SOLVING THE PROBLEMS OF BYZANTINE GENERALS USING BLOCKCHAIN TECHNOLOGY

Alexander Bogdanov[1], Alexander Degtyarev[1,3], Nadezhda Shchegoleva[1], Vladimir Korkhov[1,3], Nodir Zaynalov[2], Jasur Kiyamov[1], Aleksandr Dik[1,a] and Anar Faradzhov[1]

[1] *Saint Petersburg State University, 7-9 Universitetskaya emb., Saint Petersburg, 199034, Russia*

[2] *Samarkand branch Tashkent university of information technology, Uzbekistan*

[3] *Plekhanov Russian University of Economics, 36 Stremyanny lane, Moscow, 117997, Russia*

E-mail: [a] st087383@student.spbu.ru

The process of digitalization of the Russian economy as the basis for the transition to the digital economy is conditioned by the requirements of objective reality and is based, first of all, on the introduction of digital technologies into the activities of its actors. The most promising is the Blockchain technology, which has the capabilities of the most effective coordination of the economic interests of the actors of the digital economy and is applicable in various spheres of economic activity. The article discusses the basics of cryptocurrencies and blockchain operation, as well as the technologies in which this "Tasks of Byzantine generals" (decision-making tasks) occurs. A comparison is made for solution of the problem with different blockchain technologies with several platforms that prevent behavior dangerous for transactions network and thereby increasing the competitiveness of the cryptocurrency.

Keywords: blockchain, distributed ledger technology, database

Alexander Bogdanov, Alexander Degtyarev, Nadezhda Shchegoleva, Vladimir Korkhov, Nodir Zaynalov, Jasur Kiyamov, Aleksandr Dik, Anar Faradzhov

1. Introduction

In recent years, e-commerce has played an important role in modern economic life as a new form of platform. It developed rapidly due to its characteristics of openness and efficiency [1,2]. The development of e-commerce not only contributes to the leap of the digital economy, but also realizes the modernization of consumer consumption, thereby bringing enormous economic benefits to society. However, with the constant growth of nodes, the supply chain becomes more complex, therefore, while maintaining the block chain, we face urgent problems [3,4]. In particular, in the event of errors in the transmission of information or disruptions in logistics in the supply chain due to asymmetry and opacity of information. Consequently, these phenomena bring losses to consumers and increase supply and inventory risks for suppliers. Currently, the blockchain implements a wide range of consensus mechanisms, among which we wanted Practical Byzantine Fault Tolerance (PBFT) - one of the most effective ways to fight. PBFT reduces the complexity of reaching agreement between different nodes in a distributed system using a two-phase protocol. However, the consensus effectiveness of PBFT can hardly be guaranteed when working with the growing number of partners in the supply chain.

2. How does PBFT ensure consensus consistency?

In a private blockchain network, all the participants are whitelisted and bounded by strict contractual obligations to behave “correctly”, and hence more efficient consensus protocols such as Byzantine Fault Tolerance (BFT) can be used. BFT works on the assumption that less than one-third of the peers are faulty (f), which means that the network should consist of at least $n = 3f + 1$ peers to tolerate f faulty peers [6]. Thus $f = [(n - 1)/3]$. The network requires $2f + 1$ peers to agree on the block of transactions.

The PBFT version included only two phases: PrepareRequest and PrepareResponse. This was a necessary simplification due to the high communication costs and inherent efficiency issues faced by the first prototype. The removal of the third phase made it possible to reach consensus much faster, which is especially necessary in a P2P network with the possibility of high delays. In practice, this demonstrated how strong the opponent is, according to (A.2). We start by demonstrating three “good cases” for PBFT: (a) no replica is faulty (Fig. 1); (b) one replica is faulty, but not the primary one (Fig. 2) (c) the primary one is faulty, review of changes is required (Fig. 3).

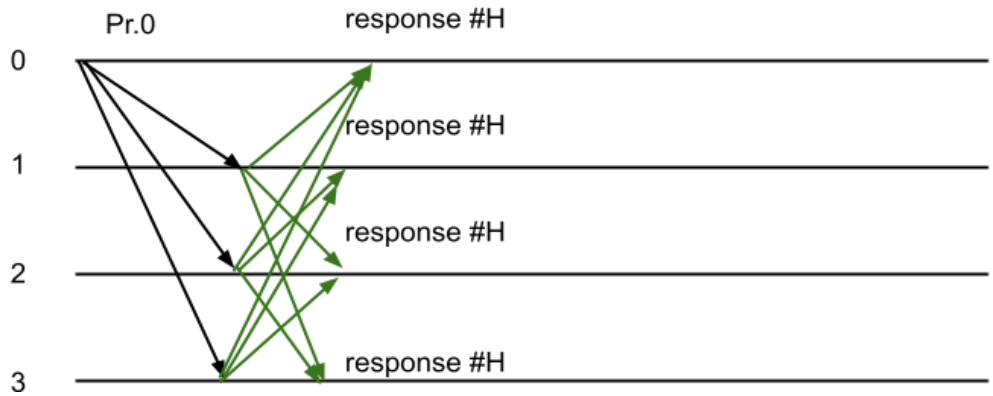

Figure 1. Ideal case. The request was sent and received by the primary (replica 0). Everybody passes block # H.

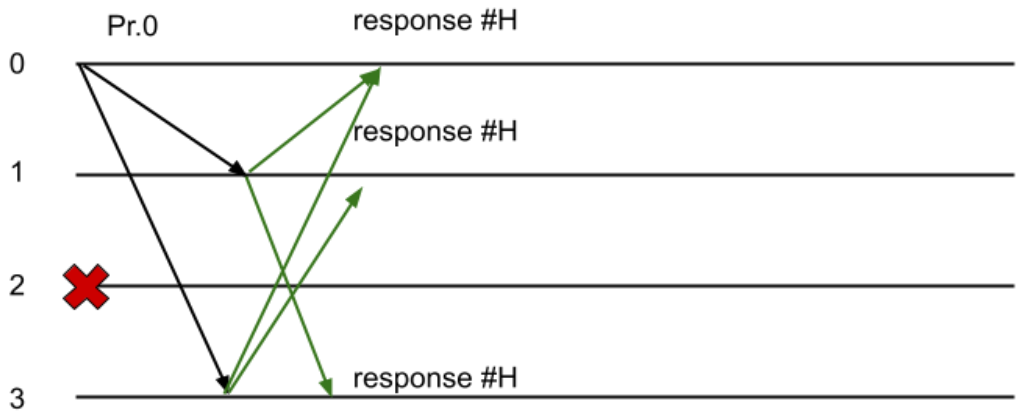

Figure 2. Good case. Request sent and received by Primary. Everything except the 2nd response block #H.

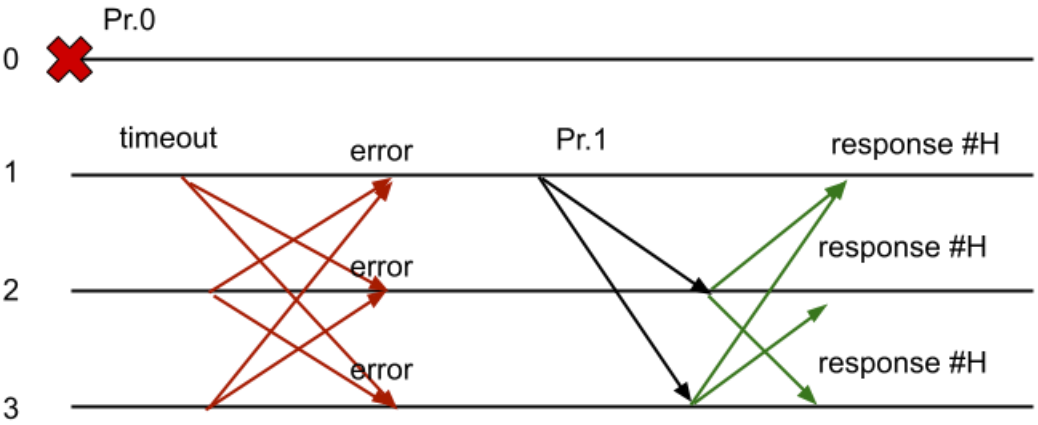

Figure 3**.** Primary 0 is dead and no offers have been received. Replicas will change appearance after a timeout (red lines) due to the fact that they did not receive each other's responses in time. Replica 1 becomes the new primary (Pr.1) and successfully generates block #H.

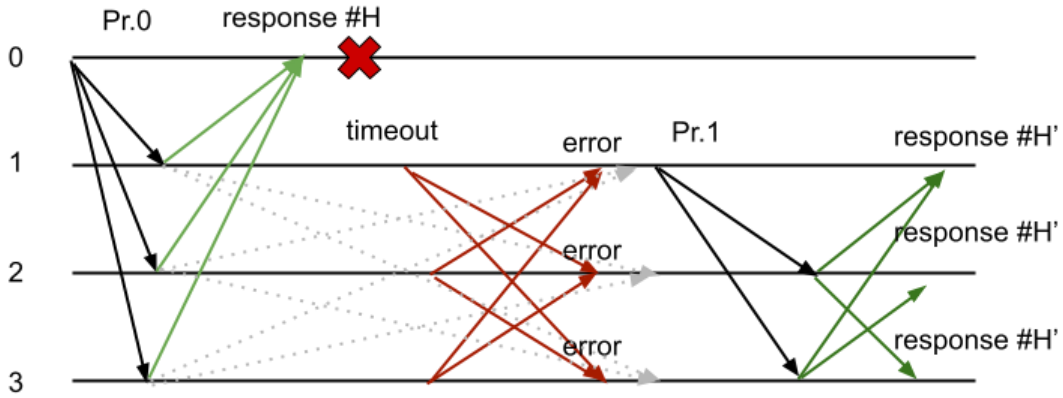

Figure 4. Request sent and received with response 0 (parameter 0 means primary view 0). The rest will not follow him, as they will change appearance (red lines) due to the fact that they do not receive each other's answers (green lines) in time. Answer 1 becomes the new primary (parameter 1) and generates "spork" (blocks #H and # H0 have the same height, but different content). The X indicates a complete node failure. Ignored messages are dashed lines (received after a view change has already occurred).

The reason for this was simple: some primary node could end up offering a valid block, receiving support from a sufficient number of nodes (at least 2f + 1), but these same nodes could only receive messages from each other after a long time. This will cause the view to change as the given timeout has expired for that view (at least for the 'M' replicas), so the other backup will become the

primary and offer another valid block for the same height, again getting enough signatures from their peers (at least 2f + 1). This situation is shown in figures 4 and 5

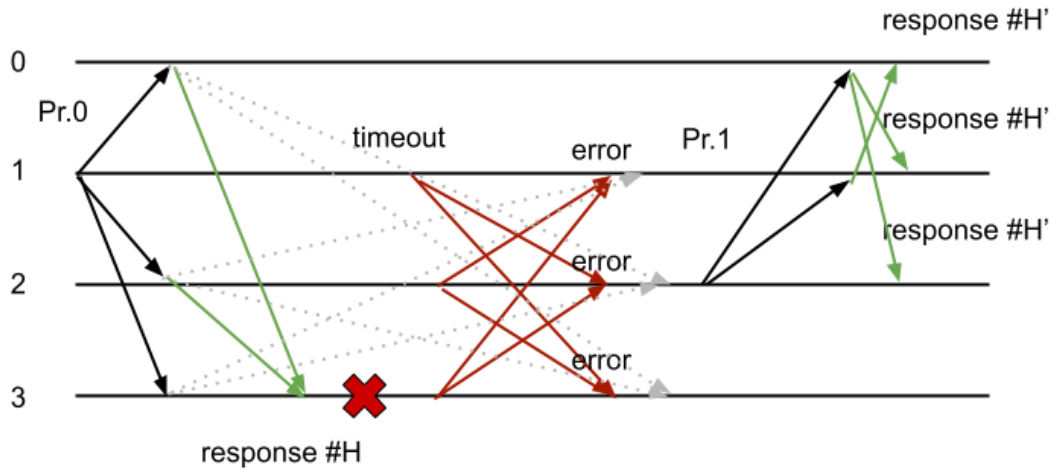

Figure 5. Request sent by Replicas 1 and 2 f PrepareResponse to Replica 3. The primary and others will not follow it because they will change views because they will not receive each other's responses. Replica 2 becomes the new primary and generates "spork" (blocks #H and #H 0).

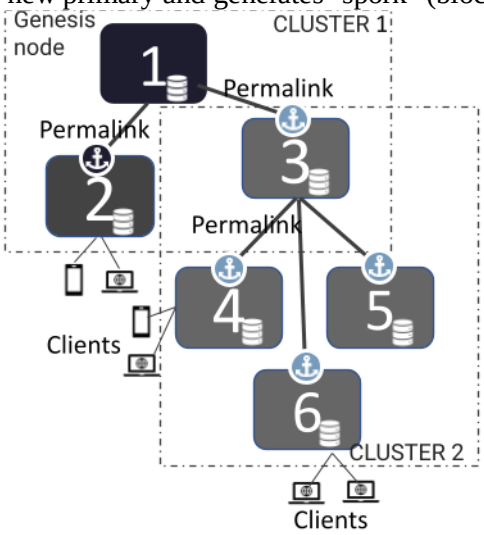

Figure 6. Genesis node for PBFT

The size of the cluster, the joining of nodes to it is determined by the topology processor, which is a separate family of transactions.

Within each cluster, a variable leader is determined, which changes after several rounds of voting [5]. If he does not respond to the leader's request within a certain time, the procedure for choosing a new leader is performed Voting is initially held in the cluster, then an arbitrator outside the cluster is selected using a special algorithm.

PFT prevents double-spending attacks due to intra-cluster voting time differences and DAG timing (state) timing, which is done through permalinks.

3. Platform Comparisons by PBFT Consensus

The solution to the problem was obvious: BFT required a third phase. It was clear that block signatures could not be represented on the network before the replica was sure that other nodes would follow, otherwise several valid blocks could be generated at each height (as shown in Figures 4-5).

This issue stems from an adversarial condition where Byzantine agents can aggregate signatures (#H) σ i from 2f + 1 replicas (sent in response to preparation), which allows a valid block #H to be created even if all 2f + 1 replicas i are finally resolved retransmit #H 0 after view change. An interesting feature of this problem from a practical/applied point of view is that it only happened in the end (several times a year), even in a blockchain with systematic block generation (15s on average). Given that all replicas are healthy, only a very "bad" combination of latency can lead to this scenario. From a theoretical point of view, the problem is easy to reproduce.

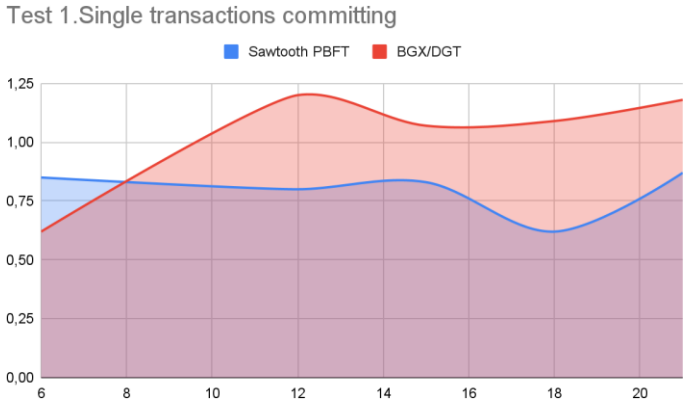

Figure 7. Comparison of Initial BGX / DGT Traffic to Sawtooth PBFT for upgrading throughput

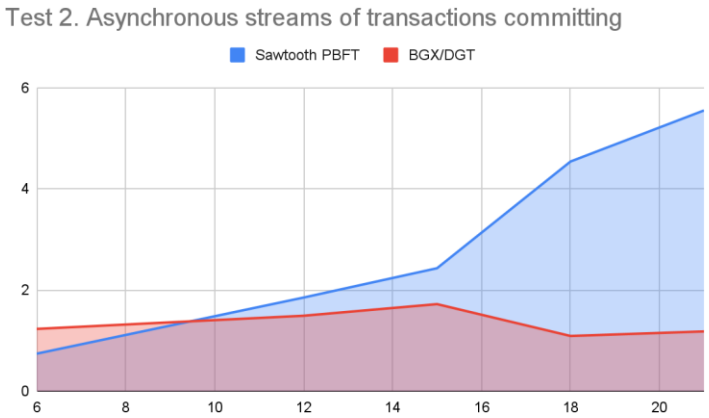

Figure 8. Comparisons of BGX / DGT Asynchronous Transaction Flows with Sawtooth PBFT for upgrading throughput

Note that the time it takes to send a block commit event from the ID is considered negligible for the subscriber since they are both on the same virtual machine and as such there is no network latency. The transaction scheduler can be either sequential or parallel. When using sequential execution of a transaction, it is blocked until the execution of the previous transaction is completed; with parallel scheduling of transactions, the subsequent transaction that does not have dependencies on the current transaction can be executed in parallel with the current transaction sent for execution.

4. Conclusion

The research purposed in this paper is to optimise the performance of the blockchain PBFT consensus mechanism applied in the supply chain scenario. Therefore, to achieve the improvement of the system's delay, throughput and other performance with less time consumption. The research work in this area has great scientific research significance and engineering application value. This article mainly starts from the current research status of PBFT consensus mechanism, introduces the current

research results of PBFT consensus mechanism. It aims at its current weak link in the blockchain system.

We made a series of tests to eliminate errors and our recommendations were passed on to the DGT developers to add a new version of the platform. The next steps are to repeat the experiments for different consensus algorithms [3] and for different workloads to evaluate the system performance for each of the two transaction processors as a whole.

References

[1] S. Pongnumkul, C. Siripanpornchana, and S. Thajchayapong, “Performance analysis of private blockchain platforms in varying workloads,” in 2017 26th International Conference on Computer Communication and Networks (ICCCN). IEEE, 2017, pp. 1–6.

[2] Y. Hao, Y. Li, X. Dong, L. Fang, and P. Chen, “Performance analysis of consensus algorithm in private blockchain,” in 2018 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2018, pp. 280–285.

[3] M. Castro, B. Liskov et al., “Practical byzantine fault tolerance,” in OSDI, vol. 99, 1999, pp. 173–186.

[4] A. B. Ayed A conceptual secure blockchain-based elec- tronic voting system // International Journal of Network Security and Its Applications. 2017.

[5] H. Zhou, X. Ouyang, Z. Ren, J. Su, C. de Laat, and Z. Zhao A blockchain based witness model for trust-worthy cloud service level agreement enforcement // IEEE INFOCOM 2019-IEEE Conference on Computer Communications. 2019.

[6] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance and Proactive Recovery", ACM Trans. Comput. Syst., vol. 20, no. 4, pp. 398-461, Nov. 2002.