Fast Tip Selection for Burst Message Arrivals on A DAG-based Blockchain Processing Node at Edge

†Xun Xiao, †‡Fengyang Guo, †Artur Hecker and ‡Schahram Dustdar
†*Munich Research Center, Huawei Technologies, Munich, Germany*
‡*Distributed Systems Group, TU Wien, Vienna, Austria*

GLOBECOM 2022 - 2022 IEEE Global Communications Conference | 978-1-6654-3540-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/GLOBECOM48099.2022.10001023

***Abstract*—With the rapid evolution of blockchain technology, a clear trend is that new blockchain systems (e.g., IOTA) tend to use a Directed Acyclic Graph (DAG) rather a chain structure to organize ledger records. Such a DAG-based blockchain system shows higher scalability as multiple locations are available in the ledger for new message attachment. To decide an attachment location, a popular type of tip selection algorithms follow an approach using weighted random walks on the DAG ledger. In a burst message arrival scenario, however, a processing node deployed at edge using such a method may become a bottleneck because sequentially repeating random walks significantly increases processing delay. In this paper, we propose a new tip selection algorithm for the burst message arrival scenario on an edge node. Our solution abandons the weighted random walk approach, instead, with similar efforts we transfer to calculate in advance the tip selection probability distribution of the DAG ledger. Such a new scheme reduces tip selection to a probability distribution sampling task, which can be done extremely fast. We implement our solution and demonstrate the benefits of our approach by comparing with the random walk approach. We believe our attempt can effectively mitigate the congestion at the edge node and inspire tip selection algorithm design with a new vision for DAG-based blockchain systems.**

***Index Terms*—Edge Blockchain Systems, Directed Acyclic Graph (DAG), Tip Selection, Random Walk,**

I. Introduction

A widespread adoption of blockchain technology is happening in various business sectors across FinTech [1], decentralized marketplace [2], and decentralized Applications (dApps) [3]. A blockchain system consists of distributed processing nodes, each of which maintains a copy of a common ledger. The ledger copy on every node is synchronized and temper-proof with a distributed consensus protocol. For the first time, blockchain technology makes information shared on Internet trustworthy in a decentralized manner.

A recent trend promotes using a *Directed Acyclic Graph* (DAG) to organize ledger records (e.g., IOTA [4]) on processing nodes, rather than a chain of transcation blocks widely used in traditional blockchains (e.g., Bitcoin [5] and Ethereum [6]). In a DAG ledger, a vertex represents a single message entry; and a directed edge represents an approval from the pointing vertex to the pointed vertex. Such a change brings several promising features e.g., multiple locations for new message attachment, lightweight consensus procedures, no miner/transaction fee and so on [7]. All these features make DAG-based blockchain systems easier to be integrated within edge/fog computing, thus bringing Internet-of-Things (IoT) applications closer to the end users [8].

On a processing node of a DAG-based blockchain system, a key processing logic is its *tip selection* module. A tip of a DAG ledger is a message without any approval. A node has to decide which tip(s) it shall approve by attaching a new message behind there. Such a decision-making process is nontrivial because the node has to make sure that: the selected tip(s) and all their connected vertices in the branches do not conflict with the new message; in addition, hopefully, the branches of the selected tip(s) can be re-selected with a higher chance afterwards, so that the new message itself can get approved earlier as well. The more (direct or indirect) approvals a vertex gets, the higher the weight the vertex earns. The *cumulative weight* is an important metric indicating how many times a message was repeatedly voted in history.

Many existing *Tip Selection Algorithm*s (TSAs) widely used on the processing nodes are designed based on the cumulative weight metric, where typically a random walk is simulated on a weighted DAG constructed from the ledger. Due to the heterogeneous weight distribution among vertices, a random walk will bias to some tips. The chance of a tip being selected reflects the collective opinion of all nodes cast in previous attachments. However, such an approach sometimes is inefficient as a processing node indistinguishably repeats random walks for tip selection of every single message. Particularly, when messages arrive on an edge node in a burst, due to a sudden long message queue and non-negligible time of doing random walks, the node might become a bottleneck in congestion, which postpones all the following phases, such as ledger consolidation and so on. Alas, this kind of burst scenarios are common in reality, e.g., crowds aggregating at a hot spot (like sport events or a public area during rush hours) to use a same service at one place. Obviously, a revisit is needed for such a scenario.

In this paper, motivated with the above concern, we reconsider the weighted random walk approach and try to seek a new solution for an edge node. The key idea of our solution is as follows: inspired by *Absorbing Markov Chain* (AMC) theory, we discover that the weighted DAG enjoys a nice property where the *Tip Selection Probability Distribution* (TSPD) can be calculated straightforwardly. We will see that pre-calculating the TSPD is worthy, because with the TSPD information, tip selections simply reduce to drawing random samples from the derived probability distribution, which not

only can be reused for multiple messages, but also can be done extremely fast. This completely avoids the tediously simulating random walks on the processing node. In summary, our contributions are listed below:

- We model the DAG ledger on a processing node as an AMC; we show that the stationary distribution of the modeled AMC represents a statistical outcome of sufficient random walks on the DAG, thus giving us directly the *Tip Selection Probability Distribution* (TSPD) of the DAG ledger;
- Based on the TSPD property, a *sampling*-based TSA is proposed for handling burst message arrivals on an edge node; the proposed TSA relies on a strategy of periodically updating the TSPD along with the evolving DAG ledger;
- We implement the proposed TSA and compare with the typical weighted random walk TSA. Evaluation results demonstrate that the proposed TSA can effectively mitigate Message Attachment Delay (MAD) in the burst message arrival scenario at edge node.

To the best of our knowledge, our work is the first to ask if there could be an alternative approach replacing the weighted random walk, especially considering a DAG-based blockchain processing node at edge.

This structure of the paper is outlined as follows. In Section II, we review the existing literature; in Section III, we formally introduce our AMC modeling; in Section IV, we introduce our sampling-based *Tip Selection Algorithm* (TSA); in Section V, we present our evaluation results and conclude this paper in Section VI.

II. Related Work

In this paper, we focus on a type of DAG-based blockchain systems where a vertex in the DAG represents a single message entry and will not develop to a multi-layer DAG topology. Exemplary systems are IOTA [4] and its variants, e.g., Graphchain [9] and Avalanche [10]. Note that there are many other graph-based blockchain systems such as Spectre [11] and Hashgraph [12]. For instance, Spectre [11] batches messages to blocks and then organizing as graphs. Nevertheless, those systems are of already a mixture with many extra components, thus considered out of scope in this work.

Within the interested scope, one line of research is to propose auxiliary strategies when doing weighted random walks for tip selection. For example, a TSA was proposed in G-IOTA [13], where the mechanism chooses three tips, the first two are selected by the weighted random walk and the third is selected from left behind tips. Later, E-IOTA introduced in [14] presented a mechanism to dynamically adjust system parameters controlling the random walk simulation to reduce the number of random walks.

Another line of research is to modify the vertex (edge) weight definition so that the random walks can achieve different purposes. For example, in [15], the authors proposed a new metric, called sharpness, to describe the extreme degree in a part of the DAG. Based on the new weight definition, the proposed algorithm aims to solve the splitting and fairness problem in IOTA. In [16], the authors gave a novel definition of message weight and time with integrating the information from IoT devices; after that, another TSA called best tip selection method (BTSM) was proposed to enhance the resistance to malicious attacks. Similarly, in [17], the authors studied a TSA optimization problem by using tree theory. They defined new labels on vertices for random walks in the DAG for tip selection and a dynamic tree will be maintained to improve the message validation efficiency.

Generally, the common ancestor of existing works is the TSA originating from IOTA [4]. Existing works are still under the framework of using a weighted random walk approach. Comparing to this, differently, our goal is to look for a new approach that does *not* require simulating weighted random walks for a processing node at edge.

Notation	*Meaning*
G_t	DAG ledger $\langle V, E\rangle$ at time t
G'_t	Sub-DAG ledger $\subset G_t$
n	Vertex size of the sub-DAG G'_t
v_i	A vertex (i.e., a message entry) in ledger G_t
e_{ji}	A directed edge from v_j to v_i (i.e., v_j approves v_i)
c_i	Cumulative weight (i.e., approval count) of v_i
$\tilde{V}$	Tip set in G_t, i.e., $c_i = 0$, $\forall v_i \in \tilde{V}$
w_{ij}	Edge weight e_{ij}, defined as $\lvert c_i - c_j\rvert$
s	Required number of tip selections for a new message
v_o	Random walk starting point/head point of G'_t
v_p	Tip vertex in G_t
p_{ij}	Jumping probability on edge e_{ij} (along reverse direction)
$\mathtt{P}_t$	Transition matrix of G'_t
$\tilde{\pi}^*_t$	Tip selection probability distribution over $\tilde{V}$ of G'_t
λ	Message arrival rate with a `Poisson` process
τ	Message window size

TABLE I: Main notations

III. Modeling Random Walk on DAG as AMC

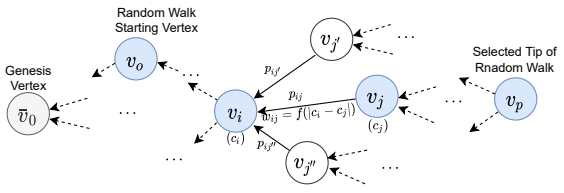

Fig. 1: Modeling weighted random walks on DAG ledger

In the following discussions, we use terms 'vertex' and 'message' interchangeably. For brevity, we also refer 'node' as the processing node of a DAG-based blockchain system deployed at an edge network. For a better readability, in addition, the main notations of this paper are summarized in Table I.

As shown in Fig. 1, we denote a DAG ledger on a node at time t as $G_t := \langle V, E\rangle_t$. A vertex $v_i \in V$ represents a message already in the ledger; a directed edge $e_{ji} \in E$

represents a direct approval from message v_j to message v_i. A directed edge e_{ji} also implies an indirect approval from v_j to all ancestor messages of v_i. G_t starts with a *genesis* vertex $\bar{v}_0$ (i.e., the leftmost gray vertex). The rest of vertices can be categorized into tip/non-tip vertices in terms of their in-degree values (i.e., the direct approval count). Non-tip vertices have their direct approval count greater than 0; and *tips* have zero approval count (e.g., the blue vertex v_p). We further denote the set of tips in G_t as $\tilde{V}$.

We denote v_i's *cumulative weight* c_i as the total count of direct and indirect approvals v_i earned at time t. This definition is also used in many existing systems such as IOTA. c_i characterizes the confidence of a message credited in the ledger. Based on c_i, we denote edge weight w_{ij} as the weight difference of its two endpoint vertices (i.e., $w_{ij} = |c_i - c_j|$). The jump probability p_{ij} from v_i to any of its direct approving vertices v_j is proportional to the edge weight w_{ij}. A weighted random walk tip selection starts at a certain vertex (e.g., the blue vertex v_o in Fig. 1), and jumps hop-by-hop with the jumping probability p_{ij} along the reverse direction of ingress edges towards a tip $v_p \in \tilde{V}$. For example, the highlighted blue vertices in Fig. 1 form a realized path of a random walk from v_o to v_p.

Notice that tips are always the final stop of a random walk because there is no edge for further jumps. This is equivalent to the *absorbing states* of an *Absorbing Markov Chain* (AMC), i.e., a Markov chain containing states with self-transition probabilities equal to 1 (e.g., the self-transition probability of v_p is 1). Actually, every random walk tip selection is a realization of state transitions on an AMC. Hence, the state transition on a sub-DAG G'_t (with a size n) starting with any v_o is fully characterized by a transition matrix $\mathtt{P}_t$ consisting of jump probabilities p_{ij} of all edges in G'_t including the self-transition probabilities of absorbing states.

An important property of an AMC is its *stationary distribution*, denoted as $\tilde{\pi}^*_t$, which tells the staying probability of every state of an AMC in the long run. On the one hand, since non-tip vertices are transient states where a random walk never stays, the staying probability of any non-tip vertex will be zero. On the other hand, only tips in $\tilde{V}$ will yield non-zero staying probabilities. If the random walk is repeated with a sufficient number of times, the selection probability of a tip is roughly equal to the proportion of occurrences where the random walks stop at the particular tip. Thus, the statistical outcome of sufficient times of random walks is equivalent to the stationary distribution $\tilde{\pi}^*_t$ of the corresponding AMC, which just tells the *Tip Selection Probability Distribution* (TSPD) of the DAG ledger at time t.

IV. A Sampling-based TSA for Edge Nodes

A. Main Idea

A processing node usually needs to select s tips for approval[1] (e.g., an IOTA node picks $s \in [2, 8]$). Thus, sk times' tip selections are needed for k messages in total. When k messages arrive at the node in a short time (i.e., a burst arrival), if the sk times' random walks are sequentially repeated for tip selections, this may significantly increase the Message Attachment Delay (MAD) on the node. For example, as illustrated by light blue blocks in Fig. 2, message v_4 can be processed only if all the previous random walks are done for the first three messages (i.e., v_1 to v_3). Clearly, congestion occurs due to the close and tight arrivals on the node.

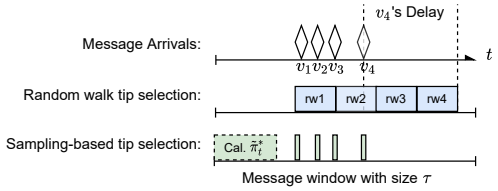

Fig. 2: Main idea illustration

Notice that here we exclude a trivial solution: parallelizing weighted random walk simulations on the node. As explained, here we consider a processing node deployed at edge, which might have limited resource onboard, e.g., a virtualized microservice instantiated in an edge/fog computing periphery. Therefore, arbitrarily parallelizing random walks on a resource-constrained node is not an easy option.

Given such a challenge, we were wondering if the TSPD $\tilde{\pi}^*_t$ of the DAG ledger G'_t can be known in advance; if so, the node can easily sample from $\tilde{\pi}^*_t$ for tip selection without doing random walk anymore. Intuitively, such a sampling-based TSA shall be faster because: 1) the TSPD is calculated only once but reused for multiple messages; and 2) repeated random walk simulations are completely avoided, largely shortening the MAD. This idea is illustrated by the light green blocks at the lower part in Fig. 2. The only question is: how to calculate the TSPD, which will be answered upon next.

B. Calculating TSPD $\tilde{\pi}^*_t$

The selection probability of a tip $v_p \in \tilde{V}$ equals the probability of realizing a random path leading v_o to v_p. Obviously, the random path can visit different sets of intermediate vertices (with or without overlaps) to reach the same tip v_p. For an intermediate jump, independently, it could be either a direct jump from $v_i \rightarrow v_j$ or an indirect jump $v_i \rightarrow [v_k] \rightarrow v_j$ via another vertex v_k. Assume the set of all such feasible v_k is $\hat{V}$, the jump probability of such a transition according to Champman-Kolmogorov equation [18] can be formally written as:

$$\begin{aligned} p(v_i|v_j) &= p_{\text{direct}}(v_i|v_j) + \sum_{v_k \in \hat{V}} p(v_i|v_k) \cdot p(v_k|v_j) \\ &= p_{ij} + \sum_{v_k \in \hat{V}} p_{ik} \cdot p_{kj} \ . \end{aligned} \tag{1}$$

Mathematically, Eq. (1) is the operation of the i-th row vector multiplying the j-th column vector of transition matrix $\mathtt{P}_t$ of G'_t. Hence, going over all rows and columns, the

[1]If two tip selections pick the same tip twice, then only one directed edge will be added.

transition probability of one-jump for all feasible cases can be calculated with a matrix product as below:

$$\mathtt{P}^{(1)} = \mathtt{P}_t \times \mathtt{P}_t = \mathtt{P}_t^2 \ . \tag{2}$$

Extending the one-jump transition probability in Eq. (2) to multiple jumps, we have:

$$\mathtt{P}^{(\ell)} = \underbrace{\mathtt{P}_t \times \cdots \times \mathtt{P}_t}_{\ell+1 \text{ terms}} = \mathtt{P}_t^{\ell+1} \ , \tag{3}$$

where ℓ is the path length of a random walk in G'_t. $\mathtt{P}^{(\ell)}$ in Eq. (3) gives the full transition probability after ℓ jumps from *any-to-any* vertices.

In our problem, the situation is much simpler because: i) the random walk always starts from a given vertex v_o (i.e., not an arbitrary vertex); and ii) the DAG ledger G'_t at any time is acyclic, thus the random walk stops in finite steps deterministically (i.e., no circle path and infinite jumps). With these nice features, we represent a random walk starting at v_o as a row vector π_0 (i.e., an initial state). π_0 has only the o-th element non-zero (value is 1) and its length is equal to n, i.e., the size of G'_t. Hence, with Eq. (3), the state transition from π_0 after ℓ jumps can be calculated by:

$$\pi^{\ell} = \pi_0 \times \mathtt{P}_t \times \cdots \times \mathtt{P}_t = \pi_0 \times \mathtt{P}_t^{\ell+1} \ . \tag{4}$$

The maximum value of ℓ is the maximum path length in the sub-DAG ledger G'_t, denoted by $\mathtt{L}$. Immediately, this gives the stationary distribution reaching any possible tip $\forall v_p \in \tilde{V}$ in G'_t as follows:

$$\tilde{\pi}_t^* = \pi_0 \times \mathtt{P}_t^{\mathtt{L}+1} \ . \tag{5}$$

$\tilde{\pi}_t^*$ given by Eq. (5) specifies the probability distribution arriving at a set of vertices after $\mathtt{L}$ jumps starting from a chosen point v_o. As mentioned, since the random walk only goes towards the tips, after $\mathtt{L}$ steps, this certainly covers the required number of jumps arriving at any other tip(s) that distance closer to v_o. Additionally, only the elements at the indices of tips (i.e., absorbing states) are non-zero in $\tilde{\pi}_t^*$, which is a known property of the stationary distribution of an AMC [19].

C. The Sampling-based TSA

Knowing the TSPD $\tilde{\pi}_t^*$ facilitates a node to quickly draw any required number of random samples from the distribution for tip selection. This can handle tip selections for multiple new messages rapidly because sampling is much faster than random walk, especially useful in a burst arrival scenario. However, this approach has to consider the fact that the DAG ledger G'_t is time-evolving after adding new messages. The topology change will also alter the tip set $\tilde{V}$ so as the transition matrix $\mathtt{P}_t$, thus $\tilde{\pi}_t^*$ too. In our proposed TSA, we introduce a *message window size* τ parameter to control the updating frequency of $\tilde{\pi}_t^*$, where after every $\tau\,s$, $\tilde{\pi}_t^*$ has to be updated in order to adapt with the latest DAG ledger topology change.

Our sampling-based TSA mainly consists of two modules: The first module (pseudo code in Algorithm 1) is a periodic TSPD update worker at the beginning of every message window. When calculating TSPD, accessing $\tilde{\pi}_t^*$ will be blocked until its updating is finished at the worker side. Locking the $\tilde{\pi}_t^*$ prevents conflict accessing from the sampling module. The second module (pseudo code shown in Algorithm 2) is the sampling routine for every new message v_m suspending in the message queue msgQ, according to the derived $\tilde{\pi}_t^*$ periodically updated by the TSPD worker in Algorithm 1.

Algorithm 1 TSPDUpdateWorker

Input: $\mathtt{P}_t$

while Timer(τ) is up **do**
  Lock $\tilde{\pi}_t^*$ ▷ *Preventing from conflict access*
  $\tilde{\pi}_t^* \leftarrow$ calculateTSPD($\mathtt{P}_t$) with Eq. (5)
  Unlock $\tilde{\pi}_t^*$ ▷ *Release for accessing*
  Reset τ Timer
end while

Algorithm 2 SamplingRoutine

Input: msgQ

loop
  if msgQ.IsNotEmpty() AND $\tilde{\pi}_t^*$ is unlocked **then**
    $v_m \leftarrow$ msgQ.Dequeue()
    $\tilde{V}_m \leftarrow$ samplingFrom $\tilde{\pi}_t^*$ for v_m
    Update $\mathtt{P}_t$ based on $G_t \cup \{v_m, E_{v_m,\tilde{V}_m}\}$
  end if
end loop

D. Remarks

First, calculating Eq. (5) practically is not time consuming. The reasons are: with a specified starting point v_o (i.e., π_0 vector), Eq. (5) reduces to a vector-matrix product. This is much faster than matrix-matrix product operations, in both time and space complexities; in addition, for a DAG, the transition matrix $\mathtt{P}_t$ is always an upper-triangle sparse matrix, thus the actual complexity of the sparse vector-matrix products in Eq. (5) is much lower than normal dense matrix multiplications. In our evaluation, its cost is slightly more than a single time random walk simulation.

Second, although calculating a TSPD consumes slightly more time, such overheads pay off because the TSPD information can benefit to the tip selections of following messages dropping into the same message window while the weighted random walk cannot. The delay can be largely mitigated after knowing the TSPD because a rapid sampling from the TSPD $\tilde{\pi}_t^*$ replaces the random walk for every message (as illustrated by the blue blocks in Fig. 2).

Third, the proposed TSA is backward compatible because it is an optimization to the local tip selection module on one node, which only relies on the information already available from the node and does not require any external interaction with neighboring nodes. It is not mandatory to have a full installation of our TSA to the whole DAG-based blockchain system.

Last but not least, the proposed TSA does not touch the principle of tip selection. An invalid message is treated in

the same way, i.e., drop and re-sample a tip until a valid one is identified. In other words, the proposed TSA considers the efficiency issue when selecting a tip, thus neutral to all following stages.

V. Evaluation Results

We implemented our sampling-based TSA (labeled as 'Sampling') in `Python` and compared with the typical weighted random walk TSA (labeled as 'RW'). In all evaluation tests, we set our sub-DAG G'_t size $n = 1000$, and we set $s = 3$ (i.e., three tips have to be selected for each message).

A. Message Attachment Delay (MAD)

Given three different scales of the message arrival rate λ, we first evaluated the two methods with Message Attachment Delay (MAD) defined as $\delta t_i = t_i^c - t_i^a$, where t_i^a is the time a message enters the message queue and t_i^c is the time its three tips are selected. For each λ, we repeated the test $K = 50$ times and in each time we randomly generated $m = 2000$ messages. We are concerned with the median, mean and worst cases of MAD with the two methods. The results are shown in Fig. 3.

When the arrival rate is low ($\lambda = 10$), the two methods have similar performances as shown in Fig. 3a's column. For the median MAD, both methods could make half of the traffic loads experience MAD around $0.04\,s$, where the Sampling method performed slightly better; for the worst case, few more numbers of tests with our method showed longer MAD (prolonging to the $[1\,s, 10\,s]$ interval), this also worsened the mean MAD of some tests with our method. As expected, the proposed Sampling method does not enormously advantage in a low arrival rate scenario due to similar costs of updating TSPD $\tilde{\pi}_t^*$ once and doing a single random walk.

However, when the arrival rate increases (e.g., $\lambda = 40$), as shown in Fig. 3b's column, the performance of the RW method severely degraded, where half of the traffic loads (i.e., the median case) experienced their MADs in between $10\,s$ and $50\,s$; for the worst case, there were 20% of tests experiencing MAD $> 50\,s$. Instead, the proposed Sampling method did not degrade. Clearly, with a burst arrival, more messages dropped in the same message window, and thereby could reuse the calculated TSPD $\tilde{\pi}_t^*$ for tip selection by sampling. Similarly, when the arrival rate doubled to $\lambda = 80$, as shown in Fig. 3c's column, the median and mean cases of MAD with the RW method prolonged to the interval of $[20\,s, 50\,s]$ and its worst case even prolonged beyond $100\,s$ in some tests. Instead, our method showed that only less than 20% of tests in the worst case prolonged beyond $1\,s$ but still less than $10\,s$.

The MAD evaluations clearly confirmed our motivation and the key benefit of the proposed sampling-based TSA, where the processing delay at the node can be effectively mitigated especially in a burst message arrival scenario.

B. Total Processing Time

We then evaluated the total processing time T consumed by the two methods for the tip selections for all new messages, given three different message window size τ values $0.02\,s$, $0.04\,s$ and $0.06\,s$. Similarly, for each τ value, we repeated the tests $K = 50$ times and each test processed $m = 2000$ messages but with a fixed arrival rate $\lambda = 40$. The evaluation result is shown in Fig. 4.

First, our proposed TSA consumed much less time to finish tip selection for all messages than the RW method did (see the three cold-color curves are all at the left-hand side of the orange curve). Specifically, with our method, nearly 90% of the tests consumed around $43\,s$, $30\,s$ and $23\,s$, respectively to finish the entire jobs. In contrast, 80% of tests with RW method consumed $70\,s$ to $90\,s$ and the rest took up to $150\,s$. This confirms that with the message window size τ increasing, the frequency of updating the TSPD of every message window became less often, while the chances of reusing a derived TSPD increased gradually. This also shows the influence of the message window size τ.

The evaluation results of the two different metrics above confirm the idea of the proposed Sampling method, which says that paying slightly more efforts to calculate TSPD $\tilde{\pi}_t^*$ is definitely beneficial. They also confirm that simulating weighted random walk is not imperative to tip selection in DAG-based blockchain systems. Instead, we do have a better approach to achieve the same goal instead.

C. Features of the Proposed TSA

Last, we are also interested in the features of the proposed Sampling method. Our Sampling method pays main efforts on periodically updating a TSPD $\tilde{\pi}_t^*$, which is different to the RW method where all time for tip selection is mainly spent on random walks. Therefore, it is helpful to quantitatively measure the time proportion of the two modules (i.e., Algorithm 1 and Algorithm 2). The result is shown in Fig. 5 still with an increasing τ value.

We can first notice that the proportion of time spent on calculating TSPD $\tilde{\pi}_t^*$ indeed dominates, comparing with the time for sampling tips (i.e., the forward slash bars are much higher than the purple bars). This again reflects the key idea of the proposed TSA, where if the TSPD can be known, the tip selection is easier. Secondly, we can find out that when the message window size τ increases, the number of times updating the TSPD $\tilde{\pi}_t^*$ (i.e., the blue point clouds) decreases from 2000 (i.e., no reuse at all) to around 450 times. This also matches our expectation where the larger the message window size τ, the less frequent TSPD updates will be.

As an initial attempt, clearly, many other interesting aspects are not covered in this work, such as impacts to the DAG topology evolution and so on, which is being undertaken as ongoing work.

VI. Conclusion

In this paper, we focused on the tip selection module of a DAG-based blockchain processing node. Instead of following the existing approach using weighted random walks, we proposed a different strategy that pre-calculates the TSPD of the DAG ledger then sampling for tip selection. Evaluation

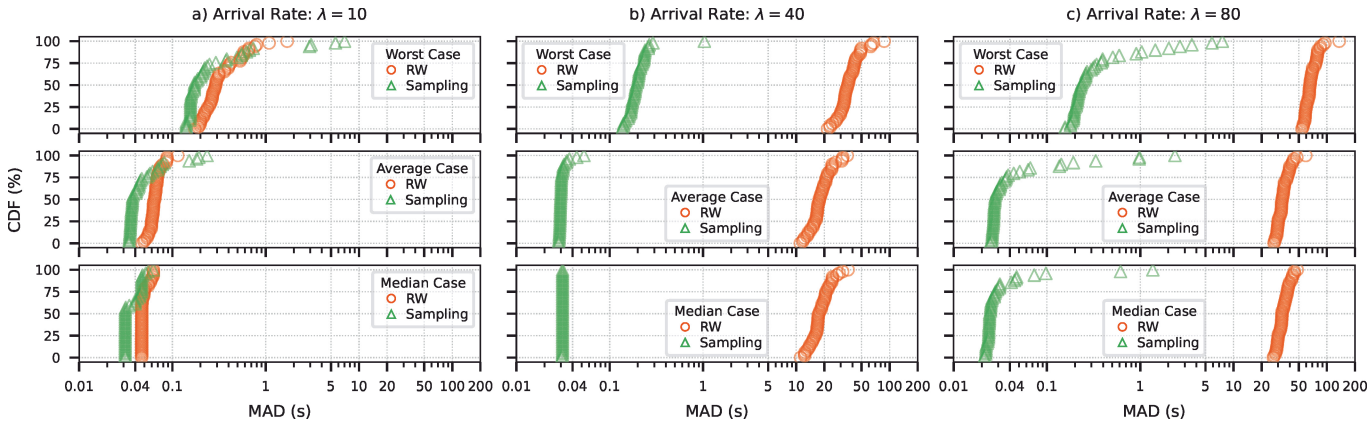

Fig. 3: Performance comparisons on Message Attachment Delay (MAD) δt_i ($\tau = 0.06\,s$)

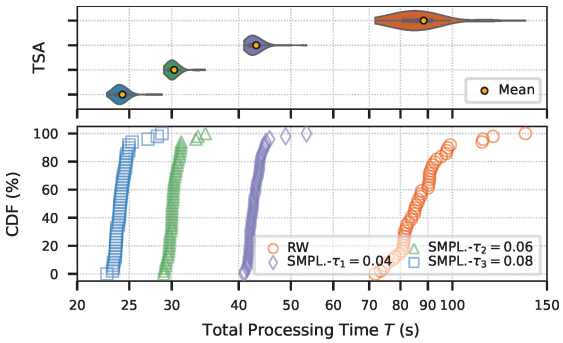

Fig. 4: Comparison on total processing time ($\lambda = 40$)

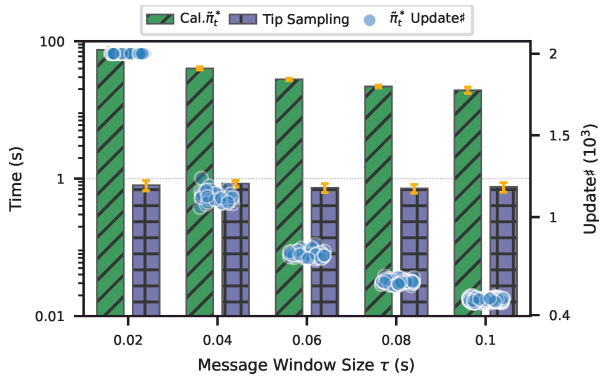

Fig. 5: Time ratio of the two modules in Sampling method ($\lambda = 40$)

results confirm that the proposed TSA can largely mitigate Message Attachment Delay (MAD) at the edge node facing burst arrivals. We believe that our new approach can further enlighten a set of new TSAs for similar blockchain systems.

References

[1] S. Fosso Wamba, J. R. Kala Kamdjoug, R. Epie Bawack, and J. G. Keogh, "Bitcoin, blockchain and fintech: a systematic review and case studies in the supply chain," *Production Planning & Control*, vol. 31, no. 2-3, pp. 115–142, 2020.

[2] H. Subramanian, "Decentralized blockchain-based electronic marketplaces," *Communications of the ACM*, vol. 61, no. 1, pp. 78–84, 2017.

[3] C. Antal, T. Cioara, I. Anghel, M. Antal, and I. Salomie, "Distributed ledger technology review and decentralized applications development guidelines," *Future Internet*, vol. 13, no. 3, 2021. [Online]. Available: https://www.mdpi.com/1999-5903/13/3/62

[4] S. Popov, "The tangle," *White paper*, vol. 1, p. 3, 2018.

[5] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, p. 21260, 2008.

[6] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[7] X. Fu, H. Wang, and P. Shi, "A survey of blockchain consensus algorithms: mechanism, design and applications," *Science China Information Sciences*, vol. 64, no. 2, pp. 1–15, 2021.

[8] Y. Wu, H.-N. Dai, and H. Wang, "Convergence of blockchain and edge computing for secure and scalable iiot critical infrastructures in industry 4.0," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2300–2317, 2020.

[9] X. Boyen, C. Carr, and T. Haines, "Graphchain: A blockchain-free scalable decentralised ledger," in *Proceedings of the 2nd ACM Workshop on Blockchains, Cryptocurrencies, and Contracts*, 2018, pp. 21–33.

[10] T. Rocket, M. Yin, K. Sekniqi, R. van Renesse, and E. G. Sirer, "Scalable and probabilistic leaderless bft consensus through metastability," *arXiv preprint arXiv:1906.08936*, 2019.

[11] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "Spectre: A fast and scalable cryptocurrency protocol," *Cryptology ePrint Archive*, 2016.

[12] L. Baird, "The swirlds hashgraph consensus algorithm: Fair, fast, byzantine fault tolerance," *Swirlds Tech Reports SWIRLDS-TR-2016-01, Tech. Rep*, vol. 34, 2016.

[13] G. Bu, Ö. Gürcan, and M. Potop-Butucaru, "G-IOTA: Fair and confidence aware tangle," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Apr. 2019, pp. 644–649.

[14] G. Bu, W. Hana, and M. Potop-Butucaru, "E-IOTA: An efficient and fast metamorphism for IOTA," in *2020 2nd Conference on Blockchain Research Applications for Innovative Networks and Services (BRAINS)*, Sep. 2020, pp. 9–16.

[15] J. Wang, J. Yang, and B. Wang, "Dynamic balance tip selection algorithm for iota," in *2021 IEEE 5th Information Technology,Networking,Electronic and Automation Control Conference (ITNEC)*, vol. 5, 2021, pp. 360–365.

[16] M. N. Halgamuge, "Optimization framework for best approver selection method (basm) and best tip selection method (btsm) for iota tangle network: Blockchain-enabled next generation industrial iot," *Computer Networks*, vol. 199, p. 108418, 2021.

[17] H. Wang and Z. Zhang, "A tsgp-based tip search optimization algorithm," in *2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*. IEEE, 2019, pp. 424–427.

[18] J. Karush, "On the chapman-kolmogorov equation," *The Annals of Mathematical Statistics*, vol. 32, no. 4, pp. 1333–1337, 1961.

[19] J. G. Kemeny and J. L. Snell, *Finite Markov chains: with a new appendix" Generalization of a fundamental matrix"*. Springer, 1983.