Community Points Engine

Table of Contents

1. Introduction

1.1 Overview

The main goal of the product is to give communities a means to create scalable points and rewards systems in a trustless manner. Due to technical limitations, the public blockchain infrastructure such as Ethereum cannot handle the required load of the modern points applications, thus a corresponding scaling solution should be introduced.

Community Points Engine (CPE) is a working proof of concept application that utilizes OMG Network as a trustless, high throughput settlement rail for community points transactions. The full demonstration consists of a working application, and this document which contains details on how special claim, burn, and subscription transactions can be implemented on the network. We demonstrate a full picture of how end users can have true ownership of community points and transact them in a manner that is fast, fee-less, trustless, and usable. All with the security guarantee offered by the Ethereum blockchain.

The demonstration consists of 5 different components:

1. Community Points Client: a chrome plugin that simulates client application that demonstrates Web 2.0 usability
2. Community Points Server: a fee relayer server that absorbs transaction fees for users
3. Community Points Contracts: a set of smart contracts that govern ERC20 Tokens
4. Community Points Claim demo: a script that demonstrates claim transaction
5. Community Points Admin Dashboard: a MultiBaaS dashboard that gives an administrator/moderator simple access to make transactions and treasury management

The working application doesn't include the following components:

1. Design blueprint for complete implementation
2. 25,000 subscriptions: due to time constraints, we were not able to demonstrate subscription transactions. However, we have proposed 2 designs for how this functionality could be implemented

1.2 Glossary / Terminology

CPE - Community Points Engine.

Community Points system - a general term for particular community points, rewards, or tipping system.

Community Points platform - a platform where the Community Points system is being hosted, such as Reddit.

Community Points server - a server that allows performing fee-less transactions for users of a given Community Points platform.

OMG Network - a Layer 2 scaling solution for the Ethereum network.

Rootchain - the Ethereum network.

Child chain - the OMG Network.

Plasma - the name of a scaling solution of the OMG Network.

Smart contract - a self-executing contract that is running on a blockchain network, such as Ethereum.

Mint/minting - an event of issuing/creating new tokens on the Ethereum network.

KARMA - a reputation token that can be distributed by the Community Points platform to users on the OMG Network. These tokens allow users to claim points of a given community.

ROCK - a community point token for r/OMGnetwork.

Burn/burning - an event of eliminating existing tokens out of circulation on the Ethereum or OMG Network. This event implies that the tokens are sent to the non-recoverable address.

Fee Relay - a method to allow for a third party wallet application to absorb the transaction fee for an end-user.

TCO - a total cost of ownership.

1.3 Product Requirements

User Personas

1. Customers/community members (e.g. subreddit community) - participants of a given Community Points system, service, network, website, portal, app, or forum who want to receive rewards, prizes, or discounts to stay more engaged.
2. Moderator or admin (e.g. subreddit moderator) - a person(s) who creates rules, coordinates users' activity, helps to solve problems within a particular Community Points system.
3. Management team (e.g. Reddit Community Points team) - a group of people who manage technical and operational requirements for a given Community Points system.

User Stories

1. As a user, I can send tokens to another user via OMG Network in my browser while engaging with my community.
2. As a user, I can see the balance of my tokens and my username in the browser while engaging with my community.
3. As a user, I can view the history of transactions while engaging with my community.
4. As a user, I can purchase various digital items while engaging with my community.
5. As a moderator, I can mint, distribute, and burn tokens.
6. As a management team, I can absorb transaction fees for users.
7. As a management team, I'm sure that my production service can run on the OMG Network with an expected load.

1.4 Requirements

The proposal fulfills all the below requirements.

Scaling

Over a 5 day period, your scaling PoC should be able to handle:

- 100,000 point claims (minting & distributing points)
- 75,000 one-off points burning
- 100,000 transfers

Decentralization

The solution takes numerous trade-offs in having a central point of control, but still maintains trustless ownership of user funds.

Usability

- Transactions complete in a reasonable amount of time (seconds or minutes, not hours or days)
- Free to use for end-users (no gas fees, or fixed/minimal fees that Reddit can pay on their behalf)
- Bonus points:

    - Users should be able to view their balances & transactions via a blockchain explorer-style interface
    - Exiting is fast & simple

Interoperability

- Scaling solution should be extensible and allow third parties to build on top of it
- APIs should be well documented and stable
- Documentation should be clear and complete
- Third-party permissionless integrations should be possible & straightforward
- Simple is better. Learning an uncommon or proprietary language should not be necessary. Advanced knowledge of mathematics, cryptography, or L2 scaling should not be required. Compatibility with common utilities & toolchains is expected.
- Bonus Points:
    - Build an extra use case for Community Points.

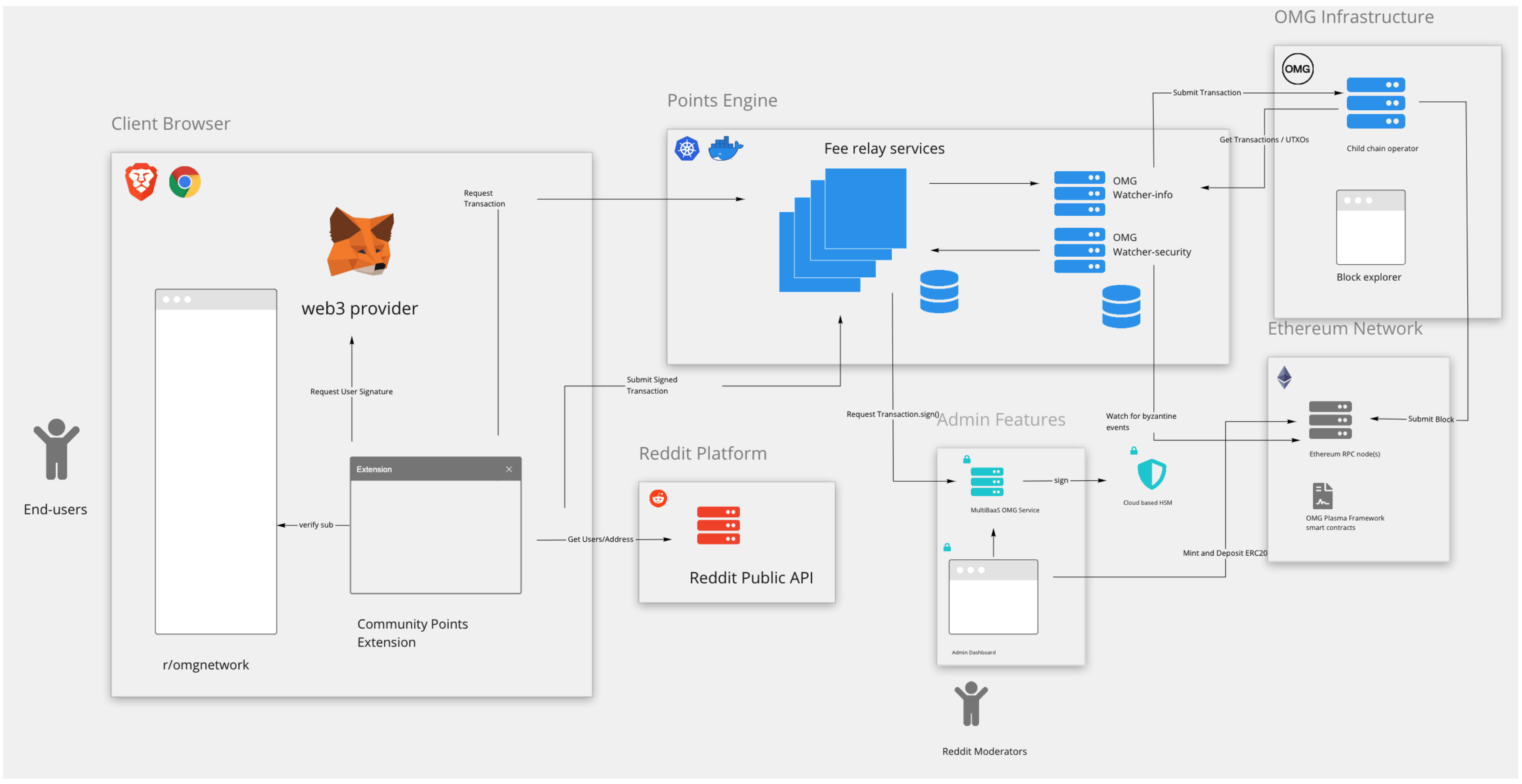
Security

The proposed solution fulfills the following security criteria:

- Balances and transactions cannot be forged, manipulated, or blocked by Reddit or anyone else
- Users should own their points and be able to get on-chain ERC20 tokens without permission from anyone else
- Points should be recoverable to on-chain ERC20 tokens even if all third-parties involved go offline
- A public, third-party review attesting to the soundness of the design should be available
- Bonus points:
    - Public, third-party implementation review available or in progress
    - Compatibility with HSMs & hardware wallets

2. Proposed Solution / Design

2.1 Demo Application Architecture



Community Points Client

We are utilizing the web browser as the user interface for this demonstration due to the simplicity of integration with current ecosystem tooling. The currently supported browsers are Brave and Chrome. Here are different components to the interface:

- r/OMGnetwork: the application is designed to work while the user is interacting in a specific community, in this case, it is `r/OMGnetwork`. This is also a way to introduce the product in a seamless manner and not risk breaking the experience of the Community Points platform.

- Web3 provider: any Web3 provider that injects a Web3 object into the DOM, exposes the current user's Ethereum account balance, and supports EIP712 transaction signing. Because we are reliant on a non-custodial solution, users will have full ownership of funds. For the demonstration, we push the responsibility of Web3 and wallet provider to MetaMask.

- Community Points extension: the user interface and core business logic for the user. It interacts with a few different services including the public Reddit API, Web3 provider, and fee relayer service. It currently offers multiple functionalities:
    i. Transact community points without fee tokens
    ii. Purchase items (e.g. Flair) with community points token
    iii. Address identity via community usernames
    iv. View balance, and transaction history

A browser extension is the easiest way to deploy and develop as a PoC. The Community Points client could be running on any other platform including natively on desktop and mobile as long as wallet signing APIs are provided.

Our extension demonstrates how Community Points functionality can be introduced in a manner that is abstracted away from typical blockchain usability thus allowing the end-user to retain the same experience as community points transactions on regular Web 2.0

Community Points Platform: Reddit

The Community Points Engine runs on top of an existing Community Points platform. In our case, we leverage Reddit's public API for the following functions:

1. *User identity*: to reduce complexity around maintaining usernames and wallet address relation, we rely on the Community Platform's API as the sole identity provider. However, a more robust or trustless identity approach could be introduced.

2. *Online items purchases*: we rely on authenticated API to fetch and redeem online items of each user. This allows a user on our platform to burn points in exchange for premium flair.

Community Points Server

The server-side of the Community Points Engine is where most of the complexity is handled. The server consists of multiple horizontally scaled Node.js services connected to a relational database. The services can perform the following features:

1. *Fee-relayed transactions*: the clients of the CPE can make fee-less transactions. From the usability perspective, it means that the user can make transactions on the network without paying transaction costs. This is also a useful abstraction for user onboarding without requiring additional ERC20 tokens and ETH. Fee-relay transactions is a design pattern and can be implemented natively on the OMG Network's UTXO transaction model.

2. *Serve multiple concurrent client transactions*: we utilize different services as a way of handling complex UTXO management. This means that the higher load on the network will not deteriorate the experience of other users. More client applications may transact with Points at the same time. This service meets the requirements for better transactional scalability and usability while preserving Plasma security.

3. *Cloud-based HSM module*: Given that compromised private keys can have catastrophic consequences for operators, we ensure that different key management approaches and standards are possible. High throughput hot wallet transactors like fee relayers are best managed on a Cloud-based hardware security module to be compliant with enterprise best practices. This implementation utilizes the MultiBaaS module along with Azure Vault.

We support modern operational standards and run this application inside a Docker container, orchestrated on Kubernetes with Google Cloud as the deployment target. However, infrastructure requirements can change based on the client's demand.

Admin/Moderator Dashboard

Treasury and fund management are difficult to achieve on a public blockchain, especially for platform administrators and moderators who are non-crypto natives. We show that we can provide a highly usable abstraction for back-office operations via an admin dashboard provided on the MultiBaaS.

Our community moderators can sign in via role-based access (RBAC) to make deposits, transactions, and Community Points distribution right on the dashboard without prior Web3 experience. They can have a choice to choose between non-custodial wallets such as MetaMask and Cloud HSM.

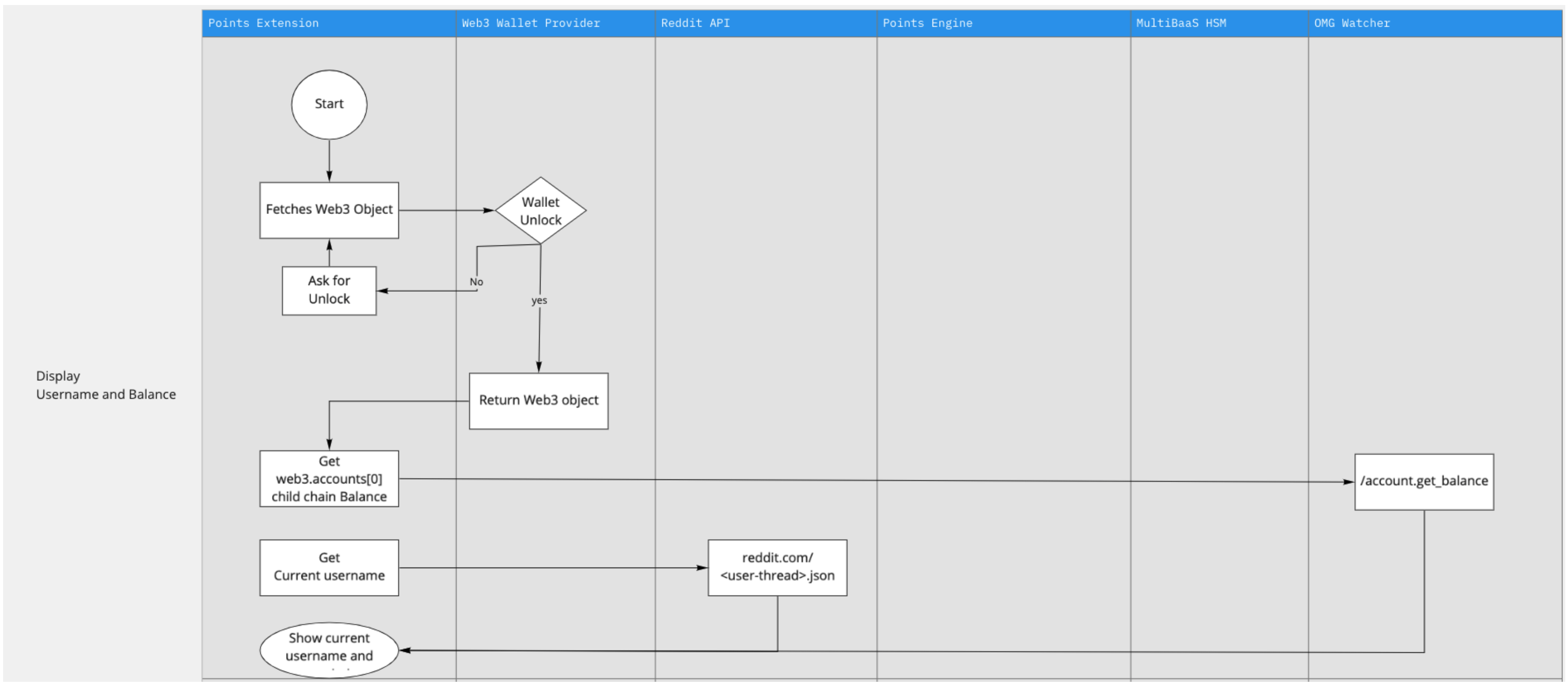
Blockchain Infrastructure

This is the settlement layer of community points transactions as well as how the Community Points Engine maintains its trustlessness guarantees. You can read more about the Plasma protocol here.

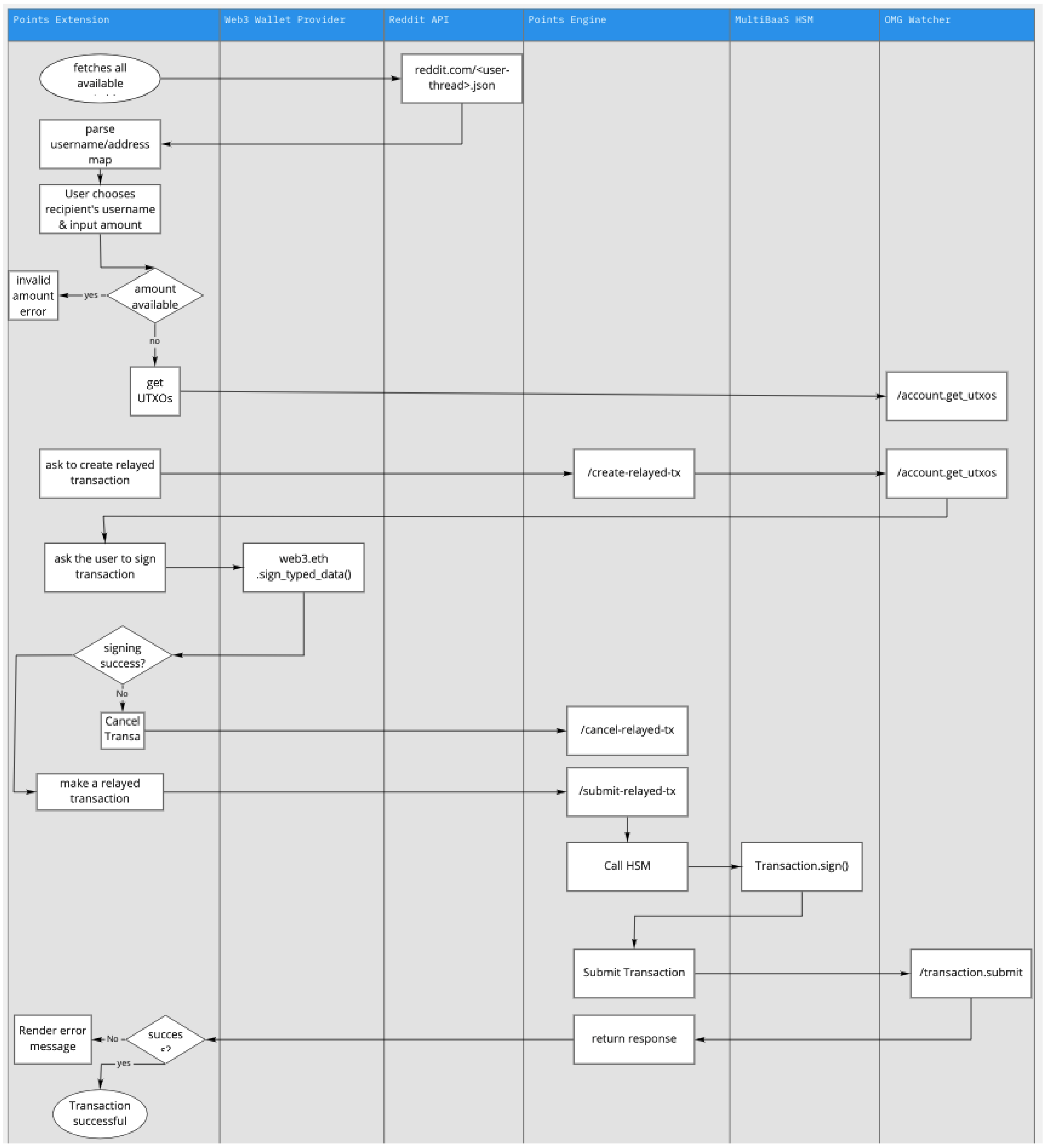
2.1.1 Application Flow & Diagrams

Swimlane Diagrams

1. *Display username and balance*



2. *Make a fee-less transaction*



Application Flow

The overall flow using such architecture involves the following steps:

1. A user opens a browser and installs the Community Points extension.
2. A user goes to a subreddit that supports community points (e.g. `r/OMGnetwork`).
3. An extension verifies that the user is located on a supported subreddit and prompts a signature request to connect via a Web3 wallet (e.g. MetaMask).
4. A user confirms the signature and appears on the main screen of the extension.
5. An extension fetches the user's data from the Reddit Public API and presents this info to the extension interface.
6. A user fills the amount and recipient username and initiates the transaction request to a fee relayer via an extension.
7. An extension fetches and takes spendable user's UTXO(s) that can cover the defined amount in the transaction.
8. An extension makes a POST request `/create-relayed-tx` to create a relayed transaction.
9. A fee relayer fetches its UTXO that can cover the transaction fee, and creates a transaction based on the user's UTXO(s) and fee relayer's UTXO.
10. A fee relayer requests a user to sign a relayed transaction.
11. A user signs a transaction with a Web3 wallet. If a user doesn't sign a transaction, the extension cancels it by sending a POST request `/cancel-relayed-tx` to a fee relayer.
12. An extension sends a POST request `/submit-relayed-tx` to a fee relayer to submit a transaction.
13. A fee relayer signs the transaction and submits it to the OMG Network via the Watcher service. If you want the fee relayer to sign a transaction via HSM, you need to make a request to a MultiBaas platform first.
14. Child chain operator creates a block on the OMG Network and submits it to the Ethereum network.
15. After the transaction is confirmed, the user receives a notification and can view the transaction in the Block Explorer.

2.1.2 Server

Fee Relayer API

This section demonstrates a general overview of the Fee Relayer API. For more details, please refer to Community Points Swagger document.

Create a Relayed Transaction

```
POST /create-relayed-tx
```

Constructs a relayed transaction.

Attributes:

Name	Type	Required	Description
utxos	[tx_input]	Yes	Transaction UTXOs
amount	BigNum, string, int	Yes	Amount of tokens to send
to	string	Yes	Receiver address

Submit a Relayed Transaction

```
POST /submit-relayed-tx
```

Submits a relayed transaction.

Attributes:

Name	Type	Required	Description
tx	tx	Yes	Transaction body
signatures	[string]	Yes	A list of signatures to sign a transaction

Cancel a Relayed Transaction

```
POST /cancel-relayed-tx
```

Cancels a transaction

Attributes:

Name	Type	Required	Description
tx	tx	Yes	Transaction body

2.1.3 Client

Views

Main View

The main view displays the name of the community points page (e.g. subreddit), the wallet address, and the current balance of the token that represents this particular Community Points system. A user should be able to copy the address by selecting it or clicking the copy button near the address.

Additionally, the view contains `Transfer`, `History`, and `Merch` tabs that represent corresponding data or actions. By default, a selected tab is set to `Transfer`.

Transfer View

Each user should be able to transfer funds to another user. One should put an amount and a recipient address that can represent a username (e.g. subreddit account name) or an Ethereum address, and confirm the transaction with a `Transfer` button. The username should be searchable.

History View

The History View contains a list of transaction history for a token that represents a defined points system, in which the user is currently located (e.g. r/OMGnetwork subreddit).

Each transaction item includes:

- An icon that represents sending or receiving of funds
- `Sent` or `Received` message for a corresponding transaction type
- Transaction status: `Pending` or `Confirmed`
- A wallet address of the recipient or sender (depending on transaction type)
- The amount and token symbol
- A copy button that copies the recipient or sender

Also, each item should be clickable and refer to a blockchain explorer of the network used during a transaction (Rinkeby, Ropsten, or Mainnet). If history becomes too long, a user should see only the first few items, and retrieve the rest by clicking on the `Load more` button.

Merch View

The Merch View contains a list of digital goods (e.g. flair) a user can purchase. Each item should have its own icon and price. Each purchase is associated with certain metadata that defines a possessed item and implies the burning of tokens (sending tokens to non-recoverable address).

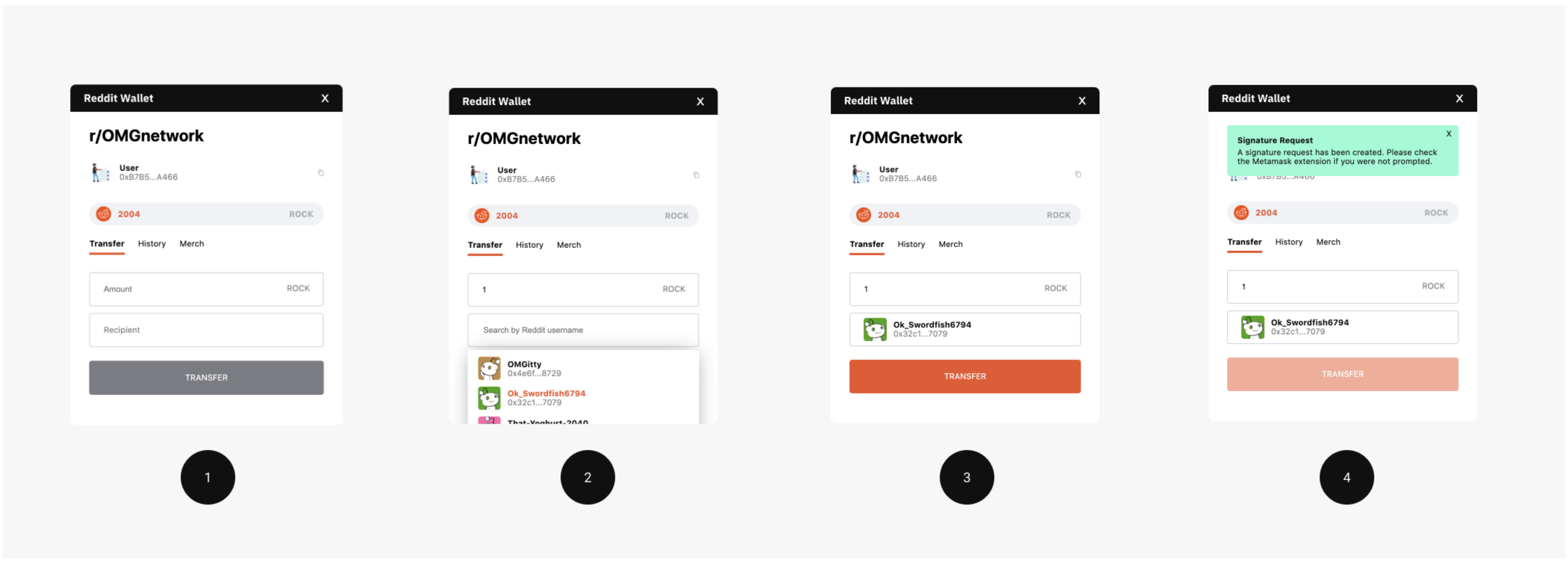
Notification Views

An extension should notify users in the following scenarios:

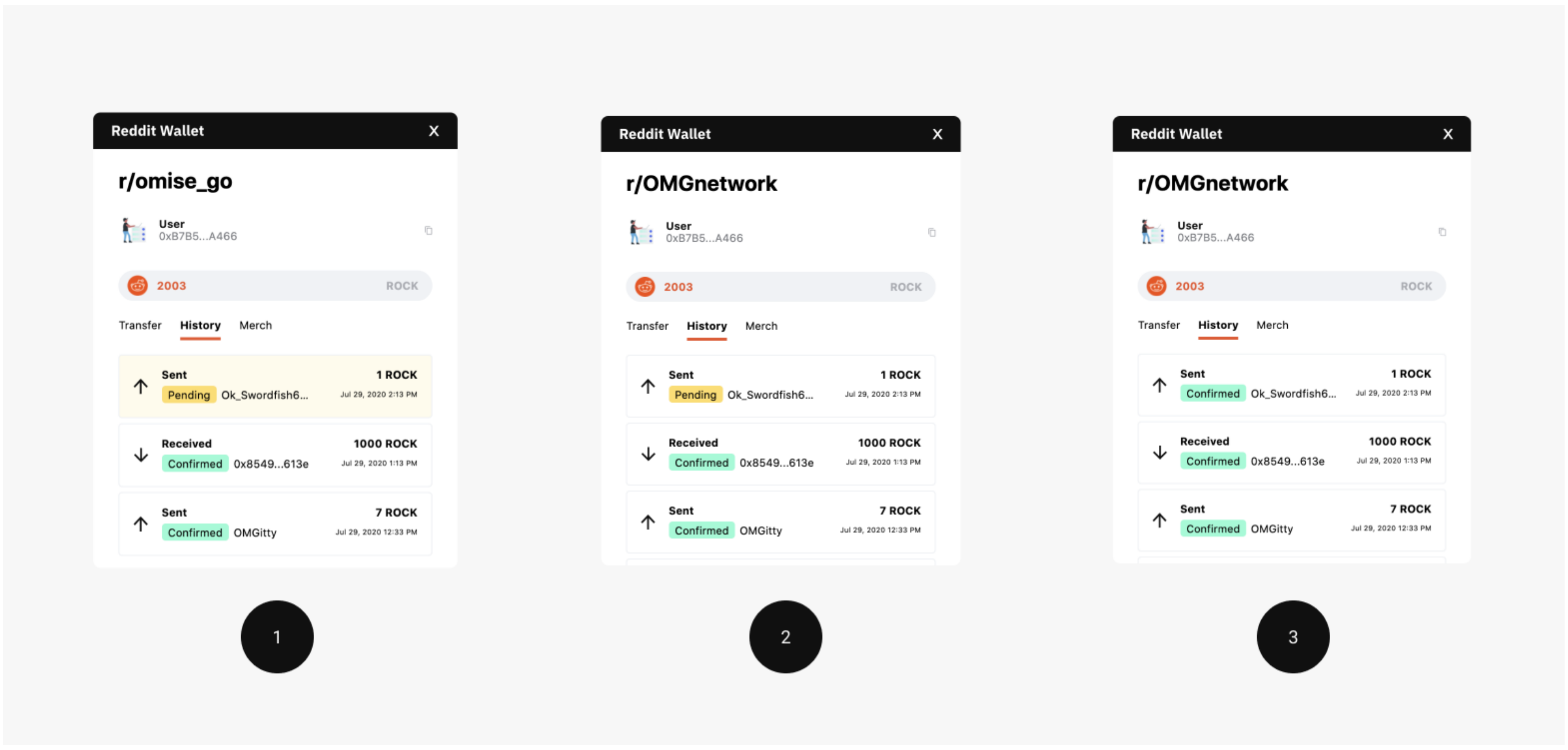
- When an extension is loading
- When a user is located on an unsupported community points page (e.g. subreddit)
- When a user doesn't have a MetaMask extension
- When a user is not logged in with the MetaMask extension
- When the user’s MetaMask is connected to the wrong network (e.g. Ropsten, instead of Mainnet)

Mockups

Main & Transfer Views



History Views



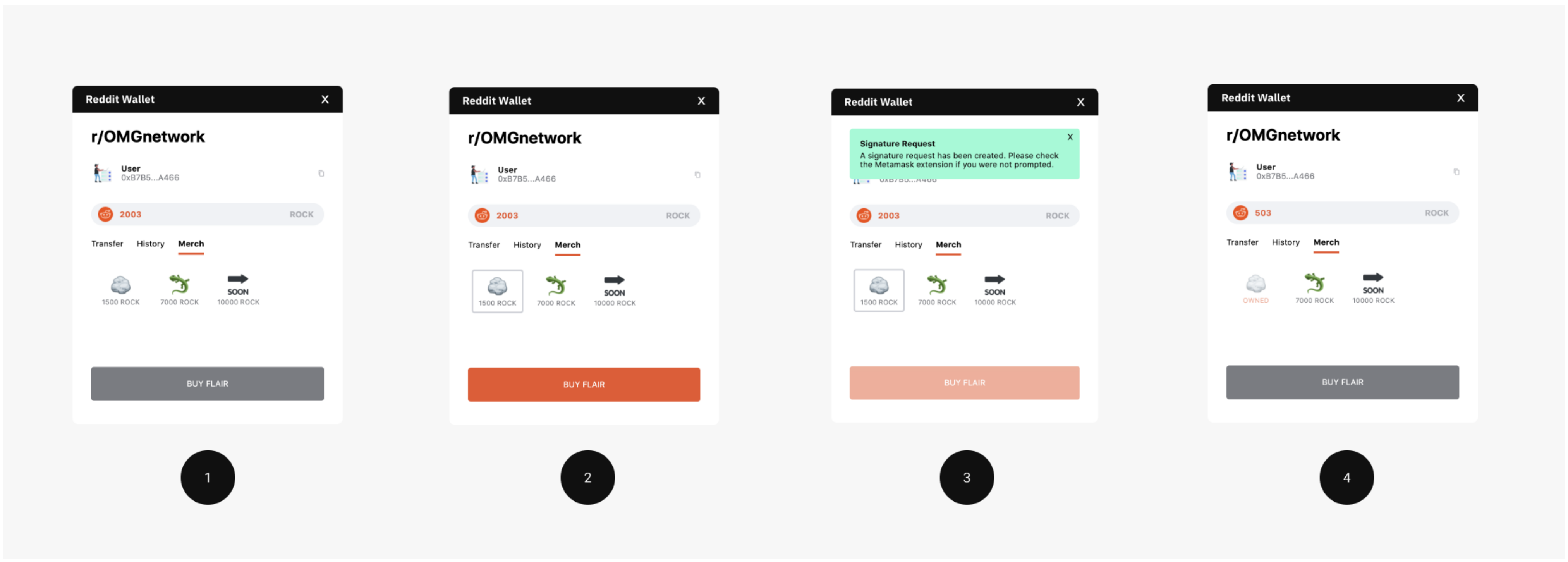
Sent 1 ROCK
Pending Ok_Swordfish6... Jul 29, 2020 2:13 PM

Received 1000 ROCK
Confirmed 0x8549...613e Jul 29, 2020 1:13 PM

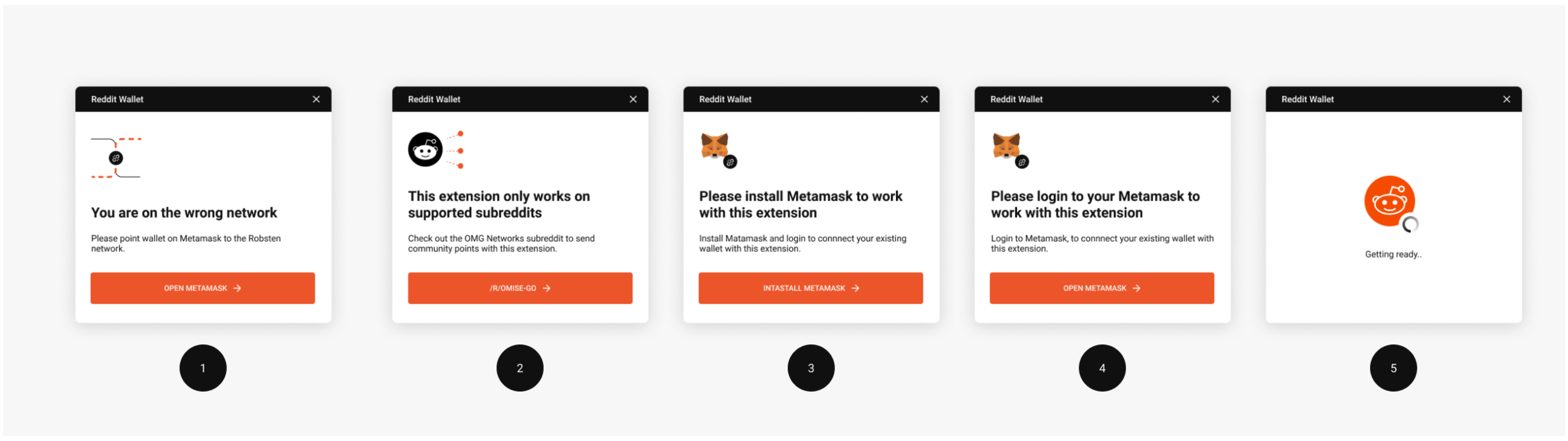
Sent 7 ROCK
Confirmed OMGitty Jul 29, 2020 12:33 PM

Sent 1 ROCK
Confirmed Ok_Swordfish6... Jul 29, 2020 2:13 PM

Merch Views



Notification Views



You are on the wrong network

Please point wallet on Metamask to the Robsten network.

This extension only works on supported subreddits

Check out the OMG Networks subreddit to send community points with this extension.

Please install Metamask to work with this extension

Install Matamask and login to connnect your existing wallet with this extension.

Please login to your Metamask to work with this extension

Login to Metamask, to connnect your existing wallet with this extension.

Getting ready..

2.2 Proposed Contracts Design

We try to balance implementation complexity with the necessary trustlessness requirement as outlined in the specs. However, tokenomics is a complex topic that requires a substantial amount of iterative design process. For the purpose of this application, we will treat the subscription process as a black box, while offering 2 design choices that Community Points can take into production with sample code for demonstration purposes. Each solution design makes tradeoffs in cost, complexity, and feature requirements.

The proposed solutions to facilitate claims and subscriptions include:

1. Simple Claim and Subscription with full compatibility with current OMG Network V1.
2. Claim and Subscription as special OMG Plasma Framework transaction types.

The following section describes the first approach. The latter one will be outlined in the third section.

2.2.1 Smart Contracts

The current design consists of 4 contracts:

- Distribution
- Subreddit Point
- KARMA token
- Subscription token

Distribution contract

This contract holds the main logic to decide how many community points (e.g. subreddit points) should be distributed on each round. This contract mints the community points according to the formula in this contract.

For each round, it records the `baseSupply` and `availablePoints`. `baseSupply` is an integer that decreases a certain percentage on each round. The `availablePoints` is the amount of points that can be distributed on a certain round. `availablePoints` equals to `baseSupply` + 50% of the burned points on the previous round.

Subreddit Point contract

This is an ERC20 contract for the subreddit points. It should enable the `Distribution contract` to mint community points on each round.

> Note, this contract was designed specifically for a Reddit use case. It may have a different name for another Community Points platform.

Karma token contract

This is a representative of KARMA. It is an ERC20 token that can be distributed by the Community Points platform (e.g. Reddit) to users on the OMG Network. Users can use the received KARMA to create a "Claim" transaction to receive points of a given community (e.g. subreddit points).

Subscription token contract

Instead of having an expiration date on each subscription, we use a simpler design where a subscription token represents the subscription status of a certain period. For instance, there can be a subscription token representing the subscription of January 2020. Proof of ownership of the token shows the subscription status.

This simplifies the timing issue on Layer 1 and Layer 2. Also, this enables the possibility of using an ERC20 token to represent subscription and do it on the current version of the OMG Network.

2.2.2 User Flow

Initiate next distribution round

This flow would be initiated by the community owner. It should call a `Distribution` contract to start the next round. Inside the `Distribution` contract it will auto call the `SubredditPoint` contract to mint the token according to the total available points of the next round.

Distribute KARMA to user

Community Points platform (e.g. Reddit) should from time to time (or on-demand) mint KARMA token (ERC20) on the Ethereum and deposit it to the OMG Network. Community Points platform should send KARMA to its users according to their activities on the platform.

User claiming community points

With the KARMA given by Community Points platform, the user can ask for an atomic swap transaction as the claim of the community points. The server generating the transaction would provide the transaction with the community points according to the percentage of the KARMA amount and the total KARMA of the distribution round.

After the transaction is generated, both the user and the Community Points server would sign the transaction then submit it to the OMG Network.

User subscribes to a periodical (e.g. monthly) subscription

The community owner would deploy and mint the subscription token contract every certain period. After that, the subscription token would be deposited to the OMG Network. For the user to subscribe, the user can ask for an atomic swap transaction that swaps the community points to the subscription token.

2.2.3 Claim Transaction Demo

The current implementation is based on the design outlined above. The demo represents a script that mints KARMA and RCP tokens (community points), deposits them to the OMG Network, and performs an atomic swap allowing a user to claim RCP.

Actors

The current demo includes the following actors:

- Distributor - an address responsible for KARMA and RCP tokens distribution.
- User - an address that wants to claim community points (RCP tokens) in exchange for KARMA tokens.
- Community Points server (e.g. subreddit server) - a server that handles claim transactions.

Application Flow

The overall flow using the demo involves the following steps:

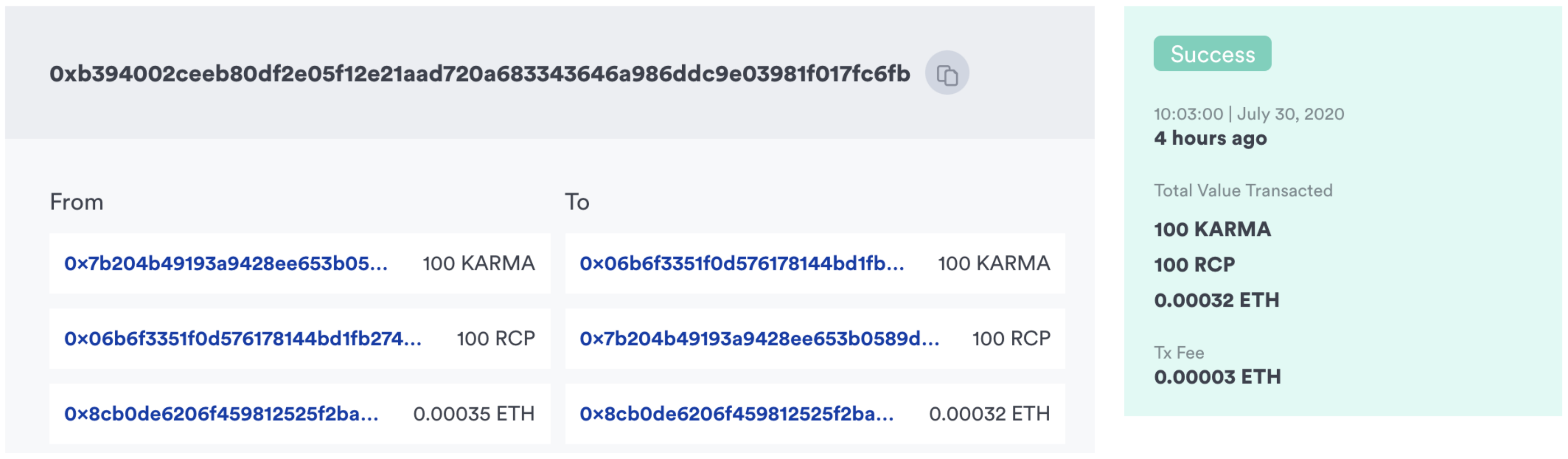
1. Distributor mints a defined amount of KARMA, and RCP tokens from the smart contracts
2. Distributor deposits ETH on the OMG Network to cover the fee costs.
3. Distributor approves KARMA and RCP tokens of a defined amount to make a deposit.
4. Distributor deposits KARMA and RCP tokens of defined amount on the OMG Network.
5. Distributor transfers a defined amount of KARMA tokens to the user on the OMG Network.
6. Distributor transfers a defined amount of RCP tokens to the Community Points server on the OMG Network.
7. User submits a signed (atomic swap) transaction to claim the RCP tokens.
8. Community Points server and Distributor sign the transaction.
9. Transaction is submitted to the OMG Network.
10. User receives a defined amount of RCP tokens.

Atomic Swap Description

The current demo uses an atomic swap to claim the community points based on the amount of KARMA tokens sent to its address during the distribution period. On a high-level, the swap represents an exchange of the same amount of RCP and KARMA tokens between a user and a Community Points server.

The claim transaction requires 3 signatures to be successful. Each token has to be signed by the corresponding owner: User - signs an input for KARMA token. Distributor - signs an input for a fee token (ETH or OMG). Community Points server - signs an input for RCP token.

Here's an example of a claim transaction:


0xb394002ceeb80df2e05f12e21aad720a683343646a986ddc9e03981f017fc6fb

From		To	
0x7b204b49193a9428ee653b05...	100 KARMA	0x06b6f3351f0d576178144bd1fb...	100 KARMA
0x06b6f3351f0d576178144bd1fb274...	100 RCP	0x7b204b49193a9428ee653b0589d...	100 RCP
0x8cb0de6206f459812525f2ba...	0.00035 ETH	0x8cb0de6206f459812525f2ba...	0.00032 ETH

Success

10:03:00 | July 30, 2020
4 hours ago

Total Value Transacted
100 KARMA
100 RCP
0.00032 ETH

Tx Fee
0.00003 ETH

Production Considerations

There are a few things to consider if you want to convert this proof of concept into a production service.

- For simplicity purposes, the Karma and RCP tokens can be minted by anyone. In production, KARMA tokens should be minted by the contract owner only. Community Points tokens (e.g. RCP tokens) may be minted by the correspondent Community Points moderator (e.g. subreddit admin). This can be changed in configurations.
- The keys for the Community Points engine and users (e.g. Subreddit server) are generated within the application, which is sufficient only for demo purposes.
- The amount in UTXO(s) used for KARMA and RCP inputs should match the exact amount of tokens you want to swap. This is due to the limitation of 4 inputs and 4 outputs on the OMG Network. In production, you should merge or split UTXO(s) to get the exact amount needed in the claim atomic swap.
- The demo mints KARMA and RCP tokens on demand when the script is starting. In production, the owner of the corresponding tokens may mint tokens beforehand.
- The demo was tested on Ropsten and Rinkeby environments due to the Ethereum network gas price on Mainnet.
- The demo includes a deposit of a fee token to the OMG Network to cover the fee costs for the distributor. Note, the fee token on the Mainnet is OMG, on Ropsten and Rinkeby - ETH.
- The KARMA/RCP exchange rate is hard-coded at 1:1, whereas this will vary in real life.

3. Further Considerations

3.1 An Alternative Claim Transaction Design

The proposed implementation is one of the approaches to handle claim transactions. An alternative solution could be using special transaction types on the OMG Network. You can read a high-level overview of the potential implementation in this document. Note, this is only a theoretical concept and hasn't been tested in any environment yet.

3.2 Third-party Services

To run any OMG Network application, it's required to use the following services:

- Ethereum node: to sync with the rootchain. You can run your own node or use one of the infrastructure providers, such as Infura.
- Web3 wallet: to sign transactions. It's recommended to use MetaMask browser extension.

Additionally, you may consider using MultiBaas to access an intuitive administration panel that allows the following:

- Issuing, approving, and burning ERC20 tokens
- Managing deposits, transfers, and exits on the OMG Network
- Signing transactions with a hardware security module (HSM)

3.3 Cost and Performance

There are various ways to determine cost and performance on the OMG Network in the context of community points. Many blockchain projects develop benchmarks on a widely different basis that negates the practicality of having direct comparisons. We try to take the most pragmatic approach to cost and throughput assumptions with the business needs in mind while trying to be transparent whenever possible.

Listed below are a few dimensions of cost and performance metrics, their definitions, and associated figures to achieve objective reasoning about network viability.

3.3.1 Throughput Performance

Maximum Theoretical Throughput

Many projects will quote theoretical throughput. It is possible to judge a Layer 2 performance based on its maximum throughput achievable by its design.

If we were to compare to other projects in this dimension, the maximum block size for a Plasma block is at 65,536 (as limited by Merkle tree). Given a 12-second Ethereum block, the maximum block throughput would be 5,461 TPS.

Although we could try to manually fabricate 65K Transactions and submit them into a Plasma block and forego the rest of the tech stack, our point of view is that this approach has not taken into account how real-world business transactions work, nor is it reflective of bottlenecks faced by real software and therefore should not be taken as a definitive figure for performance comparison. It is a non-productive benchmark for serious applications.

Real Application Transaction Throughput

We wanted to demonstrate the capabilities of our system for the suitability of managing the Reddit Community Points ecosystem. As part of this bakeoff application process, we've simulated transactions that align with Reddit's use cases on our public blockchain. We elected to use our public Integration environment on Ropsten for this as the gas prices on the production Ethereum network are high.

Our stack is built and managed using IaC (Infrastructure as Code), with our smart contracts, Terraform modules, Helm charts, and applications all versioned with SemVer. We have confidence in the consistency of environments and the ability to serve this from our Mainnet environment.

We use the acronym `ST4R` to mean "Stress Test For Reddit".

100,000 Point Claims (Minting & Distributing points)

We performed an exchange of KARMA and Community Points in one transaction, as described in the smart contract section. However, the transactions here are missing a fee relayer and the execution is done via an atomic swap with only 2 wallets.

We use transaction metadata `ST4R CLAIM` encoded into hex as `0x000000000000000000000000000000000000000000005354345220434c41494d` for these transactions. We performed 119130 transactions over two hours.

To verify, use the Block Explorer at https://blockexplorer.ropsten.v1.omg.network/ or

```
curl --location --request POST 'https://watcher-info.ropsten.v1.omg.network/transaction.all' \
--header 'Content-Type: application/json' \
--header 'Accept: application/json' \
--data-raw '{
    "metadata": "0x000000000000000000000000000000000000000000005354345220434c41494d"
}'
```

The response is limited to 200 transactions and is paginated.

25,000 Subscriptions

Due to time constraints, we were unable to complete this part of the request for the Proof of Concept. We do have two proposals that we believe satisfy this requirement. These are documented here.

5000 One-off Points Burning

For the points burning capability demonstration, we sent 81341 transactions to a `0x` address for which there is no owner. This simulates the burning of real points. We used the metadata field again so that the transactions can be verified with `ST4R BURN` encoded into hex as `0x000000000000000000000000000000000000000000000053543452204255524e` for these transactions. We performed transactions over an hour.

To verify use the Block Explorer at https://blockexplorer.ropsten.v1.omg.network/ or

```
curl --location --request POST 'https://watcher-info.ropsten.v1.omg.network/transaction.all' \
--header 'Content-Type: application/json' \
--header 'Accept: application/json' \
--data-raw '{
    "metadata": "0x000000000000000000000000000000000000000000000053543452204255524e"
}'
```

The response is limited to 200 transactions and is paginated.

100,000 Transfers

We minted ROCK tokens which are points that can be used with the `r/OMGnetwork` subreddit. These tokens were then deposited into the Plasma smart contracts and could be used to distribute as points on our Layer 2 chain.

We performed 121461 transactions over the course of four hours to demonstrate that the OMG Network is capable of handling this load. We used the metadata field of the transaction populated with `ST4R` to use as a filter for independent verification of the transaction volume. The hex value for this is `0000000000000000000000000000000000000000000000000000000053543452` and it's possible to view these transactions in our Block Explorer at https://blockexplorer.ropsten.v1.omg.network/

Alternatively, you can query the Watcher for these transactions via https://blockexplorer.ropsten.v1.omg.network/

```
curl --location --request POST 'https://watcher-info.ropsten.v1.omg.network/transaction.all' \
--header 'Content-Type: application/json' \
--header 'Accept: application/json' \
--data-raw '{
    "metadata": "0x0000000000000000000000000000000000000000000000000000000053543452"
}'
```

The response is limited to 200 transactions and is paginated.

3.3.2 Cost

There is always a cost associated with operating a system. Although cost optimization may make for a better business sense, near-zero cost networks will rarely achieve full trustlessness. We believe cost and security go hand in hand. A project picking a trustless solution on the basis of lower cost should seriously consider the degree to which they may require blockchain level of security guarantees. The risk and reward are highly dependent on the type of businesses and applications.

As with performance, cost per transaction cannot be derived or compared in a trivial manner without taking into consideration multiple other factors e.g. number of transactions per block, Ethereum gas price, gas usage.

Gas usage per number of transactions.

Due to the nature of Plasma block submission, the gas amount usage is constant for any number of transactions in a Plasma block, due to the fact that Plasma block stores the Merkle root on Ethereum with data running off-chain. The Merkle root hash is a 32-byte string no matter how many transactions are included.

Number of transactions per block	Gas used per block	Gas cost per transaction
100	73,269	732.69
500	73,269	146.538
1000	73,269	73.269
5000	73,269	14.6538
10000	73,269	7.3269
30000	73,269	2.4423
60000	73,269	1.22115

Gas cost per transactions in a block

Gas cost per transaction is dependent on multiple variables, including the current gas market on Ethereum as well as the price of ETH. We have the table that shows the gas cost per Plasma transaction based on the current gas price of 49 Gwei with a current ETH price of $332.53.

The formula is `ETH Price * Gas price * Gas used / Number of transactions in a block`

Number of transactions per block	TPS (12 seconds block time)	Gas cost per tx in $
100	8.333333333	0.01193842888
500	41.66666667	0.002387685776
1000	83.33333333	0.001193842888
5000	416.6666667	0.0002387685776
10000	833.3333333	0.0001193842888
30000	2500	0.00003979476293
60000	5000	0.00001989738147

Note that this is the cost in relation to an Ethereum transaction that submits a Plasma block. The cost incurred by Reddit for handling user transactions on the platform is discussed below.

Transaction fee charged by OMG and the operator model

Three considerations for running an application on an operator-led Layer 2 are:

1. How the operator bears the cost of the transaction.
2. The algorithm for block submission.
3. The number of transactions in a given block and frequency of submissions.

Currently, OMG Network charges transaction fees to end-users in the form of OMG tokens. The fee charged is 3x cheaper compared to the average Ethereum ERC20 transaction gas price. This is due to the performance optimization that is in place for the child chain service to submit a block every time there is at least 1 transaction. Even under low volume, the cost is fixed for the operator as described in the previous section.

However, with higher transaction numbers per Plasma block, we can expect the transaction fees to be an order of magnitude cheaper than the current 3x cost saving. This is because the higher a Plasma block is filled the greater the overall savings are compared to transacting directly on Ethereum. The transactions generated by a business on OMG will have a net positive effect on the wider ecosystem. You may refer to the above figure to get an idea of how this may look like.

The unit economics for being a Layer 2 service provider should always be taken into account for determining the TCO of the solution.

3.3 Security Considerations

Trustlessness vs Decentralization

Although the network is reliant on the child chain operator to aggregate transactions into blocks and submit the Merkle root to Ethereum, the users do not need to put trust in the operator for the security of their funds. At any point, users can leave the network in the case that the operator is compromised. You can learn more about the basics of Plasma fraud proofs and the benefits for users of the system here.

What are the potential threats and liabilities?

The potential threats for Community Points include the following:

1. Byzantine events that can happen on the OMG Network: Unchallenged exit, Invalid block, Block withholding, Invalid exit, Non-canonical in-flight exit, Invalid in-flight exit challenge, Invalid Piggyback. The events can either be challenged by another user or in the worst case, the users need to leave the chain as it has become unsafe. These events are not specific to this particular use case but are worth mentioning due to the technical implementation of the network.

2. The moderator starts manipulating token governance. Any moderator can behave maliciously and manage community points irresponsibly, e.g. issue more tokens than needed, send a large number of tokens to their friends, purchase a limited number of digital items, etc.

3. The user loses the seed phrase or the private key to a Web3 wallet. The way to manage funds on Ethereum or any other blockchain network is quite different from traditional banking services, thus security education should be one of the main priorities for Web3 wallets and similar applications.

4. A spam attack on a fee relayer. A group of people can coordinate a spam attack of transactions until the funds that are dedicated to paying for users' fees are depleted, thus it won't be possible to have free transactions using the client. This attack vector can be mitigated by leveraging the Reddit account system to whitelist users who should be able to send transactions and also control the volume that can be sent.

How will they be mitigated or responded to?

1. The occurrence of any unchallenged byzantine event should notify users to exit their funds from the OMG Network. Backend services could be introduced in a manner that automatically gets the clients to sign an exit transaction, requiring no manual operation from users themselves. All of the transactions that happen on the OMG Network are verifiable on the Ethereum network allowing any user to withdraw their funds back safely at any point in time.

2. Mitigation for moderator abuse is to enforce a multisig transaction on the Community Points platform. This way, a transaction must be signed by `M of N` signatures of all moderators. This means the problem of security for a particular community is offloaded to community governance. Other mitigations can be introduced by protocol enforcement via special transaction types.

3. The OMG Network doesn't have control over users' funds and can't verify the safety measures used by each user. Our team is doing our best to cover the best security practices in various guides and during the onboarding process of all of our services. The current demo application doesn't store any funds, thus reduces the risk of losing your tokens.

4. To prevent funds depletion the following measures are introduced:

- The wallet used to cover the fees will be refilled regularly based on a certain threshold.
- If the fee relayer account doesn't have enough UTXO(s), it will split up the fee change output of the relay transaction.

Do the end-users have to be concerned with data availability or monitor the chain for byzantine events?

It is recommended that each entity that does not wish to trust OMG as a chain operator should run their own Watcher service.

OMG Network offers a Watcher instance on Mainnet as a public service that anyone can use. The Watcher can also be managed by the community platform provider like Reddit itself, a blockchain node service provider, or self-hosted via the core community members.

4. Resources

- Contracts Audits
- Load Test Results
- Tests: contracts, fee-relayer
- FAQ
- User Guide
- Moderator Guide
- Deployment Guide