विVIDH

Collaborate • Create • Contribute

By

BChain Dynamics

Connect • Learn • Grow

SimpleFi

Sending Gasless Transaction

Team: Meta BLock
Members: Ishita Rastogi

Mentor: Koshik Raj
Guide: Nirav, Arpit

Table of Content

1. Abstract

"The greatest victory is that which requires no battle." — Sun Tzu

Meta transactions allow users to execute transactions on the blockchain with zero transaction fees. It builds a more seamless UX since users don't need to understand the inner workings like go to an exchange, pass a KYC process and wait for approval, transfer money from their bank, buy ETH, install MetaMask, learn how to use it and transfer ETH to it, etc.Due to this onboarding hurdle users are not coming and dapps are not taking off.

Meta transaction involves signing a transaction with a key that doesn't own ETH, and then wrapping it with another transaction signed by a key that does own ETH, to pay for the gas. The recipient contract "peels" the outer transaction and uses the inner one.

2. Introduction

Native meta transactions let users signal directly to smart contracts with signed messages. For instance, one could transfer an ERC20 that you own to a friend without ever having any ETH to pay for gas.

In meta transaction a third-party (called a relayer) can send another user's transactions and pay themselves for the gas cost. In this scheme, users sign messages (not transactions) containing information about a transaction they would like to execute. Relayers are then responsible for signing valid Ethereum transactions with this information and sending them to the network, paying for the gas cost.

3.Problem Statement

3.1 -> Existing System:

For most of the DApps today, user needs to follow below steps just to use the DApp

1. Download a Wallet (Metamask, Trust etc)
2. Get some ether/crypto from an exchange
   - Create an account on exchange
   - Upload documents for KYC verification
   - Wait for some days for your account to get verified
   - Buy the ether/crypto and transfer it to your wallet

Moreover

3.2 -> Problem with the existing system

Anyone who sends an Ethereum transaction needs to have ETH to pay for gas fees. This forces new users to pass KYC and purchase ETH before they can start using any dapp. This can be a major hurdle for users without prior crypto experience that are unfamiliar with the concept of needing to keep ETH in their wallet for gas.

This is also a UX pain for existing users that need to continually replenish their ETH balance to pay for gas fees even if they have tokens worth thousands of dollars in their wallet.

3.3 -> Solution

This problem can be solved by allowing end users to **use the DApp without downloading any external wallets or buying any crypto currency from an exchange**.

This is possible using the concept of *meta transactions* where the user is able to do a transaction on blockchain with zero balance account and any third party can pay for the transaction fees for the user.

We are using Native Meta transaction approach to solve the problem.
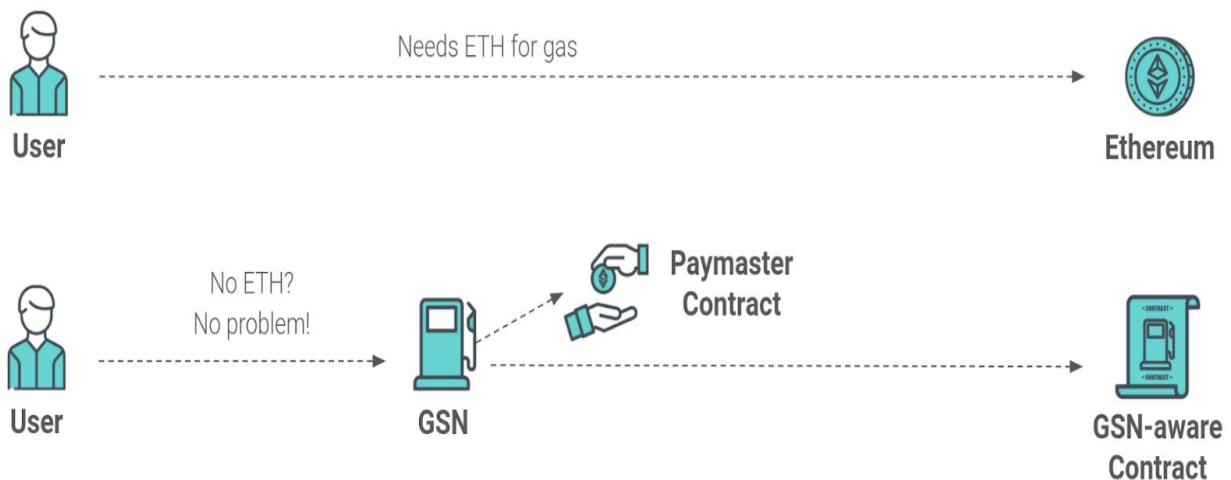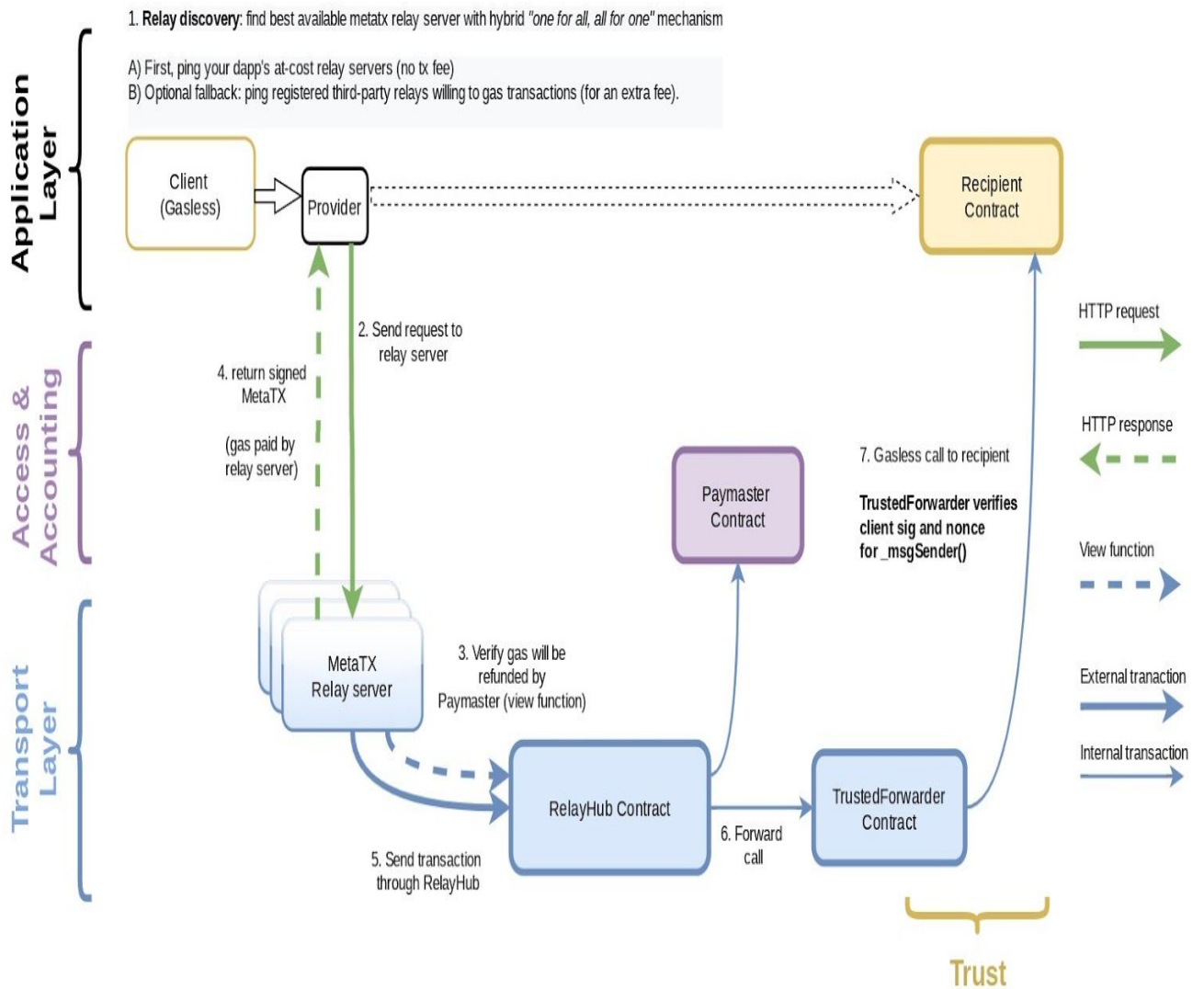
Native Meta Transaction
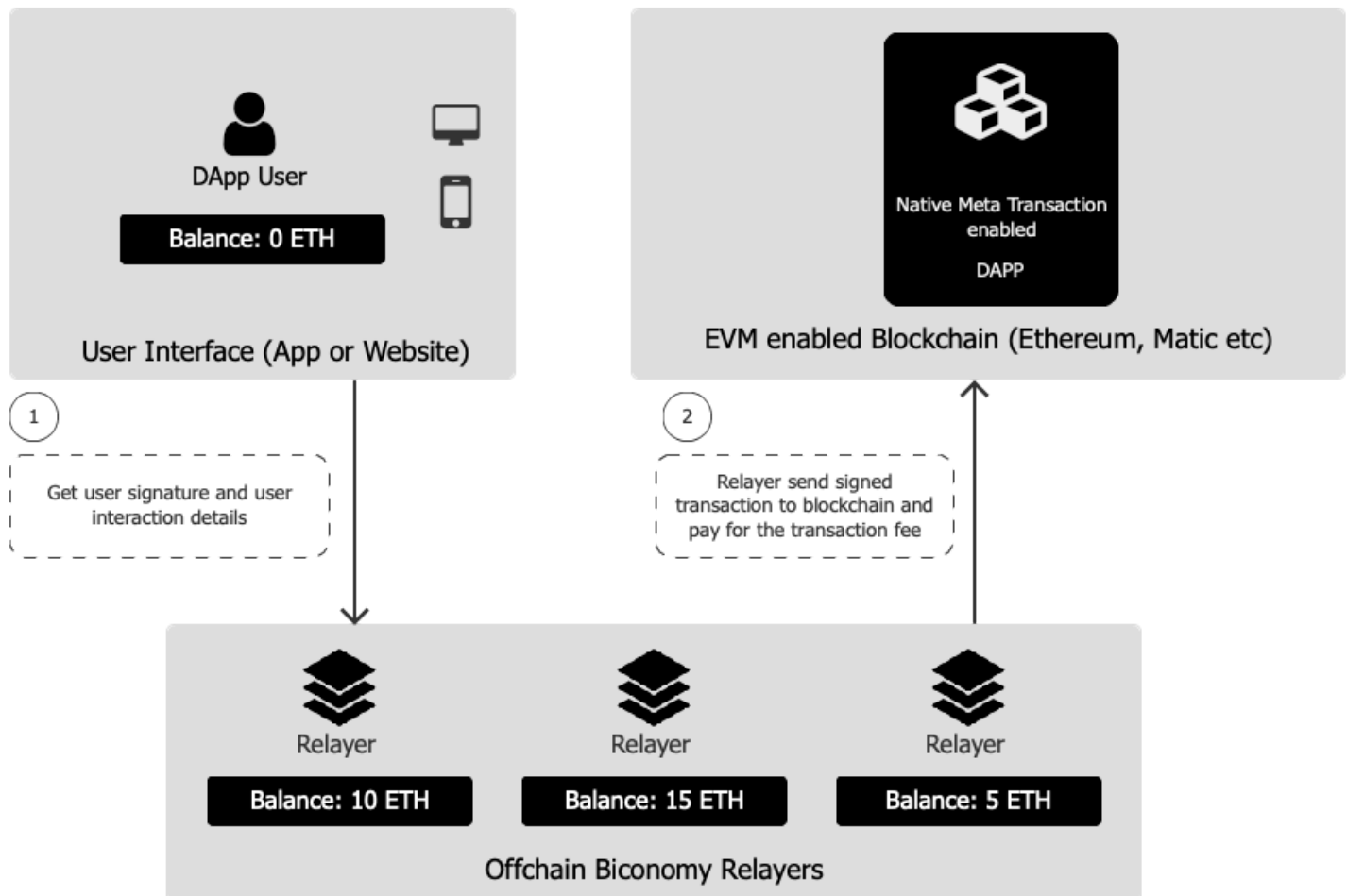
3.4-> Advantages

1.Ethereum Gas Station Network (GSN) abstracts away gas to minimize on boarding & UX friction for dapps.

2. With GSN, gasless clients can interact with Ethereum contracts without users needing ETH for transaction fees.

3.The GSN is a decentralized system that improves dapp usability without sacrificing security.

4. SYSTEM DESIGN

1. **Relay discovery**: find best available metatx relay server with hybrid *"one for all, all for one"* mechanism

A) First, ping your dapp's at-cost relay servers (no tx fee)
B) Optional fallback: ping registered third-party relays willing to gas transactions (for an extra fee).

Application Layer

Client (Gasless)

Provider

Recipient Contract

Access & Accounting

2. Send request to relay server

4. return signed MetaTX

(gas paid by relay server)

7. Gasless call to recipient

TrustedForwarder verifies client sig and nonce for _msgSender()

Paymaster Contract

Transport Layer

MetaTX Relay server

3. Verify gas will be refunded by Paymaster (view function)

RelayHub Contract

5. Send transaction through RelayHub

6. Forward call

TrustedForwarder Contract

Trust

HTTP request

HTTP response

View function

External tranaction

Internal transaction

5. Software Requirements

Truffle -> Testing and deploying our contract

Reactjs -> Building Front-end

Ganache -> CReating Local Blockchain

Open-GSN ->

6. Implementation

1. Meta transactions and how to sign these transactions.

2. Implementing ERC20 token

3. By binding all these together, will need to create a frontend that interacts with these contracts.

7. References

https://docs.openzeppelin.com/learn/sending-gasless-transactions

https://docs.opengsn.org/learn/