A Survey on Byzantine Agreement Algorithms in Distributed Systems

Sachin P C and Amit Dua

Computer Science & Information Systems, Birla Institute of Technology & Science, Pilani, IND.
h20180140p@alumni.bits-pilani.ac.in,amit.dua@pilani.bits-pilani.ac.in

Abstract. Byzantine Generals problem is considered as one of the complex problems present in Distributed Systems. Byzantine nodes with its arbitrary nature will hugely affect the reliability and efficiency of the systems and hence it is considered as one of the main problems in distributed systems where high reliability and efficiency is required. Therefore, prominence is given to achieve Byzantine agreement in Distributed systems and there have been several algorithms proposed to solve the Byzantine Agreement problems in a fault-tolerant distributed system, Cloud Computing Environments etc. In this paper, we analyze different Byzantine Algorithms and compare them to observe which algorithm performs better in which scenario.

Keywords: Agreement, Byzantine, Distributed Systems, Randomization, Validity

1 Introduction

Byzantine Agreement Algorithms are proposed for the Byzantine problems faced in the Distributed Systems where, several processors, which are separated by a distance comes to a collective agreement before a particular action is taken among the set of processors. Byzantine Algorithms are proposed to make sure that the computer systems are reliable and can handle malfunctioning computers which might give conflicting information to different computer systems. One of the very first problems which Byzantine Agreement deals with is Two generals problem. However, until now there is no protocol which has been proposed such that it can solve two-generals problem. The extension of this algorithm leads to the common problem in distributed systems which is Byzantine General problem. Byzantine General problem deals with 'n' processors where 'm' are faulty processors.The Byzantine Generals issue can be analyzed with oral communication only, as long as the number of defective nodes are less than $\frac{1}{3}$ of the maximum allowable nodes.If the number of defective nodes present in the system is greater than $\frac{1}{3}$ of the maximum given nodes, then agreement among the fault-free nodes can only be achieved by digital signatures. Byzantine Generals problem leads us to the concept of Byzantine fault tolerance.Solving the problems of Byzantine defects is the ability of the distributed systems to come to a common agreement

and validity by all the not faulty nodes irrespective of the effect of faulty processors. Byzantine Generals problem is considered as one of the complex possible failures in the distributed systems. So, how to design and build a distributed system that can survive the Byzantine Generals problem? Some of the nodes in the systems will be defective nodes which send wrong messages to multiple nodes, i.e., it sends some message to one of the processors, and some other message to another processor and hence this leads to an incorrect agreement among the fault-free processors. There are several algorithms and protocols proposed to solve Byzantine problems. But, why do we have to study Byzantine failure problems? Here is one example where Byzantine failure plays a prominent role: consider Bit coin systems, in that we cant trust everyone who uses Bit coin and hence if any user is a malicious user, we don't want them to take down the whole network of bit coin and also we don't want them to collect all the money for themselves. Also another reason why we need to solve Byzantine failures is to provide extreme reliability.Consider Air Flight System where reliability is of at most priorities. So, if the hardware failure exhibits Byzantine-like behavior and Air Flight System can receive wrong information, it leads to many problems which is unacceptable. Even Byzantine problem exists in cloud computing systems which consists of huge amount of memories and processors, etc. In cloud computing environments, every server node in the environment should have to co operate with other nodes to satisfy different customer needs. Hence, this leads to issue of problems created by defective nodes which has to be taken care of to make sure that high reliability is provided. In Table 1, we can see the presumptions made by some of the previous works to achieve Byzantine Agreement Algorithms. This alone indicates that not all algorithms can be considered for a particular scenario. Each algorithm might work better for different algorithms.

When faults are present, then attaining agreement is a very prominent problem because algorithm has to be proposed such that it can handle all the faults present and obtain a correct solution. There have been several algorithms presented to resolve the Byzantine Agreement problems. In this paper we analyze :

1. Byzantine Generals Problem[1]
2. Solution to Byzantine Algorithm when both links and processors are subjected to Hybrid Faults[2].
3. A Randomized protocol to solve Byzantine Agreement[3]
4. Protocol to solve Byzantine Agreement in Cloud Environment[4]
5. Degradable Byzantine Agreement[5]
6. Protocol to have Distributed Agreement when both links and nodes are subjected to faults[6]

2 Analysis of The Algorithms

2.1 The Byzantine Generals Problem[1]:

This algorithm is one of the first algorithms proposed to deal with Byzantine Agreement. This algorithm directly considers the actual Byzantine generals

Table 1. This table shows the assumptions made in order to achieve Byzantine Agreement.

Assumptions / Previous Works	Type of Network	Type of Processor Fault	Type of Link Fault
Lamport et al.[1]	Fully Connected Network	Arbitrary Processor Fault	-
Dolev [8]	General Network	Arbitrary Processor Fault	-
Christian et al.[9]	Fully connected Network	Arbitrary and Dormant Processor fault	-
Thambidurai and Park [10]	Fully connected Network	Hybrid Processor Fault	-
Meyer and Pradhan [11]	General Network	Hybrid Processor Fault	-
Lincoln and Rushby[12]	Fully connected Network	Hybrid Processor Fault	-
Babaoglu and Drummond [13]	Broadcast Network	Arbitrary Processor Fault	Arbitrary Link Fault
Yan and Chin[14]	Fully connected Network	-	Arbitrary Link Fault
Yan et al.[15]	Fully connected Network	Arbitrary Processor Fault	Arbitrary Link Fault

problem and gives a solution to it where only oral messages are sufficient. However, this algorithm works only if total traitors are less than $\frac{1}{2}$ of the present number of generals. Basically, In distributed systems, a conflicting faulty processor sends different information to different systems. This leads to unreliable communication in distributed systems. All the generals must need a strategy to avoid traitors to make loyal generals come to an incorrect agreement:

1. All not faulty generals should come to a common agreement once all messages are sent.
2. A bad plan should not be adopted by the loyal generals if small number of traitors are present.

Byzantine Generals Algorithm[1] states that: A commander should send the information to the other generals:

1. All not faulty generals come to the same agreement.
2. If the commander is not faulty, then all the not faulty lieutenants should obey the order sent by the commander.

These conditions are called as interactive consistency conditions.

If faulty generals are at least $\frac{1}{3}$, then the solution cannot be obtained using oral messages. Hence a signed messages algorithm is given to unfold this situation. In this, the generals send not forge-able signed messages where,

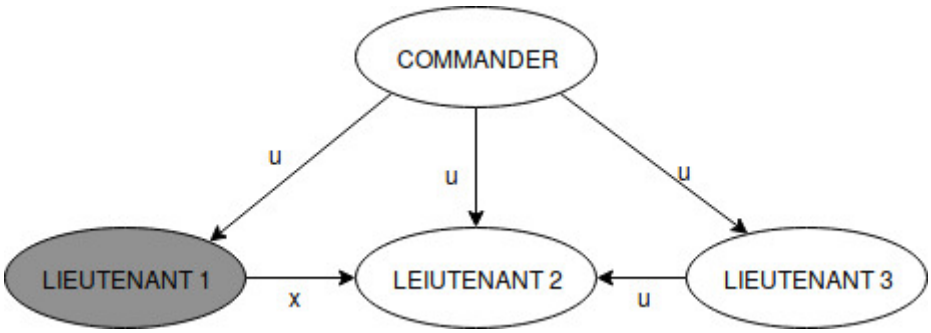

Fig. 1. This diagram shows the scenario where one of the general Lieutenant is a traitor

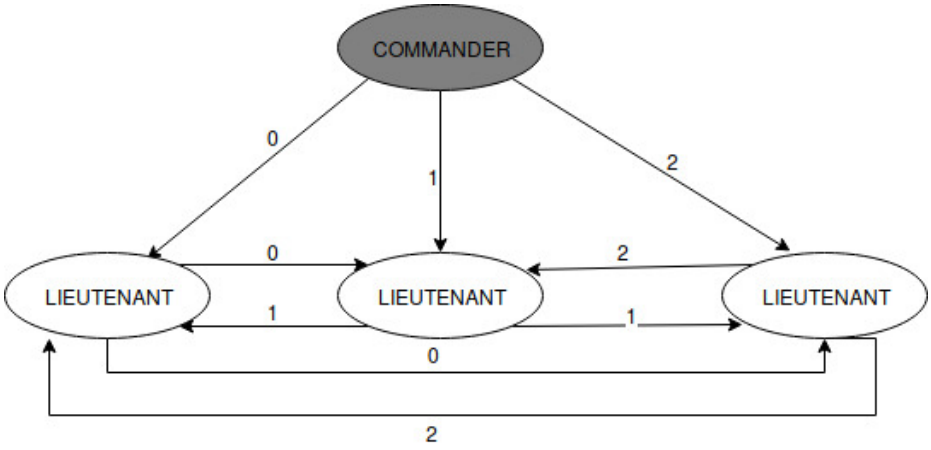

Fig. 2. This diagram shows the scenario where the Commander itself is traitor

1. A loyal general's signed messages cannot be forged and if the signed message sent is changed, it's possible to find out.
2. General's signed messages originality can be verified by everyone.

Hence this paper presents algorithms to solve Byzantine generals problem considering different scenarios and hence can be used in reliable distributed systems. However, both oral messages and signed messages algorithm require 'm+1' rounds to ensure validity and agreement which is expensive concerning both total messages required and time.

2.2 Solution to Byzantine Algorithm when both links and processors are subjected to Hybrid Faults[2]

In earlier stages of Byzantine Agreement protocols, the problem was solved for only single fault type with network being either fully connected or partially connected. However, the processor faults can be mixed. Hence this algorithm is proposed for a general network where the links and processors can both show different types of faults simultaneously(Hybrid Fault Model). For a general network consideration, in the proposed algorithm, messages sent is least and also least in terms of the total rounds performed to achieve Byzantine Agreement. For a given network, this algorithm allows the maximum possible faulty processors.

The below notions should be considered in order to solve Byzantine Agreement using the protocol used in this paper,

1. The prior information related to the systems faulty status is not required to be known by the non-faulty processors.
2. The network topology is a general topology. i.e., the network need not be entirely connected.
3. Links and processors are put through Hybrid Fault Model.

The goal of this protocol is, the number of permissible nodes which are not loyal should be maximum when both types of faults are present.Numerous prior works for the Byzantine agreement have assumed only Processors failure but in general, the links can also fail, and the link failure can also be of mixed faults. Hence in this algorithm, both link and processors are put through Hybrid faults.

Generalized Protocol for the Byzantine Agreement problem (GPBA) can be defined as the algorithm used to give a solution to Byzantine Agreement issue with respect to common network. The total tolerable faulty components in a generalized protocol for Byzantine Agreement is more than all the other current protocols proposed to solve this problem when the hybrid fault model is examined. GPBA uses oral communication to transfer messages. This protocol considers synchronous network.

The goal of GPBA is to make sure that all the not faulty processors conclude on the same agreement which was broadcast-ed.

GPBA has to satisfy the following two criteria:

1. Agreement : Every not faulty processors should come to a common agreement.

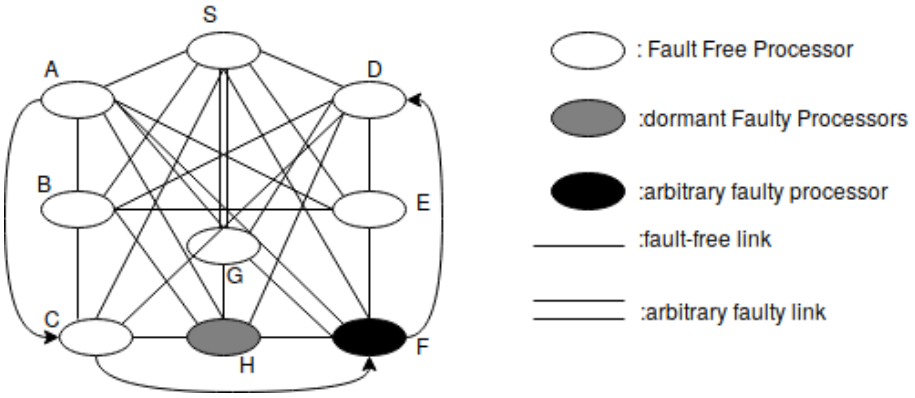

Fig. 3. Both Links and Processors to put through Hybrid Fault model where total processors 'n' is 9 and connectivity of each processors 'c' is 6).

2. Validity : All not faulty processors should agree on 'v', if the source is not faulty and the message sent by it is 'v', i.e., $v = v_s$.

$(u+1)$number of message exchange rounds are needed for each processor so that all of them agree on same value, where $u = \lfloor(\frac{n-1}{3})\rfloor$.Least rounds required to achieve agreement in GPBA algorithm is $(u+1)$.

The GPBA algorithm achieves Byzantine agreement under certain conditions such as

1. $n > (3 * S_i + S_j)$, and
2. $c > (2 * S_i + S_j) + 2 * (T_i + T_j)$

n= total processors
S_i = the arbitrary processor faults
S_j = Dormant processor faults
c = connectivity of the networks
T_i = Arbitrary link faults
T_j = Dormant Link Faults
The GPBA algorithm's complexity is measured with respect to:

1. The maximum rounds needed to attain Agreement,
2. The maximum faulty processors which can be present.

GPBA needs $(u+1)$ rounds and takes $O(c * n + u * c * n^2)$ number of messages to solve the Byzantine Agreement problem in a common network with hybrid faults, if $n > (3 * S_i + S_j)$ and $c > (2 * S_i + S_j) + 2 * (T_i + T_j)$, where
$u = \lfloor(\frac{n-1}{3})\rfloor$
GPBA is the best algorithm present when considered with respect to the faulty processors permitted and the number of rounds it takes to perform the protocol.

2.3 A Randomized protocol to solve Byzantine Agreement[3]

The previous algorithms we saw was deterministic algorithms which were proposed for a general network. The algorithm proposed in this paper is a randomized algorithm which is used to solve the Byzantine agreement problem where 't' number of nodes can be faulty out of 'n' nodes.This algorithm works only for synchronous systems. The total rounds needed for this algorithm is $O(\frac{t}{logn})$ and total message bits is $O(\frac{n^2tf}{logn})$. The algorithm is independent of how the failures are distributed over the network. The above performance mentioned can be improvised to fixed rounds and $O(n^2)$ number of message bits assuming that the faulty nodes are distributed uniformly.

This randomized algorithm works for any number of nodes 'n', if $n >= (3 * t + 1)$, 't' is the maximum allowable faulty nodes.This algorithm is simple and efficiently implemented and hence is of great need in real implementations. When performance is considered, Deterministic algorithms performance is less than this randomized algorithm's performance, and also it doesn't need any cryptographic techniques to perform this algorithm.

The randomized algorithm discussed here, for randomization, uses global coin tosses. In this algorithm, the network is assumed to be reliable and fully connected.At every round, coins might be tossed by some processors for some of the calculations. Not faulty processors, if they tossed coin, send the value they got from the coin toss instead of sending the incorrect values as sent by the faulty processors. Every processor, once the algorithm starts, sends its initial value 'v' chosen from a group of values 'V'.The aim of this randomized algorithm is that every processor will have an answer, once it has sent messages. The answers need two requirements to be satisfied:

1. *Agreement*: The answers given by all the not faulty processors is same.
2. *Validity*: If there is one value with which all the processors start the algorithm, then at the end of the algorithm, the correct answer will be the value with which all processors started the algorithm.

This randomized algorithm has three properties.

Validity: If there is one value with which all the not faulty processors start its algorithm, then during the second round of first epoch, every not faulty processor agree on that same value with which they all started.

Agreement: If a not faulty processors agree for the first time on a value, during an epoch s, then by the second round of the next epoch, all of the not faulty processors agree on that value.

Termination: There will be minimum one value in every epoch such that all the not faulty processors will agree on the correct value during the finish of the next epoch.

Under the assumption that all the faulty processors are separated evenly, then the randomized algorithm proposed is expected to run in constant time.

There are four principle limitations this algorithm faces, they are:

1. This algorithm limits the total defective nodes. The maximum allowable defective nodes should be at most $\frac{1}{3}$ of the total present nodes. It is proven that no cryptographic algorithm exists which can perform correctly if the maximum defective nodes allowable is more than $\frac{1}{3}$ of the total nodes present.
2. The assumption concerning the network is that it is a reliable communication.
3. It is assumed that the faulty processors cannot determine the coin tosses made by the not faulty processors in the future.
4. During every round it is assumed that the not faulty processors will not subvert any processors.

To achieve agreement using this randomized algorithm, the rounds which is performed is $O(\frac{t}{logn})$ and also the random bits taken by every processor is above 1. This randomized algorithm is preferable when all the faulty processors are evenly spread across the network because the expected running time becomes a constant when the faulty processors are uniformly distributed.

2.4 Protocol to solve Byzantine Agreement in Cloud Environment[4]

In recent years cloud computing has taken substantial growth, and its popularity is increasing. Even Byzantine problem exists in cloud computing systems which consists of huge amount of memories and processors and also will contain networks made up of fast transmission links. In this domain, every node has to interact with other nodes and have to communicate with each other in order to satisfy the needs of the customers. So, one of the major problems in the cloud environment is reliability where faulty nodes should be taken care of in order to achieve reliability. Hence, achieving Byzantine Agreement in a cloud system is the goals of the algorithm.

The concept used in this algorithm to achieve Byzantine Agreement is "Early stopping protocol(ESP)" which is used to make sure that all nodes agree on same value early in separate rounds. This algorithm can also be applied for mixed fault failure, and it takes optimal number of rounds when compared to other algorithms in the cloud domain. Therefore, this algorithm proposed can resolve the Byzantine Agreement in the cloud domain and also improves the reliability which is one of the important factors to be considered.

The goal of "Fault Diagnosis Agreement(FDA)" problem is that the not faulty processors use the least number of message transfers to find the processors which are faulty. This algorithm uses ESP so that the number of messages which are exchanged is minimized and improve the reliability and efficiency in the cloud computing environment.

This algorithm also considers hybrid fault for the processors and hence deals with both dormant and arbitrary faults. In this algorithm, the following conditions must be met:

Agreement: Every not faulty processor can determine the faulty components

Fairness: It is made sure that all the not faulty processors are determined as not faulty processors and faulty processors as faulty processors. Not faulty processors should not be falsely determined as faulty processors.

The algorithm proposed is "Early Diagnosis Cloud Agreement (EDCA)" protocol which solves the Byzantine Agreement. In a cloud environment, EDCA algorithm can be applied on a network in which maximum allowable faulty processors are present, and the total rounds taken to perform the algorithm is optimal. The total rounds needed to exchange messages is $r = min\{f_r + 2, f_a + 1\}$. The EDCA protocol has three phases namely, " the message exchange" phase, "the decision-making" phase, and the "fault diagnosis" phase. The EDCA algorithm helps in increasing the reliability in cloud computing environments.

For the EDCA algorithm to work correctly, the total number of processors $n > b(\frac{n-1}{3c}) + 2S_i + S_j$ and $c > 2S_i + S_j$. c=connectivity of the networks S_i= arbitrary fault S_j= dormant faults The maximum rounds needed is $min\{S_r + 2, S_i + 1\}$

The "Early Diagnosis Cloud Agreement" integrates the idea of "Early Stopping" and "Fault Diagnosis" so that Byzantine Agreement can be achieved in an efficient and quick way in cloud computing domain. Only $min\{S_r + 2, S_i + 1\}$ number of messages are needed so that every not faulty processors can agree on the correct value in a cloud computing domain.

2.5 Degradable Byzantine Agreement.

The algorithms presented before needs every not faulty nodes to agree on an answer such that it is same in all fault-free processors. In this algorithm, Byzantine agreement can be achieved if the total defectives is t($t < \frac{1}{2} * n$) and also it can accomplish a degraded form of agreement if the number of faults is u and $u >= t$. This leads to every not faulty processors to decide on two values out of which one will definitely be the default value.

In this protocol, the channels complete calculations on an input which they receive from the sender and then the value generated by every channel is produced to an outside node. This node will calculate the value which is received maximum times from every channel and using that finds the maximum value.

The algorithm should ensure certain conditions such as:

1. The outside node gets the right value by finding the majority value if the provided system's channels are 3t with only one sender and assuming the number of faulty channels are t, and the sender is not faulty.
2. Every not faulty nodes will be in the same state, where total defective nodes permissible is t.

The main motive of this algorithm is that it is much better to use default value instead of the wrong value, especially in safety-critical systems. If the total number of nodes is n, then every fault-free nodes can agree with t/u degradable agreement only if $n > 2t + u$.

The network connectivity should be minimum $t + u + 1$ for this algorithm to work properly. If the faulty processors are less than t, then this algorithm achieves Byzantine agreement, i.e., every not faulty node will come to a common agreement. If the defective nodes are $> t(but <= u)$, every not faulty processors

Table 2. This table shows the comparison between different algorithms with respect to certain parameters.

	Deterministic / Randomized	Number of Faulty processors acceptable	Number of Rounds	Network Fault Type	Processor Fault Type	Synchronous / Asynchronous
The Byzantine Generals Problem[1]	Deterministic	m, $n >= 3m+1$	m+1	Reliable and Fully Connected	Single Fault Type	Synchronous
Byzantine Agreement in the presence of Mixed Faults on Processors and Links[2]	Deterministic	t, $y = \lfloor(\frac{n-1}{3})\rfloor$, $n > (3 * S_i + S_j), (2 * S_i + S_j) + 2 * (T_i + T_j)$	u+1	Unreliable and General Network	Mixed Fault Type	Synchronous
A simple and efficient Randomized Byzantine Agreement Algorithm[3]	Randomized	t, $n >= 3t+1$	$O(\frac{t}{logn})$	Reliable and Fully Connected	Mixed Fault type	Synchronous
An Agreement Under Early Stopping and Fault Diagnosis Protocol in a cloud Computing Environment[4]	Deterministic	$S_i + S_j$, $n > \lfloor(\frac{n-1}{3})\rfloor + 2S_i + 2S_j, c > 2S_i + S_j$	$min\{S_r + 2, S_i + 1\}$	General Network	Mixed Fault type	Synchronous
Degradable Byzantine Agreement[5]	Deterministic	$u, u >= t, n > (2 * t + u), c > t + u$		General Network	Mixed Fault type	Synchronous
Distributed Agreement in the Presence of Processor and communication faults[6]	Deterministic	t , $n >= 3t+1$	$t + 1$	General Network	Mixed Fault Type	Synchronous

can decide on either one or two values, where one of the value must be the default value.

2.6 Protocol to have Distributed Agreement when both links and nodes are subjected to faults.

In this protocol, Byzantine algorithm's solution is presented where if a processor cannot send or receive messages, then that is also considered as a failure. In this algorithm considered, processors may fail in both sending and receiving of messages. i.e., a faulty processor can halt abruptly and also can fail to receive or send messages required by the protocol. A faulty processor can produce messages independent of those sent by the loyal processors. However, the message agreed

Table 3. Least amount of total processors required for t/u Degradable Agreement.

u / t	1	2	3	4	5
1	4	5	6	7	8
2	-	7	8	9	10
3	-	-	10	11	12

upon by the loyal nodes must be correct and same concerning every not faulty nodes. This algorithm provides solution to the more realist models of failures.

When given a less restricted type of network and processors and its fault types, this algorithm gives the best optimal solution concerning the time taken, messages sent and received. This algorithm performs same as that of other algorithms proposed, when the network and processor fault types are more restrictive. However, this algorithm is applied for the less restrictive type of failure hence is better than other algorithms.

If the total nodes is n and the number of nodes which perform send-faults is at most t and $n > (t + 1)$, then this algorithm makes sure that the Byzantine Agreement is achieved. Here we can observe that the algorithm does not put a bound on the number of receiver faults because this algorithm treats receiver faults as benign. Hence, this indicates that only send faults are relevant. The complexity of the early stopping protocol is $O(fn^2)$ because the amount of bits used for message sending and receiving is that much. This complexity is identical to the complexity of "crash-fault protocol[7]". But this algorithm is performed on a less restrictive failures, and hence this algorithm performs better for more variety of problems.

3 CONCLUSIONS

In all the algorithms observed, it works only for synchronous processors. Even if one of the processors is asynchronous, achieving Byzantine algorithm will be complicated. Also, all the algorithm works only if the total number of nodes in the system is greater than 3 times the number of faulty nodes. If this condition is not satisfied, then we need to use cryptographic methods to ensure agreements among processors. However, the complexity of each algorithm to perform the Byzantine Agreement algorithm is different. Byzantine Generals Problem was one of the first algorithms proposed which became the basis for all other algorithms."Randomized protocol to solve Byzantine Agreement" performs better than Byzantine generals problem. Randomized algorithm analyzed performs better than other algorithms only in a Fully connected network. "Protocol to solve Byzantine Agreement in Cloud Environment" proposed is the optimal solution for a Byzantine Agreement in a cloud environment. "Solution to Byzantine Algorithm when both links and processors are subjected to Hybrid Faults" performs much better than degradable Byzantine algorithm but the latter allows more faulty processors than the former. Hence, by the analysis of the algorithms

mentioned above, we can say mention that given a particular scenario, each algorithms performance varies and different algorithms perform better in different scenarios.

References

1. L. Lamport, R. Shostak, and M. Pease. : The Byzantine Generals Problem. ACM Trans. Programming Language Systems, vol. 4, no. 3,pp. 382-401, July 1982.
2. B. Chor And Brian A. Coan. : A Simple and Efficient Randomized Byzantine Agreement Algorithm. IEEE Transactions on Software Engineering, vol. 11, no. 3,pp. 531-539, June 1985.
3. H.-S. Siu, Y.-H. Chin and W.-P. Yang. : Byzantine Agreement in the Presence of Mixed Faults on Processors and Links. IEEE Transactions on Parallel and Distributed Systems, vol. 9, no. 4,pp. 335-345, Apr 1998.
4. M.L. chiangi, C.L. Chen and H.C. Hsieh. : An Agreement Under Early Stopping and Fault Diagnosis Protocol in a Cloud Computing Environment. IEEE Access,pp. 44868 - 44875, July 2018.
5. N.H. Vaidya and D.K. Pradhan : Degradable Byzantine Agreement. IEEE transactions on Computers, vol. 44, no. 1,pp. 146-150, Jan 1995.
6. K.J. Perry and S. Toueg. : Distributed Agreement in the Presence of Processor and Communication Faults. IEEE Transactions on Software Engineering, vol. SE-12, no. 3,pp. 477-482, Mar 1986.
7. L. Lamport and M. Fischer. : Byzantine generals and transaction commit protocols. SRI Int.,April 1982.
8. D. Dolev. : The Byzantine Generals Strike Again. J. Algorithms, vol. 3, no. 1, pp. 14-30, Mar 1982.
9. F. Christian, H. Aghili, and H.R. Strong. : Atomic Broadcast: From Simple Message Diffusion to Byzantine Agreement. Proc. Symp. Fault-Tolerant Computing, pp. 200-205, 1985.
10. P. Thambidurai and Y.-K. Park. : Interactive Consistency with Multiple Failure Modes. Proc. Symp. Reliable Distributed Systems, pp. 93-100, Oct. 1988.
11. F.J. Meyer and D.K. Pradhan. : Consensus with Dual Failure Modes. IEEE Trans. Parallel and Distributed Systems, vol. 2, no. 2, pp. 214-222, Apr. 1991.
12. P. Lincoln and J. Rushby. : A Formally Verified Algorithm for Interactive Consistency Under a Hybrid Fault Model. Proc. Symp. Fault-Tolerant Computing, pp. 402-411, 1993.
13. O. Babaoglu and R. Drummond. : Street of Byzantium: Network Architectures for Fast Reliable Broadcasts. IEEE Trans. Software Eng., vol. 11, no. 6, pp. 546-554, June 1985.
14. K.Q. Yan and Y.H. Chin. : An Optimal Solution for Consensus Problem in an Unreliable Communication System. Proc. Int'l Conf. Parallel Processing, pp. 388-391, Aug.1988.
15. K.Q. Yan, Y.H. Chin, and S.C. Wang. : Optimal Agreement Protocol in Byzantine Faulty Processors and Faulty Links. IEEE Trans.Knowledge and Data Eng., vol. 4, no. 3, pp. 266-280, June 1992.