Advances in Computational Sciences and Technology
ISSN 0973-6107 Volume 10, Number 6 (2017) pp. 1941-1952

http://www.ripublication.com

Comparative Study of Clock Synchronization Algorithms in Distributed Systems

Neha N. Dalwadi[1] and Dr. Mamta C. Padole[2]

[12]The Maharaja Sayajirao University, Vadodara, Gujarat, India

Abstract

In the Distributed Systems (DS) the nodes are communicating with each other using message passing. Many real-time applications such as banking systems, reservation systems that are implemented on distributed systems, it is important to execute each transaction/event in an ordered manner. Ordering of events is essential for proper allocation of available resources and mutual allocation. This can be implemented using clock synchronization. The paper presents a comparative study of clock synchronization algorithms in distributed systems. The paper also discusses time protocol such as Network Time Protocol and Simple Network Time Protocol.

Keywords: Synchronization Algorithms, Distributed Systems, Network Time Protocol, Clock Synchronization.

1. INTRODUCTION

Distributed System (DS) is a collection of computers connected via the high-speed communication network. Distributed system is one in which hardware and software components located on a network communicate and coordinate their actions only by message passing [1]. There are two types of Distributed Systems:

a) Homogeneous Distributed Systems (HDS): It is a distributed system such that all nodes have identical hardware, the same type of architecture and operating system. [2].

b) Heterogeneous Distributed Systems (HeDS): It is a distributed system such that each node has their own operating system and machine architecture [2].

Each node in a distributed system can share their resources, e.g. the producer-consumer processes and the client-server processes, sharing of printer or scanner. But the resources can be limited therefore; they can be shared in either cooperative or competitive modes. Resources like a printer and scanner cannot be used by multiple processes simultaneously, so it must wait for one process to complete and then give chance to the next process. Another example is like producer-consumer and client-server processes which work in cooperative mode [3]. So there is a need of proper allocation of available resources, to preserve the state of resources and coordination between processes. To resolve these conflicts, clock synchronization is important. Clock synchronization can be implemented by using the physical local clock of each node. Some of the following example shows the problems with unsynchronized clocks [3]:

a) In a distributed banking system, if the timing and ordering of financial transactions are not tracked, it may raise inconsistent state in the system.

b) A distributed online reservation system in which the last available seat may get booked from multiple nodes if their local clocks are not synchronized.

c) There is a need to transmit a message from one node to another at any time. This will become difficult if sender and receiver clocks are not synchronized with each other.

Synchronization in DS can be achieved by using physical clock of the node. For synchronization purpose, each node in the system needs to share their local clock time with another node in the system. During this transaction (message passing of current clock time value) some factors like a communication link failure, fault tolerance, propagation time, non-receipt of acknowledgment, congestion in a network, the bandwidth of the communication link and routing mechanism affect and it may raise communication delay during this message passing which directly affects clock synchronization.

In computers, clock synchronization performs based on the physical clock of a computer. Before going into detail about synchronization, first we need to understand how physical clock structure of computer is implemented and how it works.

1.1 Computer Clock Construction and Mechanism

Each computer consists of quartz crystal that oscillates at a predictable frequency when squeezed. There is a counter register which keeps track of oscillation and decrements by one for every oscillation. When it reaches to zero, an interrupt is generated and counter is reloaded from the holding register [2]. The value of holding register is decided based on the frequency of oscillation and the number of interrupts

can be controlled. The holding register value is chosen to be 60 clock ticks per second.

There may be differences in crystal oscillation, leads to the clock running at different rates, which is known as clock drift. This difference in oscillation period may be small but difference accumulates over many oscillations hence computer clock drifts from real time clock. For quartz crystal, drift rate is 10-6 seconds or 1 second for every 11.6 days. The difference between time values of any two clocks is called their skew [3].

Most accurate physical clock include atomic oscillators whose drift rate is about one part in 1013. The atomic clock is used for a standard real time known as International Atomic Time. UTC is based on atomic time. Clock synchronization in distributed system relies on this standard external clock time value for synchronization. UTC time is used as a reference clock time for physical clocks in the system. There are two ways of synchronization:

(1) External clock synchronization: External clock time is used as a reference time for other clocks in the system and computer set their time accordingly.

(2) Internal clock synchronization: Each node shares their physical clock time value with other nodes and set their new clock value accordingly for clock synchronization.

1.3 Issues in Clock Synchronization

A simple method of clock synchronization is that each node has to send a request message ‘time=?’ to the real-time server. The node gets a reply message with ‘time=t’. This method has following issues [2]:

a) The ability of each node to read another node’s clock value. This can raise errors due to delay in message communication between nodes. Delay can be computed by computing the time needed to prepare, transmit and receive an empty message in the absence of transmission errors and system load.

b) Time must never run backward since it may lead to the repetition of events or transactions creating disorder in the system. Time running backward is just a perception, not actually it goes backward.

- **Reasons for Delay in Synchronization**

As discussed above, there are many reasons for a communication delay needs to be minimized to minimize delay and get nearby accurate time.

1. Communication Link Failure: For example, when sending a request message, communication link is working properly and message reaches to the server. If at the time of receiving message, communication link may fail due to some break. And client may not be able to get reply message. After recovery, reply reaches to the client which contains false time value.

2. Fault Tolerance: During message passing if any component fails, it may cause an inaccurate reading of clock time. So the system should be fault tolerant that can work in the faulty situation and minimize the clock drift value [4].

3. Propagation Time: Due to heavy traffic or congestion in the network, it may cause large propagation time from server to client. It may cause the inaccurate reading of the clock value in the reply.

4. Non Receipt of Acknowledgement: It may be possible that due to above reasons client will not get reply within a round trip time and therefore it sends multiple requests to server for synchronization.

5. The Bandwidth of Communication Link: Due to low bandwidth of communication link, congestion may occur in the network. Therefore request for time will not be able to reach the server or reply message will fail to reach client will affect clock synchronization.

2. CLOCK SYNCHRONIZATION ALGORITHMS

Clock synchronization is a method of synchronizing clock values of any two nodes in a distributed system with the use of external reference clock or internal clock value of the node. During the synchronization, many factors affect on a network. As discussed above, these factors need to be considered before correcting actual clock value. Based on the approach, clock synchronization algorithms are divided as Centralized Clock Synchronization and Distributed Clock Synchronization Algorithm.

2.1 Centralized Algorithms

There is one time server node which is used as a reference time for the following centralized algorithms [3]:

1. Passive Time Server

Each node periodically sends a request message ‘time=?’ to the time-server to get accurate time. Time-server responds with ‘time=t’. Assume that client node send a message at time t0 and get a reply at time t1, then message propagation time from server to client node is (t1-t0)/2.

Client node receives a reply and it adjusts its clock time to t + (t1-t0)/2. For a more accurate time, there is need to calculate accurate message propagation time. There are two methods proposed to improve estimated time value.

i) Additional information available

Assume that to handle interrupt and to process request message sent by the client, time server takes time ti. Hence, better estimated time is (t1-t0-ti)/2. Therefore, clock is readjusted to t + (t1-t0-ti)/2.

ii) No additional information available (Cristian Method)

This method uses time-server connected to a device that receives a signal from a source of UTC to synchronize nodes externally. A simple estimated time t + (t1-t0)/2, is accurate if nodes are on same network. For nodes on different network the minimum transmission time can be estimated as follows [4]:

The time t_{min} refers the time needed by server to prepare reply message. And the same time t_{min} is needed by client process to dispatch reply message as shown in the figure.

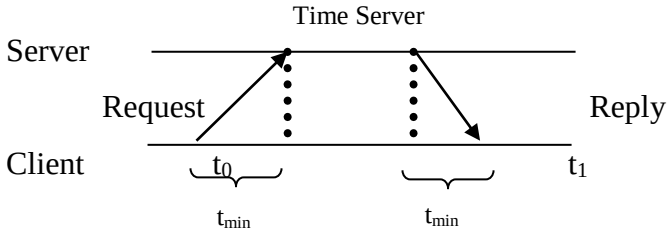

Fig. 1. *Cristian's Method*

When the reply message arrives, the time taken by server's clock is in the range of $(t_1-t_0)/2 - 2t_{min}$. Therefore, accuracy of clock time is $\pm (t_1-t_0)/2 - 2t_{min}$. There are some limitations as follows:

i. There is a single time server that might fail and thus synchronization temporarily unavailable.

ii. There may be faulty time-server that reply with spurious time or an imposter time-server that replied with incorrect time.

2. Active Time Server

Time server periodically broadcasts its clock time as 'time=t'. Other nodes receive message and readjust their local clock accordingly. Each node assumes message propagation time = ta, and readjust clock time = t + ta. There are some limitations as follows [5]:

i. Due to communication link failure message may be delayed and clock readjusted to incorrect time.

ii. Network should support broadcast facility.

3. Berkeley Algorithm

This algorithm overcomes limitation of faulty clock and malicious interference in passive time server and also overcomes limitation of active time server algorithm [4]. Time server periodically sends a request message 'time=?' to all nodes in the system. Each node sends back its time value to the time server. Time server has an idea of message propagation to each node and readjust the clock values in reply message based on it. Time server takes an average of other computer clock's value including its own clock value and readjusts its own clock accordingly. It avoids reading from unreliable clocks. For readjustment, time server sends the factor by which other nodes require adjustment. The readjustment value can either be +ve or –ve. There are following limitations:

i. Due to centralized system single point of failure may occur.

ii. A single time-server may not be capable of serving all time requests from scalability point of view. The readjustment value can either be +ve or –ve.

2 .2 Distributed Algorithms

There is no centralized or reference time-server. It performs clock synchronization based on internal clock values of each node with the consideration of minimum clock skew value among clocks of different nodes in the system. Distributed clock synchronization algorithm overcomes issue of scalability and single point of failure as there is no common or global clock required. Processes make decisions based on local information and relevant information distributed across machines [3].

1. Global Averaging Algorithm

Each node in the system broadcast its local clock time in the form of special 'resync' message when its local time is t0 + IR, where I is for interrupt time and R includes number of nodes in the system, maximum drift rate etc [2].

After broadcasting, clock process of the node waits for some time period t. During this period it collects 'resync' message from other nodes and records its receiving time based on its local clock time. At the end of waiting period, it estimates the skew of its own clock with respect to all other nodes. To find the correct time, need to estimate skew value as follows:

a) Take the average of estimated skew and use it as correction of local clock. To limit the impact of faulty clock skews greater than threshold are set to zero before computing average of estimated skew.

b) Each node limits the impact of faulty clock by discarding highest and lowest estimated skews and then calculates average of estimated skew. There are some limitations as follows [5][6]:

i. This method does not scale well.

ii. Network should support broadcast facility.

iii. Due to large amount of messages network traffic increased.

Therefore this algorithm may be useful for small networks only.

2. Localized Averaging Algorithm

This algorithm overcomes limitation of scalability of global averaging algorithm. Nodes of the system are arranged logically in a specific pattern like a ring or grid. Periodically each node exchange its local clock time with its neighbours and then sets its clock time to the average of its own clock time and the clock time of its neighbours. We have discussed set of physical clock synchronization algorithms. These algorithms can be implemented with the use of time protocol. Next we are going to discuss implementation of time protocol used to synchronize the clocks of computers.

2.3 Implementation of Network Time Protocol

NTP is an Internet protocol used to synchronize the clocks of computers with some time reference. It uses UTC (Coordinated Universal Time) as time reference. NTP is developed by David. Mills. NTP is a fault tolerant protocol that will automatically select the best of several available time sources to synchronize. It is highly scalable [7] [10].

- Working of NTP

The distributed system may consist of several reference clocks. Each node of such network can exchange information either bidirectional or unidirectional. A client node

can send request message to its directly connected time server node or nearly connected node so the propagation time from one node to another node forms hierarchical graph with reference clocks at the top. This hierarchical structure is known as strata synchronization subnet where each level is referred as strata. It is shown as below:

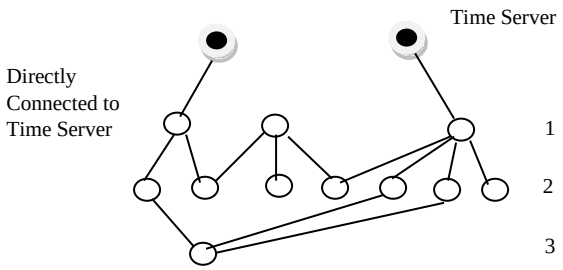

Fig. 2. *Strata Synchronization Subnet*

NTP servers synchronize with each other in one of the three modes:

i. Multicast Mode: In this mode of synchronization, one or more servers periodically multicast their time to the other servers running in the network. Other servers set their time accordingly with the addition of small delay during transaction. This method is proposed for use in a high-speed Local Area Network (LAN).

ii. Procedural Call Mode: It is similar to the process of Cristian's algorithm in which one server accepts requests from other computers and server replies with its current clock time. It gives more accurate time than multicast mode. It can be used where multicast is not supported in hardware.

iii. Symmetric Mode: In this mode of synchronization the pair of server exchanges messages. Servers supply time information in LANs and higher levels of the synchronization subnet where the highest accuracy is to be achieved.

There are some limitations of NTP protocol as follows:

i. NTP supports UNIX operating systems only.

ii. For windows there are problems with time resolution, reference clock drives, authentication and name resolution.

- **Simple Network Time Protocol (SNTP)**

This is a simplified way of clock synchronization method. SNTP is based on unicast mode of Network Time Protocol (NTP) and also operates in multicast and procedure call mode. It is recommended in network environment where server is root and client is leaf node [7]. Client node send request message at time t1 to server node and server send current time value in reply message at time t3 as shown below.

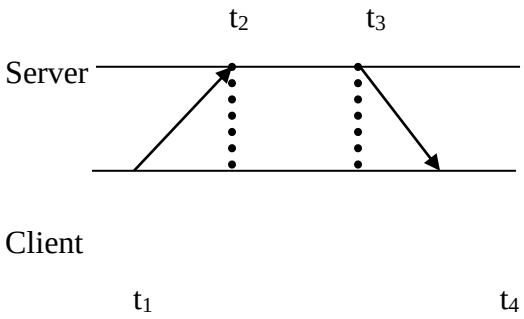

Fig.3. *SNTP Synchronization*

$$\text{Time offset (t)} = (t2 - t1) + (t3 - t4)/\ 2$$

$$\text{Current Time} = t4 + t$$

NTP protocol gives more accuracy in time than SNTP protocol. It is useful for simple applications where more accurate time is not necessary.

3 COMPARATIVE STUDY OF SYNCHRONIZATION ALGORITHMS

Table 1. Comparative Study of Synchronization Algorithms

Name of Parameter & Name of Algorithm	**Type of Algorithm**	**Approach**	**Scalability**	**Reasons for Implementation**	**Fault Tolerance**	**Limitation**
Cristian's Algorithm	Centralized	Passive Time Server and based on External clock synchronization approach	Poor	To minimize propagation time (in milliseconds)	Not	1. Single time server might be fail. 2. Faulty Time server cause server replied with incorrect time.

Berkeley Algorithm	Centralized	Active Time Server and based on Internal clock synchronization approach	Poor	To minimize the maximum difference between any two clock (in milliseconds)	Not	1. Server becomes bottleneck.
Global Averaging Algorithm	Distributed	No such time server and based on internal clock	Poor	To resolve single point of failure and to minimize skew value (in milliseconds)	Not	1. Network should support broadcast facility. 2. Congestion may occur due to large amount of message passing.
Network Time Protocol	Distributed	External Clock is used as reference time server. Based on Multiple time server arranged in levels.	Good	To minimize propagation time (in milliseconds) and faster access of correct time value.	Yes	1. Supports in UNIX system only
Precision Time Protocol	Centralized	Master-Slave approach where Master is controlled by GPS receiver	Good	More accuracy then NTP by using GPS receiver, order of timing in (in microseconds)	Yes	1. Network should support multicasting. 2. Intended for relatively localized system.

4. CONCLUSION

Clock synchronization is necessary for the ordering of events and to preserve the state of resources. As per algorithms, we can say that for clock synchronization there is need to consider propagation time of messages among each node in both types of algorithms centralized and distributed. Centralized synchronization algorithm suffers from the issues like scalability and network traffic. These issues can be overcome by distributed synchronization algorithms, but it also suffers from the issue of fault tolerance. Therefore, it is required to find a method that can give accurate average skew to find accurate local clock time.

REFERENCES

[1] George Coulouris, Jean Dollimore, Tim Kindberg, Distributed Systems: Concepts and Design, 4/E, Pearson Education Ltd, 5th edition.

[2] A. S. Tanenbaum and M. V. Steen. Distributed systems: principles andparadigms, Prentice-Hall. 2002.

[3] Sunita Mahajan, Seema Shah, Distributed Computing, Oxford University Press, 2010.

[4] F. Cristian Probabilistic clock synchronization. In Distributed Computing, volume 3, pages 146-158. Springer Verlag, 1989.

[5] Parameshwaran Ramanathan, Kang Shin, Ricky Butler, Fault-Tolerant Clock Synchronization in Distributed System.

[6] Christoph Lenzen, Thomas Locher, Philipp Sommer and Roger Wattenhofer, Clock Synchronization: Open Problems in Theory and Practice.

[7] http://www.ntp.org/

[8] Jana van Greunen , Jan Rabaey, Lightweight Time Synchronization for Sensor Networks.

[9] J. Elson, L. Girod, and D. Estrin, Fine-Grained Network Time Synchronization using Reference Broadcasts, Proceedings of the Fifth Symposium on Operating systems Design and Implementation, Boston, MA. December 2002.

[10] Network Time Protocol (Version 3): Specification, Implementation and Analysis - Mills – 1992.

[11] S. Ganeriwal, R. Kumar, and M. B. Srivastava. Timing-Sync Protocol for Sensor Networks. In Proc. 1st International Conference on Embedded Networked Sensor Systems (SenSys), 2003.

[12] K. R¨omer. Time Synchronization in Ad Hoc Networks. In Proc. 2nd ACM International Symposium on Mobile ad hoc Networking & Computing (MobiHoc), 2001.

[13] P. Sommer and R. Wattenhofer. Gradient Clock Synchronization in Wireless Sensor Networks. In Proc. 8th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), 2009.

[14] A. Boukerche and D. Turget, "Secure Time Synchronization Protocols For Wireless Sensor Networks," *IEEE Wireless Communications*, October 2007, pp. 64-69

[15] F. Sivrikaya,and B.Yener, Time Synchronization in Sensor Networks: A Survey,2004. (www.cs.rpi.edu/~yener/PAPERS/WINET/timesync04.pdf)

[16] Miklós Maróti , Branislav Kusy , Gyula Simon , Ákos Lédeczi, The Flooding Time Synchronization Protocol, In Proceedings of the 2ed ACM International Conference on Embedded Networked Sensor Systems (SenSys), pages 39 – 49, Baltimore, MD, USA, 2004

[17] J. V. Greunen and J. Rabaey. Lightweight Time Synchronization for Sensor Networks. In Proceedings of the 2nd ACM International Workshop on Wireless Sensor Networks and Applications (WSNA), San Diego, CA, September 2003.

[18] Anurag Gupta, Time in Internet of Things, AGII tech Consulting, WSTS Jun

2016, San Jose.

[19] J.Y. Halpern et al., "Fault-Tolerant Clock Synchronization,"Proc. Third Ann. ACM Symp. Principles of Distributed Computing, ACM, New York, 1984, pp. 89-102.