Efficient Fault-Tolerant Cluster-Sending*

Reliable and Efficient Communication between Byzantine Fault-Tolerant Clusters

Jelle Hellings
Department of Computing and Software
McMaster University
jhellings@mcmaster.ca

Mohammad Sadoghi
Exploratory Systems Lab
Department of Computer Science
University of California, Davis
msadoghi@ucdavis.edu

ABSTRACT

Traditional resilient systems operate on fully-replicated fault-tolerant clusters, which limits their scalability and performance. One way to make the step towards resilient high-performance systems that can deal with huge workloads, is by enabling independent fault-tolerant clusters to efficiently communicate and cooperate with each other, as this also enables the usage of high-performance techniques such as sharding and parallel processing.

To enable such efficient communication, we identify the *cluster-sending problem*: the problem of sending a message from one Byzantine cluster to another Byzantine cluster in a reliable manner, an essential communication primitive. We not only formalize this fundamental problem, but also establish lower bounds on the complexity of this problem under crash failures and Byzantine failures. Furthermore, we develop practical cluster-sending protocols that meet these lower bounds and, hence, have optimal complexity. Finally, we propose probabilistic cluster-sending techniques that only have an expected constant message complexity, this independent of the size of the clusters involved. Depending on the robustness of the clusters involved, these probabilistic techniques require only two-to-four message round-trips while supporting worst-case linear communication between clusters, which is optimal. As such, our work provides a strong foundation for the further development of resilient high-performance systems.

1 INTRODUCTION

The emergence of blockchain technology is fueling interest in the development of new data processing systems that can provide services continuously [14, 21, 23, 29, 30, 32], even during *Byzantine failures*, e.g., failures originating from network failure, hardware failure, software failure, or even coordinated malicious attacks. Recently, this has led to the development of several resilient data processing systems based on *permissioned blockchain technology* [2, 10, 11, 15, 25].

Unfortunately, the traditional fully-replicated *consensus-based* design of permissioned blockchain systems lacks the scalability required for modern data processing [28]. Consequently, recent data processing systems such as AHL [5], ByShard [18], Caper [2], Cerberus [17], Chainspace [1], ResilientDB [16], and SharPer [3] have proposed to combine sharding with consensus-based designs. Most of these systems follow a similar sharded design: the data is split up into individual pieces called *shards* and each shard is managed by different independent blockchain-driven clusters. To

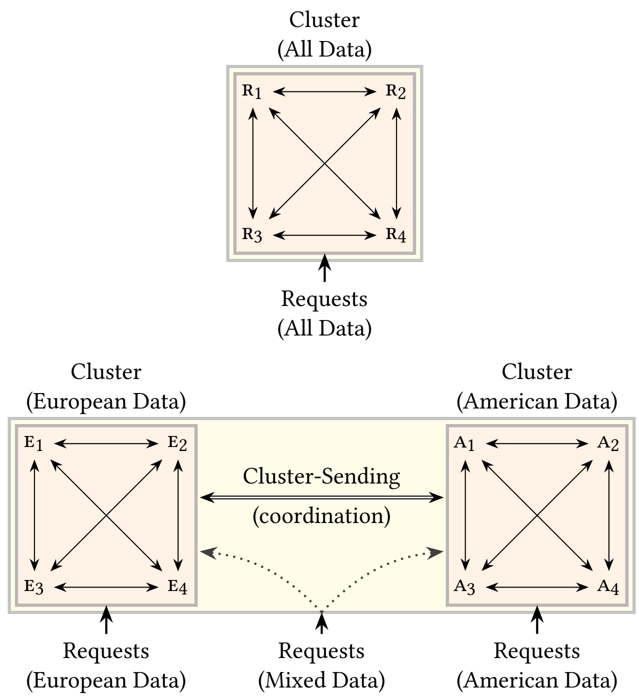


Figure 1: *Top*, a traditional fully-replicated resilient system in which all four replicas each hold all the data. *Bottom*, a *sharded* design in which each resilient cluster of four replicas holds only a part of the data.

illustrate the benefits of sharding, consider a system with a *sharded design* in which data is kept in *local Byzantine fault-tolerant clusters*, e.g., as sketched in Figure 1 by storing data relevant to American customers on systems located in the United States, whereas systems located in Europe contain data relevant to European customers. Compared to the traditional fully-replicated design of blockchain systems, this sharded design will improve *storage scalability* by distributing data storage and improve *performance scalability* by enabling concurrent transaction processing, e.g., transactions on American and European data can be performed independently of each other.

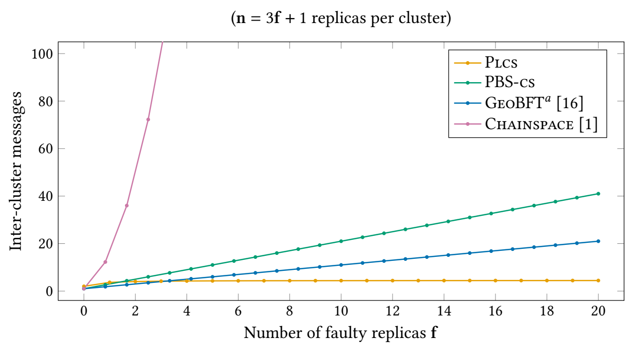
At the core of any sharded data processing system are two crucial primitives [26]. First, individual shards need primitives to independently make *decisions*, e.g., to execute transactions that only affect data held within that shard. In the setting where each shard is a fault-tolerant cluster, such per-shard decision making is formalized

*The technical reports on which this abstract is based are available at https://arxiv.org/abs/1908.01455 and https://arxiv.org/abs/2108.08541.

Figure 2: A comparison of *cluster-sending protocols* that send a value from cluster C_1 with $\mathbf{n}_{C_1}$ replicas, of which $\mathbf{f}_{C_1}$ are faulty, to cluster C_2 with $\mathbf{n}_{C_2}$ replicas, of which $\mathbf{f}_{C_2}$ are faulty. For each protocol P, *Protocol* specifies its name; *Robustness* specifies the conditions P puts on the clusters; *Message Steps* specifies the number of messages exchanges P performs; *Optimal* specifies whether P is worst-case optimal; and *Unreliable* specifies whether P can deal with unreliable communication.

	Protocol	Robustness[a]	Message Steps (expected-case)	Message Steps (worst-case)	Optimal	Unreliable
This Work	PBS-cs	$\min(\mathbf{n}_{C_1}, \mathbf{n}_{C_2}) > \mathbf{f}_{C_1} + \mathbf{f}_{C_2}$	$\mathbf{f}_{C_1} + \mathbf{f}_{C_2} + 1$		✔	✖
This Work	PBS-cs	$\mathbf{n}_{C_1} > 3\mathbf{f}_{C_1}, \mathbf{n}_{C_2} > 3\mathbf{f}_{C_2}$	$\max(\mathbf{n}_{C_1}, \mathbf{n}_{C_2})$		✔	✖
This Work	Plcs	$\min(\mathbf{n}_{C_1}, \mathbf{n}_{C_2}) > \mathbf{f}_{C_1} + \mathbf{f}_{C_2}$	4	$\mathbf{f}_{C_1} + \mathbf{f}_{C_2} + 1$	✔	✔
This Work	Plcs	$\min(\mathbf{n}_{C_1}, \mathbf{n}_{C_2}) > 2(\mathbf{f}_{C_1} + \mathbf{f}_{C_2})$	$2\frac{1}{4}$	$\mathbf{f}_{C_1} + \mathbf{f}_{C_2} + 1$	✔	✔
This Work	Plcs	$\mathbf{n}_{C_1} > 3\mathbf{f}_{C_1}, \mathbf{n}_{C_2} > 3\mathbf{f}_{C_2}$	3	$\max(\mathbf{n}_{C_1}, \mathbf{n}_{C_2})$	✔	✔
	GeoBFT [16]	$\mathbf{n}_{C_1} = \mathbf{n}_{C_2} > 3\max(\mathbf{f}_{C_1}, \mathbf{f}_{C_2})$	$\mathbf{f}_{C_2} + 1$[b]	$\Omega(\mathbf{f}_{C_1}\mathbf{n}_{C_2})$	✖	✔
	Chainspace [1]	$\mathbf{n}_{C_1} > 3\mathbf{f}_{C_1}, \mathbf{n}_{C_2} > 3\mathbf{f}_{C_2}$	$\mathbf{n}_{C_1}\mathbf{n}_{C_2}$		✖	✖

[a]Protocols that have different message step complexities depending on the robustness assumptions have been included for each of the robustness assumptions.
[b]Complexity when the coordinating primary in C_1 is non-faulty and communication is reliable.



[a]Complexity when the coordinating primary in C_1 is non-faulty and communication is reliable.

Figure 3: Comparison of the expected-case complexity of the probabilistic cluster-sending protocol Plcs, the worst-case optimal cluster-sending protocol PBS-cs, and the complexity of communication protocols proposed in related work.

by the well-known *consensus problem*, which can be solved by practical consensus protocols such as Pbft, even though it has high lower-bounds on its complexity [6–9, 12, 13, 22, 27]. Second, shards need primitives to *communicate* between each other, e.g., to coordinate the execution of transactions that affect data hold by multiple shards. Unfortunately, even though inter-shard communication is a *fundamental basic primitive*, it has not yet been studied in much detail: existing sharded blockchain-inspired data processing systems all use rather expensive system-specific techniques to enable coordination between shards (e.g., Chainspace [1] uses expensive multicasts and ResilientDB [16] uses an optimistic primitive with an expensive failure recovery path).

In this work, we improve on this situation by fully studying the problem of inter-shard communication in a permissioned fault-tolerant setting. In specific, we formalize the *cluster-sending problem*—the problem of sending a message from one fault-tolerant cluster to another fault-tolerant cluster in a reliable manner that is verifiable by all replicas involved—and fully study its complexity. Our contributions are as follows:

(1) We formalize the cluster-sending problem.
(2) We prove strict lower bounds that are only *linear* in the size of the clusters involved on the complexity of the cluster-sending problem in terms of the number of messages (when faulty replicas only crash) and in terms of the number of signatures (when faulty replicas can be malicious and messages are signed via public-key cryptography).
(3) We introduce *bijective sending* and *partitioned bijective sending*, powerful techniques to reliably perform cluster-sending with *optimal complexity* in practical environments (matching the established lower bounds).
(4) We introduce *probabilistic cluster-sending* to provide cluster-sending in *expected constant* steps and we show how to fine-tune probabilistic cluster-sending protocols to also match the established lower bounds.

A summary of our findings in comparison with existing techniques can be found in Table 2 and a visualisation of the complexity of the proposed cluster-sending protocols can be found in Figure 3.

Although *cluster-sending* can be solved using well-known permissioned techniques such as consensus, interactive consistency, Byzantine broadcasts, and message multicasting [1, 4, 6–9, 12, 13, 15, 22, 27], our lower-bound results shows that these primitives are not suitable as they are unnecessary costly. Likewise, our results show that existing approaches towards inter-shard communication (e.g., as used in Chainspace [1]) are unnecessary costly.

Our cluster-sending problem is closely related to cross-chain coordination in *permissionless blockchains* such as Bitcoin [24] and Ethereum [31], e.g., as provided via atomic swaps [19], atomic commitment [33], and cross-chain deals [20]. Unfortunately, such permissionless solutions are not fit for a permissioned environment.

From the results we provide, it is clear that cluster-sending is an independent problem with much lower complexity than traditional Byzantine primitives such as consensus. Hence, our work provides a novel direction for the design and implementation of high-performance sharded fault-tolerant data processing systems.

REFERENCES

[1] Mustafa Al-Bassam, Alberto Sonnino, Shehar Bano, Dave Hrycyszyn, and George Danezis. Chainspace: A sharded smart contracts platform, 2017. URL: http://arxiv.org/abs/1708.03778.

[2] Mohammad Javad Amiri, Divyakant Agrawal, and Amr El Abbadi. CAPER: A cross-application permissioned blockchain. *Proc. VLDB Endow.*, 12(11):1385–1398, 2019. doi:10.14778/3342263.3342275.

[3] Mohammad Javad Amiri, Divyakant Agrawal, and Amr El Abbadi. SharPer: Sharding permissioned blockchains over network clusters. In *Proceedings of the 2021 International Conference on Management of Data*, pages 76–88. ACM, 2021. doi:10.1145/3448016.3452807.

[4] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst.*, 20(4):398–461, 2002. doi:10.1145/571637.571640.

[5] Hung Dang, Tien Tuan Anh Dinh, Dumitrel Loghin, Ee-Chien Chang, Qian Lin, and Beng Chin Ooi. Towards scaling blockchain systems via sharding. In *Proceedings of the 2019 International Conference on Management of Data*, pages 123–140. ACM, 2019. doi:10.1145/3299869.3319889.

[6] Richard A. DeMillo, Nancy A. Lynch, and Michael J. Merritt. Cryptographic protocols. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pages 383–400. ACM, 1982. doi:10.1145/800070.802214.

[7] D. Dolev and H. R. Strong. Authenticated algorithms for byzantine agreement. *SIAM J. Comput.*, 12(4):656–666, 1983. doi:10.1137/0212045.

[8] Danny Dolev. The byzantine generals strike again. *J. Algorithm*, 3(1):14–30, 1982. doi:10.1016/0196-6774(82)90004-9.

[9] Danny Dolev and Rüdiger Reischuk. Bounds on information exchange for byzantine agreement. *J. ACM*, 32(1):191–204, 1985. doi:10.1145/2455.214112.

[10] Muhammad El-Hindi, Carsten Binnig, Arvind Arasu, Donald Kossmann, and Ravi Ramamurthy. BlockchainDB: A shared database on blockchains. *Proc. VLDB Endow.*, 12(11):1597–1609, 2019. doi:10.14778/3342263.3342636.

[11] Muhammad El-Hindi, Martin Heyden, Carsten Binnig, Ravi Ramamurthy, Arvind Arasu, and Donald Kossmann. BlockchainDB – towards a shared database on blockchains. In *Proceedings of the 2019 International Conference on Management of Data*, pages 1905–1908. ACM, 2019. doi:10.1145/3299869.3320237.

[12] Michael J. Fischer and Nancy A. Lynch. A lower bound for the time to assure interactive consistency. *Information Processing Letters*, 14(4):183–186, 1982. doi:10.1016/0020-0190(82)90033-3.

[13] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985. doi:10.1145/3149.214121.

[14] William J. Gordon and Christian Catalini. Blockchain technology for healthcare: Facilitating the transition to patient-driven interoperability. *Comput. Struct. Biotechnol. J.*, 16:224–230, 2018. doi:10.1016/j.csbj.2018.06.003.

[15] Suyash Gupta, Jelle Hellings, and Mohammad Sadoghi. *Fault-Tolerant Distributed Transactions on Blockchain*. Synthesis Lectures on Data Management. Morgan & Claypool, 2021. doi:10.2200/S01068ED1V01Y202012DTM065.

[16] Suyash Gupta, Sajjad Rahnama, Jelle Hellings, and Mohammad Sadoghi. ResilientDB: Global scale resilient blockchain fabric. *Proc. VLDB Endow.*, 13(6):868–883, 2020. doi:10.14778/3380750.3380757.

[17] Jelle Hellings, Daniel P. Hughes, Joshua Primero, and Mohammad Sadoghi. Cerberus: Minimalistic multi-shard byzantine-resilient transaction processing, 2020. URL: https://arxiv.org/abs/2008.04450.

[18] Jelle Hellings and Mohammad Sadoghi. Byshard: Sharding in a byzantine environment. *Proceedings of the VLDB Endowment*, 14(11):2230–2243, 2021. doi:10.14778/3476249.3476275.

[19] Maurice Herlihy. Atomic cross-chain swaps. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, pages 245–254. ACM, 2018. doi:10.1145/3212734.3212736.

[20] Maurice Herlihy, Barbara Liskov, and Liuba Shrira. Cross-chain deals and adversarial commerce. *Proc. VLDB Endow.*, 13(2):100–113, 2019. doi:10.14778/3364324.3364326.

[21] Maged N. Kamel Boulos, James T. Wilson, and Kevin A. Clauson. Geospatial blockchain: promises, challenges, and scenarios in health and healthcare. *Int. J. Health. Geogr*, 17(1):1211–1220, 2018. doi:10.1186/s12942-018-0144-x.

[22] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982. doi:10.1145/357172.357176.

[23] Laphou Lao, Zecheng Li, Songlin Hou, Bin Xiao, Songtao Guo, and Yuanyuan Yang. A survey of IoT applications in blockchain systems: Architecture, consensus, and traffic modeling. *ACM Comput. Surv.*, 53(1), 2020. doi:10.1145/3372136.

[24] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. URL: https://bitcoin.org/en/bitcoin-paper.

[25] Senthil Nathan, Chander Govindarajan, Adarsh Saraf, Manish Sethi, and Praveen Jayachandran. Blockchain meets database: Design and implementation of a blockchain relational database. *Proc. VLDB Endow.*, 12(11):1539–1552, 2019. doi:10.14778/3342263.3342632.

[26] M. Tamer Özsu and Patrick Valduriez. *Principles of Distributed Database Systems*. Springer, 2020. doi:10.1007/978-3-030-26253-2.

[27] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980. doi:10.1145/322186.322188.

[28] David Reinsel, John Gantz, and John Rydning. Data age 2025: The digitization of the world, from edge to core. Technical report, IDC, 2018. URL: https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf.

[29] Abderahman Rejeb, John G. Keogh, Suhaiza Zailani, Horst Treiblmaier, and Karim Rejeb. Blockchain technology in the food industry: A review of potentials, challenges and future research directions. *Logistics*, 4(4), 2020. doi:10.3390/logistics4040027.

[30] Horst Treiblmaier and Roman Beck, editors. *Business Transformation through Blockchain*. Springer, 2019. doi:10.1007/978-3-319-98911-2.

[31] Gavin Wood. Ethereum: a secure decentralised generalised transaction ledger. EIP-150 revision. URL: https://gavwood.com/paper.pdf.

[32] Mingli Wu, Kun Wang, Xiaoqin Cai, Song Guo, Minyi Guo, and Chunming Rong. A comprehensive survey of blockchain: From theory to IoT applications and beyond. *IEEE Internet Things J*, 6(5):8114–8154, 2019. doi:10.1109/JIOT.2019.2922538.

[33] Victor Zakhary, Divyakant Agrawal, and Amr El Abbadi. Atomic commitment across blockchains. *Proc. VLDB Endow.*, 13(9):1319–1331, 2020. doi:10.14778/3397230.3397231.