Create account Log in

Article Talk Read Edit View history Search

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Deutsch
فارسی
Français
Русский
Sicilianu
Edit links

Clock synchronization

From Wikipedia, the free encyclopedia

This article **may require cleanup to meet Wikipedia's quality standards**. The specific problem is: **Vague, non-specific writing.**.
Please help improve this article if you can. *(June 2013)*

Clock synchronization is a problem from computer science and engineering which deals with the idea that internal clocks of several computers may differ. Even when initially set accurately, real clocks will differ after some amount of time due to clock drift, caused by clocks counting time at slightly different rates. There are several problems that occur as a repercussion of clock rate differences and several solutions, some being more appropriate than others in certain contexts.[1]

In serial communication, some people[2] use the term "clock synchronization" merely to discuss getting one metronome-like clock signal to pulse at the same frequency as another one – frequency synchronization (plesiochronous or isochronous operation), as opposed to full phase synchronization (synchronous operation). Such "clock synchronization" is used in synchronization in telecommunications and automatic baud rate detection.

Contents [hide]

Problems [edit]

Besides the incorrectness of the time itself[*citation needed*], there are problems associated with clock skew that take on more complexity in a distributed system in which several computers will need to realize the same global time.

For instance, in Unix systems the *make* command is used to compile new or modified code without the need to recompile unchanged code. The *make* command uses the clock of the machine it runs on to determine which source files need to be recompiled. If the sources reside on a separate file server and the two machines have unsynchronized clocks, the *make* program might not produce the correct results.[3]

Solutions [edit]

In a centralized system the solution is trivial; the centralized server will dictate the system time. Cristian's algorithm and the Berkeley Algorithm are some solutions to the clock synchronization problem in a centralized server environment. In a distributed system the problem takes on more complexity because a global time is not easily known. The most used clock synchronization solution on the Internet is the Network Time Protocol (NTP) which is a layered client-server architecture based on UDP message passing. Lamport timestamps and Vector clocks are concepts of the logical clocks in distributed systems.

Cristian's algorithm [edit]

Main article: Cristian's algorithm

Cristian's algorithm relies on the existence of a time server.[4] The time server maintains its clock by using a radio clock or other accurate time source, then all other computers in the system stay synchronized with it. A time client will maintain its clock by making a procedure call to the time server. Variations of this algorithm make more precise time calculations by factoring in network radio propagation time.

Berkeley algorithm [edit]

Main article: Berkeley algorithm

This algorithm is more suitable for systems where a radio clock is not present, this system has no way of making sure of the actual time other than by maintaining a global average time as the global time. A time server will periodically fetch the time from all the time clients, average the results, and then report back to the clients the adjustment that needs be made to their local clocks to achieve the average. This algorithm highlights the fact that internal clocks may vary not only in the time they contain but also in the clock rate. Often, any client whose clock differs by a value outside of a given tolerance is disregarded when averaging the results. This prevents the overall system time from being drastically skewed due to one erroneous clock.

Network Time Protocol [edit]

Main article: Network Time Protocol

This algorithm is a class of mutual network synchronization protocol that allows for use-selectable policy control in the design of the time synchronization and evidence model. NTP supports single inline and meshed operating models in which a clearly defined master source of time is used ones in which no penultimate master or reference clocks are needed.

In NTP service topologies based on peering, all clocks equally participate in the synchronization of the network by exchanging their timestamps using regular beacon packets. In addition NTP supports a unicast type time transfer which provides a higher level of security. NTP performance is tunable based on its application and environmental loading as well. NTP combines a number of algorithms to robustly select and compare clocks, together with a combination of linear and decision-based control loop feedback models that allows multiple time synchronization probes to be combined over long time periods to produce high quality timing and clock drift estimates. Because NTP allows arbitrary synchronization mesh topologies, and can withstand (up to a point) both the loss of connectivity to other nodes, and "falsetickers" that do not give consistent time, it is also robust against failure and misconfiguration of other nodes in the synchronization mesh.

NTP is highly robust, widely deployed throughout the Internet, and well tested over the years, and is generally regarded as the state of the art in distributed time synchronization protocols for unreliable networks. It can reduce synchronization offsets to times of the order of a few milliseconds over the public Internet, and to sub-millisecond levels over local area networks.

A simplified verson of the NTP protocol, SNTP, can also be used as a pure single-shot stateless master-slave synchronization protocol, but lacks the sophisticated features of NTP, and thus has much lower performance and reliability levels.

Clock Sampling Mutual Network Synchronization [edit]

CS-MNS is suitable for distributed and mobile applications. It has been shown to be scalable over mesh networks that include indirectly linked non-adjacent nodes, and compatible to IEEE 802.11 and similar standards. It can be accurate to the order of few microseconds, but requires direct physical wireless connectivity with negligible link delay (less than 1 microsecond) on links between adjacent nodes, limiting the distance between neighboring nodes to a few hundred meters.[5]

Precision Time Protocol [edit]

Main article: Precision Time Protocol

A master/slave protocol for delivery of highly accurate time over local area networks

Reference broadcast synchronization [edit]

Main article: Reference Broadcast Time Synchronization

The Reference Broadcast Synchronization (RBS) algorithm is often used in wireless networks and sensor networks. In this scheme, an initiator broadcasts a reference message to urge the receivers to adjust their clocks.

Reference Broadcast Infrastructure Synchronization [edit]

Main article: Reference Broadcast Infrastructure Synchronization

The Reference Broadcast Infrastructure Synchronization (RBIS)[6] protocol is a master/slave synchronization protocol based on the receiver/receiver synchronization paradigm, as RBS. Is is specifically tailored to be used in IEEE 802.11 Wi-Fi networks configured in infrastructure mode (i.e., coordinated by an access point). The protocol does not require any modification to the access point.

Global Positioning System [edit]

Main article: Global Positioning System

The Global Positioning System can also be used for clock synchronization. The accuracy of GPS time signals is ±10 ns[7] and is second only to the atomic clocks upon which they are based.

See also [edit]

- International Atomic Time
- Network Identity and Time Zone
- Network Time Protocol
- Precision Time Protocol
- Vector clocks
- Einstein synchronisation

Time portal

External Links [edit]

- Accurate time vs. PC Clock Difference

References [edit]

1. ^ Tanenbaum, Andrew S.; Maarten van Steen (2002), *Distributed Systems : Principles and Paradigms*, Prentice Hall, ISBN 0-13-088893-1
2. ^ "Transmission on a Serial Line"
3. ^ GNU `make'
4. ^ Cristian, F. (1989), "Probabilistic clock synchronization", *Distributed Computing* (Springer) **3** (3): 146–158, doi:10.1007/BF01784024
5. ^ Rentel, C.; Kunz, T. (March 2005), "A clock-sampling mutual network synchronization algorithm for wireless ad hoc networks", *IEEE Wireless Communications and Networking Conference* (IEEE Press) **1**: 638–644, doi:10.1109/WCNC.2005.1424575, ISBN 0-7803-8966-2 `|chapter=` ignored (help)[*dead link*]
6. ^ Cena, G.; Scanzio, S.; Valenzano, A.; Zunino, C. (June 2015), "Implementation and Evaluation of the Reference Broadcast Infrastructure Synchronization Protocol", *IEEE Transactions on Industrial Informatics* (IEEE Press) **11**: 801–811, doi:10.1109/TII.2015.2396003
7. ^ "Common View GPS Time Transfer". nist.gov. Archived from the original on 2012-10-28. Retrieved 2011-07-23.

Categories: Synchronization | Clocks | Distributed computing problems

This page was last modified on 29 June 2015, at 14:16.