Teleport Network

Cross-chain Platform for Web 3.0 Applications

Whitepaper
May 2022

Version 1.0
Last Updated - May 21, 2022

Abstract

The blockchain ecosystem is moving towards a multi-chain future where different blockchains providing unique and distinct features co-exist with each other. However, fragmentation and isolation between blockchains have become a major constraint in their mainstream adoption, making cross-chain bridges all the more important. The Teleport Network provides seamless cross-chain communications upon which a diverse set of decentralized applications (dApps) can be built. Its Inter-Blockchain Communication Protocol (XIBC) improves upon Cosmos' Inter-Blockchain Communication (IBC) protocol, delivering efficient cross-chain integration, wider interoperability, versatility, upgradability and security.

Table of Contents

I. Introduction

The emergence of a multi-chain landscape has been one of the defining features of the current blockchain ecosystem. Since Ethereum, several other major layer-1 platforms such as Solana, Luna, Avalanche, Fantom, Near, and Flow have emerged, offering competitive advantages for different use cases, scenarios and applications, primarily but not limited to decentralized finance (DeFi), non-fungible tokens (NFTs), and gaming. While each of these blockchains boosts its own vibrant developer and user ecosystem, the fragmentation and isolation between these ecosystems have become a major constraint in their mainstream adoption. It is imperative that we allow developers to build on the best platform for their applications while plugging into other blockchain ecosystems, as well as enable users to interact with the blockchain of their choice best suited to their needs and requirements..

These imperatives are achieved with the emergence of cross-chain bridges. As the primary mechanism for permissionless asset transfers between different blockchains, these bridges have become important gateways for enabling the seamless flow of assets and capital, as well as frictionless communication between blockchains throughout the crypto ecosystem. By assessing the total value locked (TVL) in the bridges that connect the various layer-1 ecosystems, we can see the explosive growth of these solutions. According to The Block Research[1], TVL in cross-chain bridges increased from approximately $667 million at the start of 2021 to over $32 billion by November of the same year[2], as measured by their smart contracts on Ethereum. With the continued growth of layer-1 ecosystems, the multi-chain landscape will only become more important, and the role of cross-chain bridges will only become more critical.

II. Key design principles for seamless cross-chain communications

Today, there are nearly a hundred cross-chain bridge projects, each with various trade-offs in terms of security and trustlessness, upgradability and governance support, communications between blockchains with heterogeneous properties, and preserving the atomicity of transactions. The Teleport Network pioneers the Extensible Inter-Blockchain Communication Protocol (XIBC), which improves the Inter-Blockchain Communication (IBC) protocol[3], providing seamless and efficient

cross-chain communications between a wide variety of heterogeneous blockchain networks.

Below are some key design principles that we adopt when developing the Teleport Network.

1. *Efficient cross-chain integration*: To enable application developers to communicate with other chains easily, the network needs to interoperate with heterogeneous networks with minimal engineering effort and support a wide range of programming languages. Furthermore, the processing and storage costs for cross-chain synchronization, and data packet routing and verification should be minimized.

2. *Wide interoperability*: To support the widest range of assets and applications, the network needs to interoperate with chains of varying properties, whether they are IBC or multi-party computation chains, and fast or probabilistic finality chains.

3. *Versatility*: In a similar vein, supporting a wide range of applications requires the network to be adaptable to many different functions and activities, such as arbitrary cross-chain data communications and atomic cross-chain contract invocations.

4. *Governance and upgradability*: The protocol should be able to support cross-chain governance, including smart contract upgrades and change of witness addresses. Furthermore, the protocol needs to recognize updates in the underlying blockchains so that minimal efforts are required to support them.

5. *Security and reliability*: The network must be designed with security and reliability in mind, as a cross-chain interoperability protocol based on cryptography.

Existing protocols in the cross-chain interaction space can rarely satisfy all of these properties. For instance, the parachains of Polkadot rely on crossing a common relay chain which incurs additional costs, AnySwap does not support cross-chain contract calls, and Cosmos cannot provide direct communication between instant finality chains and probabilistic finality chains. The Teleport Network will design a protocol suite that satisfies these properties.

III. The Teleport Network

The Teleport Network provides the infrastructure and framework for frictionless cross-chain communications between disparate blockchain networks. Our vision is to enable developers to build apps that can access global liquidity and communicate with the entire blockchain ecosystem, with minimal integration efforts or the need to develop their own proprietary cross-chain bridges.

A two-tier architecture

The architecture of the Teleport network consists of two layers – the infrastructure and application layers. At the foundation is the infrastructure layer, which follows a delegated proof-of-stake (DPoS) blockchain model similar to Cosmos Hub. In addition, it deploys the XIBC cross-chain protocol (chain module and smart contract) and developer SDKs for application builders to implement cross-chain dApps integration.

Next is the application layer where developers actually deploy any variety of cross-chain applications. Some common examples include:

- **Cross-chain bridge** where users can transfer assets from one chain to another and communicate the change of states.

- **Cross-chain decentralized exchange** (DEX) where users can swap tokens and provide liquidity to another chain in a single transaction.

- **Cross-chain lending protocol** where users can deposit tokens on one chain to borrow tokens from another one.

- **Cross-chain NFTs market** where users can buy NFTs on one chain using tokens from another one.

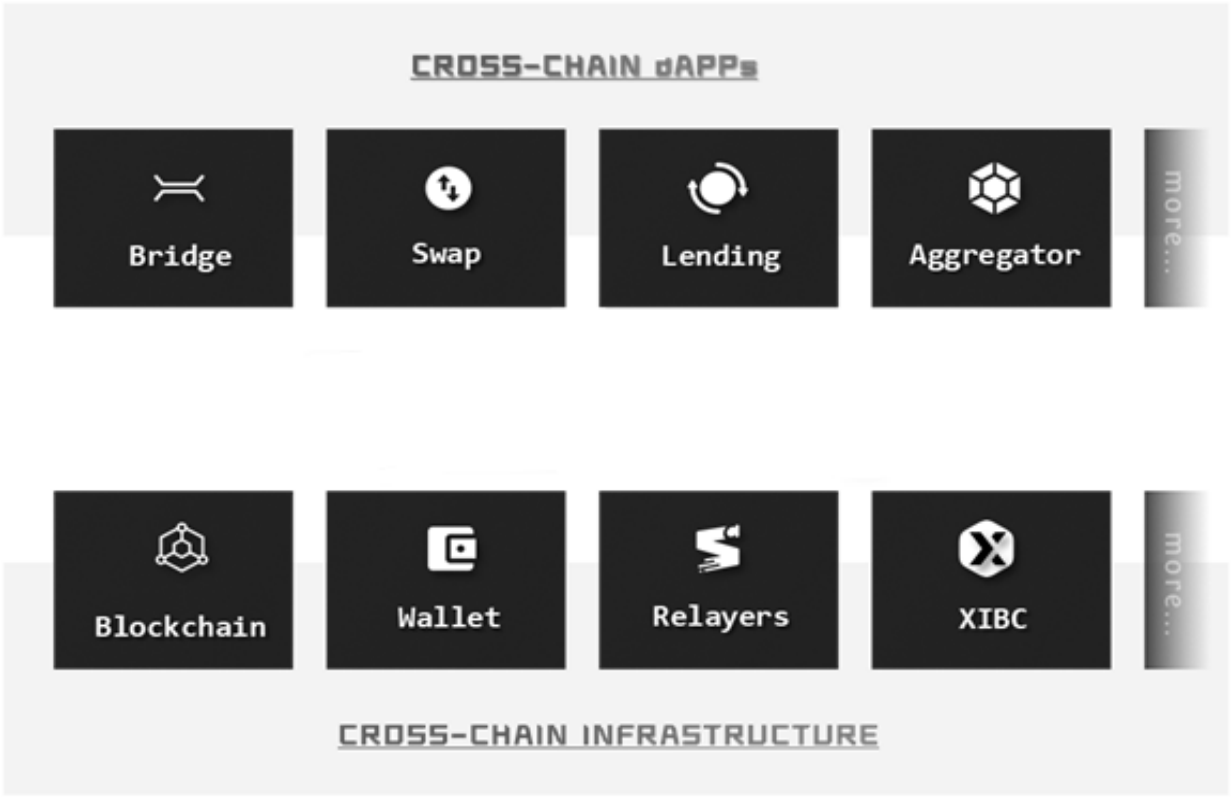

Figure 1: Teleport 2-Tier Architecture

Teleport Chain

Teleport Chain is based on the Ethermint SDK, a scalable and interoperable Ethereum library, built on Delegated Proof-of-Stake with instant-finality using the Cosmos SDK, which runs on top of Tendermint Core consensus engine[4]. The initial setup of the Teleport chain is 11 validators, an average block time of 3s, and TPS of 300+. Potentially, the validator set can be extended to 100 and TPS can be 1000+.

Ethermint allows for running vanilla Ethereum as a Cosmos application-specific blockchain. This allows developers to have all the desired features of Ethereum while at the same time benefiting from Tendermint's PoS implementation. Also, because it is built on top of the Cosmos SDK, it will be able to exchange value with the rest of the Cosmos ecosystem through the IBC protocol.

Therefore, dApps implemented in Solidity can be migrated to Teleport Chain seamlessly, as well as the MetaMask users. The feasibility of dApp deployment makes Teleport Chain not only a relay chain but also a multi-chain hub.

Additionally, Teleport Chain can also be easily integrated with both Ethereum layer2 solutions (like Arbitrum) and Cosmos scaling solutions (like Celestia).

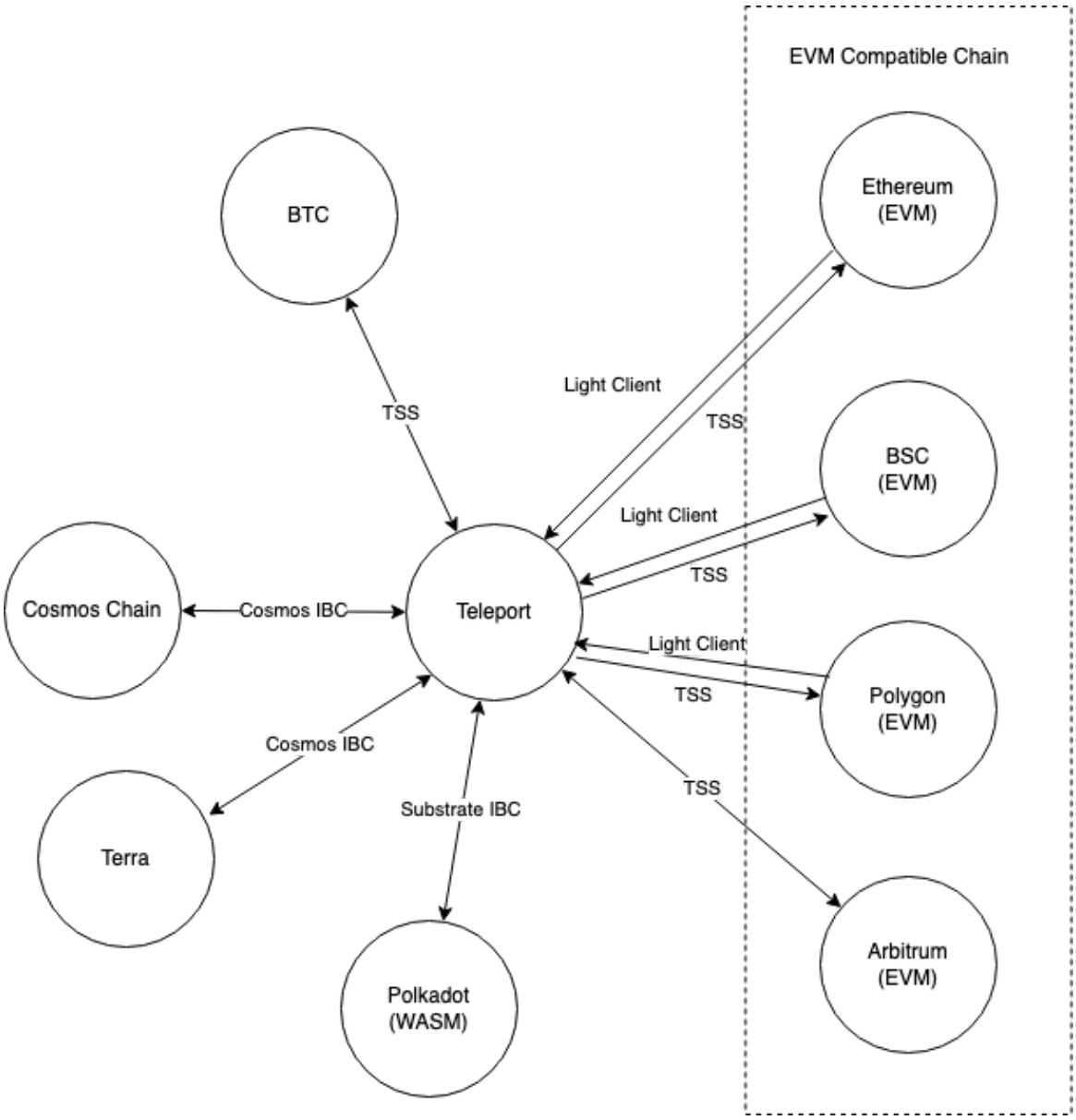

Figure 2: Teleport Chain as a Multi-chain Hub

IV. XIBC components

The core of the Teleport Network is a cross-chain protocol that provides trustless communications, verifications and deliveries. In this section, we will introduce the XIBC protocol and its four modules, namely **client**, **packet**, **routing** and **basic applications**. Under each module, there is a series of sub-components to support communication and messaging flows.

IV.a The XIBC protocol

Developed using the Cosmos SDK, the XIBC protocol is an extension of the IBC protocol. It is an interoperability protocol for communicating arbitrary data between arbitrary state machines. It can be used to build a wide range of cross-chain applications, including but not limited to token transfers, non-fungible token transfers and contract invocations. The XIBC protocol allows two chains to communicate via a third chain (e.g. Teleport Chain) which they both trust as a relay chain.

The XIBC protocol also incorporates composable state verifications including multi-party threshold signature and ZK-proofs. Also, the XIBC protocol is compatible with the Ethereum Virtual Machine (EVM), which means that Ethereum developers can port their smart contracts and dApps over to Teleport with ease, and use Ethereum's signature schemes, Solidity language and APIs as before.

XIBC makes a few improvements over IBC. First, the participating chains only need to be connected to the relay chain. This enables better support for non-BFT consensus machines such as Bitcoin and Ethereum. Furthermore, XIBC supports cross-chain interoperability for contracts, which can be achieved across multiple chains in only one packet, with all data packets automatically cleaned up at the end of their life cycle.

In summary, the XIBC protocol achieves the key design principles outlined in the previous section. It is a secure and reliable cross-chain interoperability protocol which enables easy and seamless integration. With an efficient and verifiable relay chain, it supports arbitrary cross-chain data communications and atomic cross-chain contract invocations between both homogeneous and heterogeneous chains. It also supports composable state verifications and applications.

IV.b Modules

The XIBC protocol is split into four modules: client, packet, routing and basic applications. The **client** stores the trust root which verifies the packet (or acknowledgement) commitment proof in order to guarantee the reputation and trustworthiness of the transacting address/party. The **packet** is the entry point for sending, receiving and storing cross-chain packets (and acknowledgements). The **routing** module routes each sub-packet (or sub-acknowledgement) to each basic application to execute their own logic. **Basic applications** support basic functionalities which developers can leverage on to develop applications on top of the XIBC protocol.

IV.c Clients

XIBC clients are verification machines that are identified by a unique client ID. XIBC clients track the consensus states of other blockchains and the proof specs of those blockchains that are required to properly verify proofs against the client's consensus state.

The XIBC protocol defines the following basic types of client implementation for alternative relay approaches:

- **Light client for XIBC light client relayer**: The light client is the default XIBC relayer, which is trustless and permissionless, and cross-chain packets are verified on the destination chain.

- **Threshold Signature Scheme (TSS) client for XIBC TSS relayer**: The TSS client is used where the light client is not feasible. Examples include Bitcoin which is not programmable, and Ethereum which is too expensive to verify trust roots and state proofs. Packets are verified by TSS nodes instead of the destination chain. In practice, a subset of Teleport Chain validators are selected as TSS nodes so that the XIBC clients' security level is close to the Teleport Chain.

- **ZK client**: The ZK client supports zero-knowledge proof verifications.

The above clients can be combined to verify blockchains of any properties. For the XIBC protocol, blockchains that will be supported initially include Ethereum, Cosmos, Arbitrum, Avalanche, Solona, Polygon and Binance Smart Chain.

IV.d Packet

Proofs and paths

In XIBC, blockchains do not directly pass messages to each other over the network. To communicate, a blockchain commits some states to a precisely defined path reserved for a specific message type and a specific counterparty. For example, a blockchain stores a specific endpoint as part of a packet intended to be relayed to a contract on the counterparty chain.

A relayer process monitors updates to these paths and relays messages by submitting the data stored under the path along with a proof of that data to the counterparty chain.

Packet

Contracts communicate with each other by sending packets over XIBC ports. All XIBC packets contain the following elements:

- A **sequence** to optionally enforce packet ordering
- **Source clientID**
- **Destination clientID**
- **Relay clientID**. If the relay chain is specified, it means the relay chain mode is used.
- A set of **sub-packets**. Each packet contains a **PortID** which allows the contracts to know the receiver contract of a given packet, and data.

Contracts send custom application data to each other in the `Data []byte` field of the XIBC sub-packet. Sub-packet data is completely opaque to XIBC handlers. The sender contract must encode their application-specific packet information into the `Data` field of the packets. The receiver contract must decode that data back to the original application data.

Receipts

XIBC writes a packet receipt for each sequence it has received. This receipt contains no information and is simply a marker intended to signify that the destination chain has received a packet at the specified sequence. This solves the double spend problem.

Acknowledgements

Contracts also write application-specific acknowledgements when processing a packet. Acknowledgements can be done in two ways:

- Synchronously on `OnRecvPacket` if the contract processes packets as soon as they are received from XIBC contract.
- Asynchronously if the contract processes packets at some later point after receiving the packet.

This acknowledgement data is opaque to XIBC much like the packet data and is treated by XIBC as a simple byte string `[]byte`. The receiver contracts must encode their acknowledgement so that the sender contact can decode it correctly.

The acknowledgement can encode whether the packet processing succeeded or failed, along with additional information that allows the sender contract to take appropriate action.

After the acknowledgement has been written by the receiving chain, a relayer relays the acknowledgement back to the original sender contract which then executes application-specific acknowledgement logic using the contents of the acknowledgement. This acknowledgement can involve rolling back packet-send changes in the case of a failed acknowledgement (refunding senders).

After an acknowledgement is received successfully by the original sender of the chain, the XIBC contract deletes the corresponding packet commitment as it is no longer needed.

IV.e Routing

Ports

An XIBC contract can bind to any number of ports. Each port must be identified by a unique portID. Since XIBC is designed to be secure with mutually-distrusted modules that operate on the same ledger, binding a port returns the dynamic object capability. When taking action on a particular port, for example, to send a packet with its portID, a module must provide the dynamic object capability to the XIBC handler.

XIBC contracts are responsible for claiming the capability that is returned on `BindPort`.

IV.f Basic application

Basic applications are the cornerstone of application development in the XIBC protocol. They include the basic functionalities of token-transfer, remote-contract-call and multi-contract-call, which developers can use directly for cross-chain transactions, or develop their cross-chain dApps based on them.

- **Token-transfer contract** allows users or other contracts to invoke and initiate a cross-chain token-transfer packet. It also provides a shared cross-chain funding pool for all applications between any two chains.

- **Remote-contract-call contract** allows users or other contracts to invoke and initiate a cross-chain contract-call packet.

- **Multi-contract-call contract** allows users or other contracts to call and initiate cross-chain packets, including cross-chain contract-call sub-packets with/without multiple cross-chain token transfers.

V. End-to-end flow

The XIBC protocol enables cross-chain interoperability between two chains either directly or via a relay chain. Below, we will describe how the protocol works and the various components that are activated in both scenarios.

Direct Cross-chain interoperability between two chains

In this scenario, two chains interoperate with each other via cross-chain packets generated by XIBC application contracts.

Step 1: A user sends a cross-chain transaction from the source chain via XIBC application contracts.

Step 2: Source chain XIBC application contracts generate cross-chain packets.

Step 3: Relayer syncs the trust root, cross-chain packet and commitment proof from the source chain to the destination chain. In most circumstances, the trust root is the merkle root of the state tree, and the proof of commitment is the merkle proof of the commitment in the state tree.

Step 4: Destination chain XIBC contracts receive the cross-chain packet and verify the proof. Subsequently the cross-chain packet is executed to generate an acknowledgement.

Step 5: Relayer syncs the trust root, acknowledgement and acknowledgement commitment proof from the destination chain to source chain.

Step 6: Source chain XIBC contracts receive the acknowledgement and verify the proof, and then execute the acknowledgement. For instance, a user will have his funds returned if the cross-chain transfer fails.

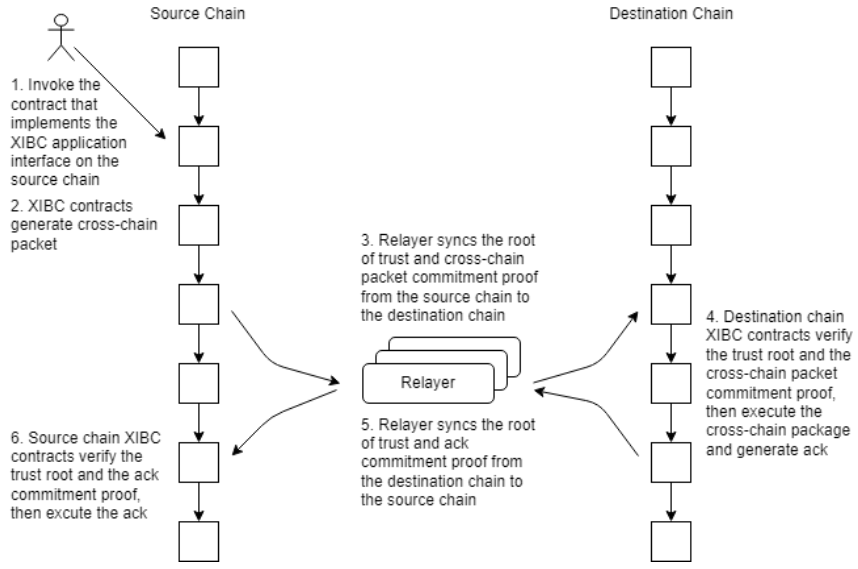

Figure 3: The communication flow in a cross-chain transaction between two chains.

Cross-chain interoperability between two chains via relay chain

In this scenario, the Teleport chain acts as a relay chain without any packet processing for business logic.

Step 1: A user sends the cross-chain transaction with the specified relay chain from the source chain via the XIBC application contracts.

Step 2: Source chain XIBC application contracts generate cross-chain packets.

Step 3: Relayer syncs the trust root, cross-chain packet and commitment proof from the source chain to relay chain.

Step 4: Relay chain XIBC application contracts receive the cross-chain data packet and verify the proof, and then store the data packet without any processing.

Step 5: Relayer syncs the trust root, cross-chain packet and commitment proof from the relay chain to the destination chain.

Step 6: On the destination chain, XIBC application contracts receive the cross-chain packet and verify the proof, then execute the cross-chain packet and generate an acknowledgement.

Step 7: Relayer syncs the trust root, the acknowledgement, and the acknowledgement commitment proof from the destination chain to relay chain.

Step 8: Relay chain XIBC application contracts receive the cross-chain acknowledgement packet and verify the proof, then store the acknowledgement without any processing.

Step 9: Relayer syncs the trust root, acknowledgement and acknowledgement commitment proof from the relay chain to source chain.

Step 10: Source chain XIBC application contracts receive the acknowledgement and verify the proof, and then execute the acknowledgement. For instance, a user will have his funds returned if the cross-chain transfer fails.

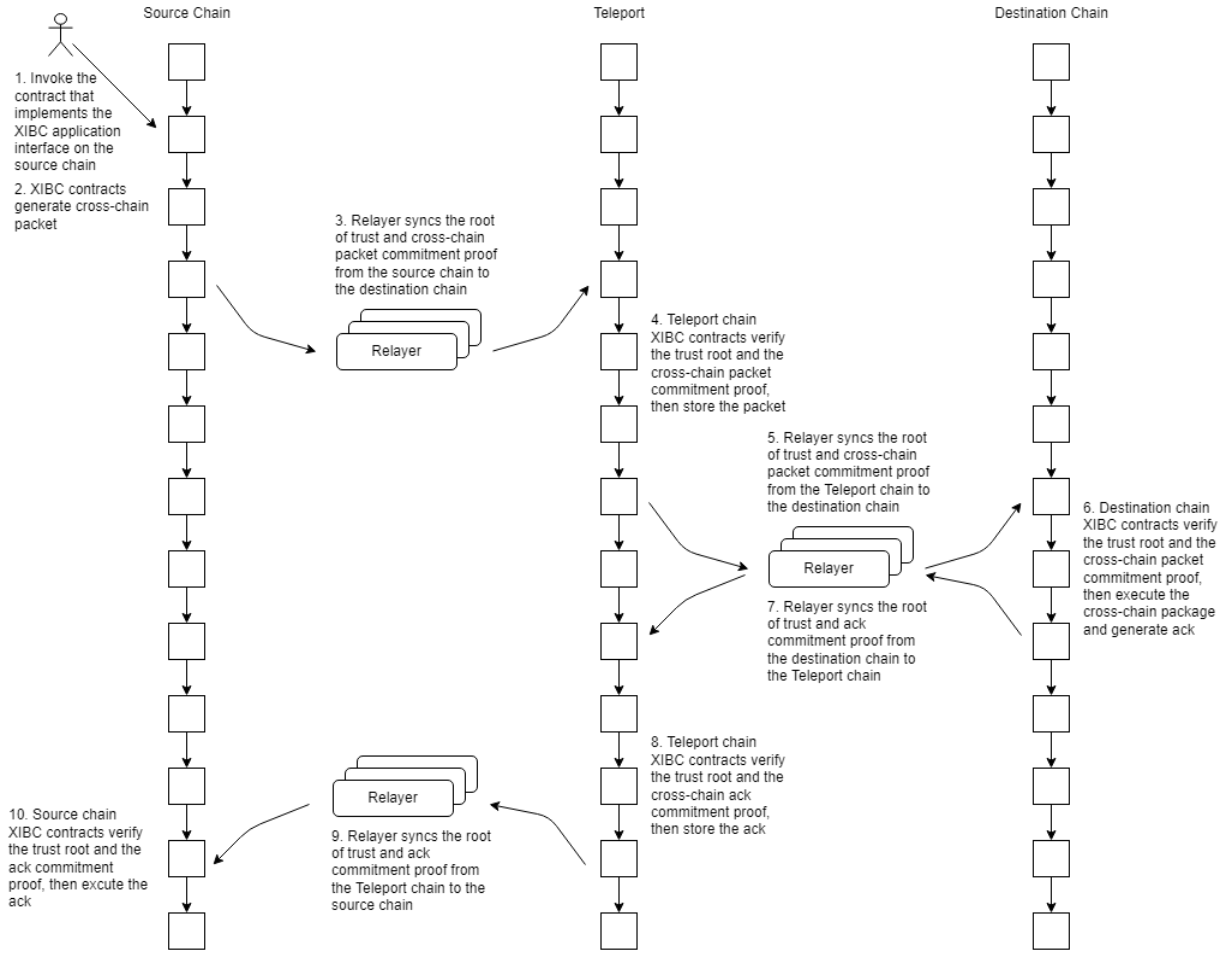

Figure 4: The communication flow in a cross-chain transaction between two chains via a relay chain.

VI. Developing cross-chain dApps

Cross-chain ecosystem

The vision of the Teleport Network is to create a cross-chain ecosystem where developers and users can interact seamlessly with each other regardless of where their assets lie. For blockchain to become a new open economic system serving the global market across all sectors, it is clear that today's isolated blockchains and layer 2 scaling solutions need to be bridged and interconnected.

To support this vision, the Teleport Network will serve as a single and unified access point providing hyper liquidity and composability, accelerating the pace of innovation for new primitives that are interoperable and universal. As the world of Web3, metaverse and DeFi scale across a greater global population, it is important to have frictionless user experience and highly composable technologies that can serve dApps in a wide range of use cases and environments[5].

While striving for this vision, we believe that the technology infrastructure for tomorrow's economy must be rooted in simplicity and easy to use interfaces for end users. Furthermore, developers must also be able to build products with exceptional user experience quickly and confidently. Our purpose is to ensure that innovators get the most out of our whole host of basic applications in the Teleport Network.

To accomplish our mission, the Teleport Network empowers application developers to build dApps easily using the universal XIBC protocol and API to communicate with the entire blockchain ecosystem. By using any combination of XIBC's basic applications described earlier (token-transfer, remote-contract-call and multi-contract-call), developers can build a variety of cross-chain interoperable contracts. Existing dApps can be integrated with the XIBC protocol seamlessly and with minimal efforts for cross-chain interoperability. For example, Uniswap can become a cross-chain DEX by simply adding the cross-chain swap functionality powered by XIBC.

Below we will illustrate step-by-step how to develop an application using XIBC.

Remote contract-call with token transfer

Step 1: A user invokes the cross-chain method of the dApp proxy.

The cross-chain dApp proxy is deployed on the source chain by the developers for their dApp on the destination chain. The user now interacts with the proxy to initiate the cross-chain transaction.

Step 2: The dApp proxy invokes the XIBC multi-call contract.

Step 3: The XIBC multi-call contract invokes the XIBC token-transfer remote-contract-call contracts to generate sub-packets.

Step 4: XIBC packet initiates cross-chain packet.

XIBC multi-call contract assembles all sub-packets into the cross-chain packet data and invokes the packet contract `[sendMultiPacket](https://github.com/teleport-network/xibc-contracts/blob/main/evm/contracts/core/packet/Packet.sol#L152)` *function. Packet contract will verify the validity of the packet and store it.*

Step 5: Relayers relay packet to the destination chain.

Relayers relay the trust root, the packet and the packet proof to the destination chain.

Step 6: The packet is executed on the destination chain.

On the destination chain, XIBC contracts receive the cross-chain packets and verifies the proof, and then routes sub-packets to the XIBC token-transfer and remote-contract-call contracts. Basic application contracts execute their own logic and return sub-acknowledgements to the XIBC packet contract. Then the packet contract aggregates all sub-acknowledgements and stores an acknowledgement corresponding to the packet.

Step 7: Relayers relay the acknowledgement back to the source chain and the acknowledgement will be executed by the XIBC token-transfer contract and remote-contract-call contracts.

The source chain XIBC packet contract receives the acknowledgement and verifies the proof, and then routes each sub-acknowledgement to XIBC application contracts. The XIBC token-transfer contract will return funds to users when encountering an acknowledgement error. The XIBC remote-contract-call contract will store the acknowledgement and users can execute their own exception handling logic by the acknowledgement.

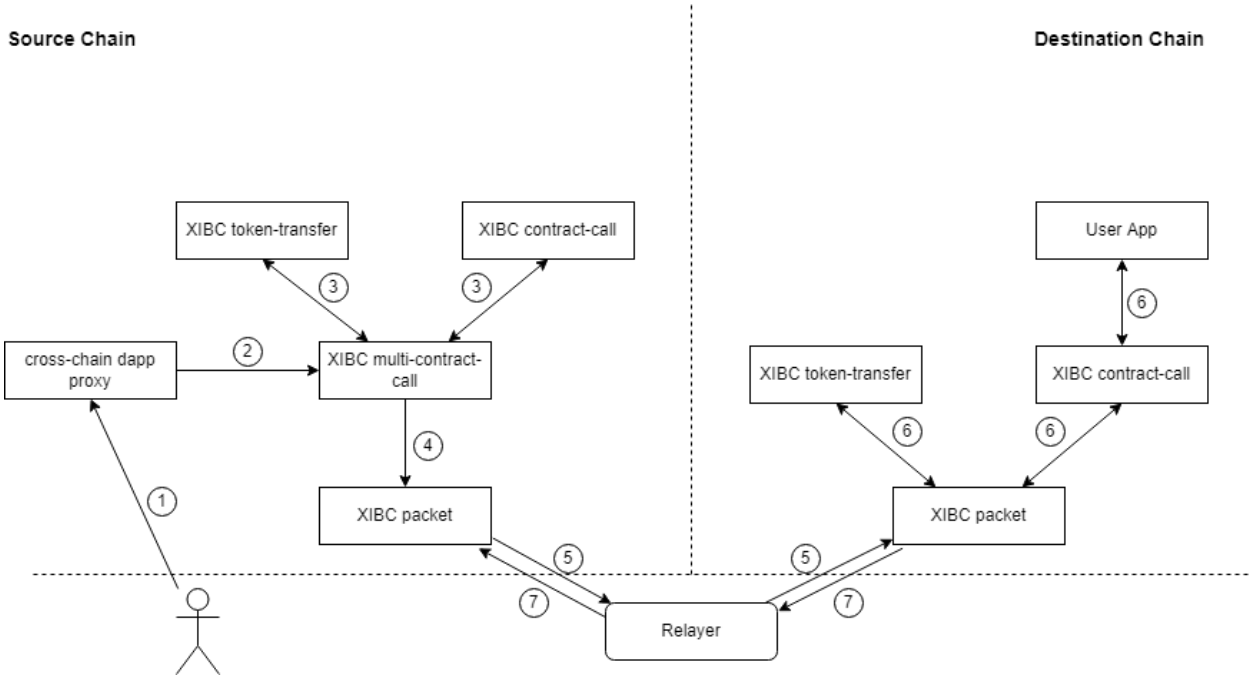

Figure 5: The communication flow of a remote contract-call with token transfer.

Cross-chain transfer via Teleport funding pool

In the above example, a token-transfer contract not only generates a cross-chain token transfer sub-packet, but also provides a cross-chain funding pool for cross-chain liquidity.

Teleport funding pool can be utilized via a cross-chain multi-call transaction.

Step 1: A user initiates a transaction on the cross-chain dApp proxy.

> *A user initiates a transaction to the cross-chain transaction agent via Teleport basic application contracts.*

Step 2: The dApp proxy invokes the XIBC multi-call contract.

Step 3: The XIBC multi-call contract invokes the XIBC token-transfer and remote-contract-call contracts to generate sub-packets.

Step 4: XIBC packet contract initiates a cross-chain packet.

Step 5: Relayers relay the packet to the Teleport chain.

Step 6: The packet is executed on the Teleport chain

Step 7: The cross-chain transaction agent contract invokes the XIBC token-transfer contract and initiates the next cross-chain packet.

> *The token-transfer receiver on the Teleport chain is the remote-contract-call contract which carries the transferred token while invoking the cross-chain transfer agent contract on the Teleport chain.*

Step 8: Relayers relay the packet to the destination chain.
Step 9: The packet is executed on the destination chain.

> *The transferred token can be utilized for the dApp contract call, such as for token swaps or staking.*

Step 10: Relayers relay the acknowledgement back to the Teleport chain.

Step 11: Relayers relay the acknowledgement back to the source chain and the acknowledgement will be executed by the XIBC token-transfer and remote-contract-call contracts.

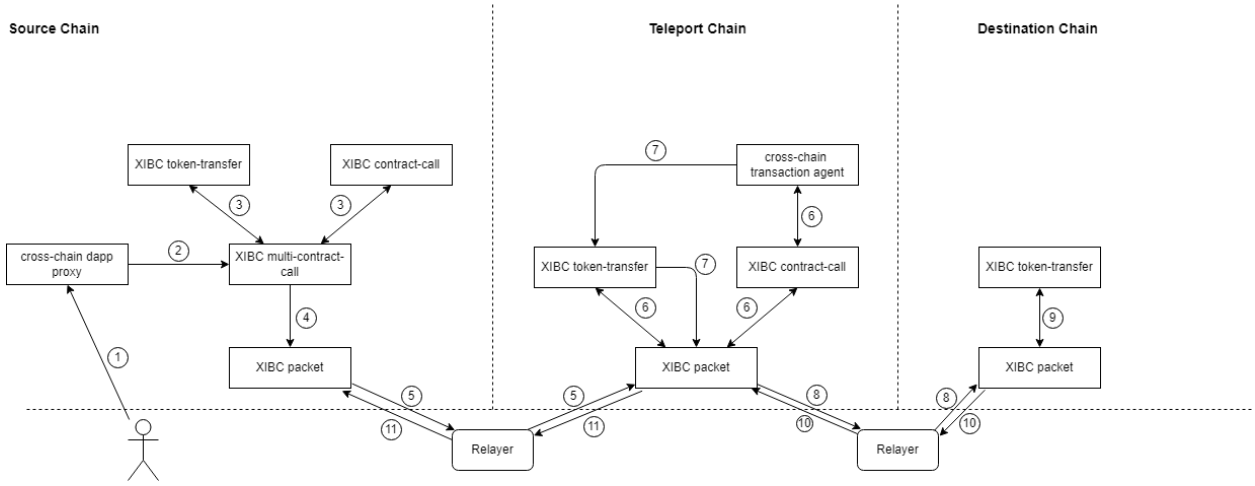

Figure 6: The communication flow for cross-chain transfer via Teleport funding pool.

VII. Conclusion

Teleport is on a mission to expand the possibilities of the blockchain ecosystem by providing infrastructural and framework support for application developers. Through improved cross-chain communications powered by the XIBC protocol, we are approaching a world where boundless blockchain interaction exists across DeFi, NFTs, GameFi and other blockchain use-cases.

Over the next few years, a diverse set of applications and assets will be built and deployed across multiple blockchain ecosystems. We seek to power an incoming era of cutting-edge blockchain solutions, where developers and users alike can seamlessly interact across several ecosystems. This paper expresses our design philosophy, approach, and detailed overview to attaining efficient cross-chain integrations in a secure and reliable manner. Our XIBC protocol, which improves upon Cosmos' IBC protocol, providing seamless cross-chain communications across heterogeneous blockchains. The Teleport Network provides the clients, packet, routing and basic applications that developers can use to build their dApps easily and communicate with blockchains in the entire ecosystem.

Furthermore, Teleport aims to be at the forefront of ushering in a Web 3 enabled future. With the support for a number of blockchains already (including Bitcoin & Ethereum), we hope to expand further into other blockchain ecosystems in the near future; thus, facilitating all blockchain transfers, interactions and communication.

Teleport's tech stack will be the go-to infrastructural solution for blockchain developers, driving innovation to newer heights and further accelerating the global adoption of blockchain technology. We envision a future of completely interoperable blockchain ecosystems powered by the Teleport network.

VIII. Reference

[1] Multi-chain future likely as Ethereum's DeFi dominance declines, Bloomberg Intelligence, February 2022
[2] 2022 Digital Asset Outlook, The Block Research, December 2021
[3] IBC Overview, Cosmos SDK
[4] High-level Overview of Ethermint, Ethermint website
[5] Closer look at the Open Metaverse OS, Outlier Ventures, March 2021