PolkaDot↔FreeTON Bridge Design and Architecture

Abstract

There are a lot of public blockchain networks with their own audience, market and products working in them. Each network has its own advantages and disadvantages that make its use effective in a particular area, but at the same time limits the rest.

Bitcoin was created as a guarantor of secure transfer of value between people in different parts of the world, but it is limited in its bandwidth.

Ethereum was created as a platform for running decentralized applications, but inherited both the problems of Bitcoin and got new scalability problems due to the need to store more information. Both platforms still have a high entry threshold and the majority of users are enthusiasts and “cryptogeeks”. Nevertheless, there is already a strong belief that these networks will not leave the market, they have taken their strong position and have a large user base.

Ethereum and Bitcoin, for all their publicity, remain closed and incompatible systems, which prevents users of one network using products from another network and safely exchanging values between them.

The new generation of blockchains seeks to avoid such problems by design, and at the same time to solve how to join the old networks to achieve interoperability. We believe that the most promising networks at the moment in this area are Polkadot and FreeTON. Polkadot is heterogeneous by design, which means the property of combining things that are initially incompatible with each other. FreeTON will also occupy its own niche and allow users to use crypto payments and services en masse as if they were using Visa or Mastercard.

We would like to connect these two promising networks through internetwork interaction and give users the benefits of both projects.

Summary

A proposal for architecture and design of Bridge Polkadot ↔ FreeTON

Status

The document is a draft and may be changed in the future

Glossary

Polkadot Parachain - A blockchain that meets several characteristics that allow it work within the confines of the Polkadot Host. Also known as “parallelized chain.”

Relay Network - A network that relays messages from one blockchain network to another.

Relay Node – A node that implements the protocol and works in the Relay Network.

Relayer - A role issued to a member of the blockchain network gives them the right to send messages to the network as a Relay Node.

VRF (verifiable random function) - It is a pseudo-random function that provides publicly verifiable proofs of its outputs’ correctness.

Distribution of relayer responsibilities - A mechanism that determines if a Relayer can participate in the current vote.

Introduction

The Protocol implies three parties implementing the bridge: Polkadot Parachain, FreeTON Smart-contracts, and Relay Network.

Developing the protocol, we were guided by the following principles:

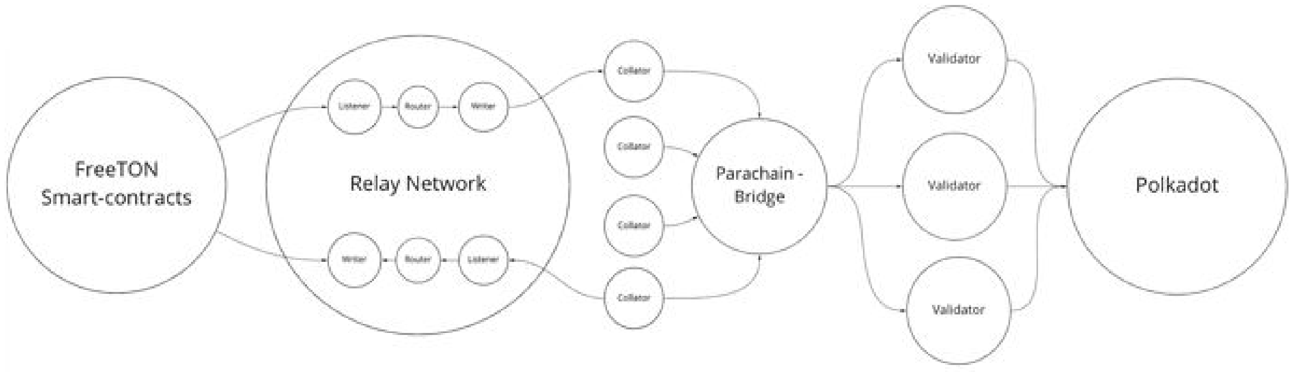
- **trustless** - the protocol should not rely on the trustworthiness of participants and be resistant to attacks.
- **non-custodial** - the protocol should not have access to the funds transferred with its application.
- **censorship resistance** - the protocol should reward participants who perform the role of Relayer for transmitting messages and punish those participants who don’t perform their role as required by the protocol.
- **governance** - the protocol should have mechanisms for managing through the community.
- **generalized** - the protocol should be sufficiently generalized to cover as many use cases as possible.

In order to provide the transmission of messages between two parties (Polkadot Parachain and FreeTON Smart-contracts), we have chosen the most proven and effective mechanism for today with the addition of an intermediary between them in the form of a Relay Network. This approach is used in many projects, including *Polkadot*, where the main connecting network is the Relay Network.

To ensure stability to censorship, we have added an economic model to the protocol, which provides a mechanism for slashing for nonfulfillment of the duties of the Relayer role. As well as a reward mechanism for maintaining the operation of the node in the Relay Network.

The protocol is “trustless” due to the voting mechanism among Relayers, no node or group of nodes has the ability to commit illegal actions, write off users’ funds, even by collusion. To provide the decentralization of the Relay Network, we introduce the “distribution of relayer responsibilities” mechanism via VRF (verifiable random function). This adds an element of randomness to the selection of those who vote for the next proposal, what makes it necessary to collude absolutely all network participants to carry out the attack. However, management is in the hands of the community. If the community sees that one or more Relayers are systematically trying to attack the network, it’s possible to hold a referendum for removing the node from the Relayer role, which will not allow the node to participate in the vote.

Technical details

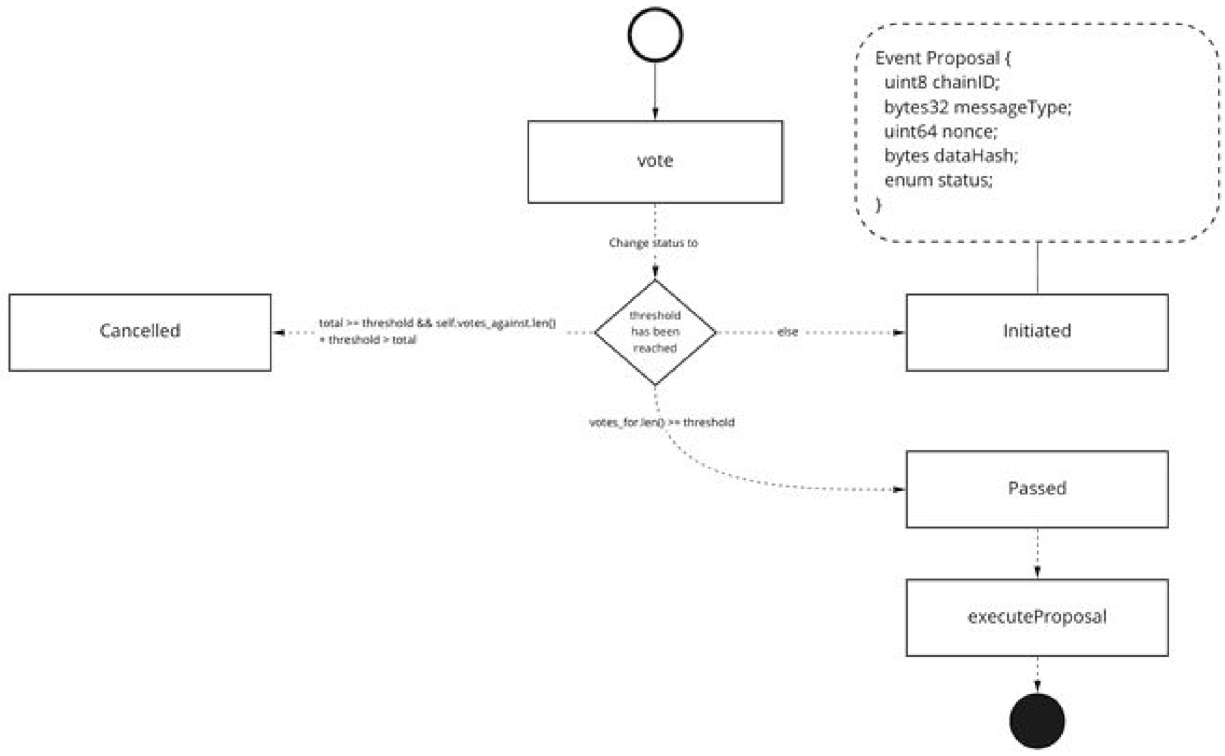


Relay Network

Relay Network listens to events in both networks and transmits messages between them by means of proposals and votes of Relay Network participants. Relay Network participants are “authority” nodes chosen by the community through voting mechanisms. Relayers don’t gossip among themselves.

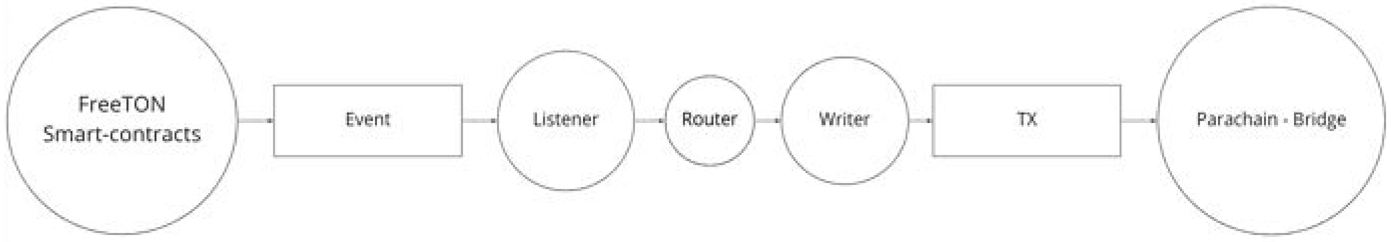
The Economic incentive for participants is described in the “Economic model” section.

Relayer launches a node that monitors the Polkadot and FreeTON networks for messages that need to be transmitted. If they are detected, signs and sends a transaction with a proposal to make changes to the target network. In order for a proposal to be accepted, it must receive enough votes from the Relay Network to pass the threshold and be executed. Thus, the protocol is protected from false records of information about transferred data and values and is also resistant to censorship. If one or more nodes agree not to transmit a message via the Relay Network, other participants who are not part of the agreement will still send the message to the network and vote for its acceptance.


```
Event Proposal {
  uint8 chainID;
  bytes32 messageType;
  uint64 nonce;
  bytes dataHash;
  enum status;
}
```
vote
Change status to
threshold has been reached
total >= threshold && self.votes_against.len() + threshold > total
Cancelled
else
Initiated
votes_for.len() >= threshold
Passed
executeProposal

The Relayer node consists of three components:

- **Listener**. Listens for events on the network that it is subscribed to and, if they occur, passes it to the "router".
- **Router**. Receives an event from the Listener and interprets it, determining where to pass it next.
- **Writer**. Accepts the message and information about its purpose, forms, signs, and sends the transaction with the offer to the target network.



Thus, when the Relayer receives an event about a new message that needs to be transmitted, it automatically votes for it. The protocol checks if there is already an offer for this message in the target network, and if there is no such offer, it is created. If a proposal exists, the Relayer just leaves its vote in that proposal.

The node should be able to listen to events, generate, sign, and send transactions, store Relayer member keys, and communicate via RPC calls for both networks.

Also, the Relay Node should have a supply of tokens from both networks to pay for transaction costs, and they will be refunded as a reward for correct operation.

Polkadot

As mentioned above, Polkadot has mechanisms designed to join networks that are initially incompatible. One of these mechanisms is “Bridge modules”.

Bridge modules

Receiving messages on Polkadot from an external, non-parachain blockchain can be built as a parachain module. The parachain module can then be deployed to Polkadot either as a system-level parachain (native extension to the core Polkadot software) or as a community-operated parachain

Bridge modules allow for non-parachains to act as a “virtual parachain” and extends the external chain’s functionality with the interoperability benefits of Polkadot.

Bridge contracts

In the case of Polkadot, it should be possible to have a bridge contract deployed on, say, an EVM-based standalone chain and a contract deployed on a smart contract capable parachain. This would not necessarily be the most efficient method of bridging, but given the generality of a Turing-complete parachain it would be possible to bridge Polkadot and any other smart contract capable blockchain.

No matter how the “Bridge” component is implemented on the Polkadot side, it should accept and store messages for transmission to the FreeTON network, as well as suggestions for adding messages to Polkadot from the FreeTON network.

For this purpose, it implements the protocol described below.

To implement a “bridge” on the Polkadot side, it is proposed to use the bridge modules approach. It is necessary to write the pallet (a substrate runtime module) in Rust that implements the protocol described below.

How to develop the pallet for substrate is described in this lesson.

To run a bridge for a specific case, it’s necessary to build a node based on the substrate framework that will include the developed pallet.

Further implementation depends on the bridge launch strategy and the specific application case and is beyond the scope of this work.

FreeTON Smart-contracts

A similar role in the FreeTON network is carried out by a set of smart contracts that implement the same protocol.

Accepts and stores messages for transmission to the Polkadot parachain network, as well as suggestions for adding messages to FreeTON from the Polkadot network.

On the FreeTON side, it is proposed to deploy a set of smart contracts that meet the described protocol. Below is an example of the structure of smart contracts for FreeTON, taking into account the specifics of TVM.

Voting

VoteMainContract - stores the code of the VoteWallet contract, deploys a specific ProposalContract to the network, and stores the proposal ids.

VoteWallet - a contract intended for voting in ProposalContract. Must store the address of the ProposalContract in which it was applied. The implementation of this contract is stored in VoteMainContract.

ProposalContract - provides details of the proposal, allows to vote using VoteWallet. Should be able to verify the VoteWallet code of the contract accessing it.

RBAC

RootRBAC - stores the code of the RBACWallet contract, is formally a bridge between the caller RBACWallet and the callee (for assigning roles.

RBACWallet - stores its own set of roles

Handler

The **Handler** contract implements the execution logic of proposal (for example, calling a non-fungible tokens contract) and stores a list of proposal ids executed through this handler.

Bridge

Facade for working with all parts of the protocol. Stores the addresses of other contracts.

Protocol specification

All bridge components implement a single Protocol described below.

Role-based Access Control

The protocol assumes that there is a Role-based Access Control implementation for determining access rights to the execution of protocol methods. The protocol has two roles: Admin and Relayer.

Admin has the right to set and delete the Relayer role for protocol participants. For the first time, the Admin role is assigned to the participant whose address will be passed during system deployment. (the Admin role can be assigned to a multisig account to provide decentralized management.)

The Admin role can only be transferred to another member by a member with the Admin role. The Admin role can only have one member at a time.

Specification

Reference AccessControl.sol

grantRole(bytes32 role, address account) public

Grants `role` to `account`.

---

revokeRole(bytes32 role, address account) public

Revokes `role` from `account`.

---

renounceRole(bytes32 role, address account) public

Revokes `role` from the calling account.

---

hasRole(bytes32 role, address account) public view returns (bool)

Returns `true` if `account` has been granted `role`.

---

getRoleMemberCount(bytes32 role) public view returns (uint256)

Returns the number of accounts that have `role`.

---

getRoleMember(bytes32 role, uint256 index) public view returns (address)

Returns one of the accounts that have `role`.

---

getRoleAdmin(bytes32 role) public view returns (bytes32)

Returns the admin role that controls `role`.

Proposals

The protocol implies the existence of a Voting system implementation for the ability to vote for Proposals.

The Proposal should contain the hash of the message, information about the original network where the message came from, and the status of the offer.

Statuses: Inactive, Active, Passed, Executed, Cancelled.

Only a member with the Relayer role can create a Proposal. Only a member with the Relayer role can vote for the Proposal.

Specification

getProposal(uint8 chainID, bytes32 messageType, uint64 nonce, bytes32 dataHash) public view returns (Proposal)

Returns a proposal.

---

vote(bool isFor, uint8 chainID, bytes32 messageType, uint64 nonce, bytes32 dataHash) public

If the proposal that the Relayer votes for is not created, the method creates it and votes for or against the proposal and carries out it if the vote threshold has been reached.

If the proposal exists, it votes for or against the proposal and carries out it if the threshold of votes has been reached.

Handler

After the Proposal status changes to Passed, the message is transmitted to the Handler and processed according to its logic. In each particular case, the Handler can have its own logic for processing the message.

Specification

---

getMessageRecord(uint8 chainID, uint64 nonce, bytes32 dataHash) public view returns (MessageRecord)

Returns a message record.

---

executeProposal(bytes calldata data) public

Proposal execution should be initiated when a proposal is finalized in the Bridge contract.

Bridge

Facade for working with all parts of the protocol, hides the details of their implementation and provides a simple interface for working with them. Stores the addresses of other contracts.

Specification

---

isRelayer(address relayer) public view returns (bool)

Returns true if {relayer} has the relayer role.

---

renounceAdmin(address newAdmin) public

Removes admin role from {msg.sender} and grants it to {newAdmin}.

---

adminAddRelayer(address relayerAddress) public

Grants {relayerAddress} the *Relayer* role.

---

adminRemoveRelayer(address relayerAddress) public

Removes relayer role for {relayerAddress}.

---

adminSetHandler(address handlerAddress, bytes32 messageType) public

Sets a new handler for message type

---

executeProposal(uint8 chainID, bytes32 messageType, uint64 nonce, bytes calldata data) public

Executes a proposal that is considered passed using a specified handler contract

---

Governance and tokens

This section is an addition to the Protocol that affects its economic model. For more information about all aspects of the economic model, see the section Economic model.

The protocol implements governance through its native Voting system and Proposals mechanisms.

Distribution of relayer responsibilities

To add the Relay Nodes distribution mechanism to the vote for proposals, it is proposed to change the method of the vote protocol by adding two arguments result and proof returned by the VRF function.

vote(bool isFor, uint8 chainID, bytes32 messageType, uint64 nonce, bytes32 dataHash, bytes32 result, bytes32 proof) public

Token DOTON [Rewards and Slashing]

The protocol has mechanisms for rewarding correct work and punishing incorrect work, which is equivalent to attempts to attack the network. For this purpose, a movable DOTON token is issued.

Specification

---

mint(uint256 tokens) public

Mints {tokens} a number of new tokens, in accordance with the TIP-3 specification.

---

burn(uint256 tokens) public

Burns {tokens} the number of tokens.

GasDOTON

A unit for calculating the commission for sending a message.

Specification

estimateGas(bytes data) public view returns (uint256)

Calculates the amount of gas required to send a message

---

estimateFee(uint256 gasAmount, uint256 gasPrice) public view returns (uint256)

Calculates the fee required to send a message.

Economic model

This section describes the economic model for Relay Network members. The economic model of the parachain that the Bridge can be used with depends on the specific use case.

General principles

Distribution of relayer responsibilities

In order to determine which of the Relay Nodes will participate in the voting, it is proposed to use the VRF (verifiable random function) mechanism.

Each Relay Node carries out a function (VRF) that accepts

- The “secret key”, a key specifically made for these die rolls.
- Hash of the transaction in which the message was sent.
- Data message hash.

The result of the calculation will be two values: RESULT (pseudorandom value) and PROOF (proof that the pseudorandom value was calculated correctly).

A RESULT that falls below the passing level defined in the implementation process becomes a viable voter candidate for this Proposal. The Relayer then forms a voting transaction and sends it to the network along with the PROOF and RESULT values.

In this mechanism, there is a chance that an insufficient number of Relayers will find a value below the passing level. Then, after the N number of blocks expires, any Relayer can vote for this Proposal and compete for a reward.

Rewards and slashing

The Relay Network requires an economic incentive to operate. As miners are rewarded for producing blocks, Relayers are rewarded for transmitting messages between networks. In turn, the sender of the message should pay a fee for its transmission.

The gasDOTON unit of account is used to pay fees. The required number of gasDOTONs is calculated depending on the length of the transmitted message. The price for gasDOTONs is determined by the sender himself (the mechanism is similar to gas in Ethereum), so messages in the network for which a large amount of reward is announced can be processed faster.

The formula for calculating the amount of gasDOTONs per message:

$$S = L^2 * G$$

Where L - is the length of the message, G - is the constant cost of the symbol in the message, and S - is the amount of gasDOTONs charged for the message.

The formula for calculating the fee in TON or DOT tokens looks like this:

$$F = S * P$$

Where S - is the amount of gasDOTONs charged for the message, P - is the price that the sender is willing to pay for each gasDOTON, and F - is the amount in TON or DOT that should be sent along with the message.

A DOTON token is ISSUED to optimize commission fee to Relayers. The commission paid in the original network is blocked in the contract of the original network until it is unblocked in exchange for the DOTON token from the target network. The DOTON token is unmovable. It can be burned to release the reward on the other side of the bridge.

For example, a member of the FreeTON network sent a message to the Polkadot network and paid a commission fee of 1 TON. The commission fee of 1 TON remains on the contract in the FreeTON network, and in return, a proportional share of DOTONs tokens is issued for each voting participant in the Polkadot network. If 10 nodes participated in the vote, each gets 0.1 DOTON, which they can later burn to release their share on the FreeTON side.

This mechanism optimizes operating expenses for commission payments. The commission is blocked in the contract on the side of the original network when the message is sent. After that, the Relayers who participated in the voting get a DOTON token in the target network. Since voting takes place in it, this doesn’t require an additional transaction. In addition, the owner of the Relay node can save DOTONs until a significant amount is accumulated and release it at once, thereby reducing the load on the network.

Also, to provide network security, Relayers block steak in both networks in return for which they also get a DOTON token that they can always burn and unlock their steak. Nodes that don’t have a stake can’t participate in voting.

The Relayer stake is used as a guarantor and the node may be slashing for incorrect behavior. The gradation of violations and the size of the fine should be worked out by the community, as well as the size of the Relayer steak.