Adapting the number of validator according the need in a BFT chain with SABINE

Guilain Leduc, Sylvain Kubler, Jean-Philippe Georges

▶ **To cite this version:**

Guilain Leduc, Sylvain Kubler, Jean-Philippe Georges. Adapting the number of validator according the need in a BFT chain with SABINE. 6th IFAC Symposium on Telematics Applications, TA 2022, Jun 2022, Nancy, France. 10.1016/j.ifacol.2022.08.011 . hal-03777351

HAL Id: hal-03777351
https://hal.science/hal-03777351

Submitted on 14 Sep 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Adapting the number of validator according the need in a BFT chain with SABINE

Guilain Leduc * **Sylvain Kubler** * **Jean-Philippe Georges** *

* *Université de Lorraine, CNRS, CRAN, F-54000 Nancy, France (e-mail: guilain.leduc@univ-lorraine.fr)*

Abstract:
Blockchain is a revolutionary technology asserting the integrity and the share of data between untrusted users and without central authority.In case of private and non-publicly accessible networks, more energy efficient consensus can be adapted than permissionless consensus such as consensus used in cryptocurrency. Theses permissionless consensuses are called BFT consensus and are based on communication between participants of the network. By design, latency and throughput of transaction commitment are impacted by the number of participants because an agreement with more participants takes more time. On the contrary, a chain shared with more participants is more secure to a same number of malicious users. Theses two properties implies that the number of validator is a trade off between security and performances that can potentially evolve according to demands. The present research proposes Sabine (Self-Adaptive BlockchaIn coNsEnsus protocol) a new consensus based on PBFT to optimally and dynamically adapt the pool of validator in order that the output transaction throughput meets the input transaction throughput, with a minimum limit on the number of validators, implying a more reactive chain, with less latency. An evaluation of Sabine on the Grid5000 shows a relative error between this throughput of 4.45% compared to 67.6% for a classic chain.

Keywords: Blockchain, BFT, Security, Consensus, Control Theory.

1. INTRODUCTION

Blockchain are distributed ledger where data are packed in blocks shared and linked by cryptography to assert their integrity. The concept of blockchain was democratized with Bitcoin (Nakamoto, 2008) to other fields such as energy, finance or supply chain (Zheng et al., 2018) because they allow untrusted peers to build an immutable ledger in a decentralized way. The key piece to ensure the consistency of the stored data is the consensus, the protocol to coordinated peers to commit same exact blocks in the same order. Two main families of consensus emerges through the literature: *BFT protocols* (Lamport, 2019) and *Proof-based protocols* (e.g., Proof of Work, Proof-of-Stake, Proof of Importance, etc.). The former are based on an agreement, while the latter are based on the periodic appearance of a proof among users whose cost of forming this proof is so high that it ensures the trust of the one offering it. The mechanics beyond these two classes of consensus result in different properties and performance characteristics which are mirrors of each other whether in terms of permission, scalability, throughput or consistency.

At the opposite of Proof-based consensus, BFT protocols are fast and energy efficient because they don't waste time and energy resources to produce a proof to compete the right of committing a new block. The competition around a proof is replaced by an agreement between at least ⅔ validators, ensured by a set of elections. The number of messages needed to reach this agreement is costly and quadratic in most of the implementation like PBFT (Castro and Liskov, 1999). The main consequence is that validation load decreases with the increase of the number of validator, and so, decrease the throughput of block commitment. But increasing the number of validator have the positive aspect of increasing the security of the chain because BFT chain tolerate until a third of malicious validator, which can compromise the chain by avoiding the commit of new blocks or supported the commit of suspicious blocks. The choice of the number of validators when configuring a chain is thus done manually by solving a trade off between two conflicting properties: the maximum throughput that the chain can reach and the number of malicious validator tolerated.

Due to this trade-off, BFT consensuses are not scalable and limited to a hundred nodes. To overcome this limit, the strategy consists of defining two types of nodes: validator which participates in the consensus, and non-validators, which gossips the agreement of the first type. The main condition of this strategy is that all nodes must agree on the same set of validators. So the simplest strategy is called *Consortium Blockchain* (Belotti et al., 2019) and consists to keep exactly the same set of validators, and selecting preferably central authorities like banks as trusted validator.

Another more decentralized strategy consists to periodically select a different group of validator among all nodes, depending on their thrust. This selection is the generalization of the proof based issue and bring to a new blockchain consensus called *Hybrid BFT-based Algorithms* (Belotti et al., 2019). It consists of the succession of two consensus. A first Proof-based consensus to select a temporary set of

trusted nodes among all nodes of the networks , followed by a BFT consensus to validate transactions and commit blocks in the chain while ensuring strong consistency, liveness and safety among validators.

Various protocols are used in this family. The PoW (*Proof-of-Work*) protocol, developed for Bitcoin (Nakamoto, 2008) where nodes solve a cryptographic puzzle to commit a block, is used in Byzcoin (Kogias et al., 2016) and ASHWAChain (Arora et al., 2020), but also most energy efficient protocol like PoS (*Proof-of-Stack*) or VRF (*Verifiable Random Function*). In PoS, every node that wants to take part in the consensus has to make a security deposit using the chain's cryptocurrency. Tendermint (Buchman et al., 2018) and LinSBFT (Qi et al., 2020) use the PoS in the context of Hybrid BFT-based Algorithm. In order to avoid the situation where richest member take the control of the chain, randomness is introduced with VRF (Goldberg et al., 2018), which provide verifiable pseudo-random mechanism to select validators. Algorand (Gilad et al., 2017) and Albatross (Berrang et al., 2019) are two examples of chains that combines VRF and PoS to select their validators. Gosig (Li et al., 2020) only use VRF to select its validators.

Although Consortium and Hybrid BFT-based algorithms allow more nodes in the chain while keeping constant performances, performances are still limited by the number of validators. And as the performance requirements may change according to the needs, the maximum throughput, and consequently the security, cannot be adapted on the fly to the needs because of a constant number of validators, which could allow an increase in the number of malicious validators tolerated in case of a low demand period. Having a fixed number of validators is thus a fixed response to a dynamic trade off between security and performances. To overcome this limitation, the present paper introduces a new BFT-based algorithm called Sabine (standing for "Self-Adaptive BlockchaIn coNsEnsus"), which self-adapt the number of validators in order to continuously meet (i.e., at any given point in time) the requested/input transaction throughput under security constraints.

Section 2 presents the problem that should be solved in order to overcome this limitation and presents Sabine, our proposition which solves that problem. Sabine is then evaluated and validated through real-life experiments in section 3; conclusions follow.

2. SABINE PROPOSAL

All the blockchain solutions previously discussed select a number of validators among a set of nodes, but this number is set constant (around 100 validators), which implies constant performance (in terms of maximum throughput, validation latency, security ...) of the chain whatever the needs, assuming nodes are homogeneous and network constant. In this paper, we propose SABINE (*Self Adaptive BlockchaIn coNsEnsus*), an algorithm that dynamically controls the number of validator in order to adapt the performance of a BFT based chain to maximize the security. The problem Sabine is trying to answer is defined in the subsection 2.1. Next, the three various steps of the executions of Sabine are described in the following subsections.

As a mathematical model which can be applied in every specific network is too complex to define, Sabine based its adaptation on a model build with Machine Learning to link the Capacity, the network delay and the number of validators for chain in its network. The training phase and the parameters of the Machine Learning engine are described in the subsection 2.2. Based on this model, Sabine estimate the ideal number of validator with a calculus detailed in subsection 2.3 and the result of this calculus is applied with mechanisms described in subsection 2.4.

2.1 Problem Statement

As was already mentioned in the introduction, the higher the number of validators, the more secure the chain. However, this has a direct consequence in the chain (output) throughput, which (quadratically) decreases, thus leading to a trade off problem between security and throughput. This trade-off problem can be expressed as a constraint to be solved, as formalized in (1).

$$\underset{\mathcal{N}}{\operatorname{argmax}} \left(n^t \mid \tau_{commit}(n^t, \delta^t) = \tau^t_{req}\right) \ s.t.\ n > N_{min} \quad (1)$$

For this purpose, Sabine adjust the number of validators of the chain n^t in real time t, which is maximum is the number of nodes $\mathcal{N}$, in order that the transaction committed throughput τ_{commit} is closed to the transaction requested throughput τ_{req}. The transaction committed throughput τ_{commit} is also affected by a delay δ, which represents time variation due to bandwidth or hardware performance variation. Sabine estimate this delay in order to improve the estimation of the ideal number of validator. A security limit N_{min} is also set by administrators in order to assert a minimal tolerance to malicious nodes.

2.2 Training Phase

Sabine aims at adapting the chain configuration to maximize the throughput according to a model estimated during a training phase. Thus, the first step of Sabine is to collect samples of the network's maximum transaction throughput for a same transaction to build the model. This particular value of throughput is called *Capacity* and is straight linked to the time needed to commit a block. To measure this capacity, the chain is deployed on the network and the committed throughput in response to a fixed requested throughput of transaction is measured. This requested throughput is determined in a primary analysis to be closest as possible but also greater than the expected committed throughput. Two parameters are varying: the number of validators of the blockchain and the (network) delay. Assuming that there is no other traffic in the network, the delay represents time variation due to bandwidth or hardware performance variation. It is simulated by nodes by a wait before sending a message to other nodes of the chain. This delay can be considered as the root cause of the system drift from the initial estimated model. Estimating such drift is hence necessary to adapt the system in case of perturbations. As a consequence, every sample consists of a triplet $(capa, \delta, n)$ and if a set of triplets is large enough, it is used as a training dataset to build a model M which associates the number of validators n, the delay δ and the capacity of the chain using machine learning based on a polynomial regression detailed in 2.

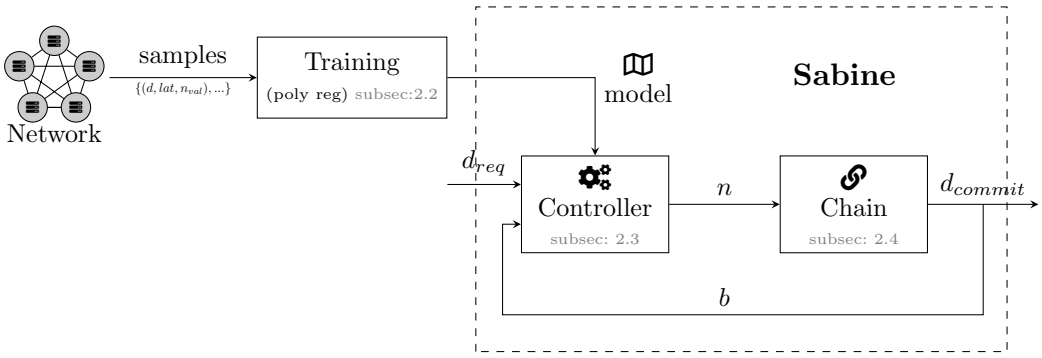


Fig. 1. Diagram of the Sabine's steps

Fig. 2. Model parameters with Sklearn

```
def invert(x, deca=70):
    return 1 / (deca + x)


polynomial_regression = make_pipeline(
    FunctionTransformer(invert),
    PolynomialFeatures(degree=34),
    StandardScaler(),
    RidgeCV(
        alphas=[0.001, 0.01, 0.1, 1, 10,
                100, 1000]
    )
)
```

The figure 3 represents the model obtained by Machine Learning on the experimentation bench described in section 3 (with a mean absolute error of 0.961). In this model, a first empirical function is applied on data, followed by a polynomial transformation and a ridge regression. Parameters are obtained with hyperparameter tuning. The given model M provides a mostly decreasing throughput $M(n, \delta)$ according to the number of nodes n and the delay δ, and a post filter treatment is applied to assert the decrease. In this post filter, for a couple (n, δ), if a higher Capacity exists for a greater n or δ, then the Capacity at (n, δ) takes this value.

$$\frac{\partial M}{\partial \delta}(n, \delta) \leq 0 \tag{2}$$

$$\frac{\partial M}{\partial n}(n, \delta) \leq 0 \tag{3}$$

2.3 Control phase

Based on this model, the Sabine algorithm solves the constraint 1. The idea is to maximize security by increasing the number of validators as long as the throughput is adequate, i.e., the committing transaction throughput is equal to the requested transaction throughput, under the condition of a minimum security level, equivalent to a minimum number of validators. Sabine solves this constraint by measuring the committed transaction throughput and the requested transaction throughput, and comparing these variables with the previous model.

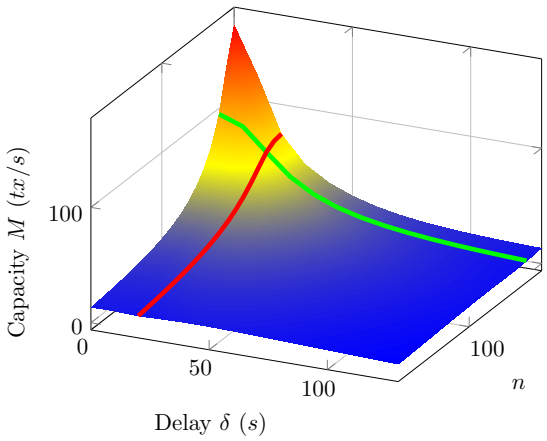


Fig. 3. Estimated Model of the link between Capacity, number of nodes and delay

In a realistic environment, a delay may emerge due to network latency or hardware performance variation and reduce chain performance. Due to the properties 2 and 3, the inverse functions ν and ϕ can be defined. ν, defined in 5, returns the maximal latency such that the model M, restricted to a given number of validator n, provides a given capacity τ. ϕ is the equivalent for the number of validator, is defined in 3, and returns the maximal number of validator n such that the model M, restricted to the latency δ, provides a given capacity τ.

$$\nu(\tau, n) = max(\{\delta | M(n, \delta) = \tau\}) \tag{4}$$

$$\phi(\tau, \delta) = max(\{n | M(n, \delta) = \tau\}) \tag{5}$$

In order to estimate an ideal number of validators in this environment, Sabine estimates the delay based on the simulated delay of the model introduce in the last subsequence. In a first step, the delay is assumed responsible for the loss of performance of the chain. Knowing the actual number of validators n^{t-1}, the delay δ_{est} is estimated by taking the simulating delay that meets with the actual capacity of the chain (green line in fig 3). The committed throughput is not a good metric to estimate the actual capacity because, depending on the requested throughput, all block may not be fulfilled with transactions. However, the block commitment throughput is more relevant, assuming that the requested transaction throughput is important enough to produce continuous blocks, which occurs if the requested throughput is greater than the capacity divided by the block size. So, the capacity is esti-

mated by multiplying the block commitment throughput b with the maximum number of transactions in a block $BlockSize$.

$$\delta^t_{est} = \nu(b^t \cdot BlockSize, n^{t-1}) \quad (6)$$

Then, the model is restricted to the estimated delay δ_{est}, forming a decreasing bijection between the maximum throughput and the number of validators (red line in fig 3). The ideal number of validators is obtained by comparing the requested transaction throughput τ_{req} to this bijection. In case of incertitude, the lower number of validators is chosen. With this configuration, the chain is able to commit with throughput equal to the requested throughput, answering the constraint 1, under the condition the estimated number of validators is greater than a security limit detailed by administrators.

$$n^t = \phi(\tau^t_{req}, \delta^t_{est}) \quad (7)$$

2.4 Action on the Chain

An instance of Sabine is running on every node of the network. Theses instances are synchronized through the chain with a special transaction that controls the number of validators. Once a time-out ended after the last special transaction, the next Proposer of the chain applies the Sabine algorithm to determine the ideal number of validators. In this way, Sabine is applied periodically.

Once this number is determined, a special transaction is inserted and process in priority to be committed. As every validator runs Sabine, they vote in favour of the commit of the special transaction if the decision agree with their own estimation. This prevents the Proposer to be malicious. In this case, validators adopt the recuperation mechanism if the Proposer is suspected to be malicious, the *RoundChange* step in our IBFT example in figure 4. If the special transaction is committed, it then transmits securely to all nodes by the gossip mechanism and the update of the number of validators is applied after the transaction commitment.

Sabine only estimates the ideal number of validator. The selection of which validator to add or remove is given to the chain policy. A simple mechanism has been implemented in the tested chain: nodes have ordered identifiers and became validator or non-validator according to their identifier. The gossip of the new number of validator is realized with a special transaction, ensuring the consistency of the modification for the next blocks. Most advanced mechanism could be adapted, liked those detailed in the introduction.

3. EXPERIMENTATION

Sabine have been implemented and adapted to a BFT chain [1] running on the Grid5000 (Balouek et al., 2013), a large scale and flexible testbed for distributed computing. The chain and the platform is described in subsection 3.1. Results have been extracted from this experimentation and are detailed in subsection 3.2.

[1] The code is available on *https://github.com/inpprenable/Sabine*

3.1 Chain Description

Sabine is tested in a chain which used a variation of the IBFT consensus. The IBFT (*Istanbul Byzantine Fault Tolerance consensus*) is a Byzantine fault tolerant consensus algorithm tolerating f faulty processes out of n, where $n \geq 3f + 1$ (Moniz, 2020). Voting mechanism for the commit of block is closed to the original PBFT consensus of Castro-Liskov (Castro and Liskov, 1999), and two states were added for the non-validator cases. The state diagram of the tested chain is represented in figure 4. The step *RoundChange* represent cases where the Proposer failed or is suspected to be malicious, but it is not implemented because the study doesn't focus on malicious or crashed nodes.

Unlike the original PBFT protocol, the leader is selected randomly among the validator with a VRF based on the previous block: $id^n_{proposer} = sha256(block^{n-1}) \bmod n^n$. As the goal of our study is to measure the impact of a BFT protocol, the block policy is set to limit blocks to 5 transactions, an arbitrary size that is low enough to favour the appearance of blocks.

The topology between nodes is ensured strongly connected thanks with a dedicated bootstrap protocol. No gossip protocols are implemented to reduce the number of messages. This ensures an estimation of the number of messages equals to the initial PBFT model. All messages are transmitted with the TCP protocol. As only overall performance is studied, nodes are not assumed to fail, so no recovery phases are attempted.

Nodes are deployed with Docker Swarm on Grid5000 on a cluster of 4 physical machines composed of 2 Intel Xeon E5-2630. Around 50 nodes are deployed on dockers on each machine for a total of 200 nodes with dedicated resources (0.3 CPUs and up to 1 Gb of Ram per node) to consider independent nodes on a single machine. Machines are directly connected to a switch in order to minimize network latency. Transactions are generated and gossiped by an external machine without resource limits to keep stable the transaction request throughput. They consist of the same string with the identifier of the transaction. Transactions, and therefore blocks, are all the same size.

3.2 Results

In order to evaluate Sabine, the (throughput) demand represented in figure 5a have been submitted to multiple chains assoicated or not with Sabine. This demand is composed of requested throughput attainable by chain with the adequate number of nodes.

Without Control Finding a perfect static number of validator is a hard exercise when initiating a BFT chain. Experiments represented in figure 5b show various chain configurations with different numbers of validators. Taking too few validators results in a chain that can perform high throughput with low relative error between the input requested transaction throughput and the output committed transaction throughput (1.29% estimated over a $60s$ window with 4 validators on our experiment) but, due to BFT properties, the chain does not tolerate many validators (only 1 in our example). On the contrary, taking all nodes

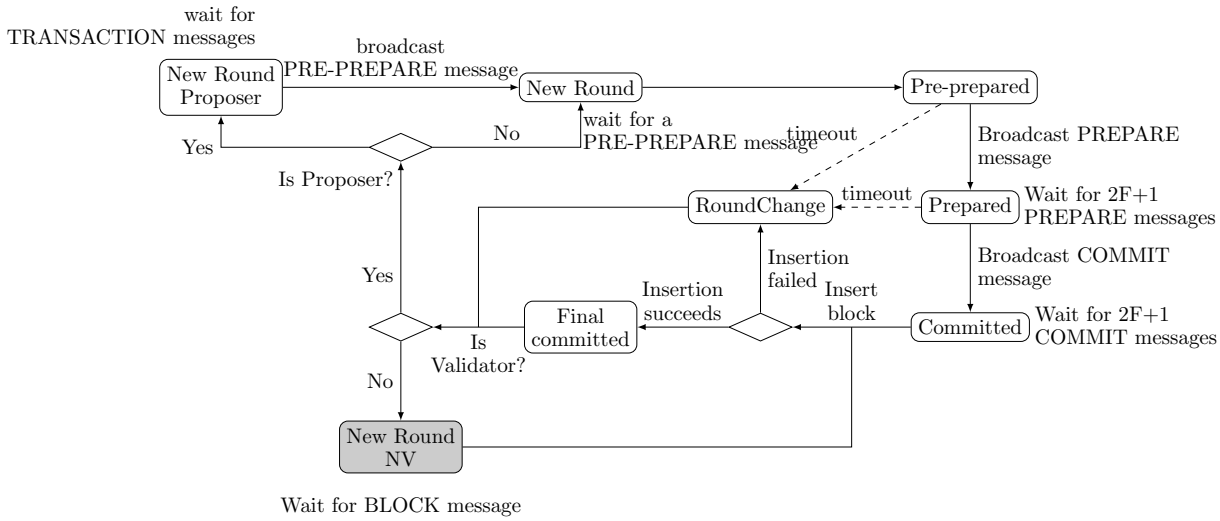


Fig. 4. Representation of scaling mechanism of BFT based blockchains

as validators tolerates a greater number of malicious nodes but throughput performances fall, which implies latency to satisfy request. In our second example with 200 validators, the chain can tolerate 66 but the relative error is equal to 67.6% before the end of demand (at $t = 3\,000\ s$). Network knowledge allows the number of validators to be set offline to satisfy the average demand, but it requires the knowledge of the future demand and does not adapt to requested throughput variation, implying a large error between input and output. In our example, the demand 5a can be achieved with 61 validators but the relative error of a chain with this number of validators remains high since it is equal to 25.6%.

With Control The same request was applied on a chain associated with Sabine in order to dynamically adjust the number of validator.The results are presented in figure 5c. every $60s$, Sabine is applied and determines the ideal number of validator, based on the time since the last occurrence of Sabine. The results show that the output of the chain is closed to the input, with an average difference of 2.04 transactions, corresponding to an absolute relative error of 4.45% with an average number of validator of 64.3. Sabine is also able to adapt the chain to delay variation. A chain has been submitted to a fixed requested transaction throughput and a varying network delay. Results represented in figure 6a show that at each occurrence of Sabine, slowdown due to delay are compensated by decrease in the number of validators. The output throughput is closed to the input throughput with a relative error of 3.27% in average.

With Control and both delay and input variations Previous experiment shows that Sabine adapts the number of validators according to input throughout and network delay independently. To evaluate the Sabine control in a more realistic case where both input throughput and network delay changes, 21 chains associated with Sabine were deployed and submitted to the reference demand 5a with an independent random delay. This network delay is around $15s$ and varies every 150 s in average. Results show the effectiveness of Sabine because the relative error between the requested and the committed throughput remains equal in average to 7.21% with a std of 0.71. This compensation of the delay was managed by a reduction of the number of validators to 70.9 in average (with a standard deviation of 6.52 between experiments).

4. CONCLUSIONS

In this paper, we present Sabine, a control on the number of nodes in a BFT chain to dynamically solve the trade off between security and throughput performance, depending of the input transaction throughput. We showed that a chain associated with Sabine can increase its security during periods of low request and further research should be done to limit the decrease of the number of validator under a temporary increase in throughput attack. The Sabine control also takes into account the network and hardware performance variation by introducing a delay in the estimation of the solution. Sabine is tested and validated on a chain based on IBFT but could be extended to every BFT consensus or to dynamically determine the number of validator in Hybrid BFT-based Consensus.

ACKNOWLEDGEMENT

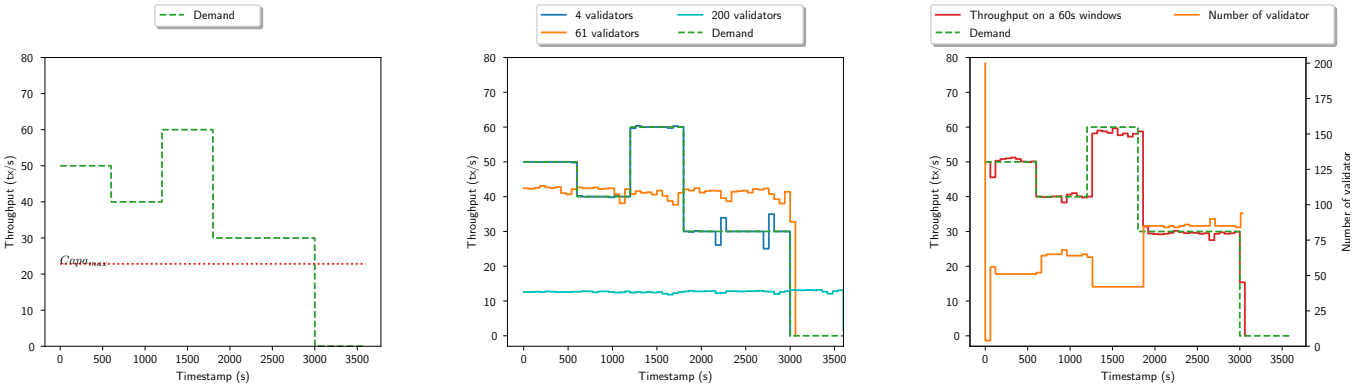
The authors acknowledge the support of the Agence National de la Recherche (ANR), under grant ANR-20-CE25-001 (project TIC-TAC-SDN).

Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).

REFERENCES

Arora, S., Jain, A., and Gujar, S. (2020). Ashwachain: A fast, scalable and strategy-proof committee-based blockchain protocol. In *Workshop on Game Theory in Blockchain at WINE*, volume 2020, 9.

Balouek, D., Carpen Amarie, A., Charrier, G., Desprez, F., Jeannot, E., Jeanvoine, E., Lèbre, A., Margery, D., Niclausse, N., Nussbaum, L., Richard, O., Pérez, C.,

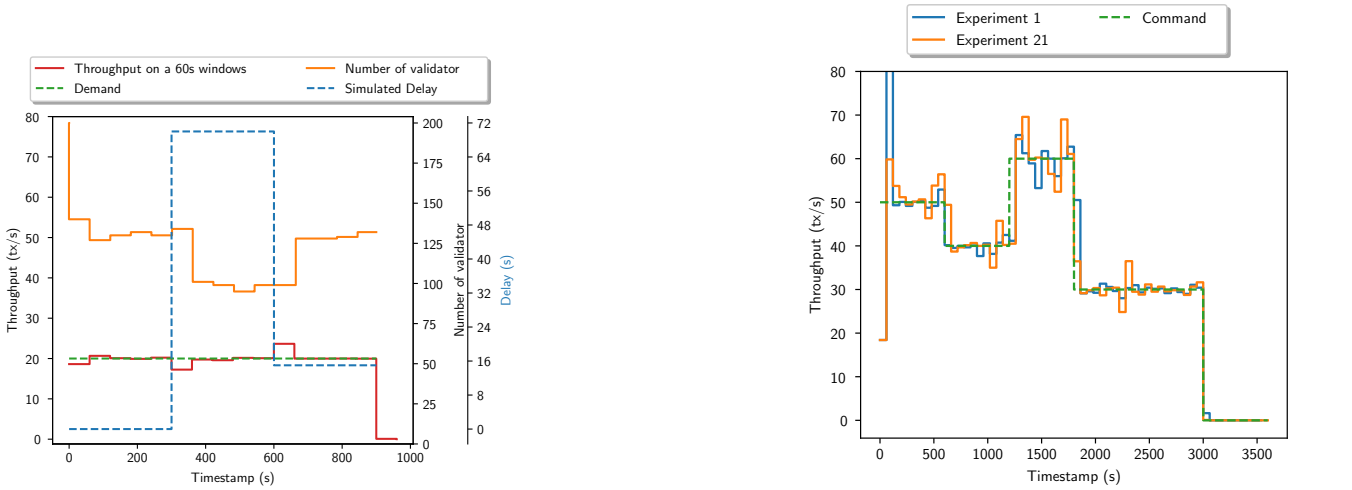


(a) Demand used to benchmark chains

(b) Committed throughput of chains with a fixed number of validators submitted to the demand 5a

(c) Committed throughput and number of validators for the demand 5a for a chain associated with Sabine

Fig. 5. Comparison of committed throughput for a common demand for chains with and without Sabine



(a) Chain with Sabine response to a delay variation with a fixed requested throughput

(b) 2 chain with Sabine, with same model and demand, but with random and independent delay

Fig. 6. Sabine's responses to delay variations

Quesnel, F., Rohr, C., and Sarzyniec, L. (2013). Adding virtualization capabilities to the Grid'5000 testbed. In I.I. Ivanov, M. van Sinderen, F. Leymann, and T. Shan (eds.), *Cloud Computing and Services Science*, volume 367 of *Communications in Computer and Information Science*, 3–20. Springer International Publishing. doi: 10.1007/978-3-319-04519-1_1.

Belotti, M., Božić, N., Pujolle, G., and Secci, S. (2019). A vademecum on blockchain technologies: When, which, and how. *IEEE Communications Surveys & Tutorials*, 21(4), 3796–3838. doi:10.1109/COMST.2019.2928178.

Berrang, P., von Styp-Rekowsky, P., Wissfeld, M., França, B., and Trinkler, R. (2019). Albatross – an optimistic consensus algorithm. In *2019 Crypto Valley Conference on Blockchain Technology (CVCBT)*, 39–42. doi: 10.1109/CVCBT.2019.000-1.

Buchman, E., Kwon, J., and Milosevic, Z. (2018). The latest gossip on bft consensus. doi: 10.48550/ARXIV.1807.04938.

Castro, M. and Liskov, B. (1999). Practical byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, OSDI '99, 173–186. USENIX Association, USA.

Gilad, Y., Hemo, R., Micali, S., Vlachos, G., and Zeldovich, N. (2017). Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, 51–68. Association for Computing Machinery, New York, NY, USA. doi:10.1145/3132747.3132757.

Goldberg, S., Vcelak, J., Papadopoulos, D., and Reyzin, L. (2018). Verifiable Random Functions (VRFs). 24.

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

Lamport, L. (2019). *Time, Clocks, and the Ordering of Events in a Distributed System*, 179–196. Association for Computing Machinery, New York, NY, USA. doi: 10.1145/3335772.3335934.

Li, P., Wang, G., Chen, X., Long, F., and Xu, W. (2020). Gosig: A scalable and high-performance byzantine consensus for consortium blockchains. In *Proceedings of the 11th ACM Symposium on Cloud Computing*, SoCC '20, 223–237. Association for Computing Machinery, New York, NY, USA. doi:10.1145/3419111.3421272.

Moniz, H. (2020). The istanbul bft consensus algorithm. doi:10.48550/ARXIV.2002.03613.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*.

Qi, X., Yang, Y., Zhang, Z., Jin, C., and Zhou, A. (2020). Linsbft: Linear-communication one-step bft protocol for public blockchains. doi:10.48550/ARXIV.2007.07642.

Zheng, Z., Xie, S., Dai, H.N., Chen, X., and Wang, H. (2018). Blockchain challenges and opportunities: a survey. *International Journal of Web and Grid Services*, 14(4), 352–375. doi:10.1504/IJWGS.2018.095647.