IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 8, Issue 3, June - July, 2020
ISSN: 2320 – 8791 (Impact Factor: 2.317)
www.ijreat.org

ANALYSIS OF PBFT PROTOCOLS ALONG WITH THE FAULTS IN BLOCKCHAIN

Ms. T.G.R. AbiramieShree[1], Dr. B. Kavitha[2], Mr.M. Thangavel[3]
1Postgraduate student, Information technology, Thiagarajar College of Engineering, Madurai, India
2Lecturer, Electronics and Communication Engineering, IRT Polytechnic College, Chromepet, Chennai, India.
3Assistant Professor, Information technology, Thiagarajar College of Engineering, Madurai, India.

Abstract

Blockchain is a decentralized peer to peer network where they exhibit transparency and trust while handling the digital ledger along with immutable security. This distributed environment is successfully driven by consensus they made while proposing the blocks of data into the network. These blocks are a collection of unconfirmed transactions which is later added into the blockchain. To achieve this, the consensus protocol should be able to withstand the faults which are common in a distributed environment. These faults range from a simple node failure fault to a complex Byzantine fault Tolerance (BFT). BFT is nothing but the nodes which act as a traitor and behave in a rogue way which makes the blockchain environment risky to achieve the consensus while proposing the blocks. It deals with one of the consensus protocols called the Practical Byzantine Fault Tolerance (PBFT). PBFT protocol helps to tolerate the faults that happen in the distributed environment even if 1/3 of the faulty nodes are present in it. Many protocols and projects which make use of PBFT but one of them is Hyperledger Fabric. It is a private permissioned blockchain and the aim is to achieve the consensus using PBFT with the help of maximum of 2/3 nodes in the network even if it contains 1/3 faulty nodes. This work aims to study the research works addressing the behavior of PBFT protocols in the presence of multiple faults in blockchain.

Keywords — *Blockchain, Fault tolerance, tendermint, hyperledger, data block, security, protocols, PBFT*

I. INTRODUCTION

Security is a major concern where each individual is accountable for the huge generation of data. These data are previously protected and recorded with conventional cryptographic methods and algorithms that provide confidentiality, Integrity, and Availability to the data. In today's world, there is a high chance of cyber-attacks and advanced hacking techniques which may cause severe damage to our data and lead to data loss. Recently, the Hungarian Government Organization faced a major cyber-attack on its digital database which contains their official documentation, contracts, and invoices. To address these problems and to enforce security in an untrusted environment there comes the Blockchain

1.1 BLOCKCHAIN

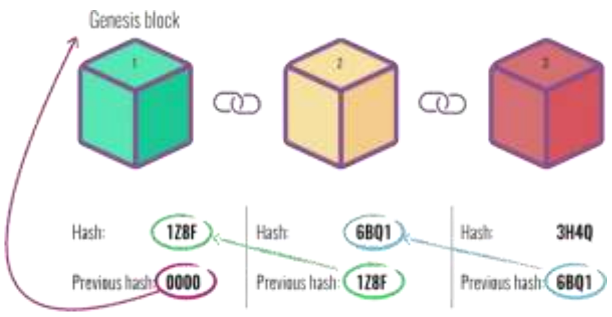

Figure1.1: Working of blockchain
(https://crypto9.co/e-dinar-coin-blockchain-technology)

Blockchain technology is one of the emerging fields in the technical world. Here, the simple definition of blockchain Figure1.1 is that the data or a transaction happens in the form of blocks where the block is chained together with the help of the hash values between them. It provides these types of services with the help of hash values of the data. The hash values are the links between the blocks. The genesis block (Origin Block) is linked to the next block where the hash value of the next block is carried by the genesis block. The blocks of data can be verified by any number of users who are all called peers where they can leave or join the environment at any time. These people are called data miners where they verify the data in exchange for bitcoins or cryptocurrencies from the blockchain of security. This is only possible in the blockchain environment. In this environment, there lies transparency in the blockchain environment but along with a high degree of the transaction is verified by a large number of peers and after the successful verification of the block they tend to be added into the blockchain. Here, once the block is verified and added to the chain there is very little chance of changing or removing a block from the blockchain. There lie three characters that support the blockchain environment.

1.1.1 DECENTRALIZATION

The service that users are using day to day is all centralized services. The centralization service is nothing but access to service from a single service provider and depends only on them for all types of services. If the service provided by them is stopped or else, they provide false services to the users there arises a loss of data or loss of service from the reliable source. In the blockchain, there is no central authority who governs and provides service to the users. The transactions happen between the intended users automatically with the help of the blocks and get verified and finally added to the chain.

1.1.2 TRANSPARENCY

The transparency in the sense does not mean that the users and the personal data are transparent to all the users in the blockchain environment. Here, the user's identity is completely masked and the representation of each user is a random generated alphanumeric value. The transaction between them can only be visible to the miners who try to verify the block. This level of security is only available in the blockchain and is not available in the earlier days of the security environment. Most of the big companies use this type of transaction in their environment where they both attain security as well as transparency over the data.

1.1.3 IMMUTABILITY

Once a block is added to the chain they cannot be modified and attacked by any third person of the system. Here, immutability is one of the added advantages of the blockchain apart from all other processes. Here, it avoids the tampering of the data that gets stored in the environment. All the data are in the form of hash values which are not understandable to any person who tries to access and change the block data. If they try to change data in a single block the hash value of the previous helps to find that our block is tried to get changed by the attackers. In this blockchain, there exists a use of algorithms such as SHA 256 or MD5 to represent the data in the hash format. It is an added advantage to blockchain technology to represent the data in hash form.

1.2 BLOCKCHAIN CONSENSUS PROTOCOLS

The consensus is one of the important phenomena that need to occur in the distributed environment at the time of decision making. The adding or rejection of blocks in the blockchain is decided by a consensus of the nodes in the blockchain network. There are different ways to achieve a consensus in a blockchain network.

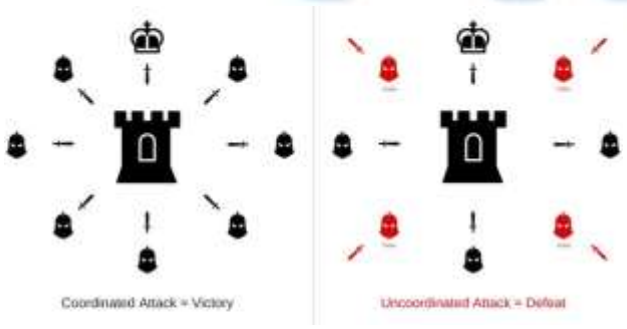

Figure1.2: Byzantine Generals Problem
(https://steemit.com/bitcoin/@humanjets/the-byzantine-generals-problem)

1.2.1 BYZANTINE FAULT TOLERANCE (BFT)

A blockchain network consists of both trusted and untrusted nodes which are generally described as Byzantine General's Problem. Here, the traitor generals lead to the failure of attack in the enemy fort. In Figure1.2 the attack becomes successful when all the generals coordinated the attack at the correct time. Otherwise, due to the presence of the traitor generals, the attack becomes uncoordinated and it leads to the failure of the attack in the enemy fort. It is future avoided by the BFT protocol, where despite the presence of traitors the blockchain network needs to make decisions at a specific situation which helps to make decisions. It is one of the most difficult types of failure.

1.2.1.1 LEADER IS A TRAITOR

If a leader is a faulty or failure then following things will happen,

a. Ignore commands
b. Assign same sequence number to different requests
c. Skip sequence numbers
d. Idle timeouts, commit timeouts
e. Invalid commands such as PREPARE message or multiple PREPREPARE messages for different blocks with the same sequence number.
f. Validators monitor primary's behaviour and trigger view changes to replace a faulty primary

A view change switches to a different leader node when there exists a fault in the previously elected leader. Requires only VIEWCHANGE and NEWVIEW messages alone.

a. VIEWCHANGE - Sent by any validator node that suspects that the primary is faulty
b. NEWVIEW - Sent by the new Leader (Proposer) to all other validators. It is to indicate that the viewchange mode ended and the new leader is elected.

Clients now broadcast messages to the validator. Validator's start timer once the timer ends they start to send VIEWCHANGE messages among themselves. View number changes from V to V+1 and waits for 2f+1 valid VIEWCHANGE messages. Once reached 2f+1 then begin Leader Election and the new primary (leader) sends NEWVIEW to all other validators.

1.2.1.2 VALIDATOR IS A TRAITOR

When a validator is a traitor then following things will happen,

a. Send incorrect commands or messages to other validators.
b. The client waits for 2f+1 correct replies from the validators before accepting the response from them.

1.2.2 PROOF OF WORK (PoW)

For an actor to be elected as a leader and choose the next block to be added to the blockchain, they have to find a solution to a particular mathematical problem. The only way to find a solution to that problem is by brute force (trying all possible combinations). In other words, probabilistically speaking, the actor who will solve the aforementioned problem first the majority of the time is the one who has access to the most computing power. These actors are also called miners.

1.2.3 PROOF OF STAKE (PoS)

Proof of Stake takes away the energy and computational power requirement of PoW and replaces it with the stake. The stake is referred to as an amount of currency that an actor is willing to lock up for a certain amount of time. In return, they get a chance proportional to their stake to be the next leader and select the next block. Various existing coins use pure PoS, such as Nxt and Blackcoin. The main issue with PoS is the so-called nothing-at-stake problem. Essentially, in the case of a fork, stakes are not disincentivized from staking in both chains, and the danger of double-spending problems increase.

1.3 PRACTICAL BYZANTINE FAULT TOLERANCE (PBFT) PROTOCOL

PBFT protocol helps to tolerate the BFT in State Machine Replication which is nothing but the replicated nodes which follow the command of the leader node. This principle applies to the nodes in the blockchain where the leader (proposer) node proposes the block into the blockchain environment and all other nodes have to make decisions despite the presence of the traitor nodes in the network. Here, the PBFT should satisfy the condition $N \geq 3f + 1$. Where the N is the total number of replicas and f is the faulty replicas. If this condition satisfies then only the blockchain network can withstand the PBFT faults.

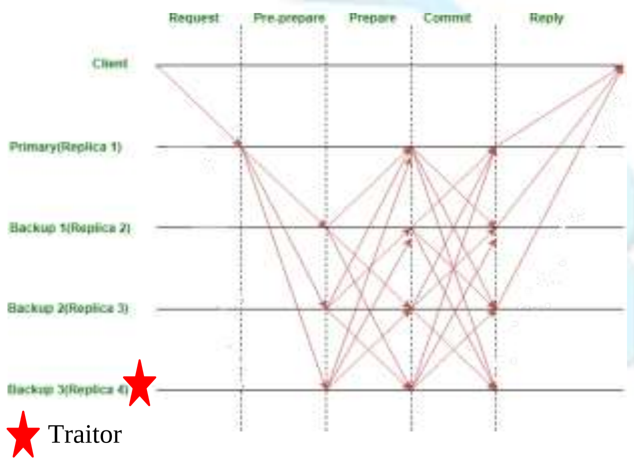

Figure1.3: Working of PBFT Protocol
(https://www.geeksforgeeks.org/practical-byzantine-fault-tolerancepbft/)

PBFT tries to provide a practical Byzantine state machine replication that can work even when malicious nodes are operating in the system. Nodes in a PBFT enabled distributed system are sequentially ordered with one node being the primary (or the leader node) and others referred to as secondary (or the backup nodes). Note here that any eligible node in the system can become the primary by transitioning from secondary to primary (typically, in the case of primary node failure). The goal is that all honest nodes help in reaching a consensus regarding the state of the system using the majority rule. A practical Byzantine Fault Tolerant system can function on the condition that the maximum number of malicious nodes must not be greater than or equal to one-third of all the nodes in the system. As the number of nodes increase, the system becomes more secure. PBFT consensus rounds are broken into 4 phases Figure1.3,

i. The client sends a request to the primary (leader) node.
ii. The primary (leader) node broadcasts the request to all the secondary(backup) nodes.
iii. The nodes (primary and secondary) perform the service requested and then send back a reply to the client.
iv. The request is served successfully when the client receives 'm+1' replies from different nodes in the network with the same result, where m is the maximum number of faulty nodes allowed.

1.4 TYPES OF FAULTS

Faults or failures are the reasons for the disruption of the blockchain framework. Many kinds of faults may happen in the blockchain because blockchain is nothing but a distributed system. The faults that are possible in the distributed system are all possible in the blockchain environment. Despite these faults, the blockchain needs to make a consensus for the smooth working of the system. Some of the major faults are,

a. Fail-stop – The node in the blockchain may fail and the process that runs in the node is stopped suddenly. It is a common fault that happens in the blockchain.
b. Arbitrary node failure – The nodes in the blockchain may fail to return a result or send an incorrect result.
c. Byzantine fault – In this the nodes are not trustworthy, where there is a chance of presence of traitors in the network.
d. Network Partition faults – Distributed systems should run despite delays in transferring messages between nodes or failure of nodes. Replica should be done along with preserving the consistency and availability of the system.

1.5 PROTOCOLS WORKS UNDER PBFT

Some of the protocols that run with the help of PBFT where the consensus algorithms are a little bit modified in their use. PBFT protocol helps to tolerate the faults that happen in the distributed environment even if 1/3 of the faulty nodes are present in it. Many protocols and projects which make use of PBFT are Tendermint, Hyperledger, and, Hotstuff/LibraBFT. They aim to achieve the consensus using PBFT with the help of a maximum of 2/3 nodes in the network even if it contains 1/3 faulty nodes.

1.5.1. TENDERMINT

Tendermint is one of the blockchain environment which are known for securely and consistently replicating an application on many machines. By securely, that mean that Tendermint environment is able to work even in the presence of the 1/3 faulty nodes. By consistently, it means that every non-faulty machine sees the same transaction log and computes the same state. Secure and consistent replication is a fundamental problem in distributed systems; it plays a critical role in the fault tolerance of a broad range of applications, from currencies, and to elections. Tendermint was implemented in the Go language. Similar to the normal case of PBFT with 5 communication steps which are request, pre-prepare, prepare, commit, and reply. The new proposer (leader) is selected and used in every new process or transaction is made. Exchange of messages between the processes using the gossip protocol.

1.5.2 HYPERLEDGER FABRIC

Hyperledger Fabric is an open-source and private permissioned distributed ledger technology (DLT) platform, designed for use in enterprise contexts that delivers some key differentiating capabilities over other popular distributed ledger or blockchain platforms. One of the differentiation is that Hyperledger was established under the Linux Foundation, which itself has a long and very successful history of developing the open source projects under which lot of developments and advances are done with them. Fabric is the first and foremost distributed ledger platform which are able to support smart contracts authored in general-purpose programming languages such as Java, Go and Node.js, rather than constrained domain-specific languages (DSL). Similar to the PBFT protocol. Here, the transaction is submitted to the peers and then the leader is chosen. The hash value is calculated and sends to other peers, the normal replicas produce the same result other than the fault replicas. 2/3 of the replicas should have the same hash value.

1.5.3 HOTSTUFF

HotStuff is a leader-based Byzantine fault-tolerant replication protocol proposed by VMware Research in the year 2018. This hotstuff protocol is similar to the other consensus protocols, where it is considered to be secure and trustworthy and also achieve this behavior in the presence of peer to peer network environment. This protocol also assures the users with the presence of the byzantine fault tolerance behavior. This algorithm makes this environment free from byzantine nodes. Here, the scalability of the algorithm is much better than the other environments. Optimize the complexity of the PBFT algorithm from O(n^3) to O(n). To preserve the Liveness property there, exist N-f non-faulty replicas. With the correct leader, the complexity reduced from O(n^2) to O(n). In view exchange protocols, the complexity reduced from O(n^3) to O(n). Similar to the LibraBFT, the chained Quorum Certificate is formed after the votes are collected from the replicas. View change requires only k+2 rounds instead of 2*k as in PBFT.

1.6 COMPARISON OF PBFT PROTOCOLS

Table1.1 Comparison of PBFT protocols

S.NO	PROPERTIES	HOT STUFF	HYPERLEDGER	TENDERMINT
1	Leader Election	Rotation Leader	Dynamic leader based on time	Rotation Leader
2	Correct leader Message Complexity	O(n)	-	O(n)
3	View change Message Complexity	O(n)	-	O(n)
4	Latency (Roundtrip)	3	-	2
5	Responsiveness	Yes	-	No
6	Security	Safety, Liveness	Safety, Liveness	Safety, Liveness
7	Throughput	Based on the no. of nodes present.	Depends on blocksize. Not efficient beyond 2MB block size.	High performance with 10,000 transactions it can handle.
8	Uniqueness	Chained Hotstuff with the help of Quorum Certificate	Gossip Protocol, Ordering Service for peers	Gossip Protocol

Here, in Table1.1 the properties of all the protocols are compared with one another. It helps to differentiate the working and advantage of one protocol over another. But the theoretical comparison doesn't show more difference than the practical use of each protocol and creates nodes with the help of them.

II. EXISTING RESEARCH WORKS

Castro et.al [1] describes an algorithm that helps to tolerate the byzantine faults. It shows that Byzantine fault tolerance is important in the future because of the increase in malicious attackers and software. This algorithm helps to avoid the faults in the nodes. It works in the state machine replication of nodes where the replicas tend to exhibit the commands of the proposer node. It also helps to preserve the safety and liveness properties of the distributed system. Here, even in the presence of faults, the system withstands that with the help of the condition $N \geq 3f+1$. Here, the f denotes the faults in the network and N denotes the number of nodes present in the system. Here, the PBFT works with the following stages which are request, pre-prepare, prepare, commit, and reply. But, the time and the messages between these stages are higher than the other protocols.

Buchman et.al [2] presents a new protocol which is called as Tendermint. Here, this protocol makes consensus with the help of the Byzantine Fault Tolerant algorithm. So, this protocol is suitable for the state machine replication system where the replicas run the command from the validator node. Here, the proposer proposes the blocks which are executed by the non-validator nodes. Here, the validator or the replicas may become a traitor. When the proposer itself a traitor then it follows the leader election algorithm to choose the next proposer in a particular view. This process is called a view change. It has the same 5 phases because the base protocol for making consensus is nothing but the PBFT protocol.

Yin et.al [3] presents a protocol that is a leader based Byzantine Fault Tolerant (BFT) called HotStuff. Recently the Facebook LibraBFT protocol is based on the HotStuff model. Here, the same PBFT protocol is the base for the working of its functionalities where the blocks are proposed by the leader. Through the leader, the proposed blocks are processed by the validator nodes in the blockchain. It is also suitable for the State Machine Replication (SMR) system. The difference is that in PBFT the view change process requires the time of O (n3) whereas the Hotstuff only takes O (n) alone. It also contains additional phases such as safeNode predicate, decides phase, and Nextview phase. Here, in the safeNode predicate phase helps to find the safety rules for accepting the proposals of blocks. Decide phase is based on the replies from the commit phase and increase the view number. In the nextView interrupt phase, it is the phase where all the replicas wait for few seconds to find this interrupt if it presents it need the new leader is going to get elected. This protocol also exhibits the safety, liveness, and complexity property as that of the normal PBFT protocol.

Androulaki et.al [4] describes the Hyperledger Fabric which is based on the Byzantine fault Tolerant (BFT) protocol. Here, the fabric contains five steps. Order service is used to broadcast the messages between the nodes to make decision and consensus. Membership Service Provider used to act as an entry point for all the peers inside the network. Here, all the peers and registered and given the keys to communicating between them. Here, in this work, the total working of the fabric is illustrated and also conducted some of the performance metrics along with them. This is paper is written by the IBM people where the total work is tested by uploading them into the IBM cloud also. Here, the various uses of the consensus algorithm such are solo, Kafka is also used. In this work, the throughput and the transaction speed comparison is also carried out. But, no such measuring tool is used to do the exact measuring of these layers. Here, only the normal performance of the environment is measured and taken into account. This paper was worked on the v.1.0 of the hyperledger fabric environment, where the raft protocols are not yet added in the hyperledger model.

Tendermint et.al [5] describes that the conduction of tests on the tendermint nodes with the help of the Jepsen tools. Here, the distributed nodes and BFT of the tendermint protocols are analyzed. It ranges the various tests such as network partition, failure of nodes, clock delays, byzantine faults, and also arbitrary node failures. Each test is conducted with the Jepsen tool which is all in the form of the test cases and each case has generators that help to stimulate these faults. These faults are all represented in the form of the graph where the time vs latency graph was stimulated. These faults and test results help to understand the withstand capacity of the Tendermint blockchain environment. It also concluded that Tendermint appeared to satisfy the safety properties of the PBFT protocols. Only the liveness got affected due to the presence of the byzantine faults in the nodes.

Nasreen et.al [6] describes the methods of tolerance of the byzantine faults in the distributed environment. The presence of malware in the software can also lead to one kind of failure. But, that fault is not taken in this work. This work includes faults such as crash failure, Omission failure, Timing failure, Response failure, and Arbitrary failure. A crash failure is said to have occurred when a server prematurely halts but was working correctly until it stopped. An omission failure occurs when a server fails to respond to a request. This kind of receive omission failure the server can also fail to receive the request from the user. It not only affects the current state of the server but also affects the messages which are sent to them. Similarly, a send omission failure occurs when the server has completed its work but somehow fails to send a response. Timing failures occur when the response lies outside a particular real-time interval. An important kind of failure is called response failure. This failure is nothing but sending the incorrect output to the request made to them. This kind of behaviour should also be avoided in a distributed environment. A server is subject to two kinds of response failures. In the value failure model, a wrong reply is sent by the server. The second class of response failure is known as a state transition failure. This kind of failure happens when the server responds unexpectedly to an incoming request. Among all other failures, the most serious are arbitrary failures, also known as a Byzantine failure. When a Byzantine failure occurs, the system may respond in any arbitrary way unless it is designed to have Byzantine fault tolerance. Byzantine fault tolerance is very critical because small arbitrary failures in one node can bring down the whole system. Here, this work discussed all kinds of failures possible in the state replication machine. But, it only covered the theoretical view of those faults instead of the stimulation of those failures.

IBM [7] describes the Hyperledger Fabric protocol which in detail which covers all layers present in it. The most important difference between the fabric from its other hyperledger projects is that it has the private channel facility which is known only to the parties involved. It contains four types of services. The membership service provider is one of the special features of this blockchain which helps to add the peers inside the environment and also helps to provide the keys needed by the peers to communicate between them. Then, the blockchain service which helps to achieve consensus without the byzantine faults. Transaction service is used to help and accept the transaction from the clients of the blockchain framework and also to verification of those transactions. Chaincode service, which is nothing but actions which is taken by running the validated transactions. This whitepaper covers the actual working model of the fabric in a very detailed manner.

Sukhwani et.al [8] proposed to investigate whether the consensus process using Practical Byzantine Fault Tolerance (PBFT) and it acts as a performance bottleneck for networks which are containing a large number of peers. In this paper, the test is conducted which helps to address the transmission of messages between peers, time is taken to process incoming messages and time taken to move to the next stage of PBFT. The increasing number of peers cause bottleneck time to commit the block. But, other than that not much analysis is made to make a comparison of the work between them. This work is limited to a specific work which it achieves to find the cause of the bottleneck.

Sousa et.al [9] proposed to achieve good ordering service in the absence of Byzantine faults with the help of PBFT protocol. It also proposed the BFT SMART protocol is speed and achieved 10000 transactions per second which are better in the time of achieving the consensus in the fabric environment. But the Wrappers which was used in the hyperledger fabric may tend to face other faults such as node failure and arbitrary faults. Here, these additional layers also cause a time delay and throughput calculation. This paper has the added advantage of the smart protocol which reduces the attacks and malicious users but along with the disadvantage of complex work and increase in time.

Andola et.al [10] proposed that there some vulnerabilities in the blockchain environment of hyperledger fabric. This paper pointed out two major vulnerabilities in the fabric framework. The first one is that the endorser is known to all the members present in the channel. which makes the possibility of Denial of Service attack. The second one is the compromising of the nodes inside the channel which leads to the leakage of the information. This paper proposed various models to address these vulnerabilities. These models are group signature verification and increasing the signature method by also adding zero-knowledge proof. Even though these methods are more time consuming for making a consensus and also the hyperledger fabric version used it older than the current advanced architecture.

Table 2.1 Comparison of the Literature Survey

REFERENCE	INFERENCE	PROS	CONS
[1]	Byzantine fault tolerance is important in the future because of the increase in malicious attackers and software. This algorithm helps to avoid the faults in the nodes.	Condition N ≥ 3f+1 is satisfied to avoid failure.	Time and internal messages are higher.
[4]	In this work, the total working of the fabric is illustrated and also conducted some of the performance metrics along with them.	Using cloud technology and tried various consensus.	Normal performance is measured without any tool.
[6]	The methods of tolerance of the byzantine faults in the distributed environment are proposed in this paper.	Illustrated all the possible faults in the distributed system	Covered the theoretical view of those faults instead of the stimulation of those failures.
[8]	Practical Byzantine Fault Tolerance (PBFT) acts as a performance bottleneck for networks which are containing a large number of peers.	The bottleneck problem is identified.	No solutions and analyses are made.
[9]	Good ordering service in the absence of Byzantine faults with the help of PBFT protocol. It also proposed BFT SMART protocol is speed and achieved 10000 transactions per second	A smart protocol which reduces the attacks and malicious users	Complex work and increase in time to commit.
[10]	Two major vulnerabilities are identified. The endorser is known to all and also the information within the channel is not safe	New cryptographic signature methods	Time Consuming and also lower version is tested.
[11]	Two phases were proposed. One is to get to know the behaviour and another one is to correct the limitations present.	Focused analysis of the services in fabric.	The older version only and also bottleneck is not addressed.
[12]	Measures the fabric environment with malicious users. The use of the PBFT protocol is analyzed with the tests in them.	Primary and secondary failures are considered in this method.	The older version is tested and not clear about the tested consensus.

Thakkar et.al [11] measured the performance of the hyperledger fabric environment and based on these performance scales this paper also helped to optimize the environment. Here, for that they proposed two phases, the first phase is to understand the fabric with various parameters such as block size, transaction throughput, and also the policies, channels in them. This phase gave the bottlenecks present in them which are endorsement policy, transaction policy, validation of the blocks, and commit them. The second phase focused on the parallelizing of these police models to increase the performance of this environment. The version used in this work is again the older one where the works of the performance are somewhat not good according to the current version. This work also not clear in addressing the bottlenecks which are the special features of this private permissioned hyperledger fabric environment.

Wang [12] proposed a paper which is to evaluate the fabric environment with the presence of the malicious attacks present in it. Here, the adoption of the PBFT protocol to make the consensus for the blocks in the environment. The presence of faults may delay the process and also lead to the attack from the attackers. Here, the author proposed an hyperledger fabric version of v0.6 is tested under the attacks. In the primary and secondary attacks, the nodes make delays in committing the blocks, and throughput is affected. This paperwork is mostly on the PBFT protocol which is not clear on the use of the protocol in them. Because the fabric has three kinds of consensus protocols in later versions of them

III. INFERENCE OF RELATED WORKS

The PBFT protocols can only able to withstand the 1/3 faults in the network. Theoretically, the faults in the nodes can cause serious problems to the blockchain. These faults may cause the blockchain to behave unrealistically and also it affects the liveliness property of the blockchain environment. A lot of open-source blockchain environments emerged during the past five years. But the elasticity of those environments has to check before being completely reliable. Tendermint, Hyperledger Fabric, and Hotstuff are some of the new open-source blockchain environments which are needed to be studied and analyzed for their consensus schemes to find whether the blockchain environment can manage the byzantine faults. Here, the Hyperledger Fabric, Tendermint and Hotstuff environments are widely being considered in many supply-chain models. The usage of this environment rises to the question for the users whether it can believe in their business activities. These doubts make the blockchain technology much less usable by the customers in their day to day activities. But, these problems can only be addressed by conducting experiments and making more projects and research works in this field. Here, this proposed work help to address those problems and also allows us to analyze this environment widely. This work also helps to address the performance of each blockchain environment and also to differentiate one from another for their specific use. This encourages users to try and utilize these blockchain environments for their needs and make good use of our society.

IV. CONCLUSION AND FUTURE WORK

The blockchain environment is now one of the decentralized trustworthy models. This leads to creating a blind belief in this technology. The consensus making of the blockchain in the environment of faults analysis helps to understand the blockchain environment and its possible outcomes. The proposed approach is to conduct the tests by introducing faults into the normally behaving blockchain environment. By doing that, the working of the blockchain environment in the presence of the faults can easily be identified. This helps to understand the real-world working of the blockchain which contains both normal nodes as well as the byzantine node. These tests need to be conducted in the Tendermint environment where the introduction of the fail and stop, arbitrary node failure, and partitioning of the network need to be done with the help of the Jepsen tool. Then, the Hyperledger Fabric environment can be taken where the tests need to be conducted and measured the performance and behaviour of them with the help of the caliper tool.

REFERENCE

[1] Miguel Castro and Barbara Liskov, Practical Byzantine Fault Tolerance, Appears in the Proceedings of the Third Symposium on Operating Systems Design and Implementation, New Orleans, USA, MIT, Pg.No 1-14, 1999.
[2] Ethan Buchman, Jae Kwon, and Zarko Milosevic, The latest gossip on BFT consensus, Cornell University, Pg No 1-14, 2018.
[3] Maofan Yin, Dahlia Malkhi, Michael K. Reiter, Guy Golan Gueta, and Ittai Abraham, HotStuff: BFT Consensus in the Lens of Blockchain, Cornell University, Pg No 1-23, 2019.
[4] Elli androulaki, Christian Cachin, Christopher Ferris, Srinivasan Murulidharan, Chet Murthy, Binh Nguyen, Manish Sethi, Chrysoula Stathakopoulou, Hyperledger Fabric: A distributed operating system for Permissioned Blockchains, Cornell University, Pg No 1-15, 2018.
[5] Tendermint, Jepsen teams, Tendermint 0.10.2, Pg No 1-4, 2017.
[6] M A Nasreena, Amal Ganeshb, Sunitha, A Study on Byzantine Fault Tolerance Methods in Distributed Networks, Computer Science & Engineering, Elsevier Pg No 50-54, 2016.
[7] IBM, Hyperledger Fabric Whitepaper, 2018.
[8] Harish Sukhwani, Jose M. Martınez, Xiaolin Chang, Kishor S. Trivedi, and Andy Rindos, Performance Modeling of PBFT Consensus Process for Permissioned Blockchain Network (Hyperledger Fabric), IEEE 36th Symposium on Reliable Distributed Systems, IEEE, Pg No 1-10, 2017.
[9] Joao Sousa, Marko Vukolic and Alysson Bessani, A Byzantine Fault-Tolerant Ordering Service for the Hyperledger Fabric Blockchain Platform, 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, IEEE, Pg No 314-320, 2018.
[10] Nitish Andola, Raghav, Manas Gogoi, S. Venkatesan, Shekhar Verma, Vulnerabilities on Hyperledger Fabric, Pervasive and Mobile Computing, ScienceDirect, Pg No 1-13, 2019.
[11] Parth Thakkar, Senthil Nathan N, Balaji Viswanathan, Performance benchmarking and optimization of Hyperledger Fabric Blockchain Platform, 2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), IEEE, Pg No 1-13, 2018.
[12] Shuo Wang, Performance Evaluation of Hyperledger Fabric with Malicious Behavior, International Conference on Blockchain, Springer, Pg No 211-219, 2019.