Sabine: Self-Adaptive BlockchaIn coNsEnsus

Guilain Leduc, Sylvain Kubler, Jean-Philippe Georges

Sabine: Self-Adaptive BlockchaIn coNsEnsus

Guilain Leduc, Sylvain Kubler, Jean-Philippe Georges
Université de Lorraine, CNRS, CRAN, F-54000 Nancy, France
(e-mail: guilain.leduc@univ-lorraine.fr)

***Abstract*—The Practical Bizantine Fault Tolerance (PBFT) consensus is a well adopted consensus protocol in private and consortium blockchains, the reason being that it is a light protocol, it is not computational intensive (particularly when compared to proof-based consensus protocols), and it allows to reach high transaction throughput. Security in this consensus depends on the number of nodes (aka validators) involved in the consensus: the higher the number of nodes, the higher the security level. However, increasing the number of validators goes along with an increase in the latency, and consequently a decrease in the throughput. The present research proposes a new Self-Adaptive BlockchaIn coNsEnsus protocol (Sabine), which tries to optimally address this trade-off by continuously adapting the size of the pool of validators with the prime objective that the output throughput (i.e., number of transactions validated at a given time) meets – *to the best extent possible* – the input throughput (i.e., requested by the application). Sabine is evaluated and validated in real-life settings, whose results show that Sabine has a relative error of 7.79% between the requested and committed transaction throughput, compared to 39.5% for a classic chain with all nodes participating in the consensus, implying a more reactive chain, with less latency.**

***Index Terms*—Blockchain, BFT, Security, Consensus, Control Theory.**

I. Introduction

Blockchain technologies are distributed ledger which are increasingly adopted in various sectors [1] such as energy, finance or supply chain, as they provide a shared, immutable and transparent history of all the transactions to build applications with trust. One of the key building blocks of any blockchain is the consensus protocol, which is in charge to guarantee the integrity of the ledgers by defining a total order on newly appended blocks of transactions. There are two main families of consensus protocols: *Proof-based protocols* (e.g., Proof of Work, Proof-of-Stake, Proof of Importance, etc.) and *BFT protocols* [2]; the former requiring a proof of thrust based on the cost or the difficulty to propose a block, while the latter relies on a common agreement among a pool of nodes (aka, validators). The logic and functioning of these two families of consensus result in different performance characteristics, whether in terms of scalability, throughput, consistency or permission (e.g., throughput of a BFT chain is higher than a Proof-based one). Overall, each consensus family has specific *pros* and *cons*, which are often in opposition to one another.

A new type of blockchain consensus, called *Hybrid BFT-based Algorithms*, is emerging [3], which aims at combining the *pros* of each family. It consists of first using a Proof-based consensus to select a temporary set of trusted nodes among all nodes of the networks (based on various criteria such as reputation or stake), and then applying a BFT consensus to ensure strong consistency, liveness and safety among validators. While BFT consensus protocols are energy efficient and fast, they are costly in terms of "communications", since every block proposal needs the agreement of more than ⅔ of the validators to be committed. The number of messages to gossip this agreement is initially quadratic. As a result, if a BFT consensus protocol provides higher throughput than proof-based ones, large-scale deployment leads to a substantial decrease in the throughput. It is the reason why BFT networks are limited to a maximum of a hundred validators, setting a limit on the security. Indeed, as BFT protocol tolerates up to a third of adversaries among validators, a greater network can tolerate more adversaries 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. But 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 II further discusses existing Hybrid BFT-based Consensus protocols, along with their limitation to continuously meet the requested input throughout. Section III presents the problem that should be solved in order to overcome this limitation. Section IV presents Sabine, which solves that problem. Sabine is then evaluated and validated through real-life experiments in section V; conclusions follow.

II. Validator Selection

In BFT-based blockchain platforms like Hyperledger or Sawtooth-PBFT [13], the selection of validators is usually defined during the setting up of the platform, resulting in a fixed number of validators throughout the operational use of the platform defining a fixed number of validators. This type

TABLE I
LIST OF BLOCKCHAIN

Chain Name	When	Whom	How Many
Albatross [4]	Fixed Size Rounds	PoS + VRF	
Algorand [5]	Each Rounds	PoS + VRF	$> 500k$
ASHWAChain [6]	Each PoW commit	PoW	
ByzCoin [7]	Fixed Size Rounds	PoW	1004 (eval)
DRBFT [8]	//	PoS + VRF	$10M$
Gosig [9]	Each Rounds	VRF	$> 10k$
Imporoved PoS [10]	Consensus failure	PoS + VRF	100
LinSBFT [11]	Fixed Size Rounds	PoS	64
Tendermint [12]	Each Round	PoS	125
Sawtooth [13]	//	//	

TABLE II
NOTATION

Var.	Description
n	The number of validator in the chain
N_{limit}	Minimum number of validators
$\mathcal{N}$	The number of nodes
d_{req}	Requested transaction throughput
d_{commit}	Committed transaction throughput
lag	Overall Delay (incl., network delay hardware delay...)
M	Model obtained by ML, giving the maximum committed transaction throughput according the number of validator and delay
$BlockSize$	Maximal number of transaction per block
b	The committed block throughput
$\mu(d, n_{val})$	Return the maximal latency such that the model M restricted to the number of validator n provide a capacity equal to d
$\varphi(d, lag)$	Return the maximal number of validator such that the model M restricted to the latency lag provide a capacity equal to d

of platform/strategy is commonly referred to as *Consortium Blockchain* [3], which moves away from a truly decentralized blockchain solution, and in large networks, the consensus becomes nearly equivalent to a PoA. Central authorities like banks [14] are often selected as validators to ensure the trust of the committee of validators. If the pool of validators must be updated, the intervention of the central authority is required with closed negotiations between validators.

A second strategy consists in selecting a random set of validators after a specific time period, which prevents it from having a trusted third party, as was the case with the first strategy. The main condition to be met is that all nodes must agree on the next pool of validators, which is where Proof-based protocols come into play. A partial selection of state-of-the-art strategies are reported in TABLE I, which are further introduced in the following. The TABLE I presents multiple chains with: when the set of validator is changed, with which protocols associated (detailed in the next subsections) and with how many nodes the solution has been tested (in the associated paper).

A. PoW

PoW (*Proof-of-Work*) protocol avoids Sybil Attacks by ensuring that the membership and the participation in the protocol are limited by the hardware capacity [15]. ByzCoin [7] and ASHWAChain [6] use this protocol to select a core of validators in order to protect the chain from these kinds of attacks. They proposed to merge a PoW and PBFT-based protocols (appending a new block in the PoW chain gives the right to take part to the consensus in the PBFT chain). In Byzcoin, a sliding window approach removes validators on the PBFT chain in order to keep a fixed number of validators, statistically represented in proportion to their hash power.

B. PoS

Selecting validators based on PoW is energy-intensive. To overcome this issue, PoS (*Proof-of-Stake*) can be applied instead, as proposed by Tendermint et al. [12]. In their approach, each node that wants to take part to the consensus must make a security deposit using the chain's cryptocurrency, and the 125 nodes with the highest deposits are then selected and included into the pool of validators. The *proposer* of each epoch is selected according to a deterministic round-robin algorithm, which selects the *proposer* according to the relative weights of the stakes. The LinSBFT blockchain presented in [11] adopts a similar PoS strategy.

C. VRF

Randomness is a critical issue in secured distributed systems. As blockchains are secure and deterministic distributed ledger, all nodes need to agree on an unbiased mechanism in a deterministic environment thus not leaving much room for randomness. Verifiable Random Functions (VRF) [16] are pseudo-random functions providing publicly verifiable proof of the correctness of the output. They are used to simulate a pseudo-random mechanism that can be used to select validators. VRF needs to ensure the following properties: uniqueness, full collision resistance, pseudo-randomness and public verifiability. Algorand [5] is based on a PPoS (Pure Proof-of-Stake) consensus, which mixes PoS and VRF. In order to avoid the PoS situation where token are concentrated in a small group, 'The rich get richer', PPoS is based on *cryptographic sortition*. At each round, a VRF is computed based on a shared seed based on the previous block and the pseudo-random output is used to select the validator of the new round. In order to counter Sybil Attack, the validator selection is weighted with the owning of the embedded cryptocurrency. With this system, even the poorest nodes have the possibility to take part to the consensus. The Albatros blockchain [4] uses a similar process, where a random number obtained with a VRF is used to select a list of validators based on a list of nodes weighed with tokens stalked by that node. Wu et al. propose a similar protocol but use a metric based on hardware and network performance instead of a cryptocurrency token [10]. Likely to Algorand, Gosig [9] uses a VRF to select a list of *potential proposers*, but with the difference that all nodes has the same weight to be selected. Finally, in DRBFT [8], a voting system is present between nodes and a VRF is used to

select a set of validator among nodes with the highest number of vote.

All the previously introduced blockchain solutions are based on the idea that a small group of validators is faster than a large group, but the current state of affairs does not highlight a blockchain with a varying size of the validator group. As was explained previously, performance of a BFT blockchain is intrinsically linked to the size of the pool of validators. To the best of our knowledge, no study has ever proposed a blockchain consensus protocol that makes it possible the self-adaptation of the pool size over time in order to meet the input throughput.

III. Problem Statement

All the blockchain solutions previously discussed do 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, latency, security, ...) of the chain whatever the needs, assuming nodes are homogeneous. As was already mentioned in section I, 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) according to the statement of table II.

$$\underset{\mathcal{N}}{\operatorname{argmax}} \left(n^t \mid d_{commit}(n^t, lag^t) = d^t_{req}\right) \; s.t.\; n > N_{limit} \quad (1)$$

For this purpose, Sabine adjust the number of validators of the chain n^t in real time t in order that the *Capacity*, i.e., the maximum throughput that the chain reach, is close to the recent requested throughput, implying that the requested throughput d_{req} is equal to the committed throughput d_{commit}, whatever a delay lag is present in the network.

IV. Sabine steps

Due to the specificity of each network where Sabine can be applied, it is not possible to identify a mathematical model taking into account the link capacity, the delays and the number of validators. Falling that, Sabine uses Machine Learning to link this value and control the number of validators. The principles and the execution of Sabine follow the steps presented in the figure 1. Three main steps emerged and are discussed in the current section: a training phase, detailed in subsection IV-A, is required to build a network-specific model, linking capacity, delay and the number of validators. This model is used by the controller of Sabine, detailed in subsection IV-B1, to estimate the ideal number of validators, which is applied in the chain by the mechanism detailed in subsection IV-B2.

A. 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 throughput 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 as close 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 $(capa, lag, n)$ and if a set of triplets is large enough, it is used as a training dataset to build a model which associates the number of validators, the delay and the capacity of the chain using machine learning based on a polynomial regression. The given model M provides a mostly decreasing throughput $M(n, lag)$ according to the number of nodes n and the delay lag, and a post filter treatment is applied to assert the decrease.

$$\frac{\partial M}{\partial lag}(n, lag) \leq 0 \quad (2)$$

$$\frac{\partial M}{\partial n}(n, lag) \leq 0 \quad (3)$$

B. Operational Phase

1) Controller: 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.

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 4 and 3 can be defined:

$$\mu(d, n) = max(\{lag | M(n, lag) = d\}) \quad (4)$$

$$\varphi(d, lag) = max(\{n | M(n, lag) = d\}) \quad (5)$$

In order to estimate an ideal number of validators in this environment, Sabine estimates the delay based on the simulated delay of the model obtained by ML. This calculus follows the scheme 4. 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 lag_{est} is estimated by taking the simulating delay that meets with the actual capacity of the chain (green line in fig 2). The committed throughput is not a good metric to estimate the actual capacity because,

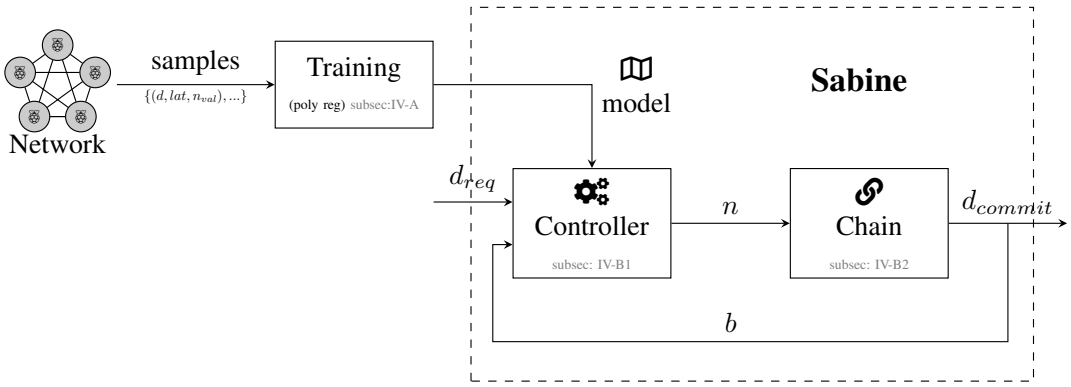


Fig. 1. Phases of Sabine

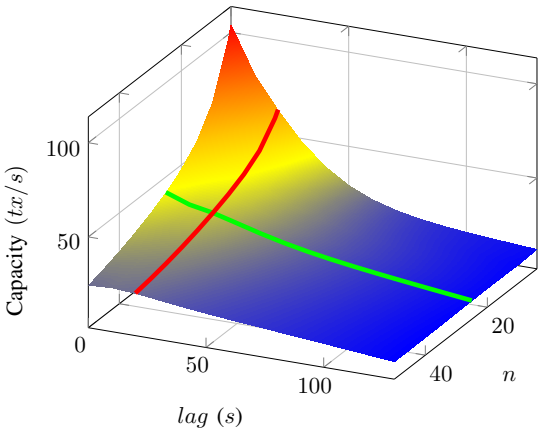


Fig. 2. Estimated Model

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

$$lag^t_{est} = \mu(b^t \cdot BlockSize, n^{t-1}) \tag{6}$$

Then, the model is restricted to the estimated delay lag_{est}, forming a decreasing bijection between the maximum throughput and the number of validators (red line in fig 2). The ideal number of validators is obtained by comparing the requested transaction throughput d_{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.

$$n^t = \varphi(d^t_{req}, lag^t_{est}) \tag{7}$$

2) Action on the Chain: Sabine is launched in every node of a chain to keep the system decentralized. After the end of a time-out without validator changes, the Proposer of the blockchain applies the Sabine algorithm to determine the new ideal number of validators, which means that Sabine is called periodically by one node of the network. Once this number is determined, a special transaction is inserted and committed firstly in the chain to transmit to all nodes the update of the number of validators and modification are applied after the transaction commitment. The selection mechanism for new validators depends on the chain and all mechanisms listed in the table I can be adapted. In our example chain, no particular mechanism is implemented, nodes are ordered on a list and a pivot separates validators from non-validators. Using a transaction in a BFT chain to transmit the validator update ensures the consistency of the modification for the next blocks.

V. Experimentation

Sabine has been implemented and adapted to a BFT chain[1] running on a Raspberry PI platform. The chain and the platform is described in subsection V-A. Results have been extracted from this experimentation and are detailed in subsection V-B.

A. Chain Description

The initial source of inspiration of the tested BFT Chain is the Ethereum's EIP 650 which describe the *Istanbul Byzantine Fault Tolerance consensus* [17]. It is a state machine which realized the original PBFT consensus of Castro-Liskov [18] and follows the state diagram represented in figure 3. The added states correspond to state where a node is non-validator. 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.

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

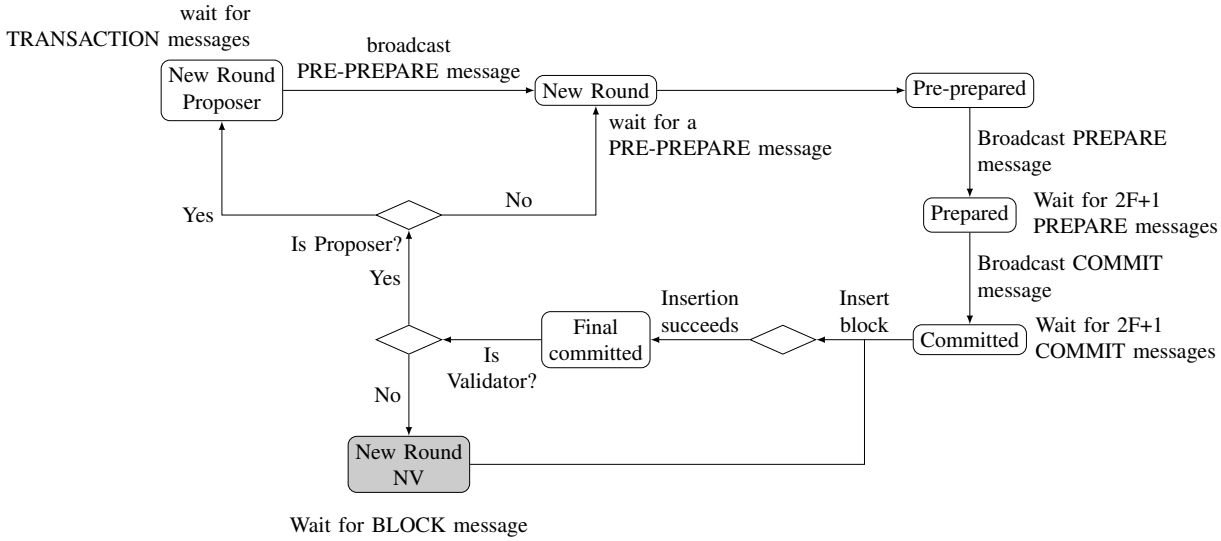


Fig. 3. Representation of scaling mechanism of BFT based blockchains

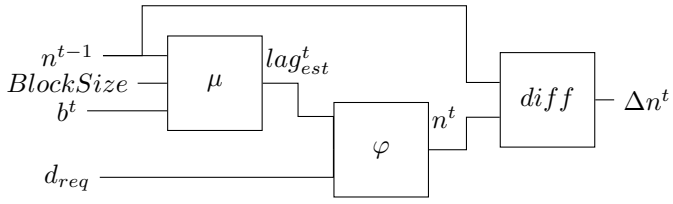


Fig. 4. Sabine Calculus explained with a diagram, each block represents a function detailed in subsection IV-B1

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.

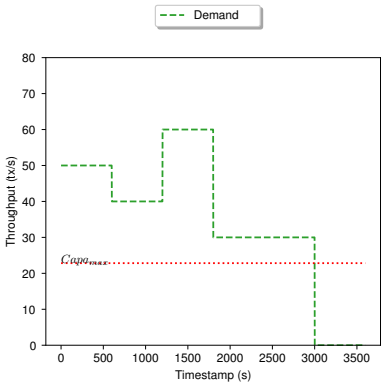
Nodes are deployed with Docker Swarm on a cluster of 10 Raspberry PI 4. Around 50 nodes are deployed on dockers with dedicated resources (0.5 CPUs and up to 1 Gb of Ram per node) to consider independent nodes on a single machine. Raspberry PI are directly connected with a dedicated switch in order to minimize network latency. Transactions are generated and gossiped by an external computer without resource limits to keep stable the transaction request throughput. They consist of the same string with a growing salt. Transactions, and therefore blocks, are all the same size.

B. Results

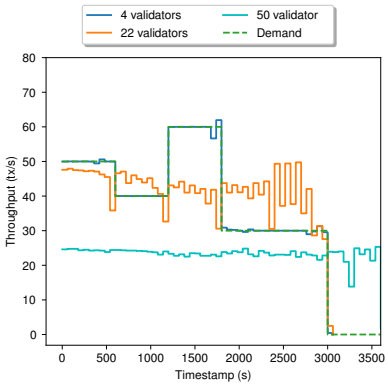
The figure 5a shows an example of a (throughput) demand that the chain must support. The proposed control is designed to achieve this support by determining an appropriate number of validators between the minimum and the maximum numbers of validators. At the end of each experiment, the demand is fixed to zero in order to highlight the presence of stacked transactions in the chain and let the chain the time for committing these transactions. The demand's variations remain close to the capacity of the maximal number of validators since it corresponds to the area the most sensitive to the number of validators (shown in figure 2).

Figure 5b shows the difficulty of finding a perfect number of validators for a chain to make its committed transaction throughput meet the demand. Considering few validators (like 4, the minimal in BFT consensus) helps to satisfy correctly the demand with a low relative error of 0.58% between the requested and committed (estimated on a window of 60s) throughputs, but the security is reduced because only one single malicious validator is tolerated. However, setting the security to the maximum number of validators (50 nodes, so it can tolerate 16 malicious validators) won't satisfy the demand without a high delay and with a high relative error of 39.5% before the end of demand (at $t = 3\,000\ s$). Finding offline a perfect number of validators need the knowledge of the average requested transactions throughput and cannot adapt to its variation. Indeed, in this example, the average throughput of the demand can be satisfied by 22 nodes, but the relative error of a chain with this number of validators is still high since equal to 24.1%. Those errors are mainly due to requested throughput variations stages.

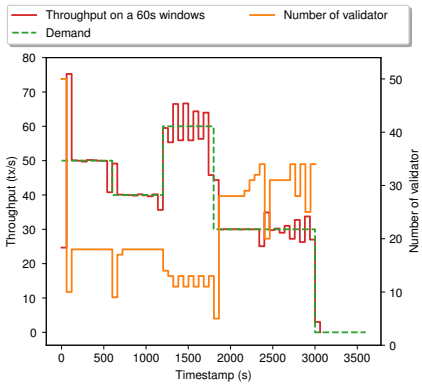
The figure 2 represents the model obtained by Machine Learning on the experimentation bench described above (with a mean absolute error of 0.465). As expected, the *Capacity*, i.e., the maximal committed throughput, decreases with the number of validators and the simulated delay. This study shows that the Sabine algorithm control is interesting if the desired throughput is between the capacities at the number of validator limits. If it is lower, the throughput is committed without need of adaptation, higher, the committed throughput is limited by the capacity with the minimum of validator (4



(a) Demand used to benchmark chains

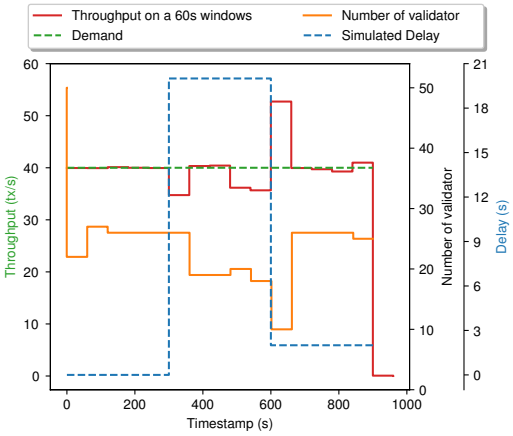


(b) Responses to the demand 5a with a fixed number of validators

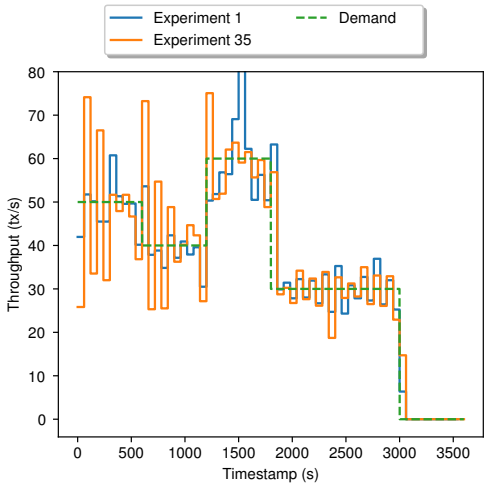


(c) Sabine committed throughput and number of validators for the demand 5a

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

in our study, the BFT minimum).

With this model, the chain can adapt itself to a demand like represented in figure 5a, with the response shown in figure 5c. Here the number of validators is adapted online to the incoming transaction throughput. As results, the committed transaction throughput is around 7.79% of the requested throughput (corresponding to an absolute error of 1.71 transactions) and with an average number of 21.3 validators. The time needed to estimate an ideal number of nodes is equal to $60s$, which correspond to an occurrence of Sabine's control.

Taking into account an arbitrary delay permits to adapt the chain in case of the emergence of an unknown delay. To illustrate this property, a delay is introduced and vary in a run of a chain. The figure 6a shows the response of the chain with Sabine for a fixed transaction request throughput with a change of the delay. The relative error between the demand and the committed throughput in this scenario is now equal to 4.87% in average. As a consequence, our control adapt the validators pool by decreasing the number in order to accelerate the committed throughput and compensate the slowdown due to the delay.

With this model, Sabine is able to manage the number of validators according both to the demand and the delay variations. 35 chains associated with Sabine were deployed and submitted to the demand 5a with a random delay different for each experiment. This delay is around $15s$ and changes every 150 s in average. Two of these experiments are represented in figure 6b (all experiments are not represented for visibility). Results show that the committed throughput is closed to the demand. Decreased due to delay are quickly compensated because the relative error between the demand and the committed throughput remains equal in average to 17.45%. This compensation of the delay was managed by a reduction of the number of validators to 18.2 in average (with

a standard deviation of 3.0 between experiments).

VI. Conclusions

We have performed the first control on the number of validators in a BFT chain in order to adapt the capacity of the chain to its need, based on the requested throughput while keeping a security limit. In this paper, we proposed a new Self-Adaptive BlockchaIn coNsEnsus protocol (Sabine) and we showed that a BFT chain using Sabine can increase it security during a period of low request by increasing the number of validators. Real experimentations and benchmarks show that Sabine significantly increased the satisfaction of demand throughput rates while sill covering security concerns.

In our study, Sabine was used on a chain which uses a classic PBFT protocol but the control can be adapted to more efficient BFT protocols (for which performances are also limited by the number of validators). Sabine can also be adapted on chains which solve the problem of selection of validator like Hybrid BFT-based Consensus. These consensuses solve the problem of which validators to choose, Sabine how many.

Future work may consider include other criteria such as the energy efficiency. BFT protocol are energy efficient but the energy spent in the protocol increase with the number of nodes involved in the consensus. By taking into account the energy to define the model like a new dimension, Sabine could adapt the number of validator in order to respect an energy limit.

Acknowledgement

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

References

[1] M. B. Mollah, J. Zhao, D. Niyato, K.-Y. Lam, X. Zhang, A. M. Y. M. Ghias, L. H. Koh, and L. Yang, "Blockchain for Future Smart Grid: A Comprehensive Survey," *IEEE Internet Things J.*, vol. 8, no. 1, pp. 18–43, Jan. 2021.

[2] L. Lamport, *Time, Clocks, and the Ordering of Events in a Distributed System*. New York, NY, USA: Association for Computing Machinery, 2019, p. 179–196.

[3] M. Belotti, N. Božić, G. Pujolle, and S. Secci, "A vademecum on blockchain technologies: When, which, and how," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3796–3838, Fourthquarter 2019.

[4] P. Berrang, P. von Styp-Rekowsky, M. Wissfeld, B. França, and R. Trinkler, "Albatross – an optimistic consensus algorithm," in *2019 Crypto Valley Conference on Blockchain Technology (CVCBT)*, June 2019, pp. 39–42.

[5] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*, ser. SOSP '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 51–68.

[6] S. Arora, A. Jain, and S. Gujar, "Ashwachain: A fast, scalable and strategy-proof committee-based blockchain protocol," in *Workshop on Game Theory in Blockchain at WINE*, vol. 2020, 2020, p. 9.

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

[8] Y. Zhan, B. Wang, R. Lu, and Y. Yu, "Drbft: Delegated randomization byzantine fault tolerance consensus protocol for blockchains," *Information Sciences*, vol. 559, pp. 8–21, 2021.

[9] P. Li, G. Wang, X. Chen, F. Long, and W. Xu, "Gosig: A scalable and high-performance byzantine consensus for consortium blockchains," in *Proceedings of the 11th ACM Symposium on Cloud Computing*, ser. SoCC '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 223–237.

[10] Y. Wu, P. Song, and F. Wang, "Hybrid consensus algorithm optimization: A mathematical method based on pos and pbft and its application in blockchain," *Mathematical Problems in Engineering*, vol. 2020, p. 7270624, Apr 2020.

[11] X. Qi, Y. Yang, Z. Zhang, C. Jin, and A. Zhou, "Linsbft: Linear-communication one-step bft protocol for public blockchains," 2020.

[12] E. Buchman, J. Kwon, and Z. Milosevic, "The latest gossip on bft consensus," *CoRR*, vol. abs/1807.04938, 2018.

[13] F. Hyperledger, "Introduction to hyperledger sawtooth," 2022. [Online]. Available: https://sawtooth.hyperledger.org/docs/1.2/

[14] Consensys, "Quorum," Nov. 2021, original-date: 2016-11-14T05:42:57Z. [Online]. Available: https://github.com/ConsenSys/quorum

[15] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, 10 2008.

[16] S. Goldberg, J. Vcelak, D. Papadopoulos, and L. Reyzin, "Verifiable Random Functions (VRFs)," p. 24, 2018.

[17] Y.-T. Lin, "Istanbul Byzantine Fault Tolerance · Issue #650 · ethereum/EIPs," June 2017. [Online]. Available: https://github.com/ethereum/EIPs/issues/650

[18] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, ser. OSDI '99. USA: USENIX Association, Feb. 1999, p. 173–186.