# StreamNet: Enabling Large Scale DAG based Nakamoto Consensus through Streaming Graph Computing

***Abstract*—Considering major issues in the existing DAG systems such as centralization caused by the introduction of Coordinator, double-spending and sybil attacks, slow transaction speed etc. We designed and implemented the StreamNet system which is forked from the IOTA implementation. By leveraging the Topological sorting and Katz centrality computation, we infer a pivotal chain along the path of each topological level in the graph, and each block in the pivotal chain will have the highest Katz centrality score, which could be used as the entry point for the tip selection algorithm. In addtion, with comprehensive integration of information such as transaction (vertices), transaction approval information (edges), and network structure (such as community structure) we designed the tip selection algorithm that obeys the local modifier method and can detect and avoid double spend / sybil attacks.**

***Keywords*-Block chain, DAG**

## I. Basic concepts

### A. Basic data structure

StreamNet 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 instance, node 0 in Figure 1 represents the Genesis transaction, which is by default a confirmed transaction (in theory, it is also a 100%confirmed transaction). Node 1, on the other hand, represents the first transaction, which is confirmed by the subsequent Node 2, 3 and 4. When a new transaction is not confirmed, it is called a tip. For example, in Figure 1, Node 6 is a tip.

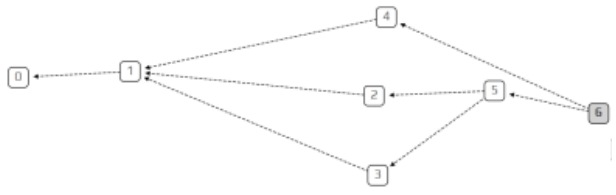


Figure 1. Example of the StreamNet data structure [6].

### B. Transaction

*1) Genesis transaction:* There is no concept of mining in StreamNet, and all tokens are included in the Genesis Node. In Figure 2, an initial transaction is created with 5 tokens.


Figure 2. The creation of genesis node.

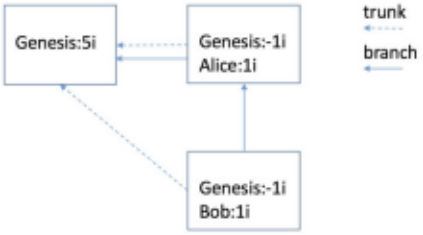


Figure 3. An example of token transfers.

*2) Transaction 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 node to StreamNet, the resulting $DAG$ is shown in Figure 3. Here, each transaction must find two tip transactions to confirm, namely trunk and branch transactions. For example, the Genesis $\rightarrow$ Alice transaction confirms the Genesis transaction itself, while the Genesis $\rightarrow$ Bob transaction confirms the Genesis transaction and the Genesis $\rightarrow$ Alice transaction. When a transaction wants to be attached to StreamNet, it must do enough Proof of Work (POW) [5], but this POW differs from Bitcoin in that its difficulty is fixed and therefore does not need the participation of miners.

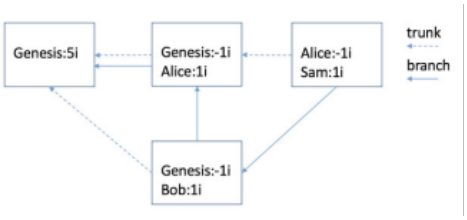


Figure 4. An example of succesfull token transfer.

*3) Transaction validation (approve):* Because a new transaction needs to find two tip transactions to approve,

the validation process is diveded into two steps:

- Start from 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 [6]. For example, in Figure 4, the Alice → Sam transfer needs to validate transactions it indirectly or directly approves. It constructs a topological transfer sequence, namely (Genesis) → (Genesis → Alice) → (Genesis → Bob) → (Alice → Sam), and find that each step does not violate the principle of transfer, which means the verification is successful. In Figure 5, the same topology sequence, when verifying (Genesis → Bob), because the Genesis balance will be reduced to -1, the verification fails. As an honest node, Alice → Sam transaction Will find a new tip to verify, but it can also choose to cheat, attach this transaction to the selected tip. However, it is likely to result in subsequent rejection of this transaction.
- At the mean time, the signature of the transaction needs to be checked to ensure that the link relationship has not been tampered with.

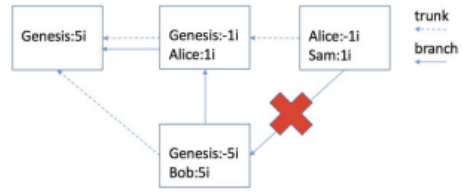


Figure 5. An example of errornous token transfer.

*4) Tip Selection:* There are two basic concepts in StreamNet, one is the transaction rate $\lambda$, which indicates the number of transactions per time unit. For convenience, we set the time unit to seconds. The other is invisible period $h$, indicating how many time units a transaction has not been seen by other incoming blocks after attachment. Because of $h$, the transaction rate $\lambda$ has an important influence on the shape of StreamNet. For example, in Figure 6, when the transaction rate is slow, StreamNet is more like a chain. In the case of high throughput transaction in Figure 7, the shape of StreamNet is a star.

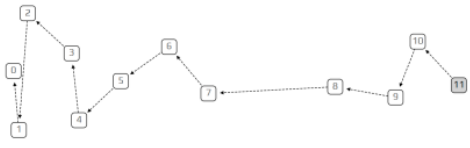

Figure 6. The shape of StreamNet when the txn rate is low.

One of simplest tip selection algorithms is the random walk with equal probability starting from Genesis, as shown

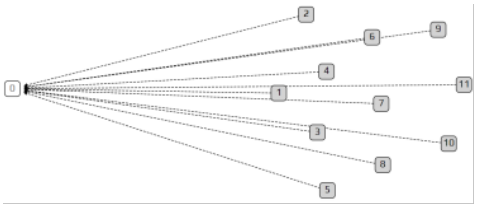

Figure 7. The shape of StreamNet when the txn rate is high.

in Figure 8. Suppose Alice → Sam transaction wants to select tip, which starts from Genesis transaction. There are two options, one is Genesis → Alice, the other is Genesis → Bob, the probability of selecting Genesis → Alice is $\frac{1}{2}$, while Genesis → Bob is $\frac{1}{2}$.

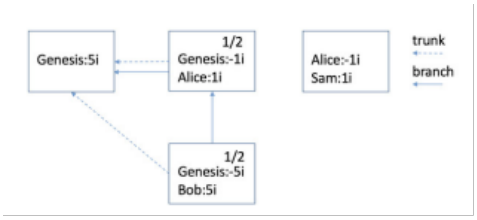


Figure 8. Random walk with equal probability.

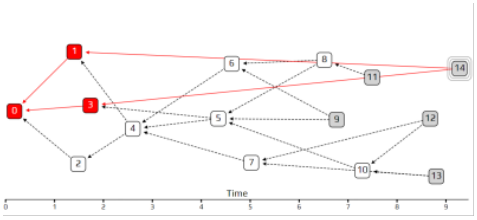


Figure 9. An example of lazy transaction.

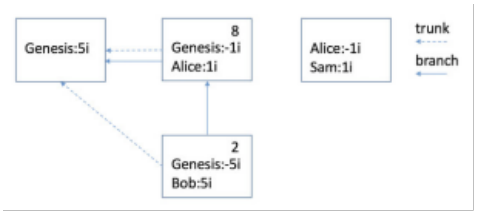


Figure 10. Random walk with equal probability.

The problem with the simple 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 → Alice is 8, and Genesis →

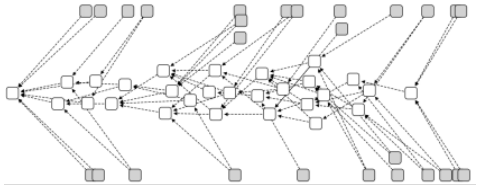

Figure 11. Example of the result using super-weight algorithm.

Bob is 2, then the probability of selecting Genesis → Alice is $\frac{8}{10}$, and Genesis → Bob is $\frac{2}{10}$. 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 weights are used then it is 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. If purely using probability weighting, then it is a super-probability algorithm. The trade-off between the two is represented by a $\alpha$, and it can be considered that the larger the $\alpha$ is, the smaller the randomness is. The method of specifically using $\alpha$ 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 $y$.

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

*5) Transaction Consensus in the Gossip network:* 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 block 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 to the local StreamNet, it broadcasts the changes to neighboring StreamNet nodes. When the neighbors receives the update, it will follow the same principles to merge these changes and further broadcast the updates. If this update cannot be accepted by a large number of neighbors, the probability of its being accepted by the entire network is largely reduced. There are several problems in this:

- Issues with offline updates. Offline updates are implemented in StreamNet, but when offline nodes rejoin, they will broadcast all local updates, but their local updates will only be accepted as a whole and will not be partially accepted [3]. Example of partial updates is shown in Figure 13.
- Existing DAGs are currently unable to guarantee strong consistency in the network environment.

### B. Double Spending Problem

The double spending problem refers to the problem that the same token is used multiple times, which is shown in the example in Figure 14, where the two transactions of $w$ and $y$ are double spending transactions, and the transaction 5 is the one that finds it out. Given a confidence level (say 95%) in the random walk algorithm, one of the transactions will naturally receive more transaction approvals in the natural state until it reaches a state sufficient to be confirmed. For example, as shown in Figure 14, $w$ receives more confirmations in the future, and slowly $y$ is isolated to lose the possibility of being confirmed later. 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-spending, it is necessary to accumulate $34\%$ of the computing power of the whole network to achieve the goal [9]. However, considering the low number of transactions in the entire network in the early stage, 34% of the power attack is actually not difficult. To solve this problem, the concept of coordinator is introduced, the transaction confirmed by the coordinator is absolutely valid. Regardless of the computing power of the follow-on attacker, it can not beat the coordinator's one-vote veto . The introduction of coordinator is a centralized solution, and in the future, as more and more devices join the StreamNet network, how to remove it to turn DAG into a decentralized network is a challenge.

### 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 $\lambda$, and the second is the randomness $\alpha$. From Figure 15, it can be seen that

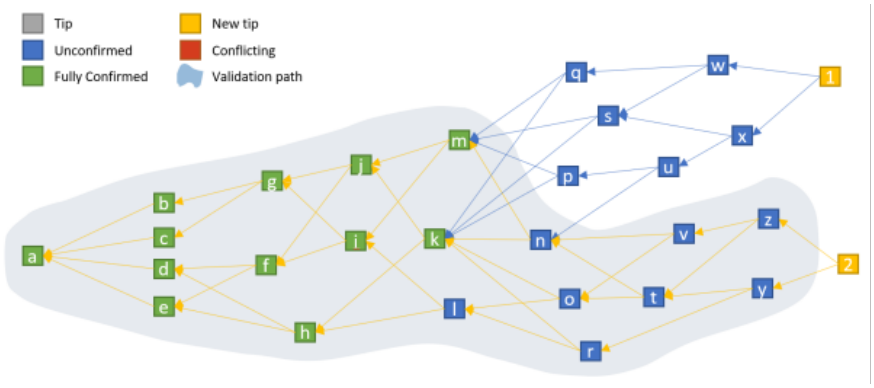


Figure 12. Example of fully confirmed transactions.

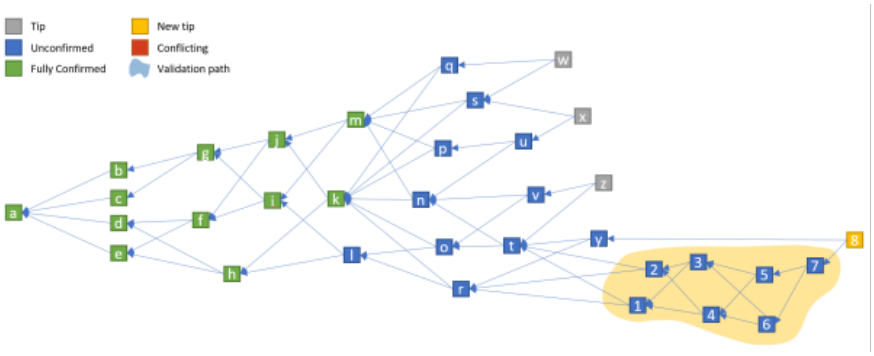


Figure 13. Example of offline update [3]

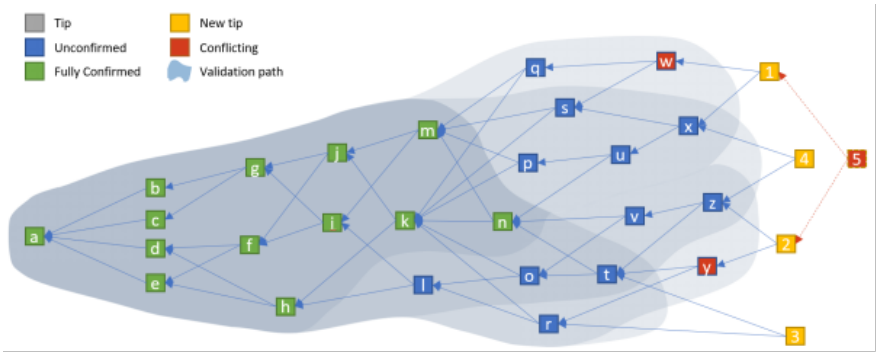


Figure 14. Example of double spending problem

the probability of increasing the success of the transaction is mainly due to the increase the $\lambda$ and reduces $\alpha$. and in Figure 15, a more intuitive expression of the effect of $\alpha$ can be seen in Fifure 16, under the same conditions, the higher $\alpha$ will result in more unconfirmed transactions. However, because the transaction rate in the whole network is relatively slow, there are not many active nodes, thus some ad-hoc optimization methods have been proposed. For example, the method of coordinator mentioned before and the method of Reattach, rebroadcast etc.

### *D. Encryption Algorithm Vulnerabilities*

Because the current DAG encryption algorithm is based on Trytes, and the encryption algorithm is invented from scratch, the method pointed out in [2] can find the hash collision in a few minutes using commodity hardware. The attacker can use this vulnerability to fake other users. Signature, fundamentally disintegrating the security of IOTA.

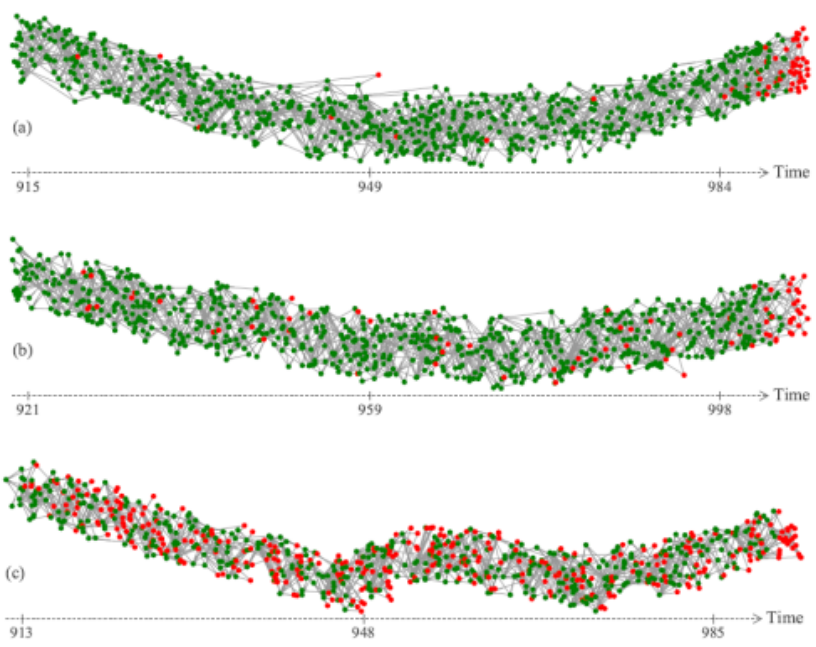


Figure 11: Graphical representation of the Tangle for $\lambda = 10$, $h = 1$ over 100 units of time, all using MCMC with (a) $\alpha = 0.05$, (b) $\alpha = 0.1$ and (c) $\alpha = 0.5$

Figure 16. A more intuitive expression of the role

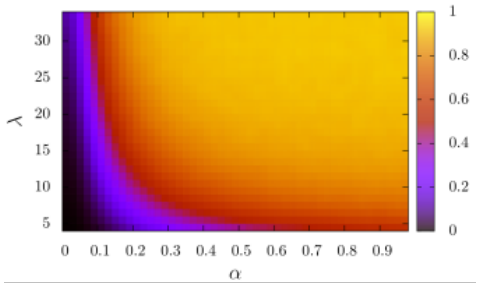


Figure 15. The probability of a transaction being permanently stuck (failed), $\alpha$ is randomness, $\lambda$ is the transaction rate [1]

### E. replay attack

Replay attacks In addition to the use of power to attack in the double-spending 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 [4].

## III. StreamNet main algorithm

### A. Storage

### B. DAG total ordering algorithm

### C. New Tip Selection Algorithm based on Local Modifier Strategy

*1) Entrypoint selection:* When performing tip selection, a entry point is neccessary, and in IOTA mainet, it will not start from the genesis transaction, but will simply start from a coordinator as the entry 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 decentralized DAG. So we need a consensus authoritative transaction as a entrypoint rather than a coordinator mandated by a centralized node. Here we choose katz centrality [7] as the entry point for selection criteria. In StreamNet, we use an Adjacency Matrix to represent the direct link relationship between transactions, which is represented by $A$, and a second-order link matrix $A^2_{ij}$ (representing the number of nodes that jump from node $i$ to node $j$ by two steps). Similarly, we represent k-order adjacency matrix $A^k$. Then the importance vector of each node can be calculated by formula (2). Where $\alpha$ is vector which measures the vertex importance, which $I$ is an identity matrix of all ones. Because the transactions in StreamNet are constantly entering the network, if recalculate the Katz

centrality every time a tip selection is performed, then the complexity will be intolerable as the network is growing constantly. So a streaming computing framework is needed. To dynamically update the Katz centrality based on the newly added nodes, we use an incremental algorithm to deal with the streaming graph calculation [8].

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

It should be noted that we do not need to find the transaction with the largest Katz centrality score in the whole network, because this transaction is always the Genesis transaction. So we specify a depth and find the one with the largest score on this depth.

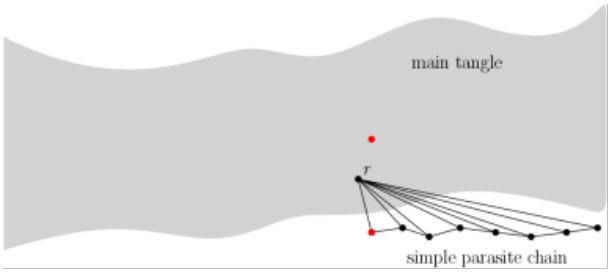


Figure 17. An example of simple parasite chain attack, a series of cheat transaction will refer one specific double spent transaction, when the $SPC$ grows to a certain amount, the double spent will be successful, the two red node in the figure respents the conflict transactions.

*2) Tip selection Considering Edge Information:* Because the attacker can attack the main network (Main StreamNet) in different forms, a typical scenario in which we consider the double-spending problem is the Simple Parasite Chain attack. A simple side chain is shown in Figure 17. In [4], the author proposed to use local modifier to solve the attack, but since there is no relevant code, we will not discuss it in detail. Here we discuss our weight update algorithm with edge 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 by edges. And the information of the edge is mainly determined by time information. For example, in Figure 18, assume that each edge is assigned a weight $w1$ to $w12$, because transaction $5$ is approved by transactions 6, 7, 8, as a result its weight is $[5, 7 * (w1 * w6 + *w2), 8 * w3, 6 * w6]$.

The reason for the adding of edge information is that the attacker often sends out a large number of transactions within a short period of time to achieve the purpose of rapidly growing the side chain. If edge information is used to rescale the weight, the effects of these attacks are attenuated. On the contrary, becasue the issuing rate of 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.

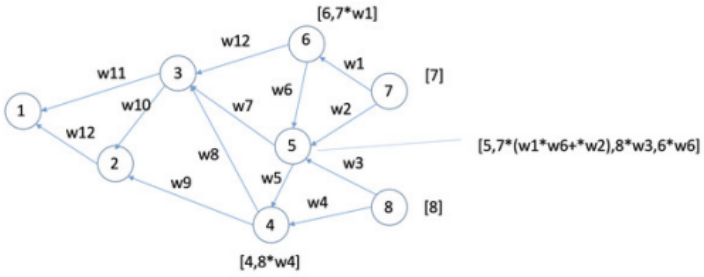


Figure 18. Weight calculation based on edge information.

*3) Weight update algorithm:* When updating the weight, the static graph algorithm needs to recalculate the weight of each node every time when there is an update. The complexity of this algorithm is $O(V^2)$. If the static result is cached and the new tip is added, by updating the already cached information, the complexity will be amortized.

## IV. Applications

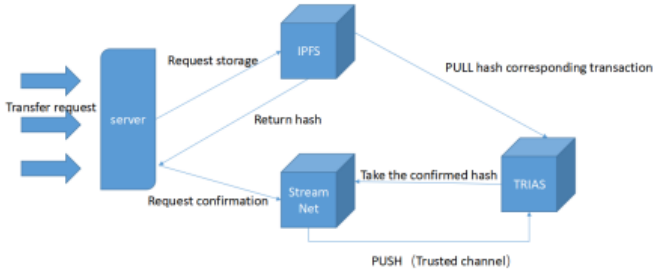


Figure 19. StreamNet shows the TRIAS transfer request cache

### A. 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 ability of elasticity. The structure is shown in Figure 19. There is a distributed transaction service. When an transaction request comes over, it first stores the information in the $IPFS$ and gets a hash. Then constructs node in StreamNet using this hash. When the traffic is small, StreamNet can directly push the hash of the transaction to $TRIAS$ after confirming it, so $TRIAS$ can get the specific transfer information from $IPFS$ and perform the consensus on the transaction. When the traffic is large, StreamNet will continue to confirm These transactions. When $TRIAS$ is idle, it will pull the confirmed hash

from StreamNet, and get the specific transfer information from $IPFS$ and perform consensus on the transaction.

## V. Experimental Results

## References


[1] "Confirmation rate in the tangle." [Online]. Available: https://blog.iota.org/confirmation-rates-in-the-tangle-186ef02878bb

[2] "Cryptographic vulnerabilities in iota." [Online]. Available: https://medium.com/@neha/cryptographic-vulnerabilities-in-iota-9a6a9ddc4367

[3] "Iota transactions, confirmation and consensus." [Online]. Available: https://github.com/noneymous/iota-consensus-presentation

[4] "On the tangle, white papers, proofs, airplanes, and local modifiers." [Online]. Available: https://blog.iota.org/on-the-tangle-white-papers-proofs-airplanes-and-local-modifiers-44683aff8fea

[5] "Pow on the tangle." [Online]. Available: https://docs.iota.org/introduction/tangle/proof-of-work

[6] A. Gal, "The tangle: an illustrated introduction iota," Jan 2018. [Online]. Available: https://blog.iota.org/the-tangle-an-illustrated-introduction-4d5eae6fe8d4

[7] L. Katz, "A new status index derived from sociometric analysis," *Psychometrika*, vol. 18, no. 1, pp. 39–43, 1953.

[8] E. Nathan and D. A. Bader, "Incrementally updating katz centrality in dynamic graphs," *Social Network Analysis and Mining*, vol. 8, no. 1, p. 26, 2018.

[9] S. Popov, "The tangle," *cit. on*, p. 131, 2016.