Omnite: Multichain NFT token solution

Omnite foundation

13 March 2022

Abstract

The Paper introduces the first offering of a new multi-chain solution aimed to revolutionize the way users mint, sell, and purchase non-fungible tokens, fundamentally altering the current NFT industry. This is achieved by utilizing a revolutionary cross-chain technology - LayerZero[2] and Omnite protocol smart contracts around it.

Today's NFT token creators are forced to choose a specific blockchain platform on which to launch their collection, thus limiting the number of potential buyers, available marketplaces, and native tokens as payment options. Omnite removes these limitations by introducing the capabilities of multichain NFT token collections, without having to rely on a centralized custodian and current slow, expensive and centralized token bridging mechanisms.

This document outlines all of the disadvantages of adopting a single blockchain from the perspectives of creators, investors, and marketplaces, and how they may profit from a multi-chain solution. It also describes how the Platform's approach can reduce excessive gas fees and difficulties with the process of trading NFTs while also allowing for complete decentralization of the implementation procedure.

1 Introduction

Omnite is a smart contract-based protocol that allows NFT tokens to be moved between blockchain networks in a completely decentralized manner without the need for a centralized custodian. It dismantles the boundaries that exist between blockchains, allowing NFTs to easily move across networks with the emergence of revolutionary trustless omnichain interoperability LayerZero protocol.

The platform operates on all networks supported by Layer Zero protocol, with the first phase focusing on EVM-based blockchain networks such as ETH, POLYGON, AVAX, FANTOM, BSC. It is intended to enable networks based on various ecosystems in future phases: Cosmos, Terra Luna, Solana as a result of the LayerZero protocol's development.

The protocol supports tokens created using the OMNITE launchpad (hereafter referred to as "native tokens") as well as all other NFT tokens created in the erc721 standard since the inception of the blockchain network. The Omnite protocol addresses four major issues that affect platforms/tools that transfer NFT tokens between networks: centralization, a convoluted token transfer procedure, excessive transaction fees, and a long wait time. The Omnite's procedure of transferring Tokens between networks is based solely on smart contracts and it doesn't require any additional steps from the user's point of view other than when transferring NFT tokens within the same network.

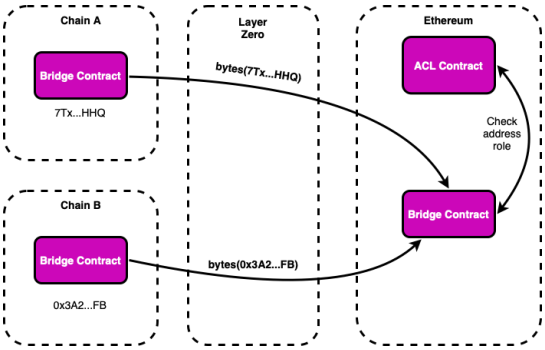
2 Protocol Design

The core of Omnite is an NFT transfer protocol that allows tokens to be moved between networks and guarantees that at any given point in time within a given collection, there is only one instance of the token on all supported networks. The protocol is a collection of decentralized smart contracts that are distributed across all of the networks that our solution supports. Their functioning and command are self-contained.

A centralized custodian has been replaced by a revolutionary communication protocol - Layer Zero, whose endpoints are used to transfer encrypted data between networks in a decentralized manner.

As the system is using multiple chains (in the future even different blockchain types) different issues had to be overcome which do not exist in single-chain applications.

a) **Multi wallet type support in access control**



One of the biggest challenges was to prove that a contract from another chain is allowed to call our OmniteBridge as the smart contract address of bridge on Solana or Terra Luna chain have a different address structure which is not correct (and supported) on EVM-based chains,

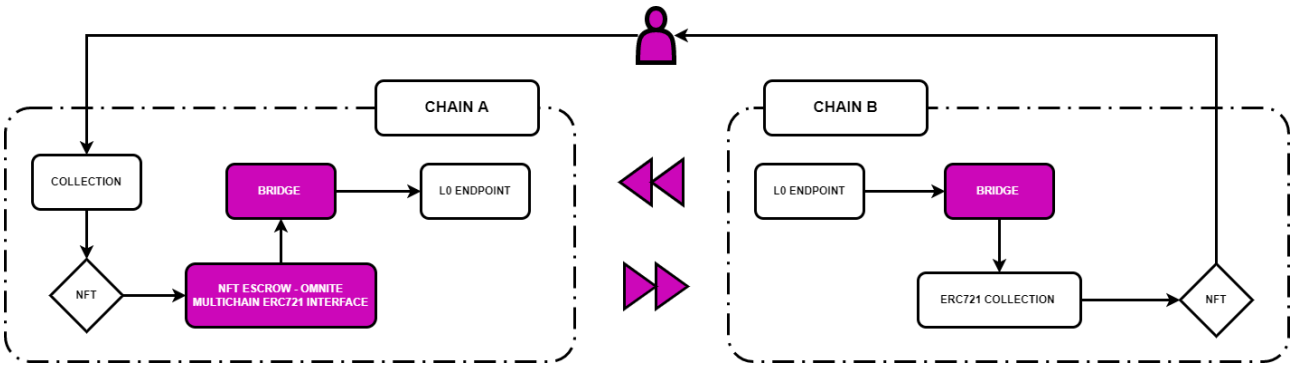


Figure 1: NFT token multichain transfer flow with OMNITE protocol

thus there was need to implement custom solution to grant roles for different types of addresses.

b) **Upgrading code on multiple chains at once (Hot swaps)**

All modern EVM applications use a Proxy pattern to be able to update smart contracts code in case of new features or security fixes. Omnite Hot Swaps introduces a revolutionary way to upgrade multiple instances of the same contract at once in a secure-multichain way. Hot Swap consists of 3 components - Blueprints which are representing various smart contract implementations (e.g. NFTs), UpgradeableBeacon which keeps track of the newest possible implementation for a particular contract type (e.g. the newest implementation for ERC721[1]), and BeaconProxy which is visible to the user as an NFT contract and is responsible for delegating calls to real implementation. Our approach of using a proxy pattern reduces gas consumption of deploying a new ERC721 NFT collection by more than 75% and gives Omnite users the ability to receive new features implemented by the team and community in the future with no effort.

c) **DDOS & bridge block protection**

A problem that is impossible to solve on-chain is a gas estimation on the different blockchain networks, as one blockchain network does not have access to the data of other networks. Layer Zero requires to pay for a particular amount of gas needed to process operation, thus one can never be sure that the gas amount provided by the frontend application is not malicious (or calculated wrongly) and will be sufficient to execute the transaction on the destination network correctly. Omnite sets a minimum gas that the bridge can accept and takes the approach that the transactions never revert with errors, producing error events instead. This was achieved by using sophisticated low-level calls on the very entry stage of Layer Zero callback in the bridge.

d) **ERC20 Token**

Omnite token is designed as a multi-chain token. The token itself also uses Layer Zero infrastructure to transfer value between different blockchains, allowing to implement decentralized multi-chain stakings and vestings.

3 Protocol Components

Omnite Bridges are the heart of multi-chain communication. They have direct access to LayerZero endpoints and work as a message layer between Omnite and LayerZero. They consist of Bridge Sender and Bridge Receiver. The contracts are responsible for encoding and decoding low-level calls, executing them on different networks, and receiving messages on the other side. They are trustless by design, for instance, any sender can deploy multiple contracts to different networks in just one transaction.

Contract Factory is a contract holding blueprints of the tokens. It is used to deploy new contract instances (e.g. Different ERC721 implementations) by utilizing a proxy pattern with upgradable contracts and beacon proxies. It supports versioning of the blueprints and deployment by contract type, which allows saving gas: LayerZeroBridge doesn't have to transfer the whole bytecode in order to bridge the contract, it only needs a blueprint name along with constructor params needed to instantiate a storage for the proxy contract.

Collection Registry holds the records of all Omnite's ERC721 native and non-native tokens. Each ERC721 deployed or bridged by the service is stored in this registry. Each record holds the owner, address, and name of the collection. CollectionRegistry provides the data required to validate incoming and outgoing bridge requests.

Access Control List (ACL) specifies which contracts are granted access to other contracts, as well as what operations are allowed on given contracts. It is based on OpenZeppelin's[4] AccessControl library, but extended to support non-EVM networks' address structures.

System Context keeps track of smart contracts in the Omnite system, allowing contracts to sub-

scribe for address changes using callbacks. For example, when a new version of the ContractFactory registry is set, the system context notifies all interested parties.

Native Tokens are contracts deployed directly from Omnite collection creator. Owner of the collection can easily mint them on the platform and bridge them afterwards. Each token's smart contract on particular networks has a minting range, which restricts minting to particular ids. Native tokens are deployed as upgradable contracts.

Non-native Tokens are contracts which were deployed outside of the Omnite platform, i.e. Invisible Friends, Bored Apes, etc. They have to implement ERC721Metadata interface. To bridge those tokens between networks, they need to be wrapped first. There is no possibility to mint them through Omnite.

4 Protocol Fees & staking

Omnite project due to its multi-platform nature is equipped with a payment mechanism using many native coins. Depending on which network the user calls the service (creates collections, mints tokens, or sends a token to another network), payment will be in the source network coin. Fees are accumulated on wallet contracts deployed on all supported networks. These contracts act as a treasury where the capital is accumulated and can be exchanged for other tokens via the Stargate platform. The fees charged represent both a profit for the project and are used to transfer and exchange money via the Stargate platform. Fees are to be exchanged for the OMNITE token when the threshold is passed. This operation can be initiated by any entity. Purchased OMTs go into the main staking contract: Omnite Staking. This contract uses the OMT token as both a collection token and a reward token. By blocking tokens on a contract, the user is able to multiply the number of tokens held. Tokens are paid out linearly to each staking participant at a contractually specified rate. The reward release rate is defined as a percentage amount of tokens over a set period of time: for example, 10% of the pool in a month.

5 Case Study: Bridge Non-native NFT

In this section, we briefly describe the details of how we implemented the basic feature of the Omnite platform, which is bridging existing collections and sending the user's token to the target blockchain. Most of today's existing collections rely on a set of Openzeppelin's libraries and contracts, which follow the ERC721Metadata standard. This interface exposes basic fields and functions, such as name, symbol, and tokenURI. They are needed to properly manage the collection in the Omnite ecosystem. When a user holds a particular token for the collection which hasn't been yet brought to the Omnite's platform, first, it needs to be wrapped by Omnite's ERC721 wrapper, which follows the interface used by the platform's bridge (ITokenBridgeable).

All bridgeable tokens on Omnite's platform have to implement this interface, therefore wrapping the source contract is crucial. This step is executed along with multichain deployment. The step consists of:

1. Deploying the wrapper on the source network
2. Deploying the Omnite's ERC721 contracts on target networks

Both operations are executed in one transaction. The caller needs to provide the original contract's address, chain IDs that identify particular networks in the LayerZero ecosystem, and gasAmount for each deployment (gas amount is calculated in separate calls). The function being called is payable, since the gas cost for target networks is deducted from the value sent by the user. The surplus is sent back to the user. Omnite's bridge, in order to deploy contracts on target networks, packs the deployment message by encoding it to the Data struct with a Deploy operation type, contract type to be deployed, and collection ID, which is generated by encoding block timestamp and caller's address. Next, it sends the message to the bridges contracts on target blockchains. The message goes through LayerZero endpoint and is forwarded to the receiving Omnite bridge. The bridge unpacks the data and the deployment request is forwarded to ContractFactory. ContractFactory stores all the data needed for deployments and versioning of the contracts. Omnite utilizes beacon proxies: each new ERC721 is in fact a set of contracts following this pattern. They all share one blueprint (via beacon) and can be upgraded if needed. After deploying the BeaconProxy with the UpgradableBeacon and constructor params needed to initialize the new collection with ERC721 fields, the new collection is registered in CollectionRegistry, which is responsible for storing all the collections which were deployed or wrapped by Omnite smart contracts.

After the deployments, the token is now ready to be bridged. After an approval transaction, which is needed for the wrapper to be able to transfer the user's asset, the bridging begins. This transaction consists of:

1. Locking the token on the source network - it can be unlocked when the asset is bridged back. It is a much safer solution than burning the token, since the asset will always be unlocked. Token is locked on its smart contract and can be released whenever it is bridged back.
2. Calling the bridge to mint the token on the target network.

The move call is packed with CallData operation type and packedData bytes. It includes the function selector for the minting, an ID of the token to

be minted, and a tokenURI for this id. There is also the gas amount value and message value, similar to the deployment transaction. After receiving the message on the target network, the bridge unpacks the data and executes the minting function on the target ERC721. The address of the token is retrieved from CollectionRegistry. At the end of this whole process, there is an unlocked token on the source chain and a newly minted token on the target network. If the user decides to go back with his token, the asset is locked once again and an original one is unlocked. The process can be executed in perpetuity. If a particular target network holds the locked asset, there is no need to mint a new one.

6 Marketplace compatibility

Omnite smart contracts are designed to be fully compatible with leading NFT marketplaces. By following the metadata standard, each collection created on the Omnite platform has a metadata URI and an ownership management. After bridging the token to the target network, the user is still the owner of the NFT and can manage it with complete freedom. It can be listed on Opensesa[3], Rarible, etc. After selling the token, the new owner can bridge it back through the Omnite platform.

References

[1] *Ethereum ERC721.* URL: `https://eips.ethereum.org/EIPS/eip-721`. (accessed: 13.03.2022).

[2] *Layer Zero protocol.* URL: `https://layerzero.network/`. (accessed: 13.03.2022).

[3] *OpenSea Marketplace.* URL: `https://opensea.io/`. (accessed: 13.03.2022).

[4] *Openzeppelin proxy documentation.* URL: `https://docs.openzeppelin.com/contracts/4.x/api/proxy#BeaconProxy`. (accessed: 13.03.2022).