ISSN:0975-9646

An Effective Solution for the Byzantine Agreement Problem through Lamport- Shostak-Pease Algorithm.

K.Nagaraju [1] and S.V.Hemanth [2]

[1] *Dept. of Computer Technology, Kavikulguru Institute of Technology and Science, Ramtek, RTM Nagpur University, Nagpur,Maharashtra, India.*

[2] *Dept. of Computer Technology, Kavikulguru Institute of Technology and Science, Ramtek, RTM Nagpur University, Nagpur,Maharashtra, India.*

Abstract

Distributed computing systems consisting of several computers that do not share a memory or a clock, the computers communicate with each other by exchanging messages over a communication network. To achieve authenticity, the fault-tolerance scheme of the distributed computing system must be reconsidered. This type of problem is known as a Byzantine Agreement Problem (BAP). It requires all fault-free processors to agree on a common value, even if some components are fault. Consequently, there have been significant studies of this agreement problem in distributed systems. However, the traditional Byzantine Agreement protocols focus on running $k \geq 3m+1$ rounds of message exchange continuously to make each fault-free processor reach an agreement. In other way, since having a large number of messages results in a large protocol overhead, those protocols are inefficient and unreasonable, especially for some network environments which have large number of processors.,Byzantine Agreement protocol can collect, compare and replace the received values to find the authentic processors and replace the values sent by the fault processors. Where sites (or processors) often compete as well as cooperate to achieve a common goal, it is often required that sites reach mutual agreement. [1]

Keyword: *Distributed systems, Fault Tolerance, Byzantine Agreement Problem, Lamport- Shostak-Pease Algorithm (LSPA)*

1. INTRODUCTION

In this paper, authenticate communication in distributed systems. One important aspect is how the system effectively copes with failures. There exist many well-studied failures, like crash failure which failed component simply stops communicating messages. Yet some other failures may send out conflicting messages to different system components. Lamport et al. investigated this failure, and they model it as the Byzantine Generals Problem, which made this special type of failure well known as the "Byzantine failure'' model. Their simple conclusion is that, using only oral messages (which implied the message is forgeable), the problem is solvable if more than two thirds of the generals are loyal. With unforgivable (written) messages, the problem is solvable for any number of generals and possible faulters. These results indeed are very interesting and let us see how the authors unfold this story little by little. The authors motivated the problem as the decision making process of the Byzantine Generals, among who may exists faulters. They made goals for the Byzantine Generals that:1. All loyal generals decide upon the same plan of action; 2. A small number of faulters cannot cause the loyal generals to adopt a bad plan. These two reasonable goals say about the outcome of the decision, yet they are hard to formalize (especially goal 2). Instead, authors took another angle by considering how the actions (decisions) were taken. Given all the messages the generals communicate: v_1, v_2,... ,v_k, our goal is to find a combining scheme to generate a single plan of action out of all these values. Applying this strategy to convert our goals into a formal definition.[2]

2. TAXONOMY OF PROBLEMS

All non-faulty processors must agree on value(s) from a non-faulty processor

Byzantine agreement:

The source processor broadcasts its initial value to all other processes.

Agreement: All nonfaulty processors agree on the same value.

Validity: If the source processor is nonfaulty, the common agreed upon value by all nonfaulty processors should be the initial value of the source

Consensus:

Every processor broadcast the initial value to all other processors.

Agreement: All nonfaulty processors agree on the same value.

Validity: If the initial value of every nonfaulty processor is v, then the agreed upon common value by all nonfaulty processors must be v.

Interactive Consistency:

Every processor broadcasts its initial value to all other processors.

Agreement: All nonfaulty processors agree on the same vector. (v_1, v_2, ..., v_k).

Validity: If the ith processor is nonfaulty and its initial value is v_i, then the ith value to be agreed on by all nonfaulty processors must be v_i.[3]

3. LAMPORT-SHOSTAK-PEASE ALGORITHM

The authors provided the *Oral Message algorithm (OM)* for this scenario. Given one commander and n−1 lieutenants,

for all nonnegative integers m, we have recursive algorithms

Algorithm OM(0).

(1) The commander sends his value to every lieutenant.
(2) Each lieutenant uses the value he receives from the commander, or uses the value RETREAT if he receives no value.

Algorithm OM(m), m>0.

(1) The commander sends his value to every lieutenant.
(2) For each i, let v_i be the value lieutenant i receives from the commander, or else be RETREAT if he receives no value. Lrutenant i acts as the commander in Algorithm OM(m-1) to send the value v_i to each of the n-2 other lieutenants.
(3) For each i, and each j ≠ i, let v_j be the value lieutenant i received from lieutenant j in step (2), or else RETREAT if he received no such value. Lieutenant i uses the value majority (v_1, v_2,v_3, ..., v_{k-1}).[4]

4. RESULTS

Impossibility Result:

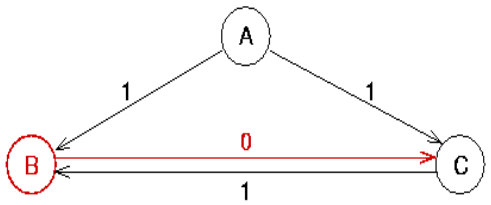
Byzantine agreement is impossible

if $m >$ ë(k-1)/3 û

e.g., ë (3-1)/3 û = 0

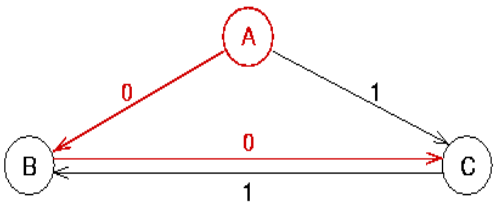
Byzantine agreement is impossible with < (m+1) message exchanges

LSPA algorithms for solving the Byzantine agreement problem that falls within these bounds. However, we will also see that the algorithms are fairly complex. This should naturally lead one to think twice when designing a system, to see if there is a way to avoid creating situations that require agreement. See the following simple example with 3 processors, from text. The arrows indicate state information made available to other nodes. In the first case, processor A initiates the agreement protocol and processor B is maliciously faulty.



C sees that B has decided for 0 and A has decided for 1. To satisfy the Byzantine agreement problem, C must decide for 1, since A is not faulty and A has decided for 1. This implies that the algorithm followed by C (and hence by any non-faulty non-initiating processor) must break ties in favor of the initiating processor.

The next case is where the processor A is a faulter, and reports different values to B and C.



B thinks A has decided for 0 and C thinks A has decided for 1. If the algorithm breaks ties in favor of the initiator, C must decide for 1. However, B must follow the same algorithm, and so it must decide for 0. This means we have no agreement among the two nonfaulty processors.

Proof of the full theorem generalizes this reasoning to a larger number of processors.[5]

Possibility Results:

Byzantine Agreement Conditions

1. Agreement: All loyal generals agree on the same value.
2. Validity: If the commander is loyal, then the common agreed upon value for all loyal lieutenants is the initial one given by the commander.

Agreement Theorem

Theorem: For any m, $OM(m)$ satisfies the Validity and Agreement Conditions if there are more than $3m$ generals and at most m of them are faulters.

Proof:

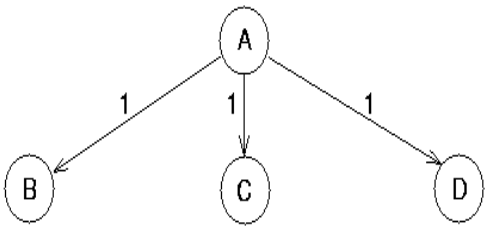
The proof is by induction on m, similar to that of the Validity Lemma. As a basis for the induction, we consider the case of $OM(0)$. If there are no faulters, it is easy to see that $OM(0)$ satisfies the Validity and Agreement Conditions. We therefore can assume the theorem is true for $OM(m\text{-}1)$ and prove that it is true for $OM(m)$, $m > 0$.

For the induction step, have $m \geq 1$. We consider two cases, depending on whether the commander is a faulter.

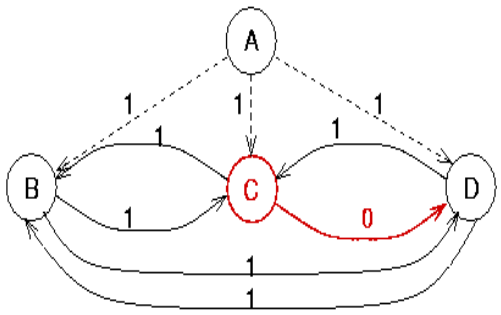
1. Suppose the commander i is loyal. By taking k equal to m in the Validity Lemma, we see that $OM(m)$ satisfies the Validity Condition. Moreover, since we are assuming the commander is loyal the Agreement condition is also satisfied.
2. Suppose the commander is a faulter. At most m-1 of the lieutenants can be faulters. Since there are more than $3m$ processes, there are more than $3m\text{-}1 > 3(m\text{-}1)$ processes in k-$\{i\}$. We may therefore apply the induction hypothesis to conclude that $OM(m\text{-}1)$ satisfies the Agreement and Validity Conditions. Hence, any two loyal lieutenants get the same vector of values $v_1, v_2, \ldots, V_{k-1}$, and therefore obtain the same value $majority(v_1, v_2, \ldots, Vk_{-1})$ in Step 3, proving the Agreement Condition.

For an example, for 4 processors, interactively

Four Processor Examples: Nonfaulty Commander

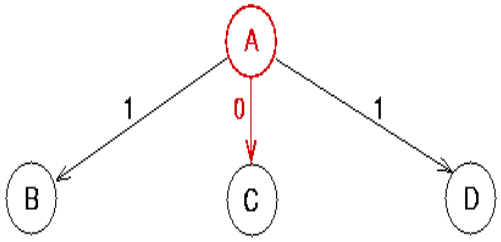


Round 1: processor A executes OM(1), where processor C (in red) is faulty.

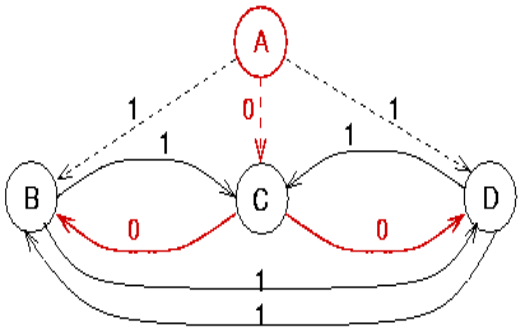


Round 2: processors B, C, and D execute OM(0). Dashed lines indicate messages sent during the previous round.

Three Processor Examples: Faulty Commander



Round 1: processor A executes OM(1), where processor A is faulty.



Round 2: processors B, C, and D execute OM(0).[6].

5. CONCLUSION

Byzantine Agreement is important both in the theory and practice of distributed computing. However, protocols to reach Byzantine Agreement are usually expensive both in the time required as well as in the number of messages exchanged. In this paper, we present a self-adjusting approach to the problem. The Mostly Byzantine Agreement is proposed as a more restrictive agreement problem that requires that in the consecutive attempts to reach agreement, the number of disagreements (i.e., failures to reach Byzantine Agreement) is finite. For t faulty processes, we give an algorithm that has at most disagreements for $4t$ or more processes. Another algorithm is given for $k \geq t+1$ processes with the number of disagreements below $t^2/2$. Both algorithms use $O(n^3)$ message bits for binary value agreement.[7].

ACKNOWLEDGEMENTS

We would like to thank all our supporters for kind support.

REFERENCES

[1] Mukesh Singhal and Niranjan G.Sivaratri."Advanced Concepts in Operating Systems",p.178-194.

[2] Lamport, L.; Shostak, R.; Pease, M. (1982). "The Byzantine Generals Problem" (PDF). ACM Transactions on Programming Languages and Systems. 4 (3): 382–401. Doi:10.1145/357172.357176. Archived from the original (PDF) on 7 February 2017.

[3] Kirrmann, Hubert (n.d.). "Fault Tolerant Computing in Industrial Automation" (PDF). Switzerland: ABB Research Center. p. 94. Retrieved2015-03-02.

[4] Mukesh Singhal and Niranjan G.Sivaratri."Advanced Concepts in Operating Systems",p.178-194.

[5] Driscoll, K.; Hall, B.; Paulitsch, M.; Zumsteg, P.; Sivencrona, H. (2004). "The Real Byzantine Generals": 6.D.4–61–11.doi:10.1109/DASC.2004.1390734.

[6] Walter, C.; Ellis, P.; LaValley, B. (2005). "The Reliable Platform Service: A Property-Based Fault Tolerant ServiceArchitecture":34–43.doi:10.1109/HASE.2005.23.

[7] Feldman,P.;Micali,S.(1997). "Anoptimalprobabilistic rotocolforsynchronousByzantineagreement" (PDF). SIAMJ. Comput. **26** (4):873–933.doi:10.1137/s0097539790187084.

AUTHORS

Mr. K.Nagaraju pursed B.Tech and M.Tech Computer Science and Engineering from Jawaharlal Nehru TechnologicalUniversity Hyderabad India in 2005, 2010. And currently working as Assistant Professor in Department of Computer Technology, RTM Nagpur University of Nagpur Maharastra India 2013. He has published more than 05 research papers in reputed international journals His main research work focuses on Cryptography Algorithms, Network Security, Cloud Security and Privacy,Data Mining, Distributed ,Parallel Computing and Computational Intelligence based education.He has 12 years of teaching experience.

S.V.Hemanth pursed B.Tech and M.Tech Computer Science and Engineering from JawaharlalNehruechnologicalUniversity Hyderabad India in 2006, 2010. And currently working as Assistant Professor in Departmentof Computer Technology, RTM Nagpur University of Nagpur Maharastra India 2013. He has published more than 06 research papers in reputed international journals His main research work focuses on Cryptography Algorithms, Network Security, Cloud Security and Privacy,Data Mining, Parallel Computing and Computational Intelligence based education . He has 11 years of teaching experience.