CHAPTER 1

INTRODUCTION

Chapter 1

Introduction

1.1. Background

A distributed network will consist of nodes which are spatially dispersed and the nodes may be heterogeneous. These nodes are required to carry out collective functions in close coordination to one another. This is possible only when they share some common notion of time. This ensures temporal precedence order across the network. The distributed nodes may function independently or they may be required to work with other nodes to perform some collective tasks. If the nodes are functioning independent of the other nodes, the nodes may not require coordinating among themselves. When these nodes are required to carry out common functions then they have to maintain close coordination with one another. This is possible only when they share some common notion of time. This ensures temporal precedence order across the network. This means these nodes need to have at least a clock from where they can access the time. This clock can be a common clock, master clock or reference clock, from where every node get time through some mechanism. Another possibility is all these nodes maintain their individual clock, called local clock, from where the node can get the time. A third possibility is that there is both a common clock and local clock. In our work we have considered the second scenario as this set up is usually observed and perhaps the most challenging in terms of clock synchronization.

In general time is maintained by an instrument which ticks periodically at every fixed small interval of time. This instrument is the physical clock. The physical clocks can be of various types and come with wide range of specifications for their intended purposed. Quartz clocks are generally used as local clock in different network components. As introduced by Newton, real time exists and progresses at a consistent pace throughout the Universe. A perfect physical clock can maintain a consistent time without any deviation i.e. it can keep real time. Every physical clock designed fails to do this because of the physical

properties of the material along with external factors like temperature, pressure and aging. This means every physical clock value deviate from the real time. This is also true for quartz clocks also.

The deviation of a clock with respect to the perfect clock is the drift of the clock. The rate at which the clock is drifting is the drift rate, ρ. Mathematically drift rate can be represented as $(dC(t))/dt = \rho$ where t is the real time. For perfect clock $(dC(t))/dt = 1$. Different physical clocks have different drift rates and hence all clocks drift with respect to one another. Figure 1.1 depicts the notion of perfect, fast and slow clocks.

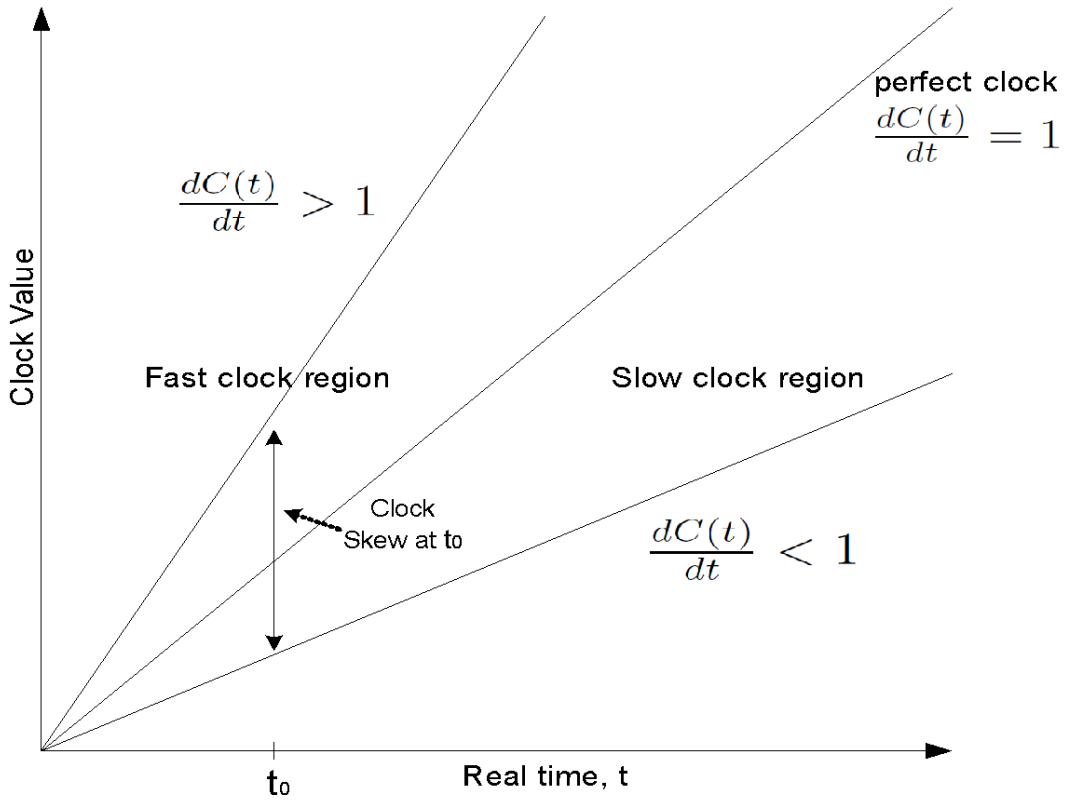

Figure 1.1: Perfect, fast and slow clocks

The measure of clocks drift with regard to each other is clock skew. Clock drift is a special case of clock skew with respect to the perfect clock. The diagrams below depict clock drift and clock skew:

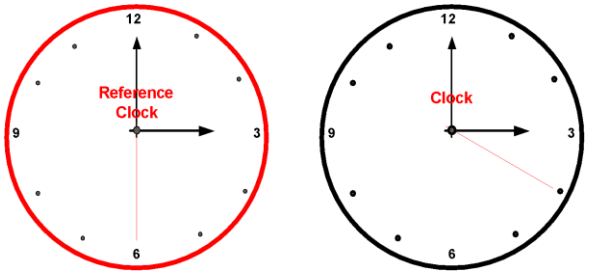

Figure 1.2(a): Clock Drift – Time difference with the Reference clock

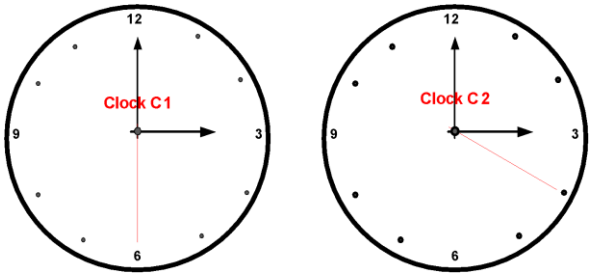

Figure 1.2(b): Clock Skew – Time difference between any two clocks

The drift rates an ordinary quartz clock drifts is around one to two seconds per ten to twelve days i.e. approximately one microsecond per second. A high precision quartz clock drifts around ten to hundred nanosecond per second. The most accurate clock, Atomic clock, which gives the coordinated Universal Time or UTC, drifts at around hundred femto second per second. Hence there is no perfect physical clock but for all purpose the atomic clock is considered the perfect clock.

In an ideal case we want all nodes to share the exact notion of time but due to drifting this is not possible. So, we relax the meaning of 'exactness'. We define the 'Relax exactness' in sense of an interval which is ensured by clock synchronization. Clock synchronization ensures that these drifting clocks are within a thin envelope of time. This agreed bound is the maximum deviation, δ, within which all the good clocks should lie. In other words, maximum deviation is the maximum skew among all the good clocks. We know all these Clocks continue to drift further apart with time as all of them are drift with some rate even after synchronization. After a particular length of time these clocks will cross the agreed bound. Therefore, clock synchronization is done periodically to maintain the clock synchrony.

1.1.1. Accuracy and Precision

Accuracy is the degree of how close a value is to a true value. Precision is the measure of closeness among various values. Accuracy and Precision can be visualized as depicted in the figure below. Accuracy is the measure of the distance of the mean of the values to a reference value. Therefore, in context of networks accuracy of a network is the measure of closeness of the clocks in the network to a reference clock. This is related to the clock drift as describe above paragraph. The precision is the measure of the spread of the values as shown in the figure 1.3. In a network it is the measure maximum time distance among the clocks. This is similar to the idea of clock skew.

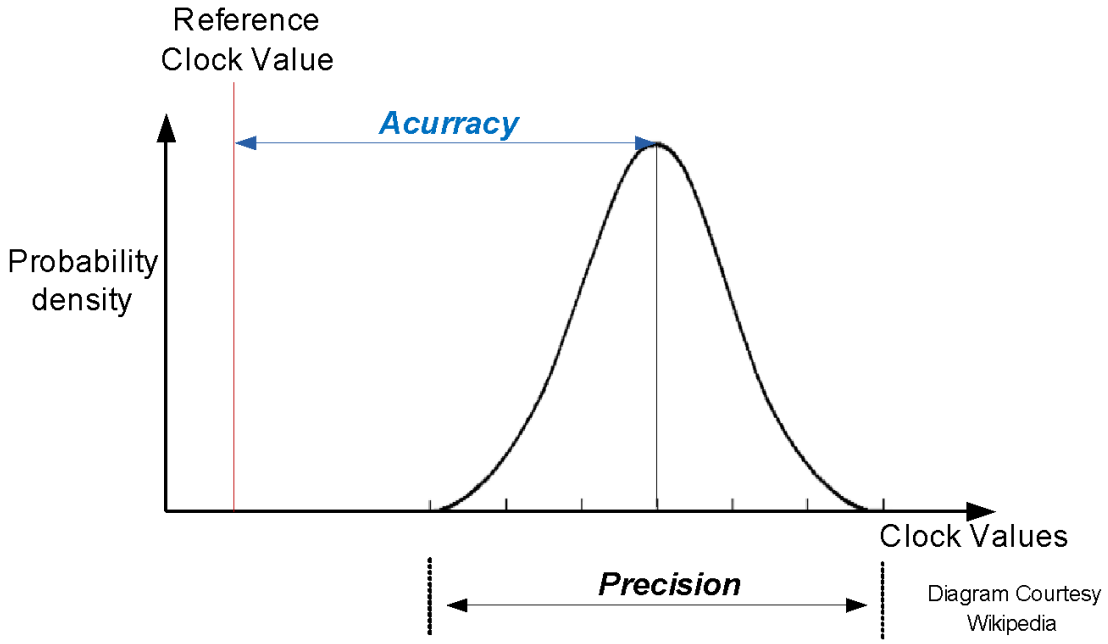

Figure 1.3: Accuracy and Precision

Here we use the concept of logical clock [C. Lenzen, et al, 2010] in the synchronization algorithm for distributed networks. Each of the nodes has a physical clock. Each node derives their logical clock from its physical clock. We achieve synchronization using the logical clock which is software defined.

The network consists of nodes, communication links and clocks. There are various failures possible in a network namely node failure, communication link failure and clock

failure. In this thesis our concern is clock failure. The various possible scenarios of clock failure in the network are as follows. Our algorithm provides the solution of the first and the second case.

(i) All clocks in the network are good i.e. they all behave. This is the ideal case and clock synchronization is easily achievable.

(ii) Good clocks are in majority and bad clocks are in minority. Here clock synchronization is achievable by using like majority voting function and various convergence functions. These bad clocks may include malicious clocks [L. Lamport, et al, 1982] with arbitrary behaviour.

(iii) Good clocks are in minority and bad clocks are in majority. Clock synchronization is difficult but possible. Clock synchronization is achieved by non-averaging convergence function.

In this thesis we propose two algorithms which use a weighted averaging method as a convergence function to achieve better precision. The first algorithm has a significant advantage. It offers improved Precision. The algorithm tolerates just under one third faulty nodes. The second algorithm offers better accuracy while keeping the precise synchrony.

In this section we give a brief on the background of fault tolerant clock synchronization. The motivation for taking up this subject for my thesis work is then brought out there after. The basic goal of clock synchronization in a distributed network is to make the clocks agree upon some time values within a thin envelope. These agreed time values may different from the outside world time. The task is to design a mechanism by which all the clock values come close to one another within a small bound. Since these clocks will continue to drift the mechanism will be required to be performed rapidly. The frequency for the same depends upon how much the clocks are allowed to drift before being brought closer. This process is basically the synchronization process.

1.1.2. Fault Tolerant

A process is fault tolerant if the process behaves satisfactory even in presence of faults. Clock synchronization is fault tolerant if the clocks are synchronized even in presence of faults. The synchronization process becomes challenging in the presence of a bad clock. Bad clocks are those clocks which fail to behave according to the design parameters. There are

many types of bad clocks but two common bad clocks are timing failure, bad clock and malicious clock. Timing failure, bad clock or simply bad clock drifts faster than the maximum drift rate. This type of bad clock can easily be handled as they are easily identifiable and accordingly segregated. A more serious problem is a malicious clock. A malicious clock can give arbitrary values and can also give different time to different clocks at the same time. The malicious clock is also called the Byzantine clock. The concept of the Byzantine clock had come from the Byzantine general Problem. The Byzantine General problem was first introduced by Prof. Leslie Lamport in the late 1970s.

Prof. Leslie Lamport initially proposed three different algorithms for fault tolerant clock synchronization for three different setups. Thereafter various algorithms were proposed by many authors using varieties of methods. The details of these algorithms are brought out in chapter 2. Many synchronization algorithms used averaging techniques as convergence functions with varying degrees of efficiency and complexities. One of the earlier algorithms [L. Lamport & et al, 1985] utilize a function called as an '***Egocentric averaging function'***. In this function the mean of the clock value is calculated and thereafter returned as the new clock value of the nodes. For calculation of the mean all clock values are taken except for those which are more than δ distance from it where the node replaces such with its own value. This method suffers from clustering if δ is very small and the quality of the precision is reduced if δ is very large. In a method proposed in [Jennifer Lundelius Welch & et al, 1988] the new clock value is the midpoint of the range which is calculated by discarding the highest and the lowest clocks value. The underlying assumption for the algorithm is good clocks lie in between bad clocks which are faster or slower than the good clocks. We have also studied two variants of '***Sliding Window Method***' used for obtaining clock synchronization are given in [Pfluegl & et al, 1995]. In the method, first a window of appropriate size chosen and then the mean / median of the chosen window is assigned as the new clock value. Another method for calculation is given in [Riccardo Gusella &el at, 1986; Stephen R. Mahaney & et at, 1985] where the average is calculated for all clocks value within δ for at least n-f other nodes for total of n nodes and f faulty nodes. There are also other algorithms based on non-averaging convergence techniques [Joseph Y, 1984; F. Cristian, et al, 1986; T. K. Srikanth and Sam Toueg, 1987] for clock synchronization. Some of this technique guaranteed optimal precision.

1.1.4. Motivation

The challenge of achieving better closeness in terms of accuracy and precision within the network in presence of faults, especially malicious faults and how mathematical tools can be utilized to optimize motivated this research. Here during our course of the research, we intent to study and develop concepts & precepts of clock synchronization with an aim to develop / design innovative clock synchronization algorithms which offer realistic, near optimal accuracy and precision for clocks synchronization. Our algorithm should work in a distributed network in presence of bad clocks and particularly in presence of malicious clocks. Mathematical tools are the ultimate instruments available with us with a prospect to further optimize the closeness.

1.2. Basic definitions

This section starts with the definitions of various terms used in the thesis hereafter. Then we give the system model. Subsequently we discussed some of the types of methods used for attaining clock synchronization. The last part of the section gives a brief of some of the previous works related to our problem at hand.

We have used some terms generally used in field of clock synchronization in this thesis. Many of these terms may be familiar with the readers but for the sack of clarity and also in attempt to bring the readers to a common platform, we have included the following definitions:

Definition 1.2.1. Time - A standard time as be defined as is a '**set of instants'** with a temporal precedence order '<' satisfying certain conditions: Linearity, Irreflexively, Eternity and Density.

Definition 1.2.2. Clock - Clock is a way of assigning a number to an event, where the number is the time when the event occurred. Clock can be more precisely defined as a function C_i for assigning time $C_i(e)$ to an event e of a process, p_i .

Definition 1.2.3. Physical Clock - A physical clock is a machine *m* that ticks on each small interval of time. The value of that physical clock is denoted by P_m (t), where t is real time.

Definition 1.2.4. Logical Clock - Logical clock L is a function that maps an event e in a system to an element in the time domain T, denoted as L(e) and called the time stamp of e and such that the following property is satisfied: for events e_1 and e_2 :

$$e1 \rightarrow e2 \Rightarrow L(e_1) < L(e_2)$$

Definition 1.2.5. Clock Drift - Any physical clock tends to diverge from the perfect clock due to the physical property of the material used in the clock or due to the design of the clock. This difference of the clocks value with respect to the perfect clock value is called the drift of the clock at that time. Mathematically drift rate can be represented as $\frac{dC(t)}{dt} = \rho$ where *t* is the real time. For perfect clock $\frac{dC(t)}{dt} = 1$.

Definition 1.2.6. Clock Skew - The phenomena of clocks drifting with one another is called is called clock skew. Clock drift is a special case of clock skew with respect to the perfect clock.

Definition 1.2.7. Clock Synchronization - Clock synchronization in a distributed network is any process that ensures every node including spatial diverged nodes shared a common notion of time.

Definition 1.2.8. Correct Behaviour - Correction Behaviour in context of distributed network is the performance of various function by the network component as expected according to the designed or algorithm.

Definition 1.2.9. Failure - A failure is a deviation from a correct behaviour. Any component or subsystem, which deviates from their correct behaviour, is considered to be failed.

Definition 1.2.10. Faulty Component - Any component, which fails, is faulty.

Definition 1.2.11. Malicious Behaviour - Malicious behaviour is an arbitrary fault that occurs during the execution of an algorithm by a distributed system. It is a serious type of failure that gives inaccurate, untimely and conflicting information.

1.3. System Model

This section has two sub-sections. within the first sub-section we provide a description of the Network design including that of the assorted elements used. The failure model of the network utilized in the thesis is described within the second sub-section.

1.3.1. Network, Nodes and Clock

We consider a partially and totally connected distributed network where all the nodes can communicate to at least one another directly by-passing messages. The network is static and then no node is allowed to return in or leave from the network. The network works in decentralized distributed fashion where there's no master node and every nodes are of same hierarchy for the first algorithm. For the second algorithm we used hierarchical model of network with three different types of nodes. Each of the nodes features a local physical clock from where the logical clock of the node is obtained. We assumed that the physical clocks run continuously with regard to the important time. This physical clock can drift at rate ρ aside from the actual time because of factors like temperature, pressure or aging. The nodes don't have access to any global or a centralized master clock for the first algorithm and some of the nodes do have access in the second algorithm. We make the subsequent assumptions in our thesis:

Drift-Bound: The Physical Clock for any node n_i , $p_{n_i}(t)$ ρ-bounded for all real time t as following:

$$(1+\rho)^{-1} \leq \frac{dP_{n_i}(t)}{dt} \leq (1+\rho)$$

The logical clock is related to the physical clock as L (t) = P (t) + A(t) where A(t) is the called the adjustment. The adjustment, A(t), can be performed in many ways like say, discretely or linearly.

Initial synchronization: The logical clock of any two nodes n_i and n_j at t_0, where t_0 is the real time at the start of the synchronization algorithm, differ by real time α.

$$\left|L_{n_i}(t) - L_{n_j}(t)\right| \leq \alpha$$

This is the initial synchronization assumption. Here we are assuming that all the nodes are initially synchronized at real time t_0.

1.3.2. Failure Model

Here in our system, failures can be because of the following: -

(I) Failure of nodes,

(II) Their clock or because of link failure between the nodes or

(III) A mix of some or of these failures.

A clock is taken into account failed if the above-mentioned ***Drift-Bound*** assumption is violated. We are basically considering two varieties of clock failures namely

(I) Timing failure and

(II) Malicious clock failure

We also assume that whenever a clock gives

(I) Inaccurate,

(II) Untimely And

(III) Conflicting Information

i.e. started behaving like a 'dual-faced' clock then we assumed the clock could be a **Malicious Clock**. A clock may be a good clock if it's running with none of the failure. We also assumed that the utmost number of bad clocks is f.

Maximum Clock Failure: The maximum number of faulty clocks, f, allowed in the network for our algorithm is given by *3f+1* nodes.

1.4. Problem Statement

Our algorithms use a concepts of minimum variance window, sliding Window, weighted averaging method etc for attaining synchronization of the nodes in a distributed network. In one of the algorithms, we also use GPS clock inputs as reference clock value. The synchronization algorithm satisfies the following properties:

Agreement Property: For good clocks of nodes n_i and n_j at real time t immediately after re-synchronization, $C_{n_i}(t)$ and $C_{n_j}(t)$ satisfy:

$$\left|C_{n_i}(t) - C_{n_j}(t)\right| \leq \pi$$

where $\pi \leq (\delta + \varepsilon)$ is the precision of our synchronization algorithm.

Incremental Step property: Here we assume that a correct clocks will change every time by some amount say ξ at each synchronization.

Accuracy property: The correct clock of node n_i, for some constants a, b, c and d holds for real time $t_1 < t_2$ the following:

$$\frac{(t_2 - t_1)}{a} - b \leq C_{n_i}(t_2) - C_{n_i}(t_1) \leq (t_2 - t_1)c + d$$

Our aim is to make π as minimum as possible so that clocks are closer to one another, the ultimate condition being π = 0. The validity of the Agreement property and Incremental Step property implies that our good clocks are running linearly with relevance real time. we might also prefer to make $(t_2 - t_1)$ as close as possible to $C_{n_i}(t_2)$ - $C_{n_i}(t_1)$ for any node n_i. This ensures the time elapsed by the clock of any node n_i is same to the real time elapsed in the interval. For optimal accuracy a and c should equal to one and b and d should be zero in expression of Accuracy property.

1.5. Research Contribution

We have developed two new algorithms for clock synchronization for a distributed static network. Mathematical tools are employed to optimize the clock synchronization algorithms. The algorithms work in presence of various faults including malicious clocks. Both the algorithm is designed in such a way that the effects of bad clocks and malicious clocks which are preventing the good clocks to synchronize are mitigated in two steps without identifying them. Using the first algorithm we are able to achieve at least 33% tighter precision when compared to [Pfluegl & et al, 1995]. The second algorithm offers high accuracy while keeping the clock précised. We have carried out rigorous theoretical analysis of both the algorithms. Simulation of both the algorithms is also done to understand and validate working of the synchronization in near real network environment.

1.6. Organisation of the Thesis

The rest of the thesis is organised as under-numerated. In chapter 2, we discussed various literatures which were studied prior and during the course of this research. We have uses various tools of applied mathematics during developing of our research, some of such details are shared in chapter 3. In Chapter 4, we explain our first algorithm, Weighted Average Synchronization Algorithm (WASA). The detail of the simulation environment and the input data analysis is presented in chapter 5. In chapter 6, we explain our second algorithm followed by chapter 7 where we present simulation environment and the input data analysis of the second algorithm. In chapter 8, we give our concluding remarks and the future research direction.