TX Mesh: A General-Purpose Mesh Network Facilitating Off-Chain Data Flow and Service Logic

TX Labs, Version 1.2

December 08, 2021

Abstract

Ever since the inception of Ethereum in 2015, a rich tapestry of alternative blockchains has expanded creating the Web3 ecosystem. While each is different in its respective focus and advantages, the overarching purpose of these distributed networks is to further the adoption of on-chain, decentralized applications across industry sub-genres like DeFi, non-fungible tokens (NFTs), GameFi, etc. The technical infrastructure of these various blockchains, while publicly verifiable and open-source, are designed as closed ecosystems. As such, blockchain technology alone is unable to support multi-layered services that require resources outside of the blockchain ledger. These applications are often referred to as off-chain applications.

Building an off-chain application often requires developers to create and maintain an independent node network. This results in a range of inefficiencies. At the time of this writing, there is still no general purpose platform for building and hosting off-chain applications similar to how Ethereum helped enable its ever-expansive world of on-chain applications.

To solve this problem, we propose TX Mesh: a general-purpose mesh network to help facilitate off-chain data flow and services. TX Mesh functions as a decentralized, blockchain-agnostic network with service-centric, application-wide local consensus to help coordinate transactions off-chain. Business logic can be deployed by service/application providers to the TX Mesh nodes that execute them via Procedures, which are smart-contract-like code that can be deployed to the TX Mesh network.

To illustrate the dynamics between developers and the TX Mesh network, we will introduce an off-chain token swap service application. The application will provide native cross-chain support to directly facilitate transactions between on-chain arbitrageurs and retail users.

As the TX Mesh network grows and continues to host an ecosystem of services and applications, we will also create a SaaS service marketplace that enables interoperability between Web2 and Web3 networks.

1 Industry Background and Pain Point

For blockchain networks, transaction data on a blockchain can only come from:

1. Input when a transaction is initiated.
2. The state of the network at a given block, as illustrated in Figure 1.

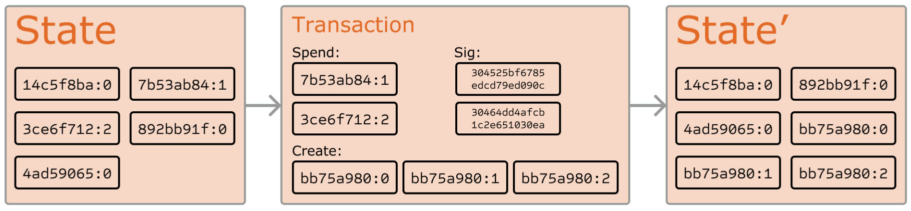

Figure 1: Blockchain as a State Transition System, *Ethereum White Paper*

Therefore, from a *data source perspective*, blockchain is a centralized transaction processing system. This means that the blockchain, as a standalone technology, does not offer the tools or infrastructure needed for building off-chain applications that require data and resources outside of the ledger. Example applications can include:

1. **Oracle Services** which help transfer and link data in and out of a blockchain.
2. **Cross-chain Services** which help enable interoperability between different blockchains.
3. **Scalability Solutions** such as roll-ups, state channels, etc.
4. **Utility Solutions** such as private computation, storage, data streaming, etc.
5. **Any Web3 application that wishes to benefit from Web2 services**, such as prediction markets and synthetic asset platforms. For instance, Web3 applications utilizing existing microservice providers on AWS marketplace to acquire related data.

However, projects and developers that are currently building off-chain applications are required to create entire node infrastructures independently and maintain them constantly. This is a significant obstacle: in order for off-chain applications to establish their business logic, they must first invest in the technological workload required to build a decentralized node infrastructure. Further, this investment must be repeated by every single off-chain application.

Currently, there is still no general-purpose platform that can help developers streamline the creation and support of off-chain applications. This issue is only further exemplified when we contrast it with the ever-expansive state of Ethereum and various L1 & L2 solutions, thanks to their ability to help developers create and host a disparate range of on-chain applications.

Off-chain applications have never been given the chance to prosper compared to on-chain applications. Yet, they are of no less importance in enhancing the functionality of Web3 as on-chain applications. In fact, off- and on-chan applications are complementary and essential to each other. Off-chain applications offer meaningful solutions to Web3's costs and resource constraints.

As such, the absence of a general-purpose platform not only inhibits the overall growth of off-chain applications, but also further highlights the inefficiency and outstanding waste of resources in Web3 thus far.

2 Project Overview

TX Mesh is a general-purpose, lightweight, off-chain mesh network[1] with composable tools to coordinate data flow and services. TX Mesh will enable developers to streamline the creation and support of off-chain applications.

Similar to how Ethereum helps developers build a diverse array of applications on its network, so too can off-chain application developers create a myriad of customizable tools and resources on the TX Mesh network – all tailored to suit their business needs, without having to build and maintain their own node infrastructure.

In addition, for node operators, TX Mesh does not require its nodes to maintain a native blockchain network, nor does it rely on a global consensus. This structure offers more flexibility to suit a wide range of solutions and use cases.

Combining robust core infrastructure with lightweight design, TX Mesh lowers the entry barrier for service providers to operate on decentralized node infrastructures. Low cost, credit-card sized computers such as Raspberry Pi can provide enough power to run the majority of transactions needed.

The TX Mesh network is also naturally scalable, as nodes are not limited by aspects like block time and block size. Instead, nodes function in a spontaneous manner, similar to how trading bots operate. The service provider has full-control over how these nodes behave.

In order to achieve these stated objectives, TX Mesh will provide a range of built-in resources, inclusive but not limited to:

1. Service discovery protocol
2. Wallet-based identification protocol
3. Wallet-based access control protocol
4. EIP-1193[2] compatible API provider
5. Pluggable data flow control and billing
6. Pluggable Byzantine Fault Tolerant protocol

[1] A mesh network is a local network topology in which the infrastructure nodes connect directly, dynamically, and non-hierarchically to as many other nodes as possible and cooperate with one another to efficiently route data from/to clients. Please refer to https://en.wikipedia.org/wiki/Mesh_networking for more details.

[2] A JavaScript Ethereum Provider API for consistency across clients and applications. For more information, please refer to https://github.com/ethereum/EIPs/blob/master/EIPS/eip-1193.md

2.1 Project Value Proposition

The vision of TX Mesh is divided into two core phases.

2.1.1 TX Mesh Phase 1: A General-Purpose Infrastructure for Off-Chain Communication and Services

The initial phase of the TX Mesh network will provide a set of general-purpose tools to facilitate off-chain data flow and services. This includes the TX Mesh Node Communication Protocol, SDK, and a set of tools as mentioned above. Through this preliminary component, developers will be able to focus on business-centric logic instead of wasting valuable time and resources building standard node infrastructures.

2.1.2 TX Mesh Phase 2: A SaaS Marketplace

In the subsequent phase, TX Mesh will provide an AWS-style SaaS marketplace that connects the Web3 ecosystem with mainstream Web2 services and applications.

This SaaS marketplace will enable robust composability between Web2 and Web3 applications and services in the same way DeFi projects are able to connect and build on each other.

2.1.3 TX Mesh Use Case Examples

The following examples demonstrate how the TX Mesh infrastructure can enable applications and functionalities from both a Web2 and Web3 perspective.

Example 1*: How Web3 applications can benefit from connecting to Web2 services via TX Mesh*

Since TX Mesh acts as a proxy or gateway for Web2 services outside of a blockchain ecosystem to reach the Web3 world in a non-centralized fashion, all microservice providers on AWS can sell their services to Web3 applications directly via TX Mesh without having to build their own infrastructure.

One example of a beneficiary of this is on-chain prediction markets. Since these platforms have no direct access to vital information needed to determine the outcome of a prediction their users bet on, they typically rely on an open reporting and dispute process that involves the network's token holders.

This creates room for malicious attacks and manipulations that puts users' funds in danger. It is possible for a party holding large amounts of platform tokens to leverage their power and present data that is false or incorrect in the real world as true on-chain (e.g. the weather in NYC on a particular day).

By utilizing TX Mesh, a service provider such as OnPoint Historical Weather Data can tap into the TX Mesh's service network to sell their services to a prediction platform and enable this platform to determine the valid outcome of a prediction.

These Web2 service providers will also enable the users of prediction platforms to create a more holistic and complex range of predictions that are currently difficult to make. This is because, due to the current state of the market, many types of information that determine the outcome of an event are difficult to access/validate by common Web3 platform users.

Example 2: *How Web2 applications can benefit from connecting to Web3 services via TX Mesh*

To demonstrate how Web2 platforms can benefit from connecting to TX Mesh, consider the example of an online store or vendor that sells randomly-generated digital items (e.g. a blind box collection for in-game equipment). Traditionally, in this context of centralized platforms, buyers are required to blindly trust that the store or vendor will not do anything to corrupt or bias the results of the randomly-generated item.

With the help of TX Mesh, however, the vendor will be able to directly and easily utilize transparent Web3 tools to allow the public to verify and gain evidence of the process themselves. This creates a better, more trustful user experience as well as a competitive advantage against other centralized and Web2 platforms.

3 Project Design

3.1 Terms and Definitions

1. **Client**: The client or user that initiates a transaction.
2. **Node**: The transaction engine within the TX Mesh network. Nodes receive and send data and also verify the integrity of each transaction.
3. **Transaction**: The sequence by which a unit of information or data is exchanged. It also encompasses any related activity or units of information that satisfy a request, and ensures the integrity of node and on-chain data on the network.
4. **Procedure**: The procedure/code of a transaction as defined by service providers. Procedures interact with the TX Mesh network through the Node.
5. **Proxy Contract**: On-chain smart contracts that control the boundaries of the transaction. They handle the on-chain operations at each stage of the transaction.
6. **Executor/Witness**: A subset of nodes that specifically executes a Procedure or verifies the results of a Procedure.
7. **SrcChain**: The source chain from which a transaction is initiated.
8. **TargetChain**: The target chain to which a transaction is associated.

3.2 Transaction Properties

As a flexible and distributed transaction framework, the TX Mesh network nodes can each deploy one or multiple Procedures to provide different services. The service provider can define their Procedures according to various scenarios and related applications. TX Mesh will provide a basic transaction negotiation process and a data input/output model for the service providers to construct their own Procedures and Transaction Properties.

Different Procedures will naturally correspond to different transaction models. In the following example we outline some broad qualitative rules for these transactions according to the ACID framework[3], namely Atomicity, Consistency, Isolation, and Durability.

3.2.1 Atomicity

Within the TX Mesh framework and system, transactions are based on the rule of atomicity. In database systems, atomicity is one of the ACID transaction properties. An atomic transaction is an indivisible and irreducible series of database operations such that either all occur, or nothing occurs. A guarantee of atomicity prevents updates to the database occurring only partially, which can cause greater problems than rejecting the whole series outright.

To elaborate on how this relates to TX Mesh: a transaction is started by a Client committing a *TX-PREPARE* message to the SrcChain and locking the designated resources on-chain. The state of on-chain data is changed only when the Proxy Contract receives the correct *TX-EXEC-ACCEPT* message. If the Proxy Contract receives a *TX-EXEC-REJECT* message, it releases the on-chain resources locked by *TX-PREPARE*.

In general, for the TX Mesh network, a transaction starts on-chain A and finishes at chain A. However, this does not mean that TX Mesh cannot handle cross-chain transactions. As described in the later section, TX Mesh can support cross-chain and off-chain data interactions very easily through the *Confirm* process.

3.2.2 Consistency

Consistency in database systems refers to the requirement that any given database transaction must change only change affected data and in allowed ways. For a database to be consistent, data written to the database must be valid according to all defined rules, including constraints, cascades, triggers, or any combination. Consistency does not guarantee correctness of the transaction in all ways an application programmer might expect (that is the responsibility of application-level code). Instead, consistency merely guarantees that programming errors cannot result in the violation of any defined database constraints. The consistency of a TX Mesh transaction depends heavily on the isolation level[4] of the transaction, as well as the implementation of the procedure. It should also only be viewed on a case-by-case basis. TX Mesh will provide common tools to help Procedures achieve correct internal consistency.

3.2.3 Isolation

In TX Mesh, concurrent transactions can affect each other. We have established four isolation levels to determine the concurrent effect of different transactions. They are as follows:

[3] In database management, ACID is a set of properties of database transactions intended to guarantee data validity despite errors, power failures, and other mishaps.

[4] The input type of the data source determines the isolation level of the transaction.

1. **Read Uncommitted**: The lowest isolation level, where one transaction can read the unsubmitted result of another transaction. All concurrent transaction problems can occur.

2. **Read Committed**: Only after a transaction is finalized are the updated results available to be read and processed by other transactions. This solves the dirty read problem.

3. **Repeated Read**: For one transaction, the accessed results for the same data is always the same, regardless of whether other transactions operate on the data and whether the transaction is committed. Dirty reads and non-repeatable reads can be resolved at this isolation level.

4. **Serialization**: This is the highest isolation level. Transactions are serialized and the transaction concurrency is sacrificed. This isolation level solves all of the problems of concurrent transactions.

In practice, we often need to compromise isolation to have better system performance. Concurrent transactions can read intermediate data submitted to the Proxy Contract, but they cannot read uncommitted results processed by Procedure. It is also possible for a Procedure to read off-chain data, which is dependent on the Procedure's implementation and data source. Therefore, strictly speaking, TX Mesh cannot solve the dirty read problem, and is between the Read Uncommitted and Read Committed isolation levels. However, if we consider the Proxy Contract as a global lock[5], we can raise transaction isolation to the highest Serialization level.

3.2.4 Durability

As long as the Proxy Contract accepts the *TX-EXEC-ACCEPT* message, the data changes of the transaction will be stored permanently on the SrcChain, and the results will be reflected on the TargetChain.

3.3 TX Mesh Network Transaction Flow

The process of transactions on the TX Mesh network follows a fairly common distributed system design with four main components.

3.3.1 Prepare

1. **The Client** prepares a *TX-PREPARE* message and broadcast the transaction through the TX Mesh network.

2. **The Executor** receives the *TX-PREPARE* message and decides whether to answer the message by sending back *TX-PREPARE-ACCEPT*.

3. **The Client** collects as many *TX-PREPARE-ACCEPT* responses as possible within the timeout limit and selects the Executor according to the algorithm.

4. **The Client** sends a *TX-EXEC* message to the Executor selected in step 3.

[5]As the on-chain Proxy Contracts are the last entity in charge of whether the data is sent on-chain or not.

3.3.2 Execute

1. **The Executor** uploads the *TX-PREPARE* message on-chain.
2. **The Executor** locally executes the task/transaction described in *TX-PREPARE*.
3. **The Executor** bundles the results of the locally executed transaction in a *TX-EXEC-CONFIRM* message and sends it via TX Mesh.

3.3.3 Confirm

1. **The Witness** receives the *TX-EXEC-CONFIRM* message, checks execution result, and either:
   (a) Accepts the execution result and sends a *TX-EXEC-ACCEPT* message to the Executor via the TX Mesh network.
   (b) Rejects the execution result and broadcasts a *TX-EXEC-REJECT* message via the TX Mesh network.

3.3.4 Commit/Rollback

1. If **the Executor** receives enough[6] *TX-EXEC-ACCEPT* messages, it will acknowledge the result and upload it to complete the transaction commit.
2. Any **Node** that receives enough *TX-EXEC-REJECT* messages in the TX Mesh network can upload the rollback results to complete the rollback of the transaction.

3.4 TX Mesh Network Service Flow

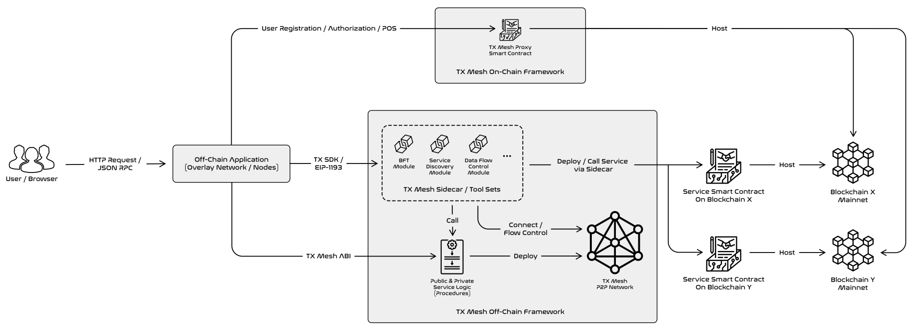
User Registration / Authorization / POS
TX Mesh Proxy Smart Contract
TX Mesh On-Chain Framework
Host
User / Browser
HTTP Request / JSON RPC
Off-Chain Application (Overlay Network / Nodes)
TX SDK / EIP-1193
BFT Module
Service Discovery Module
Data Flow Control Module
...
TX Mesh Sidecar / Tool Sets
Deploy / Call Service via Sidecar
Service Smart Contract On Blockchain X
Host
Blockchain X Mainnet
Call
Connect / Flow Control
TX Mesh ABI
Deploy
Public & Private Service Logic (Procedures)
TX Mesh P2P Network
Service Smart Contract On Blockchain Y
Host
Blockchain Y Mainnet
TX Mesh Off-Chain Framework

Figure 2: TX Mesh Network Flow Chart

After an application specifies its business logic in Procedures, it is ready to provide services for its users.

[6]Standard BFT rules apply here. The result will be acknowledged when $\frac{2}{3}$ of the nodes confirm the transaction and will be subsequently rejected if $\frac{1}{3}$ of the nodes send in *TX-EXEC-REJECT* messages.

When a user interacts with said application, the application will request to utilize different composable tools (EIP-1193, data flow control, BFT, service discovery, on-chain identification, etc.) through TX Mesh to perform tasks/transactions regulated by defined Procedures.

On-chain resources for transactions will be managed by the Proxy Contracts deployed on different blockchain networks with TX Mesh. When transactions are performed and verified by Witnesses/Executors in the TX Mesh P2P network, the Proxy contracts will then commit the results on-chain.

4 Domain Protocols

We previously outlined the internal dynamics of the TX Mesh protocol that addresses the basic process for how transactions interact on the network. On the other hand, the aforementioned description of transaction consistency and durability shows that the implementation of a TX Mesh application is not only complex but also requires detailed attention to the interaction between the Proxy Contract and the Procedure.

To demonstrate the use case of the TX Mesh network and reduce the difficulty of service providers using the platform, we will elaborate on several common domain protocols that outline and standardize the interaction between Proxy Contract and Procedure on top of the base protocol.

4.1 TX Swap

One of these domain protocols is ***TX Swap***, an off-chain decentralized token swap with aggregated liquidity and cross-chain capabilities.

4.1.1 Protocol Overview

Over the past several years, arbitrageurs have been reducing friction and creating greater price stability for a range of financial assets. Specific to cryptocurrencies, arbitrage is often viewed as a way to profit through leveraging various disparate price points of an asset in the market.

Arbitrageurs functioning as cross-pair, cross-platform, and at times even cross-network liquidity providers, however, are often overlooked due to the way many trading products are structured.

TX Swap is a TX Mesh-based platform that allows buyers/sellers of crypto assets to interact directly with arbitrageurs so they may access liquidity on different networks via DEXs, CEXs, dark pools, lending pools, etc.

The TX Swap platform provides an interface for regular users to connect with arbitrageurs and to access liquidity in all centralized and decentralized markets, as well as pre-built tools and strategic templates for arbitrageurs to easily provide a range of services.

4.1.2 Protocol Core Features

OTC-Styled Exchange: On the TX Swap platform, Arbitrageurs function similarly to automated OTC market makers. They compete against each other in terms of liquidity depth, the number of tokens supported, prices, fees, etc. Users will then be able to swap tokens with any chosen arbitrageur while experiencing no nominal price spread or slippage.

Aggregated Liquidity: Since arbitrageurs will compete to provide the best rates for users, they will also have to access as many liquidity sources as possible. This includes DEXs and lending pools across different public blockchains and various centralized exchanges.

Cross-Chain Token Swap: Data related to transactions will be transferred via the TX Mesh off-chain network, allowing a smooth user experience for cross-chain exchanges.

Market Data Provision: TX Swap constantly provides data from the recent transactions of arbitrageurs. For instance, if there are a number of recent token swap requests involving XYZ token, then all arbitrageurs will be informed and can prepare their liquidity accordingly.

Proof-of-Coverage: The platform will periodically broadcast random token swap requests to arbitrageurs. Rewards will be distributed to the arbitrageurs that accept these requests.

4.1.3 Sample Token Swap Procedures

Although there are several ways that arbitrageurs can strategize respective business models, below we have outlined an oversimplified CEX token flow to help demonstrate one of the many ways they can, and will, engage with TX Mesh. Token flow for a different type of liquidity source, or a combination of different liquidity sources all follow the same logic. As an example, although you can replace the example of CEX here with Dex/Darkpool etc. or a combination of the above – the same flow and process will apply to all.

1. **Arbitrageur X** has 100 *ETH* in their on-chain wallet and 200,000 *USDC* in multiple CEX accounts. **Client Y** wishes to acquire 10 *ETH* via TX Swap, and the best quote they get is 2010 *USDC* per *ETH* from **Arbitrageur X**.

2. **Client Y** then deposits 20,100 *USDC* into a TX Swap-controlled smart contract for buying 10 *ETH* from **Arbitrageur X**.

3. **Arbitrageur X** transfers 10 *ETH* directly from his on-chain wallet to **Client Y's** to complete the request.

4. TX Swap releases the deposited 20,100 *USDC* to **Arbitrageur X's** on-chain wallet to close the transaction.

5. **Arbitrageur X** purchases 10 *ETH* via a CEX API call with 20,000 *USDC*, which might not be executed completely and immediately.

6. After some time, **Arbitrageur X's** on-chain wallet contains 90 *ETH* and 20,100 *USDC*, while their CEX account contains 180,000 *USDC* and 10 *ETH*.

7. **Arbitrageur X** acquires a profit of 100 *USDC*.

Note that, in practice, the arbitrageur will likely utilize a CEX dark pool (Wootrade etc.) together with a number of order books and AMM DEXs to provide the client with a better rate while still generating profits. The profit margin will also be lower to help create a better competitive rate and structure for their clients.

Since users are able to choose which arbitrageur to use, and it is unclear to the arbitrageurs who will be chosen and when the trade will be carried out, front-running a user's trade is economically unfeasible for arbitrageurs on TX Swap.

At the time of this writing, the core team plans to develop the full infrastructure of TX Mesh after the basic structure of TX Mesh is finalized. As the purpose of this document is to provide a more generic outline and introduction to TX Mesh, a more detailed outline of TX Swap protocol structure will be released separately.

5 Token Utility

TX is the native utility token of TX Mesh. Its utility is inclusive but not limited to:

1. **Staking**: All nodes are required to stake TX-bonded LP tokens to provide services in the network.
2. **Rewards**: The network incentivizes nodes by rewarding those that are continually online, providing services and penalizing those that aren't.
3. **Voting**: There will be a pluggable BFT consensus protocol for applications[7]. The more TX-bonded LP tokens a node stakes, the more voting power it has.
4. **Community incentives**: The foundation will offer grants and conduct similar initiatives using the TX token to incentivize developers to create more tools and applications on the network.
5. **Accruing fees**: Part of the revenue generated by applications using the network will be collected and distributed to stakers of TX-bonded LP tokens.
6. **DAO governance**: Network and TX token parameters will be governed by a Decentralized Autonomous Organization composed of TX-bonded LP token holders.

After token distribution, a part of the raised funds will be converted to ETH to create TX-ETH LP tokens on AMMs. After the launch of the platform, the foundation will start to offer TX token grants to developers who are building tools and applications on the platform. The community can start earning by acquiring TX tokens to form TX-ETH LP tokens, and stake them to run nodes on the network.

The initial total token supply for TX is 1,000,000,000 (1 billion), with a static (3%) annual inflation distributed to network nodes, and a percentage of accrued fees will be used to conduct TX token burns. As the network matures, we will

[7]This is NOT a global consensus for the entire TX Mesh network. Rather, this is a local consensus mechanism for a specific application, meaning there can co-exist multiple local application-wide consensus within the TX Mesh network.

also introduce a dynamic inflation adjustment mechanism that calculates the current inflation rate based on network size and stability of services.

In later stages of the project when there are multiple revenue-generating applications built with TX Mesh, a TX-ETH LP owner will be able to mint an NFT which represents their locked LP tokens and use this NFT to earn fees generated by applications on the network.