DAGsim: Simulation of DAG-based distributed ledger protocols

Manuel Zander
Department of Computing
Imperial College London
London, UK
manuel.zander17@ic.ac.uk

Tom Waite
Department of Computing
Imperial College London
London, UK
thomas.waite14@ic.ac.uk

Dominik Harz
Department of Computing
Imperial College London
London, UK
d.harz@ic.ac.uk

ABSTRACT

Scalability of distributed ledgers is a key adoption factor. As an alternative to blockchain-based protocols, directed acyclic graph (DAG) protocols are proposed with the intention to allow a higher volume of transactions to be processed. However, there is still limited understanding of the behaviour and security considerations of DAG-based systems. We present an asynchronous, continuous time, and multi-agent simulation framework for DAG-based cryptocurrencies. We model honest and semi-honest actors in the system to analyse the behaviour of one specific cryptocurrency, IOTA. Our simulations show that the agents that have low latency and a high connection degree have a higher probability of having their transactions accepted in the network with honest and semi-honest strategies. Last, the simulator is built with extensibility in mind. We are in the process of implementing SPECTRE as well as including malicious agents.

Categories and Subject Descriptors

C.2 [**Networks**]: Network types—*Peer-to-peer networks*

General Terms

Design, Experimentation, Measurement, Performance, Security, Standardization, Theory

Keywords

cryptocurrency, blockchain, distributed public ledger, directed acyclic graph, simulation

1. INTRODUCTION

Bitcoin introduced the possibility to transact monetary value between parties without the need to trust an intermediary and provides for the prevention of censorship [Nakamoto 2008]. A combination of consensus algorithms, cryptographic proof, and game theory allows mutually distrusting participants to interact on distributed ledgers. However, eliminating trust requirements comes with trade-offs. One of the main obstacles for cryptocurrency adoption is its limited scalability of a few transactions per second while being independent of trusted third parties [Sompolinsky and Zohar 2013, Decker and Wattenhofer 2013, Bamert et al. 2013]. Several proposals have been made to increase the scalability of distributed ledgers [Croman et al. 2016], including alternative data structures like directed acyclic graphs (DAGs) [Lewenberg et al. 2015, Sompolinsky and Zohar 2018].

Symposium on Cryptocurrency Analysis (SOCCA) 2018 Toulouse, France

DAGs are proposed as a generalisation of blockchains, e.g. [Sompolinsky et al. 2016, Sompolinsky and Zohar 2018, Popov 2016, Baird 2016]. Instead of having a sequence of blocks, blocks or transactions can exist in parallel. The network reaches consensus not about a single next block, but over a range of several blocks. Yet, DAG-based cryptocurrencies are little understood. Most DAG-based systems have not been formally analysed, for example with regard to the asynchronous communication model by Pass et. al. [Pass et al. 2017] or as extension of the formal framework by Kiayias and Panagiotakos [Kiayias and Panagiotakos 2016]. Meshcash, a recent work by Bentov et. al. [Bentov et al. 2017], is an exception.

Contribution Simulations help to verify formal work and are useful to check analytical calculations. We contribute an asynchronous, continuous time, and multi-agent simulator of DAG-based cryptocurrencies. Specifically, we implement our simulator in Python with extensions for various DAG-based protocols in mind. In our current version we implement an asynchronous representation of IOTA [Popov 2016] and an initial, yet incomplete, version of SPECTRE [Sompolinsky et al. 2016]. We evaluate the performance of our simulator and note that we depend on the computational complexity of the algorithms we simulate. Further, we present results of attachment probabilities (i.e. having transactions accepted by the network) in relation to the latency and degree of connections of an agent in the IOTA network.

Outline Our article is structured as follows. We explain our simulation framework in section 2. Section 3 details our results *w.r.t* simulator performance and the IOTA protocol. We present future work and conclude in section 4.

2. SIMULATION FRAMEWORK

The framework enables the simulation of arbitrary scenarios, and evaluates whether the implemented consensus algorithms have the desired effects in terms of stability and security. Adversaries deviating from a protocol are currently not implemented.

2.1 A Network of Agents

Nodes are called *agents* in our framework, and multiple agents are supported by default. Pass et. al. [Pass et al. 2017] propose a way of modelling blockchain protocols in a fully asynchronous setting, including an arbitrary number of players with identical computational power. This is superior to models assuming fully or partially synchronous network channels, which is a strong assumption (the Internet and most peer-to-peer (P2P) networks are essentially asynchronous networks with network latencies). Thus, we call the framework an asynchronous multi-agent simulation frame-

work, generally supporting $N \in \mathbb{N}$ agents, and each newly incoming transaction is issued by an agent i.

Along the lines of [Gal and Shikhelman 2018], agents can be thought of being part of an agent network, which is a weighted undirected graph. We define the network $G \equiv (A, E, f)$ where A is the set of agents, $E \subseteq A \times A$ is the set of edges and $f : A \times A \to \mathbb{R}$ is the weight function. We enforce $(a_1, a_2) \in E \Leftrightarrow (a_2, a_1) \in E$ (because G is undirected) $f(a,b) \geq 0$ (non-negative weights), $f(a,a) = 0$ (zero self-distance), and extend f such that $f(a,b) = \infty \Leftrightarrow (a,b) \notin E$.

Essential for implementing an asynchronous model are the network latencies between agents, which are stored in a distance matrix d. This matrix can be obtained by considering the shortest paths between agents $d(a_1, a_2)$, thus the distance function $d\ A \times A \to \mathbb{R}$ is defined as: $d(a_1, a_2) \equiv \min_{P \in P(a_1 \to a_2)} \sum_{e \in P} f(e)$, where $P(a_1 \to a_2)$ is the set of all paths from a_1 to a_2. We then obtain a $N \times N$ distance matrix d over $\mathbb{R}$, such that the following holds:

- For all $i : d_{ii} = 0$
- For all $i \neq j : d_{ij} > 0$
- For all $i, j, k : d_{ij} + d_{jk} \geq d_{ik}$

2.2 The DAG model

There are two classes of DAGs we have considered for our simulator. Vertexes in the graph can either transactions (i.e. IOTA [Popov et al. 2017]) or blocks that contain multiple transactions (i.e. SPECTRE and PHANTOM [Sompolinsky et al. 2016, Sompolinsky and Zohar 2018]). We refer to the first class as *txDAG* and the second class as *blockDAG*. The DAG data structure built up during a simulation can be defined as a stochastic process:

Let $\mathscr{G}$ be the set of all DAGs $G = (V, E)$ with the following properties:

- G is finite and the root of the DAG is called genesis G, such that $\deg_{\text{out}}(v) \geq 1$ for all $v \in V \backslash \{\text{G}\}$, and $\deg_{\text{out}}(\text{G}) = 0$.
- Any $v \in V$ such that $v \neq \text{G}$ references G.

At time $t \geq 0$ the state of the DAG ledger is $\mathscr{L}(t) \equiv (V(t), E(t))$, where $V(t)$ is the set of vertices and $E(t)$ is the multi-set of directed edges. Furthermore, the following dynamics hold:

- The initial state of the DAG at $t = 0$ is defined by $V(0) = \{\text{G}\}, E(0) = \emptyset$.
- Over time, the DAG grows: if $0 < t_1 < t_2$ then $V(t_1) \subseteq V(t_2)$ and $E(t_1) \subseteq E(t_2)$.
- Transactions arrive with a rate $\lambda > 0$ according to a Poisson process. They are either included in a block (i.e. blockDAG) or become vertices of the DAG (i.e. txDAG).
- A miner attaches a vertex v to the graph (i.e. a transaction or a block). The consensus protocol votes on accepting the vertex. If a vertex arrives at time t, then $V(t+) = V(t) \cup \{v\}$ and $E(t+) = E(t) \cup \{(v, v')\}$.

2.3 Design and Implementation Details

There are two main methods being executed during a simulation, `setup()` and `run()`. `setup()` initialises `Agent` and `Transaction` instances, whereas `run()` iterates over all transaction instances except for the genesis G in the order of their arrival times. This is the main simulation loop, i.e. the arrival of each transaction is one event. Arrival times of incoming transactions are sampled from the exponential distribution according to the rate of incoming transactions λ, which follows a Poisson process.

Together with a parameter h, defining the average computational cost for transaction propagation (mainly POW), the network latencies stored in d are crucial in deciding of which part of the network is visible for an agent at a certain point in time (during the simulation). As explained above, we assume that agents have the same computational power, such that we can fix $h = 1$ for simplicity.

Note that the presented Algorithm 1 deals with the creation of the DAG data structure and that the method called in line 11 differs for every protocol, e.g. IOTA has a different method of adding transactions to the DAG as opposed to other DAG-based protocols such as SPECTRE. The methods to determine the part of the ledger that the agents agree on as the *true state* are not included in this algorithm, because different protocols use different ideas and methods to resolve this question. Besides, how confirmed a transaction is does not directly play a role when the ledger is formed but is of interest for a merchant who needs to decide whether or not to accept a transaction. Consequently, this is not of interest during the simulation, but can be done after `run()` is finished, together with any other methods concerned with the analysis of the resulting ledger. More important for correct functioning of the asynchronous simulation framework is that an agent in every iteration of the main simulation loop deals with the currently visible part of the DAG, i.e. it is bounded by the network latencies contained in d.

It is also important to note that when this simulation framework is extended to blockDAG-protocols, such as SPECTRE, then transaction instances in this algorithm are instead blocks. Each block contains a list of transactions.

2.4 Current limitations

One of the main limitations of the currently implemented simulation framework is that it assumes honest and semi-honest agents, as opposed to specifically including malicious agents as well. These honest and semi-honest agents might influence the build DAG structure within the range of the simulated protocol only. However, the simulator is built in a flexible and modular way, which would make future extensions possible. Thus, in future versions, malicious agents could be included, and scenarios of specific attacks against the network could be modelled, where these agents purposely break the protocol and use different consensus algorithms.

3. RESULTS

3.1 Performance results

The efficiency of the framework determines at which scale simulations can be performed. IOTA was simulated with 5000 transactions [Zander 2018], using the Python module cProfile to generate profiling statistics of the simulation. The simulation is completed in 10 minutes 14 seconds on an i7 processor with 8 threads and 16 GB RAM. Significantly, it was found that the process of updating the cumulative weights [IOTA Foundation 2018b] took the largest amount of time - responsible for 61.81% of the run-time [Zander 2018]. This analysis, together with the investigation of the effect of different weight update algorithms, allows us to conclude that the simulator efficiency is fundamentally restricted by the computational complexity of the underlying weight update algorithm. As outlined in the IOTA documentation [IOTA Foundation 2018b, Gal 2018], the run-time complexity is $\mathscr{O}(n^2)$.

In practice, the IOTA implementation avoids this efficiency problem through the use of milestones [IOTA Foundation 2018b]. These are checkpoints in the ledger past that define a sub-graph upon which the tip selection process, and as a result the weight update

Algorithm 1: Setup and run of a simulation

Input : Number of transactions x, number of agents N, transaction rate λ, POW parameter h, distance matrix d

Output: DAG data structure with all transactions

procedure setup()
  Create set of agents A, consisting of N agents
  Generate x inter-arrival times $\tau_1, \ldots, \tau_x \sim \text{Exp}(\lambda)$
  Initialise $V(t)$ with arrival times: at_v for each $v \in V(t)\backslash\{\text{G}\}$ is given as $at_1 = \tau_1$ and $at_i = at_{i-1} + \tau_i$
  $at_{\text{G}} = 0$
end

procedure run()
  foreach $v \in V(t)\backslash\{\text{G}\}$ **do**
    Choose issuing agent $i \in A$ with uniform probabilities
    Determine which transactions are visible and valid for agent i according to h and d
    Add v to DAG according to chosen protocol
  end
end

process, is applied. However, this relies on a centralised and trusted entity called the coordinator [IOTA Foundation 2018a].

It is anticipated that our investigations of SPECTRE will yield similar efficiency results, as it is noted that the published core block wise ordering algorithm has a run-time complexity of $\mathcal{O}(n^3)$ [Sompolinsky et al. 2016]. Potentially, we can reduce the complexity and optimise resources in the simulation by holding intermediary computation steps in memory with frameworks such as Apache Spark [Zaharia et al. 2016].

3.2 IOTA protocol

In this section we present results obtained by using our multi-agent simulation framework to simulate the IOTA protocol, first presented in [Zander 2018]. The created DAGs contain 10,000 transactions. We generate random sparsely connected agent networks and examine attachment probabilities per agent, a measure concerned with the relative "importance of agents", i.e. telling us how likely the transactions issued by an agent are being part of the consensus part of the formed ledger (accepted by the network). We show the development of this measure over time (during a simulation) and how it is affected by the network latencies between agents. Sparse networks are commonly found in practice and have a number of links L between nodes N (here agents) *much* smaller than the maximal possible number of links L_{max}. Similarly to [Sompolinsky et al. 2016], we generate a Erdős-Rényi random network topology [Erdos and Rnyi 1960] with $N = 10$ agents for the experiments.

Quantifying heaviness The IOTA protocol uses a Markov Chain Monte Carlo algorithm as core of the consensus mechanism [Popov 2016]. When a new transaction comes in and the protocol specifies a transaction to attach to for the incoming one, a weighted random walk is used. This walk is biased towards heavy transactions or branches of the ledger. Heaviness is a measure specifying how many other transactions are directly or indirectly referencing any transaction in the DAG (the integer storing this is called cumulative weight). The larger the cumulative weight, the more likely a transaction is to be part of the main, agreed-on branch of the DAG (over time). We can measure the following, at any specified iteration of the simulation: How likely are the currently unreferenced transactions (called tips) of an agent to be referenced by an incoming transaction. Since this calculation incorporates exit probabilities, which are the specific probabilities of the weighted random walk returning a given tip (i.e. resembling the Markov Chain Monte Carlo algorithm), our proposed measure called attachment probabilities tells us the relative "importance" of agents in the system [Zander 2018].

Experimental results We want to examine the impact of network latencies on attachment probabilities. The idea is to find a relation between the centrality of agents in the random agent network and the development of attachment probabilities per agent. We expect well connected agents to refer to each other more frequently, thus building heavier branches and becoming more "important", i.e. more likely to being attached to (higher attachment probability). Since the network latencies in our simulation framework are represented by the shortest paths between agents, which are held in a distance matrix, then a measure for agent centrality, namely closeness centrality, proves useful [Sabidussi 1966]. Closeness centrality is high (an agent is central in a network), if his average shortest path to all other agents is small.

In the initial agent network, the distance between two neighbouring agents is one by default. When ten agents are simulated, this causes the relative differences in network latencies between agents to be small (multiples of 1). To make differences in network latencies more pronounced, we scale the distance matrix by 10 (network latencies are multiples of 10). All agents issue the same amount of transactions (equal hashing power). We run 20 simulations for each parameter set and average the attachment probabilities. We also report results for different α-values. α is a parameter used in the IOTA consensus mechanism regulating the strength of the bias in the weighted random walk. In other words, a higher α-value means that heavier transactions (tips) are favoured when attaching new transactions to the DAG.

Table 1 suggests a clear trend, i.e. higher closeness centrality corresponds to larger average attachment probability. Furthermore, this effect becomes pronounced with higher α-values, where the most central agent builds even heavier transaction branches. However, we cannot show what happens in the long-run, and whether the values eventually stay stable at their respective level. Larger simulations should examine this matter. More results can be found in [Zander 2018].

Overall, this shows how latency in the asynchronous IOTA network can play together with the consensus mechanism for the advantage of well connected agents.

4. CONCLUSION

Summary This paper contributes an asynchronous, continuous time, and multi-agent simulator for DAG-based protocols. The framework does not yet model malicious agents and run-time complexity remains a concern. IOTA has been simulated. This has shown that tips issued by centrally located agents are more likely to be referenced by an incoming transaction, i.e. these agents have a higher probability of having their transactions accepted in the network over time, assuming honest and semi-honest strategies. Thus, in the real world IOTA implementation, adjustments of α might be needed for less well connected agents.

Future Work The next steps involve simulating additional DAG-

Agent	Closeness centrality	Average attachment prob.		
		$\alpha = 0.001$	$\alpha = 0.005$	$\alpha = 0.01$
1	0.6	0.127	0.127	0.145
2	0.53	0.123	0.129	0.13
3	0.47	0.109	0.107	0.116
4	0.43	0.099	0.094	0.101
5	0.39	0.092	0.088	0.094
6	0.39	0.095	0.097	0.084
7	0.36	0.09	0.094	0.081
8	0.36	0.098	0.093	0.098
9	0.36	0.089	0.092	0.081
10	0.27	0.078	0.078	0.07

Table 1: Transaction attachment probabilities depending on closeness centrality of agents in the P2P network with average results for 20 simulation rounds per α-level.

based protocols, refining the existing work and simulating larger networks. The initial focus is on blockDAG-protocols, such as SPECTRE [Sompolinsky et al. 2016] and PHANTOM [Sompolinsky and Zohar 2018]. An initial, albeit incomplete, implementation of SPECTRE is under development. SPECTRE separates the mining and consensus protocols into two distinct stages. The core of the consensus algorithm is made up of a pairwise vote over the order of the blocks, the outcome of which is then used to help determine which transactions in each block are accepted. A robust transaction algorithm then proceeds to specify a subset of accepted transactions that are guaranteed, up to a particular error probability, to remain accepted [Sompolinsky et al. 2016].

Acknowledgments

This research is funded by the Outlier Ventures grant. The authors thank Will Knottenbelt and Aron van Ammers for feedback and discussions on the simulator. We further thank Alon Gal for his help and support on the IOTA specifics.

5. REFERENCES

Leemon Baird. 2016. The swirlds hashgraph consensus algorithm: Fair, fast, byzantine fault tolerance. *Swirlds, Inc.Technical Report SWIRLDS-TR-2016* 1 (2016).

Tobias Bamert, Christian Decker, Lennart Elsen, Roger Wattenhofer, and Samuel Welten. 2013. Have a snack, pay with Bitcoins. In *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*. IEEE, 1–5.

Iddo Bentov, Pavel Hubcek, Tal Moran, and Asaf Nadler. 2017. Tortoise and Hares Consensus: the Meshcash Framework for Incentive-Compatible, Scalable Cryptocurrencies. *IACR Cryptology ePrint Archive* 2017 (2017), 300.

Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, and Emin Gn Sirer. 2016. On scaling decentralized blockchains. In *International Conference on Financial Cryptography and Data Security*. Springer, 106–125.

Christian Decker and Roger Wattenhofer. 2013. Information propagation in the bitcoin network. In *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*. IEEE, 1–10.

Paul Erdos and Alfrd Rnyi. 1960. On the evolution of random graphs. *Publ.Math.Inst.Hung.Acad.Sci* 5, 1 (1960), 17–60.

Alon Gal. 2018. Algorithm for calculating cumulative weights. (2018). `https://github.com/alongalky/iota-docs/blob/master/cumulative.md` Accessed: 2018-09-29.

Alon Gal and Clara Shikhelman. 2018. Partitioning in the Tangle: a Multi-Agent Extension. (2018). Unprinted, Accessed: 2018-09-29.

IOTA Foundation. 2018a. IOTA docs: Consensus. (2018). `https://docs.iota.org/introduction/tangle/consensus` Accessed: 2018-09-29.

IOTA Foundation. 2018b. IOTA docs: Tip selection. (2018). `https://docs.iota.org/introduction/tangle/tip-selection` Accessed: 2018-09-29.

Aggelos Kiayias and Giorgos Panagiotakos. 2016. On Trees, Chains and Fast Transactions in the Blockchain. *IACR Cryptology ePrint Archive* 2016 (2016), 545.

Yoad Lewenberg, Yonatan Sompolinsky, and Aviv Zohar. 2015. Inclusive block chain protocols. In *International Conference on Financial Cryptography and Data Security*. Springer, 528–547.

Satoshi Nakamoto. 2008. Bitcoin: A Peer-to-Peer Electronic Cash System. (2008). `https://bitcoin.org/bitcoin.pdf` Accessed: 2018-09-29.

Rafael Pass, Lior Seeman, and Abhi Shelat. 2017. Analysis of the blockchain protocol in asynchronous networks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 643–673.

Serguei Popov. 2016. The Tangle. (2016). `https://www.iota.org/research/academic-papers` Accessed: 2018-09-29.

Serguei Popov, Olivia Saa, and Paulo Finardi. 2017. Equilibria in the Tangle. *arXiv preprint arXiv:1712.05385* (2017).

Gert Sabidussi. 1966. The centrality index of a graph. *Psychometrika* 31, 4 (1966), 581–603.

Yonatan Sompolinsky, Yoad Lewenberg, and Aviv Zohar. 2016. SPECTRE: Serialization of proof-of-work events: confirming transactions via recursive elections. (2016). `https://eprint.iacr.org/2016/1159.pdf` Accessed: 2018-09-29.

Yonatan Sompolinsky and Aviv Zohar. 2013. Accelerating Bitcoins transaction processing: Fast Money Grows on Trees, Not Chains. (2013). `https://eprint.iacr.org/2013/881` Accessed: 2018-09-29.

Y. Sompolinsky and A. Zohar. 2018. PHANTOM, GHOSTDAG: Two Scalable BlockDAG protocols. (2018). `https://eprint.iacr.org/2018/104.pdf` Accessed: 2018-09-29.

Matei Zaharia, Reynold S Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J Franklin, et al. 2016. Apache spark: a unified engine for big data processing. *Commun. ACM* 59, 11 (2016), 56–65.

Manuel Zander. 2018. A Multi-Agent Simulation Framework and Analysis of the IOTA Tangle. (2018). Master's thesis. Imperial College London, UK.