Executive Summary - Computer Network Time Synchronization

from *Alice's Adventures in Wonderland*, Lewis Carroll

The executive is the one on the left.

Related Links

- Network Time Synchronization Project
- Executive Summary - Computer Network Time Synchronization
- Network Time Protocol Timescale and Era Numbering
- Network Time Protocol Timescale and Leap Seconds
- NTP Protocol Conformance Statement
- Bibliography on Computer Network Time Synchronization
- NTP software Documentation

Table of Contents

Introduction

The standard timescale used by most nations of the world is Coordinated UniversalTime (UTC), which is based on the Earth's rotation about its axis, and the Gregorian Calendar, which is based on the Earth's rotation about the Sun. The UTC timescale is disciplined with respect to International Atomic Time (TAI) by inserting

leap seconds at intervals of about 18 months. UTC time is disseminated by various means, including radio and satellite navigation systems, telephone modems and portable clocks.

Special purpose receivers are available for many time-dissemination services, including the Global Position System (GPS) and other services operated by various national governments. For reasons of cost and convenience, it is not possible to equip every computer with one of these receivers. However, it is possible to equip some number of computers acting as primary time servers to synchronize a much larger number of secondary servers and clients connected by a common network. In order to do this, a distributed network clock synchronization protocol is required which can read a server clock, transmit the reading to one or more clients and adjust each client clock as required. Protocols that do this include the Network Time Protocol (NTP), Digital Time Synchronization Protocol (DTSS) and others found in the literature (See "Further Reading" at the end of this article.)

Protocol Design Issues

The synchronization protocol determines the time offset of the server clock relative to the client clock. The various synchronization protocols in use today provide different means to do this, but they all follow the same general model. On request, the server sends a message including its current clock value or *timestamp* and the client records its own timestamp upon arrival of the message. For the best accuracy, the client needs to measure the server-client propagation delay to determine its clock offset relative to the server. Since it is not possible to determine the one-way delays, unless the actual clock offset is known, the protocol measures the total roundtrip delay and assumes the propagation times are statistically equal in each direction. In general, this is a useful approximation; however, in the Internet of today, network paths and the associated delays can differ significantly due to the individual service providers.

The community served by the synchronization protocol can be very large. For instance, the NTP community in the Internet of 2002 includes over 230 primary time servers, synchronized by radio, satellite and modem, and well over 100,000 secondary servers and clients. In addition, there are many thousands of private communities in large government, corporate and institution networks. Each community is organized as a tree graph or *subnet*, with the primary servers at the root and secondary servers and clients at increasing hop count, or stratum level, in corporate, department and desktop networks. It is usually necessary at each stratum

level to employ redundant servers and diverse network paths in order to protect against broken software, hardware and network links.

Synchronization protocols work in one or more association modes, depending on the protocol design. Client/server mode, also called master/slave mode, is supported in both DTSS and NTP. In this mode, a client synchronizes to a stateless server as in the conventional RPC model. NTP also supports symmetric mode, which allows either of two peer servers to synchronize to the other, in order to provide mutual backup. DTSS and NTP support a broadcast mode which allows many clients to synchronize to one or a few servers, reducing network traffic when large numbers of clients are involved. In NTP, IP multicast can be used when the subnet spans multiple networks.

Configuration management can be a serious problem in large subnets. Various schemes which index public databases and network directory services are used in DTSS and NTP to discover servers. Both protocols use broadcast modes to support large client populations; but, since listen-only clients cannot calibrate the delay, accuracy can suffer. In NTP, clients determine the delay at the time a server is first discovered by polling the server in client/server mode and then reverting to listen-only mode. In addition, NTP clients can broadcast a special "manycast" message to solicit responses from nearby servers and continue in client/server mode with the respondents.

Security Issues

A reliable network time service requires provisions to prevent accidental or malicious attacks on the servers and clients in the network. Reliability requires that clients can determine that received messages are authentic; that is, were actually sent by the intended server and not manufactured or modified by an intruder. Ubiquity requires that any client can verify the authenticity of any server using only public information. This is especially important in such ubiquitous network services as directory services, cryptographic key management and time synchronization.

NTP includes provisions to cryptographically authenticate individual servers using symmetric-key cryptography in which clients authenticate servers using shared secret keys. However, the secret keys must be distributed in advance using secure means beyond the scope of the protocol. This can be awkward and fragile with a large population of potential clients, possibly intruding hackers.

Modern public-key cryptography provides means to reliably bind the server identification credentials and related public values using public directory services.

However, these means carry a high computing cost, especially when large numbers of time-critical clients are involved as often the case with NTP servers. In addition, there are problems unique to NTP in the interaction between the authentication and synchronization functions, since each requires the other for success.

The recent NTP Version 4 includes a revised security model and authentication scheme supporting both symmetric and public-key cryptography. The public-key variant is specially crafted to reduce the risk of intrusion, minimize the consumption of processor resources and minimize the vulnerability to hacker attack.

Computer Clock Modelling and Error Analysis

Most computers include a quartz resonator-stabilized oscillator and hardware counter that interrupts the processor at intervals of a few milliseconds. At each interrupt, a quantity called *tick* is added to a system variable representing the clock time. The clock can be read by system and application programs and set on occasion to an external reference. Once set, the clock readings increment at a nominal rate, depending on the value of *tick*. Typical Unix system kernels provide a programmable mechanism to increase or decrease the value of *tick* by a small, fixed amount in order to amortize a given time adjustment smoothly over multiple *tick* intervals.

Clock errors are due to variations in network delay and latencies in computer hardware and software (jitter), as well as clock oscillator instability (wander). The time of a client relative to its server can be expressed

$$T(t) = T(t_0) + R(t - t_0) + 1/2\ D(t - t_0)^2,$$

where t is the current time, T is the time offset at the last measurement update t_0, R is the frequency offset and D is the drift due to resonator ageing. All three terms include systematic offsets that can be corrected and random variations that cannot. Some protocols, including DTSS, estimate only the first term in this expression, while others, including NTP, estimate the first two terms. Errors due to the third term, while important to model resonator aging in precision applications, are neglected, since they are usually dominated by errors in the first two terms.

The synchronization protocol estimates $T(t_0)$ (and $R(t_0)$, where relevant) at regular intervals τ and adjusts the clock to minimize $T(t)$ in future. In common cases, R can have systematic offsets of several hundred parts-per-million (PPM) with random variations of several PPM due to ambient temperature changes. If not corrected, the resulting errors can accumulate to seconds per day. In order that these errors do not

exceed a nominal specification, the protocol must periodically re-estimate *T* and *R* and compensate for variations by adjusting the clock at regular intervals. As a practical matter, for nominal accuracies of tens of milliseconds, this requires clients to exchange messages with servers at intervals in the order of tens of minutes.

Analysis of quartz-resonator stabilized oscillators show that errors are a function of the averaging time, which in turn depends on the interval between corrections. At correction intervals less than a few hundred seconds, errors are dominated by jitter, while, at intervals greater than this, errors are dominated by wander. As explained later, the characteristics of each regime determine the algorithm used to discipline the clock. These errors accumulate at each stratum level from the root to the leaves of the subnet tree. It is possible to quantify these errors by statistical means, as in NTP. This allows real-time applications to adjust audio or video playout delay, for example. However, the required statistics may be different for various classes of applications. Some applications need absolute error bounds guaranteed never to exceeded, as provided by the following correctness principles.

Correctness Principles

Applications requiring reliable time synchronization such as air traffic control must have confidence that the local clock is correct within some bound relative to a given timescale such as UTC. There is a considerable body of literature that studies these issues with respect to various failure models such as fail-stop and Byzantine disagreement. While these models inspire much confidence in a theoretical setting, most require multiple message rounds for each measurement and would be impractical in a large computer network such as the Internet. However, it can be shown that the worst-case error in reading a remote server clock cannot exceed one-half the roundtrip delay measured by the client. This is a valuable insight, since it permits strong statements about the correctness of the timekeeping system.

In the Probabilistic Clock Synchronization (PCS) scheme devised by Cristian, a maximum error tolerance is established in advance and time value samples associated with roundtrip delays that exceed twice this value are discarded. By the above argument, the remaining samples must represent time values within the specified tolerance. As the tolerance is decreased, more samples fail the test until a point where no samples survive. The tolerance can be adjusted for the best compromise between the highest accuracy consistent with acceptable sample survival rate.

In a scheme devised by Marzullo and exploited in NTP and DTSS, the worst-case

error determined for each server determines a correctness interval. If each of a number of servers are in fact synchronized to a common timescale, the actual time must be contained in the intersection of their correctness intervals. If some intervals do not intersect, then the clique containing the maximum number of intersections is assumed correct *truechimers* and the others assumed incorrect *falsetickers*. Only the truechimers are used to adjust the system clock.

System clock correctness principles require that clock readings must be always monotonically increasing, so that no two successive clock readings will be the same. As long as the reading latency exceeds the hardware resolution, this behavior is guaranteed. With reading latencies dropping below the microsecond in modern processors, the system clock in modern operating systems runs in nanoseconds, rather than the microseconds used in the original Unix kernel. With processor speeds exceeding 1 GHz, this assumption may be in jeopardy.

Data Grooming Algorithms

By its very nature, clock synchronization is a continuous process, resulting in a sequence of measurements with each of possibly several servers and resulting in a clock adjustment. In some protocols, crafted algorithms are used to improve the time and frequency estimates and refine the clock adjustment. Algorithms described in the literature are based on trimmed-mean and median filter methods. The clock filter algorithm used in NTP is based on the above observation that the correctness interval depends on the roundtrip delay. The algorithm accumulates offset/delay samples in a window of several samples and selects the offset sample associated with the minimum delay. In general, larger window sizes provide better estimates; however, stability considerations limit the window size to about eight.

The same principle could be used when selecting the best subset of servers and combining their offsets to determine the clock adjustment. However, different servers often show different systematic offsets, so the best statistic for the central tendency of the server population may not be obvious. Various kinds of clustering algorithms have been found useful for this purpose. The one used in NTP sorts the offsets by a quality metric, then calculates the variance of all servers relative to each server separately. The algorithm repeatedly discards the outlyer with the largest variance until further discards will not improve the residual variance or until a minimum number of servers remain. The final clock adjustment is computed as a weighted average of the survivors.

Clock Discipline Algorithm

Modern computer clocks use a hardware counter to generate processor interrupts at intervals in the order of a few milliseconds. At each interrupt the processor increments the software system clock by the number of microseconds or nanoseconds between interrupts, often called the tick and sometimes the jiffy. The software resolution of the system clock is defined as the tick value. Most modern processors implement some kind of high resolution hardware counter that can be used to interpolate between ticks. The hardware resolution of the system clock is defined as the time between increments of this counter. However, the actual reading latency due to the kernel interface and interpolation code can range from a few tens of microseconds in older processors to under a microsecond in modern processors.

At the heart of the synchronization protocol is the algorithm used to adjust the system clock in accordance with the final adjustment determined by the data grooming algorithms. This is called the clock discipline algorithm or simply the discipline. Such algorithms can be classed according to whether they minimize the time offset or frequency offset or both. For instance, the discipline used in DTSS minimizes only the time offset, while the one used in NTP minimizes both time and frequency offsets. While the DTSS algorithm cannot remove residual errors due to systematic frequency errors, the NTP algorithm is more complicated and less forgiving of design and implementation mistakes.

All clock disciplines function as a feedback loop, with measured offsets used to adjust the clock oscillator phase and frequency to match the external synchronization source. The behavior of feedback loops is well understood and modelled by mathematical analysis. The significant design parameter is the time constant, or responsiveness to external or internal variations in time or frequency. Optimum selection of time constant depends on the interval between update messages. In general, the longer these intervals, the larger the time constant and vice versa. In practice and with typical network configurations the optimal poll intervals vary between one and twenty minutes for network paths to some thousands of minutes for modem paths.

Further Reading

1. Cristian, F. Probabilistic clock synchronization. In Distributed Computing 3, Springer Verlag, 1989, 146-158.
2. Digital Time Service Functional Specification Version T.1.0.5. DigitalEquipment Corporation, 1989.

3. Gusella, R., and S. Zatti. TEMPO - A network time controller for a distributed Berkeley UNIX system. IEEE Distributed Processing Technical Committee Newsletter 6, NoSI-2 (June 1984), 7-15. Also in: Proc. Summer 1984 USENIX (Salt Lake City, June 1984).

4. Kopetz, H., and W. Ochsenreiter. Clock synchronization in distributed real-time systems. IEEE Trans. Computers C-36, 8 (August 1987), 933-939.

5. Lamport, L., and P.M. Melliar-Smith. Synchronizing clocks in the presence of faults. JACM 32, 1 (January 1985), 52-78.

6. Marzullo, K., and S. Owicki. Maintaining the time in a distributed system. ACM Operating Systems Review 19, 3 (July 1985), 44-54.

7. Mills, D.L. Adaptive hybrid clock discipline algorithm for the Network Time Protocol. *IEEE/ACM Trans. Networking 6, 5* (October 1998), 505-514.

8. Mills, D.L. Improved algorithms for synchronizing computer network clocks. *IEEE/ACM Trans. Networks 3, 3* (June 1995) 245-254.

9. Mills, D.L. Internet time synchronization: the Network Time Protocol. IEEE Trans. Communications COM-39, 10 (October 1991), 1482-1493. Also in: Yang, Z., and T.A. Marsland (Eds.). Global States and Time in Distributed Systems, IEEE Press, Los Alamitos, CA, 91-102.

10. Mills, D.L. Modelling and analysis of computer network clocks. Electrical Engineering Department Report 92-5-2, University of Delaware, May 1992, 29 pp.

11. NIST Time and Frequency Dissemination Services. NBS Special Publication432 (Revised 1990), National Institute of Science and Technology, U.S. Department of Commerce, 1990.

12. Schneider, F.B. A paradigm for reliable clock synchronization. Department of Computer Science Technical Report TR 86-735, Cornell University, February 1986.

13. Srikanth, T.K., and S. Toueg. Optimal clock synchronization. JACM 34, 3 (July 1987), 626-645.

14. Stein, S.R. Frequency and time - their measurement and characterization (Chapter 12). In: E.A. Gerber and A. Ballato (Eds.). Precision Frequency Control, Vol. 2, Academic Press, New York 1985, 191-232, 399-416. Also in: Sullivan,

D.B., D.W. Allan, D.A. Howe and F.L. Walls (Eds.). Characterization of Clocks and Oscillators. National Institute of Standards and Technology Technical Note 1337, U.S. Government Printing Office (January, 1990), TN61-TN119.

 Home Page

 *David L. Mills <mills@udel.edu>*