# A Byzantine Fault-Tolerant Ordering Service for the Hyperledger Fabric Blockchain Platform

(Short Paper)

Alysson Bessani
LaSIGE, Faculdade de Ciências,
Universidade de Lisboa
Portugal

João Sousa
LaSIGE, Faculdade de Ciências,
Universidade de Lisboa
Portugal

Marko Vukolić
IBM Research Zurich
Switzerland

## Abstract

We briefly describe the preliminary work on the design, implementation and evaluation of a Byzantine-fault tolerant ordering service for the Hyperledger Fabric Blockchain platform using the BFT-SMaRt replication library.

***CCS Concepts*** • **Information systems → Data replication tools**; • **Security and privacy → Security services**; • **Computer systems organization → Reliability**; *Dependable and fault-tolerant systems and networks*;

***Keywords*** Byzantine Fault Tolerance, Blockchain

**ACM Reference Format:**
Alysson Bessani, João Sousa, and Marko Vukolić. 2017. A Byzantine Fault-Tolerant Ordering Service for the Hyperledger Fabric Blockchain Platform: (Short Paper). In *SERIAL'17: ScalablE and Resilient InfrAstructures for distributed Ledgers, December 11–15, 2017, Las Vegas, NV, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3152824.3152830

The impressive growth of Bitcoin and other blockchain platforms based on the Proof-of-Work (PoW) technique, made evident the performance limitations of this approach. These limitations are mostly related with perfomance: existing systems are capable of processing 10s-100s transactions per second and present transaction confirmation latencies of up to one hour. Several alternative blockchain platforms proposed in the last years try to overcome these limitations by employing more traditional Byzantine Fault-Tolerant (BFT) consensus protocols for establishing consensus on the blocks in a blockchain.

*SERIAL'17, December 11–15, 2017, Las Vegas, NV, USA*
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-5173-7/17/12…$15.00
https://doi.org/10.1145/3152824.3152830

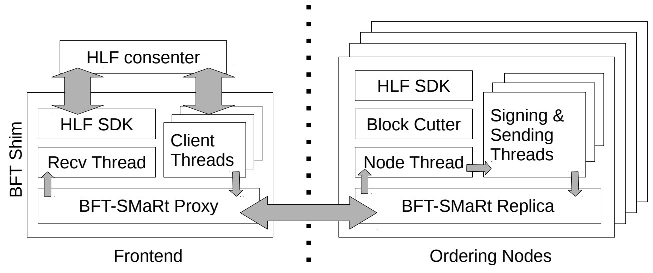


**Figure 1.** BFT-SMaRt ordering service architecture.

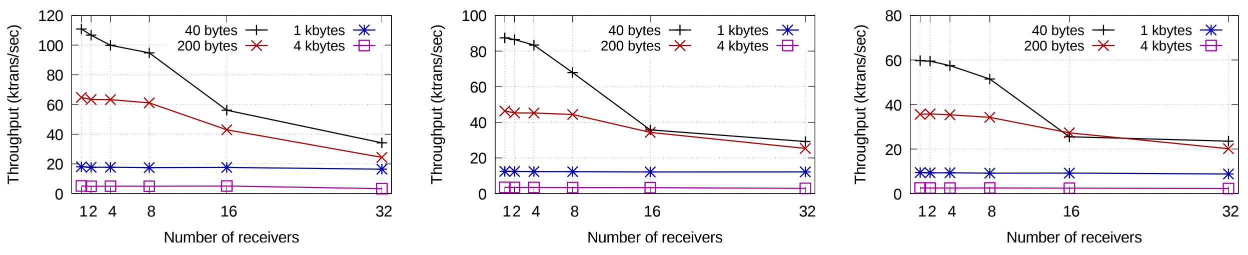
Hyperledger Fabric[1] (HLF) is one of these platforms, targeting business applications. It is built with flexibility and generality as key design concerns, supporting thus a wide variety of non-deterministic smart contracts (here called chaincode) and pluggable services [3]. The support for pluggable components, gives the HLF an unprecedented level of extensibility, and in particular the support for multiple ordering services for writing transactions on the blockchain. Despite of that, version 1.0 (launched in early 2017) comes without any Byzantine fault-tolerant (BFT) ordering service, supporting only crash tolerance through an ordering service based on Apache Kafka.[2]

We addressed this limitation by designing and implementing a BFT ordering service for HLF 1.0 based on the BFT-SMaRt state machine replication/consensus library [1]. The developed service is currently being integrated in the HLF distribution. More details about this ongoing work can be found in the full technical report available online [2].

## 1 BFT-SMaRt Ordering Service

HLF blockchain is implemented by nodes that can play many roles, such as clients, endorsers, validators and consenters. Clients submit transactions to the system, while endorsers and validators maintain the blockchain and implement the chaincode execution [3]. Consenters are the nodes that implement the consensus protocol. They are responsible for creating ordered blocks of transactions and disseminate them to registered validators.

[1]https://www.hyperledger.org/projects/fabric.
[2]https://kafka.apache.org/



**(a)** 4 orderers **(b)** 7 orderers **(c)** 10 orderers

**Figure 2.** BFT-SMaRt Ordering Service throughput for different transaction and cluster sizes.

The BFT-SMaRt ordering cluster is composed by a set of $3f + 1$ nodes, tolerating up to $f$ Byzantine faults, that collect transactions from other clients and execute BFT-SMaRt's replication protocol to totally order those transactions. Once a node gathers a predetermined number of transactions, it creates a new block containing these transactions and the hash of the previously created block, generates a digital signature on the block, and disseminates the block across all frontends, which collect $2f + 1$ matching blocks from ordering nodes.

The frontend is composed by the HLF consenter and a BFT shim (see Figure 1). The HLF consenter is implemented in Go and provides an interface for the HLF codebase to submit transaction. These transactions are relayed to the Java shim using sockets. This shim maintains a client thread pool that receive transactions from the consenter and relays them to the ordering cluster, and a receiver thread that collects blocks from the cluster. Transactions (resp. blocks) are sent to (resp. received from) the cluster through the proxy.

The ordering nodes are implemented on top of the BFT-SMaRt service replica, thus receiving a stream of totally ordered transactions. Each node maintains an object named *blockcutter*, where they store the transactions received from the service replica. Once the blockcutter holds a certain number of transactions (the block size), it creates the next block. The new block is associated with a header containing a sequence number and the hash of previous block, and then submitted to the signing thread pool. Notice that this thread pool does not cause non-determinism across the nodes because (1) the block header and transactions to be assigned to new blocks are generated sequentially within the node thread, and (2) the only structures each node needs to maintain as the application state is the block header from the previous iteration of the node thread. Once a block is signed, it is transmitted to all active frontends.

## 2 Preliminary Results

We conducted some preliminary experiments to evaluate the BFT-SMaRt ordering service by using clients that emulate the behaviour of multiple ordering service frontends. The environment is comprised by a cluster of Dell PowerEdge R410 nodes, running Linux, and connected through a Gigabit ethernet. We conducted experiments for different transaction sizes, each one representing: (1) a SHA-256 hash (40 bytes); (2) three ECDSA endorsement signatures (200 bytes); and (3) general transactions of 1 and 4 kbytes (the expected size of a small HLF transaction). Figure 2 shows throughput results for different ordering cluster sizes and number of receivers (registered frontends for endorsers/validators). Performance is constrained by the CPU load of block signing, the maximum throughput of the consensus algorithm, and the dissemination of the blocks after ordering.

The results show that, even though the throughput drops when increasing the number of receivers, the impact of the number of receivers is considerably smaller for larger (more representative) transactions (1k and 4 kbytes). This happens because in this workload, the overhead of the replication protocol is greater than the overhead of signing or transmitting blocks of 100 and 400 kbytes to a large number of receivers. For smaller transaction sizes, the transmission of blocks becomes the predominant overhead. Even transmitting blocks of 400 kbytes to 32 receivers in a cluster of 10 nodes, the ordering service still reaches a peak throughput of approximately 2200 transactions/second – which is more 2× of Ethereum's theoretical peak of 1000 transactions/second, and vastly superior to Bitcoin's 7 transaction/second.

***Acknowledgments.*** This work was supported by FCT through projects LaSIGE (UID/ CEC/00408/2013) and IRCoC (PTDC/EEI-SCR/6970/2014), by the European Commission through the H2020 SUPERCLOUD project (643964) and by an IBM Faculty Award.

## References

[1] Alysson Bessani, Joao Sousa, and Eduardo Alchieri. 2014. State Machine Replication for the Masses with BFT-SMART. In *Proceedings of the 44th IEEE/IFIP International Conference on Dependable Systems and Networks.*
[2] J. Sousa and A. Bessani. 2017. *A Byzantine Fault-Tolerant Ordering Service for the Hyperledger Fabric Blockchain Platform.* Technical Report arXiv:1709.06921. Cornell University Library.
[3] Marko Vukolić. 2017. Rethinking Permissioned Blockchains. In *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts.* Abu Dhabi, United Arab Emirates, 5. https://doi.org/10.1145/3055518.3055526