ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/289202520

Clock Synchronization in Distributed Distant Objects

Article · December 2015

CITATIONS
0

READS
807

1 author:

Charles Kinyua
Chuka University, Kenya
6 PUBLICATIONS **8** CITATIONS

SEE PROFILE

All content following this page was uploaded by Charles Kinyua on 04 January 2016.

The user has requested enhancement of the downloaded file.

ISSN: 2321-7782 (Online)
Volume 3, Issue 12, December 2015
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

Clock Synchronization in Distribued Distant Objects

cgkinyua@chuka.ac.ke
Charles K. Gitonga
Department of Computer Science
Chuka University
Chuka, Kenya

Abstract: Clock synchronization in distributed system relies on an existence of external clocks for synchronization to occur. The messages are sent over a network. For distant objects, there can be a communication breakdown. The study aims to highlights the shortcomings of clock synchronization in related distant objects when there is breakdown in the communication links. Clock synchronization in distributed systems is briefly discussed with the shortcomings of each outlined. The proposed algorithm seeks to overcome these shortcomings in distant objects. The algorithm needs to be developed and tested further to ascertain its correctness. Development of such algorithm has not been accomplished in this study.

Keywords: clock synchronization, distant objects, algorithms, Nodes.

I. INTRODUCTION

What is the time? Ask this question to a group of audience and you will always get different responses. In centralized systems, time is unambiguous as each process can make a system call to the kernel to get the current time and any other process making a later will get advance time [6]. In distributed system (DS), there exist difficulties in agreement on time. In normal operations, it doesn't affect much. However, there exist systems and programs that require time to be precise. Unix make program is such an example.

Distributed systems consist of a collection of distinct processes (called nodes) which are spatially separated and which communicate with one another by exchanging Messages [1]. The messages travel through communication links. Communication links failures, attacks and delays may lead to the loss of the global information-carrying ability of the established network [7].

Clock synchronization is a problem in computer science which deals with issues of maintaining clocks in several nodes, especially in distributed system synchronized. Time varies owing to the fact that different clocks tick at different rates. As a result these clocks will drift. This causes time in the distributed system to differ.

The absence of universal concept of time carriages serious problems for the correct workings of the distributed applications and protocols which requires synchronized clocks to function. Consequently, clock synchronization algorithms are therefore required to enable the synchronization of clocks in distributed environments. It is fundamental to provide a distributed clock synchronization algorithm that ensures that that the nodes are able to acquire a common notion of time [6]. Clock synchronization algorithms are based on exchanging clock information among the nodes and try to eliminate the effects of non-determinism in the message delay and data processing time [2]. To this effect, various clock synchronization algorithms have been developed to deal with this problem. However, there exists inadequate information on the effects of communication links failures to clock synchronization.

© 2015, IJARCSMS All Rights Reserved

II. CLOCK SYNCHRONIZATION ALGORITHMS

In a distributed system, each node runs its own clock [6]. Every node keeps a timer which consists of mainly quartz crystals kept under tension to oscillate at a defined frequency. Their oscillation alters the content of counter and holding registers in the computer system which subsequently generates an interrupts to the computer CPU when the counter value is zero. Programming can be done to generate the required frequency. Each interrupts generates a clock tick which interrupts the service procedure in the software adding to the stored time thus keeping the system software time up to date. In Distributed systems, it is impossible to guarantee that the crystals will oscillate at the same frequency [6], thus time will start drifting apart in several nodes.

Time in these clocks vary due to clock skew which cause time to be faster or slower in some nodes. Since every machine on the networks keeps its own independent clock, the clock time may drift fast or slow, according to figure one below:

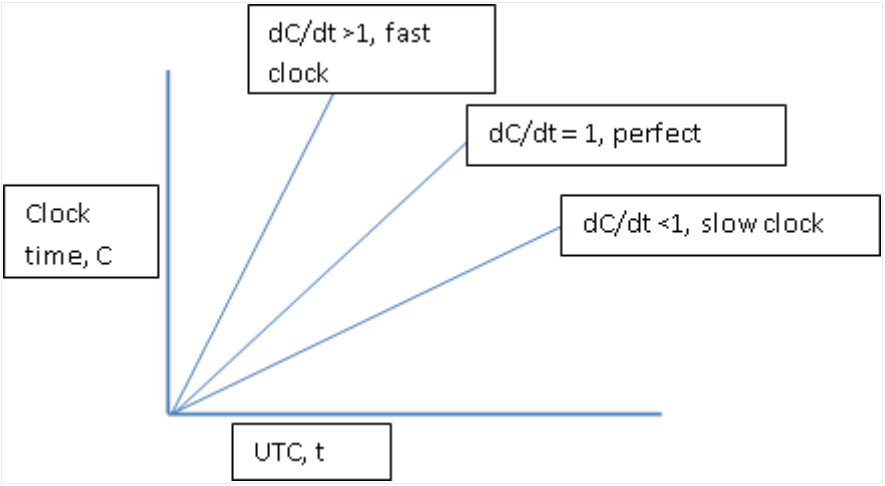

Figure 1: clock drifts in distributed system.

The drifting of clocks is caused by clock skew which causes clocks to gradually get out of sync with each other. Though there is a universally coordinate time, UTC, there exist the problem distributing time to each machine.

To synchronize clocks, we define 1- ρ <= dC/dt<=1+ρ, ρ is the maximum drift rate, this is what clock synchronization algorithms attempts achieve.

a) The Berkeley Algorithm

This algorithm requires a time server amongst the nodes. From time to time, the server pools every machine for its time. The received time is averaged periodically and then the server advices each node on how to slow down or advance its clock.

The algorithm may not work if there is breakdown in the communication system. The algorithm has a single point of failure, the time server.

b) Decentralized Algorithm

Every machine broadcasts its time periodically, for fixed length resynchronization interval. Each machine then averages the values from all other machines (or averages without the highest and lowest values).

c) Network Time Protocol

The algorithm uses both centralized and decentralized concepts. The algorithm allows nodes to contact a Time Server will provide the correct time. It is used as a protocol for synchronizing the clocks of computers over packet-switched, variable-latency data networks (i.e., Internet). NTPv4 can usually maintain time to within 10 milliseconds (1/100 s) over the public Internet, and can achieve accuracies of 200 microseconds (1/5000 s) or better in local area networks under ideal conditions [6]. However, they are latency delays in contacting the time server resulting in outdated time[6] forcing the algorithm to rely on time estimation. This algorithm may be ineffective in case of breakdown in communication networks or an attack on the time server.

© 2015, IJARCSMS All Rights Reserved

III. Deficiency of the above Algorithms

The sampled algorithms work well as long as the nodes maintain a communication with each other or with at least the server in a communication network without delays.

If we consider distant objects to be out of touch with any other communicating devices for a long period of time, probably due to communication breakdown, the problem of clock synchronization re-occurs. The distant nodes may be tasked with carrying out important tasks for us like studying distant galaxies. After a period of time, say 10 years after which they project back to our solar system to deliver information. Consider the study of the universe using distributed systems. Distributed systems are more reliable as they don't pose a single point of failure. They have strength in numbers as several nodes or processes can cooperatively solve a problem.

To study the universe deeply, scientist should employ the usage of distributed systems to minimize the risks of failures and improves on the reliability of data.

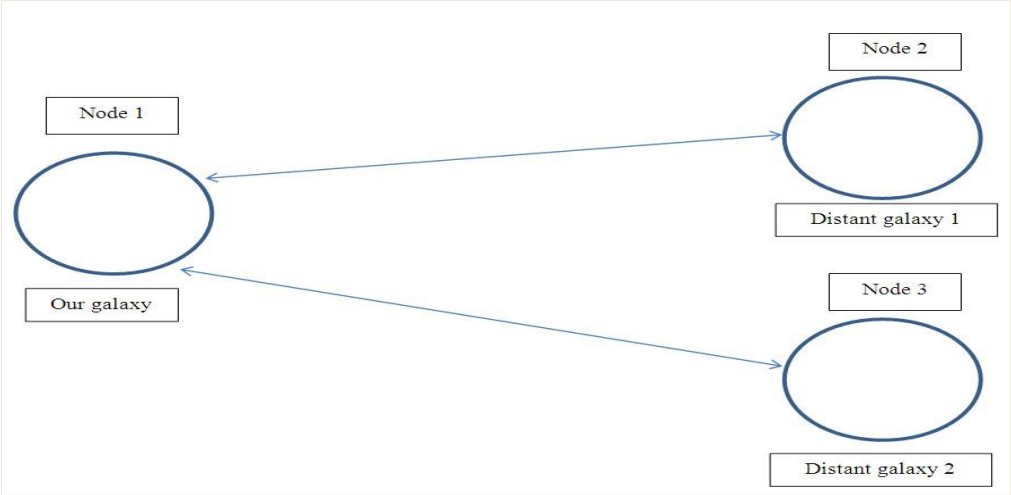

Figure 2: distributed distant object

IV. The Proposed Algorithm

This proposed algorithm will be based on holding virtual timers. The virtual timer, based on machine learning algorithms, learns how fast of slow the drifting in time is occurring in each node. Over a period of time, it will be able to add to or subtract some time to the existing time. We can also define a virtual server to maintain this algorithm and pass the drifting information the nodes.

The virtual time server and the nodes maintain a separate learning algorithm. The algorithms exchanges information to determine how to calculate their times accurately and independently. The correctness of the algorithm as to be based on complex machine learning algorithms not discussed in this study.

The algorithm can be defined as follows:

Given a set of nodes in a communication system, the server keeps the UTP time. The server periodically queries the time from other nodes, calculate their time drifts and advices them to advance or slow down their clocks. This is done over a period of time. Every time this occurs, the learning algorithms analyses the data and determines how its node behaves. Of course it will keep records of the clock on its respective node. The algorithm will then calculate the drifting pattern of its clock. If the algorithm is implemented on virtual timer server, after the elapse of some variable time, the time server can then hands over the data to the independent nodes, which will in turn learn how control their time independently.

© 2015, IJARCSMS All Rights Reserved

V. CONCLUSION

The existing clock synchronization algorithm fails to work in the loss of communication links. They also suffer various inadequacies in addressing the correct time. The alternative is to have self-synchronizing clocks based on learning algorithms.

ACKNOWLEDGMENT

We acknowledge the management of Chuka University and all the support staff during this study.

References

1. L. Lamport, “Time, clocks, and the ordering of events in a distributed system.” Commun. ACM 21, 7 (Jul. 1978), 558-565.
2. Yıldırım, K. S., & Kantarcı, A. Clock Synchronization in Distributed Systems.
3. Kopetz, Hermann, and Wilhelm Ochsenreiter. "Clock synchronization in distributed real-time systems." Computers, IEEE Transactions on 100.8 (1987): 933-940.
4. Dolev, Shlomi, and Jennifer L. Welch. "Self-stabilizing clock synchronization in the presence of byzantine faults." Journal of the ACM (JACM) 51.5 (2004): 780-799.
5. Hoch, Ezra N., Danny Dolev, and Ariel Daliot. "Self-stabilizing byzantine digital clock synchronization." Stabilization, Safety, and Security of Distributed Systems. Springer Berlin Heidelberg, 2006. 350-362.
6. Tanenbaum, Andrew S., and Maarten Van Steen. Distributed systems. Prentice-Hall, 2007.
7. Albert, Réka, Hawoong Jeong, and Albert-László Barabási. "Error and attack tolerance of complex networks." nature 406.6794 (2000): 378-382.

AUTHOR(S) PROFILE

Charles K. Gitonga, received the M.Sc degree in Computer Science at the University of Hull, UK. Currently he is a Lecturer and a research fellow in the department of computer science, Chuka University, Kenya.

© 2015, IJARCSMS All Rights Reserved

View publication stats