For the future of non-financial applications on Ethereum.

Author: Xuxin “Garvey” Wang

Editors: Feng “Bill” Shi, Zerui “Marco” Cheng, Pengcheng “Paul” Yu, Chenxi “Dylan” Zhang

Pond: https://twitter.com/PondDiscoveries

Preface

When blockchain technology was first proposed, there was an abundance of rhetoric about its potential to disrupt the industry and our society. However, the current reality falls short of those lofty expectations.

Apart from a few applications of financial domain, there have been limited achievements. The whole community are looking forward to the emergence of a truly killer application in non-financial domains, which would give hope to the disillusioned blockchain idealists and showcase the real power of blockchain technology.

We typically reflect on the current situation from the technical perspective. For example, we often approach it from the standpoint of Ethereum’s own technology: although smart contracts are Turing complete, they are still functionally limited compared to traditional programming languages, which makes it challenging to describe more complex use cases. However, we rarely contemplate the transaction-driven model of Ethereum and its implications for innovation.

What does transaction-driven mean? Ethereum is essentially a state machine that changes its state through a series of transactions. Transactions are a form of connection that link the sender and the recipient. Ethereum alters the state of the sender and the recipient through transactions. Although this connection is recorded in historical blocks, we believe its significance is primarily for changing the account state. Have we ever considered the limitations of this weak connection-strong entity model on our thinking?

The transaction-driven model maintains the correct functioning of the blockchain in an economic way. The more complex functions, whether financial or non-financial, are implemented on the smart contract layer. But we still bring this idea from the economy into the construction of the upper layer applications. For example, various human connections are considered assets to possess, such as a person is indirectly expressed as being associated with certain organizations or certain things by holding a Soulbound Token, but this should be able to be expressed more directly. To enable blockchain technology to have a disruptive impact in other fields, we need to consider what fundamental demands can be addressed beyond financial liquidity using this transaction model. We believe that one major problem this model can solve is the recording and consensus of human social identities and relationships. In the words of sociologists, individuals are the sum of their social relationships. The relationships a person has with other individuals, organizations, or things largely define their identity, status, and other social attributes. Since blockchain can describe transactions and financial assets beyond the smart contract layer already, why can’t it describe other kind of relationships? We believe it is possible, but it requires a mindset shift. Once the mindset opens up, we further realize that we should not limit ourselves to certain types of relationships. We can fundamentally change blockchain’s weak connection-strong entity paradigm, thereby endowing blockchain with the ability to describe various connections in the vast world directly.

When a technological framework possesses better capabilities to describe the world, it becomes easier to promote and adopt it. For example, although procedural programming languages are sufficient to address many needs, people still prefer to use object-oriented approaches for programming. Object-oriented programming languages have made large-scale software engineering feasible. From this, we boldly imagine a better blockchain model could lay the foundation for the prosperity of the whole ecosystem and contribute to its flourishing across various domains beyond finance.

Before discussing how to enable blockchain to describe relationships in general, let’s introduce a mathematical concept called graph. Graph is the most commonly used tool and data structure for describing connections. There are two core concepts: entities and relationships. Entities are abstractions of things, such as people in the real world or accounts in the blockchain world. Relationships are the associations between entities, such as the employment relationship between employees and companies or transactional relationships in the blockchain world.

[

Screenshot 2023-05-24 at 05.08.01

1426×908 111 KB

](https://ethresear.ch/uploads/default/original/2X/2/257c16538ee71604bbdecf2256522d35ddfbc2c0.jpeg)

Now let’s take a look at the types of relationship data on Ethereum. At the most fundamental level, the basic relationship is transaction, which we have described earlier. It primarily serves the purpose of changing the state of entities. Once the state change occurs, the remaining value of this data is primarily in traceability and verification. Compared to account entities, transaction relationships can be considered secondary citizens, and some nodes can even choose not to store transaction data. (Diagram 1)

At the smart contract level, let’s take the example of the ERC20 standard. The explicit entities are still accounts, and the explicit relationships are “Transfer” and “Approve” that occur between accounts. If we implicitly abstract the asset token as an entity, there is an implicit relationship of “Own,” where an account owns a certain amount of tokens.

It is evident that the traditional transaction model of Ethereum is well-suited and natural for applications in the financial domain. However, it falls short when it comes to describing relationships in other domains. Yet, the world we live in is filled with diverse types of relationships. For example, to describe the relationship between a student and the school they graduated from, the state-of-the-art is SBT: The school issues SBTs for its graduates and the student owns the SBT to prove they graduate from the school. This is actually a convoluted way to represent a simple “student - graduate from - school” relationship.

At the smart contract level, the status of relationships might be a little better. We can define, create, and store the relationships we desire within the contracts. However, due to the expensive nature of operating blockchains, it is not feasible to store a large amount of relationship data directly on the main chain of Ethereum. Therefore, we propose a persistent relationship layer, also known as a Layer 2, built on top of Ethereum, to address these two issues: 1) representation of relationships, and 2) cost-effective storage of relationships.

By adding a native, decentralized relationship solution - almost like a graph database - as part of the Ethereum ecosystem, we can provide a more effective technology to model the real world. This infrastructure will serve as a foundation for the proliferation of killer blockchain applications, as it enables the Ethereum platform to handle a broader range of relationship-related use cases.

Blockchain is a Graph Database

Indeed, by adopting the perspective of a graph database, we can reexamine the structure of Ethereum and recognize that Ethereum itself is a natural graph. There are already teams engaged in data analysis who attempt to visualize Ethereum as a graph by considering accounts as entities and transactions as relationships. This approach enables analysts to explore various patterns, identify trends, and gain a deeper understanding of the flow of assets within the network. (Diagram 2 above)

While the focus often lies on graph visualization of Ethereum, the underlying structure holds significant value that may go unnoticed. Let’s dive into it. In the discussion above, we mentioned accounts on Ethereum are entities, and transactions are relationships. The data structure of accounts is not easily generalizable, but the data field of transactions is highly customizable. This means that, disregarding the operational mechanisms of Ethereum, the Ethereum structure itself has the ability to depict a graph with various types of relationships. If we consider accounts as concrete individuals and assign specific semantics to the data field of transactions, then with slight modifications, we can construct the following graph with rich meanings (see Diagram 3 above).

It is important to note that, unlike the original transaction model, the relationships in this graph will no longer be secondary. In the original Ethereum model, transactions serve their purpose of triggering changes in account states, but once that mission is accomplished, transactions have little use beyond traceability and verification functions. In fact, in some cases, they can be burdensome and can be omitted from storage in certian nodes. However, the modified relationships have semantic characteristics. If you attempt to remove these relationships, you will find that the information conveyed by the graph will be compromised.

Compared to a simple transaction graph, the modified graph is equiped with unimaginable semantics. Of course, in practice, Ethereum will not be used as a graph database in this manner (it is theoretically feasible but expensive in practice). However, this demonstrates the potential of Ethereum’s fundamental structure to depict relationships with rich semantics. This graph structure, built on the foundation of Ethereum technology, also inherits the tamper-resistance of blockchain, and Merkle tree enables lightweight and feasible proof of existence.

Since the expansive main chain of Ethereum is responsible for maintaining state changes of the accounts, we propose to build a Layer 2 on top of Etheruem’s fundamental structure and make targeted modifications to create a graph database filled with rich semantics. This approach would leverage the inherent features of Ethereum while providing a more scalable and cost-effective solution for capturing and representing complex semantic relationships in the real world.

Consequences of Abstracting the Transaction Layer into a General Relationship Layer

Let’s take a another look at Ethereum’s transactions from these three aspects of relationships: description, effect, and long-term usage. As a concrete example, the description of a transaction can be summarized by sender address, receiver address, value, data, and so on. The effect of a transaction is to modify the Ethereum state tree, and its long-term usage lies in its verification functionality.

When we generalize from simple transactions to semantically rich relationships, each aspect - description, effect, and long-term usage - can be further generalized and enhanced. The description can be abstracted to include sender and receiver addresses along with other attributes. The resulting effect can be customized, including the rules for state transitions, which can be highly customized through smart contracts. In addition to verification, the long-term usage can serve as a means of persisting semantics and can be further enhanced by modifying the Ethereum infrastructure to provide additional functionality for smart contracts.

Once this generalization is implemented, the transaction-based economic model will face dramatic changes. Transactions, as a special type of relationships, have the effect of increasing the balance in the recipient’s account, which however, is not necessarily true for general relationships. Think about the case where the sender sends a transaction with 0 value to the receiver. In this process, it seems that the only beneficiary is the miner who collects transaction fees. But this doesn't mean there is no value in having relationships; rather, it's just that the value is not reflected in the change of account balances. When the sender is an individual or organization with certain authority, the relationship is persisted on the blockchain as a form of proof or authentication. For example, it can be used to prove that a specific account is an employee of a certain organization. To some extend, the transaction fee can be seen as a payment to purchase on-chain storage. And the reason why any rational individual would take such actions is that they believe the value gained from functions like authentication or proof is no less than the value of the transaction fees.

Furthermore, under the current transaction system, miners are motivated by the ability to continuously process and include transaction data in blocks, thereby earning rewards. However, the generation of certain relationships may not be frequent, depending on the nature and scale of the ecosystem around those relationships. For example, in supply chain, the read and write operations on relationships between produtcs may have significant timeliness and seasonality. There may be frequent writes during the production period, while reads may be more frequent after the products are obtained by distributors. If the application layer built on top of this data ecosystem does not generate enough transactions to sustain the normal operation of miners, a reasonable "dormancy" mechanism needs to be designed, which allows the ecosystem to survive periodic "data winters". Alternatively, we can build a sufficiently large graph that encompasses various data ecosystems, and the overlapping cycles of different data ecosystems can, to some extent, provide miners with activities and profitability.

Last but not least, in this system where relationships are put foreground, the degree of decentralization might be impacted. The use of light nodes may be limited or changed, because the presence of relationship data becomes a prerequisite for nodes to provide services. Running a small-scale data ecosystem may be feasible with a PC, but as the scale grows, more powerful hardware with larger storage capacity is required. This necessitates lowering the barriers in terms of data verification to ensure the decentralization of the system. Moreover, the impact of data scale on the cost of virtual machine operations needs to be controlled at a level below logarithmic scale to be practically usable. This requires a robust indexing layer, and techniques from traditional graph databases can be borrowed here.

Use Cases

1. Graph-based DID

A person is defined by the sum of their social relationships, and the determination of a person's identity is largely associated with the relationships they have with various entities, such as education relationship with academic institutions, employment relationships with companies, and personal relationships with other individuals. A graph can clearly and intuitively represent these different relationships, offering better identity solutions.

1. Quantification of social capital

Just like financial assets, a person's social relationships also hold value. When the social graph reaches a certain scale and smart contracts with enhanced computational capabilities are implemented, it becomes possible to further process data on the blockchain. We can establish mechanisms to evaluate and measure the value of social relationships. This opens up possibilities for leveraging social capital in various domains, such as lending, reputation systems, and collaborative networks. Social collateral, which represents the value derived from one's social relationships and reputation, can be used as a form of collateral or trust guarantee in financial transactions, partnerships, and other interactions.

1. A new narrative of asset ownership - shared assets

Shared assets refer to assets that are collectively owned or accessed by multiple individuals or entities. Once ownership is represented as a special type of relationships, then it is super easy to depict co- onwership - it is a one-to-many pattern of relationships.

1. Negative reputation

When relationships are imbued with semantics, they gain the ability to capture brief evaluations, allowing positive or negative feedback to be recorded. This enables the tracking of reputational information associated with specific entities.

1. Unifying identity verification and access control

Based on the graph layer, it is possible to establish a standard at the top level for unifying identity verification and access control across various dApps. It also enables the possibility of social recovery for lost accounts.

Implementation Plan

1. Modify fields of accounts and transactions to make them more suitable as a graph database. This includes allowing accounts to have customizable attributes, enabling richer representation of entities.
2. Modify smart contract module to provide programmable customization for complex relationship operations such as create, modify, and delete. For example, unique constraints on relationships can be defined and enforced at the smart contract level. Additionally, schema restrictions can be defined, such as enumerating available relationship types, to standardize the usage of a graph database.
3. Modify the virtual machine to provide necessary support for smart contract operations. This ensures efficient and

reliable execution of graph-related operations within smart contracts.

4. Add pluggable indexing layers to enable efficient data retrieval.

5. Integrate zero-knowledge (zk) techniques (such as zkRollup) for scalability and performance.

Value of a Native, Decentralized and Graphized Layer2

1. This system can be seen as a blueprint that generates different instances during actual operation. When a new application scenario is identified, a new cluster of nodes can be established to support the operation of the graph database for that specific application. When there are multiple ecologies of graph databases on chain, such as a social ecology and an identity ecology, Ethereum’s expressive power will reach the next level. From this perspective, the Ethereum main chain can be considered as a special instance of a graph database, where its relationships are limited to transactions.

2. It is also possible to aggregate data from various ecosystems, such as educational and professional ecosystems, into a single graph, creating a unified graph database that covers cross-sector relationships.

3. The fundation with richer semantics can provide programming assistance for writing complex smart contracts that require handling relationships. Smart contracts will be more powerful in describing and processing relationship-related tasks, as they directly interface with a graph database.

4. The persistent layer of relationships, as the infrastructure, can be separated from the smart contract layer, providing more flexibility in operations. For example, data can be migrated between different EVM chains, different graph databases can be merged when necessary, or one graph database can be split into multiple. The underlying changes can be made without or with minimal modifications to the upper-level applications, benefiting from the separation of layers.

Incentive Model

1. Transaction fee

2. Issuing token

Prevention and Remediation of Abnormal Data

When an account is hacked or intentionally engages in malicious activities, abnormal relationship data might appear in the graph. While such problem already exist in current smart contracts, when the relationship layer serves as an infrastructure providing services to upper-layer applications, these abnormal data can propagate to the upper-layer level, causing bigger impact. Therefore, it becomes even more important to prevent and address such issues.

Preventive measure: Use multi-signature mechanisms for onboarding a relationship. The probability of multiple accounts being compromised is lower than that of a single account being compromised. Additionally, certain rules can be implemented to allow for the replacement of an account in the event of theft or loss, ensuring the continuity of normal account operations. While these measures may not completely eliminate the occurrence of abnormal data, they do increase the difficulty of malicious activities to some extent.

Corrective measure: Have an adjustable observation period before fully creating a relationship. During this observation period, if any abnormalities are detected, the community or administrators can initiate a rollback process to rectify the situation. This allows for the identification and removal of erroneous or malicious relationship data, ensuring the integrity and accuracy of the database.