I J C T A, 9(18) 2016, pp. 9035-9040

Distributed Clock Synchronization Algorithms: A Survey

C. Sharma[*] and Pooja S[*]

ABSTRACT

Distributed System is an agglomeration of sovereign computers which are linked together by a network and communicate with each other by message passing. Communication between processes in a distributed system can have unpredictable delays, processes can fail, and messages may be lost. Synchronization in distributed systems is harder than in centralized systems because the need for distributed algorithms. In this paper we see few of the algorithms which achieve clock synchronization in distributed systems.

Keywords: distributed systems, distributed algorithms, clock synchronization, Berkeley algorithm, Cristian's algorithm, NTP, intersection algorithm

I. INTRODUCTION

System is an agglomeration of sovereign computers which are linked together by a network and communicate with each other by message passing. Distributed system uses must discern a single, unified computing adroitness even thought it implemented by various systems in different locations. There can be various definitions of distributed systems based on the following different perspectives: Operational perspective, User perspective and distributed system characteristics perspective. Coulouris defines a distributed system as "a system in which hardware or software components located at networked computers communicate and coordinate their actions only by message passing" [1] and Tanenbaum defines it as "A collection of independent computers that appear to the users of the system as a single computer" [2]. According to Lamport, distributed system "is a system that prevents you from doing any work, when a computer that you have never heard of crashes" [3].

In order to support a single system view to the users of the distributed system, an abstraction layer needs to be implemented in software and is placed in middle of higher and lower layers called the 'middleware'. Higher layers consists of users/applications and lower layer consists of OS and networks as showing in Fig. 1.

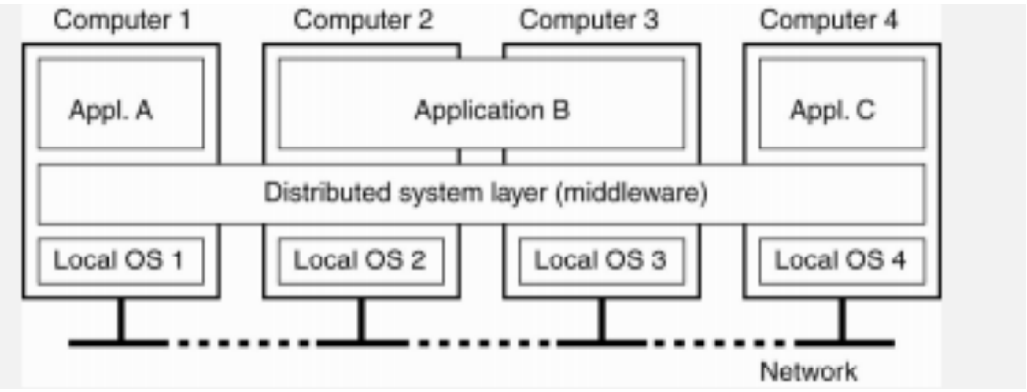

Figure 1: Distributed system

[*] Department of Information and Communication Technology, Manipal Institute of Technology, Manipal.

II. BENEFITS & DESIGN GOALS OF DISTRIBUTED SYSTEMS

As discussed, distributed systems should be able to provide single system image to its users even though it might be spread across the geographical locations. It might be notable to know that it is easier and cost-effective to connect multiple computers together rather than using a single machine.

(A) Benefits

The following are few benefits of distributed systems

i) Performance/Cost Ratio: If you pay double the price for a single system you can't expect to get double the performance. But, if you have multiple CPU's you might get (almost) double the performance for double the money as long as you keep the processors engaged.

ii) Computer supported cooperative networking: Users who are at different geographical locations can work or play together using the concept on which distributed systems are built. For example emails, file transfer, audio/video conferencing, Age of empires, etc.

iii) Reliability: Data can be backed-up at different locations. This multiple replication of data increases reliability.

iv) Economy: Since resources are shared, cost of ownership is reduced.

(B) Goals

There are basically four goals that should be met for an distributed system to be efficient.

i) Connecting Users and Resources: It is one of the mail goals of a DS. A distributed system should make it easier for users to access and share in a controlled way the remote resources. Collaboration of information can be made easier by connecting users and resources.

ii) Openness: Another important goal is to offer services to users of distributed systems according to definitive rules which describes semantics and syntax of those services.

iii) Scalable: Scalability of a system can be measured with respect to three dimensions. First dimension is size using which more users and resources can be added as and when required to the system. Second dimension is the geographical extent using which both the users and the resources can be far away from each other. Third dimension is administrative scalability using which the distributed systems can be easily manage even though the different distributed entities spans across many independent administrative organization.

iv) Transparency: The main aim of this concept is to hide the fact that the processes and the resources of a distributed system are actually spread across multiple computers which is necessary to maintain the single system image for the users.

Table I
Different forms of transparency in a distributed system

Transparency	*Description*
Access	Hide differences in data representation and how a resource is accessed
Location	Hide where a resource is located
Migration	Hide that a resource may move to another location
Relocation	Hide that a resource may be moved to another location while in use
Replication	Hide that a resource is replicated
Concurrency	Hide that a resource may be shared by several competitive users
Failure	Hide the failure and recovery of a resource

There are different forms of transparency in a DS as shown in Table I [4]

III. CLOCK SYNCHRONIZATION IN DISTRIBUTED SYSTEMS

Distributed systems are used by most of us due to its fault tolerance methods, scalability and stability which makes it possible for its users to make applications which are robust. Apart from the benefits that the distributed systems provide, they come with few challenges which include synchronization.Synchronization in distributed systems is harder than in centralized systems because the need for distributed algorithms. In clock synchronization, usually there will be a network of systems which should maintain a universal time [5]. Having a universal notion is necessary and important for most of the internet based applications. Each system would be having its own clock but it might not be accurate due to clock drift. In order to ensure that there is an agreement between the clocks of different systems in a network of systems, the drifts among individual clocks which can be achieved by sharing messages which contain information about current state of their clocks [5].

Data synchronization in a DS is of huge challenge. Just by the physical timestamp it's not possible to identify which version of the data is the most recent and/or most up-to-date. To solve the problem associated with physical clocks once can make use of logical clocks which is based on order of events rather than the timestamps to create partially ordered sets. Suppose we want to order events that occur in a DS in such a way that it should reflect their possible connections. Posolutely if an event X happens before an event Y, then X definitely is not caused by Y. X is not caused because of Y but X might have influenced Y. This can be characterized as "happened-before" relation on events.

The Leslie Lamport's "happened before" relation ($\rightarrow$) is defined as follows:

- $X \rightarrow Y$ if X and Y are within the same process and X occurred before Y.
- $X \rightarrow Y$ if X is the event of sending a message M in one process and Y is the event of receiving M by another process
- if $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$

IV. CLOCK SYNCHRONIZATION ALGORITHMS

We saw a brief introduction about clock synchronization in distributed systems now we will see few of the synchronization algorithms which could me made use of to achieve clock sync in a distributed scenario. The algorithms which would be discussed in this paper are Berkeley algorithm, Cristian's algorithm and NTP.

(A) Cristian's Algorithm

The simplest way to get a common notion of time among different entities of a distributed system is through RPC to a time server and obtain the time. But the so obtained time does not account for processing and networking delays. This can be compensated by measuring the local system time at which the request to the tome server is sent (T_0) and the time at which a response is received (T_1). Assume that the network delay caused to and from the server are symmetric. So the overhead caused can be estimated as (T_1 - T_0)/2. The new time now can be set as the sum of time returned by the server and the overhead. Fig.2 and Fig.3 shows a sample scenario of Cristian's algorithm [6].

$$T_{new} = T_{server} + \frac{T_1 - T_0}{2}$$

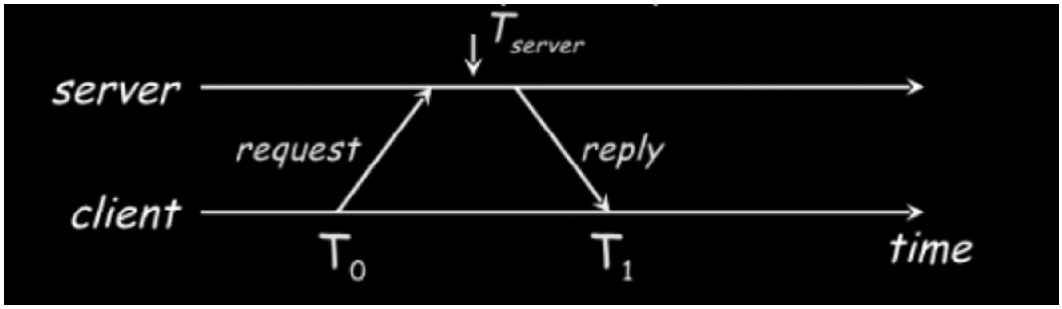

Figure 2: Cristian's Algorithm

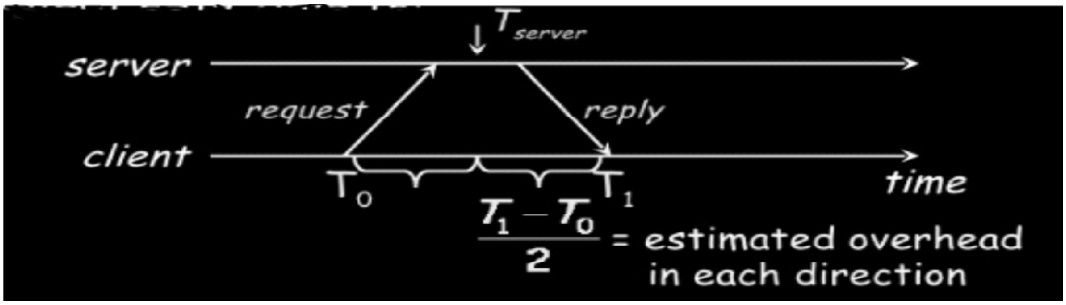

Figure 3: Cristian's Algorithm

If the smallest time interval for a message to travel from client to server and vice-versa is $T_{min,}$ the earliest time at which the server can generate a timestamp is $T_0 + T_{min}$. The latest time at which the server could generate timestamp is $T_1 - T_{min}$. The range of these times would be: $T_1 - T_0 - 2T_{min}$ as in Fig.4[6]and the accuracy would be [6]

$$\pm \left| \frac{T_1 - T_0}{2} - T_{\min} \right|$$

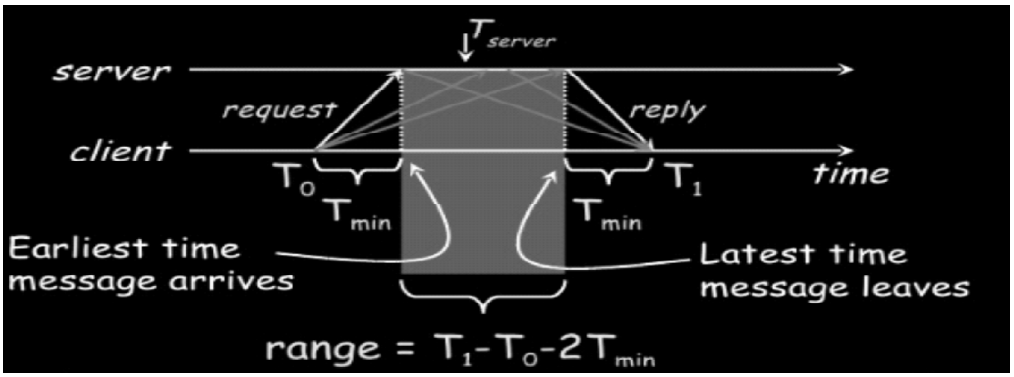

Figure 4: Error bounds in Cristian's algorithm

In Cristian's algorithm, many requests are simultaneously sent to the server in a hope that one of the requests might be delivered faster than others. This method can improve accuracy [6].

Main disadvantage of Cristian's algorithm is that if the time server fails the entire synchronization process might came to a standstill.

(B) Berkely Algorithm

Berkeley Algorithm was developed in 1989 by Zatti and Gusella. Unlike Cristian's algorithm, this algorithm does not depend on a single time server to get a universal notion of time, instead it gets average time from

all the participating systems and synchronizing all the systems to that average. A time daemon process would be running in all the systems which are in involved in synchronization [6]. This daemon process is responsible for implementing the protocol. Out of all the systems, one will be designated as the master and others will slave systems. Fig. 5 shows basic process involved in Berkeley algorithm.

Times of individual systems are noted and the master calculates the average time. The main aim of doing so is the hope that this average time avoids tendencies of individual clock's to run slow or fast. Once the average is calculated it's not sent to the slave system immediately, it calculates offset to compensate error bounds. The calculated average time plus the offset will be sent to the systems which adjusts their internal clocks accordingly.

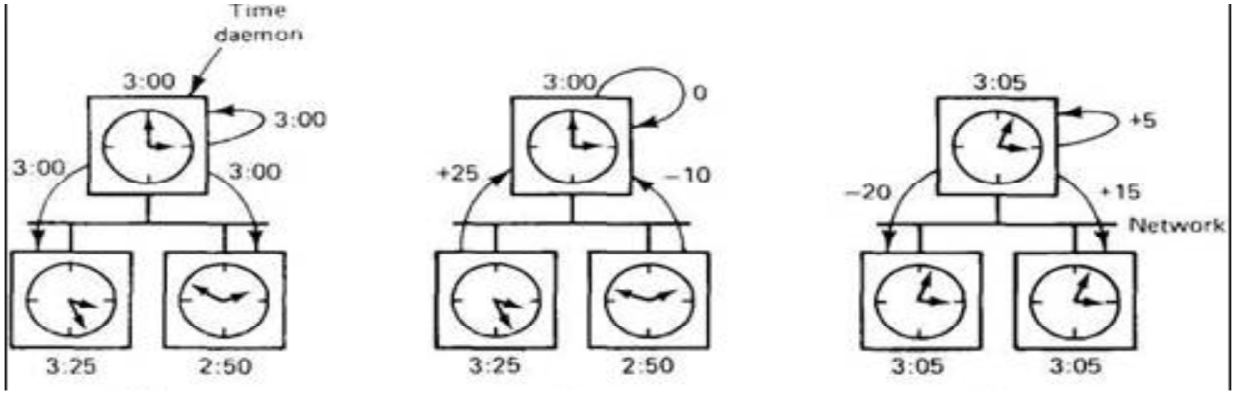

Figure 5: Berkeley Algorithm

Main advantage of the Berkeley Algorithm when compared to the Cristian's algorithm is that if the master fails, one of the slaves will take over the role of the master and the process continues.

(C) Network Time Protocol

NTP or the Network Time Protocol is networking protocol which used for clock synchronization. Network time protocol was originally developed by David L.Mills of University of Delaware. Keith Marzullo invented Marzullo's algorithm as a part of his Ph.D dissertation in 1984. Marzullo's algorithm deals with an algorithm which is used to estimate a universal notion of time from a number of noisy time sources. A modified or more refined version of Marzullo's algorithm forms the "Intersection Algorithm" which is made use of in NTP.

Marzullo's algorithm will get you the smallest interval consistent which contains largest number of sources. The intervals that are returned need not contain the calculated offset or the center point. The modified intersection algorithm on the other hand includes the intervals returned by Marzullo's algorithm plus the center points which results in a larger interval which reduces the jitter [7]. NTP is one of the best clock synchronization for systems in local area networks and under ideal conditions it can achieve better than 1ms accuracy.

Suppose there are N intervals of the form [c-r ,c+r], the algorithm aims to find an interval with N-k sources, where f is the number of erroneous sources called the falsetickers. If the number of falsetickers is least then that interval is considered the best. If k is less than half of the number of intervals then the results are valid and an interval is sent or else a failure is returned by the algorithm [7]

The intersection algorithm starts by creation of table of tuples{offset,type}. The types can be -1, 0 and +1 which are called the lower endpoint, midpoint and the upper endpoint respectively. Thus the entries in the tables would be {c-r,-1}, {c,0} and {c+r,+1} [7].

Algorithm [7]:

Step 1: Initialize value k and set counters endcountand midcount to zero.

Step 2: Starting from the lowest offset of the sorted table, from each touple{offset, type}, the type is subtracted from the endcount.

If (type==0) midcount++;

For some entry, if endcount ≥ N-k

Offset of that entry = lower offset of the intersection

Else {

k++;

repeat above procedure

}

Step 3: Value of the type is added to midcount and without changing the value of k or the midcount a similar method is employed to find the upper offset

Step 4: After calculating both upper and lower offset if midcount d" k, the algorithm continues, else k is increased by 1 and entire algorithm is repeated.

Advantages of NTP are that it is highly scalable, fault tolerant and if a network is temporarily not available then NTP uses past values to estimate current time and error [8].

V. CONCLUSION

Distributed System is an agglomeration of sovereign computers which are linked together by a network and communicate with each other by message passing. In this paper Berkeley algorithm, Cristian's algorithm and NTP which are the most used algorithms for clock synchronization in a distributed scenario are discussed along with their respective advantages. Future works could include optimizing the discussed algorithms to get more effective results.

REFERENCES

[1] G. Couloris, J. Dollimore, and T. Kinberg, Distributed Systems - Concepts and Design, 4th Edition, Addison-Wesley, Pearson Education, UK, 2001.

[2] A. Tanenbaum and M. Van Steen, Distributed Systems: Principles and Paradigms, Prentice Hall, Pearson Education, USA, 2002.

[3] Krishna Nadiminti, Marcos Dias de Assunção, and Rajkumar Buyya, "Distributed Systems and Recent Innovations: Challenges and Benefit"

[4] *http://csis.pace.edu/~marchese/CS865/Lectures/Chap1/Chapter1a.htm*

[5] Christoph Lenzen, Thomas Locher, Philipp Sommer, and Roger Wattenhofer "Clock Synchronization: Open Problems in Theory and Practice" Computer Engineering and Networks Laboratory TIK.

[6] Paul Krzyzanowski "Lectures on distributed systems – Clock Synchronization" Rutgers University.

[7] Romain Jacotin "MARZULLO's agreement algorithm and DTSS intersection algorithm (NTP) in Go programming language" , 2014.

[8] NTP FAQ Available: http://www.ntp.org/ntpfaq/NTP-s-def.htm#S-DEF-OV.