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

***Abstract*—To achieve high throughtput in the POW based blockchain systems, a series of methods has been proposed, and DAG is one of the most active and promising field. Considering major issues in the existing DAG systems such as centralization, double-spending, sybil attacks, and slow transaction speed etc. We designed and implemented the StreamNet system aiming to tackle with these problems. By leveraging the Topological sorting and Katz centrality computation, we infer a pivotal chain along the path of each epoch in the graph, and each block in the pivotal chain will have the highest Katz centrality score (as oppsed to the $GHOST$ rule). When attaching a new vertex in the DAG, two tips are selected. One is the 'parent' tip whose definition is the same as in $Conflux$ [20], another is using Markov Chain Monte Carlo ($MCMC$) technique by which the definition is the same as $IOTA$ [28], with Katz centrality as the weight determinator. In addtion, with comprehensive integration of information such as transaction (vertices), transaction approval information (edges), and network structure (such as community structure) a tip filtering algorithm can be optionally configured to detect and avoid double spend / sybil attacks and accelerate the transaction approval rate.**

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

I. Introduction

Ever since bitcoin [25] has been proposed, blockchain technology has been widely studied for 10 years. Extensive adoptions of blockchain technologies was seen in real world applications such as financial services with potential regulation challenges [23], [33], supply chains [17], [34], [5], health cares [6], [36] and IOT devices [8]. The core of blockchain technology resides in the consensus algorihtms applying to the real distrubuted computing world. Where computers can join and leave the network and these copmuters can cheat.

As the first protocol that can solve the so called Byzantine general's problem, bitcoin system suffers from the problem of low throughput with a transaction per second (TPS) of approximately 7, and long confirmation time (about an hour). As more and more machines joined the network, they are competing for the privileges to attach the block (miners) which result in huge waste of electric power. While sky rocketing fees are payed to make sure the transfers of money will be placed in the chain. On par, there are multiple proposals to solve the low transaction speed issue.

One method intends to solve the speed problem without changing the chain data structure, for instance, the bitcoin cash (BCH) fork of bitcoin (BTC) system tries to improve the throughput of the system by enlarging the data size of each block from 1 Mb to 4 Mb. To minimize the cost of POW, a proof of stake method POS [35] is proposed to make sure that only those who in the network can have the privilege to attach the block only if they have a large amount of token shares. Anohter idea targeting at utilizing the power in POW to do useful and meaningful tasks such as training machine learning models are also proposed [22]. In addition, inspired by the $PBFT$ algorithm [7] and a set of its relateted variations, so called hybrid chain was proposed. The general idea is to use two step algorithm, the first step is to elect a commiette, the second step is collecting committee power to employ $PBFT$ for consensus. Bitcoin-NG [12] is the early adoptor of this idea, which splits the blocks of bitcoin into two groups, one is for master election and another for regular transaction blocks. Honey-badger [24] is the system that firstly introducted the consensus commitee, it uses a predefined memebers to perform $PBFT$ algorithm to reach consensus. The Byzcoin system [16] brought forth the idea of POW for the commitee election, and uses a variation of $PBFT$ called collective signing for speed purposes. The Algorand [13] utilizes a random function to anonymously elect commetee and use this commitee to commit blocks, and the member of the commitee only have one chance to commit block. All these systems have one common feature, the split of layers of players in the network, which results in the complexity of the implementation of the system.

While aforemetioned methods are trying to avoid side chains, another thread of effort is put on using direct acyclic graph DAG to merge side chains. The first ever idea comes with growing the blockchain with trees instead of chains [31], which results in the well known $GHOST$ protocol [32]. If one block links to ≥ 2 previous blocks, then the data structure grows like a DAG instead of tree [29], [30], [19]. There is a improvement of the $GHOST$ based DAG algorithm which can achieve 6000 of TPS in reality [20]. Another set of methods tried to avoid finality of constructing a linear total order by introducing the probability of confirmation in the network [28], [9]. However, suffering from engineering issues, mainly due to the lack of transaction frquency and the growing complexity due to the network expansion. These system in reality are rely on centralized mehods to maintain their stability. Some of the side chain methods also borrows the idea of DAG, such as nano [18] and vite [21]

All of the current DAG systems used the idea of head counting method to infer main chains which does not consider the network structure. Dating back to the time $Google$ uses $PageRank$ [27] to sort importance of web page instead of number of references, and this method has achieved a huge susccess. And this method can also be utilized to help growing the DAG, in this paper, we use the Katz centrality metric [15]. Emerging social network research has introduced the method of streaming graph analysis [11], [14], [10] which deals with how to quickly maintain information on a temporally or spatially changing graph without traversing the whole graph. The main contribution of this paper is how to utilize the streaming graph analysis methods to bring the DAG systems into real decentralized, and stabilized growing system.

II. 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.

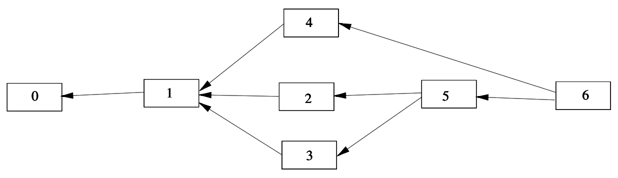
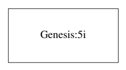


Figure 1. Example of the StreamNet data structure [?].

B. Transaction



Creation trading block

Figure 2. The creation of genesis node.

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.

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,

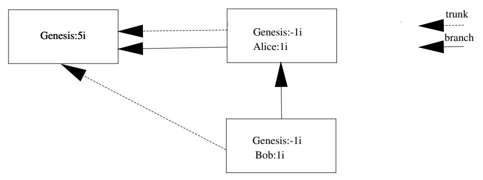


Figure 3. An example of token transfers.

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) [?], but this POW differs from Bitcoin in that its difficulty is fixed and therefore does not need the participation of miners.

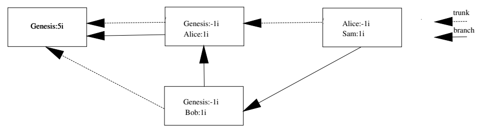


Figure 4. An example of succesfull token transfer.

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 [?]. For example, in Figure 4, the Alice $\rightarrow$ Sam transfer needs to validate transactions it indirectly or directly approves. It constructs a topological transfer sequence, namely (Genesis) $\rightarrow$ (Genesis $\rightarrow$ Alice) $\rightarrow$ (Genesis $\rightarrow$ Bob) $\rightarrow$ (Alice $\rightarrow$ 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 $\rightarrow$ 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 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.

4) Tip Selection: There are two basic concepts in StreamNet, one is the transaction rate λ, 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

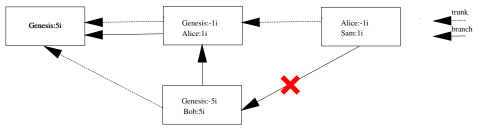


Figure 5. An example of errornous token transfer.

of h, the transaction rate λ 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.

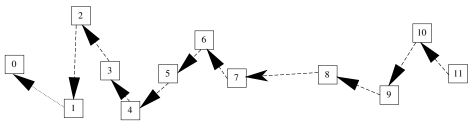


Figure 6. The shape of StreamNet when the txn rate is low.

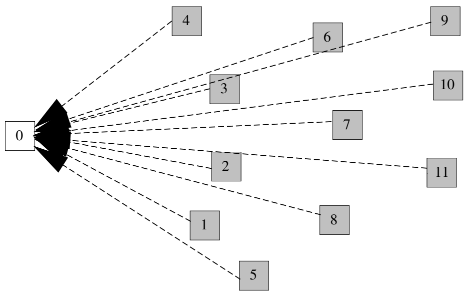


Figure 7. The shape of StreamNet when the txn rate is high.

One of simplest tip selection algorithms is the random walk with equal probability starting from Genesis, as shown in Figure 8. Suppose Alice $\rightarrow$ Sam transaction wants to select tip, which starts from Genesis transaction. There are two options, one is Genesis $\rightarrow$ Alice, the other is Genesis $\rightarrow$ Bob, the probability of selecting Genesis $\rightarrow$ Alice is $\frac{1}{2}$, while Genesis $\rightarrow$ Bob is $\frac{1}{2}$.

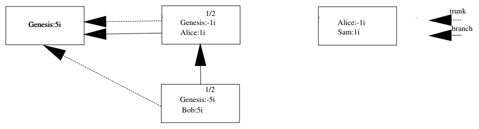


Figure 8. Random walk with equal probability.

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 $\rightarrow$ Alice is 8, and Genesis $\rightarrow$ Bob is 2, then the probability of selecting Genesis $\rightarrow$ Alice is $\frac{8}{10}$, and Genesis $\rightarrow$ 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 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.

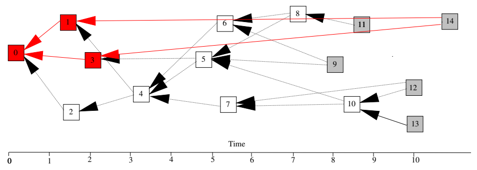


Figure 9. An example of lazy transaction.

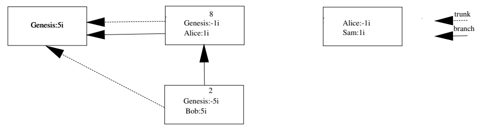


Figure 10. Random walk with equal probability.

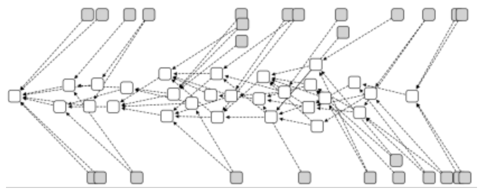


Figure 11. Example of the result using super-weight algorithm.

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

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

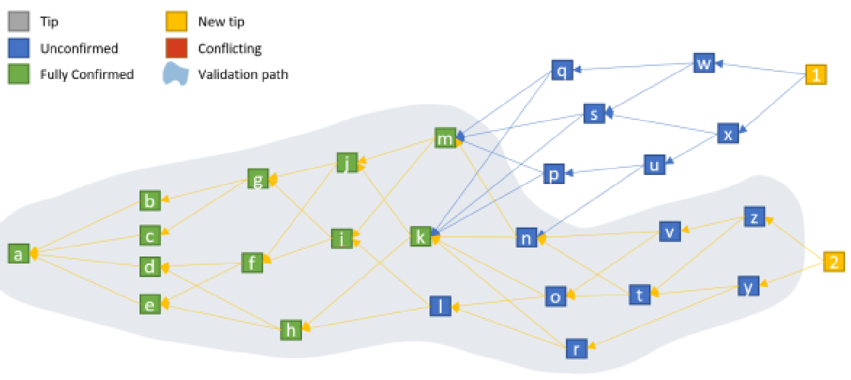


Figure 12. Example of fully confirmed transactions.

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.

III. 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 [28]. 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 λ, and the second is the randomness α. From Figure 15, it can be seen that the probability of increasing the success of the transaction is mainly due to the increase the λ and reduces α. and in Figure 15, a more intuitive expression of the effect of α can be seen in Fifure 16, under the same conditions, the higher α 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

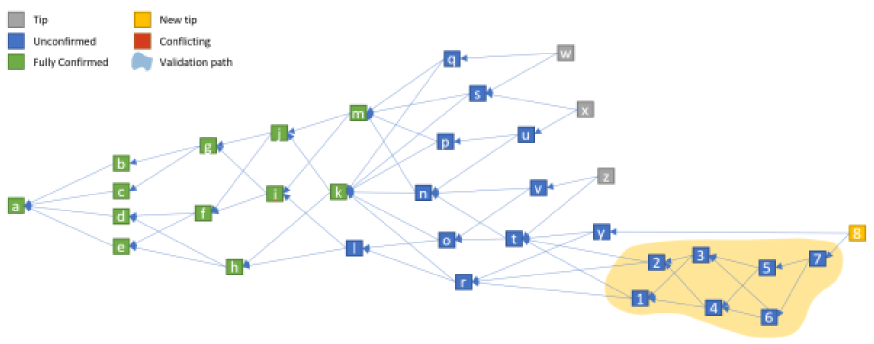


Figure 13. Example of offline update [3]

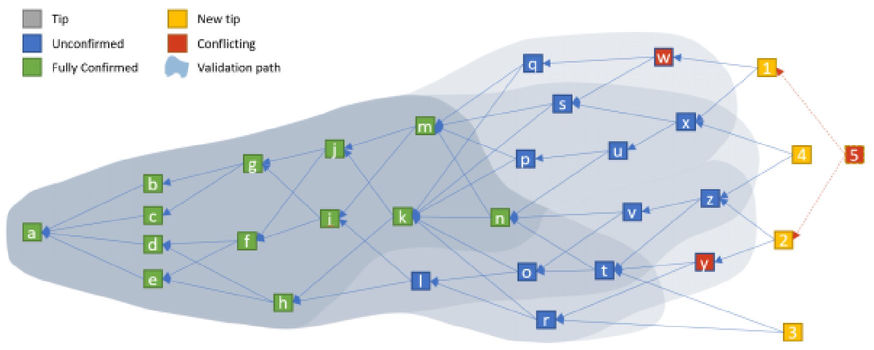


Figure 14. Example of double spending problem

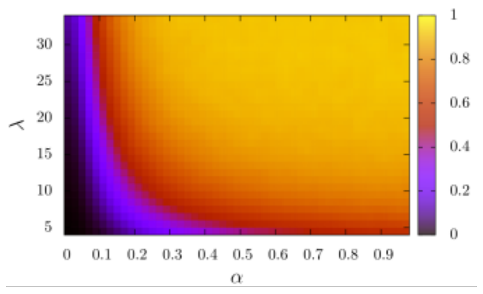


Figure 15. The probability of a transaction being permanently stuck (failed), α is randomness, λ is the transaction rate [1]

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.

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].

IV. STREAMNET MAIN ALGORITHM

A. *Storage*

B. *UTXO and Hash Functions*

1) Metrics for Pivotal Chain Selection: In Phantom [30] and Conflux [20], the $GHOST$ rule [32] is applied for selecting the pivotal chain. Here we introduce a new metric with Katz centrality [15] as the weighting criteria. In StreamNet, we use an Adjacency Matrix to represent the direct link relationship between blocks, 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 α 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 pivotal chain computed, then the complexity will be

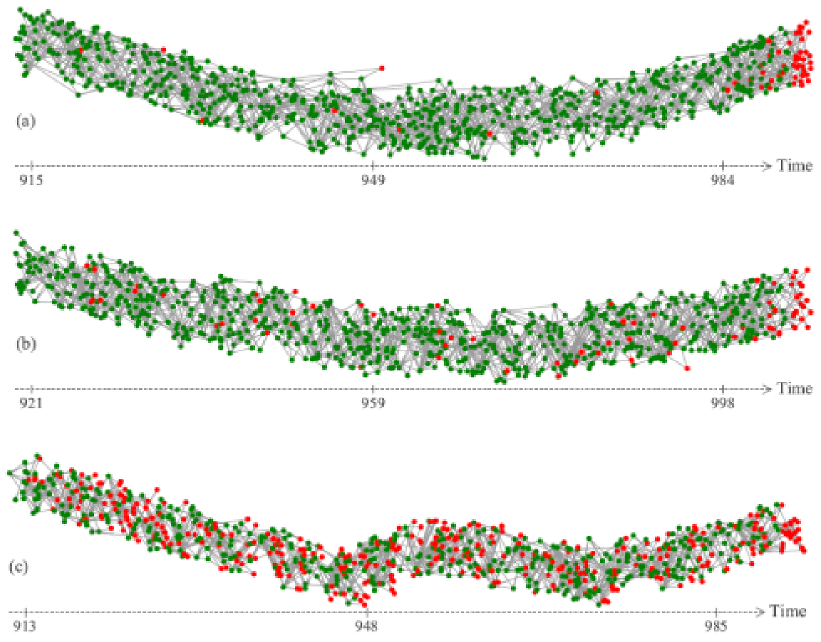


Figure 11: Graphical representation of the Tangle for $\lambda = 10$, $h = 1$ over 100 units of time, all using MCMC with (a) $\alpha = 0.05$, (b) $\alpha = 0.1$ and (c) $\alpha = 0.5$

Figure 16. A more intuitive expression of the role

intolerable. 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 [26].

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

C. DAG total ordering algorithm

D. Tip Selection Algorithm with Configurable Local Modifiers

In $StreamNet$, when one new block wants to be attached to the main network, it should find a parent block and a reference block. We call this procedure the tip selection method. The parent block is found by calling the total ordering algorithm and the reference block is found by calling the $MCMC$ from the entry point. The algorithm is as Algorithm 1 shows.

Algorithm 1: TIP SELECTION TIPSEL(G, B, S, W, d).

```
Input: Graph $G$, Block $B$, score $S$, random walker $W$,
       and depth $d$
Output: Parent tip $T_p$ and reference tip $T_r$
1 $totalOrderChain$ = totalOrder($G$, $B$) ;
2 $T_p$ = $totalOrderChain$.last() ;
3 $entryPoint$ = $totalOrderChain$.at($d$) ;
4 do
5     $T_r$ = $W$.Walk($G$, $S$, $entryPoint$) ;
6 while localModifier($T_r$) != $false$;
7 return $T_p$, $T_r$ ;
```

1) Entrypoint selection: When performing tip selection, an entry point is neccessary, and in $IOTA$ mainet, it will not start from the genesis transaction, but will simply start from a coordinator (COO) as the entry point. This will leads to a centralization problem. So one of the most important question we considered when designing StreamNet was how to remove 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. 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 block with this depth on pivotal chain.

2) Local Modifier 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. And in our tip selection algorithm, the local modifier is used for filtering malformed tips and the strategy is config-

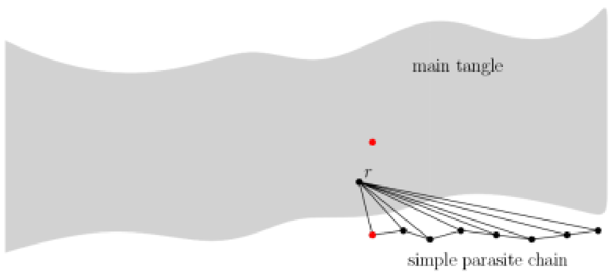


Figure 17. An example of simple parasite chain attack, a series of cheat transaction will refer one specific double spent transaction, when the SPC grows to a certain amount, the double spent will be successful, the two red node in the figure respents the conflict transactions.

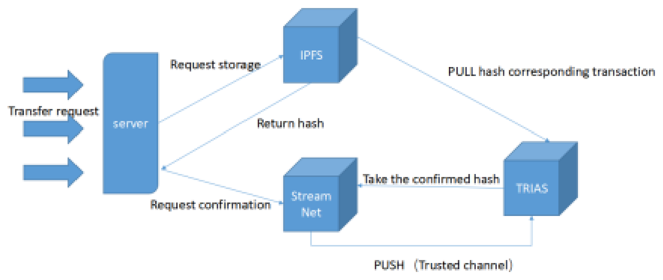


Figure 19. StreamNet shows the TRIAS transfer request cache

urrable. Here we discuss one of the local modifier 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 $(w1 + w2 + w3 + w6 + w1 \cdot w6)$.

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.

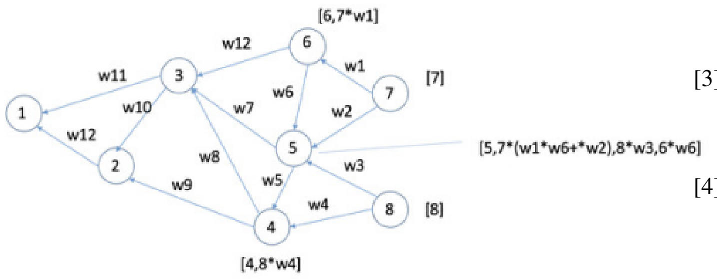


Figure 18. Weight calculation based on edge information.

V. Applications

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.

VI. 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] S. A. Abeyratne and R. P. Monfared, "Blockchain ready manufacturing supply chain using distributed ledger," 2016.

[6] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "Medrec: Using blockchain for medical data access and permission management," in *Open and Big Data (OBD), International Conference on*. IEEE, 2016, pp. 25–30.

[7] M. Castro, B. Liskov *et al.*, "Practical byzantine fault tolerance," in *OSDI*, vol. 99, 1999, pp. 173–186.

[8] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *Ieee Access*, vol. 4, pp. 2292–2303, 2016.

[9] A. Churyumov, "Byteball: A decentralized system for storage and transfer of value," *URL https://byteball. org/Byteball. pdf*, 2016.

[10] D. Ediger, R. McColl, J. Riedy, and D. A. Bader, "Stinger: High performance data structure for streaming graphs," in *2012 IEEE Conference on High Performance Extreme Computing*. IEEE, 2012, pp. 1–5.

[11] D. Ediger, J. Riedy, D. A. Bader, and H. Meyerhenke, "Tracking structure of streaming social networks," in *2011 IEEE International Parallel & Distributed Processing Symposium Workshops and PhD Forum*. IEEE, 2011, pp. 1691–1699.

[12] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-ng: A scalable blockchain protocol." in *NSDI*, 2016, pp. 45–59.

[13] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 2017, pp. 51–68.

[14] O. Green, R. McColl, and D. Bader, "A fast algorithm for incremental betweenness centrality," in *Proceeding of SE/IEEE international conference on social computing (SocialCom)*, 2012, pp. 3–5.

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

[16] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, "Enhancing bitcoin security and performance with strong consistency via collective signing," in *25th USENIX Security Symposium (USENIX Security 16)*, 2016, pp. 279–296.

[17] K. Korpela, J. Hallikas, and T. Dahlberg, "Digital supply chain transformation toward blockchain integration," in *proceedings of the 50th Hawaii international conference on system sciences*, 2017.

[18] C. LeMahieu, "Nano: A feeless distributed cryptocurrency network," *Nano [Online resource]. URL: https://nano. org/en/whitepaper (date of access: 24.03. 2018)*, 2018.

[19] Y. Lewenberg, Y. Sompolinsky, and A. Zohar, "Inclusive block chain protocols," in *International Conference on Financial Cryptography and Data Security*. Springer, 2015, pp. 528–547.

[20] C. Li, P. Li, W. Xu, F. Long, and A. C.-c. Yao, "Scaling nakamoto consensus to thousands of transactions per second," *arXiv preprint arXiv:1805.03870*, 2018.

[21] C. Liu, D. Wang, and M. Wu, "Vite: A high performance asynchronous decentralized application platform."

[22] S. Matthew and E. T. Nuco, "Aion: Enabling the decentralized internet," *Aion project yellow paper*, vol. 151, pp. 1–22, 2017.

[23] J. MICHAEL, A. COHN, and J. R. BUTCHER, "Blockchain technology," *The Journal*, 2018.

[24] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, "The honey badger of bft protocols," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 31–42.

[25] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[26] 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.

[27] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Tech. Rep., 1999.

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

[29] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "Spectre: Serialization of proof-of-work events: confirming transactions via recursive elections," 2016.

[30] Y. Sompolinsky and A. Zohar, "Phantom, ghostdag."

[31] ——, "Accelerating bitcoins transaction processing," *Fast Money Grows on Trees, Not Chains*, 2013.

[32] ——, "Secure high-rate transaction processing in bitcoin," in *International Conference on Financial Cryptography and Data Security*. Springer, 2015, pp. 507–527.

[33] A. Tapscott and D. Tapscott, "How blockchain is changing finance," *Harvard Business Review*, vol. 1, 2017.

[34] F. Tian, "An agri-food supply chain traceability system for china based on rfid & blockchain technology," in *Service Systems and Service Management (ICSSSM), 2016 13th International Conference on*. IEEE, 2016, pp. 1–6.

[35] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, pp. 1–32, 2014.

[36] X. Yue, H. Wang, D. Jin, M. Li, and W. Jiang, "Healthcare data gateways: found healthcare intelligence on blockchain with novel privacy risk control," *Journal of medical systems*, vol. 40, no. 10, p. 218, 2016.