ISSN:0975-9646

Taxonomy of Distributed Consensus Algorithms

Suman Kumari
M.Tech. Scholar
Department of Computer Science and Applications
Ch. Devi Lal University, Sirsa- 125055, Haryana (India)

Dr. Harish Rohil
Asst. Professor
Department of Computer Science and Applications
Ch. Devi Lal University, Sirsa- 125055, Haryana (India)

Abstract – In distributed database computing transactions, group agreements of all the participating nodes is the crucial task. In such transactions, distributed algorithms are used for solving the problem of consensus. Distributed consensus algorithms are mainly of two types one is used in the case of link failures and second for process failures. For process failures, the algorithms used for solving the consensus problem are stopping, byzantine failures and commit protocols. The 2-phase commits (2-PC) and three-phase commit protocols (3-PC) are distributed algorithms that make all nodes in a distributed system agree to either commit or abort a transaction. The algorithms like FloodSet, EIGStop, EIGByz, PAXOS and E3PC are also used in last few years for solving this problem. This paper presents taxonomy of Distributed Consensus Algorithms. Different types of distributed algorithms for solving consensus problem in distributed database computing transactions are discussed here.

Keywords- Distributed Consensus Algorithms; 2-phase commit protocols; 3-phase commit protocols; stopping failures; byzantine failures.

1. INTRODUCTION

Distributed algorithms cover a wide range of application in distributed computing environment. In which we include telecommunication, distributed information processing, scientific computing and real time process control. Distributed algorithms are designed to nun on hardware consisting of many interconnected processors. Distributed algorithms used for distributed computing in which problem of consensus (agreement) arise [12, 16]. For solving the problem of consensus we use the different types of distributed consensus algorithms, because in the distributed computing it is very important to know that whether a particular transaction is to be committed or aborted. For this purpose different types of consensus algorithms are used [3] .The main objective of this paper is to describe the distributed consensus algorithms for solving the problem of consensus in distributed computing. For this purpose distributed consensus algorithms are used. Taxonomy of Distributed consensus algorithms divided into two main parts: Distributed consensus algorithms for link/communication failures and distributed consensus algorithms for process failures [8]. Distributed consensus algorithms for link/communication failures include coordinated attack problems and randomized coordinated attacks problem. In the process failures algorithms for stopping failures, algorithms for byzantine failures and commit protocols are included [6, 17].

2. DISTRIBUTED CONSENSUS ALGORITHMS

Distributed Consensus algorithms are used for solving the problem of agreement in the distributed computing. Consensus is a problem while computing in the distributed environment [3, 15].

Consensus: Consensus is the task of getting all processes in a group to agree on some specific value based on the votes of each processes [2, 3].
Some examples where consensus is necessary are as follows.

a) Deciding to commit or abort for distributed transactions.
b) Electing a leader (Mutual exclusion).
c) Distributed, fault-tolerant logging with consistent sequencing.
d) Synchronizing state machine.

In Distributed Consensus Algorithms two types of models are used.
a) Timing Model
b) Failure Model

In the timing model is divided into three parts: Synchronous Model, Asynchronous model and partial synchronous model. For the Synchronous model design of distributed consensus algorithms is easy. But for the asynchronous model it is very complex task to design the distributed consensus algorithms [3, 17]. In the failure model two types of failures included: Link/Communication failures and process failures.
Consensus protocol must satisfy the following for properties.
a) Termination
b) Validity
c) Integrity
d) Agreement

3. TAXONOMY OF DISTRIBUTED CONSENSUS ALGORIHMS

Distributed algorithms used for solving various types of problem. The taxonomy of distributed consensus algorithms are used for solving the agreement problem in the distributed computing. Taxonomy of distributed consensus algorithms shown in figure 1. This figure represents different types of distributed consensus algorithms used for solving the agreement problem. These algorithms are discussed in details in this paper.

3.1 Distributed Consensus algorithms for Link/Communications failures: In this failure model loss of message is accepted. When there are no failures of system components, distributed computing problems are usually easy to solve, using a simple exchange of message. In the link failures we include two types of failures [17].
a) Coordinated Attack Problem
b) Randomized Coordinated Attack problem

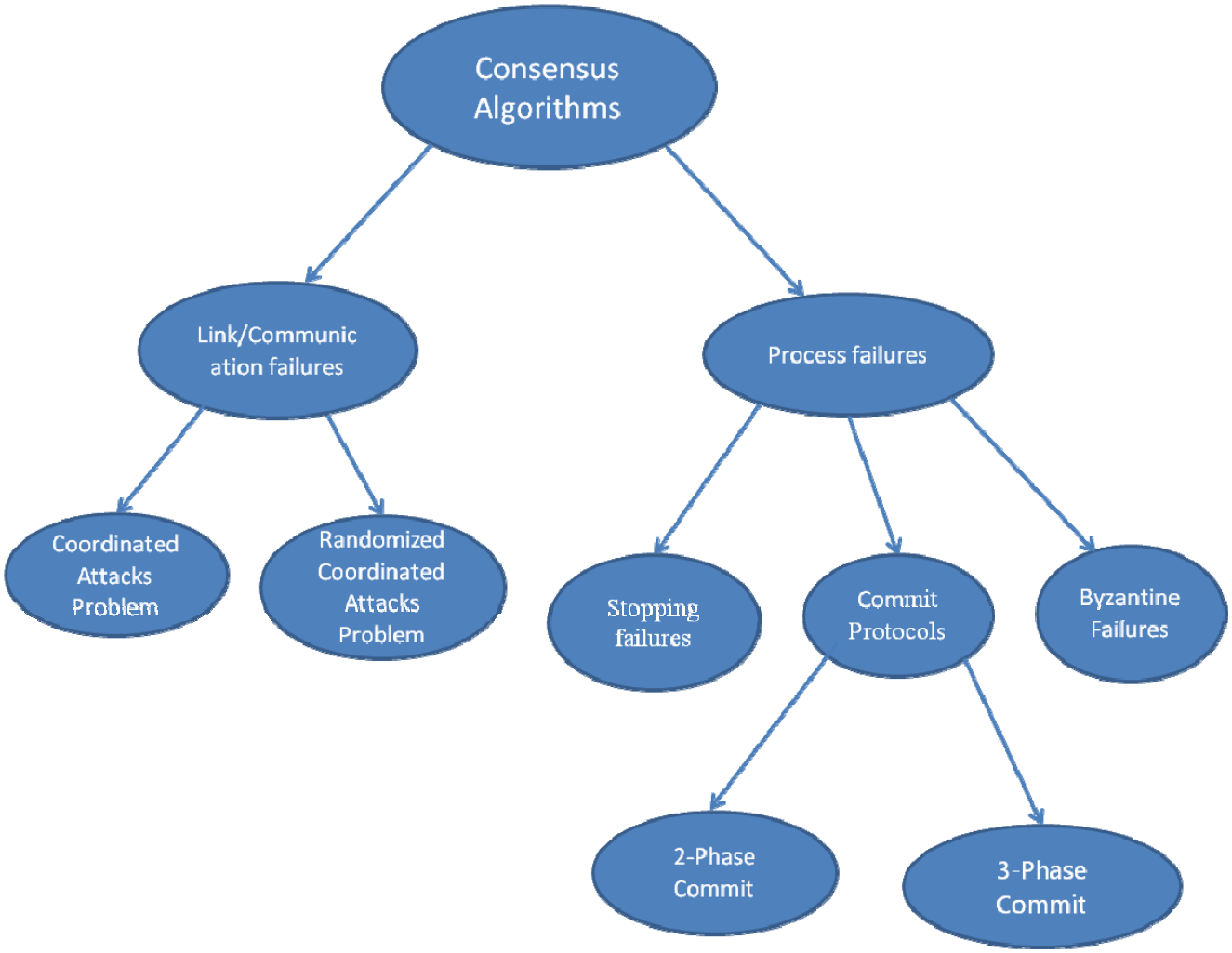

Figure 1. Taxonomy of Distributed Consensus Algorithms.

3.1.1 Coordinated Attack Problem: In coordinated attack problem each process starts with some value of a type. Even though the inputs can be arbitrary all process must output the same value. Process must agree on possible outputs for each pattern of inputs through a validity condition. It is easy to solve in a synchronous network with no failures. In the coordinated attack problem does not solve the problem of agreement satisfactorily.

3.1.2 Randomized Coordinated Attack Problem: In randomized coordinated attack problem is the advanced version of the coordinated attack problem. In the coordinated attack problem does mot work properly in the link failures. In the failure model in which the message may be lost, this approach doesn't work. In fact, there exist no algorithms that always solve the agreement problem properly.

3.2 Distributed Consensus Algorithms for Process Failures: In this failure model we use the synchronous system in which we allow the possibility that a number of processes might fail and the link are reliable –it means that no message loss is accepted. The process failures are further described as follows [15, 17].

a) Stopping failures
b) Commit protocols
c) Byzantine failures

3.2.1 Algorithms for Stopping Failures: In the stopping failures model a process may stop in the middle without warning. At any point of time during the execution of the algorithms simply process may stop. FloodSet and EIGStop algorithms are used for solving the problem of consensus. But the complexity of these algorithms is very high [3, 17].

3.2.2 Commit Protocols: Commit protocols are used in the distributed database transaction for committing the transactions that is, either all the effect of a distributed transaction is persisting or none persisting The commit protocols used for solving the problem in which no message loss , but only process failures is accepted. Commit protocols are types [2,4, 9].

a) 2-Phase Commit
b) 3-Phase Commit

3.2.2.1 2-Phase Commit: 2-Phase Commit is a blocking protocol. The algorithms used for this protocol may blocked indefinitely .This means that the other processes competing for the resources locks held by the blocked processes will have to wait for the locks to be released. 2-PC is a blocking protocol is blocking in nature yet it is used in the distributed computing because of low complexity than the 3-PC. Algorithms used for solving the consensus problem take only two rounds [6, 7, 11].

3.2.2.2 3-Phase Commit: In 3 –Phase commit is a non-blocking protocol. The algorithms used for this protocol overcome the limitation of the 2-phase commit protocol but it does not work in the segmented network. The main disadvantage of this protocol is that complexity of the algorithms is very high than the 2-phase commit and it is not feasible to implement this in the real world applications. Its cost is very high. Due to the complexity of 3-PC algorithms for solving the consensus problem PAXOS algorithm and E3PC is used [10,13, 14].
But both these algorithms provide greater complexity for the implementation. So, we need to optimize the 2-PC and 3-PC, so that these two algorithms can be used for solving the consensus problem in the distributed database transactions [1, 2].

3.3 Byzantine Failure: In the Byzantine failures the system may behave in the unpredicted way (crash failures). In byzantine failures the network and computers may behave in unexpected way due to the hardware failures. EIGByz algorithm used for the solving the consensus problem in the process failures [5, 17].

4. CONCLUSION

In this paper, we discussed the various types of distributed consensus algorithms for solving the problem of consensus in the distributed computing and the types of distributed consensus algorithms with their sub-types. vConsensus is a problem of group agreement in the distributed computing in which distributed algorithms are used. For solving the consensus we discuss the FloodSet, EIGStop, EIGByz , 2-Phase Commit Algorithm, 3-Phase Commit algorithm,Paxos algorithm and E3PC .These all algorithms are used for solving the problem of group agreement. But the implementation of these algorithms is very difficult due to their higher complexity and cost. So, due to this we use the 2-PC yet this is blocking in nature. But 3-PC is very complex to use in the real world applications. So, in this paper we mention the need of optimization of 2-PC and 3-PC so that both can be used in the better way and can be used for the real world application with the low cost and complexity of these algorithms is also reduced.

REFERENCES

[1] Idit Keidar and Danny Dolev, “Increasing the resilience of atomic commit, at no additional cost”, Proceedings of the fourteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, San Jose, California, United States, May 1995, pp. 245-254.

[2] Jim Gray and Leslie Lamport, “Consensus on transaction commit”, ACM Transactions on system (TODS); Vol. 31, Issue 1, March 2006, pp. 133-160.

[3] Nancy Lynch, Distributed Algorithms, First Edition, Morgan Kaufmann Publication, April 1997.

[4] M. L. Liu, D. Agrawal, and A. El Abbadi, “The Performance of Two-phase Commit Protocols in the Presence of Site Failures”, Distributed and Parallel Databases, Springer Netherlands, Vol. 6, Issue 2, April 1998, pp. 157-182.

[5] George Samaras, Kathryn Britton, Andrew Citron, and C. Mohan, “Two-Phase Commit optimizations and Tradeoffs in the Commercial Environment”, Proceedings of the Ninth International Conference on Data Engineering, Washington, DC, USA, April 1993, pp. 520-529.

[6] Gopi K. Attaluri and Kenneth Salem, “The Presumed-Either Two-Phase Commit Protocol”, IEEE Technical Committee on Data Engineering, Vol. 17, Issue 1, March 1994, pp. 22-28.

[7] Monica Caniupan and Leopoldo Bertossi “Optimizing Repair Programs for Consistent Query Answering” Carleton University School of Computer Science Ottawa, Canada.

[8] 1C.P. Eswaran, J.N. Gray, R.A. Lorie, and I.L. Traiger “. The Notions of Consistency and Predicate Locks in a Database System” IBM Research Laboratory San Jose, California.

[9] Impossibility of distributed consensus with one faulty process M. Fisher, N. Lynch, and M. Patterson, Journal of the ACM, Vol. 32 No. 2 April (1985) 274-382

[10] Revisiting the PAXOS algorithm: Theoretical Computer Science, Volume 243, Issues 1–2, 28 July 2000, Pages 35-91Roberto De Prisco, Butler Lampson, Nancy Lynch

[11] Philip A. Bernstein, V. Hadzilacos, and Nathan Goodman. Concurrency Control and Recovery in Database Systems. Addison-Wesley, Reading, Massachusetts, 1987.

[12] Jorge Cortés Distributed algorithms for reaching consensus on general functions Department of Mechanical and Aerospace Engineering, University of California, San Diego, CA, 92093, USA Received 29 October 2006; received in revised form 30 June 2007; accepted 16 July 2007

[13] Tushar Chandra, Robert Griesemer, Joshua Redstone Paxos Made Live - An Engineering Perspective June 26, 2007

[14] Robbert van Renesse Cornell University Paxos Made Moderately Complex March 25, 011.

[15] H. Attiya, A. Bar Noy, and D. Dolev. Sharing memory robustly in message passing systems. Journal of the ACM, 42(1):121{132, 1995.

[16] Attiya H. and Welch J., Distributed Computing: Fundamentals, Simulations and Advanced Topics (2nd Edition), Wiley Interscience, 414 pages, 2004.

[17] Lynch, Nancy (1996). Distributed Algorithms. San Francisco, CA: Morgan Kaufmann Publishers. ISBN 978-1-55860-348-6.