to 64, the respective lines in the plot change from being straight, which indicates that all transfers had their final step completed at the same time, to having multiple steps over time. Each step represents the interval elapsed between the completion of a batch of transfers. For

example, the line for 2 blocks contains one step, meaning that half of the transactions were completed in one block and the other half was completed in another block, hence the step. The number of steps in each line is proportional to the size of the interval they represent.

We conclude that submitting cross-chain transfers in large batches, while easier to carry out, as one does not have to coordinate submissions over multiple blocks, severely increases completion latency. Increasing the submission window from 1 to 16 blocks provides a 70% reduction in completion latency. We note however, that spreading transfer submission over larger block intervals has an inverse effect, increasing completion latency from 138 seconds to 441 seconds, a 320% increase. Additionally, splitting the submission into multiple blocks causes transfers to start being completed sooner, in increments proportional to the number of transfer requests inside each block.

4.4.3 Summary of findings

We now present a summary of the research findings derived from the results obtained from our throughput and latency experiments. We elaborate on the importance of each finding as well as its impact on the current systems.

- ***Research finding 1:*** Tendermint blockchains can process and commit transactions more than 10 times faster than relayers can transfer them across blockchains. This means that during periods of high network activity, the blockchain is capable of initiating more cross-chain operations than the relayers can manage. Without a viable approach to increase relayer throughput, this imbalance has the potential to cause network congestion.

- ***Research finding 2:*** When a single cross-chain channel is served by two relayers instead of one, the channel's throughput decreases by 33%. This is a result of the relayer's inability to communicate with one another, leading them to spend resources on the delivery of redundant messages. Furthermore, this causes relayers to incur losses, as redundant messages must also pay fees to be processed before being rejected. This finding shows that the throughput

of a cross-chain channel cannot be scaled through an increase in the number of relayers.

- ***Research finding 3:*** Submitting large batches of cross-chain transfers at once results in a high confirmation latency, proportional to the number of transfers in that batch. This issue is a consequence of the relayer's strategy when processing transfers, which involves retrieving the whole batch of transactions included in a block processing them sequentially. To minimize confirmation latency, the submission of large batches of transfers should be avoided. Our findings indicated that distributing transaction submission into multiple blocks can significantly reduce confirmation latency. For instance, spreading the submission of 5,000 transfers over 16 different blocks reduces confirmation latency by 70% compared to a single, large batch submission.

- ***Research finding 4:*** Although IBC allows for the creation of multiple cross-chain channels between a pair of blockchains, different channels cannot be utilized to increase the resulting cross-chain throughput between those blockchains. This issue stems from the fact that IBC uses a concept called denominations to represent tokens sent across blockchains. A denomination is an unique identifier of a token, and is generated based on information from the source blockchain and the channel through which the tokens were sent. As a result of this design, the submission of transfers through multiple channels essentially transforms the received tokens into different types. This can potentially create multiple, non-fungible representations of the same token, making it difficult for the receiving blockchain to manage them.

- ***Research finding 5:*** The data queries performed by the relayer, required to retrieve information from the communicating blockchains, constitute a major bottleneck for cross-chain operations. To build IBC packets, relayers query the blockchain for relevant information through its RPC endpoint. Given the current design of Tendermint's RPC endpoints, those queries cannot be processed in parallel, causing them to wait in a queue until the previously received queries are processed. As a consequence, when relayers have to

process large amounts of cross-chain operations, the resulting wait times can be in the order of several minutes. This finding indicates a need for improvements in Tendermint's RPC servers, specifically to enable the support for parallel data querying, thus allowing the blockchain efficiently handle multiple queries during peak network activity.

4.5 Deployment Challenges

In this section we summarize technical challenges faced throughout this work with two goals in mind. First, to further give grounds for some of our design choices and elaborate on the impact they had on our benchmarks. Second, to bring attention to those issues and contribute to the improvement of the affected systems.

- ***Timestamp mismatch***: Analysis of experimental data revealed a difference between the timestamp of events registered by the Tendermint blockchain and those registered by the relayer application. We observed that events logged by the blockchain are a few seconds behind, e.g, the timestamp for a transaction broadcast recorded by the relayer is set after the same transaction has already been included in the blockchain according to the block timestamp. This mismatch in the timestamps recorded by both applications may lead to incorrect metrics if both sources of information are utilized together. For this reason we chose to rely on the timestamps recorded by Hermes for our cross-chain performance analysis.

- ***Account sequence mismatch***: When attempting to create transactions that contain cross-chain messages through the relayer command line interface (CLI), we were unable to submit more than one transaction per block for each user account (public key). Attempting to submit a new transaction from the same account while a previously submitted transaction has not yet been confirmed yields the *"Account sequence mismatch"* error. This can be circumvented by issuing multiple transfer messages in the same transaction, allowing a user to perform multiple payments without having to wait for each transaction to be confirmed. In practice, however, this workaround has the downside of requiring the payer to accumulate outgoing transfers in order to include them all in

the same transaction. Furthermore, users may be still unable to issue new transactions if a previous one remains awaiting confirmation. To overcome this limitation we chose to use multiple user accounts to submit transactions, allowing us to control the volume of messages submitted per block by changing the number of concurrent users issuing transactions containing transfer messages.

- ***Transaction data collection***: During the development of our tool we explored the options provided by the available data endpoints in order to choose the queries that best suited our needs. Retrieving the list of the transaction hashes associated with the transactions included in a block allows us to query the blockchain for those specific transactions and their cross-chain message contents. Given the large amount of transactions we generate (e.g., 2250) during benchmark execution, one of our main goals was to minimize the number of queries required for data collection to reduce execution time.

  Upon a careful examination of the queries supported by the Tendermint RPC and the Gaia blockchain we were unable to find an option that returned the list of transaction hashes in a specific block without returning a substantial amount of additional transaction data. Both the `tx_search` query parameter from Tendermint RPC and the `query txs --events tx.height=X` from the Gaiad CLI return a large amount of information. For instance, querying a local node for the content of a block that includes 20 transactions with 100 *MsgTransfer* each returned 331,706 lines of output and took an average time of 2.9 seconds to complete using our setup. Querying a block with 20 transactions containing 100 *MsgRecvPacket* each returned 579,919 lines of output and took an average time of 5.7 seconds. It is essential for us to collect the hashes of the transactions contained within a block to perform the our analysis, therefore we chose to use the aforementioned queries despite their negative impact on performance. As such queries may return a substantial amount of data, we had to deal with pagination as some blocks can be too large to fit in a single request.

- ***WebSocket space limit***: When blocks are minted, the relayer application queries the blockchain to retrieve their contents and identify transactions containing pending IBC messages. If the amount of data to retrieve exceeds the Tendermint Websocket maximum message size (16MB), the relayer emits the *"Failed to collect events"* error. This issue happens every time a block contains a large amount of IBC events, such as transfers, that need to be processed. We observed that when this error is raised, if the relayer packet clear interval is set to 0, pending IBC transfers neither get completed nor fail even when their timeout is exceeded. To analyze the impact this has on packet relaying, we ran an experiment and adjusted the parameters with the intent to trigger this error. We generated a block containing 1,000 cross-chain transactions with 100 IBC transfers each, causing the *"Failed to collect events"* error. From the transactions in this block, 2.5% were completed successfully, 15.7% timed out and 81.8% got stuck, meaning they were committed on the source chain but neither were relayed successfully or timed out, even after four times the number of blocks required for timeout were appended to the chain. Shortly after, we submitted multiple transactions containing a single IBC transfer. All single message transfers submitted after the error were committed to the source blockchain, but were not delivered by the relayer, causing them to time out. This indicates that the WebSocket error not only prevents transactions that failed to be collected from being completed, but also impacts future transactions.

- ***Incomplete logging for blockchain data retrieval***: By analyzing the execution logs generated by the Hermes Relayer we were able to collect data regarding the operations performed by the application and their timestamps. However, only a fraction of the operations that query the blockchain RPC endpoints for IBC message data are recorded in the logs. This is the operation we are most interested in, given that it is the main bottleneck in the process of cross-chain communication. We observed that the retrieval of data from the first block of transactions in our experiments is correctly logged, but not that of transactions included in subsequent blocks. Upon verification we confirmed

that transactions included in subsequent blocks are queried for their data, as their packets get built and delivered by the relayer, despite the data pull operation not being recorded in the logs. We suspect this is caused by an implementation issue.

4.6 Related Work

Most previous works on empirical blockchain performance analysis have essentially targeted the execution of workloads in isolated blockchains. Those workloads are usually designed to stress test specific blockchain functionalities such as consensus protocols, smart contract languages or a decentralized applications.

Table 4.5 presents an overview of related work on blockchain performance analysis including the platforms and criteria evaluated.

Table 4.5: Overview of related work on blockchain performance evaluation.

Related Work	*Platforms*	*Code available*	*Cross-chain analysis*	Throughput	Latency	Scalability	Fault tolerance	Resource usage	Gas consumption
Thakkar et al.[52]	Hyperledger Fabric	✓	✗	●	●	○	○	○	○
Kuzlu et al.[48]	Hyperledger Fabric	✗	✗	●	●	●	○	○	○
Baliga et al.[41]	Hyperledger Fabric	✗	✗	●	●	○	○	○	○
Nakaike et al.[122]	Hyperledger Fabric	✗	✗	●	●	○	○	○	○
Saingre et al.[49]	Hyperledger Fabric, Ethereum	✓	✗	○	○	○	○	●	●
Dinh et al.[44]	Hyperledger Fabric, Ethereum, Parity	✓	✗	●	●	●	●	○	○
Wang et al.[51]	Hyperledger Fabric, Ethereum, Hyperledger Sawtooth, FISCO-BCOS	✗	✗	●	●	○	○	●	○
Aldweesh et al.[40]	Ethereum	✗	✗	○	○	○	○	●	●
Sedlmeir et al.[53]	Ethereum, Parity, Quorum	✓	✗	●	●	○	○	○	○
Gramoli et al.[54]	Algorand, Avalanche, Diem, Ethereum, Quorum, Solana	✓	✗	●	●	○	●	○	○
Baliga et al.[42]	Quorum	✗	✗	●	●	○	○	○	○
This work	Cosmos (Tendermint BFT)	✓	✓	●	●	●	○	○	○

The performance of Hyperledger Fabric has been extensively studied. The latency, throughput and scalability of Fabric v1.0 are analyzed by Baliga et al. [41] using a set of micro-benchmarks that combine different numbers of read and write operations to the blockchain and vary the number of channels and peers in the blockchain network. The same version of the platform is evaluated with the use of custom workloads by Thakkar et al. [52]. Based on experimental results, the

authors propose optimizations that provide a 16x increase in system throughput. A newer version of Hyperledger Fabric, v1.4 is analyzed by Kuzlu et al. [48]. The authors evaluate the platform's performance by conducting several experiments characterized by distinct types of transactions and submission rates.

Quorum, an Ethereum-based blockchain platform for consortium blockchains is evaluated by Baliga et al. [42]. Two different consensus algorithms are analyzed. The evaluation is conducted by executing micro-benchmarks with distinct amounts of read and write operations to smart contracts to obtain the transaction throughput and latency achieved by the platform.

BLOCKBENCH, proposed by Dinh et al. [44], presents the first blockchain benchmarking framework that can be used to evaluate the performance of private blockchains. The framework divides blockchains into three distinct architectural layers and uses different macro and micro-benchmarks to analyze the performance of each layer. The provided workloads are designed to evaluate system throughput, latency, scalability and fault tolerance. The authors display the capabilities of their framework by applying it to analyze the performance of the Ethereum, Parity and Hyperledger platforms.

Saingre et al. propose *BCTMark* [49], a generic framework for blockchain benchmarking. *BCTMark* aims to be portable and provide reproducible results by providing tools for load generation, network emulation and metric management.

DLPS, an open source framework for the performance evaluation of enterprise blockchains is proposed by Sedlmeir et al. [53]. Similar to the majority of related blockchain benchmarking frameworks the authors chose only throughput and latency as performance indicators. However, unlike most works, they provide formal definitions for those metrics. To showcase their proposed framework, the authors employ it to analyze Ethereum, Hyperledger Fabric, Hyperledger Indy, Quorum and Hyperledger Sawtooth blockchain platforms.

Opbench is a benchmarking tool proposed by Aldweesh et al [40]. It focuses on measuring the CPU time required for miners to process operation codes in Ethereum smart contracts. The application measures the performance of Python and Golang-based Ethereum clients and aims to aid miners in choosing tasks that

will reward them adequately based on the computational work they are required to perform.

Another general blockchain performance evaluation tool, called *xBCBench*, is proposed by Wang et al. [51]. The proposed tool is used to analyze the throughput and latency achieved by Ethereum, Hyperledger Fabric, Hyperledger Sawtooth and FISCO-BCOS. However, the authors do not define a standard set of workloads to be used in the evaluations. Instead, different benchmarks are utilized for the evaluated platforms without further elaboration on those choices.

The novelty and contributions of our work come from the evaluation of cross-chain transactions. Such an evaluation presents additional challenges compared to the analysis of isolated blockchains due to the involvement of different systems in addition to external applications for the transport of data. Recent works discuss the performance and conformity of cross-chain communication [123], [59], [97], however, none of them provide a concrete framework to guide the process of empirically evaluating the performance of cross-chain communication. Furthermore, this work is the first to provide a comprehensive performance evaluation of cross-chain communication using the Inter-blockchain Communication Protocol and Tendermint blockchains [124], [125].

4.7 Chapter Summary

This chapter presented a novel framework to guide the empirical evaluation of cross-chain communication protocols and used it to guide the implementation of an open-source performance benchmarking tool. The developed tool was leveraged to conduct the first comprehensive evaluation of the throughput, latency and scalability of the IBC protocol paired with the Hermes Relayer. Based on experimental results, we present an analysis that identifies limitations in the design of both the IBC protocol and the Hermes Relayer. Those limitations lead to bottlenecks that decrease the protocol's performance and also prevent it from scaling to meet increased demand. Furthermore, this chapter uncovers challenges, caused by implementation and design choices, that add complexity to the deployment and utilization of IBC and the Hermes Relayer. As a contribution to future research and development of cross-chain communication and the examined systems, a dataset

containing 158GB of execution logs, obtained through our experiments, is made publicly available.

Chapter 5

Towards Scalable Cross-Chain Messaging

Relayers form the backbone of cross-chain data transportation in multiple platforms such as the Cosmos Network, Ethereum and Polkadot. In most cases, the data relaying process is made permissionless in order to avoid the introduction of additional trust requirements. As a result, relayers work independently and are able to join and leave the system at any moment. While this increases the flexibility of cross-chain relaying models and enables anyone to assist in maintaining the network by transporting data, it also introduces limitations that are neither explored nor addressed in permissionless settings.

This chapter addresses *Research Objective 3* by identifying and exploring three issues that are present in the design of current permissionless cross-chain relaying models. It proposes a novel protocol to improve scalability and eliminate the concurrency problems in those models by coordinating the delivery of cross-chain data among relayers. Based on the proposed protocol, this chapter also presents a detailed analysis of the trade-offs that must be taken into account when designing cross-chain relaying protocols for permissionless environments.

5.1 Introduction

Despite the abundance of existing strategies, there are still opportunities for the improvement of cross-chain communication, particularly when it comes to identifying and addressing the performance limitations of existing protocols [32]. For example, recent work highlights unadressed performance and scalability issues that affect cross-chain relayers and the transport of data in the context of the Inter-Blockchain Communication Protocol (IBC) [35].

In this work we explore this research gap and focus on improving the transport of data between blockchains. In particular, we look at the shortcomings of cross-chain relayers and how they can adversely impact a system's performance, scalability and security. Relayers are an essential part of the infrastructure of various

cross-chain protocols and bridging services[1] including the IBC protocol, Nomad Protocol, Gelato Network, Keep3r Network, Darwinia Network and LayerZero.

We identify three issues that impact cross-chain relayers deployed in a permissionless setting, namely race conditions, poor scalability, and transaction reordering and censorship. Specifically, we discuss how those issues arise due to the lack of coordination among relayers, a common characteristic among current systems. We take the first step towards addressing race conditions and scalability issues by proposing a protocol to enable coordination among independent relayers.

We design our protocol to be adaptable and operate without requiring trusted third parties. Leveraging it as a foundation, we provide an in-depth discussion about the considerations and the trade-offs associated to the design of relayer coordination protocols including incentives, task allocation and reliability. Through this work we provide a foundation for further research on cross-chain communication. Our contributions are summarized as follows:

- ***Contribution 1*** *(Section 5.2)*: We identify three issues derived from the lack of coordination mechanisms in cross-chain relaying systems, namely race conditions, poor scalability and transaction reordering and censorship.
- ***Contribution 2*** *(Sections 5.3, 5.4, 5.5)*: We introduce a novel protocol that addresses race conditions and scalability issues by enabling coordination among independent relayers.
- ***Contribution 3*** *(Section 5.6)*: We provide an in-depth discussion on the trade-offs and challenges associated with the coordination of independent relayers in a permissionless setting.

5.2 Cross-chain message relaying

Relayers are off-chain applications responsible for delivering data between a pair of blockchains. They usually operate by scanning blockchains for pending

[1]https://docs.nomad.xyz/the-nomad-protocol/overview;
https://docs.gelato.network/introduction/architecture;
https://keep3r.network/;
https://docs.darwinia.network;
https://layerzero.gitbook.io/docs/ecosystem/relayer

cross-chain transactions or by receiving jobs from a component that aggregates and assigns data delivery tasks.

In the context of permissionless blockchains, it is desirable for relayers to operate in a trustless fashion. This allows them to be employed without introducing additional points of trust. In practice, this is achieved by performing the validation of all cross-chain data in the communicating blockchains. This approach enables blockchains to detect when data has been tampered with during transport and in turn, allows the relaying process to be permissionless. This virtually removes the barrier to entry and allows anyone to deploy a relayer at any moment.

A permissionless relaying model contributes to the decentralization of data delivery, however, it simultaneously introduces challenges related to concurrency, incentive mechanisms and performance bottlenecks. Although some blockchain services acknowledged and sought to address part of those issues [126]–[128], details regarding the proposed approaches are either unavailable or lacking in depth.

5.2.1 Relayer issues and games

By leveraging the performance analysis tool presented in *Chapter 4* and discussions from recent works on cross-chain communication protocols [129], [130] we have observed three issues that negatively impact permissionless relaying systems.

5.2.1.1 Race conditions

In many settings, such as cross-chain channels established using the IBC protocol, relayers are able to join and leave the network at any moment. In this case, the set of relayers working to deliver data between a pair of blockchains is unknown, both to the blockchains and among the relayers themselves. Given their lack of knowledge regarding other participants, when multiple delivery tasks are available, the rational choice for honest relayers is to attempt to deliver as many tasks as possible. This behavior is compounded when mechanisms to reward the delivery of messages are in place.

In current reward mechanisms, users submit operation requests accompanied by fee information and an amount of tokens to a source blockchain. The tokens

are held by the blockchain while the operation is pending. When messages are delivered, their fee information is utilized to unlock rewards for the corresponding relayer. A key point is that the reward fee is only released upon the successful delivery of a message, and is given only to the address of the relayer that delivered it. Should a message fail to be delivered before a user-defined timeout interval, the locked tokens are refunded to the user who initiated the operation.

From a relayer's standpoint, adopting a reward seeking behavior can be justified by the two types of expenses they incur to operate [131]. The first expense stems from maintaining the infrastructure required to deploy and run blockchain nodes. The second is associated with the fees that must be paid for the submission of transactions to the blockchains. However, while such profit seeking behavior is logical from a relayer's own standpoint, in the presence of multiple independent relayers it leads to race conditions and winner-takes-all games.

Assume two independent blockchains, $Chain_A$ and $Chain_B$, connected by a set of independent relayers $\{R_1, R_2, R_3\}$. Consider a pool of pending tasks containing three cross-chain transactions $\mathcal{P} = \{tx_1, tx_2, tx_3\}$ to be delivered from $Chain_A$ to $Chain_B$. To deliver cross-chain transactions, relayers must pay a transaction processing fee, referred to as *gas*, to block miners in the receiving blockchain. Transactions require a minimum amount of gas to be accepted, proportional to the computational effort they require to be executed and included in the blockchain by the miners. Gas fees can be adjusted and any excess amount is awarded as a bonus to miners. This means that relayers can choose to pay increased fees for the transactions they deliver, causing any rational, profit-seeking miners to prioritize the inclusion of those transaction in the blockchain. Different relayers can submit the same transaction, however, only the first valid submission is rewarded. Moreover, redundant submissions are rejected, but they still have to pay fees. This leads to several scenarios with different outcomes:

- *Scenario I* - Each of the relayers, R_1, R_2 and R_3, attempts to deliver tx_1, tx_2, tx_3 to $Chain_B$ as soon as they are included in $Chain_A$ and become available in the pool of pending tasks. All relayers use a default fee estimation algorithm and pay the same gas fee for each of the three transactions. The three relayers

manage to submit all three transactions each to $Chain_B$ at roughly the same time. As transactions are selected for the next block, it is up to the miner to order the transactions submitted by R_1, R_2 and R_3 as long as the order of transactions submitted by a single user is preserved (e.g, given R_1's transactions, tx_2 may not be executed before tx_1). Given that they all pay the same fee and there's no profit to be gained by choosing one over the other, the miner orders them arbitrarily such that those submitted by R_1 are executed first. This causes the transactions delivered by R_1 to be executed successfully and leads those delivered by R_2 and R_3 to be rejected upon execution, preventing them from changing the blockchain's state. Despite R_2's and R_3's transactions being rejected, the fees spent for their submission is still consumed. This outcome not only causes R_1 to be the only relayer rewarded for their work but also incurs a loss for R_2 and R_3.

- *Scenario II* - R_1, R_2 and R_3 attempt to deliver tx_1, tx_2, tx_3 to $Chain_B$ as soon as they are included in $Chain_A$. This time, however, R_3 chooses to pay an increased gas fee for each one of the transactions while R_1, R_2 both pay a smaller, default fee. All transactions reach $Chain_B$ at roughly the same time. A rational miner chooses to include the transactions submitted by R_3 first as they are more profitable. This leads R_3 to be rewarded, albeit less given that the received rewards must also be used to cover the higher fees paid for transaction submission. It also causes R_1 and R_2 to have their transactions rejected and their fees spent. In this case, R_3 managed to get its transaction executed first by choosing to forgo a portion of the rewards. It may also be possible that all relayers choose to pay increased fees, leading to even greater losses for those whose transactions are not executed first.

- *Scenario III* - When tx_1, tx_2 and tx_3 are included in $Chain_A$, both R_1 and R_2 attempt to deliver all of them to $Chain_B$. R_3, however, chooses to deliver only tx_1 and tx_2 first, before attempting to deliver tx_3. R_1 and R_2 deliver the transactions at approximately the same time. R_3 however, spends less time retrieving transaction data from $Chain_A$ and processing it since it only attempts to deliver two out of the three available transactions. This leads R_3

to deliver tx_1 and tx_2 before block h, whereas both R_1 and R_2 deliver their transactions after block h and before block $h + 1$. When block h is minted, the miner includes the transactions tx_1 and tx_2 delivered by R_3. For the next block, the miner decides between R_1's and R_2's tx_3, based on the amount of fees they paid for the transaction. In this scenario, R_2 takes advantage of the fact that processing and delivering fewer transactions requires less time and claims the reward for both tx_1 and tx_2, leaving the other relayers to compete for the delivery of tx_3.

5.2.1.2 Performance and scalability

Performance issues exist as a direct consequence of the competition among independent relayers. Increasing the number of workers does not lead to an increase in cross-chain throughput. This issue is compounded by the fact that, in the presence of multiple relayers, performance is decreased during times of high network volume [35]. Moreover, when there is a surge in the number of pending transactions, relayers may attempt to retrieve and deliver a large amount of data at once. This may cause them to crash or behave unexpectedly [132], [133]. As a consequence, in a system where relayers work independently, not only performance is unable to scale but it may also be negatively impacted.

Another performance concern is derived on the reliance of relayers on blockchain full nodes to retrieve transaction data. For example, on Tendermint-based blockchains, the RPC server run by full nodes processes data queries sequentially. This may degrade performance and significantly increase waiting times when relayers attempt to retrieve data for a large amount of transactions at once [35]. Distributing tasks among relayers can assist in mitigating this problem as relayers often run their own blockchain nodes for data retrieval purposes. In this manner, different nodes can process different subsets of the data queries concurrently.

5.2.1.3 Transaction reordering and censorship

Due to the open nature of distributed systems the set of relayers connecting a pair of blockchains is likely to be composed by heterogeneous machines with ac-

cess to different network infrastructure. This leads to some relayer operators being capable of processing and delivering cross-chain transactions faster than others. A similar situation happens in the presence of well-funded relayers who are willing to play a high-risk and high-reward game by paying increased transaction fees in an attempt to outbid its competitors. In either case, a single relayer or a group of colluding relayers may end up monopolizing the delivery of transactions between a pair of blockchains. In the long run, this might lead less powerful relayers to leave due to sustained financial loss. The remaining relayer or group of relayers will then be capable of selectively censoring transactions.

In the absence of mechanisms to enforce transaction ordering across different blockchain networks, monopolistic relayers can reorder transactions to perform attacks similar to front-running and sandwich attacks [134], [135]. Consider, for example, Tendermint blockchains, which constitute the majority of the blockchains that currently take part in cross-chain communications via the IBC protocol. These blockchains employ a first-in, first-out memory pool system, where the execution order of transactions depends on their arrival sequence at a node's memory pool [136]. In the event that a powerful relayer is left as the only maintainer of a channel, it becomes the only link between the blockchains. Without transaction ordering mechanisms, the relayer can retrieve transactions from a source blockchain and alter their order during submission to the destination to perform an attack. For example, consider a transaction tx_1, committed to a source blockchain $ChainA$, containing a request for tokens to be swapped in a decentralized finance protocol in $ChainB$. Once tx_1 is included in $ChainA$, the relayer becomes aware of it and realizes the attack opportunity. Before relaying tx_1 to $ChainB$, the relayer operator crafts two transactions, tx_2 and tx_3 and submits them to $ChainA$. Once tx_2 and tx_3 are committed, the relayer submits all transactions to $ChainB$ in such a way that tx_1 is "sandwiched" between tx_2 and tx_3. Given the memory pool's first-in first out implementation, the malicious relayer successfully affected the order in which transactions are executed. Similarly, the relayer may craft a single transaction to preclude another user's transaction in the destination blockchain, effectively executing a front-running attack.

5.2.2 Our proposal to improve cross-chain relaying

Improving performance and eliminating the issues present in current cross-chain message relaying models requires enabling coordination between independent, potentially distrustful relayers. One way to achieve this is by designing a protocol to distribute the message delivery load among all available relayers, allowing each of them to work independently on a subset of the pending messages. However, merely distributing messages does not suffice for the purpose of achieving coordination in a permissionless setting. Relayers must also have an incentive to follow the protocol. While relayers might be motivated to cooperate and deliver only their respective subset of messages for the benefit of the network, they are also compelled to deviate from the protocol by the prospect of increasing their profit, e.g, by getting paid for delivering more messages. As a consequence, the aforementioned issues are likely to persist. Taking this into consideration, we conclude that in order for a coordination protocol to be effective in a permissionless setting it must implement adequate accountability mechanisms. In this light, we take the first step towards addressing the issues presented in this section by proposing a protocol to coordinate cross-chain message delivery, particularly in scenarios involving multiple relayers.

5.3 System Model

5.3.1 Blockchains and transactions

We consider two independent, tamper-proof and append-only blockchains, $Chain_A$ and $Chain_B$. Both satisfy the persistence and liveness properties of distributed ledgers [137]. A blockchain consists of a sequence of blocks, each containing a set of transactions that modify the state of the system when executed. Transactions are submitted by users, participants who possess a blockchain account secured by a public and a private key. Blocks and their contents are publicly visible and are generated according to a protocol that is specific to each blockchain.

5.3.2 Contracts

A contract is a publicly visible, deterministic program deployed in a blockchain. Each contract has an account capable of sending and receiving tokens, but unlike user accounts, they are controlled by contract code. Contracts define a set of functions that can be invoked by users to read, perform operations and write data to the ledger where it resides. Functions are invoked by submitting transactions containing function calls to the blockchain where the contract is deployed. Contracts cannot directly access information outside the ledger, but can receive external data (such as from other ledgers) as input.

5.3.3 Network model

We assume a synchronous network, where messages are delivered within a known upper bound of delay. Nodes are well-connected and communication is reliable, therefore messages that have been sent are not lost.

5.3.4 Cross-chain communication

Blockchains are unable to directly exchange information with any external systems. To communicate, two blockchains implement a common cross-chain messaging format such as XCM [138], and employ relayers to perform the transport of information between them. We refer to the connection between a pair of blockchains as a channel and say that relayers who work to deliver messages between $Chain_A$ and $Chain_B$ maintain the channel between those systems. When a blockchain sends information through the channel, it is referred to as the source. When it receives information, it is referred to as the destination.

5.3.5 Threat model

We consider that cross-chain channels are maintained by relayers who deliver messages for profit. Therefore, relayers may attempt to deviate from the protocol in an attempt to increase their rewards. Furthermore, relayer operators are able to determine the geographic locations of blockchain nodes and strategically position their infrastructure to gain an advantage in transaction delivery times. We also

consider the existence of relayers with diverse economic capabilities. For example, some relayers may be well-funded and willing to pay higher fees in an attempt to give their transactions priority, while other relayers may be risk-averse.

5.3.6 Cryptographic assumptions

Blockchain users have a public and private key. Public keys are used to receive transactions and are known to all system participants. Private keys are used to sign transactions and prevent forgery. Our protocol assumes a uniformly distributed, collision-resistant cryptographic hash function.

5.4 Design goals

Our objective is to design a protocol that enables coordination between multiple relayers working to deliver messages for a cross-chain channel. By achieving coordination we address two issues with current relaying models. First, we allow the cross-chain message delivery load to be balanced among available relayers, increasing the throughput of cross-chain communication. Second, we eliminate race conditions, encouraging more relayers to join the network and consequently reducing the chance of transaction censorship attacks. We now put forth three properties that are key to achieving our goals and use them to guide the design of our protocol.

Property I (*Scalability*). Scalability specifies that the throughput collectively achieved by the set of relayers that maintain a cross-chain channel should increase when the number of relayers increases.

Because different relayers may have access to different hardware and network resources, performance cannot be guaranteed to be directly proportional to the number of relayers. Therefore we only assume that when the number of relayers maintaining a channel increases from n to $n + 1$, the throughput of the channel also increases. By satisfying this requirement a protocol is able to adapt to a growth in cross-chain message delivery demands by employing more relayers.

Property II (*Accountability*). Accountability specifies that a relayer must be held responsible, particularly through the imposition of punishment, should they fail to deliver messages assigned to them before their specified timeout.

This property is essential for protocols operating in permissionless environments. Well-designed accountability mechanisms contribute to minimizing the occurrence of misbehavior and help ensure that participants act in the best interest of the network.

Property III (*Fair allocation*). This property specifies that a channel's message delivery workload should be uniformly distributed among the set of relayers maintaining the channel.

We consider this strategy fair as it ensures that every relayer has an equal opportunity to deliver messages and profit from supporting the network. While this approach may not be ideal in the context of traditional distributed systems, where load balancing algorithms can tailor the distribution of tasks according to the performance of each specific machine, blockchains present unique constraints that make the implementation of complex load balancing algorithms particularly challenging. We discuss this further in Section 5.6.

5.5 A protocol for relayer coordination

The first step towards achieving coordination among relayers is enabling them to communicate. Despite being unable to directly exchange information with each other, relayers share a connection with the blockchains at both ends of the channel. We leverage this in our design and propose an on-chain mechanism to manage the cross-chain channel between a pair of blockchains. We call it the COORDINATOR.

The COORDINATOR must be implemented by both of the connected blockchains. We envision the COORDINATOR as a smart contract, as multiple major blockchain platforms support such constructs. This increases the compatibility of our proposal. One benefit of contracts is their ability to be deployed on-chain to enable cross-chain communication without the need to modify the system's source code. This allows various applications to deploy different COORDINATOR contracts tailored to their own needs. But, although we assume COORDINATORS to be contracts, they may also be implemented as modules in blockchains that support modular applications, like those built using the Substrate Framework and the Cosmos SDK. Implementing the COORDINATOR as a blockchain module offers advantages, such as the use of programming languages with more advanced features, over-

coming certain contract restrictions such as limited access to certain types of blockchain data, and avoiding the high fees associated with computations performed by smart contracts. However, deploying modules requires changes to the blockchain's source code, therefore it requires high privileges, making it unachievable for the majority of blockchain users.

Deploying the COORDINATOR contracts requires the assistance of an external party. This process, however, does not introduce any trust assumptions because contract code is publicly visible and deterministic. Once deployed, both contracts can be inspected for correctness before being used. As the contracts are deployed in different blockchains, they are unable to directly verify operations executed by one another. To overcome this limitation, cross-chain protocols implement a mechanism to keep track of the counterparty blockchain's state and validate incoming data. In our protocol, we assume that the contracts implement a light client each for cross-chain proof verification [139]. Relayers are responsible for keeping the light clients updated by appending updated block headers to incoming cross-chain messages.

To enable coordination and achieve *scalability, accountability* and *fair allocation* our protocol requires a COORDINATOR to implement the logic for membership management, task allocation, message delivery and incentivization.

5.5.1 Membership management

Active relayers must be tracked in order to perform the allocation of tasks and distribute incentives. We achieve this by requiring relayers to undergo a registration procedure when joining a channel.

Registration. Relayers can start the registration process by publishing a transaction that invokes the `register()` function implemented by the COORDINATOR. This function receives the relayer's public key $pubkey$ and an amount of collateral defined by the contract. The collateral is held and slashed to punish the relayer in case of misbehavior, providing *accountability*. The contract then generates a unique identifier id for the newly registered relayer. The relayer's public key, id and amount of deposited collateral are stored by the contract and it is included in

the set of active relayers $\mathcal{R}$.

Withdrawal. When a relayer no longer wishes to work towards maintaining a channel it publishes a blockchain transaction containing an invocation to the `withdraw()` function. This function receives the relayer's public key and starts an unbonding period, after which the relayer can reclaim its collateral. When the function is triggered, the requesting relayer is immediately removed from $\mathcal{R}$ to prevent it from receiving new message delivery tasks. Pending tasks allocated to this relayer before the start of the unbonding period must still be delivered. In the event a relayer fails to deliver any of the pending messages, it is punished proportionally by having its collateral slashed before it can be reclaimed. To ensure that this punishment cannot be evaded, the unbonding period must be set to a duration that exceeds the longest timeout associated with the relayer's pending tasks. For example, suppose that a relayer wishes to withdraw from a channel but still needs to deliver one pending message that times out at block h. The unbonding period must not end before block $h+1$ and ideally should be made longer to accommodate possible delays in message delivery and confirmation. Another possibility is for a relayer with a small amount of collateral remaining to attempt to avoid punishment by ceasing to deliver messages after sending a withdraw request. In this scenario, the penalty for failed deliveries may exceed the amount of locked collateral, allowing the relayer to get away without paying the full penalty. As a countermeasure, the contract can enforce a minimum amount of remaining collateral in order to continue receiving tasks. Once this amount is reached the start of the unbonding period is automatically triggered.

5.5.2 Task allocation

To eliminate concurrency and achieve *scalability*, message delivery tasks must be distributed among the relayers that maintain a channel. We approach this challenge with the aforementioned *Fair allocation* property in mind and propose two task allocation approaches. These are based on the concept of modular hashing, a technique that helps in uniformly distributing tasks among the relayers. For simplicity, we consider only the transfer of tokens between blockchains in our exam-

ples, however, the message handling logic can be applied to any cross-chain operation implemented by the contracts.

Approach 1. This approach allocates transactions to relayers when they are submitted via the COORDINATOR. As a prerequisite, the programming language used to implement the COORDINATOR contract must offer a method for obtaining the transaction hash of transactions that invoke the contract, analogous to the `getTxId()` method available to Hyperledger Fabric chaincode [21]. Task allocation is performed when a cross-chain operation is successfully initiated through the `Coordinator`. Contracts are unable to modify transactions, therefore users that wish to initiate cross-chain operations must submit data that follows the format supported and enforced by the contracts. This can be achieved by using an external application to prepare and submit transactions. Suppose a user submits a transaction invoking the `transfer()` function, which implements the logic to send tokens from an account in a source blockchain to another account in a destination blockchain. As input, it receives the sender's account, the recipient's account, the amount of tokens to be transferred and a timeout for the operation. Upon successfully validating the transaction data, the contract escrows the user's tokens in order to prevent them from being spent while the cross-chain transfer is pending. Then, it retrieves the hash of the transaction that invoked the `transfer()` function, $txHash$. The transaction hash is used to perform a modulo hashing operation $i = H(txHash) \mod m$, where i is the resulting index, H is a cryptographic hash function and m is the size of the registered relayer set $\mathcal{R}$. The resulting index is used to access the position of a relayer in $\mathcal{R}$ and get its corresponding id. The COORDINATOR maps the id to $txhash$ and stores that information together with the transaction's timeout to create a pending delivery task. As the new transaction is included in the blockchain, relayers are able to retrieve information regarding its allocation.

Approach 2. This approach requires users to compute the allocation of transactions and submit it to the COORDINATOR. As a prerequisite, the COORDINATOR contract must be capable of retrieving a recorded transaction for a given hash value. An example is the `getHistoryForKey()` method supported by Hyper-

ledger Fabric chaincode [140]. Alternatively, the same logic may be implemented by utilizing an oracle service to query the blockchain for confirmed transactions.

Similar to Approach 1, a user invokes a contract function to initiate a cross-chain operation and provides transaction data. The contract validates the transaction, but we assume that it does not have access to the hash of the contract invocation transaction, therefore it cannot perform the modulo hashing operation at this time. Instead, the contract implements an `assignTasks()` function that can be invoked to provide the contract with the allocation for the newly added transactions. This function receives a list of transaction hashes together with the id of the assigned relayer. A user or relayer can perform task allocation by watching the blockchain for new cross-chain transactions and querying the COORDINATOR for the current set of registered relayers. This information can then be used to perform the modulo hashing operation. When the contract receives task allocation information via the `assignTasks()` function, it verifies if the received transaction hashes identify a cross-chain operation included in the blockchain and not yet assigned. If so, it executes the task allocation mechanism to check if the allocation was properly performed according to the set of registered relayers. In case the allocation is correct, information regarding the allocated tasks is stored by the contract and the transaction is published in the blockchain. Otherwise, the COORDINATOR marks the transaction as reverted. Users can be incentivized to perform task allocation by providing a reward for the first correct submission.

This approach incurs a drawback proportional to the blockchain's block interval time. Assume a blockchain in which the average block interval is equal to 10 seconds. Once transactions are included in the blockchain, users must perform the allocation and invoke the `assignTasks()` function. However, the function will only be executed when a new block is created, leading to a delay of at least 10 seconds before the task allocations are published in the blockchain. This delay will be proportional to the blockchain's block interval, therefore this approach needs to be considered in a case-by-case basis. We argue that for blockchains with a block interval of a few seconds, this trade-off is beneficial. According to recent work, when a channel is being maintained by a single relayer the latency for cross-

chain transaction completion may be in the order of several minutes during times of high network load [35]. Consequently, this approach trades a small delay, which is required to perform the distribution of the delivery load among relayers, for the ability to prevent prolonged wait times during periods of high network usage.

5.5.3 Message delivery

Relayers retrieve and deliver data by interacting with blockchain full nodes. To receive transaction delivery tasks from a blockchain's COORDINATOR, relayers must be registered with it. To deliver a transaction, however, registration with the destination blockchain's COORDINATOR is not required. We discuss this design choice in detail in Section 5.5.4.

Transaction flow. In our model we consider that cross-chain operations are composed by three steps, each requiring one transaction to be included in the communicating blockchains. The first transaction is included in the source blockchain to *request* a cross-chain operation, the second transaction is included in the destination blockchain as a *receipt* for the processed the operation, the third and final transaction is included in the source blockchain as an *acknowledgement* of the receipt and the result of the operation. When a relayer is assigned to a *request* transaction it becomes responsible for retrieving it from the source, submitting it to the destination and bringing the receipt back to the source blockchain.

When registered, a relayer continually scans the blockchain for new cross-chain transactions and their allocation information. Upon encountering a task assigned to itself, it retrieves the corresponding transaction data from the source blockchain, formats it according to the appropriate cross-chain logic (e.g, by adding extra information) and submits it by invoking the `deliverTx()` function implemented by the COORDINATOR in the destination blockchain. This function receives transaction data and validates it according to the cross-chain communication logic implemented by the COORDINATOR. The contract then executes the logic required by cross-chain operation in the destination blockchain. For a token transfer, for example, it may mint new tokens and send them to the recipient account specified in the received transaction data. The transaction is then included in the destination

blockchain. The relayer retrieves data corresponding to the *receive* transaction from the destination blockchain, performs additional formatting if required and submits a transaction to the source blockchain to invoke the `proveDelivery()` function implemented by the COORDINATOR. This function receives the transaction receipt included in the destination and validates it. The contract may then execute additional logic to complete the operation, if required. In the case of a token transfer, the contract may destroy tokens previously held by the contract when it was initiated. After the received information is processed, the transaction is included in the blockchain to notify the *acknowledgement* of the operation.

Handling timeouts. Transactions may time out, either due to congestion in the destination blockchain or due to relayer inactivity. Timeout intervals are transaction specific and are determined by the issuing users. They are also measured in blocks. For example, a transaction with an associated timeout of 10 blocks will be regarded as having failed to be delivered when 10 blocks are added to the destination blockchain. This interval starts to elapse when the transaction is committed to the source blockchain, preventing it from timing out while it awaits confirmation at the transaction pool. This dispenses with the need for a bound on transaction commit latency. It is important to enforce a minimum value for transaction timeout, so as to prevent users from submitting transactions with very short timeouts to intentionally punish relayers. Such minimum value must be considered on a case-by-case basis according to the average block time interval of the blockchains.

When a transaction times out, the timeout event may be informed to the source blockchain by invoking the `submitTimeout()` function. This function receives and verifies a proof-of-absence, which demonstrates that a given transaction was not included in the destination blockchain within the expected block range. Examples of how such proofs may be constructed can be found in existing work [141], [142].

5.5.4 Relayer incentives

In permissionless settings, it is essential to ensure that participants will not deviate from the protocol for their own profit. We address this challenge and pro-

vide *accountability* by designing an incentive mechanism. This mechanism rewards successful transaction deliveries, punishes inactivity and discourages relayers from delivering transactions that were not assigned to them. As a consequence the proposed mechanism also assists in achieving *scalability* by encouraging relayers to work on separate delivery tasks concurrently.

Delivery fees. Users incentivize relayers through the payment of a delivery fee for completed tasks. The corresponding tokens are sent to the COORDINATOR and held when a cross-chain transfer is initiated. To encourage relayers to take responsibility over all the data delivery steps required for the completion of a cross-chain operation, delivery fees should only be released in the source blockchain when the delivery receipt is brought from the destination. In addition, the reward should always be sent to the relayer to which the task was initially allocated, regardless of who delivered it. This strategy achieves two important goals. First, it compels relayers to deliver a *request* from the source to the destination, retrieve the corresponding *receipt* and submit it back to the source as an *acknowledgement* of the operation before they can earn rewards. Second, it prevents a relayer from earning rewards by delivering transactions allocated to a different relayer. Deviating from the protocol by abandoning an ongoing operation or stealing transactions yields no rewards. This is compounded by the fact that submitting transactions to invoke contract functions requires the payment of gas fees. Consequently, if relayers deviate from the protocol they sacrifice their rewards and incur a loss due to transaction submission fees.

Delivery fees should be carefully considered in order to guarantee the generation of profit for relayers. Considering the two transactions that must be submitted to complete an operation (*receipt* and *acknowledgement*), the corresponding reward should exceed the fees paid to submit those transactions to the destination and source blockchains, respectively.

Punishing misbehavior. When a transaction times out, the relayer responsible for its delivery is punished by having its collateral slashed. This strategy, however, does not encourage a relayer to report timeouts as it leads to a loss of funds. In order to guarantee that timeouts are reported back to the source blockchain, other

relayers may be rewarded for submitting a proof to report the timeout of a transaction to the source blockchain's COORDINATOR. They may track the timeouts by comparing the transaction allocation information stored in the source blockchain's COORDINATOR contract against the state of the destination blockchain. The first submission of a valid proof of timeout is rewarded with tokens deducted from the inactive relayer's collateral.

When a transaction times out, the user that initiated the operation also incurs a loss due to the gas fees spent on transaction submission. In order to mitigate this, users can also receive a portion of the tokens deducted from the inactive relayer's collateral to cover the cost of their transaction.

5.6 Discussion and research challenges

Similar to classical distributed systems, there is no one-size-fits-all approach when it comes to balancing workloads and allocating tasks in the context of cross-chain communication. The specific requirements of each application and the environment it will be deployed in must be considered to pick a suitable strategy. This often means prioritizing some features over others, resulting in trade-offs. We now discuss those trade-offs and design choices. Furthermore, we explore how the proposed protocol can be implemented in order for its performance to be evaluated in real-world scenarios.

5.6.1 Performance vs. reliability

By enabling relayers to work on independent tasks concurrently, a protocol achieves scalability and becomes capable of adapting to increases in cross-chain transaction volume. Although this improves performance, it removes redundancy and consequently reduces the reliability of a relaying service. When different relayers attempt to deliver the same task simultaneously, the likelihood of it being successfully delivered is higher compared to when it is assigned to a single relayer. When tasks are allocated with the objective of eliminating redundant work, system liveness may be compromised if several relayers stop responding simultaneously. In contrast, in current models, as long as a single relayer remains operational liveness is maintained (assuming that it can handle the channel load). A potential

solution to this problem is an incentive mechanism that discourages relayers from stealing other's tasks but incentivizes their delivery if they are near their timeout. Other possibilities are allowing redundancy to be configured through a parameter that dictates the number of relayers which will receive the same task. A system may also adapt to demand dynamically, prioritising performance over redundancy as the volume of transactions increases. In such systems, incentive models should be considered carefully to prevent inactive relayers from being rewarded for the effort of others, a phenomenon known as free-riding.

5.6.2 Efficient task allocation

We acknowledge that while the simplicity of our task allocation mechanisms make them easier to understand and implement, it may also be seen as a limitation of our work. Distributed systems are often composed by participants with heterogeneous resources. For this reason task allocation can be improved by tailoring each relayer's workload to maximize performance without exceeding their capabilities as it may lead to delays or crashes. While simple task allocation strategies are easier to implement and demand less resources to be executed, their performance may be inferior to more sophisticated, resource-intensive alternatives. Finding the optimal point between resource consumption and performance is a challenge when designing task allocation strategies for blockchain-based systems and is complicated further due to their unique constraints. For example, blockchains proceed in block intervals that may range from a few seconds to several minutes in different systems. Due to this, any change to previously assigned jobs, such as migrating it to another relayer, can only be published to the blockchain upon the creation of a new block and may incur a long delay. Another constraint is the inability of blockchains to access information regarding the hardware and infrastructure resources of participants. This can be addressed by dynamically computing the performance of each individual relayer as it completes tasks and using the resulting metrics as a criterion for task allocation. This, however, can be prohibitively costly if allocation is managed by smart contracts, given the substantial amounts of fees they require to perform complex computations.

5.6.3 Incentives and quality of service

If given the choice, honest, rational relayers will always prioritize the delivery of tasks that yield more profits. For this reason, users often choose to pay higher delivery fees when they wish their transactions to be prioritized. When tasks are automatically allocated relayers are denied the opportunity to select them based on the corresponding reward and users become unable to provide extra incentives to give their transaction priority. Another consequence of removing the reward dynamics in the context of message relaying is the potential impact in the urgency with which relayers deliver tasks. In a competitive environment, relayers are motivated to invest in better infrastructure in order to maximize their chance of profit, leading to an increase in the quality of the service provided. Conversely, in the absence of competition, relayers are not encouraged to invest resources in order to maximize their output. Therefore, while winner-takes-all games pose a disadvantage to less powerful relayers, they simultaneously encourage participants to provide better service in order to earn profits and as a consequence benefit the system.

5.7 Related Work

Despite an abundance of works discussing strategies to achieve communication between independent blockchains, few works explore the performance of cross-chain communication [123], [143] and in particular, data relaying systems [35], [129]. To the best of our knowledge, we are the first to identify the issues associated with permissionless cross-chain relaying systems. We are also the first to take action towards addressing those issues by proposing a protocol to allocate tasks and coordinate independent relayers.

The subject of task allocation and load balancing in classical distributed systems has been widely studied, with works dating from several decades ago [144]–[146]. Such strategies have also been investigated in systems composed by non cooperative participants [147]–[149]. Those subjects, however, have not been explored in the context of cross-chain communication.

5.8 Chapter Summary

Relaying services play a crucial role in the transport of data between blockchains, yet their limitations have not been thoroughly studied. This chapter identifies and discusses unexplored issues in the current design of permissionless cross-chain relayers, including race conditions, inability to scale and security concerns. To address these challenges, a novel protocol to coordinate independent cross-chain relayers in a permissionless environment is proposed. The protocol introduces mechanisms to perform the allocation of tasks among relayers, incentivize their completion and penalize misbehavior. Leveraging our proposal as a foundation, this chapter presents a detailed analysis of the trade-offs that require consideration when designing cross-chain relaying protocols for permissionless settings. Those trade-offs concern performance, reliability, efficiency of task allocation and the quality of service provided by relayers.

Chapter 6

Conclusions and Future Work

This thesis contributed to the advancement of the practice and theory of cross-chain communication by filling critical gaps in the existing literature. Specifically, it addresses *Research Objective 1* by providing an in-depth characterization of blockchain interoperability systems, focusing on their architectural aspects and features. An evaluation framework is proposed and utilized to present a comparative analysis between different blockchain interoperability architectures and the systems that adhere to them. Our framework satisfies *Research Objective 1* by providing a way to identify the unique features of distinct blockchain interoperability systems, thereby facilitating the process of selecting them based on specific requirements. The comparative analysis presented in *Chapter 3* discusses aspects with organizational implications such as portability, decentralization and functional suitability. Additionally, it highlights the limitations of distinct blockchain interoperability systems, enabling practitioners to make key-decisions regarding the adoption of the analyzed solutions. For example, according to our analysis, Atomic Cross-Chain Swaps are among the simplest solutions to deploy, however, they are limited to the execution of token transfers. In contrast, blockchain ecosystems such as Polkadot and the Cosmos Network offer greater flexibility by enabling arbitrary message passing, thereby supporting a virtually unlimited variety of cross-chain operations. However, these ecosystems differ in their security assumptions. While Polkadot provides shared security across its entire ecosystem, Cosmos zones are responsible for their own security. By discussing and comparing those characteristics, this thesis provides a holistic view of the current state of practice and assists the blockchain community in addressing their cross-chain communication needs.

The first comprehensive evaluation of the performance of the IBC protocol is presented in *Chapter 4* of this thesis, thus satisfying *Research Objective 2*. As a part of this process, we proposed a novel framework to guide the development of tools to empirically evaluate cross-chain communication protocols. The proposed frame-

work was applied to guide the implementation of an open-source tool to benchmark cross-chain communications between Cosmos blockchains connected by the Hermes Relayer. Our benchmarking tool was leveraged to conduct several experiments focused on measuring the throughput, latency and scalability of cross-chain communications realized through the IBC protocol. This thesis presented a detailed analysis of the results obtained through those experiments and identified several issues that hinder the performance of the IBC protocol and the Hermes Relayer. For instance, cross-chain channels maintained by two independent relayers achieve a throughput that is 33% lower than the throughput achieved by a single relayer. We identified the source of this issue as the simultaneous execution of pending tasks by all the relayers that work for a channel. This issue also renders the protocol unable to scale. We concluded that this problem cannot be overcome by establishing multiple single-relayer channels between blockchains. This can be explained by the current design of the IBC protocol, which causes transferred assets to be fragmented into multiple non-fungible representations when they are sent through different channels. Addressing this issue is critical for enabling the protocol to adapt to the rapidly increasing demand for cross-chain communication [1]. Furthermore, we also observed that the RPC endpoint provided by Tendermint blockchains is unable to process queries in parallel. Relayers rely on those RPC endpoints to retrieve data associated with pending cross-chain transactions. In our analysis, we discovered that relayers allocate approximately 70% of their processing time (313 out of 455 seconds) to waiting for replies from RPC endpoints while handling data for 5,000 cross-chain transfers. Given the significance of this latency, this thesis identifies the performance of RPC endpoints as a critical source of bottlenecks, and raises the development of strategies for their improvement as an open research challenge. Upon further investigation, we discovered a strategy to alleviate the impact of data queries on the latency of cross-chain communication. This approach consists in dividing the submission of cross-chain transfers among spaced intervals rather than submitting them in a single batch. While this strategy reduces processing time by roughly 70%, it trades a decrease in latency for an increase in both transaction fees

[1] Between August and September of 2023, over 56 million cross-chain transactions were performed using IBC. Source: https://mapofzones.com/home?columnKey=totalTxs&period=30d

and the effort required to submit transfers.

Lastly, this thesis attains *Research Objective 3* by building upon the results presented in *Chapter 4* to explore the limitations and design shortcomings of cross-chain data relayers. In relaying models followed in permissionless settings, such in the Cosmos Network and various EVM-based applications, relayers are unable to communicate with each other, therefore they cannot work together to optimize the data transportation process. In our analysis, we identified three primary limitations associated with current relayer models: they instigate competition among relayers and lead to winner-takes-all games; they prevent performance from scaling; and they enable relayers with superior infrastructure to perform transaction reordering and censorship attacks. This thesis introduces those issues as open challenges and takes the first step towards addressing them. This is accomplished by proposing a novel protocol to facilitate coordination among independent relayers operating in permissionless environments. Our protocol introduces two distinct mechanisms. The first mechanism performs the allocation of tasks to relayers associated with a specific cross-chain channel. The second mechanism manages incentives, encouraging the delivery of tasks and penalizing misbehavior. Due to the complexity of efficiently performing task allocation and load-balancing in blockchain systems, our approaches to task allocation are based on a modulo hashing technique. Consequently, we raise as an open challenge the design of more versatile task allocation strategies for relayers. In addition, this thesis leverages the proposed coordination protocol as foundation for a comprehensive discussion on the trade-offs inherent to the design of relayers for permissionless settings. For instance, while the distribution of data delivery tasks among relayers enables performance to scale, it eliminates the redundancy that increases the reliability of current models. Additionally, the absence of competitive dynamics due to task allocation can potentially reduce the urgency for relayers to invest in better infrastructure, thereby impacting performance.

This thesis makes a significant contribution to the advancement of the state of the art in cross-chain communication by taking concrete steps towards enabling blockchains to interact seamlessly.

6.1 Future work

As a result of the work conducted in this thesis, we have identified several directions for future research. We outline these open problems, encouraging their resolution and further advancement of the field of cross-chain communication.

- The Cross-Consensus Message Passing Protocol (XCMP) needs to be evaluated to assess its performance and design. As a significant project in the field of cross-chain communication, the Polkadot's ecosystem protocol is as important as the Inter-Blockchain Communication (IBC) protocol. Performing a comprehensive analysis of XCMP is essential to determine if it also faces throughput, scalability and design challenges like the ones identified in *Chapter 4*. This analysis could be invaluable for the improvement of the protocol, especially in its early stages.

- In this thesis, we evaluated the performance of the IBC protocol when used to connect two Tendermint-based blockchains. Although IBC's current compatibility with other blockchain platforms is limited, it is essential to analyze its performance with heterogeneous blockchain platforms as they start being supported. This is specially important given IBC's goals of being a general messaging protocol aimed at connecting various blockchain platforms beyond the Cosmos Network.

- The results of our evaluation in *Chapter 4* indicate that attempting to scale the performance of an IBC channel by adding more relayers actually results in decreased throughput. Our study, however, is limited to experiments with up to two relayers per channel. Conducting further studies with a larger number of relayers, and possibly faulty or byzantine relayers, could provide additional insights into potential areas for the improvement of the protocol.

- In *Chapter 5*, we proposed a novel protocol to enable coordination among independent relayers. This protocol was specifically designed to address performance and scalability issues present in current permissionless relayer systems. However, the protocol has not yet been evaluated in a real-world sce-

nario. Such an evaluation could provide insights into the protocol's efficacy and potentially highlight issues that may remain unresolved.

- In this thesis we have identified several issues that negatively affect the performance, scalability and security of permissionless cross-chain relayers. Although we have proposed a protocol to address the performance and scalability challenges, security issues caused by the lack of mechanisms to enforce cross-chain transaction ordering remain unaddressed. Further research in this direction is essential to prevent cross-chain relayers from manipulating and unfairly benefiting from the order in which cross-chain transactions are executed.

Bibliography

[1] TT Kuo, HE Kim, and L Ohno-Machado, "Blockchain distributed ledger technologies for biomedical and health care applications", *Journal of the American Medical Informatics Association*, vol. 24, no. 6, pp. 1211–1220, 2017.

[2] D Dujak and D Sajter, "Blockchain applications in supply chain", in *SMART supply network*, Springer, 2019, pp. 21–46.

[3] Z Zheng, S Xie, HN Dai, X Chen, and H Wang, "Blockchain challenges and opportunities: A survey", *International Journal of Web and Grid Services*, vol. 14, no. 4, pp. 352–375, 2018.

[4] P Tasatanattakool and C Techapanupreeda, "Blockchain: Challenges and applications", in *ICOIN 2018*, IEEE, 2018, pp. 473–475.

[5] M Pilkington, "Blockchain technology: Principles and applications", in *Research handbook on digital transformations*, Edward Elgar Publishing, 2016.

[6] S Underwood, "Blockchain beyond bitcoin", *Commun. ACM*, vol. 59, no. 11, 15–17, Oct. 2016.

[7] N Kshetri, "1 blockchain's roles in meeting key supply chain management objectives", *International Journal of Information Management*, vol. 39, pp. 80–89, 2018.

[8] IEEE, "Special Report on Blockchain World", *IEEE Spectrum*, 17th ser., vol. 10, 2017, `https://spectrum.ieee.org/static/special-report-blockchain-world`.

[9] Grand View Research, "Blockchain technology market size, share & trends analysis report by type (private cloud, public cloud), by application (digital identity, payments), by enterprise size, by component, by end use, and segment forecasts, 2023 - 2030", Tech. Rep., 2023. [Online]. Available: `https://www.grandviewresearch.com/industry-analysis/blockchain-technology-market`.

[10] S Nakamoto, "Bitcoin: A peer-to-peer electronic cash system", 2008.

[11] A Baliga, "The blockchain landscape", *Persistent Systems*, vol. 3, no. 5, p. 4, 2016.

[12] S Schulte, M Sigwart, P Frauenthaler, and M Borkowski, "Towards blockchain interoperability", in *Business Process Management: Blockchain and Central and Eastern Europe Forum: BPM 2019 Blockchain and CEE Forum, Vienna, Austria, September 1–6, 2019, Proceedings 17*, Springer, 2019, pp. 3–10.

[13] TTA Dinh, A Datta, and BC Ooi, "A blueprint for interoperable blockchains", *arXiv preprint arXiv:1910.00985*, 2019.

[14] R Brown. "The five ingredients of blockchain interoperability". (2020), [Online]. Available: `https://www.forbes.com/sites/richardgendalbrown/2020/02/13/the-five-ingredients-of-blockchain-interoperability/\#5af0257558a1`.

[15] S Jagati. "Blockchain interoperability: The holy grail for cross-chain deployment". (2020), [Online]. Available: `https://cointelegraph.com/news/blockchain-interoperability-the-holy-grail-for-cross-chain-deployment`.

[16] Chainlink. "What are cross-chain smart contracts?" (2023), [Online]. Available: `https://chain.link/education-hub/cross-chain-smart-contracts`.

[17] Chainlink. "77+ smart contract use cases enabled by chainlink". (2022), [Online]. Available: `https://blog.chain.link/smart-contract-use-cases/`.

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

[19] C Natoli, J Yu, V Gramoli, and P Esteves-Verissimo, "Deconstructing blockchains: A comprehensive survey on consensus, membership and structure", *arXiv preprint arXiv:1908.08316*, 2019.

[20] A Back, M Corallo, L Dashjr, M Friedenbach, G Maxwell, A Miller, A Poelstra, J Timón, and P Wuille, "Enabling blockchain innovations with pegged sidechains", *URL: http://www. opensciencereview.*

com/papers/123/enablingblockchain-innovations-with-pegged-sidechains, vol. 72, 2014.

[21] JO Chervinski, J Yu, and X Xu, "Characterizing blockchain interoperability systems from an architecture perspective", in *International Summit Smart City 360°*, Springer, 2022, pp. 504–520.

[22] A Kiayias and D Zindros, "Proof-of-work sidechains", in *Financial Cryptography and Data Security: FC 2019 International Workshops, VOTING and WTSC, St. Kitts, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers 23*, Springer, 2020, pp. 21–34.

[23] P Gazi, A Kiayias, and D Zindros, "Proof-of-stake sidechains.", *IACR Cryptology ePrint Archive*, vol. 2018, p. 1239, 2018.

[24] J Nick, A Poelstra, and G Sanders, "Liquid: A bitcoin sidechain", 2020.

[25] V Buterin, "Chain interoperability", *R3 Research Paper*, 2016.

[26] S Imoto, Y Sudo, H Kakugawa, and T Masuzawa, "Atomic cross-chain swaps with improved space and local time complexity", in *International Symposium on Stabilizing, Safety, and Security of Distributed Systems*, Springer, 2019, pp. 194–208.

[27] A Deshpande and M Herlihy, "Privacy-preserving cross-chain atomic swaps", in *International Conference on Financial Cryptography and Data Security*, Springer, 2020, pp. 540–549.

[28] J Gugger, *Bitcoin-monero cross-chain atomic swap*, Cryptology ePrint Archive, Report 2020/1126, `https://eprint.iacr.org/2020/1126`, 2020.

[29] R van der Meyden, "On the specification and verification of atomic swap smart contracts", in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, IEEE, 2019, pp. 176–179.

[30] R Han, H Lin, and J Yu, "On the optionality and fairness of atomic swaps", in *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, 2019, pp. 62–75.

[31] M Herlihy, B Liskov, and L Shrira, "Cross-chain deals and adversarial commerce", *Proceedings of the VLDB Endowment*, vol. 13, no. 2, 100–113, 2019. DOI: 10.14778/3364324.3364326. [Online]. Available: http://dx.doi.org/10.14778/3364324.3364326.

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

[33] J Lofgren. "Ibc: Expanding polkadot's interoperability". (2022), [Online]. Available: https://www.coinbase.com/cloud/discover/insights-analysis/ibc-polkadot-interoperability.

[34] S Network. "Ibc: Expanding polkadot's interoperability". (2021), [Online]. Available: https://scrt.network/blog/secret-plasm-bridge-cosmos-polkadot.

[35] JO Chervinski, D Kreutz, X Xu, and J Yu, "Analyzing the performance of the inter-blockchain communication protocol", in *2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2023. DOI: 10.1109/DSN58367.2023.00026.

[36] A Zamyatin, M Al-Bassam, D Zindros, E Kokoris-Kogias, P Moreno-Sanchez, A Kiayias, and WJ Knottenbelt, "Sok: Communication across distributed ledgers.", 2019.

[37] P Lafourcade and M Lombard-Platet, "About blockchain interoperability", *Information Processing Letters*, p. 105 976, 2020.

[38] M Herlihy, "Atomic cross-chain swaps", in *Proceedings of the 2018 ACM symposium on principles of distributed computing*, 2018, pp. 245–254.

[39] I Foundation. "Ibc protocol: A review of the major developments of 2022". (2022), [Online]. Available: https://medium.com/the-interchain-foundation/ibc-protocol-a-review-of-the-major-developments-of-2022-cff533cf6490.

[40] A Aldweesh, M Alharby, M Mehrnezhad, and A Van Moorsel, "Opbench: A cpu performance benchmark for ethereum smart contract operation code", in *2019 IEEE International Conference on Blockchain (Blockchain)*, IEEE, 2019, pp. 274–281.

[41] A Baliga, N Solanki, S Verekar, A Pednekar, P Kamat, and S Chatterjee, "Performance characterization of hyperledger fabric", in *2018 Crypto Valley conference on blockchain technology (CVCBT)*, IEEE, 2018, pp. 65–74.

[42] A Baliga, I Subhod, P Kamat, and S Chatterjee, "Performance evaluation of the quorum blockchain platform", *arXiv preprint arXiv:1809.03421*, 2018.

[43] M Dabbagh, M Kakavand, M Tahir, and A Amphawan, "Performance analysis of blockchain platforms: Empirical evaluation of hyperledger fabric and ethereum", in *2020 IEEE 2nd International Conference on Artificial Intelligence in Engineering and Technology (IICAIET)*, IEEE, 2020, pp. 1–6.

[44] TTA Dinh, J Wang, G Chen, R Liu, BC Ooi, and KL Tan, "Blockbench: A framework for analyzing private blockchains", in *Proceedings of the 2017 ACM international conference on management of data*, 2017, pp. 1085–1100.

[45] Hyperledger Performance and Scale Working Group, "Hyperledger blockchain performance metrics whitepaper", Tech. Rep., 2018. [Online]. Available: `https://www.hyperledger.org/learn/publications/blockchain-performance-metrics`.

[46] R Wang, K Ye, and CZ Xu, "Performance benchmarking and optimization for blockchain systems: A survey", in *International Conference on Blockchain*, Springer, 2019, pp. 171–185.

[47] C Fan, S Ghaemi, H Khazaei, and P Musilek, "Performance evaluation of blockchain systems: A systematic survey", *IEEE Access*, vol. 8, pp. 126 927–126 950, 2020.

[48] M Kuzlu, M Pipattanasomporn, L Gurses, and S Rahman, "Performance analysis of a hyperledger fabric blockchain framework: Throughput, latency and scalability", in *2019 IEEE international conference on blockchain (Blockchain)*, IEEE, 2019, pp. 536–540.

[49] D Saingre, T Ledoux, and JM Menaud, "Bctmark: A framework for benchmarking blockchain technologies", in *2020 IEEE/ACS 17th International Conference on Computer Systems and Applications (AICCSA)*, IEEE, 2020, pp. 1–8.

[50] X Xu, G Sun, L Luo, H Cao, H Yu, and AV Vasilakos, "Latency performance modeling and analysis for hyperledger fabric blockchain network", *Information Processing & Management*, vol. 58, no. 1, p. 102 436, 2021.

[51] R Wang, K Ye, Y Wang, and CZ Xu, "xBCBench: A Benchmarking Tool for Analyzing the Performance of Blockchain Systems", in *International Conference on Blockchain and Trustworthy Systems*, Springer, 2021, pp. 101–114.

[52] P Thakkar, S Nathan, and B Viswanathan, "Performance benchmarking and optimizing hyperledger fabric blockchain platform", in *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, IEEE, 2018, pp. 264–276.

[53] J Sedlmeir, P Ross, A Luckow, J Lockl, D Miehle, and G Fridgen, "The dlps: A new framework for benchmarking blockchains", 2021.

[54] V Gramoli, R Guerraoui, A Lebedev, C Natoli, and G Voron, "Diablo: A benchmark suite for blockchains", in *Proceedings of the 18th ACM European Conference on Computer Systems (EuroSys), 2023 (to appear)*, 2023.

[55] JO Chervinski, D Kreutz, and J Yu, "Analysis of transaction flooding attacks against monero", in *2021 IEEE international conference on blockchain and cryptocurrency (ICBC)*, IEEE, 2021, pp. 1–8.

[56] SS Kushwaha, S Joshi, D Singh, M Kaur, and HN Lee, "Systematic review of security vulnerabilities in ethereum blockchain smart contract", *IEEE Access*, vol. 10, pp. 6605–6621, 2022.

[57] M Möser, K Soska, E Heilman, K Lee, H Heffan, S Srivastava, K Hogan, J Hennessey, A Miller, A Narayanan, *et al.*, "An empirical analysis of traceability in the monero blockchain", *arXiv preprint arXiv:1704.04299*, 2017.

[58] SS Lee, A Murashkin, M Derka, and J Gorzny, "Sok: Not quite water under the bridge: Review of cross-chain bridge hacks", in *2023 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, IEEE, 2023, pp. 1–14.

[59] R Belchior, L Riley, T Hardjono, A Vasconcelos, and M Correia, "Do you need a distributed ledger technology interoperability solution?", *Distributed Ledger Technologies: Research and Practice*, 2022.

[60] N Kannengießer, M Pfister, M Greulich, S Lins, and A Sunyaev, "Bridges between islands: Cross-chain technology for distributed ledger technology", 2020.

[61] A Lohachab, S Garg, B Kang, MB Amin, J Lee, S Chen, and X Xu, "Towards interconnected blockchains: A comprehensive review of the role of interoperability among disparate blockchains", *ACM Computing Surveys (CSUR)*, vol. 54, no. 7, pp. 1–39, 2021.

[62] A Zamyatin, D Harz, J Lind, P Panayiotou, A Gervais, and W Knottenbelt, "Xclaim: Trustless, interoperable, cryptocurrency-backed assets", in *2019 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2019, pp. 193–210.

[63] R Belchior, A Vasconcelos, S Guerreiro, and M Correia, "A survey on blockchain interoperability: Past, present, and future trends", *arXiv preprint*, 2020.

[64] A Kiayias and D Zindros, "Proof-of-work sidechains", in *International Conference on Financial Cryptography and Data Security*, Springer, 2019, pp. 21–34.

[65] K Karantias, A Kiayias, and D Zindros, "Proof-of-burn.", *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 1096, 2019.

[66] J Kwon and E Buchman, *Cosmos whitepaper*, 2019.

[67] G Wood, "Polkadot: Vision for a heterogeneous multi-chain framework", *White Paper*, 2016.

[68] K Sekniqi, D Laine, S Buttolph, and E Sirer, "Avalanche platform", 2020.

[69] P Frauenthaler, M Sigwart, C Spanring, M Sober, and S Schulte, "Eth relay: A cost-efficient relay for ethereum-based blockchains", in *2020 IEEE International Conference on Blockchain (Blockchain)*, 2020, pp. 204–213.

[70] I Project. "Interledger protocol v4". (), [Online]. Available: `https://interledger.org/rfcs/0027-interledger-protocol-4/`.

[71] E Foundation and Consensys. "Btc relay". (), [Online]. Available: https://github.com/ethereum/btcrelay.

[72] J Burdges, A Cevallos, P Czaban, R Habermeier, S Hosseini, F Lama, HK Alper, X Luo, F Shirazi, A Stewart, *et al.*, "Overview of polkadot and its design considerations", *arXiv preprint arXiv:2005.13456*, 2020.

[73] V Buterin, *Ethereum whitepaper*, 2021. [Online]. Available: https://ethereum.org/en/whitepaper/.

[74] E Androulaki, A Barger, V Bortnikov, C Cachin, K Christidis, A De Caro, D Enyeart, C Ferris, G Laventman, Y Manevich, *et al.*, "Hyperledger fabric: A distributed operating system for permissioned blockchains", in *Proceedings of the thirteenth EuroSys conference*, 2018, pp. 1–15.

[75] E Abebe, Y Hu, A Irvin, D Karunamoorthy, V Pandit, V Ramakrishna, and J Yu, "Verifiable observation of permissioned ledgers", in *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, IEEE, 2021, pp. 1–9.

[76] Y Xiao, P Zhang, and Y Liu, "Secure and efficient multi-signature schemes for fabric: An enterprise blockchain platform", *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 1782–1794, 2020.

[77] DP Le, G Yang, and A Ghorbani, "A new multisignature scheme with public key aggregation for blockchain", in *2019 17th PST*, IEEE, 2019, pp. 1–7.

[78] TM Project, *About monero*, 2021. [Online]. Available: https://www.getmonero.org/resources/about/.

[79] EB Sasson, A Chiesa, C Garman, M Green, I Miers, E Tromer, and M Virza, "Zerocash: Decentralized anonymous payments from bitcoin", in *2014 IEEE Symposium on Security and Privacy*, IEEE, 2014, pp. 459–474.

[80] ISO/IEC 25010, "System and software quality requirements and evaluation", International Organization for Standardization, Tech. Rep.

[81] K Network. "Peace Relay". (), [Online]. Available: https://github.com/KyberNetwork/peace-relay.

[82] A Kiayias, A Miller, and D Zindros, "Non-interactive proofs of proof-of-work", in *Financial Cryptography and Data Security: 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10–14, 2020 Revised Selected Papers 24*, Springer, 2020, pp. 505–522.

[83] W Zhang, J Yu, Q He, N Zhang, and N Guan, "Tick: Tiny client for blockchains", *IEEE Internet of Things Journal*, vol. 9, no. 16, pp. 14 172–14 184, 2020.

[84] J Dilley, A Poelstra, J Wilkins, M Piekarska, B Gorlick, and M Friedenbach, "Strong federations: An interoperable blockchain solution to centralized third-party risks", *arXiv preprint arXiv:1612.05491*, 2016.

[85] A Kiayias, A Miller, and D Zindros, "Non-interactive proofs of proof-of-work.", *IACR Cryptol. ePrint Arch.*, vol. 2017, no. 963, pp. 1–42, 2017.

[86] D Fuelling. "Interledger protocol - hashed-timelock agreements (htlas)". (2020), [Online]. Available: `https://github.com/interledger/rfcs/blob/master/0022-hashed-timelock-agreements/0022-hashed-timelock-agreements.md`.

[87] R Protocol. "Ethereum Atomic Swap". (), [Online]. Available: `https://github.com/confio/eth-atomic-swap`.

[88] Decred. "Decred-compatible cross-chain atomic swapping". (), [Online]. Available: `https://github.com/decred/atomicswap`.

[89] L Gudgeon, P Moreno-Sanchez, S Roos, P McCorry, and A Gervais, "Sok: Layer-two blockchain protocols", in *Financial Cryptography and Data Security: 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10–14, 2020 Revised Selected Papers 24*, Springer, 2020, pp. 201–226.

[90] I Foundation. "Interledger.js monorepo". (2023), [Online]. Available: `https://github.com/interledgerjs/interledgerjs`.

[91] I Foundation. "Ilp connector". (2021), [Online]. Available: `https://github.com/interledgerjs/ilp-connector`.

[92] I Foundation. "Interledger-rs". (2022), [Online]. Available: `https://github.com/interledger/interledger-rs`.

[93] I Foundation. "Interledger protocol - rafiki". (2023), [Online]. Available: https://github.com/interledger/rafiki.

[94] I Foundation. "Ilp client". (2018), [Online]. Available: https://github.com/interledgerjs/ilp.

[95] I Foundation. "Javascript ledger plugin interface version 2". (2019), [Online]. Available: https://interledger.org/rfcs/0024-ledger-plugin-interface-2.

[96] I Foundation. "Interledger ethereum payment channel plugin". (2019), [Online]. Available: https://github.com/interledgerjs/ilp-plugin-ethereum.

[97] T Koens and E Poll, "Assessing interoperability solutions for distributed ledgers", *Pervasive and Mobile Computing*, vol. 59, p. 101 079, 2019.

[98] EN Tas, R Han, D Tse, F Yu, and K Nazirkhanova, "Interchain timestamping for mesh security", *arXiv preprint arXiv:2305.07830*, 2023.

[99] M Vukolić, "The quest for scalable blockchain fabric: Proof-of-work vs. bft replication", in *International workshop on open problems in network security*, Springer, 2015, pp. 112–125.

[100] K Croman, C Decker, I Eyal, AE Gencer, A Juels, A Kosba, A Miller, P Saxena, E Shi, E Gün Sirer, *et al.*, "On scaling decentralized blockchains", in *International conference on financial cryptography and data security*, Springer, 2016, pp. 106–125.

[101] A Gervais, GO Karame, K Wüst, V Glykantzis, H Ritzdorf, and S Capkun, "On the security and performance of proof of work blockchains", in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 3–16.

[102] A Malkov. "Web3 interoperability: The next great leap for businesses". (2022), [Online]. Available: https://www.forbes.com/sites/forbesbusinesscouncil/2022/08/05/web3-interoperability-the-next-great-leap-for-businesses.

[103] A Clarke. "Why interoperability is the key to blockchain technology's mass adoption". (2022), [Online]. Available: https://cointelegraph.com/news/why-interoperability-is-the-key-to-blockchain-technology-s-mass-adoption.

[104] E Abebe, D Behl, C Govindarajan, Y Hu, D Karunamoorthy, P Novotny, V Pandit, V Ramakrishna, and C Vecchiola, "Enabling enterprise blockchain interoperability with trusted data transfer (industry track)", in *Proceedings of the 20th International Middleware Conference Industrial Track*, 2019, pp. 29–35.

[105] WJ Gordon and C Catalini, "Blockchain technology for healthcare: Facilitating the transition to patient-driven interoperability", *Computational and structural biotechnology journal*, vol. 16, pp. 224–230, 2018.

[106] S Johnson, P Robinson, and J Brainard, "Sidechains and interoperability", *arXiv preprint arXiv:1903.04077*, 2019.

[107] P Gaži, A Kiayias, and D Zindros, "Proof-of-stake sidechains", in *2019 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2019, pp. 139–156.

[108] T Hardjono, "Blockchain gateways, bridges and delegated hash-locks", *arXiv preprint arXiv:2102.03933*, 2021.

[109] R Belchior, A Vasconcelos, M Correia, and T Hardjono, "HERMES: Fault-tolerant middleware for blockchain interoperability", *Future Generation Computer Systems*, vol. 129, pp. 236–251, 2022.

[110] *Cross-Consensus Protocols (XCMP, VMP, HRMP)*, 2022. [Online]. Available: https://wiki.polkadot.network/docs/learn-xcm/#xcmp-cross-chain-message-passing.

[111] *Hyperledger Labs YUI*. [Online]. Available: https://labs.hyperledger.org/labs/yui.html.

[112] *An Introduction to Polymer Labs, Cosmos, and IBC (Inter-Blockchain Communication)*, 2022. [Online]. Available: https://tinyurl.com/polymer-labs-IBC (visited on 02/25/2023).

[113] JO Chervinski, D Kreutz, J Yu, and X Xu, *IBC Performance Analysis Tool*, version 1.0.0, 2023. [Online]. Available: `https://github.com/joaotav/ibc-benchmarking`.

[114] E Buchman, J Kwon, and Z Milosevic, "The latest gossip on bft consensus", *arXiv preprint arXiv:1807.04938*, 2018.

[115] C Goes, "The interblockchain communication protocol: An overview", 2020. [Online]. Available: `https://media.githubusercontent.com/media/cosmos/ibc/old/papers/2020-05/build/paper.pdf`.

[116] S Chen and A Seredinschi, *The hermes relayer v1 architecture*, 2022. [Online]. Available: `https://informal.systems/blog/hermes-v1-architecture` (visited on 11/09/2022).

[117] J Yu, D Kozhaya, J Decouchant, and P Esteves-Verissimo, "Repucoin: Your reputation is your power", *IEEE Transactions on Computers*, vol. 68, no. 8, pp. 1225–1237, 2019.

[118] EK Kogias, P Jovanovic, N Gailly, I Khoffi, L Gasser, and B Ford, "Enhancing bitcoin security and performance with strong consistency via collective signing", in *usenix security*, 2016, pp. 279–296.

[119] R Neiheiser, M Matos, and L Rodrigues, "Kauri: Scalable BFT consensus with pipelined tree-based dissemination and aggregation", in *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*, 2021, pp. 35–48.

[120] M Yin, D Malkhi, MK Reiter, GG Gueta, and I Abraham, "Hotstuff: BFT consensus in the lens of blockchain", *arXiv preprint arXiv:1803.05069*, 2018.

[121] C Goes, *Interchain Standards 18 - Relayer Algorithms*. [Online]. Available: `https://github.com/cosmos/ibc/blob/main/spec/relayer/ics-018-relayer-algorithms/README.md`.

[122] T Nakaike, Q Zhang, Y Ueda, T Inagaki, and M Ohara, "Hyperledger fabric performance characterization and optimization using goleveldb benchmark", in *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, IEEE, 2020, pp. 1–9.

[123] R Belchior, P Somogyvari, J Pfannschmid, A Vasconcelos, and M Correia, "Hephaestus: Modelling, analysis, and performance evaluation of cross-chain transactions", 2023.

[124] E Buchman, "Tendermint: Byzantine fault tolerance in the age of blockchains", Ph.D. dissertation, University of Guelph, 2016.

[125] D Cason, E Fynn, N Milosevic, Z Milosevic, E Buchman, and F Pedone, "The design, architecture and performance of the tendermint blockchain network", in *SRDS*, IEEE, 2021, pp. 23–33.

[126] D Network, *What can a cross-chain message relayer earn?*, 2022. [Online]. Available: `https://medium.com/darwinianetwork/what-can-a-cross-chain-message-relayer-get-ffb683b689cb`.

[127] BS Chain, *Cross-chain: Relayer incentives*, 2022. [Online]. Available: `https://bnbsmartchain.com/cross-chain-relayer-incentives/`.

[128] G Network, *Executor operators*, 2022. [Online]. Available: `https://docs.gelato.network/introduction/executor-operators`.

[129] D Olshansky and RR Colmeiro, "Relay mining: Verifiable multi-tenant distributed rate limiting", *arXiv preprint arXiv:2305.10672*, 2023.

[130] AR Raj, *Moving relayer incentives on-chain: Fee middleware, fee grant and budget modules*, 2022. [Online]. Available: `https://medium.com/the-interchain-foundation/moving-relayer-incentives-on-chain-fee-middleware-fee-grant-and-budget-modules-b6a13eced375`.

[131] AR Raj, *Channel upgradability and fee middleware: A closer look at relayer costs and incentivisation*, 2023. [Online]. Available: `https://medium.com/the-interchain-foundation/channel-upgradability-and-fee-middleware-f94d942c56a9`.

[132] R Ruetschi, *Rpc: Allow configuring the underlying tungstenite websocket connection #974*, 2021. [Online]. Available: `https://github.com/informalsystems/tendermint-rs/issues/974`.

[133] G Szabo, *Websocket size constraints #1299*, 2021. [Online]. Available: https://github.com/informalsystems/hermes/issues/1299.

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

[135] L Heimbach and R Wattenhofer, "Eliminating sandwich attacks with the help of game theory", in *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, 2022, pp. 153–167.

[136] Tendermint. "Tendermint documentation: Mempool". (), [Online]. Available: https://github.com/tendermint/tendermint/blob/main/docs/tendermint-core/mempool.md.

[137] J Garay, A Kiayias, and N Leonardos, "The bitcoin backbone protocol: Analysis and applications", in *Advances in Cryptology-EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, Springer, 2015, pp. 281–310.

[138] W Foundation, *Introduction to cross-consensus message format (xcm)*, 2023. [Online]. Available: https://wiki.polkadot.network/docs/learn-xcm.

[139] P Chatzigiannis, F Baldimtsi, and K Chalkias, "Sok: Blockchain light clients", in *International Conference on Financial Cryptography and Data Security*, Springer, 2022, pp. 615–641.

[140] Hyperledger Foundation, *Hyperledger Fabric SDK - ChaincodeStub*, 2021. [Online]. Available: https://hyperledger.github.io/fabric-chaincode-node/release-2.2/api/fabric-shim.ChaincodeStub.html\#getTxID__anchor.

[141] J Yu, M Ryan, and C Cremers, "Decim: Detecting endpoint compromise in messaging", *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 1, pp. 106–118, 2017.

[142] J Yu, V Cheval, and M Ryan, "Dtki: A new formalized pki with verifiable trusted parties", *The Computer Journal*, vol. 59, no. 11, pp. 1695–1713, 2016.

[143] I Mihaiu, R Belchior, S Scuri, and N Nunes, "A framework to evaluate blockchain interoperability solutions", Tech. rep., TechRxiv (Dec. 2021). doi: 10.36227/TECHRXIV. 17093039, Tech. Rep., 2021.

[144] S Srinivasan and NK Jha, "Safety and reliability driven task allocation in distributed systems", *IEEE transactions on Parallel and Distributed Systems*, vol. 10, no. 3, pp. 238–251, 1999.

[145] Y Jiang, "A survey of task allocation and load balancing in distributed systems", *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 585–599, 2015.

[146] JA Bannister and KS Trivedi, "Task allocation in fault-tolerant distributed systems", *Acta Informatica*, vol. 20, pp. 261–281, 1983.

[147] D Grosu and AT Chronopoulos, "Noncooperative load balancing in distributed systems", *Journal of parallel and distributed computing*, vol. 65, no. 9, pp. 1022–1034, 2005.

[148] H Kameda, J Li, C Kim, and Y Zhang, *Optimal load balancing in distributed computer systems*. Springer Science & Business Media, 2012.

[149] T Roughgarden, "Stackelberg scheduling strategies", in *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, 2001, pp. 104–113.