Experimental Scalability Study of Consortium Blockchains with BFT Consensus for IoT Automotive Use Case

Luc Gerrits
Cyril Naves Samuel
Roland Kromes
François Verdier
luc.gerrits@univ-cotedazur.fr
cyril-naves.samuel@etu.univ-cotedazur.fr
roland.kromes@univ-cotedazur.fr
francois.verdier@univ-cotedazur.fr
Université Côte d'Azur, LEAT / CNRS UMR 7248
France

Severine Glock
Patricia Guitton-Ouhamou
severine.glock@renault.com
patricia.guitton-ouhamou@renault.com
Renault Software Labs
France

ABSTRACT

Private or consortium blockchain networks have fewer verified participants and offer better throughput and transaction efficiency than public networks. However, as more and more blockchain consensuses are designed for private or consortium networks, their performances are often estimated without a practical use case implementation. In our use case, participants do not have to trust each other but still work together to build an ecosystem where users control their data and information. This paper analyzes the performance (transaction throughput, rejections, node participants) of Byzantine Fault Tolerant Consensus (BFT) using two blockchains: Hyperledger Sawtooth and Ethereum.

CCS CONCEPTS

• **Computer systems organization → Distributed architectures**.

KEYWORDS

Blockchain, Ethereum, PBFT, Clique, IBFT, QBFT, Cloud

1 INTRODUCTION

Since the birth of the Bitcoin [13] network in 2008, new decentralized ledger technologies (DLTs) have emerged to enable new use cases. Moreover, through smart contracts, blockchains can execute custom business logic in a decentralized network. The consensus algorithm, allowing all network peers to agree on each transaction, is a crucial part of any blockchain and determines the entire network's security, scalability, and consistency. Thus consensus algorithm plays a significant role in transaction speed limit and network resilience to byzantine participants. Blockchain requirements are different depending on the use case, but these two previous features are crucial elements for worldwide adoption.

This paper focuses on the performances (i.e. transaction processing, transaction throughput, forks, and scalability) of blockchain in a consortium or private configuration. For this reason, we have targeted only BFT consensuses that are the most used in this private/consortium type of use case, which require to be resilient to faulty nodes and maintain consistency in the network. BFT consensus algorithms are designed for networks that require limited trust among a predefined number of validators. Although the network participants are assumed to be verified actors in the private/consortium network, the nodes can be subject to communication or faulty nodes which demands the need for BFT algorithms.

Study on BFT consensuses already trends to show scalability limitations, thus it is not used for public blockchain networks. When increasing the number of nodes in the private/ consortium network, the transaction throughput decreases [20]. Schäffer *et al.* [19] studied the Ethereum private network throughput bottleneck showing an average of 328 transactions per second. The authors use a variety of configurations and conclude that Ethereum has limited scalability.

This paper vary the blockchain input workload similarly to the reasearch of Pongnumkul *et al.* [15]. The authors have tested Ethereum private network and concluded a maximum of 38.93 transactions per second. Considering the results are dated from 2017, it should be noticed that we expect a better transaction rate due to software and hardware improvements. Also, we notice a lack of practical implementation with a industrial use case. We aim to get practical results using a cloud architecture to verify the limits and discover the maximum capabilities of new BFT consensuses.

1.1 Use case

In this work, the selected use case of Renault is an accidentology scenario represented in Figure 1. Here the blockchain network is defined as a consortium network of partners involved in an accident claim and alert process. Our node participants (i.e. validators) are Renault - Original Equipment Manufacturer (OEM), Accident Insurance Providers, Medical Service, State Actors for legal and dispute resolution, Police and other emergency services.

The novel concept of this use case is to enable the blockchain as a distributed ledger that stores the accident's transactions. Accident's transactions details are the latest vehicle speed, vehicle condition, radar information, driver condition, etc. These are contextual data stored in a database maintained by the consortium partners. The hash of this data is stored as proof-of-existence on the blockchain.

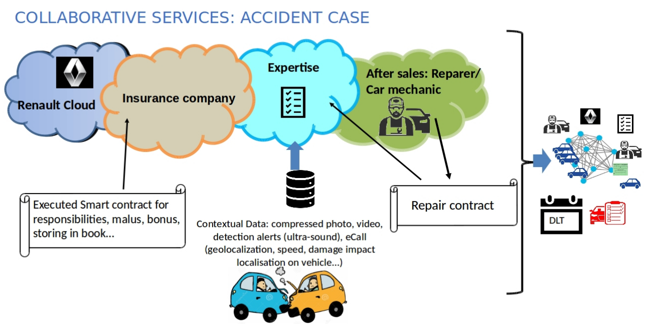
COLLABORATIVE SERVICES: ACCIDENT CASE
Renault Cloud
Insurance company
Expertise
After sales: Reparer/ Car mechanic
Executed Smart contract for responsibilities, malus, bonus, storing in book...
Contextual Data: compressed photo, video, detection alerts (ultra-sound), eCall (geolocalization, speed, damage impact localisation on vehicle...)
Repair contract
DLT

Figure 1: Renault accident use case

Three permissions roles are implemented using smart contracts that allow a vehicle to send valid transactions and be part of the blockchain network. These permissions are: 1) the admin who deploys the smart contract and has a "super-user" role in the consortium network 2) the factory of OEM: Designated Factories who add the vehicle when it leaves the production line 3) the vehicle owner: Individual owner who reports the accident.

If an accident occurs, the vehicles can send data through transactions to the blockchain network using smart contracts. The embedded microcontroller inside the vehicle can send data such as the car speed, radar information, odometer, etc. This collaborative blockchain technology aims to prevent drivers from fraud, allowing automated refunds and verifiable data for insurances, enabling an ecosystem of accidentology partners to exist.

This paper focuses on implementing a consortium blockchain satisfying the above use-case and studying its performances. A consortium blockchain is deployed on a private cloud infrastructure to simulate a real-world industrial environment. The number of nodes in the network represents the use case ecosystem's number of participants (i.e. Renault, insurance, etc.).

The use case's security perspectives (e.g. GDPR and authentication) highly depends on the smart contract design and data management. Therefore, the paper only implements a basic role base permission logic to prevent unauthorized transaction execution. BCTrust, created by Hammi *et al.* [9], is one of many blockchain authentication mechanisms. They focus on IoT integration and present the power consumption and execution time of their solution. Furthermore, Haque *et al.* [10] conducted a systematic literature review about GDPR blockchain compliance. As a result, they show GDPR-blockchain research trends and articles addressing the problem of GDPR compliance integration with blockchain.

1.2 BFT consensuses

The Byzantine fault-tolerant (BFT) consensus family ensures network liveliness and security even when some nodes are faulty or acting malicious, as long as a minimum number of nodes are connected, operating correctly, and behaving honestly. BFT assumes an asynchronous network where there may be network failure or individual node failures. In our work, we consider a family of BFT algorithms: Practical Byzantine fault-tolerant (PBFT), Proof of Authority (PoA), Clique, IBFT, and QBFT.

PoA algorithms have less message communication compared to PBFT but suffer from consistency and availability problems in an asynchronous setting [5]. We analyze this family of consensus algorithms through our industrial, automotive (Renault) use-case.

2 BFT BLOCKCHAINS

2.1 Hyperledger Sawtooth PBFT

The Hyperledger blockchain framework family, created by the Linux Foundation Hyperledger, developed Sawtooth as a modular and enterprise-focused blockchain [11]. Sawtooth blockchain consists of a validator, the core of the blockchain peer, one or multiple transaction processors handling transaction business logic, and a REST API providing convenient HTTP communication with the peers. Moreover, Sawtooth supports two main consensus algorithms, Proof-of-Elapsed-Time (PoET) and Practical-Byzantine-Fault-Tolerance (PBFT).

Practical Byzantine Fault Tolerance consensus is a voting-based algorithm which implementation in Sawtooth is based on the work of Barbara Liskov, and Miguel Castro [3]. PBFT properties are Byzantine fault-tolerant, non-forking, leader-based, and communication-intensive.

2.2 Ethereum Geth (Clique)

Ethereum, one of the most prominent public blockchain, has a mechanism to build private networks with a proof-of-authority (PoA) algorithm named Clique [16] implemented in Geth client. In the algorithm, the creation of a new block is restricted to a fixed set of n nodes called sealers, in which a maximum of $f < n/2$ can be faulty nodes or *Byzantine*. Every sealer can seal a block at a fixed time, but it has to wait until it is not sealed recently for $(n/2) + 1$ blocks until its last block. If a designated sealer signs a block for a particular sealing round, it is termed as in-order sealing. On the other hand, if the designated sealer is subject to byzantine conditions and cannot seal a block, any other sealer may propose a block after waiting for the block period termed as out-of-order sealing. Out-of-order sealing can occur quite frequently in case of short block period times and large network time delays between nodes. If the previous conditions are met, it can eventually cause more forks in the network, which is an issue in Clique and lacks chain finalization compared to other consensus algorithms.

2.3 Ethereum Hyperledger Besu (IBFT)

Hyperledger Foundation has created Besu, a client who has implemented the IBFT [2] consensus algorithm, which has immediate finality of the chain. The creation of a single block at a particular height avoids the problem of the forks. Also, the need for $n/3$ majority for completing consensus makes a forked chain less probable. In this algorithm, out of a set of n validators, an arbitrary node is selected to be a block *proposer*. If the other validators accept the proposer as a block creator and validate the block, it is considered accepted. To avoid the creation of multiple blocks, a block locking mechanism is introduced when a super majority of validators accept the block proposition. Then a new round change is proposed with a new validator for the next block creation.

2.4 Ethereum Hyperledger Besu (QBFT)

QBFT algorithm was created to avoid safety and liveness issues such as [1] [17] in IBFT protocol where two valid nodes can lock different blocks at the same height. This can be attributed to transmission delay between the nodes and there is no provision to unlock the blocks in the consensus algorithm. Due to these drawbacks, the Quorum blockchain has developed a variant of IBFT also termed as QBFT algorithm [12]. For each round, a block proposal is created who broadcasts a pre-prepare message to the rest of the validators. Other validators on receiving the pre-prepare message broadcasts a prepare message. On receiving pre-prepare message, it then sends a commit message. The majority for each of the state (pre-prepare, prepare and commit) is thus $2N/3$. Finally, a new block is inserted by the proposer into the blockchain after $2N/3$ commit messages. Then the next round is invoked after $2N/3$ round changes.

Considering the above four BFT consensus algorithms, we compare them across different desirable properties in a distributed system as in table 1.

Property	**Clique**	**QBFT**	**IBFT**	**PBFT**
Binary	Geth	Besu	Besu	Sawtooth
Chain-Finality	Prob*	Det*	Det*	Det*
Forks	Yes	No	No	No
Block-Locking	No	No	Yes	No
Chain-Reorganisations	Yes	No	No	No
Liveness	Upto $N/3$ failures	Upto $N/3$ failures	Upto $N/3$ failures and Prone to dead-lock	Upto $N/3$ failures
Throughput	High	< Clique	< QBFT	< QBFT

Prob*: Probabilistic , Det*: Deterministic

Table 1: Comparison of BFT Consensus Consortium Blockchains

3 EXPERIMENT SETUP

3.1 Hyperledger Sawtooth

In previous research, Gerrits *et al.*[7] showed the performances limits of Hyperledger Sawtooth in the same car accident context as this paper. The experiment setup uses a cloud infrastructure, and a distant client sends transactions (i.e, input transaction per second, TPS) according to the use case. The input TPS and different implementation configurations demonstrate the software and PBFT consensus limits.

3.2 Ethereum

In Ethereum, the client binaries chosen for constructing the private consortium networks are Geth v1.10.7 for Clique and Besu v21.7.2 for IBFT and QBFT consensus algorithms, respectively. The Ethereum implementation of the car accident use case is designed as three layers. They are:

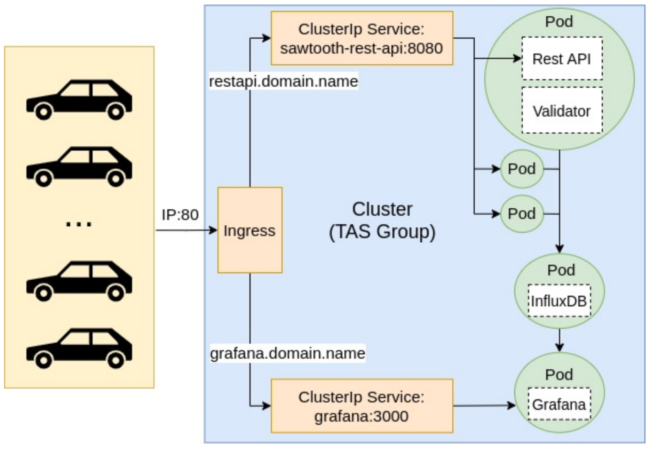

Figure 2: Hyperledger Sawtooth network cloud deployment

(1) Consortium Network Layer: Each participant in the consortium is deemed to be a sealer in the blockchain private network. These sealers are considered validators who participate in consensus to validate transactions and create blocks. A boot node is installed to connect the other nodes in the blockchain peer-to-peer network. The network enables the smart contract to be deployed via the Application programming interface. The private network is configured to be of negligible gas price as we are focus only on the enterprise use-case than a cryptocurrency transactions.

(2) Accident Smart Contract: Smart Contract deployed on the Ethereum network executes on the Ethereum Virtual Machine of each node. Smart contract is built using solidity. Smart contract creates four types of roles in the blockchain network. They are 1) Admin 2) Factory Admins 3) Cars. Using the smart contract, permission system is enabled in which an admin can deploy the smart contract and adds the Factory Admin Nodes who can participate in the network. The Factory Admins can add the vehicles which can participate in the network and send the accident transaction.

(3) Client Implementation: A use-case accident smart contract is deployed by the manufacturer entity. Next, a client based out of web3 library is built to communicate with the blockchain network. This client will be embedded in the Electronic Control Unit of the vehicle and it sends the smart contract transaction to the network via WebSocket API. In addition, client is equipped to send permission transactions to add new participants, send accident transactions and retrieve the latest accident hash.

4 RESULTS

Performance measurement for our Sawtooth - PBFT, and Ethereum - Clique, IBFT, and QBFT are performed in TAS Group cloud infrastructure based in Sophia Antipolis, France. The underlying infrastructure hosts a Kubernetes cluster, enabling us to create Kubernetes pods, each hosting a blockchain node. We monitor the test performance using telemetry from the blockchain node on the Grafana dashboard and MongoDB for test log maintenance.

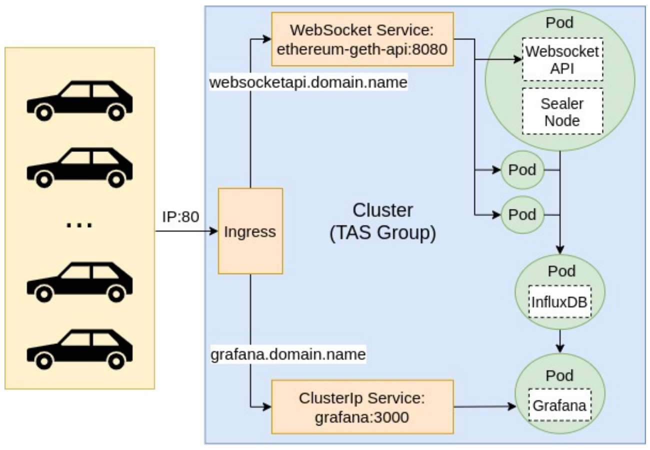

Figure 3: Ethereum network cloud deployment

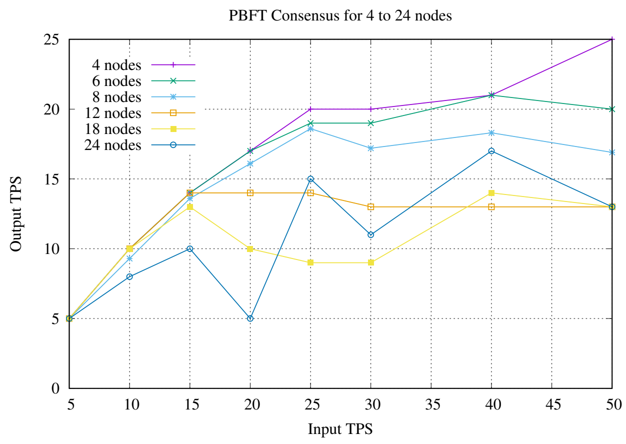

Figure 4: Hyperledger Sawtooth PBFT Consensus Performance

We assume here a partially asynchronous network as there is a minute delay counting the cloud virtualization, Kubernetes inter-pod routing, node computation load, and peer-to-peer factor. We do not consider any threat model to be included in our test and assume a legitimate consortium. We vary the input transaction rate against the number of nodes participating in the consensus. We consider here the output transaction to be transactions that are validated and ordered successfully after the consensus. Format for the test performed in each blockchain varies a bit since the implementation of Sawtooth-PBFT, Ethereum -Clique, IBFT, and QBFT are completely different organizational and design structures. But the fundamental idea is to test the blockchains performances by varying the nodes, also named *scalability*. Performances is also tested by incrementally changing the input TPS (Transaction-Per-Second) submitted to the blockchain node and pushing the maximum threshold it can achieve. Our developed test suite for this paper is available online at [8] for more information about code, configuration and further contributions.

4.1 Sawtooth performance measurement

A previous in-depth studied of Hyperledger Sawtooth revealed a strong limitation of the blockchain transaction processing [7]. This study demonstrates the same use case and smart contract business logic. The simulation consist of sending car crashes using IoT devices.

Figure 4 shows the transaction processing speed of the blockchain by varying the input transaction per second. We see a maximum of 25 transaction per second using the 4 nodes configuration. When increasing the number of nodes, the transaction processing speed decreases down to 13 transactions per second.

The study on Sawtooth also discuss the possible factors reducing the transaction speed. The consensus and the software are the main factors reducing throughput. The results on Figure 4 shows the consensus network limitation with the number of node validators. The software limitation is due to single threaded and intrinsic language (Python) performance issues.

4.2 Ethereum performance measurement

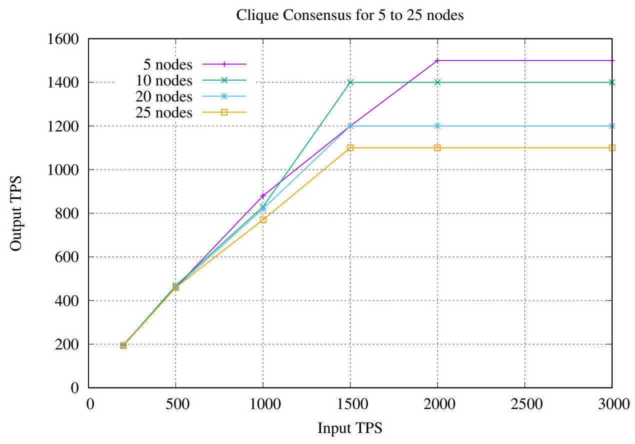

Figure 5: Ethereum Clique Consensus Performance

In Ethereum, we perform the test for accident transactions totaling 30000 transactions for each iteration in 3 iterations and measuring the average throughput results. For Clique, as in Figure 5, we see that the output TPS constantly increases for five nodes until it becomes a plateau at input TPS of 2000 outputting 1500 TPS due to Ethereum EVM (Ethereum Virtual Machine) computation overlead and a minute scalability factor. Minute here signifies that the consensus proportion is less at N/2 and the phase is less compared to other consensus algorithms IBFT, QBFT where we see a significant drop with the number of nodes as in Figure 6 and 7.
In comparison to the work as in [18] where the test was performed on the Microsoft Azure network, we noticed a similar performance for Clique and IBFT. Still, the format of the test varies with no variation across input TPS. Also, the number of clients for IBFT is a single instance that is multithreaded, but in the latter, it was

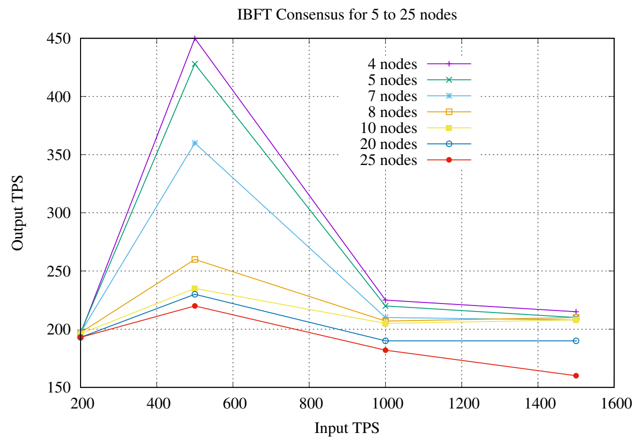

Figure 6: Ethereum IBFT Consensus Performance

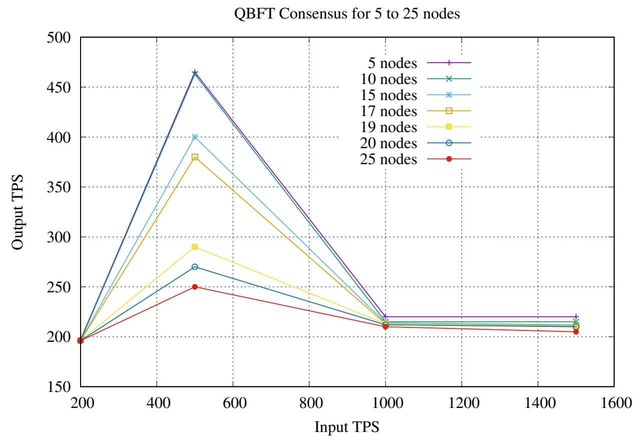

Figure 7: Ethereum QBFT Consensus Performance

multiple instances that might improve the API processing of transactions but not significantly as the transaction are stocked in queue immediately before validation and consensus.

In Clique for more nodes we can see the saturation arriving quickly at input TPS of 1500 as it has less message overhead. In IBFT and QBFT, we see a gradient increase up to 500 input TPS outputting 460 average TPS. At this point, it has a hash calculation load of the Bouncy Castle Library, which it uses in its java implementation. Aside from the computational hash overload, we see a drop as message overload increases, but in IBFT, it is more significant as we have block locking latency in the consensus to consider here. For an input rate of 1000 TPS, we see a substantial drop for both as the consensus and implementation reasons discussed earlier causes it.

In Clique we notice frequent forks and chain reorganization which affects the performance and stability of the chain. On the other hand IBFT and QBFT has no forks in the chain which ensures the finalisation of the chain. In IBFT, we encounter a stalling issue due to the block locking mechanism. Block locking for each consensus round is implemented in IBFT to improve consistency but instead, it affected the liveness of the network. In QBFT, we did not notice any stalling of the network nor the presence of forks in the chain.

5 DISCUSSION AND CONCLUSION

Our analysis of the BFT family of consensus blockchains from the above results can be based on three perspectives: 1) Consistency Availability and Partition Tolerance (CAP) 2) Performance 3) Applicability to Industry. We apply CAP Theorem [6] perspective as in [4], to our above blockchain implementations. We consider the consistency to be fork affinity, ordering, and replicated transactions and messages. Availability means that the blockchain can respond and accept a valid transaction to be added to the chain. Partition tolerance is when the network partition or peering problem doesn't impede the system, and it can recover from it.

CAP	Clique	IBFT	QBFT	PBFT
Consistency	Low	High	Medium	High
Availability	High	Medium	Medium	Low
Partition-Tolerance	High	Medium	Medium	Medium

Table 2: CAP Analysis of BFT Consensus Algorithms

We notice as listed in table 2 that the Clique suffers from consistency issues. Different sealers can have different network views and create a fork, which eventually leads to an unresolved fork affecting partition tolerance. But the chain may progress at N/2 participant consensus, but it would be difficult to finalize as we cannot decide the ephemeral chain. In IBFT, PBFT, and QBFT, it favors more consistency and partition tolerance. As it is subject to multiple phases at each consensus fork is not created, this also brings down the availability. Since it waits for N/3 participant response at each round, it has to wait if a network or node fails. The performance of BFT Blockchain in our results considers many factors, such as :

(1) Number of Nodes
(2) Message Communication
(3) Leader Selection Phase
(4) Consensus Phase Count
(5) Implementation Bottleneck

Number of nodes augments the message communication overload. Also, the leader, selection, and the number of consensus phases in each round are vital factors. The implementation bottlenecks like EVM processing for Geth, Sawtooth Transaction Processor, Besu Bouncy Castle Hash calculation also counts for the drop in transaction processing. So Clique performs better in this case as it has a predefined leader operation in a round-robin mode, with fewer phases in each consensus and a minor EVM implementation problem. IBFT and QBFT have an average impact on all the considered factors, with improvement needed in hash calculation. Sawtooth PBFT has a massive drawback in the implementation part compared to others as well as its consensus has more phases and communication.

We consider the applicability of the blockchain to the Renault automotive use-case, which requires a minimum of 25 transactions

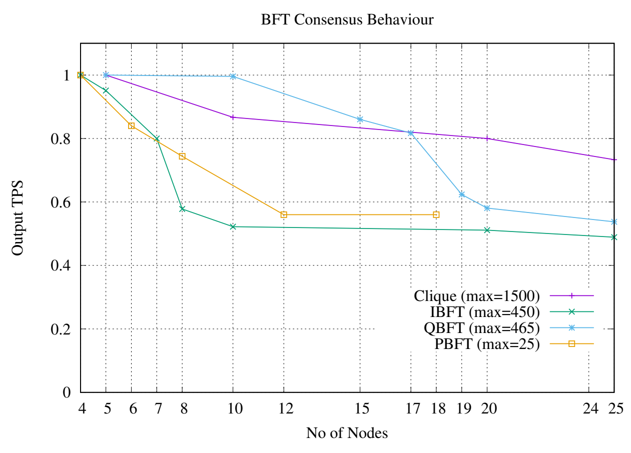

Figure 8: Normalised Output TPS Behaviour of BFT Consensuses

per hour to be processed by the blockchain as per ONISR Report [14]. In our figure 8 we plot the best normalised over the maximum output TPS performance of each variant of BFT blockchain at optimal input TPS to exclude the implementation overload. We consider the Output TPS coefficient of 1 as the best TPS performance for each of the blockchain and 0 as the lowest TPS. Output TPS Coefficient here shows the variation of the performance as the nodes are increased in the consensus process and their behaviour. Globally, in the BFT consensus family, we note that the best performances are obtained when using 4 to 6 nodes. Except for QBFT the descent starts to arrive slowly at 12 nodes but in comparison to Clique the processed transaction throughput is less. In this range of nodes, the transaction processing performances of the entire blockchain is best. It lies in the range desired for the number of validator participants we can have for an accident use case. After we surpass 10 nodes, all the BFT consensus blockchains descend in their performance until they reach a stable rate of output TPS. This can be attributed to the fundamental reason behind all the BFT algorithms of communication overhead to prevent byzantine faults. Even though each BFT consensus algorithm varies in the number of phases and leader election, they have the same behavior. We consider for each blockchain consensus its best performance with optimum participants, and extrapolate the transaction supportable by the network for an entire day, we get the table 3. It shows the applicability check of the consensus to the automotive use-case in general as well as the accident use-case.

Applicability	Clique	IBFT	QBFT	PBFT
TPS per day	129600000	38880000	39744000	2160000
Ideal for Accident Use-Case	Yes	Yes	Yes	Yes
Ideal for Automotive Use-Case	Medium	Medium	High	Low

Table 3: BFT Blockchain Applicability transactions

As in the above table, we can conclude that Clique has high performance but suffers from consistency issues, QBFT has an average performance with better scalability, and PBFT has less performance but more consistency. Based on these results and conclusions, BFT consensus is suitable for this automotive use case with some improvements needed, but it depends on the design choice factor to consider CAP, performance, or applicability properties.

ACKNOWLEDGMENTS

This work has been supported by the French government through the UCA^{JEDI} and EUR DS4H Investments in the Future projects managed by the National Research Agency (ANR) with the reference number ANR-15-IDEX-0001 and ANR-17-EURE-0004.

REFERENCES

[1] M. Chuan. [n.d.]. *Istanbul BFT's design cannot successfully tolerate fail-stop failures.* https://github.com/ConsenSys/quorum/issues/305

[2] Y.-T. Lin. [n.d.]. *Istanbul Byzantine Fault Tolerance.* https://github.com/ethereum/EIPs/issues/650

[3] Miguel Castro and Barbara Liskov. 1999. Pratical Byzantine Fault Tolence. *Processing of the Third Symposium on Operating Systems Design and Implementation* (February 1999).

[4] Stefano De Angelis, Leonardo Aniello, Federico Lombardi, Andrea Margheri, and V. Sassone. 2017. PBFT vs proof-of-authority: applying the CAP theorem to permissioned blockchain.

[5] Tien Tuan Anh Dinh, Ji Wang, Gang Chen, Rui Liu, Beng Chin Ooi, and Kian-Lee Tan. 2017. BLOCKBENCH: A Framework for Analyzing Private Blockchains. *CoRR* abs/1703.04057 (2017). arXiv:1703.04057 http://arxiv.org/abs/1703.04057

[6] Alan Fekete. 2018. *CAP Theorem.* Springer New York, New York, NY, 395–396. https://doi.org/10.1007/978-1-4614-8265-9_80644

[7] Luc Gerrits, Edouard Kilimou, Roland Kromes, Lionel Faure, and François Verdier. 2021. A Blockchain cloud architecture deployment for an industrial IoT use case. In *2021 IEEE International Conference on Omni-Layer Intelligent Systems (COINS).* 1–6. https://doi.org/10.1109/COINS51742.2021.9524264

[8] Luc Gerrits and Cyril Naves Samuel. 2021. Experimental Scalability Study of Consortium Blockchains with BFT Consensus for IoT Automotive Use Case. https://github.com/projet-SIM/acm-blocksys-2021

[9] Mohamed Tahar Hammi, Patrick Bellot, and Ahmed Serhrouchni. 2018. BC-Trust: A decentralized authentication blockchain-based mechanism. In *2018 IEEE Wireless Communications and Networking Conference (WCNC).* 1–6. https://doi.org/10.1109/WCNC.2018.8376948

[10] Akm Bahalul Haque, A. K. M. Najmul Islam, Sami Hyrynsalmi, Bilal Naqvi, and Kari Smolander. 2021. GDPR Compliant Blockchains–A Systematic Literature Review. *IEEE Access* 9 (2021), 50593–50606. https://doi.org/10.1109/ACCESS.2021.3069877

[11] Intel Corporation. [n.d.]. *Hyperledger Sawtooth.* https://sawtooth.hyperledger.org/docs/core/releases/latest/

[12] Henrique Moniz. 2020. The Istanbul BFT Consensus Algorithm. *CoRR* abs/2002.03613 (2020). arXiv:2002.03613 https://arxiv.org/abs/2002.03613

[13] Satoshi Nakamoto et al. 2008. Bitcoin: A peer-to-peer electronic cash system. (2008).

[14] ONISR. [n.d.]. *Bilan 2019 de la sécurité routière.* https://www.onisr.securite-routiere.gouv.fr/

[15] Suporn Pongnumkul, Chaiyaphum Siripanpornchana, and Suttipong Thajchayapong. 2017. Performance Analysis of Private Blockchain Platforms in Varying Workloads. In *2017 26th International Conference on Computer Communication and Networks (ICCCN).* 1–6. https://doi.org/10.1109/ICCCN.2017.8038517

[16] Péter Szilágyi. [n.d.]. *EIP-225: Clique proof-of-authority consensus protocol.* https://eips.ethereum.org/EIPS/eip-225

[17] Roberto Saltini. 2019. Correctness Analysis of IBFT. *CoRR* abs/1901.07160 (2019). arXiv:1901.07160 http://arxiv.org/abs/1901.07160

[18] Cyril Naves Samuel, Severine Glock, François Verdier, and Patricia Guitton-Ouhamou. 2021. Choice of Ethereum Clients for Private Blockchain: Assessment from Proof of Authority Perspective. In *IEEE International Conference on Blockchain and Cryptocurrency, ICBC 2021, Sydney, Australia, May 3-6, 2021.* IEEE, 1–5. https://doi.org/10.1109/ICBC51069.2021.9461085

[19] Markus Schäffer, Monika di Angelo, and Gernot Salzer. 2019. Performance and Scalability of Private Ethereum Blockchains. In *Business Process Management: Blockchain and Central and Eastern Europe Forum*, Claudio Di Ciccio, Renata Gabryelczyk, Luciano García-Bañuelos, Tomislav Hernaus, Rick Hull, Mojca Indihar Štemberger, Andrea Kő, and Mark Staples (Eds.). Springer International Publishing, Cham, 103–118.

[20] Wei Yao, Junyi Ye, Renita Murimi, and Guiling Wang. 2021. A Survey on Consortium Blockchain Consensus Mechanisms. arXiv:2102.12058 [cs.DS]