TRIAS StreamNet: DAG platform based on flow graph calculation

Zhaoming Yin
TRIAS Lab
Hangzhou, China
yinzhaoming@trias.one

Junqing Wang
TRIAS Lab
Hangzhou, China
wangjunqing@trias.one

Ming Ni
TRIAS Lab
Hangzhou, China
wangjunqing@trias.one

Ming Wei
TRIAS Lab
Hangzhou, China
wangjunqing@trias.one

Anbang Ruan
TRIAS Lab
Hangzhou, China
wangjunqing@trias.one

***Abstract*—TRIAS StreamNet is a new design based on the existing mature DAG system. It is prone to double-flowering and replay attacks against existing systems, and the transaction speed is slow. The introduction of Coordinator leads to the centralization assumption. Based on the flow chart calculation in the graph calculation, a new DAG platform is designed that comprehensively considers transaction information (points), transaction approval information (edges), and network structure (such as Katz Centrality).**

***Keywords*-Near Repeat, Mapreduce, Graph Analysis**

I. Basic concepts

A. basic data structure (StreamNet)

StreamNet is the underlying data structure in DAG. It is a Directed Acyclic Graph, where each node in the DAG represents a transaction and the directed edge represents a confirmation relationship between transactions. For example, site 0 in Figure 1 represents the Genesis transaction, which is initially identifiable as a confirmed transaction (in theory, it is also a 100%confirmed transaction). Site1, on the other hand, represents the first transaction, which is confirmed by the subsequent site 2, 3, 4. When a new transaction is not confirmed, it is called a tip. For example, in Figure 1, site 6 is a tip.

B. Trading

1) Creation Trading: There is no concept of mining in StreamNet, and all tokens are included in the creation transaction block. In Figure 2, a creation transaction is described with an initial token number of 5.

2) Trading Content: Assume that in Figure 2, Genesis wants to transfer 1 token to Alice, and then hopes to transfer 1 token to Bob and attach the transaction block corresponding to this transfer to StreamNet, then the resulting StreamNet is shown in Figure 3. Here, each transaction must find two tip transactions to confirm, namely trunk and branch transactions. For example, Genesis $\rightarrow$ Alice's transaction confirms the Genesis transaction itself, while Genesis $\rightarrow$ Bob's transaction confirms the Genesis transaction itself and Genesis $\rightarrow$ Alice's transaction. When a transaction wants to be attached to StreamNet, it must do enough Workload Proof (POW) [2], but this POW differs from Bitcoin in that its difficulty is fixed and therefore does not need to be The participation of miners [3].

3) transaction verification (approve): Because a new transaction needs to find two tip transactions for approval, the verification method has two steps:

- with Genesis, verify all transactions that are directly or indirectly referenced, mainly to see if this will result in a negative balance or loss of the token [1]. For example, in Figure 4, Alice $\rightarrow$ Sam's transfer needs to verify transactions that are indirectly or directly referenced. Then it constructs a topological sequence according to the characteristics of DAG, namely (Genesis) $\rightarrow$ (Genesis $\rightarrow$ Alice) $\rightarrow$ (Genesis $\rightarrow$ Bob) $\rightarrow$ (Alice $\rightarrow$ Sam), and finds that each step does not violate the principle of transfer. Then the verification is successful. In Figure 5, the same topology sequence, when verifying (Genesis $\rightarrow$ Bob), because the Genesis balance will be reduced to -1, the verification fails, as an honest node, Alice $\rightarrow$ Sam transaction Will find a new tip to verify, but it can also choose to spoof, attach this transaction to the selected tip. However, such an approach is likely to result in subsequent rejection of its own transactions.
- At the same time, the signature needs to be checked during the verification of the topology sequence to ensure that the link relationship has not been tampered with.

4) Select tip: There are two basic concepts in StreamNet, one is the transaction rate, which indicates the number of transactions per time unit. For convenience, we set the time unit to seconds. The other is invisible time, indicating how many time units a transaction has not been seen by other exchanges after attach. Because of some existence, the transaction rate has an important influence on the shape of StreamNet. For example, in Figure 7, when the transaction rate is slow, StreamNet is more like a chain. In the case of block trading in Figure 8, the shape of StreamNet is a star.

One of the most basic tip selection algorithms is to start from Genesis to move the transaction with approval to an equal probability until a tip is selected, as shown in Figure 8. Suppose Alice $\rightarrow$ Sam's transaction wants to select tip, which starts from Genesis transaction. There are two options for this time, one is Genesis $\rightarrow$ Alice, the other is Genesis $\rightarrow$ Bob, the probability of selecting Genesis $\rightarrow$ Alice is 1 /2, while Genesis $\rightarrow$ Bob is 1/2.

The problem with the basic random walk algorithm is that it produces lazy transactions. For example, in Figure 9,transaction 14 is a lazy transaction that causes new transactions to approve older transactions without being penalized. This problem can be solved by using Monte Carlo Random Walk (MCMC). In Figure 10, there is a weight on both transactions, for example, Genesis $\rightarrow$ Alice is 8, and Genesis $\rightarrow$ Bob is 2, then the probability of selecting Genesis $\rightarrow$ Alice is 8/10, and Genesis $\rightarrow$ Bob is 2 /10. The determination of the weight is determined by how many transactions have directly or indirectly approved the transaction. The more approved transactions, the greater the weight. If only deterministic use weights are then a super-weight algorithm, meaning that large-weight transactions are always preferred. The problem with this algorithm is that there are many transactions that can never be confirmed. Figure 11 shows the results of a super-weighted algorithm. For example, if purely using probability weighting, then it is a super-probability algorithm. The trade-off between the two is represented by a α, and it can be considered that the larger the α is, the smaller the randomness is. The method of specifically using α to calculate the jump probability is expressed in the formula (1). Which represents the weight of the trading node.Where P_xy represents the probability of jumping from x to y. H_y represents the weight of the trading node y.

$$P_{xy} = \frac{e^{\alpha H_y}}{\sum_{z:z\rightarrow x} e^{\alpha H_z}} \tag{1}$$

5) Trading Consensus: There are currently three ways to confirm a transaction in StreamNet:

- The first way is that the common nodes covered by all the previous tips are considered to be fully confirmed; for example, in Figure 12, the nodes referenced or indirectly referenced by tip1 are blue and yellow line covered transactions, while the tip2 reference Or the indirectly referenced node is a yellow line covered transaction. If there are only 1, 2 tips in StreamNet, then the green node is a fully confirmed transaction, while the green node is an unconfirmed transaction.
- The second way is that the system sends a Coordinator tip every 1 minute. This tip is called milestone and is attached to StreamNet. All transactions referenced by this Coordinator tip are confirmed.
- The third way is to use Monte Carlo Random Walk (MCMC). Call N times to select a tip using the tip selection algorithm. If a trading node is referenced by this tip, its credibility is increased by 1. After M selections have been cited M times, then the credibility is M / N.

II. Problems with existing DAGs

A. Network synchronization

When a node accepts a request to attach a transaction and attaches the transaction to the local StreamNet, it broadcasts the changes made to its own neighboring machine. When the neighboring machine receives the update, it will follow the basic principles of attaching StreamNet. This update is attached to its own StreamNet and this update is further broadcast. If this update cannot be accepted by a large number of neighboring machines, the probability of being accepted by the entire network is greatly reduced. There are several problems in this:

- Issues with offline updates. Offline updates are recognized in StreamNet, but when they are re-online, it will broadcast all local updates, but their local updates will only be accepted in the entire network and will not be partially accepted [4].
- How to recognize that a locally confirmed transaction is a transaction accepted by the whole network. How to express it on the wallet is a relatively big problem. In Bitcoin, there is no honest chain and dishonest chain. It is mainly realized by the cost of lying, that is, the POW of exponential growth, wanting to revert the transaction, or constructing the wrong chain. The price is very high. For a wallet, the time to verify a chain is the complexity of polynomial. In this way, we must believe that the data of a certain node in the whole network can be done. Although there are POWs in the current DAG, the complexity of constructing StreamNet itself is polynomial, so for the lying node, it can use a large computing power to construct a wrong StreamNet to mislead the transaction wallet.
- Existing DAGs are currently unable to guarantee strong consistency in the network environment [5].

B. double flower problem

The double flower problem refers to the problem that the same token is used multiple times, which can be represented by the example in Figure 9, where the two transactions of w and y are double flower transactions, and the transaction 5 is the one that finds that this is a double flower transaction. Transaction. In the transaction consensus, it is obviously impossible to use the 100% tip confirmation, because there will always be new transactions coming in to break the 100% confirmation fact. This is called propagation delay [4]. If a random walk algorithm is used, given a confidence level (say 95%), one of the transactions will naturally receive more transaction approvals in the natural state (without being subjected to a power attack), thus giving it confidence. The degree reaches a state sufficient to be confirmed. For example, as shown in Figure 10, w receives more confirmations in the future, and slowly y is isolated to lose the possibility of being confirmed. But when the fraudster's power is strong enough, it can issue enough transactions to approve w after a transaction is confirmed, then in this case, the previously confirmed transaction will in turn be marginalized. To achieve the purpose of double-flowering, such an attack is a replay attack, and it is necessary to accumulate 34% of the computing power of the whole network to achieve the goal [6]. However, considering the low number of transactions in the entire network in the early stage [7], 34% of the power attack is actually not difficult. In order to solve such problems, the transaction confirmed by the coordinator is absolutely valid, so regardless of the computing power of the follow-on attacker. Powerful, can not beat the coordinator's one-vote veto [1].

Therefore, the double flower problem is essentially a computing power problem, and it is also the essential problem of the existing DAG system. The introduction of the coordinator is a centralized solution, and in the future, as more and more devices join the StreamNet network, the role of the coordinator It will become unnecessary, and how to remove it to turn DAG into a decentralized network is a challenge [8].

C. *transaction confirmation speed problem*

The speed at which the transaction is confirmed and the likelihood that the transaction will be finalized depends on two factors, the first is the transaction rate λ, and the second is the randomness α. From Figure 16, it can be seen that the probability of increasing the success of the transaction is mainly to increase the transaction. The rate reduces randomness, and in Figure 17, a more intuitive expression of the effect of α can be seen, under the same conditions, the higher the unconfirmed transaction becomes more. However, because the transaction rate in the whole network is relatively slow, there are not many active nodes, and some optimization methods have been proposed. For example, the method of coordinator mentioned above and the method of Reattach, rebroadcast [10], etc. It can be attributed to the method of human intervention.

D. *Encryption Algorithm Vulnerabilities*

Because the current DAG encryption algorithm is based on Trytes, and the encryption algorithm is invented by itself, the method pointed out in [11] can find the hash collision in a few minutes using ordinary computers. The attacker can use this vulnerability to fake other users. Signature, fundamentally disintegrating the security of IOTA.

E. *replay attack*

Replay attacks In addition to the use of power to attack in the double-flower problem, two attack modes are mentioned in the white paper, the first one is a side chain attack and the other is a double-sided chain attack [12].

III. StreamNet main algorithm

A. *Based on the centrality (katz centrality) to COO starting point selection algorithm*

At this stage of the DAG, when selecting the tip, it will not start from the creation transaction, but will simply start with a coordinator as the starting point. This will lead to a centralization problem. So the first question we considered when designing StreamNet was how to remove the COO and achieve a truly distributed DAG. So we need a consensus authoritative transaction as a starting point rather than a coordinator mandated by a centralized node as a starting point. Here we choose katz centrality [13] as the starting point for selection criteria. In StreamNet, we use an Adjacency Matrix to represent the link relationship between transactions, which is used to represent, and a second-order link matrix (representing the number of nodes that jump from node to node by two steps). Similarly, we represent k-order link matrix. . Then the importance vector of each node can be calculated by (2). Where is an importance weight vector, which is a matrix of all ones. Because the transactions in StreamNet are constantly entering the network, if you need to recalculate the Katz center degree every time you make a tip selection, then the complexity will be very large, so you need a streaming computing framework. To dynamically update the Katz center degree based on the newly added nodes, we use an incremental algorithm to deal with the flow graph calculation [14].

$$\sum_{k=1}^{max} \alpha^{k-1} A^k = A(I - \alpha A)^{-1} \tag{2}$$

It should be noted that in the calculation results, we do not need to find the most transactions in the center of Katz, because this transaction is always a Genesis transaction, so we should find this in the recent transaction between Katz centrality and time. Initial node.

B. *Considering the transaction weight algorithm of edge information*

Because the attacker can attack the main network (Main StreamNet) in different forms, a typical scenario in which we consider the double-flower problem is the Simple Parasite Chain attack. A simple side chain is shown in Figure 18. In [15], the initiator proposed to use local modifier to solve the attack, but since there is no relevant code, we will not discuss it here. Here we discuss our weight update algorithm with side information. The framework of this algorithm is consistent with the framework of the weight update algorithm in the existing DAG. The difference is that when making a set join between two approve transactions, a weighted set join is performed, and the weight is determined. It is determined by the information of the side, and the information of the side is mainly confirmed by time. For example, in Figure 1, assume that each edge is given a weight w1-w12 transaction 5 because it is approved by transactions 6, 7, 8 and its weight information is [5,7*(w1*w6+*w2) , 8*w3, 6*w6].

The reason for the side information is that the attacker often sends out a large number of transactions within a short period of time and approves each other to achieve the purpose of rapidly growing the side chain. And because the edge information is used to rescale the transaction, the trading effects of these attacks are attenuated. On the contrary, the non-attack type transaction is similar to the speed of the whole network, and its weight update is similar to the result of the original algorithm.

C. Weight update algorithm based on flow graph calculation

When updating the weight, the static graph algorithm needs to calculate the weight of each node from the beginning of the initial recognition node. The complexity of this algorithm is , if our static calculation information is cached, when the new tip is added. By updating the already cached information, the complexity of calculating the weight each time the tip is added can be differentiated . But there is also a problem with this, the spatial complexity of this algorithm will become . How to design a streaming algorithm to reduce the complexity of time space complexity to is a challenge.

D. Using StreamNet in conjunction with IPFS to cache TRIAS transfer requests

StreamNet can be used to cache and pre-confirm other low-traffic blockchain systems because of its high throughput. TRIAS as a cross-blockchain system can take advantage of this feature to achieve its high degree of expansion, which can achieve the ability of elastic The specific structure is shown in Figure 20. There is a distributed transfer service. When an external transfer request comes over, it first stores the information transferred from the IPFS and gets a hash. The transfer server then constructs a hashNet transaction. When the traffic is small, StreamNet can directly push the hash of the transaction to TRIAS after confirming the transaction hash, so TRIAS can get the specific transfer information from IPFS and package the transfer. When the traffic is large, StreamNet will continue to confirm These transactions. When TRIA is idle, it will pull the confirmed hash to StreamNet, and get the specific transfer information from IPFS and package the transfer.

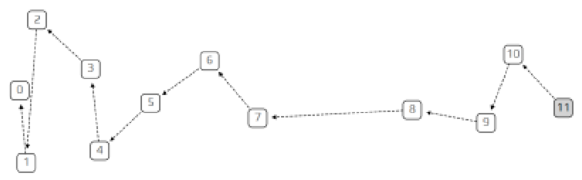

Figure 6. StreamNet shape under slow trading

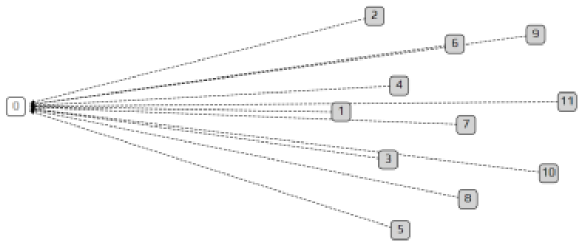

Figure 7. StreamNet shape under fast trade

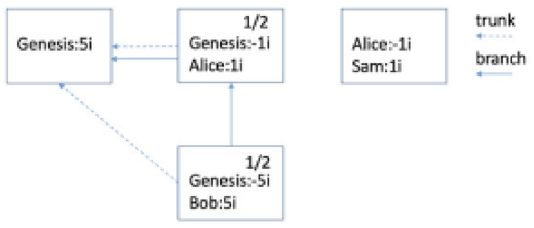

Figure 8. Equal Probability Random Walk Algorithm

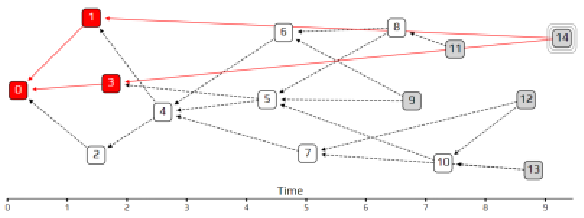

Figure 9. lazy transaction example

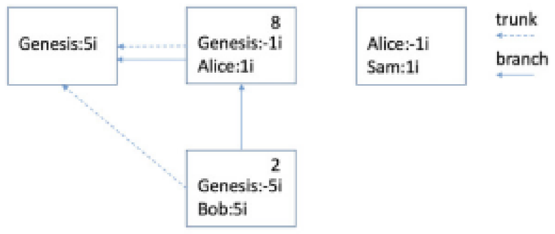

Figure 10. Random walk selection tip

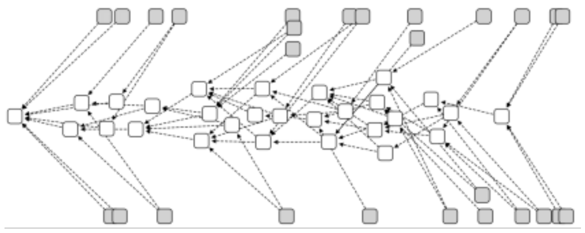

Figure 11. super-weight algorithm example diagram

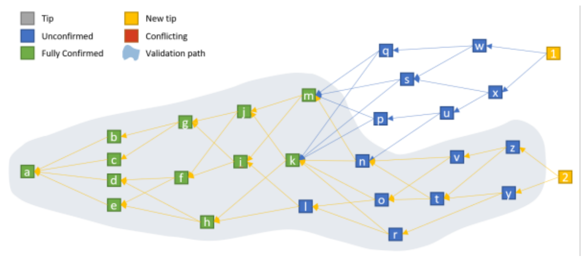

Figure 12. fully confirms the node, unconfirmed node [4]

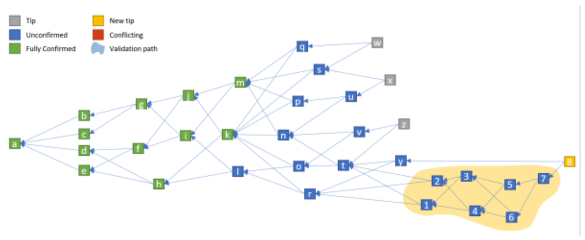

Figure 13. Example of offline update [4]

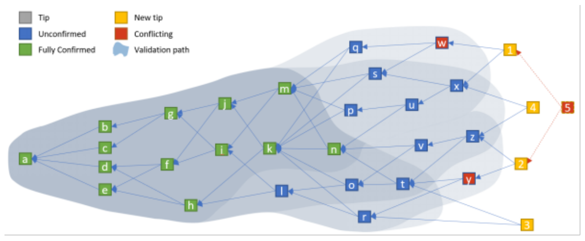

Figure 14. Example of double flower problem

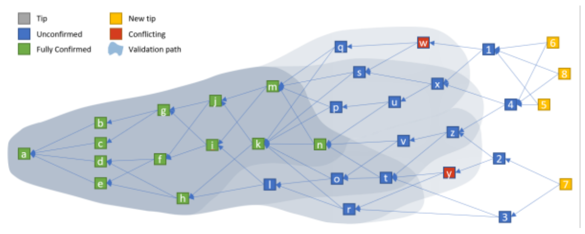

Figure 15. Randomly solve the problem of double-flowering