International Journal of Advance Research in Science and Engineering
Volume No.07, Special Issue No.01, April 2018
www.ijarse.com
IJARSE
ISSN: 2319-8354

Huang's Process Termination Detection Algorithm and Its Resemblance with Interactive Consistency Problem

Rajeev Ranjan Kumar Tripathi

Buddha Institute of Technology, GIDA, Gorakhpur, Uttar Pradesh, (India)

ABSTRACT

In Distributed System multiple nodes, residing into different geographic region, work together to achieve a common goal. Distributed System has two major challenges: absence of shared memory and global clock. We cannot schedule any operation based on time in Distributed System. Message passing is always used to trigger an event as this exchange takes place in real time. Huang process termination detection is entirely based on message. Controlling Agent when receives all the sent messages back from participating processes, declares that a computation is over. Agreement Protocol (problem) in Distributed System is used when we have to come on an agreement. Agreement Protocol is broadly categorized into three ways: Byzantine Agreement Problem, Consensus Problem and Interactive Consistency Problem. This paper finds a close resemblance in between Huang's Process Termination Algorithm and Interactive Consistency Problem.

Keywords- Distributed System, Huang's Process Termination Detection Algorithm, Controlling Agent, Agreement Protocol, Byzantine Agreement Problem, Consensus Problem, and Interactive Consistency Problem.

I.INTRODUCTION

In Distributed System various machines are connected with each other and they work together to achieve a common goal. Computers may be spatially separated by any distance. Distributed System has following consequences:

- **Concurrency** In Distributed System, concurrency is a norm; one process is executing on one machine while other process is running on any other machine. Resource sharing is taking place among machines. On demand additional resources may be added in the system.
- **No Global Clock.** In Distributed System there is no notion of global clock. Machines and processes are coordinating with each other by message passing.
- **Independent Failures** Failure may take place any time in the Distributed System. It is the responsibility of designer to consider all the possible failures and to provide appropriate solutions so that in case a failure rest of machines may continue working.

Distributed System has following challenges:

1.1 Heterogeneity Distributed System may own heterogeneity in form of networking standards, computer hardware, operating system and programming languages. Different machines have different character sets and if a transaction of message is required in between the machine there should be a common data format. Either the machines may agree on a common data format or the sender sends data with some additional information so that the receiver may convert the sent data into its own character set with the help of this additional information. Generally Middleware is used in Distributed System to mask this heterogeneity.

1.2 Openness This property says that "Whether the system may be extended and re-implemented in other ways." Heterogeneity restricts the openness. Standards are always published and it is implemented by the designers of Distributed System.

1.3 Security Transactions of messages are taking place in between processes and machines. Sometimes executable codes are also part of this transaction. These executables and the mobile codes are a major challenge for security. A mobile code coming from another machine may violate the security policy of a host machine. Java is found more suitable for mobile codes as it provides a concept of Sandbox. Sandbox is a restricted execution environment for mobile codes. Java applets are well known mobile codes.

1.4 Scalability On demand resources may be added in the system. A system is said to be scalable if it will remain effective when there is a significant increase in the number of resources and users. Theoretically it sounds good but practically we have a performance bottleneck. It is very difficult to manage the increased resources and users. Performance bottleneck is a point from where a system starts degrading its overall performance.

1.5 Failure Handling Unexpected failures may also occur in the system about which designers have not paid attention. In this case we have to reduce the unknown problems into the known problems. We have already solutions of known problems and hence these solutions may be used to fix these unexpected failures.

1.6 Concurrency Though concurrency is a norm in Distributed System often conflicts arise in accessing shared resources at the same time. A schedule is always required to access those resources.

1.7 Transparency Resources may be on local machines or on remote machines. Transparency hides this fact from the processes and users that resources are not located on remote machines but these are on local machines [7].

Even in presence of above challenges Distributed System is providing an economical solution where we require more and more processing capabilities. Processes running on a single machine can be easily controlled as they share the same memory: processor may be same or different but machine is always same. In this case, on requirement, any process may be brought into memory and can be swept out of memory. Termination of a process means de-allocating the resources assigned to a process. In Distributed System a computation may use different processes located on different machine. Initiation of participating processes and declaration of "Computation is over now" is a typical assignment here. Huang Process Termination Detection uses messages to achieve this goal. Next section is describing the system model which is used in Huang's approach.

II. SYSTEM MODEL FOR HUANG'S PROCESS TERMINATION DETECTION ALGORITHM

Global clock is absent in Distributed System, different nodes may reside into different geographic location having different time zone. One can think about having a synchronized clock which may be used to trigger an action based on time. ***Clock drift rate*** restricts us to do so. This is the reason why messages are used to trigger an action into Distributed System. A process may be either in passive state (idle state) or in active state. A process may switch to any state at any time either by receiving a computation message (from passive state to active state) or by returning the computation message (from active state to passive state). In a computation if many processes are involved, we have a Controlling Agent which initiates the computation and deals with the computation messages. Controlling Agent sends messages to all participating processes. Initially all participating processes are in passive state; on reception of this computation message participating process changes its state from passive to active. A participating process Pi cannot permanently hold the computation message. When Controlling Agent collects all the sent messages back from participating processes, it declares that "Computation is over now." Every process is equipped with a variable called weight "(W)". If W is 0 then process is considered into passive state and when W>0 then process is considered into active state. Controlling Agent has initially its weight equal to 1. Controlling Agent splits its weight and sends a fraction of W as a computation message to the all participating processes. After completion of the assigned jobs every process returns this computation message back to the Controlling Agent. This system model has following assumptions:

- No participating process can hold the computation message infinitely.
- Communication channels are reliable, no message loss takes place.
- Participating process returns the same weight what it has received from Controlling Agent.
- At every time $\sum W=1$(including the all weights; weight of Controlling Agent, all the weights which are in transit state, weights of all participating processes)

Following notations are used in this algorithm.

B (DW): Computation message sent by Controlling Agent to participating process with weight DW.

C (DW): Computation message sent back by participating process to Controlling Agent with weight DW.

III.HUANG' PROCESS TERMINATION DETECTION ALGORITHM

To discuss the algorithm this paper is considering only two processes, which are residing on different machines in different time zones, are involved in a computation: say P1 and P2. Process P1 is the Controlling Agent and P2 is the participating process. Initially P1 has W=1.

Step1:

Process P1 splits its weight (W) as:

W=W1+W2

Where, W1>0 and W2>0.

P1 (W) =W1 (Controlling Agent keeps W1 with itself and sends W2 as DW to process P2). On reception of this W2, P2 becomes active as: P2 (W) =W1.

Step2:

After completion of the assigned task, P2 returns the weight (W2) to P1 as C (W2) and becomes idle.

P2 (W) =0 (Process P2 is idle now)

C (W2) is returned to the P1.

Step3:

The C (W2) is in transit state and we can sum the weight of P1 and C (W2) as:

P1 (W) +C (W2) =1

Step4: The computation message with weight C (W2) is received by P1 and P1 (W) becomes 1 as:

P1 (W) =P1 (W) +C (W2)

P1 (W) =1.

Process P1 declares that computation is over now. This paper skips the proof of correctness of this algorithm [6,7]. Next section is shedding light on Agreement Protocol.

IV. AGREEMENT PROTOCOL

Agreement Protocol is used to reach on an agreement. In this process, processors/processes participate and they can be categorized as faulty processor or non-faulty processor. In this paper processors and processes are interchangeable terms. Suppose we have three processes P1, P2 and P3. Process P1 is sending a value "1" as an agreement to process P2 and "0" as an agreement to process P3. Note that only one value must be sent by P1 to both P2 and P3, P1 will be referred as faulty processor. If a single value is broadcasted by a processor to the all processors of a system then and only then it will be referred as non-faulty processor. If we have 3 processors in a system and two processors are faulty then we cannot reach on an agreement. Let we have 'n' processors in a system and 'm' processors are faulty then we cannot reach on an agreement if ***m> (n-1)/3***. Next section is describing the system model used into Agreement Protocols [1-5].

V. SYSTEM MODEL FOR AGREEMENT PROTOCOL

Agreement Protocols are always studied under the following system model.

- There are 'n' processors in the system and at most 'm' processors may be faulty. To reach on an agreement m<=(n-1)/3.
- Each processor is logically fully connected and they communicate with each other by message passing.
- Communication channels are reliable i.e. no error takes place in channels.

- Every receiver knows the identity of sender of message.

Next section is categorizing the Agreement Protocols.

VI. CLASSIFICATION OF AGREEMENT PROTOCOL

Agreement Protocol (Problems) can be categorized into three ways: Byzantine Agreement Problem, Consensus Problem and Interactive Consistency Problem. In all the above three problems, all non faulty processors must reach on an agreement i.e. they are all agreed on a common value. Byzantine Agreement problem is considered as a base problem and using its solution we can derive the solution of other two agreement problems. In this section we will discuss about the three agreement problems.

6.1. Byzantine Agreement Problem

In this problem we randomly select a source processor which broadcasts its initial value to all the other processors of the system. Solution of Byzantine Agreement problem must meet with the following two objectives.

6.1.1. Agreement All non-faulty processors must agree on the same value.

6.1.2. Validity If the source processor is non-faulty then the value on which other processors are agreed is the value broadcasted by the source processor.

6.2. Consensus Problem

Every processor broadcasts its initial value to all the other processors. Initial value of each processor may be different. A protocol for reaching the consensus should meet the following conditions.

6.2.1. Agreement All non-faulty processors should agree on a common value.

6.2.2. Validity If the initial value of every non-faulty processor is 'V' then the agreed upon common value by all non-faulty processors must be 'V'.

6.3. Interactive Consistency Problem

Every processor broadcast its initial value to all other processers. The initial values of the processors may be different. A protocol for solving the interactive consistency problem should meet the following conditions.

6.3.1. Agreement All non-faulty processors agree upon the same vector (V1, V2, V3,…….,Vn).

6.3.2. Validity If the i^{th} processor is non-faulty and its initial value is Vi then the i^{th} value to be agreed on by all non-faulty processors must be Vi.

In Byzantine Agreement and Consensus agreement problem there is a single value on which agreement is made. In Interactive Consistency Problem there is no any agreement on a single value. Let Pi broadcasts its initial value as "1" then all other processors, say Pj will assume the initial value of Pi as "1". If Pj receives a value other than "1" from Pi, Pj declares Pi as a faulty processor. Interactive consistency Problem is a generalization of Byzantine and Consensus Problem [1, 2, 3].

VII. Resemblance of Huang's Process Termination Detection Algorithm and Interactive Consistency Problem

In termination detection, participating processes return the same weight what they have received earlier from Controlling Agent. Termination is declared over only if Controlling Agent gets all the sent weights back. If a single process has not returned its weight in a computation where 'n' processes are participating including Controlling Agent, the Controlling Agent has to wait for this message and computation cannot be declared over. However a maximum time limit can be set up to which a Controlling Agent should wait the returned message. If this time limit expires Controlling Agent may either re-initiate the computation or may abort the computation. The over all agreement in case of process termination is that "Whether computation will be declared successfully completed or it will be aborted and will be re-initiated later". Participating processes are considered as non-faulty processes only in the case when they return the computation message back to the Controlling Agent. Controlling Agent treats all the returned computation messages as the initial value of participating processes and based on this it takes decision.

VIII.CONCLUSION

Various available literatures conclude that "Clock synchronization and Atomic Commit in Distributed Database are the application of Agreement Protocol". This paper briefly discusses the Huang's Process Termination Algorithm and Agreement Protocol with their used system model and successfully finds a close resemblance in between the aforesaid. Further discussion can be also made on "execution of critical section problem, reservation protocol in medium access control ,content negotiation in web engineering and common data representation in Distributed System" have a resemblance with the Agreement Protocol. Agreement does not mean that participants are agreed on a common value, agreement may be on a schedule, common character set and common data format also.

REFERENCES

[1.] Strong R and Dolev D., "Byzantine Agreement", Proceedings of the Spring Compcon-83, March-1983.

[2.] Rabin M.,"Randomized Byzantine Generals", Proceeding of 24th Symposium on Foundatios of Computer Science,1983.

[3.] Dolev D., "The Byzantine Generals Strike Again", Journal of Algorithm, Januray 1982.

[4.] Babaoglu O., "On the Reliability of Consensus-Based Fault Tolerant Distributed Computing System", ACM transactions on Computer System, November,1987.

[5.] Srikant T.K. and Toueg S.,"Optimal Clock Synchronization", Journal of the ACM , January-1987.

[6.] "Advanced Concepts in Operating System" by M. Singhal and N.G. Shivaratri,McGraw Hill Education, Indian Edition,38th reprint 2015.

[7.] "Distributed Systems", by G. Coulris,J.Dollimore and T. Kindberg, Pearson, 4th Edition.