Holyheld: Blockchain Reconciliation and Remittance Record

Anton Zagorodnikov, PhD & Anton Mozgovoy & Holyheld

May 2024

Abstract

This document describes the reasoning, design principles, and architecture of a system serving as a decentralized, crosschain, multiasset service for clearing and reconciliation of information between multiple parties. We propose BRRR: a decentralized clearing house. BRRR is a set of smart contracts on a collective of blockchain networks that allows execution layer participants to securely combine intent-based offchain execution of onchain logic. Networks, protocols, and services opt-in by interacting with BRRR's smart contracts. By sending an instruction, BRRR allows execution of the 3rd party smart contract logic with settlement across all participants, creating a new cryptoeconomic layer. By opting into BRRR, decentralized applications (dApps), smart contracts, bridges, keepers, and solver networks use the execution layer of consensus networks and data availability networks to allow for the programmability of traditional payment systems. BRRR creates a connection layer access across all of them. It creates an opportunity to use crypto as an open settlement layer, where customers of both onchain and offchain services can benefit from the programmability of the onchain layer, atomically executing an instruction guaranteeing further settlement. Finally, BRRR creates a new era of permissionless innovation, in which neither traditional nor onchain protocols and services have to rely on the execution guarantee of each side, but instead, create services and products that access both cryptoeconomic and traditional layers secured by BRRR.

Contents

1 Introduction

1.1 The problem: Existing payments are not programmable

Financial institution is a mechanism for organizing society's resources to solve problems. The new radically "different" business models that are simultaneously evidence of the existence of trends and will directly respond to the tasks and challenges associated with changes around us.

Trends related to the behavioral change in economic activity and changes in the macro environment allow us to see in which direction the financial sector is moving. A look at new business models gives us a feeling of how protocols can serve our financial needs, with new foundational functionality. But most importantly, it helps us understand how our attitude toward finance is changing in a distributed environment.

The closing era is primarily associated with the Uberization of all industries. Namely through using of a digital platform to create added value for service by combining customers and service providers on it. In other words, companies create marketplaces based on the existing infrastructure to serve new needs.

It became clear that, practically, any problem can be solved on the go: book a hotel during a taxi ride, book a taxi while you leave the hotel. Everyday purchases, various services, and products have all become as affordable as possible. At the same time, access to finance, besides various interface intermediaries, has not changed much. Better-looking interfaces with core financial rails built on foundational principles from the previous century. On the surface, access to your money got easier, but in principle, settlement, reconciliation, and remittance remain the same. Most importantly, access to new financial primitives was reformed from serving the customer to monetizing the customer.

Social networks received a powerful push in the consumer goods industry: when the YouTube mobile app began to generate more traffic than the desktop site; and when social networks on smartphones started to accelerate the speed of spreading news and trends. Now, a single tweet or a post on Instagram can activate the spread and reaction of hundreds of thousands and even millions of people around the planet in minutes. At the same time, we started seeing the emergence of a digital economy. Uber and Bitcoin are products of the same year. However, the polarities of the development of their ecosystems have only recently started to show the benefits.

In addition to the web, smartphones brought a new format — mobile applications, and a new direction of mobile software development was born. As a result, certain niches of mobile developers immediately became scarce and highly paid. An interesting manifestation of these trends was the feeling of new freedom, people began to travel more actively and share their experiences through blogs, video blogs, and, as a result, even streaming platforms. The concept of a new cosmopolitanism has emerged — digital nomads, people who, thanks to the boom in the development of the Internet and mobile technologies, no longer want to physically be attached to

countries and cities. Which in turn poses an even more acute problem of decentralization of finance.

Since the demand of instantly appearing digital consumers was not closed by the traditional industry, new, mobile-first, and digital companies appeared and covered this "gap". However, building digital storefronts and user interfaces using the infrastructure of traditional industries. So began that very widespread Uberization.

In a relatively short period, Uber-like giants Deliveroo and Instacart appeared as food delivery aggregators, Uber and Lyft in the taxi segment, Spotify and Apple Music in music consumption, and Airbnb and Booking.com in the real estate and travel segment. WeWork and Regus are leading the field of coworking.

All these companies have one thing in common — they are fast and customer-oriented intermediaries between digital consumers and the huge infrastructure built before their existence.

Today, we can say that this approach turned out to be doubly successful and reasonable. These companies were able to provide the main value to their customers (building interfaces for new user requests). The worse the existing companies covered this need, the more the catharsis was felt, for example, in the financial sector. At the same time, abstracting from complex and expensive investments and operations in infrastructure. The main evidence for this model is the fact that companies not only appeared but also grew to multi-billion capitalization in a variety of markets.

However, the era of Uberization is approaching the end. Firstly, by removing the access layer to 3rd party agents, closed ecosystems are forced to verticalize the offering, gaining efficiency short term, but slowing down the progress long term. Secondly, most industries where it was possible to experiment with previously created infrastructure using a digital storefront already exist. Thirdly, consumer preferences, needs, and patterns continue to evolve. The market is growing in the number of people of generations Y and Z who are so used to smartphones and Uberization that having a user-friendly interface is not an advantage, but only a necessity. New consumers are putting forward new, more serious requirements for companies, goods, people, and services, which cannot be satisfied as a digital intermediary in a closed ecosystem.

In-between consumer and product there are numerous intermediaries each one of them executing, transmitting, or relaying a simple instruction. At each step, there is a reconciliation of debit and credit instructions. It is not possible to program or enrich such instructions. At the same time, the onchain layer allows for programmability, but it previously lacked the compatible settlement layer, creating a huge gap in the market. And so the emergence of exchanges and so-called *"on and off-ramps"*.

Such a difference makes it difficult to achieve a universal solution without an introduction of additional liability risk for using onchain assets in traditional systems. For instance, the average Ethereum Mainnet block time is 12 seconds, however, finality is achieved in 2 epochs or approximately, *15 minutes* [1]. Block time of optimistic rollups[1] can be less than 1 second, but the finality is *7 days* [2]. For ZK rollups[2] finality can be variable, depending on the security model: from 24 hours to only *1 hour* [3]. It means that unless the counterparty is willing to *accept the risk of not achieving finality*, the transfer can be recognized in as little as 12 seconds or as long as 15 minutes.

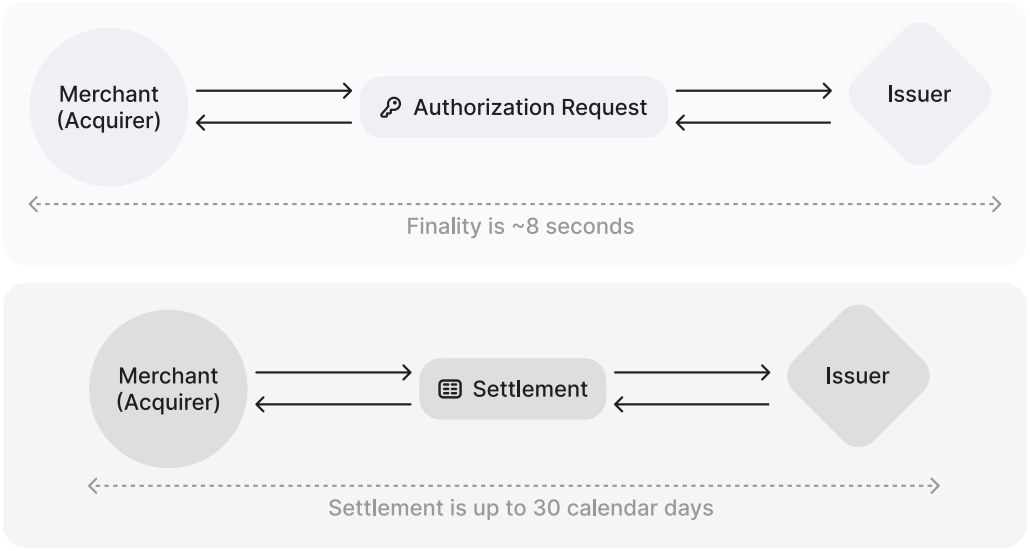

Figure 1: *Traditional payment systems achieve fast finality, whereas settlement is delayed (ranging from T+2 to T+30, where T is 1 calendar day). The core difference between a traditional payment and an onchain transaction is fundamentally a different approach between finality and settlement.*

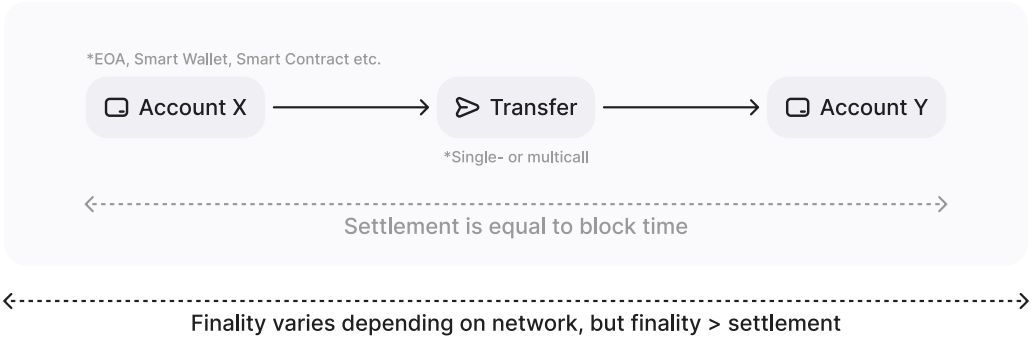

Figure 2: *Onchain transactions*[3] *achieve fast settlement at the speed of the block confirmation, whereas finality depends on the network security model.*

[1]Optimistic rollup — assumes all Layer 2 transactions are valid unless challenged and proven wrong by an honest network validator.

[2]ZK rollup - assumes all transactions are false until proven valid through Zero-Knowledge Proofs (ZKPs).

[3]This is a simple abstract example not including complexities arising due to presence of swap in the onchain transaction, MEV, chain reorgs, and, especially, crosschain interactions.

To make matters worse, *signing and sending* transactions is not a sufficient condition for it to be included in a block — it goes to mempool, which is a waiting buffer for transactions. Block space is limited, and a block can be filled by transactions having a higher gas price set. A block can be mined as a 'private' block with custom transactions included. Transactions can be reordered and even made to revert in several conditions. It makes the settlement process less reliable and providing transaction estimation time a much more complex task than simply relying on average block time interval[4].

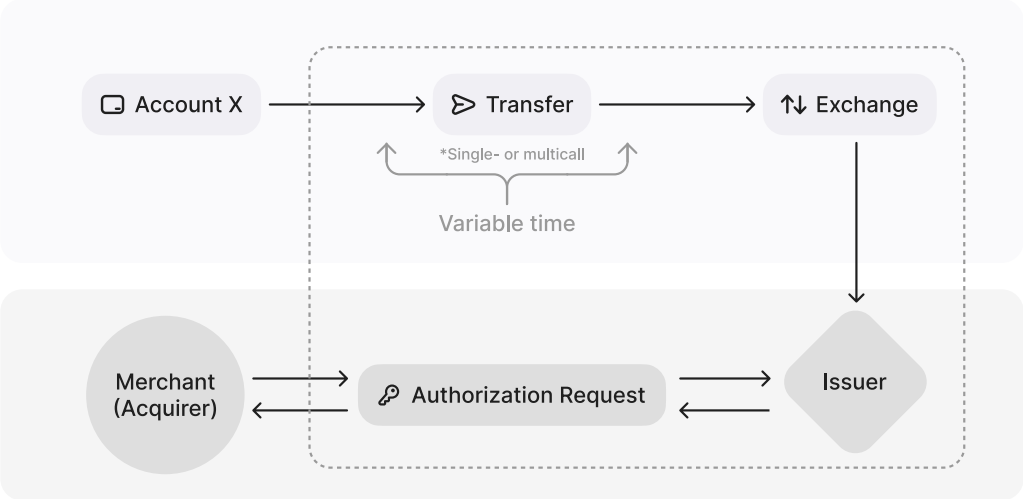

Figure 3: *Real-time processing (via PoS at a store, e.g.) of a transaction originating in crypto is very complicated. It requires dynamic finality assessment and, in most scenarios, will not satisfy payment network timing requirements.*

1.2 Limitations of existing payments

1.2.1 Limitations of fiat money transactions

Settlement of a payment instruction occurs in a legal tender currency accepted by the merchant that requires the issuer to operate with the acquirer via existing debit and credit mechanisms. Modules or Smart Contracts deployed on the EVM [4] and validated by a protocol inside Ethereum can contain enriched data. Such constructs are not possible to settle, thus excluding cryptoeconomic transactions from the global economy. There are three main downsides in the present organization of the on/off-ramping ecosystem:

- **Value leakage.** When exchanging a crypto asset for a fiat currency, there will always be an agent-broker commission for the service due to the accumulated costs of providing such service: licensing, compliance, technology, and liquidity costs.

[4]In practical scenarios, operations can include smart contract 'view' methods requests (via a blockchain RPC), asset price verification requests, conversion rate querying, compliance checks and other requests/response actions – all introduce additional latency.

- **Absence of generalized dApps support.** Smart contract logic and transfer of asset custody require two state changes: one from the completion of the smart contract and the second from the asset custody transfer to confirm off-ramping. This diversion of flow does not allow for generalized onchain logic support.
- **Higher trust model for consumers.** The current approach allows for off-chain services to create a tokenized representation of bonds, debt, and value, but it does not allow for onchain logic to be interpreted the other way around. Generally speaking, any of the financial institutions' middleware dependencies may be both a target of an attack and a compatibility blocker. Thus, access to a wider range of services and users is corrupted.

1.2.2 Limitations of programmable money transactions

Ultimately, EVM's goal is to give its users access to trustless computing and storage capacities. In the past, the only way to access trustless computing on Ethereum was for computation to be performed and verified by all nodes. With the expansion of proving techniques, more and more computation is happening in coprocessors, however, this introduces a main incompatibility with traditional payment networks.

- **Social discoordination.** Different networks have asynchronous and proprietary sequencing. Coprocessors and base layer blockchains process and order transactions in blocks in asynchronous asset ledgers and smart contracts. Block times and computational throughput, risk parameters, developer tooling, applications, and abstraction all differ, thus creating a fragmented global state. One of the implications is that CCIP[5] requires extensive time to confirm the finality of the transactions. It creates fast settlement but long finality, whereas traditional payment systems have, in general, fast finality, but longer settlement.
- **Liquidity fragmentation.** Different networks rely on so-called bridges to transfer the value (liquidity) from one network to another. A requirement to use a third-party protocol to reconcile the value between two service providers limits the business opportunity cost of serving users on different networks that offer different technology value effects.

[5]CCIP — is a global standard for crosschain communication allowing for arbitrary messaging and token transfers.

2 BRRR Architecture

2.1 Concepts

BRRR introduces two novel ideas, *segregated execution* and *open settlement layer*, which serve to facilitate the execution of onchain logic to any payment system.

Segregated execution — BRRR allows gasless[6] preauthorization requests to the account (wallet) specifying source chain, token, amount, and expiry time. Authorization is configurable – it can be valid for single use or multiple interactions, with specified time and amount constraints. By separating the authorization, settlement, and reconciliation, BRRR achieves gasless execution and reduces the number of mandatory authorizations from an account, providing better user experience and enabling financial operations automation.

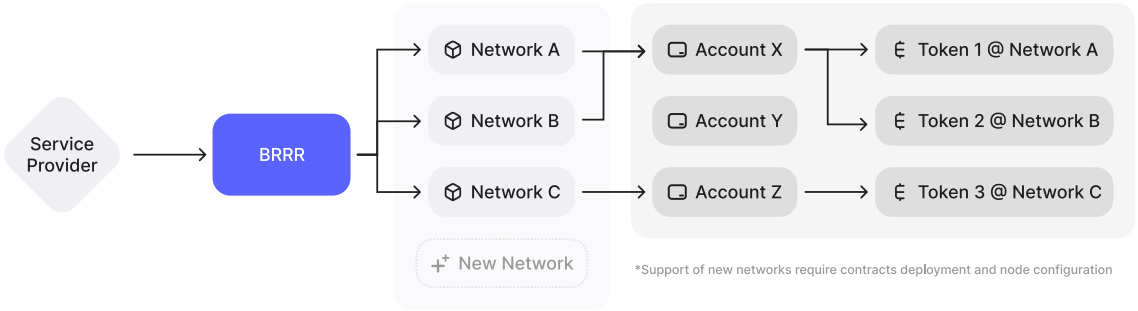

Figure 4: *A Service Provider can use BRRR to request a preauthorization with specific parameters. It also allows cross-service of the same users (accounts/wallets) by different service providers.*

Open settlement layer — upon the payment network transaction authorization request, a request emitted to BRRR triggers onchain settlement. A settlement operation can be a single transfer or a 'multicall' transaction with several steps. After the request, BRRR responds if the settlement is pending and to be completed (providing means of optimistic settlement) or if it has failed. Execution with BRRR removes the need for the account to trigger an action manually and for the service provider to maintain extensive onchain infrastructure (block scanning, transaction broadcasting, transaction construction, etc.). Furthermore, it allows a service provider to reconcile with another service provider post-settlement in an automated and trustless manner. An intent-based settlement allows service providers to decide on an offchain transaction authorization request before achieving onchain finality on the source network. In addition, it creates an opportunity for the upfront liquidity market to exist, allowing two service providers to reconcile at a desired rate between any two assets on any two BRRR enabled blockchains.

[6]Gasless transaction — is a signature mechanism allowing one to execute an action without submitting an onchain transaction, hence, not requiring a network fee. Please note, that native (gas) tokens on each specific network are not supported unless wrapped into an ERC20-compatible standard.

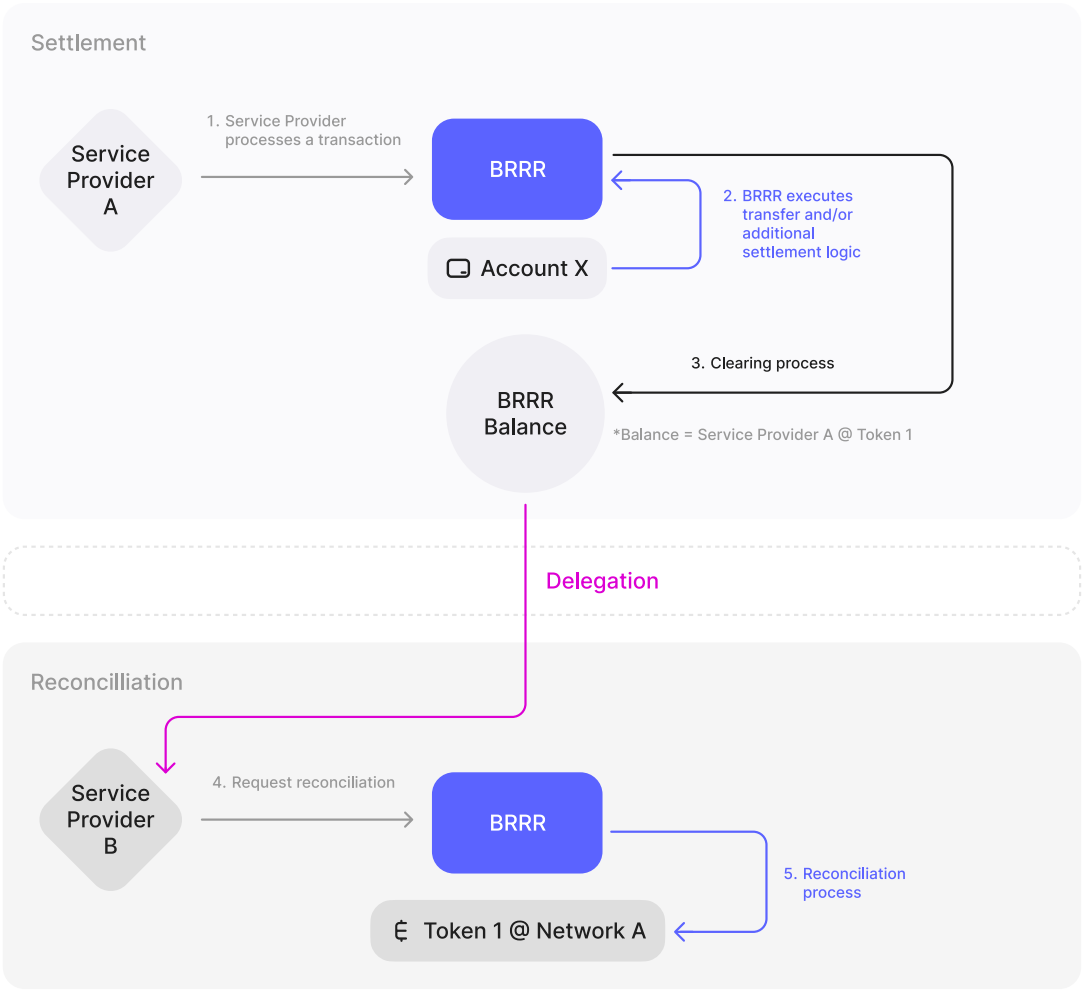

Figure 5: *BRRR Balance can be delegated from one Service Provider to another retaining the ability to claim assets.*

By combining these concepts, BRRR provides a chain-agnostic, gassless execution environment, less cumbersome user experience for accounts, and easier integration of onchain logic for service providers. BRRR serves as an open marketplace where onchain executors can provide access to offchain services and users. BRRR solves various problems in the current payment ecosystems:

- **Value accrual.** Smart Contracts can monetize their logic of transaction enrichment by participating in BRRR, receiving emission rewards for every processed transaction routed via said Smart Contracts. The open layer allows for competition on various networks across similar sectors of the economy leading to better and fairer fees. At the same time, service providers can offer their customers new features, services, and products powered by enriched transaction logic via smart contracts.

- **Capital cost.** Separating settlement and reconciliation creates a more efficient approach and removes the requirement to reconcile upon every transaction or a need to have address forwarding infrastructure[7], which does not allow for atomic multicall smart contract logic or transaction enrichment.
- **Complexity encapsulation.** Currently, the industry is composed of a growing number of blockchains (and not all are EVM-compatible), bridges, protocols, smart contracts, wallets, RPC providers, and others, resulting in the high complexity and costs of infrastructure setup, product development, and operation. BRRR absorbs multiple complexity layers (chain abstraction, gas management, transaction execution, and monitoring).
- **Standardization.** A unified adapter to the smart contract registry allows for universal access to transaction composition by service providers. If an interaction step requires this, a security deposit and slashing mechanism create an incentive and economic protection for the transaction layer participants.

2.2 Main components

BRRR consists of the following main components in the system (Fig. 6):

- **Smart Contracts.** There are 3 sets of smart contracts:
  - **Enabled chain** smart contracts: Token, Security Deposit, Service Provider registry. To allow native security deposit and registering on any supported network core contracts are an 'ecosystem ledger';
  - **Accounting chain** smart contracts: Balances registry, Operations, and Delegation ledgers. The accounting chain is an open and verifiable ledger of all operations in the system;
  - **Processing** smart contracts: (deployed on any network supported by BRRR): Transfer proxy, underlying Balance holder, Clearing and Reconciliation contracts.
- **Node software.** A configurable piece of software acts as a network peer, message processor for Service Provider, API endpoint, etc.
- **SDKs.** Lightweight libraries for popular languages to provide easy integration with BRRR API and contracts.
- **Explorer**, (auxiliary) Dashboards and other tools. Public Transaction Explorer provides operational transparency. Based on open architecture, there can be any number of public-based tools available.

[7] Address forwarding is a technique commonly used by Service Providers to segregate deposits by creating unique wallets either for each user, or for each transaction. Such approach requires extensive infrastructure for monitoring, and clearing deposits.

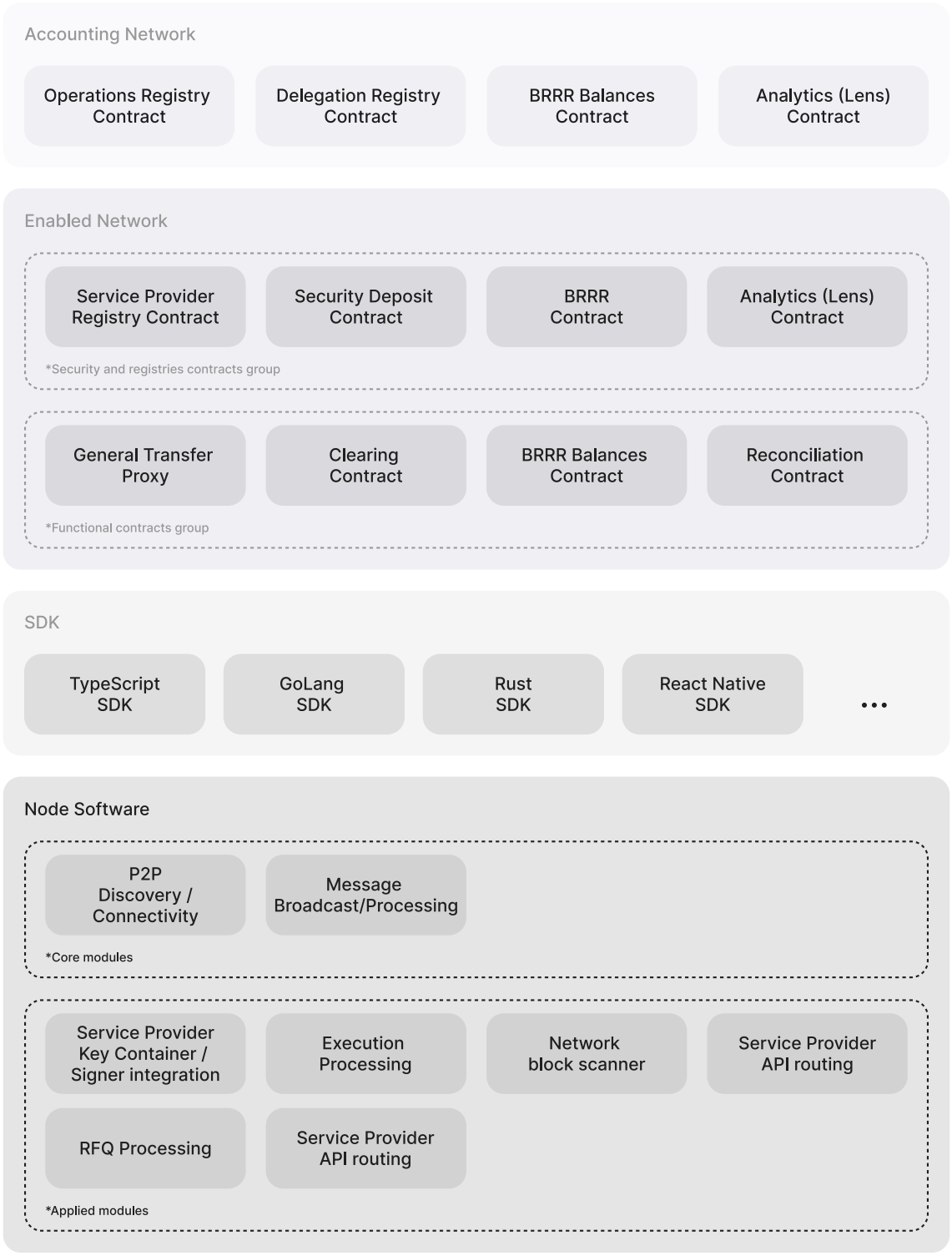

Figure 6: *High-level category view of main BRRR components. Node software is modular and configured to serve as an actor from a lightweight Service Provider peer, up to a public API/analytics endpoint.*

2.2.1 Blockchains

While remaining chain agnostic, BRRR uses an Accounting Chain for data persistence throughout its operation. It is required to keep a ledger for operations and accounting: Service Provider balance, settlement, and reconciliation records (Operations ledger), including peer-to-peer reassignment of BRRR Balance. Accounting chain records are publicly verifiable and are a source of truth for the nodes to work with.

2.2.2 Smart Contract Registry, Registration Fee, Security Deposit

To have minimal impact and requirements, a Security Deposit (SD) is a measure that is only required when no other means are available. A scenario where the SD is required implies that assets are transferred to a Service Provider upfront while waiting for the assets on the first leg of an operation to be retrieved back. An example of such an operation would be a bridge Service Provider.

The SD is verified beforehand the transaction (verification is signed). The equivalent portion of the SD is locked when an operation begins and unlocks upon successful completion. When an SP agrees to and signs such an operation, it must provide an SD valuation (allowing the SD to be fractionalized if agreed by both counterparties). If the SD value is too high, each counterparty may abort the operation and return an acceptable amount in the RFQ response.

If a reconciliation token is used to reconcile, the initial (maximum) SD value is automatically calculated by BRRR. However, BRRR deems an over-collateralization factor of 1.1x required by default.

Upon completion of the operation via destination contract routing, such transaction emits an appropriate event to unlock the SD portion in the accounting chain.

A service provider is required to register each smart contract participating in the reconciliation flow on one or more enabled chains. This process is fully automated. A one-time registration fee is required to emit the registration event. Upon registration, SP's Smart Contract will be listed in the registry, along with the required metadata for the contract.

2.2.2.1 Service Provider Smart Contract metadata

- Service Provider ID;
- Callback endpoint – where to receive HTTP callbacks when there is a change in operation status;
- Supported chain (implied, location determined by the registry shard);
- Broadcast ID;

- Interaction scope, one of:
  - Only when the interaction is requested with the specific smart contract;
  - Swap RFQ (same source and destination chain);
  - Crosschain ('bridge') RFQ (different source and destination chains).

2.2.2.2 RFQ request and response schema

All RFQ request and response schemas are standardized and can be implemented in one of two ways by a Service Provider:

- As 'view' methods interface in a smart contract (fully onchain);
- As a BRRR Node RFQ handler through the web endpoint provided.

An operation can include bridging. In the event of on-ramp operation, there are scenarios where a problem may arise if the end destination is a 3rd party SP operating via an unknown smart contract adapter.

2.2.2.3 Registering a Smart Contract in a registry

To register a smart contract, an SP is required to pay the registration fee. In certain scenarios, as described earlier, an SP is also required to provide a security deposit:

- Registration is required on every enabled chain where the Smart Contract is operating;
- The SD can be claimed back if the smart contract is delisted from the registry and is no longer operating.

2.2.2.4 Accounting chain smart contracts

- **BRRR Balances.** Records of tokens & amounts of an Account on a particular chain, or chain agnostic for reconciliation tokens. BRRR balances can be delegated to another Service Provider;
- **Operations ledger.** Maintains records of the clearing and reconciliation operations. In the case of composable transactions (e.g., a transaction that includes a token swap) containing a signature of the Service Provider and parameter information of such actions confirmed;
- **Delegations ledger.** Maintains trustless operation and history of delegation or exchanging BRRR balances between Service Providers;

2.2.3 Node software

A configurable open-source executable service that can run as a container (Docker, etc.) or in any on-premise configuration. Core modules include peer discovery through seed nodes and message broadcasting. Other applied modules can be set up by any actor or Service Provider depending on their functional requirements. Node configuration includes[8]:

- **Reconciliation tokens configuration example**:

```
"reconciliationTokens": [
  {
    "name": "Token X",
    "tokenId": "50d29e4e0be18" // Unique reconciliation token ID (broadcasted to Node configuration)
    "chains": [ // Networks where token is available and counted as unified token available for direct reconciliation
      "0": {
        "token": "0xA0b869...06eB48", // Token X address on Network A
        "amount": "14995123.051211" // Amount available for reconciliation
        "priceoracle": "0x123123...123@slot12" // Token price oracle on this network (if available)
      },
      "10": {
        "token": "0x0b2C63...97Ff85", // Token X address on Network B
        "amount": "1022252.142" // Amount available for reconciliation
      }, ...
    ], ...
  }, ...
]
```

- **Chain configurations** Networks can have different drivers (EVM-compatible chains share one driver, but have different configurations) [9]:

```
chainparams: {
  {
    "id": 0,
    "name": "Network A",
    "driver": "EVM", // Blockchains of same architecture are connected with same driver
    "configuration": {
      ...
      "gasTokenPriceOracle": "0x123123...123@slot12", // Native (for EVM Networks) token price oracle
    }
    "tokenmatch": {
      "": "0 x123 ...123" ,
      ...
    }
  }
  ...
}
```

[8]Examples are abstracted and simplified for readability. For example, different oracles may have different decimals and stall detection mechanics.

[9]Service Providers can configure their own RPC endpoints for every network.

Receiving and broadcasting configuration messages of crosschain tokens and network parameters applied after a 24-hour cooldown.

2.2.4 Lightweight node network

Even within a single network, computation-heavy operations are not possible entirely within the smart contract interaction. For instance, multihop swaps require querying liquidity sources, building optimal execution routes, and other computational tasks that are economically expensive to be fully onchain. One of the most critical aspects, and also one of the most complex, is the support of all networks, even network architectures.

This is the reason for having a lightweight node network, initially acting primarily as an integration adapter and providing transparency in system operations.

The more nodes and, respectively, more SPs register smart contracts, the more optimal the transition to multiparty computation using threshold signatures is. The further expansion allows enabling a quorum for signing operations that traditionally require trusted verifications.

3 Operating principles

3.1 Account asset clearing

Service providers have two options when querying for clearing of assets from a customer:

- Check that asset or operation is available, initiate BRRR execution, and make further processing on a callback when asset clearing is de-facto executed (finality is determined by the source network, as described earlier);
- Check that the asset or operation is available and process the transaction optimistically (instant response, synchronous reply). Depending on the requirement of the SP, such an option can be suitable for real-time or near-real-time settlement. What is assumed if optimistic execution is requested:
  - BRRR uses the data for the allowances provided by the owned account by monitoring blockchain events (it allows for the allowance and/or balance data to be updated in the background). It allows BRR to provide instantaneous if operation is possible in principle. Optimistic settlement may be exposed to a potential risk of malicious behavior by the SP customer of double spending, or causing a race condition of the transaction to be cleared.
  - BRRR responds with the confirmation of the beginning of optimistic settlement, and synchronosly performs the claiming process. Optimistic settlement prioritizes speed and can be used depending on the SP risk and other in-house policies.

Even if optimistic execution is requested, the SP can wait for the confirmation from BRRR of the transaction finality, or even wait for an arbitrary time interval determined by in-house policies.

Asset-clearing methods supported by BRRR:

- **Legacy**: Users can sign transactions on demand (an appropriate call data for the SP application is created by SDK). Such transfers can be processed as soon they enter the block by the SP. In such a method, a signature is not guaranteed, as well as, block number entry, gas price, and other parameters of the transaction, etc.);
- **Approval/Allowance**: Currently, the most common method of claiming funds on demand. The user approves the allowance of a token to the spender (unlike approval, allowance can be reused, deeming it a more efficient method);
- **Permit**: A signature for any ERC20 token supporting permit. It does not require any gas, support allowance protection, or single- or multi-operation (similar to Permit2). When the user signs the permit message, BRRR executes a call to execute `permit()` call for the requested token, resulting in allowance being set without any gas requirement or transaction from the user;

- **Permit2**[5]: A signature method for any ERC20 token. Permit2 allowance can be shared gaslessly across all smart contracts that utilize this method if the user has already provided such allowance previously. If the user provided an allowance when using Uniswap Smart Contracts, sharing the allowance to a different Smart Contract will not require an onchain transaction);
- **EIP-4337**[6] Account Abstraction allows the use of Paymasters[10] market on top of BRRR to sponsor or offer gasless transactions;
- (near future) **EIP-3074**[7] When `AUTH/AUTHCALL` becomes available, the SP will be able to cover gas costs (or other mechanics, like EIP-7702 **EIP-7702**[8])) on behalf of the user, and BRRR to abstract it away for both the account and the SP;

[10]Paymasters are smart contracts that enable flexible gas policies, allowing SPs to sponsor operations for their users, or accept gas fee payments in ERC20 token different to native gas token.

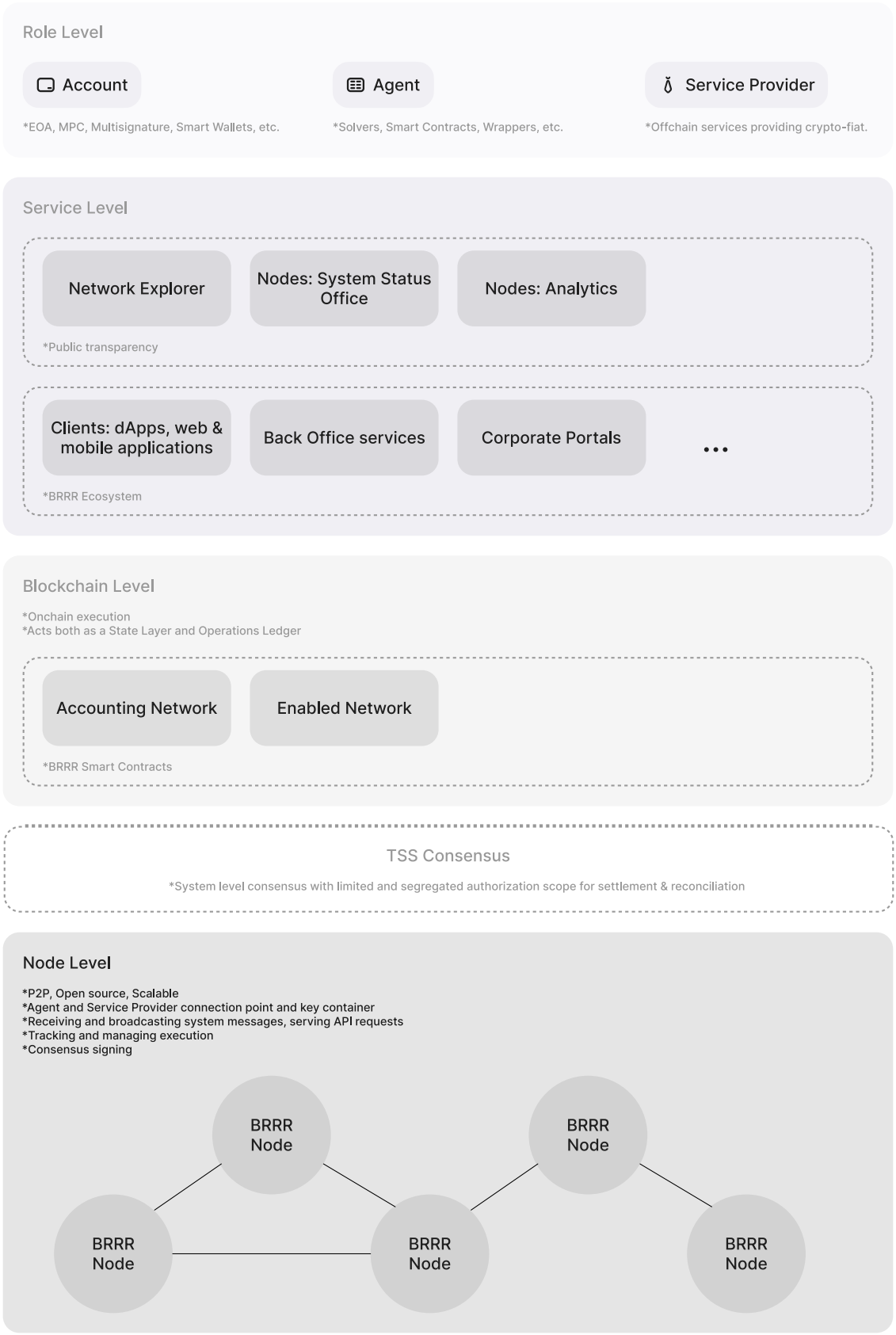

Figure 7: *A top-down view of the system execution layers.*

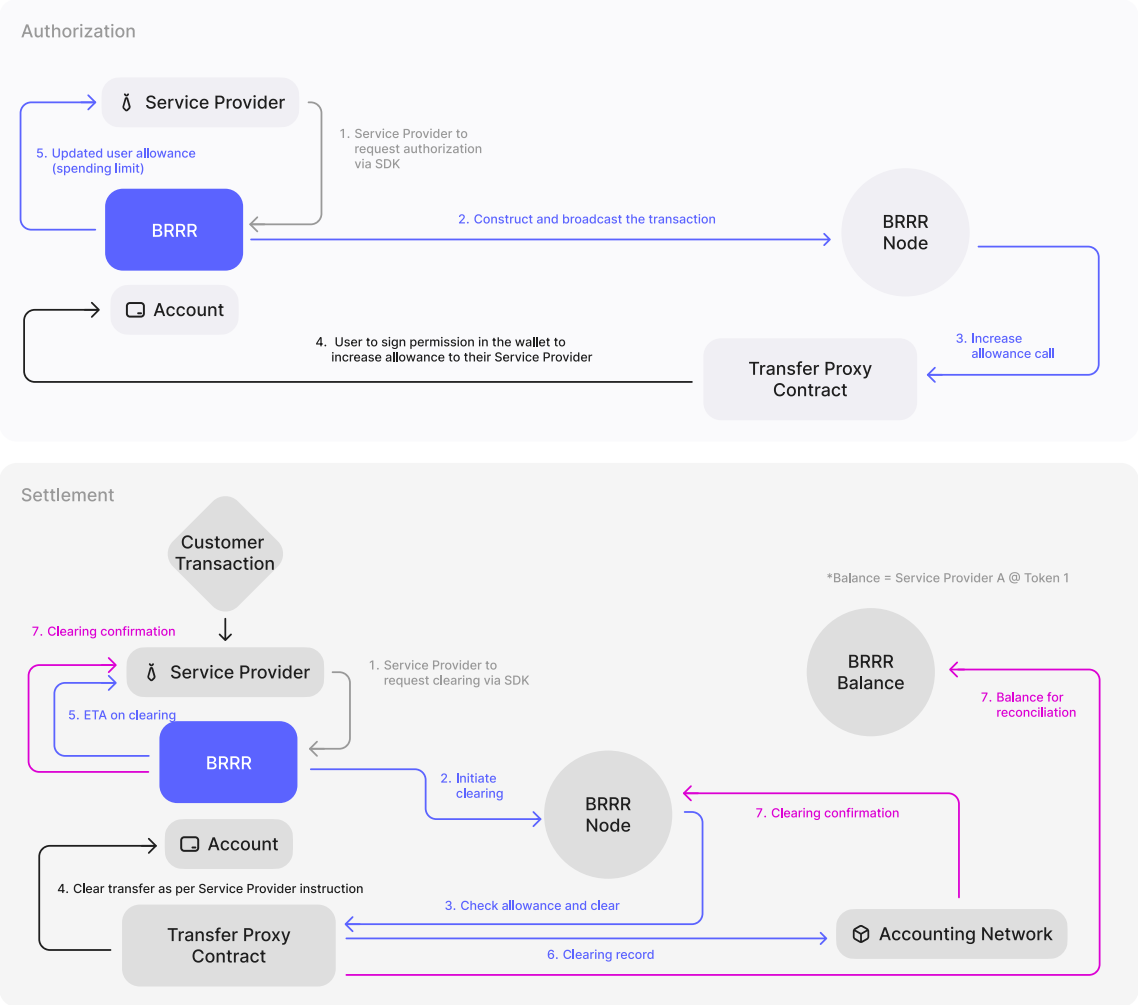

Figure 8: *End-2-End example of a SP using BRRR to provide real-time payment for the user paying with a crypto asset, while reconciling at a later stage.*

3.2 Multi-asset support

Token swap integration. A token swap can be integrated into the clearing transaction. An SP (if sufficient tokens are available on the user wallet) can issue an RFQ API call to swap a token during the clearing transaction. In such cases, BRRR broadcasts the RFQ and collects offers from participants (BRRR is not required to know the value of input or output tokens). The final route and the Swap Service Provider are selected based on the best `minAmountReceived` provided (aka best quote).

Swap interface could be standardized into three operations:

1. **Request for Quote.** RFQ is broadcasted via BRRR nodes. BRRR expects quotes from participating SPs in the Smart Contract Registry within the designed deadline. A response (depending on the type of request) must contain either `amount` of input token or `minAmount` of output token should be present:

```
{
  "inputAsset": {
    "chainId": "1", // Netwrok ID (string, as not necessarily only EVM chains)
    "token": "0x132475...153124" // Token address
    "amount": "3500.00" // Amount of input token that should be swapped
  },
  "outputAsset": { // Network assumed as the same, as swap must be packed in a clearing transaction
  "token": "0x789342...847593" // Token address
  "minAmount": "1200.00" // Amount of output token that should be guaranteed to be received
  }
}
```

2. **SP Confirmation.** The best minimum expected output amount quote is selected, and a request for exact call data is sent and processed. Such operation can be unified with p.1 if the Swap Service Provider supports it. The SP's node signs this operation to confirm acceptance of the terms provided. This information is recorded in node logs and the BRRR operations ledger on the accounting chain:

```
{
  "inputAsset": {
    "chainId": "1", // Network ID (string, as not necessarily only EVM chains)
    "token": "0x132475...153124" // Token address
    "maxAmount": "3412.21" // maximum amount of input token that would be swapped
  },
  "outputAsset": { // Network assumed as the same, as swap must be packed in clearing transaction
    "token": "0x789342...847593" // Token address
    "minAmount": "1200.00" // Minimum amount of output token that guaranteed to be received
  },
  "allowanceTarget": "0x111111...123123", // Allowance target (should be present in Service Provider registry)
  "calldata": "0xa1b712e021...092be", // Call data to execute the swap
  "maxGasSpent": "232341" // Maximum estimated gas spent (in swap chain gastoken)
}
```

3. **Execution.** A swap quote is packed in a single transaction after reclaiming a token from an account or during a reconciliation request by the SP. BRRR reclaims minAmount of the token, then checks and provides an allowance to swap contract, after that executes the swap call, and finally, checks that swap results are as expected. In the positive (expected) outcome, BRRR Balance is assigned to an SP for the exact amount of output (requested) token that was received.

The "Swap" term is used to keep things semantically simple. Various operations can also be used with the same interaction, for example, wrapping and unwrapping of tokens.

If the user provides an allowance to multiple tokens, the SP can clear specifying the exact token that should be used for the operation. The business logic depends on the in-house policy or the product. For example, the user may instruct the SP on the order in which the tokens should be used or the conditions when they should be used. For example:

- Prioritize one token over another;

- If a single token balance is insufficient to cover an entire transaction, use two different tokens for the settlement;
- Use specific tokens to clear only specific types of transactions (e.g. subscriptions, direct debit, etc.);

3.3 Execution flow

3.3.1 Simple Settlement

Preauthorization flow. An SP (the user-facing interface provided by the SP) uses BRRR SDK to set the allowance for the token (to BRRR transfer proxy contract) for the user account. For the allowance setting options, please refer to Section 3.1.

Settlement flow. An SP requests the settlement specifying the token, amount, and network (optional) via BRRR API. BRRR verifies the allowance set (preauthorization flow). Alternatively, such data is already updated via block scanners (Node service) with instantaneous response available, if an optimistic settlement is requested

API request `GetFunds(chain, token, amount)` – one of two responses is expected:

- Cannot clear (no allowance, not enough funds, etc.);
- Can clear and is in progress (also contains ETA on the clearing).

  BRRR initiates the clearing transaction on behalf of the Service Provider. After it is mined or confirmed into a block (timing depends on the source chain), the SP receives a message (or a Webhook, if callback URL was provided), signed by BRRR, that clearing is complete, and the transaction is recorded as BRRR Balance operation in accounting ledger contract.

The SP can query available BRRR balance via API (aggregating balances across all networks in a single call)[11]:

```
{
  "Token A": {
    "tokenId": "50d29e4e0be18" // Unique reconciliation token ID (broadcasted to Nodes configuration)
    "credit": "2523.101211" // Total amount of BRRR balance availble for the token
    "chains": { // Networks where this token is available and counted as a single token available for direct reconciliation
      "0": {
        "token": "0xA0b869...06eB48", // Token A address on Network A
        "amount": "14995123.051211" // Amount available for reconciliation
      },
      "10": {
        "token": "0x0b2C63...97Ff85", // Token A address on Network B
        "amount": "1022252.142" // Amount available for reconciliation
      }
    },
    ...
  },
  "Token B": { // Token outside of reconciliation tokens set
    "credit": "150.00", // Total amount of BRRR balance availble for the token on a specific network
    "chainId": "10", // Network where this token is accounted
    "token": "0x123123...123123", // Token address
  }
}
```

After receiving the BRRR balance, the Service Provider can assume funds are received. The next step depends on the SP's in-house business logic and policies. For example:

1. The SP can execute offchain operations (e.g. process an offchain payment), rely on the information provided by BRRR without any additional interaction, or reconcile funds when/if required.
2. BRRR balance can be aggregated across multiple transactions and reconciled in larger batches. BRRR balance can also be delegated to another SP in exchange for another BRRR balance (acting like a swap and secured by BRRR) or between counterparties at their discretion.
3. BRRR balance can be reconciled (underlying token claimed) on any specified enabled network (for reconciliation tokens only) if the BRRR balance is sufficient.

3.3.2 Reconciliation flow

Reconciliation is available anytime for any BRRR token balance recorded on the accounting ledger contract. There are several scenarios of how this can happen within the mechanics of the system.

If a token is not on the reconciliation tokens list (not crosschain), then BRRR must always have a sufficient amount corresponding to the balance issued on the

[11]The SP can query available balances for a customer as well, most idempotent queries are skipped for example to remain concise.

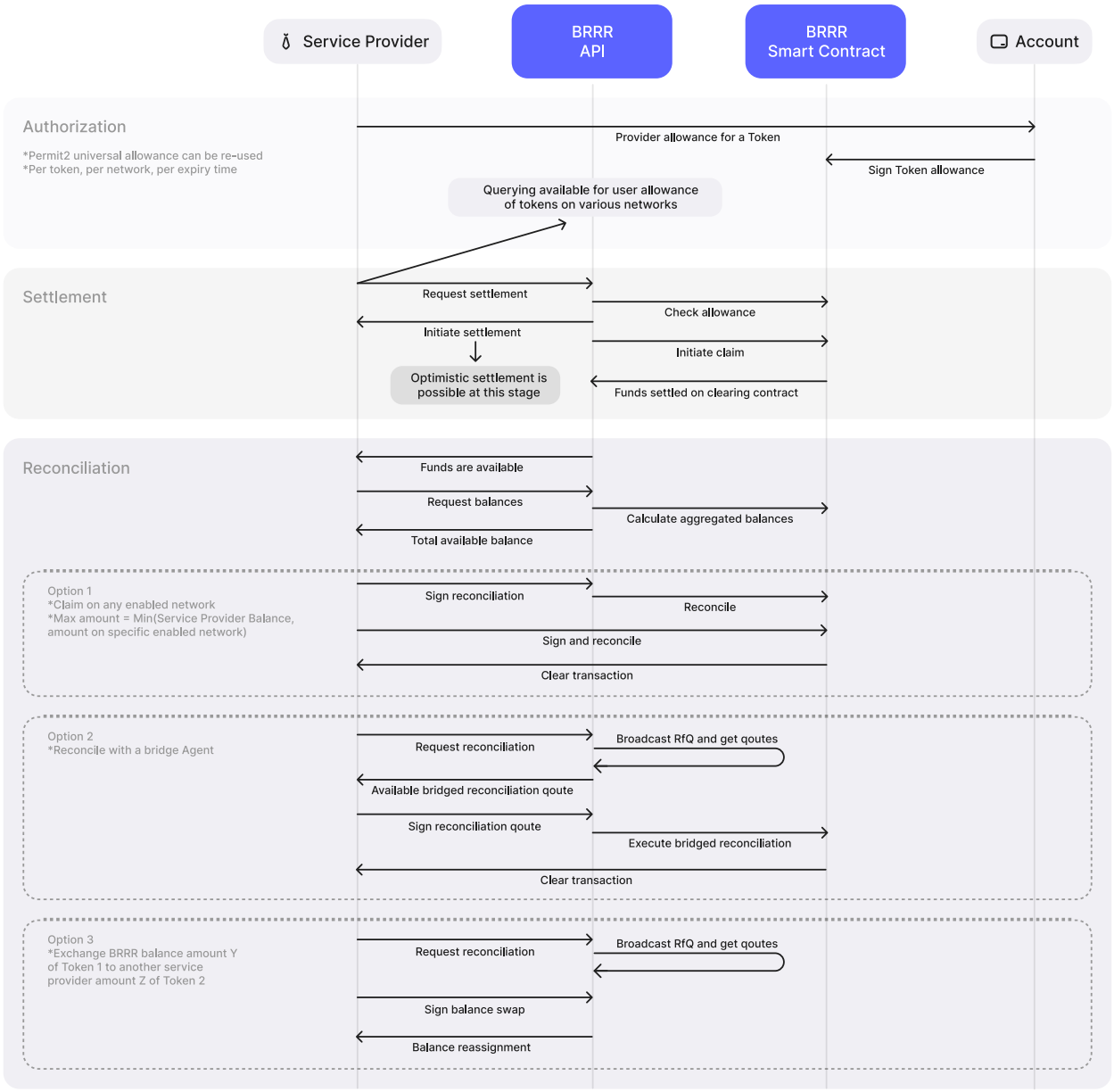

Figure 9: *A simplified example of authorization and settlement flows when using BRRR. Please note that by significantly reducing interactions required from the user (steps after 'Preauthorization flow' can repeat multiple times), the user experience is greatly improved.*

accounting network (as settlement and reconciliation balances must match). In such cases:

1. Service Provider signs a claim message that allows to claim a certain amount of token on a target chain.
2. Service Provider executes a transaction to receive tokens on the supplied verified address or makes a request to BRRR to execute a transaction on his behalf (not requiring any gas tokens from SP or wallet infrastructure) and receives a message when funds are deposited on the provided address.

If a token is a reconciliation token (crosschain), there are several possible scenarios:

- If there is enough liquidity available on a desired chain, then the SP can claim the token the same way as a non-reconciliation token;
- If there is no sufficiently available liquidity on the desired chain, then BRRR sends an RFQ message to registered (Bridge) SPs to provide liquidity in exchange for a BRRR balance for the specific operation;
  - Integrated Bridge Provider. Upon the signature of the preferred quote by the Service Provider, the BRRR balance is reassigned, after the confirmation to the Bridge SP.
  - Adapter to external bridge. Upon the signature of the preferred quote by the Service Provider, BRRR executes a bridging operation, providing an underlying token to the bridge to execute the operation.
  - Direct BRRR balance exchange. Two SPs can exchange balances between themselves by providing signatures through BRRR. This operation is not limited to bridges or swaps, but can include any other mechanics, depending on the SPs' business models);

Segregated execution allows abstract bridging (with related latency and difficulty) from the customer experience. Reconciliation in the open settlement layer provides better gas efficiency of $O(1)$ instead of $O(n_{txs})$ through batched execution and allows Service Providers to build finance logistics onchain trustlessly with other Service Providers.

3.4 Cross-chain support and composable intents

Besides the settlement and reconciliation scenarios described in the previous section, BRRR is designed to handle more complex intents. For example (Fig. 10), bridging with the call to a particular SP's smart contract method can be included in the settlement operation. Such a flow can be specified by the SP's API call to the BRRR node to construct the execution flow. Please note that for any execution path step that includes interaction with smart contracts from other SPs, an RFQ is issued and (in addition to automatic signature by participating counterparts) signed by the SP on behalf of which execution flow is constructed.

Settlement and *Reconciliation* are semantic definitions of **system operation** patterns projected to have applied usage and value. But such Operations (or operation 'types') are not hard coded and can be composed of arbitrary steps. With each verified, executed, and publicly written on the Accounting Chain operations ledger.

Steps are defined by having input and output tokens, including amounts, specified. If input and output networks are different, this operation step ('bridging') is considered to have an offchain execution component. It separates execution into multiple transactions on different BRRR-enabled chains. Taking into consideration the SD lock and unlock flow following the requirements signed by both counterparties.

The bridge interface is standardized: the main distinction of a 'bridge' operation is that source and destination chains are different, resulting in an operation that

cannot be executed in a single transaction on a single chain. Worth mentioning that as a part of the bridge operation, a swap can also be present.

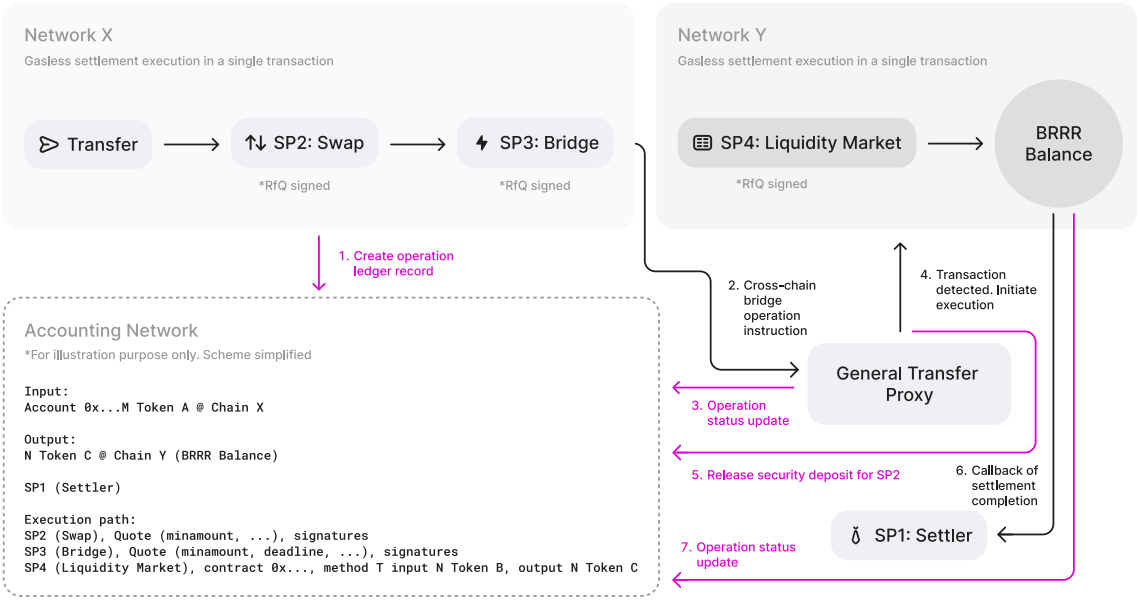

Figure 10: *If required, settlement requests can be constructed to include a particular SP smart contract call and cross-chain interactions. For example: to fetch a 'source' asset from an Account and provide it as collateral on another chain, borrowing the required 'destination' asset.*

3.5 Delegation

Input and output parameters of an **Operation Step** can also be a **BRRR balance** of a token. It allows two SPs to exchange (transfer) their BRRR balance on the terms counterparts agree to, enabling a settlement and clearing layer. Economic incentives to interact using BRRR balance can vary. For example, SPs (acting as bridges or LPs) can charge a fee to provide instant liquidity for the BRRR balance. On the other side, technical incentives are:

- **Chain-agnostic** – BRRR balance records are moved on the Accounting chain;
- **Gasless** – gas fee sponsored by the executing contract;
- **Fast and atomic** – balance changes are always in a single transaction;
- **Trustless** – works through RFQs and Operation signatures;
- **API-first** – interaction through API calls, executed by the BRRR smart contract;
- **Transparent and verifiable** – via the Operations or Accounting ledger.

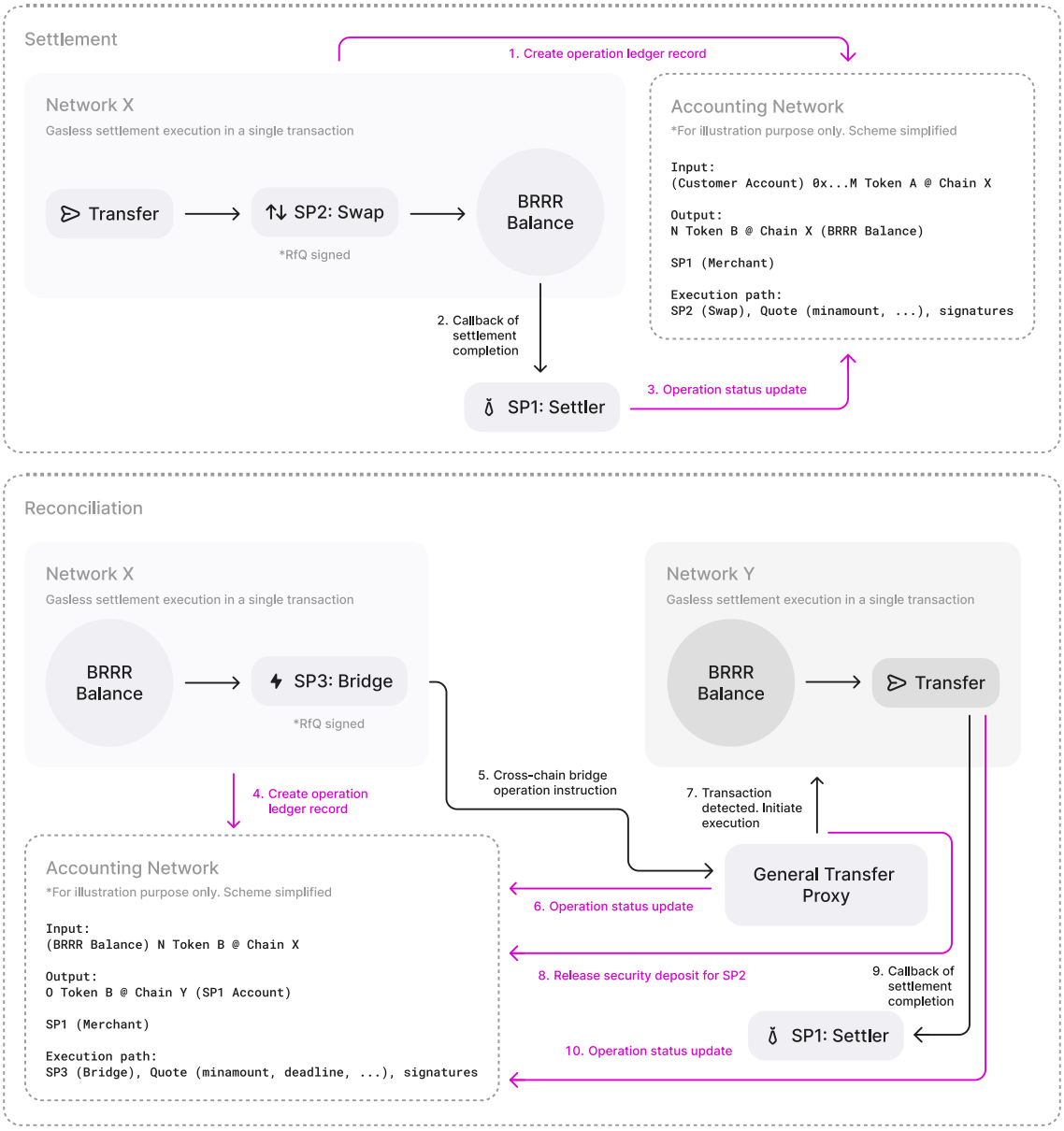

Figure 11: *"Optimal" scenario: settlement on BRRR-enabled chain to BRRR balance, while reconciliation is realized via a separate execution process. When performing a crosschain settlement, network complexity is abstracted from the Account (user). Segregated execution reduces the risk for an SP to operate optimistically and allows to achieve faster settlement finality.*

3.6 Prices, oracles and operations value calculation

Tracking asset prices (especially for transactional purposes) is not trivial. It is even more difficult with multiple networks and multiple network architectures.

Swap quoting (or any Operation Step) does not require BRRR to know the price of assets being swapped. Output parameters are the amount of output tokens and the minimum expected amount. If the amount received is larger than required (positive slippage), it is accounted to the SP BRRR Balance. The SP can further decide what to do: to credit the user or to keep the interchange earned. At the same time, if the required output amount of tokens is not met, BRRR reverts the Operation

execution.

BRRR is designed to minimize internal requirements of knowing token prices on multiple networks:

- **Native or Gas token** prices on enabled chains are defined by several supported schemas (support is determined by a blockchain driver), such as onchain oracle address and storage slot;
- **Reconciliation token** prices (optional) – to reduce the SD requirements for crosschain operations involving a reconciliation token;

It removes the price tracking requirement for every asset while allowing the use of BRRR as a crosschain gas token for executing transactions. Operation execution expenses (gas) are quoted in BRRR at the moment of execution and charged in batches from the SP's SD.

3.7 Disputes

Every payment contractual obligation is always triggered by an event or state change. There are also two possible outcomes: success or failure. In the event of a failure, or contract breach, typically there is a dispute process. Financial institutions have always served as arbitrators between two counterparties: user and merchant, or underlying users of an issuer and acquirer. For this reason, each party is always monitoring if the underlying state triggered an obligation owed to them. However, the verification process of a third party (arbitrator or "court") has always been a costly, asynchronous, and not atomic process.

Such a process is even more complicated when transacting between two various transaction types: crypto (onchain) transaction and traditional payment. As earlier described, the difference between settlement and finality performance does not allow for an atomic dispute process. When there are two counterparties to contract: Alice and Bob, each has obligations — xA and xB — that they have agreed to perform. If the contract terms are met, Alice receives $vA(xB) - cA(xA)$ while Bob receives $vB(xA) - cB(xB)$. In each case, if an obligation is not performed, it costs the performer nothing and reduces the value received by the other to 0. For instance, let's examine Alice and her incentives to undertake her contractual obligations. If Alice chooses to perform her obligations, then Bob faces a choice as to whether to confirm that performance or not. As Bob does not know Alice's decision, it allows Bob to challenge Alice's performance and require her to produce hard evidence of the contract execution. If she has performed, that is possible and she incurs z, and the matter is settled. If not, Bob can only dispute the failure of the contractual obligation by Alice. If there is no dispute resolution process, Alice has no incentive to prove otherwise, Bob has no incentive to confirm performance and Alice has no incentive to perform her obligations. Thus, the contract agreement would be infeasible.

On the other hand, suppose that if Bob is successful in his dispute, he receives d while Alice pays d. In this case, so long as $z < d$, Alice is incentivized to prove otherwise, and so long as $cA(xA) + z < d$, Alice is incentivized to honor the contractual agreement. In addition, Bob will gain $d - m$ if he confirms non-performance and $-m$ if he confirms performance. It means that Bob's most efficient strategy is to monitor, p, to be such that Alice has an incentive to perform: that is, so that $cA(xA) < p(d - z)$. Thus, Bob's expected costs of monitoring would be pm which is at least $mcA(xA)/(d - z)$. These costs represent a weight reduction of the set of contracts that would otherwise be feasible. At the same time, if z is too high, regardless of Bob's strategy the contractual execution is not feasible. It means that the contract will only be feasible if $d > z$ and $vA(xB) + vB(xA) > (cA(xA) + cB(xB))(d - z + m)/(d - z)$.

We've determined that for the contract disputing process, and thus the security model of such a contract (transaction) to be efficient, the cost of observability (m), and cost of verification (z) should decline over time.

3.7.1 Slashing

In a scenario when a desired output or a condition of an operation is not met, an SP's portion of the SD is slashed. For example, when an SP performs a bridging operation and is required to get tokens upfront. Such an operation is not atomical and cannot be a single transaction. To make it capital efficient, only an agreed portion of the SD of an SP making such an Operation Step is locked until the expected output of such a step is scanned by BRRR Node.

A positive or negative outcome of such Operation Step can always be formally estimated and publicly verifiable, as a condition is in the form of:

```
Minimum of Token X @ Chain Y deadline Timestamp Z
```

- If this condition is met (the asset appears in the transaction in the block with a timestamp earlier than the deadline, BRRR will release the lock on the SD of the corresponding SP.
- If this condition is not met before the deadline, BRRR will fail the Operation, slash the executing SP, allocating the funds to an impacted SP as a BRRR balance. In the event of late delivery (beyond the agreed deadline), the Operation will not be processed, and a subsequent dispute is to be initiated.

4 Emission

To ensure long-term alignment of incentives for the participating SPs and users in the system, the emission schedule is designed to dynamically react to the changes in the usage of the system. The emission schedule is bound by two parameters: processed transaction (activity) and elapsed time (operational time).

Token parameters	
Maximum supply, S_{max}	[redacted]
Transactions target, T_{tr}	1 000 000 000 transactions
Incentives lifespan, L	10 years (120 months)
Percentage of supply for incentives, P_s	[redacted]%
Month-to-month decrease percent, D_{MoM}	5.6%

Amount of tokens to mint as incentives is: $S_{incv} = S_{max} * P_s =$ [redacted].
Mint target portion for month i compared to month 1 (assumed as 1.0) is: $P_i = (\frac{100-D_{MoM}}{100})^{i-1}$, total portions: $\sum_{i=1}^{120} P_i$, thus, *target* amount of tokens to mint in a particular month is:

$$M_i^{target} = S_{incv} * P_i / \sum_{j=1}^{120} (\frac{100 - D_{MoM}}{100})^{j-1}.$$

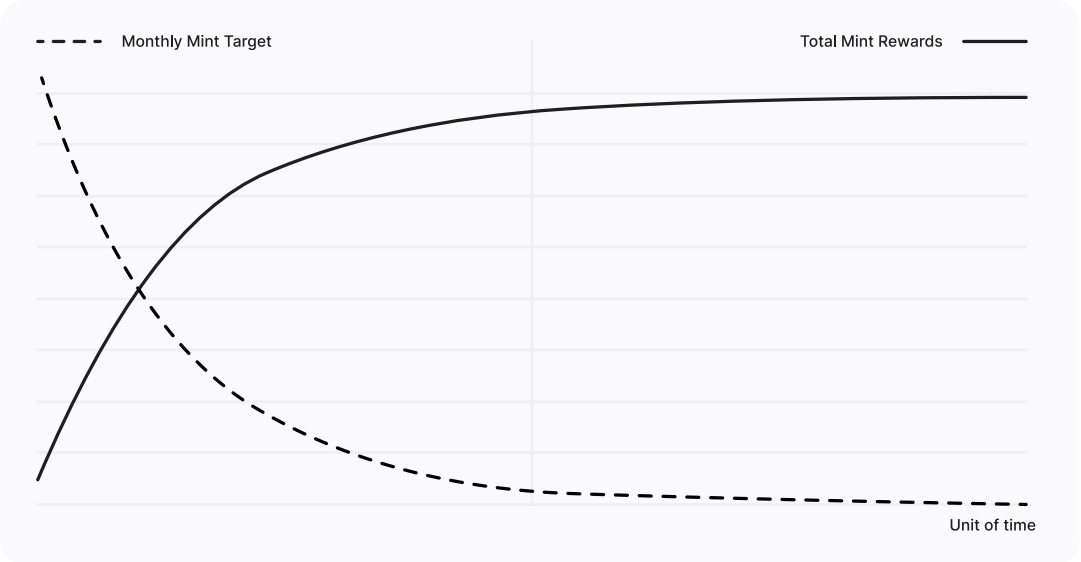

Figure 12: *Overall BRRR token minting rewards for system operation, reaching designated fixed upper limit.*

The target amount means that the mint (emission) rewards depend on the key metric – transaction activity. BRRR emission schedule is designed to target the first one billion transactions over the first ten operation years. The first tokens to be distributed in the first epoch mark (set) the base target. In every following epoch if the transaction target is not reached – fewer tokens are distributed. It allows for more tokens to be minted for the same amount of transactions in the following epoch. The idea is similar to the Bitcoin network's difficulty adjustment process[12]. However, it is not the block time that is tracked as a constant but an emission curve target. A parameter $D_{MoM} = 5.6\%$ value roughly results in 1/2 of the monthly token minting target after the first 12 months (similar to the so-called "the halving").

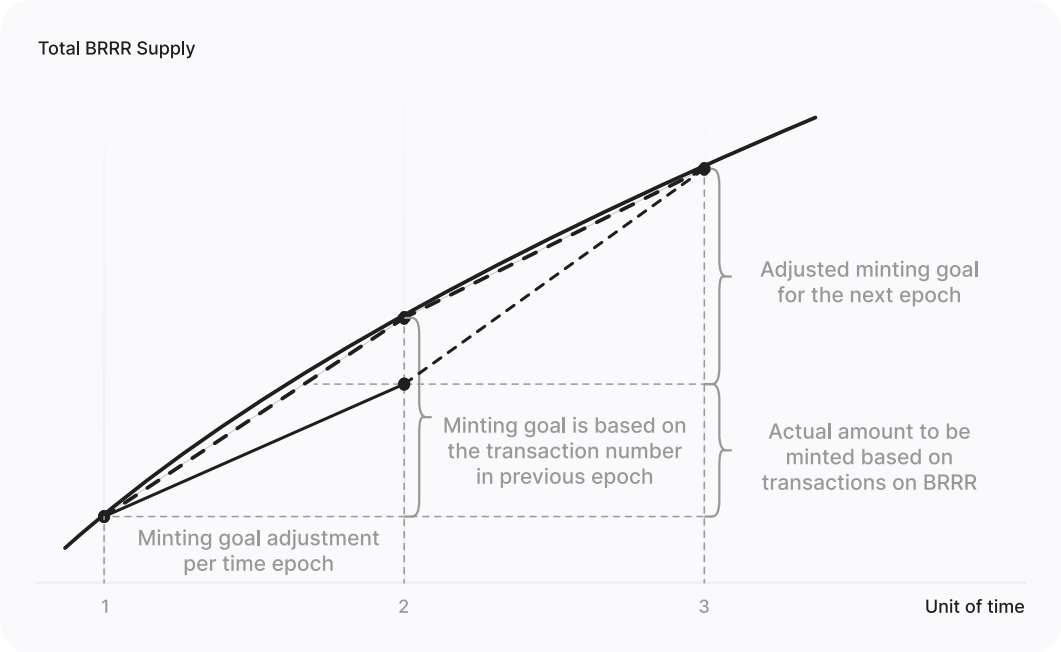

Figure 13: *Adjustment of the emission curve.*

5 New cryptoeconomic layer

5.1 Business models with BRRR

BRRR opens opportunities also for licensed payment service providers and participants of building on top of and in conjunction with the BRRR protocol:

- **Cash and salary advance** – traditionally, only employers registered with the Employer Salary Advance Scheme (ESAS) can offer a salary advance option. Service providers built on the BRRR can extend the scheme offer to a wider range of employers using the intent-based settlement powered by the BRRR security model.
- **Real-time salary payouts** – converging BRRR intent-based settlement service providers can settle in real-time payments routed via streaming protocols. It creates a new time-based value model, where instead of bi-weekly or monthly payments, employees can be paid per chosen unit of time, for example, every second. Figure 3 (Service provider offering upfront liquidity due to a settlement guarantee per unit of time).
- **Onchain FX marketplace** – while traditional forex markets are closed during nonbusiness hours, various brokers may offer after-market hours trading. Onchain forex markets can offer 24/7/365 access to the currency exchange, creating an even more efficient reconciliation between service providers on BRRR (if included in the BRRR onchain registry). In addition, service providers may choose to atomically avoid exposure to specific currencies while offering more options to their customers.
- **Cross-border remittance for expats** – with stablecoins offering a path to cheaper and faster settlement, the BRRR reconciliation system allows for the settlement of cross-border transactions between service providers via delegated access, offering faster liquidity access. Figure 4 (Two service providers can offer services to their respected customers by having access to the unified onchain settlement layer).
- **DeFi savings accounts** – staking, restaking, governance incentives, maker fees, and other onchain primitives in smart contracts can be included in the BRRR Smart Contract registry allowing service providers access to the most cost-effective, and yield-effective options to their users natively. In addition, opening new multicall logic opportunities, for example, separating principal and yield amounts to cover payment instructions.
- **Tokenized KYC and KYT** – the network effect of tokenized Know Your Customer (KYC) and Know Your Transaction (KYT) is increased by $2^n x$ as per Metcalfe's Law, where N is the number of participating service providers, and X is the number of unique users using at least one service provider on the network. Services offering tokenized KYC and KYT can further increase the network effect, by allowing various service providers to reduce friction during the onboarding of an existing user on BRRR.

- **RWA marketplace** – tokenization of real-world assets allows the creation of new collateral-based experiences, where tokenization of real estate collateral, or debt obligations can create new access layers to yield opportunities. Service providers can provide competitive rates to mortgages, and commercial credit lines, and with debt tokenization unlocking programmable paths to yield. We foresee services created on top of BRRR to allow tokenization of the real estate, unlocking liquidity onchain to be utilized in the yield-generating assets serving as additional income or inflation hedge.

5.2 Multi-call/composite transactions

BRRR supports atomic composition of multiple method calls within one transaction, verifying each Step (or reverting), or even more complex flows spanning more than one transaction in one Operation. That allows integration of complex interactions:

- **Swaps** – Automated Market Makers (AMM) is a core onchain primitive, allowing for the decentralized exchange of various assets. Integration of various solvers AMM, RFQ, and Dark Pools, into the BRRR smart contracts registry, opens opportunities for service providers to settle in a token different from the token used by users.
- **(Un)Wrap** – functions in smart contracts can create more modularity, and, hence, more competitive prices being offered to users. Being composable with various vault token standards (e.g. ERC-4626) allows LST (Liquid Staking Derivatives) and LRT (Liquid Restaking Derivatives) to be supported.
- **Lend and Spend** – peer-to-peer lending protocols allow users to collateralize and further borrow assets against the collateral. In certain events where the collateralized asset is bearing yield, with the yield being higher than the interest on the borrowed asset, lend and spend can become self-repaying[12].
- **Cashback** – real-time paid cashback in the user's desired token to an external self-custody wallet via the settle function of BRRR.
- **Pre-confirmations [9] and upfront liquidity** – while there are multiple strategies for reaching universal single-block or even pre-block confirmation finality, it is still not possible to guarantee a stable market for block proposers, validators, and node operators. The BRRR model allows for the creation of the single-block and pre-block confirmation marketplace for solvers. Figure 10 (External provider of upfront liquidity market). In the cases where liquidity needs to be provided atomically, an external solver, if added via reconciliation delegation by the service provider, can offer liquidity upfront, even before the finality on the source chain is reached [10].

[12]Self-repaying loan is the term popularized by Alchemix. Such a feature allows users to unlock liquidity without liquidating an original asset.

6 Terms / Glossary

Account — a tool, software, or entity that represents assets the user is holding. It can be an address (EOA), a smart contract (e.g. multisig wallet), a Multi-Party Computation (MPC) wallet, or an Account Abstraction (AA) wallet.

BRRR Balance — a record on the Accounting chain representing a token & amount in the system to an Account (a registered Service Provider). BRRR balances can be exchanged (swapped) between SPs or reconciled for the underlying token (chain agnostic if a reconciliation token).

Settlement — operation of clearing user funds by the Service Provider.

Optimistic settlement — a clearing flow in which the SP executes via BRRR a task to clear user funds but requires a near-to-instant confirmation or absence of such. Such a method allows us to have a real-time (near-instant) expected response with actual clearing taking place synchronously in the background.

Reconciliation ("Crosschain") tokens — tokens for which BRRR accounts a level of conformity across one or more chains and related price (defined through price oracles in most scenarios). For Reconciliation tokens BRRR balance is chain agnostic and SP in certain scenarios should consider optimistic clearing/execution when targeting for minimal finality time possible.

Service Provider (SP) — an account hosting at least one BRRR Node, executing at least one System Operation step, or handling, executing, or providing service to end-user Accounts.

Operation Step — a logical part of the execution flow. It is defined by input and output tokens, and amounts (can also be defined as minimum or maximum amounts provided and expected). In addition, it can also have a particular registered smart contract method call specified. Operation Step is signed by both the executing SP and the SP initiating an operation.

System Operation (Operation) — a defined execution flow that includes one or more **Steps** (starting or ending step could be 'offchain'). Usually involves an Account and one or more Service Providers. The operation has a structured definition. It is publicly stored and is verifiable on the Accounting Chain. Operation be unsigned (entered in Draft status) or signed by counterparties (Account, Service Providers) to be executed on BRRR.

7 References

[1] Single slot finality
https://ethereum.org/en/roadmap/single-slot-finality/

[2] Why is the Optimistic Rollup challenge period 7 days
https://kelvinfichter.com/pages/thoughts/challenge-periods/

[3] Finality on zkSync Era
https://docs.zksync.io/zk-stack/concepts/finality.html

[4] Ethereum.org EVM Documentation
https://ethereum.org/en/developers/docs/evm/

[5] permit2 GitHub
https://github.com/Uniswap/permit2

[6] Ethereum EIP-4337
https://github.com/ethereum/ercs/blob/master/ERCS/erc-4337.md

[7] Ethereum EIP-3074
https://github.com/ethereum/EIPs/blob/master/EIPS/eip-3074.md

[8] Ethereum EIP-7702
https://github.com/ethereum/EIPs/blob/master/EIPS/eip-7702.md

[9] Based preconfirmations
https://ethresear.ch/t/based-preconfirmations/17353

[10] Intents Architecture in Across
https://docs.across.to/concepts/intents-architecture-in-across
(UMA Optimistic Oracle https://uma.xyz/)

[11] Uniswap V3 Oracle
https://docs.uniswap.org/concepts/protocol/oracle

[12] Bitcoin: A Peer-to-Peer Electronic Cash System
https://bitcoin.org/bitcoin.pdf