ISSN 2277-3061

Improvement of Physical Clock Synchronization Algorithm by Two-Level Synchronization

K. Ashwin Kumar , M. Praveen Kumar Reddy, S. Rajesh Kumar, RA.K. Saravanaguru.
kavaliashwinkumar@gmail.com
Praveenkumar.onmails@gmail.com
rajeshvit2012@yahoo.com

ABSTRACT

Synchronization of the clocks is one of the essential thing for many applications in distributed systems. Clock synchronization is very important because they improve the performance and reliability of distributed systems. The main purpose of clock synchronization algorithms is to provide the common time to essential parts of the distributed systems. In this paper the problem considered is synchronization of clock with bounded clock drift and proposing a two level synchronization algorithm which synchronizes the processors local clocks by combining both internal and external clock synchronization.

Keywords

Internal synchronization, External synchronization, Software Clock, Hardware Clock.

Academic Discipline And Sub-Disciplines

Internal and External Clock Synchronization

SUBJECT CLASSIFICATION

Distributed Systems-Physical Clocks

TYPE (METHOD/APPROACH)

Survey paper

INTRODUCTION

A distributed system is a system in which hardware and software components are located at different network networks, communicate and coordinate their actions only by message passing and doesn't have any access to global clocks[1]. Computer system clocks are basically classified in to two types. One among them is hardware clock and the other is software clock. The other name for hardware clock is also known as Timer. A computer timer is usually an electronic device that counts oscillations at definite frequency[1].The operating system reads node's hardware clock value $H_i(t)$ scales it and adds appropriate offset value to produce software clock[1]. The software clock can be given as $C_i(t) = \propto H_i(t) + \beta$. This paper is formatted as follows. In section 2, the state-of-art-study of the paper is discussed. In section 3, problem of the existing approach is discussed and in section 4 algorithm of the proposed approach is explained. Finally in section 5 conclusion of the paper is discussed.

STATE-OF-ART-STUDY

The major reasons for difficulty in designing clock synchronization algorithms are Remote clock global view and variation of each process transmission delay, drift rates 10^{-5} seconds per second variations in temperature, fault elements and so on. Clock synchronization with respect to an external time reference is called external clock synchronization and synchronize clocks themselves called internal clock synchronization. The main goal of external clock synchronization is all clocks time must be as close to its time source i.e., UTC time and the goal of internal clock synchronization is to minimize the maximum difference between any two clocks.

Many software clock synchronization algorithms are developed to get the clocks synchronized by using some standard networks. Software clock synchronization algorithms are classified into statistical, deterministic and probabilistic algorithms. Deterministic algorithm assumes some precision at upper bound (difference between two clock values). Statistical and probabilistic algorithms do not make any guesses about delays. As mentioned earlier clocks can be internally or externally synchronized. External synchronization is always accurate to bound where as internal synchronization is always agree within the bound. This means that externally synchronized clocks are always internally synchronized but vice versa is not true.

The clock synchronization problem in fault free systems was first addressed by Lamport et.al [2,3]. They addressed interactive convergence algorithm to solve the synchronization algorithm. Srikanth et.al [4,5] proposed an algorithm based on averaging function which solves the Lamport's clock synchronization problem. These algorithms are completely based on reliable connected networks which will run in rounds for resynchronizing to correct the clock drift. Lundelius et.al [6] proposed an algorithm which was based on lower and upper bounds. This algorithm ensures that the clock values do not drift away from hardware clocks.

Marzullo et.al [7] improved their algorithm to handle byzantine faults by altering clock rates, calculating new interval, in addition to the clock times. The algorithm proposed by Halpern et.al [8] mainly focuses on link failures in non faulty processor systems. The problem of this approach is that authentication is needed at every phase. A fault tolerant system proposed by Cristian et.al [9,10] was based on combination of internal and external clock synchronization. This fault tolerant algorithm will works well when we have 2F+1 are faulty out of F reference time servers. The algorithm degrades to internal synchronization when the faulty reference time servers exceed F. In this paper the internal synchronization is partially based on interactive convergence algorithm [3] but differs in the block of clock correction.

EXISTING PROBLEM ANALYSIS

The classical problem of distributed systems is clock synchronization problem and shown in the Fig. 1.

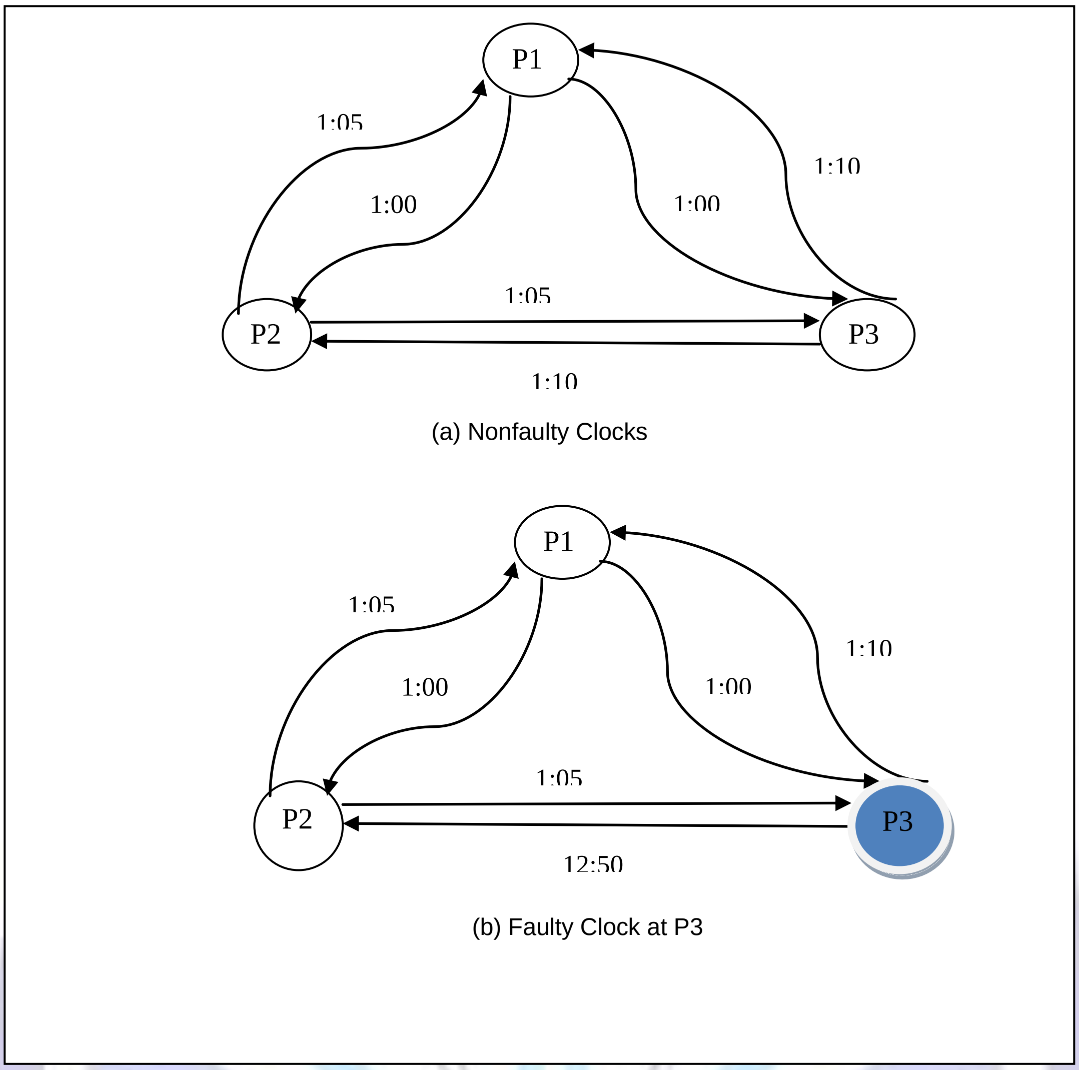

Fig 1. Clock Synchronization Byzantine Faults

In Fig. 1 A three node system was shown where each node has its own clock. If all the nodes are nonfaulty the clocks are synchronized by adjusting their median clock values. In Fig. 1b the node P3 is faulty and reports incorrect timings to P1 and P2. To handle this type of byzantine faults several algorithms are proposed and works well with any one synchronization method i.e., either by internal or external synchronization.

The problem with existing solution is, suppose when few of the external time receivers are failed (unavailable) then the system lost its synchronization because the clocks of those will be failed to be externally synchronized. To overcome this problem, an algorithm was proposed which is the combination of both internal and external clock synchronization.

PROPOSED ALGORITHM ANALYSIS

The proposed algorithm synchronizes the processors clock by two level synchronization. Here each node is assigned with an UTC receiver. In this approach the synchronization level is divided into two intervals. First interval also called as external interval receives UTC time and synchronizes its local time with the obtained UTC time. Second interval also called as internal interval helps in achieving tight synchronization between local clocks

Pseudo code of External and Internal Clock synchronization Algorithms

```
Algorithm InternalSnchronization()
{
        I:=$I_0$ //The first internal synchronization level
        While(true)
        {
                Wait until $SC_a$=I; // wait till software clock becomes I
                Broadcast I;
                While receive (k,z) do // receive message k from node z
                $M_a$[z]=$SC_a$// A stores it's time in array M
                Calculate the difference and update $correct_a$= $correct_a$+difference;
                $SC_a(t)=HC_a(t)+correct_a(t)$;
                I:=I+N; //update I for next synchronization level N
        } //End While
}
Algorithm ExternalSynchronization()
{
        T:=$T_0$ // First external synchronization level
        while (true)
{
        wait until $SC_a$ becomes T;
        if (UTC time from receiver is available) then modify
        $SC_a$ and make correction to synchronize it with hardware time receiver;
        end if
        call InternalSynchronization();
        T:=T+Q //Q is the next synchronization level
}
}
```

Fig 2 Internal and External Clock Synchronization Pseudo code at node a

The pseudo code provides the informal description of both synchronization algorithms. Periodical execution of external algorithm keeps the clocks synchronized because every node consists of time receiver. If UTC time is unavailable then the control pass to the internal clock synchronization and synchronizes its local clocks.

CONCLUSION

In this paper we proposed a two level synchronization algorithm which synchronizes processors local clocks. The major advantage of this two level synchronization algorithm is it provides tight synchronization by internal synchronization and synchronizes with real time by external synchronization and is decentralized and fault tolerant.

REFERENCES

[1] George Coulouris, Jean Dollimore and Tim Kindberg (2009). *Distributed Systems Concepts and Design.* India: Pearson.

[2] L. Lamport, “Time, Clocks, and the Ordering of Events in a Distributed System." , Communications of the ACM, vol. 27, no. 7, pp. 558 – 565, July 1998.

[3] L. Lamport and P. Melliar-Smith, “Synchronizing Clocks in the Presence of Faults." ,Journal of the ACM, vol. 32, no. 1, pp. 52 – 78, Jan. 2001.

[4] D. Dolev, N. Lynch, S. Pinter, E. Stark, and W.Weihl, “Reaching Approximate Agreement in the Presence of Faults." . Journal of the ACM, vol. 33, no. 3, pp. 499 – 516, July 2000

[5] T.K. Srikanth and S. Toueg, “Optimal Clock Synchronization." Journal of the ACM, vol. 34,no. 3, pp. 626 – 645, July 1999.

[6] J. Lundelius and N. Lynch, “An Upper and Lower Bound for Clock Synchronization." . Information and Control, vol. 62, nos. 2/3, pp. 190 – 204, Aug./Sept.2008.

[7] K. Marzullo and S Owicki, “Maintaining the Time in a Distributed System.” ,Technical Report No. 83-247 , Stanford University. , Aug 2006.

[8] J. Halpern, B. Simons, R. Strong, and D. Dolev, “Fault-Tolerant Clock Synchronization." , Proceeding: 3rd Annual. ACM Symposium. On Principles of Distributed Computing, pp. 89 –102, Aug.2004.

[9] F. cristian and C. Fetzer, “Integrating External and Internal Clock Synchronization.”, Journal of Real-Time Systems, vol. 12 , no. 2., 1997, pp. 123-172.

[10] P.Verissimo, L Rodrigues, and A. Casimiro.”CESIUMSPRAY : a Precise and Accurate Global TimeService for Large-scale Systems.” . Real- Time Systems, vol. 12 , pp. 243-294, 2007.