CHAPTER 2

LITERATURE REVIEW

Chapter 2

Literature Review

2.1. Introduction

In this chapter we carry out literature review of previous works carried out on the subject. We also explain the difficulties in clock synchronization followed by a brief on some of the methods of clock synchronization.

2.2. Basics of Clock Synchronization

As discussed, earlier clock synchronization is the process of ensuring all clocks in a network to have a common knowledge of time. Therefore, clock synchronization is an algorithm, which ensures clock synchrony. Developing this type of algorithm may become challenging because of some of the factors like:

(i) Every clock tends to drift and they drift with different rate. So, these clocks are required to be synchronized repeatedly.

(ii) These clocks not only have different drift rate in comparison to one another but their own drift rate is also changing with time.

(iii) Due to transmission delay, no clock can have the knowledge of all the other clocks instantaneously.

(iv) In a distributed network it is but natural to have some faulty clocks, which add to the challenge of clock synchronization. A faulty clock of arbitrary type, malicious clock, is the most difficult fault to handle.

A clock synchronization process should able to effectively take care of the difficulties mentioned above to ensure all clocks have a common knowledge of time.

There are various methods of clock synchronization available. Some of the important types of clock synchronization are under numerated:

(i) External Clock Synchronization and Internal Clock Synchronization – External clock synchronization is the type of clock synchronization which achieve clock synchrony with reference to external clock, usually a real time clock. The external time source may a universal coordinated time, a GPS source etc. The **Network Time Protocol** (**NTP**) [J. Jasperneite, 2004] is an example of external clock synchronization protocol. In Internal clock synchronization there no external reference clocks available. The Clock synchronization is achieved by a process, which ensures that the time difference between the clocks is minimum.

(ii) Hardware Clock Synchronization and Software Clock Synchronization – Hardware clock synchronization [12 (SR 87)] uses synchronization hardware at each node and require a separate network solely for synchronization. This process gives a very tight synchronization. Software clock synchronization does not use any special hardware for attaining synchronization. In the method synchronization messages are pass among the nodes using the existing network. The synchronization obtained may be as tight as given by the hardware clock synchronization.

(iii) Software clock synchronization can be of three types namely

(a) Deterministic Clock Synchronization

(b) Probabilistic Clock Synchronization

(c) Statistical Clock Synchronization

(iv) Deterministic Clock Synchronization - In this method it is assumed that there is a upper bound on the message transmission delay. This type of clock synchronization gives an upper bound of the precision obtained by the method.

(v) Probabilistic Clock Synchronization – Here there is no assumption on the upper bound on the message delay. The method guarantees that the synchronized clocks are within a maximum bound but this guarantee has a non-zero probability if there are faulty clocks in the network.

(vi) Statistical Clock Synchronization - In this case it is assumed that the expectation and the standard deviation of the delay distribution is known but the there is no guarantee that the synchronized clocks are within a maximum bound but a statistical reasoning is given that clocks are within a maximum bound with certain probability.

In the thesis, we have taken an internal software deterministic clock synchronization problem. This type of synchronization can be achieved by using two different type of convergence functions namely Averaging function and Non-Averaging function. Convergence Averaging function uses a set of remote clocks values to compute a new corrected clock value. It uses some kind of averaging method to arrive to the corrected value. The Convergence Non-Averaging method uses the fact that it has received certain fixed amount of remote clocks values to compute the new clock value.

2.3. Convergence Averaging methods

In this sub-section we give some of the previous works done in the field of convergence averaging method. As the name suggest this method use some kind of convergence function to obtain the new corrected clock value. Some of the popular and pioneer work as discussed below.

The first paper we would like to discuss is 'Synchronizing Clocks in the Presence of Faults [L. Lamport and P. M. Melliar-Smith, 1985]' by Leslie Lamport and P M Melliar-Smith. This paper is one of the initial works done on fault tolerant clock synchronization in distributed network. The sole aim of the paper is to present solutions to maintain clocks synchronization once the clocks are initially synchronized. The authors have proposed three different algorithms toward this end. The following are the algorithms:

(i) Interactive Convergence Algorithm

(ii) Interactive Consistency Algorithm

(iii) Clock synchronization Algorithm

The first algorithm, Interactive Convergence Algorithm, uses a convergence averaging function called 'egocentric convergence function'. The other two algorithms use convergence non-averaging algorithm.

The interactive convergence algorithm considers three assumptions for the design. The following are the assumptions considered:

(i) All good clocks in the network are initially synchronized.

(ii) All good clocks run at approximately the same rate.

(iii) A good clock can read the time difference between any clock and itself with a small error.

The idea of the algorithm is fairly straight forward. All the clocks share their clocks with one another. The clocks then calculate the average of these clocks and set the value as their new clock. While calculating the average the clocks check whether the remote clock is within a predefined bound, δ of itself. If it is within δ then the remote clock value is taken for the average calculation otherwise the clock replace the value of remote clock with that of its own. This averaging method is called 'Egocentric Averaging'. The algorithm is a fault tolerant clock synchronization algorithm. The algorithm works even if just less than one third of the clocks are malicious. The algorithm gives a precision of:

$$\pi = (6f + 2)\epsilon + (3f + 1)\rho T,$$

where f is the number of faulty nodes, ρ is the drift rate and T is the resynchronization time period.

The advantage of the algorithm is that the algorithm neither attempt to find the faulty clock nor reject the faulty clock. This led to reduced amount of complexity of clock correction. However, this algorithm has one prominent disadvantage namely *clustering* – the value of δ has to be chosen carefully. If the value of δ is too small, then the slow clock start synchronizing with only the slower clocks and faster clocks start synchronizing only with faster clocks. If the value of δ is chosen too large then the precision obtain is not tight.

The second paper, which we are discussing, is 'A New Fault-Tolerant Algorithm for Clock Synchronization [Jennifer Lundelius Welch and Nancy Lynch, 1988]'. The author of the paper is Jennifer Lundelius Welch and Nancy Lynch. They proposed an algorithm that maintained the synchronization of the clocks of the logical clock assuming that the logical clocks in the network are initially synchronized. The algorithm is fault tolerant and can tolerate just under one third of total clocks being faulty. This algorithm works on the following assumptions:

(i) The drift rate of the logical clocks is within a bound, ρ. ρ is a small fixed quantity which signifies maximum drift rate of a good clock.

(ii) The number of faulty clocks *f* is fixed and satisfy the condition

$$\mathbf{n \geq 3f+1},$$

where n is the total number of clocks in the network.

(iii) If α is a constant then the logical clocks are within α of each other at the start of the algorithm.

The proposed method uses a convergence algorithm to maintain clock synchronization. The convergence algorithm used by the algorithm is called Fault Tolerant Mid-point

Convergence function. This algorithm functions in the following way: Initially all the clocks were synchronized at the start of the algorithm. All the clocks share their values with one another. These values are stored and sorted out by each of the clocks. The highest f and the lowest f are then discarded and average of the remaining is taken. The averaged value is the new clock value. This algorithm is designed with a basic assumption that all the faulty clocks are either running too fast or too slow and hence the good clocks are general within these values. The precision offered by the algorithm is

$$\pi = 5\varepsilon + 4\rho\delta + 4\rho T,$$

where ε is the upper bound of message delay, ρ is the maximum allowable drift rate of good clocks, δ is the maximum deviation between two good clocks and T is the resynchronization time period. The disadvantage of the algorithm is that the faulty clocks especially malicious clocks can be anywhere which go against the basic idea of the algorithm.

Subsequently in the paper, the authors proposed further refinement in their algorithms. They proposed a clock synchronization algorithm were new nodes can join the network and the requirement of initial synchronization has been relaxed.

Thirdly, we would like to deliberate on a paper written by M J Pfluegl and D M Blough called 'A New and Improved Algorithm for Fault-Tolerant Clock synchronization [Pfluegl and D. M. Blough , 1995]'. This algorithm uses a convergence function called Sliding Window Algorithm (SWA) convergence function. As the name suggests the algorithm consist of two main parts, in the first section a sliding window protocol is executed to identify a particular window and secondly this window is used to obtain the new corrected clock value. The authors have suggested two algorithms in the paper namely (i) SWA_{mean}^{det} and (ii) SWA_{median}.

Before going further with the description of the algorithms let us first discussed the assumptions made while designing the algorithms:

(i) The drift rate of the logical clocks are within a bound, ρ. ρ is a small fixed quantity which signifies maximum drift rate of a good clock

(ii) The number of faulty clocks f is fixed and satisfy the condition

$$\mathbf{n \geq 4f}$$

(iii) All the nodes execute the same clock synchronization algorithm.

The algorithm is a fault tolerant algorithm and it works in presence of various types of fault provided they are within the upper bound of fault tolerance.

Now coming to the first algorithm SWA_{mean}^{det}, first all the clocks shared their clock values with one another. These clock values are sorted out and aligned on a timeline. These clocks values can be visualized as points on the timeline. Then this is passed on to the convergence function. The convergence function carries out a sliding window protocol to select a window. The size of the window is taken to be *n-f.* The sliding window start from the left most clock value on the timeline. When the window finds a clock value on the left edge of the window, it is considering a single instance. This window instance will have some clock values. At every instance the number of the clock value are noted and the window slide ahead by until it finds the next clock value on its left edge. When it finds the next clock value on its left edge, it considers that instance as the second instance and note down the number of clock value in that window. This window slides until the right edge of the window contain exactly the rightmost clock value. There will be n-f+1 instance. The algorithm selects the window instance which contain the largest number of clock values for further computation. If more than single window instances are present than the left most instance is chosen. Then the mean of the window is taken as the new corrected clock value. The precision offered by this algorithm is

$$\Pi \leq \varepsilon + f\,\frac{(\delta+\varepsilon)}{n-f} + f\,\frac{(2\delta+\varepsilon)}{n-f+1}\,,$$

where δ is the maximum deviation allowed and ε is the upper bound on the message delay.

SWA_{median} also chose the window which contain largest number of clock value after conducting a sliding window protocol in the same way as describe above. The difference is when the there are multiple window instances with largest number of clock values, it chose the one with the minimum variance for further calculation. The new clock is the median of the selected window instance.

The advantages offered by these algorithms are numerated below:

(i) The algorithms can tolerate faulty nodes of more than 50% of the total nodes even though the malicious tolerated if just below one-fourth of the nodes.

(ii) The tightness obtained by the algorithm degrades gracefully in presence of when the number of faulty clocks increases.

2.4. Convergence Non-Averaging methods

The convergence non averaging methods unlike the averaging use the fact that some fixed number of remote clock values are receive in order to derive the new corrected clock value. One of the most prominent algorithms are 'Optimal Clock Synchronization [T K Srikanth and Sam Toueg, 1987]' authored by T K Srikanth and Sam Toueg. They claim that optimal precision can be achieved using their algorithm.

2.5. Some of the Latest Synchronization Methods

Time and message format for synchronization are defined in IEEE 1588 standards [IEEE Std. 1588-2002, 2002; IEEE Std. 1588-2008,2008]. In the IEEE standards a grand Master node propagates periodically synchronization information to the slave nodes over the boundary clock and the transparent clocks. The synchronization information may also be carried over either by boundary clock or transparent clock only. There are multiple Precision Time Protocol (PTP) ports in a boundary clock. The boundary clock can also act as master clock which can synchronize other clocks, as slave clocks. Here in algorithm like this we have Grand Master and others clocks called Slave clocks. The Grand Master will share their clock value with their neighbouring slave clocks and the slave clocks will assigned itself the clock value and hence are now in synchrony with thw Grand Master, then, these slave clock values are share with oether slave clocks and they in turn get synchronized. Hence those initial slave clocks act like a Master Clocks.

In due course of time, there have been lots of research work and many researchers had developed many Precised Clock Sync algorithms. Many of these algorithms are IEEE 1588 Compliant. Some of them had also demonstrated their algorithm [D. Köhler, 2007; R. Holler, et al, 2003; T. Neagoe, et al, 2006; T. Cooklev, et al, 2007;] J. Kannisto, , et al, 2004;]. These suggested algorithms mainly cater for single hop network. Ability of these algorithms working in multi hop distributed network is suspect since for such multi hop environment, the requirement for high accuracy is mandatory. The clock synchronization error increases as number of hops increases in a distributed network. This is a major limitation in boundary clock synchronization scheme. Peak – to – peak clock synchronization error increases to 0.8, 08, 40 & 300 microsec for 01, 03, 05 & 10 hops respectively for such scheme. Simulation for the same is reported in [J. Jasperneite, 2004]. Many researchers had proposed various clock synchronization algorithm catering for various requirement including that of IoT [11] – [16]. Maintaining time records of information sent and received is a basic function of synchronization algorithm. Timestamp for each for such bits of information is always enclosed with the information sent/received. The slave clock is able to synchronize by calculating various clock parameters like clock offset, compensate time etc.

2.6. Mathematical Tools and Simulation

In order to understand various mathematical tools and techniques, we have done literature reviews of many papers and books. After going through these literatures, we have come to the understanding that some area of applied mathematics will be useful in optimization of our algorithms. The area of Operation Research and mathematical optimization particular interest us since it covers various tools like convex optimization, stochastic process, Markov chain, Random walk model and simulation. Cree S. Dawson in his papers Operations Research at Bell Laboratories Through the 1970s: Part I, Part II & Part III, clearly brought out the journey of Operation Research from early 1940s to present. He also talks about importance of the subject and its various applications. Bouchekara in his paper "Most Valuable Player Algorithm: a novel optimization algorithm inspired from sport" illustrated the modern applications of Operation Research and bring out a powerful lesson of the impact of the subject in the present era and particularly in sports, even though the application is always open to one's requirements and imagination.

Simulation involves creation of a model according to our problem at hand. For creation of the model, we should first formulate the problem for which we are looking for a solution. We then start analysing and defining the problem. This and various other aspects of simulations are dwell in details by Richard E. Nance in his paper "Perspectives on the evolution of simulation". Two Papers on Applications of Stochastic Processes to Road Traffic Problems: Delays on a Two-Lane Road is a paper on application of Stochastics Process by Tanner, J. C. Some of the most daily application of applied mathematics and especially in the field of Operation Research is observed in this paper. The author describe how road traffic can be optimally managed by utilizing the knowledge of Stochastic process.

Carsten Chong in his paper Stochastic PDEs with heavy-tailed noise published in Stochastic Processes and their Applications journal talks about analysis of heavy tailed noise using stochastic process.

We have also studied Markov chain/process during the course of our research, one of the most interesting paper on the subject is “Basket Credit Derivative Pricing in a Markov Chain Model with Interacting Intensities” by Kangquan Zhi published in the journal Mathematical Problems in Engineering. Some of the paper studied in the field of Random Walk modelling are apart from the regular text books are “Random walks and percolation: an analysis of current research on modeling natural processes” by Aaron Zwiebach, Massachusetts Institute of Technology, Department of Mathematics, Cambridge, Massachusetts, United States and Random Walks: A Review of Algorithms and Applications by Feng Xia & et al in **“IEEE Transactions on Emerging Topics in Computational Intelligence”**. In both the paper various aspects of random walk starting from basic concept to advance application are discussed.