BOSTON COLLEGE

HONORS THESIS

Byzantine Concensus: Theory and Applications in a Dynamic System

Author:
Yifan Zhang

Supervisor:
Lewis Tseng

May 19, 2021

BOSTON COLLEGE

Abstract

Department of Computer Science

Byzantine Concensus: Theory and Applications in a Dynamic System

by Yifan Zhang

This survey paper aims to study the theory and applications of Byzantine general problems. In particular, we compare models and methods being used to study Byzantine broadcast problems in dynamic systems, where nodes may join and leave at any time. Byzantine consensus is one of the most fundamental problems in distributed algorithms. There exist two main variants of Byzantine consensus problems: Byzantine Broadcast (BB) and Byzantine Agreement (BA). Byzantine broadcast problems further have several variants including Byzantine reliable broadcast (BRB) and Byzantine consistent broadcast (BCB). Byzantine consistent broadcast has relaxed conditions compared to the original Byzantine broadcast protocol, and has significant applications in blockchain systems. This paper describes the history of research on Byzantine consensus problems and their applications in a dynamic system, and helps readers understand the common techniques being used in these works.

Acknowledgements

Special thanks to professor Lewis Tseng for inspiring me in choosing this particular topic, believing in me, leading me to through my thesis and answering my questions.

Contents

Chapter 1

Introduction

Consensus is the task of getting all nodes in a group to agree on some specific value based on the votes of each processes. All nodes must agree upon the same value and it must be a value that was submitted by at least one of the nodes. A fundamental problem in distributed computing and multi-agent systems is to achieve overall system stability, where all nodes agree on some value needed for computation, in the presence of a number of faulty processes. This often requires coordinating processes to reach consensus. Byzantine consensus is the problem for n nodes to agree on a value, despite the fact that up to f of them may behave arbitrarily, which is called Byzantine failures. Since Byzantine failures imply no restrictions they can confuse the failure detection systems, which makes fault tolerance difficult. Prior works have done extensive study on Byzantine consensus problems under various models, such as the asynchronous system [22][7][24] and the synchronous [5] system. Besides faulty nodes model, there are studies on reaching consensus under faulty links model [22][25] as well.

There exist a few variants formulations for the Byzantine consensus problem, including Byzantine Broadcast(BB) and Byzantine Agreement(BA) problems. In Byzantine agreement, each party has an input value, and all parties try to decide on the same value. Lamport et al. [12] showed that more than 2/3 of the participating parties must be honest to reach consensus. Since then, Byzantine agreement has been studied under both synchronous and asynchronous settings and in deterministic [9] and randomized [2][20] models.

Chapter 2

Preliminaries

We look at some commonly used techniques to study Byantine consensus problems, especially BCB, in dynamic system. Studies in broadcast problems generally assume digital signatures and public-key infrastructure (PKI), and use $\langle x \rangle_r$ to denote a message x signed by party r rounds. Under synchronous setting, if an honest party sends a message at the beginning of some round, an honest recipient receives the message at the end of that round. To be more specific, a round consists of three phases (i) nodes send messages in the current round, (ii) nodes receives messages sent at the beginning of the current round and (iii) process the received messages and update local state. There are a set of n_t nodes in our model, and n_t is the number of nodes in the system at time t. As we have mentioned earlier, nodes can join and leave the system anytime in a dynamic system. A node that is not faulty throughout the execution is said to be honest and faithfully executed in the protocol. It is only assumed that each node knows who's in the system but it does not know who is faulty. We use the term quorum to mean the minimum number of all honest parties in round t, i.e., $n_t -$

Chapter 3

Byzantine broadcast problems

As mentioned earlier, a designated sender denoted by r_s has an input v_{in} to broadcast to all parties in broadcast. The broadcast problem further has several variants. The consistency and validity conditions of BA, BB and BRB are the same as the conditions for BCB protocol, and the only difference is termination condition. The Byzantine reliable broadcast [4] has a totality condition that is more relaxed than the termination condition for Byzantine Broadcast. As shown in the definitions, BCB has more relaxed requirements compared to the original byzantine broadcast protocol and Byzantine reliable broadcast because it allows some parties to decide while others do not.

Definition 1.

and (iii) totality: if an honest party decides a value, then every honest party decides a value.

Now we compare different methods from studies on Byzantine broadcast problems. As mentioned earlier, there are two main variants of Byzantine consensus:

(SMR), the replicas engage in the consensus protocol but need to convince clients of the results. For example, if there are f Byzantine replicas, the client needs to communicate with at least $f+1$ replicas to know that it has communicated with at least one honest replica. While Bratcha's reliable broadcast protocol does not use signature schemes, all replicas in SMR know the others' public keys to verify signatures.

There will be a primary node responsible for ordering the requests from the clients. However, if the primary node does not work, view changes will be carried out to elect a new primary node. Similar to Bracha's reliable broadcast, this model also adopts a three-phase protocol: pre-prepare, prepare and commit. The primary node multicasts the sequence of proposed requests from the client to other nodes, like what the sender does in broadcast protocol. Upon receiving a pre-prepare message, nodes enter the prepare phase to echo the message. Like Bracha's reliable broadcast model, Castro and Liskov's model used a quorum mechanism (which is explained in the previous chapter). Once collecting a quorum of $2f$ prepare messages, nodes will commit that message. Then replicas execute the message and send a reply to the client. Lastly, the client waits for $f+1$ replies from different replicas with same result and valid signatures. This is the result of the operation.

Similarly, Momose and Ren [18] desgined an algorithm in BCB protocol where everyone interacts with the designated sender only. The protocol proceeds in synchronous rounds. There are four phases in the algorithm: Propose, Echo, Vote and Forward. Although BCB protocol has more relaxed conditions compared to Byzantine broadcast, BCB has important applications as well. Momose and Ren studied how BCB protocol can be used to close the considerable gaps in the communication complexity of Byzantine consensus. The linear communication complexity when $f < n/2$ can be achieved using an expander graph. An expander graph has a constant number of edges per vertex, which ensures constant communication per party and good connectivity to detect inconsistent values effectively.

its overlay neighbor and from the node from which it received the gossip. This greatly reduces the number of messages generated by the protocol since nodes do not need to be sent $f + 1$ times like they do in previous protocols.

In reality, the implementation and specification of Byzantine broadcast protocols can vary based on the environments. In particular, asynchronous reliable broadcast is widely used as building blocks for other distributed computations in reality, such as secure distributed storage. A client starts the dispersal protocol as it decides to store a file in a distributed storage system provided by n servers. The file is split into n different blocks, each one being stored by one of the n servers. Cachin and Tessaro [6] adapted asynchronous reliable broadcast to implement a simple asynchronous verifiable information dispersal scheme. A key concept is the gateway, which is an non-faulty party through which clients access the servers comprising the storage system. The idea is to replace the gateway by an asynchronous reliable broadcast protocol such as [3] so that the scheme is robust against corrupted clients, and then the server can keep its own block in memory together with the list of hashes.

Chapter 4

Dynamic Systems

Byzantine consensus problems have been extensively researched in dynamic networks. However, many previous works assume that all nodes are fault-free , but the network is controlled by a message adversary. Kuhn et al. [8] showed that eventual consensus is hard in the absence of a good initial upper bound on the size of the network. Fugger et al. [15] proved tight lower bounds on the contraction rates of asymptotic consensus algorithms in dynamic networks. More realistic settings consider node failures. Augustine et al. [10] studied Byzantine agreement in dynamic networks and proposed randomized distributed algorithms that achieve almost-everywhere Byzantine agreement. They assumed that the total number of nodes in the network remains constant while both nodes and edges in the expander graph can change arbitrarily.

Instead of changing nodes and edges in an expander graph, Tseng [14] studied eventual consensus and presented a simple algorithm in both static and dynamic systems where nodes leave and join the system. The communication channel is assumed to be fair-loss. The message will be eventually delivered only if both ends are fault-free. The eventual property allows us to solve the problem in asynchronous systems with crash or even Byzantine failures. Tseng's algorithm in a dynamic system introduced a History variable to store previous values. Tseng's work also showed the importance to clarify the notion of "nodes currently in the system" and n_t in a dynamic system. Nodes that do not execute **Join** function properly at the beginning of round t are not considered to be in the system in round t. Similarly, nodes that do not properly execute **Leave** function at the beginning of round t, including crashed nodes, are still in the system but are faulty nodes in round t. Besides, nodes that do not leave properly, including Byzantine nodes, are considered faulty nodes.

A dynamic system is desirable when it may not be possible or convenient to stop the entire system to allow modification to part of its hardware or software in a large distributed system. Hence, Hao et al. [28] proposed a Dynamic PBFT based on the first practical Byzantine-fault-tolerant protocol (PBFT), so that users could add or take out any node without stopping the whole system. They assume that each node has three states: Benign, Absent and Malicious . The state of a new node should be initialized to Benign. If a node exit actively, the state will change to Absent. There is a primary node that is similar to the sender in a broadcast protocol. The primary node can change in a dynamic PBFT model. The **join** function is a three-phase protocol message: (1) new node j registers at the primary node; (2) new node j multicasts a request message to all replicas; (3) nodes verify the signature upon receiving the request. Node **Exit** has two parts: active exit and passive

Changes have to be validated so that they are compatible with the existing system. Based on the dynamic configuration, we can determine the properties required by languages and their environments to support the dynamic configuration.

The following table gives an overview on models that we have discussed so far:

Model	Assumptions	Resilience	Model	Key techniques
BA [13]	No cryptography: Digital signatures:	$f < n/3$ $f < n/2$	synchronous	Quorum mechanism
BB [12]	No cryptography: Digital signatures:	$f < n/3$ $f < n$		

f

Chapter 5

Bitcoin/Blockchain

Many practical peer-to-peer systems such as Bitcoin is a dynamic system. Nakamoto [19] introduces a peer-to-peer version of eletronic cash that allows online payments to be sent directly from one party to another. This is possible because messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will. Due to the success of Bitcoin, blockchain technologies are taking the world by storm. All nodes in the blockchain have equal status. Theses nodes achieve consensus by using the prior agreement of the rules and following the principle of majority dominance.

The blockchain technology is built on top of four fundamental building blocks, and each block has key properties achieved through specific mechanism: (1) Identifying the source and destination of a transaction: Users serve from digital identities called "address" to send and receive transactions. (2) Transactions: Transactions are generated by the sender and broadcasted to the network of peers. Transactions are invalid unless they have been recorded in the public history of transactions, the blockchain; (3) Condition for auto-processing a transaction: The transfer of any value with the blockchian or the execution of any function through the blockchain should be locked by a logic conditions; (4) Consensus: updates must be agreed by all parties. Outchakoucht et al. [1] claimed that blockchain combined with IoT (Internet of Things) is of great importance for blockchain in the future. He proposed a dynamic and fully distributed security policy based on blockchain. To make the system more secure, he adopted an authorization process where new nodes register, request access to nodes in the blockchain.

rules on join/leave protocols in a dynamic system, PoW lets new nodes join easily but makes it difficult for malicious nodes to attack the system. Hence, the PoW

Chapter 6

Conclusion

In conclusion, Byzantine consensus problems are challenging to study but they have important applications such as Bitcoin. Most early works focused on study-

Bibliography

[1] H. Es-samaali A. Outchakoucht and J.P. Leroy. In: International Journal of Advanced Computer Science and Applications, Vol. 8, No.7, 2017 ().

[2] Michael Ben-Or. “Another advantage of free choice (extended abstract): Completely asynchronous agreement protocols”. In: In Proceedings of the second annual ACM symposium on Principles of distributed computing, pages 27–30 (1983).

[3] G. Bracha. “An asynchronous [(n – 1)/3]-resilient consensus protocol”. In: Proc. 3rd ACM Symposium on Principles of Distributed Computing (PODC) (1984).

[4] G. Bracha and S. Toueg. “Asynchronous consensus and broadcast protocols”. In: Journal of the ACM (1985). DOI: https://doi.org/10.1145/4221.214134.

[5] M. Bravo, G.Chockler, and A. Gotsman. “Making Byzantine Consensus Live”. In: 34th International Symposium on Distributed Computing (DISC 2020) (2020). DOI:

[8] Y. Moses F. Kuhn and R. Oshman. “Coordinated consensus in dynamic networks”. In: Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing (2011).

[9] M. Fischer and N. Lynch. “A lower bound for the time to assure interactive consistency”. In: Information processing letters, 14(4):183–186 (1982).

[10] G. Pandurangan J. Augustine and P. Robinson. “Fast Byzantine agreement in dynamic networks”. In: Proceedings of the 2013 ACM symposium on Principles of distributed computing (2013).

[11] J. Kramer and J. Magee. “Dynamic Configuration for Distributed Systems”. In: IEEE Transactions on Software Engineering (1985).

[12] R. Shostak L. Lamport and M. Pease. In: Concurrency: the Works of Leslie Lamport (2019).

[13] Robert Shostak Leslie Lamport and Marshall Pease. “The Byzantine Generals Problem”. In: (1982).

[14] L.Tseng. “Eventual Consensus: Applications to Storage and Blockchain”. In: 2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton) (2019), pp. 840–846. DOI: 10.1109/ALLERTON.2019.8919675.

[15] T. Nowak M. Fugger and M. Schwarz. “Tight Bounds for Asymptotic and Approximate Consensus”. In: Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing (2018).

[16] B. Liskov M.Castro. “Practical Byzantine Fault Tolerance”. In: OSDI (1999).

[17] M.S. Paterson M.J. Fischer N.A. Lynch. “Impossibility of distributed consensus with one faulty process”. In: Journal of the ACM (1985).

[18] A. Momose and L. Ren. “Optimal Communication Complexity of Byzantine Consensus under Honest Majority”. In: (2020).

[19] Satoshi Nakamoto. “Bitcoin: A Peer-to-Peer Electronic Cash system”. In: bitcoin.org (2008).

[20] Michael O. Rabin. “Randomized byzantine generals”. In: In Proceedings of the 24th Annual Symposium on Foundations of Computer Science, pages 403–409 (1983).

[21] M. Reiter. “Secure agreement protocols: Reliable agreement in the presence of faults”. In: Proc. 2nd ACM Conference on Computer and Communications Security (1994).

[22] N. Santoro and P. Widmayer. “Agreement in synchronous networks with uniqitous faults”. In: Theor. Comput. Sci (2007).

[23] S.King and S. Nadal. “PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake”. In: (2012).

[24] M. Larrea T. Crain V. Gramoli and M. Raynal. “DBFT: Efficient Leaderless Byzantine Consensus and its Application to Blockchains”. In: 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA) (2018).

[25] B. Weiss U. Schmid and I.Keidar. “Impossibility results and lower bounds for consensus under link failures”. In: SIAM J.Comput (2009).

[26] M. Segal V. Drabkin R. Friedman. “Efficient Byzantine broadcast in wireless ad-hoc networks”. In: 2005 International Conference on Dependable Systems and Networks (DSN'05) (2005).

[27] M. Segal V. Drabkin R. Friedman. “Efficient Byzantine Broadcast in Wireless ad-hoc Networks”. In: 2005 International Conference on Dependable Systems and Networks (DSN'05) (2005).

[28] L. Zhiqiang L. Zhen X. Hao L. Yu and G. Dawu. “Dynamic Practical Byzantine Fault tolerance”. In: 2018 IEEE Conference on Communications and Network Security (CNS) (2018).