# An Extensive Agent-based Simulation Study of Sycomore$^{++}$, a DAG-based Permissionless Ledger

Aimen Djari
University Paris-Saclay, CEA, List
Palaiseau, France

Emmanuelle Anceaume
CNRS / IRISA
France

Sara Tucci-Piergiovanni
University Paris-Saclay, CEA, List
Palaiseau, France

**ACM Reference Format:**
Aimen Djari, Emmanuelle Anceaume, and Sara Tucci-Piergiovanni. 2022. An Extensive Agent-based Simulation Study of Sycomore$^{++}$, a DAG-based Permissionless Ledger. In *Proceedings of ACM SAC Conference (SAC'22).* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3477314.3507245

## 1 INTRODUCTION

A recent evolution of the blockchain structure is emerging to address the performance issue of permissionless chain-based ledgers, in particular the small number of transactions confirmed per second. To address such an issue, new designs have been brought forward, including Bitcoin-NG, which favors an off-chain mechanism in which blocks refer to a leader in charge of validating transactions batched in micro-blocks out of the chain [6]; Lightning [10], which follows the same principle but only publishes the outcome of repeated transactions among a set of parties. Others propositions such as HashGraph [2], ByteBall [5], and Iota [4] leverage the presence of well known institutions to get rid of blocks, while Ghost [12] and Spectre [11] protocols family modifies the blockchain data structure from a totally ordered sequence of blocks to a directed graph of blocks. Blocks are built so that they commit the state of the directed graph at the time blocks were created which decreases the opportunity for powerful attackers to create blocks in advance. Regarding the graph-based approach, the absence of mechanisms to prevent the presence of conflicting records (i.e., blocks with conflicting transactions) or the presence of cycles in the directed graph (Spectre [11] organises blocks in a directed, but not acyclic, graph of blocks) require that participants execute a complex algorithm to extract from the graph the set of accepted (i.e., valid) transactions [11]. Sycomore[1] is an immutable permissionless distributed ledger whose structure is a particular directed acyclic graph of blocks, called SYC-DAG [1]. Its design differs from existing distributed ledgers in that its graph structure dynamically adapts to fluctuations in transaction submission rates: When the leaf block of a chain (more precisely the last blocks appended to a chain) of the graph exceeds a maximal loading threshold (the load is measured in Bytes), subsequent blocks are partitioned over two sibling chains, and these blocks are mined in parallel (as will be described shortly, even if blocks are appended in parallel to the SYC-DAG they cannot be conflicting, i.e., each valid transaction cannot appear in more than one block). Conversely, when the leaf blocks of two sibling chains (again the last blocks of two sibling chains) fall short of a minimal loading threshold, subsequent blocks will belong to a unique chain. The decision to split a leaf chain of the SYC-DAG or to merge two sibling ones is locally taken by each miner, and soundness of this decision is verifiable by everyone at any time [1]. Actually, Sycomore has been designed to meet the following properties [1]:

**P1. Self-adaptation to transaction load.** A rise or a drop in the current number of submitted transactions is dynamically handled by the progressive creation or disappearance of sibling leaf chains in the SYC-DAG;

**P2. Balanced partitioning of transactions.** There does not exist any transaction that belongs to two different blocks.

**P3. Unpredictability of the predecessor.** The leaf chain to which a new block is appended can neither be chosen nor predicted among all the leaf blocks of the SYC-DAG.

**P4. Chain fairness.** All the leaf chains of the SYC-DAG grow at the same speed.

**P5. Negligible probability of forks.** The probability of forks is maximal when the SYC-DAG is reduced to a single chain (i.e, $1,2 \times 10^{-3}$ in the time interval of 30 seconds) and decreases proportionally with the number of leaf blocks.

*Motivations and Contributions of this Work.* By dynamically adapting the width of the graph, that is the number of leaf blocks of the SYC-DAG, to the actual transaction load of the network, one might expect that Sycomore would guarantee an almost optimal transaction latency. By transaction latency it is meant the time that elapses between the instant at which a user submits a transaction to the network and the instant at which this transaction is confirmed in the SYC-DAG, i.e., belongs to a block that is deeply settled down into the SYC-DAG. This is actually true right after the periodic readjustment of the mining difficulty. In most of the proof-of-work based distributed ledgers, the mining difficulty is periodically calibrated to cope with variations of the network computational power. In Sycomore, such a readjustment is executed every time the height of the SYC-DAG has been increased by $H_{max}$ blocks since the previous readjustment period, where $H_{max} = 2016$ as in Bitcoin (i.e., every 14 days). Still, it is highly probable that during any period of 14 days, the transaction submission rate will drastically vary, modifying accordingly the width of the SYC-DAG. Hence, the mining difficulty which was computed to fit both the hash rate of the network and the state of the graph at the last readjustment may currently be either under-estimated or over-estimated w.r.t. the current state of the SYC-DAG. Although this remark applies to other PoW-based distributed ledgers, the consequences in Sycomore may be magnified compared to other designs due to its dynamic adaptation. Let us consider the scenario where, at the time readjustment took place, the SYC-DAG's width was very large, i.e., a large number of (valid) blocks could be appended in parallel to the ledger. Subsequently, the number of submitted transactions significantly dropped, leading to a progressive diminution of the number of leaf

[1] Sycomore is the french word for sycamore, a large broad-leaved tree tolerant to wind.

*SAC'22, April 25 –April 29, 2022, Brno, Czech Republic*
2022. ACM ISBN 978-1-4503-8713-2/22/04. . . $15.00
https://doi.org/10.1145/3477314.3507245

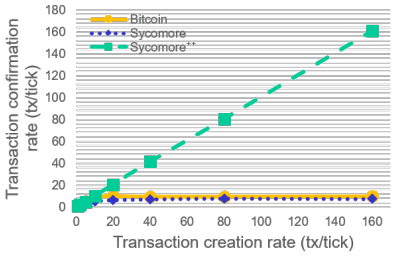
chains, and thus an under-estimated mining difficulty w.r.t. the current state of the SYC-DAG. Such a scenario shows a possible breach of security: Adversarial miners can take advantage of a too low mining difficulty to degrade the quality of the ledger [7], that is the maximal proportion of blocks contributed by the adversary in the ledger of any honest node. The opposite scenario may also happen, where a sudden augmentation of the transaction rate will give rise to an over-estimated mining difficulty, and thus a very high power consumption, which will be in the worst case similar to Bitcoin's one. As a consequence, transaction latency will be very high up to the next readjustment.

## 2 SYCOMORE$^{++}$

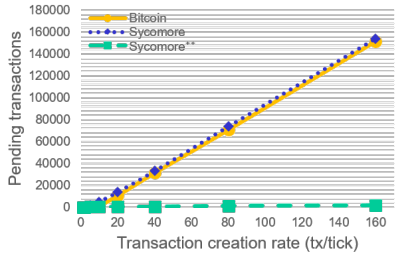
To prevent such critical issues, we propose an extension to Sycomore, called Sycomore$^{++}$, which aims at guaranteeing that whatever the structure of the SYC-DAG, a constant inter-block creation delay is maintained on any of its chains. The main idea of Sycomore$^{++}$ is to continuously adapt the block creation difficulty to the actual number of leaf chains of the SYC-DAG, that is to divide the block creation difficulty by the current number of leaf chains. Note that this adaptative adjustment does not replace the periodic global computational power re-adjustment. The former adapts the difficulty to the structure of the SYC-DAG while the latter adapts the difficulty to the global computational power of the system.

***Experimental validation.*** To validate the behavior of Sycomore$^{++}$ in presence of adversarial environments (i.e., sophisticated attacks, large communication delays, and sudden and substantial variations of the system workload demand), and to compare it with both Sycomore and Bitcoin, we have implemented these three distributed ledgers on an agent-based simulator dedicated to blockchain systems, called Multi-Agent eXperimenter (MAX) [9], which is based on the MaDKit framework [8]. MAX offers generic libraries to easily develop distributed ledger protocols and a large range of simulation scenarios. The simulator is a discrete event simulator, where the unit of simulation time is referred to as a tick. Message-passing libraries allow us to configure different types of communication schemes and message delays. In this work, the communication schema is configured as a reliable broadcast with configurable delay. All the experiments for Sycomore$^{++}$, Sycomore and Bitcoin have been run on Grid'5000, a large-scale and flexible test-bed for experiment-driven research [3].

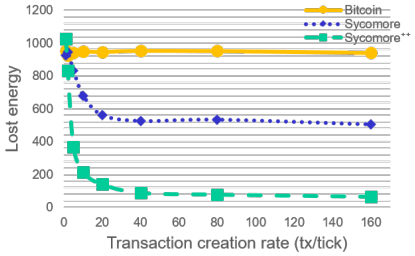
In this paper we only focus on both the scalability and reactivity properties. Figure 1 illustrates the main results regarding the capability of Sycomore$^{++}$, Sycomore and Bitcoin to handle high transaction submission rates. Specifically, we evaluate the transaction confirmation rate, the transaction latency, i.e., the average time elapsed between the submission of a transaction in the network and the time at which the transaction is confirmed, and the average number of pending transactions at the end of the simulation (i.e., waiting to be embedded in a block) as a function of the submission rate of transactions $f_{req}$. $f_{req}$ is equal to the number of transactions submitted per tick of simulation and is set at the beginning of each experiment. The energy lost by each protocol is also measured. The lost energy is the sum for each miner $u$ and for each created block $b$ not appended to the distributed ledger of the time spent working on $b$ times the computational power $cp_u$. Let us first focus on the confirmation rate of transactions as a function of their creation rate (see Figure 1a). The main observation regarding Bitcoin and Sycomore is that whatever the computational power of the network, no more than 10 txs/tick are confirmed, which illustrates the impact of the globally constant inter-block delay (i.e. a block is mined every 10 ticks in average). Sycomore shows slightly worse results than Bitcoin, which is due to the augmentation of the number of chains, in which blocks can be moderately loaded. On the other hand, by continuously adapting the mining difficulty to the number of leaf blocks, and thus to $f_{\text{req}}$, Sycomore$^{++}$ exhibits an optimal behavior regarding confirmed transactions, i.e., $\forall f_{\text{req}}$, the transaction confirmation rate equals the transaction creation rate. For the sake of comparison, confirming 160 txs/tick with the simulator (actually we cannot stress more the simulator) means confirming $6,400$ txs/mn in the real life. Figure 1b shows the average number of pending transactions at miners, that is the average number of transactions that accumulate at miners before being embedded in blocks. It clearly shows that for both Bitcoin and Sycomore, this number linearly increases with the transaction submission rate once it exceeds 10txs/tick since this corresponds to the global inter-block creation delay. In contrast, by adapting the number of created blocks to $f_{\text{req}}$, Sycomore$^{++}$ drastically reduces the average number of pending transactions. For example, for $f_{\text{req}} = 10$ txs/tick, this number is equal to 432 transactions, and for $f_{\text{req}} = 160$ txs/tick, it is equal to $1,957$ transactions, compared to $154,000$ ones in both Bitcoin and Sycomore. Figure 1c illustrates the lost energy as a function of $f_{\text{req}}$. In Bitcoin, since the inter-block delay, the number of miners, and the difficulty do not vary during each experiment, the same amount of energy is lost regardless of $f_{\text{req}}$. While this setting also applies to Sycomore, the fact that the SYC-DAG becomes larger with increasing values of $f_{\text{req}}$ gives rise to a uniform distribution of the global computational power over the leaf chains, and thus decreases miners' competition. Thus less work is wasted w.r.t Bitcoin. Regarding Sycomore$^{++}$, for increasing values of $f_{\text{req}}$, the SYC-DAG becomes larger and blocks are created faster (as the mining difficulty adapts to the SYC-DAG structure), which allows Sycomore$^{++}$ to reach an optimal number of leaf chains more quickly than Sycomore does. As a consequence, we get a better parallelism of miners' work, and thus a drastic reduction of energy loss. Figure 1d illustrates the average transaction latency as a function of $f_{\text{req}}$. Recall that the transaction latency measures the time elapsed between the instant at which a transaction is submitted to the network by the user and the time it becomes confirmed in the ledger. In contrast to all the other experiments, transaction latency has been measured as follows: transactions are submitted at $f_{\text{req}}$ for a while, then $f_{\text{req}}$ is set to 0, and simulations stop once all the submitted transactions have been confirmed. The first observation is that, in Bitcoin, once $f_{\text{req}} \geq 10$ txs/tick, transaction latency linearly increases with $f_{\text{req}}$, which clearly corroborates both Figures 1a and 1b. Regarding Sycomore, the loss of performance w.r.t Bitcoin is due to the fact that blocks in all the sibling chains are not necessarily fully loaded, which delays accordingly transaction latency. On the other hand, Sycomore$^{++}$ enjoys an average constant latency (equal to 50 ticks regardless of $f_{\text{req}}$). Essentially, the more transactions are submitted, the more blocks are filled until the optimal shape of the graph is reached (which can take time on Sycomore), and therefore the first



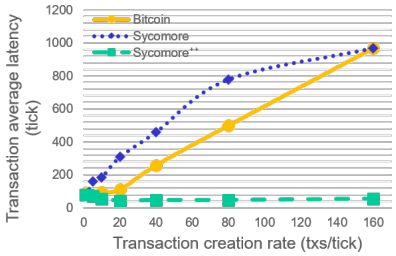

**(a) Transaction confirmation rate as a function of the transaction creation rate.**



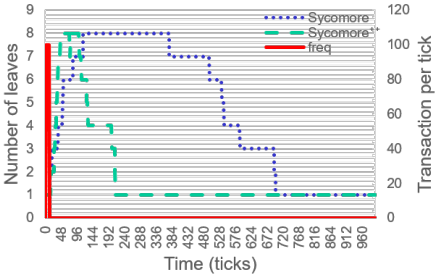

**(b) Number of pending transactions as a function of the transaction creation rate.**




**(c) Lost energy as a function of the transaction creation rate.**




**(d) Transaction average latency as a function of the transaction creation rate.**




**(e) Reactivity of both Sycomore and Sycomore$^{++}$ in presence of a peak of load.**

**Figure 1: Scalability of Bitcoin, Sycomore and Sycomore$^{++}$ and Reactivity of both Sycomore and Sycomore$^{++}$.**

blocks are often loaded at 100%. This phenomenon is noticeable for $f_{\text{req}} = 160$ txs/tick.

Finally, Figure 1e illustrates reactivity of both Sycomore and Sycomore$^{++}$ SYC-DAG, i.e., their capacity to react to sudden and abrupt fluctuations in the creation transaction rate. We omit Bitcoin from this evaluation since Bitcoin chain does not adapt to transaction demand. The sudden and abrupt fluctuation in the transaction rate is simulated by the presence of a load peak (illustrated by the red constant function from $t = 1$ to $t = 11$ ticks at $f_{\text{req}} = 100$txs/tick). Both Sycomore and Sycomore$^{++}$ initially undergo a series of splits, and then progressively move on to a series of merge up to converging to a single chain of blocks. Sycomore$^{++}$ differs from Sycomore in its rapidity to split and merge: Sycomore$^{++}$ succeeds in coping with the load pick 75% faster than Sycomore does, and 33% faster than Sycomore to cope with the sudden shrink of load.

## 3 CONCLUSIONS

In this paper we have presentedSycomore$^{++}$, a graph based distributed ledger that clearly takes advantage of the computational power provided by the system, and succeeds in always guaranteeing a transaction confirmation rate that adapts to their creation rate. As a consequence, the number of pending transactions is almost null in average, and transaction latency is almost constant and optimal. In addition, its SYC-DAG structure and its ability to adapt extremely quickly to the transaction creation rate makes it a less energy-intensive solution than its competitors. As future work, we intend to analyse the computational cost of adversarial strategies in these distributed ledgers in presence of transient network partitions.

## REFERENCES


[1] E. Anceaume, A. Guellier, R. Ludinard, and B. Sericola. 2018. Sycomore: A Permissionless Distributed Ledger that Self-Adapts to Transactions Demand. In *Proceedings of the IEEE 17th International Symposium on Network Computing and Applications (NCA)*.

[2] L. Baird. 2016. *The Swirlds hashgraph consensus algorithm: Fair, fast, Byzantine fault tolerance.* Technical Report. http://www.swirlds.com/downloads/SWIRLDS-TR-2016-01.pdf

[3] D. et al. Balouek. 2013. Adding Virtualization Capabilities to the Grid'5000 Testbed. In *Cloud Computing and Services Science*, Ivan I. Ivanov, Marten van Sinderen, Frank Leymann, and Tony Shan (Eds.). Communications in Computer and Information Science, Vol. 367. Springer International Publishing, 3–20.

[4] G. Bu, Ö. Gürcan, and M. Potop-Butucaru. 2019. G-IOTA: Fair and confidence aware tangle. *CoRR* abs/1902.09472 (2019). arXiv:1902.09472 http://arxiv.org/abs/1902.09472

[5] A. Churyumov. 2017. ByteBall : A Decentralized System for Storage and Transfer of Value. https://byteball.org/Byteball.pdf

[6] I. Eyal, A. E. Gencer, E. Gün Sirer, and R. Van Renesse. 2016. Bitcoin-NG: A scalable blockchain protocol. In *Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*.

[7] J. A. Garay, A. Kiayias, and N. Leonardos. 2015. The Bitcoin Backbone Protocol: Analysis and Applications. In *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques - Advances in Cryptology (EUROCRYPT)*.

[8] O. Gutknecht and J. Ferber. 2000. The MadKit Agent Platform Architecture. In *Proceedings of the International Workshop on Infrastructure for Multi-Agent Systems: Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems*.

[9] Nicolas Lagaillardie, Mohamed Aimen Djari, and Önder Gürcan. 2019. A Computational Study on Fairness of the Tendermint Blockchain Protocol. *Information* 10, 12 (2019). https://doi.org/10.3390/info10120378

[10] J. Poon and T. Dryja. 2016. *The bitcoin lightning network.* https://lightning.network/lightning-network-paper.pdf

[11] Y. Sompolinsky, Y. Lewenberg, and A. Zohar. 2016. SPECTRE: A Fast and Scalable Cryptocurrency Protocol. *IACR Cryptol. ePrint Arch.* (2016), 1159.

[12] Y. Sompolinsky and A. Zohar. 2013. Accelerating Bitcoin's Transaction Processing. Fast Money Grows on Trees, Not Chains. *IACR Cryptology ePrint Archive* 2013 (2013).