March 2014
Geoff Huston

Protocol Basics: The Network Time Protocol

Back at the end of June 2012[0] there was a brief IT hiccup as the world adjusted the *Coordinated Universal Time* (UTC) standard by adding an extra second to the last minute of the 30th of June. Normally such an adjustment would pass unnoticed by all but a small dedicated collection of time keepers, but this time the story spread out into the popular media as numerous Linux systems hiccupped over this additional second, and they supported some high-profile services, including a major air carrier's reservation and ticketing backend system. The entire topic of time, time standards, and the difficulty of keeping a highly stable and regular clock standard in sync with a slightly wobbly rotating Earth has been a longstanding debate in the *International Telecommunication Union Radiocommunication Sector* (ITU-R) standards body that oversees this coordinated time standard. However, I am not sure that anyone would argue that the challenges of synchronizing a strict time signal with a less than perfectly rotating planet is sufficient reason to discard the concept of a coordinated time standard and just let each computer system drift away on its own concept of time. These days we have become used to a world that operates on a consistent time standard, and we have become used to our computers operating at sub-second accuracy. But how do they do so? In this article I will look at how a consistent time standard is spread across the Internet, and examine the operation of the *Network Time Protocol* (NTP).

Some communications protocols in the IP protocol suite are quite recent, whereas others have a long and rich history that extends back to the start of the Internet. The ARPANET switched over to use the TCP/IP protocol suite in January 1983, and by 1985 NTP was in operation on the network. Indeed it has been asserted that NTP is the longest running, continuously operating, distributed application on the Internet[1].

The objective of NTP is simple: to allow a client to synchronize its clock with UTC time, and to do so with a high degree of accuracy and a high degree of stability. Within the scope of a WAN, NTP will provide an accuracy of small numbers of milliseconds. As the network scope gets finer, the accuracy of NTP can increase, allowing for submillisecond accuracy on LANs and sub-microsecond accuracy when using a precision time source such as a *Global Positioning System* (GPS) receiver or a caesium oscillator.

If a collection of clients all use NTP, then this set of clients can operate with a synchronized clock signal. A shared data model, where the modification time of the data is of critical importance, is one example of the use of NTP in a networked context. (I have relied on NTP timer accuracy at the microsecond level when trying to combine numerous discrete data sources, such as a web log on a server combined with a *Domain Name System* (DNS) query log from DNS resolvers and a packet trace.)

NTP, Time, and Timekeeping

To consider NTP, it is necessary to consider the topic of timekeeping itself. It is useful to introduce some timekeeping terms at this juncture:

Stability How well a clock can maintain a constant frequency

Accuracy How well the frequency and absolute value of the clock compares with a standard reference time

Precision How well the accuracy of a clock can be maintained within a particular timekeeping system

Offset The time difference in the absolute time of two clocks

Skew The variation of offset over time (first-order derivative of offset over time)

Drift The variation of skew over time (second-order derivative of offset over time)

NTP is designed to allow a computer to be aware of three critical metrics for timekeeping: the *offset* of the local clock to a selected reference clock, the *round-trip delay* of the network path between the local computer and the selected reference clock server, and the *dispersion* of the local clock, which is a measure of the maximum error of the local clock relative to the reference clock. Each of these components is maintained separately in NTP. They provide not only precision measurements of offset and delay, to allow the local clock to be adjusted to synchronize with a reference clock signal, but also definitive maximum error bounds of the synchronization process, so that the user interface can determine not only the time, but the quality of the time as well.

Universal Time Standards

It would be reasonable to expect that the time is just the time, but that is not the case. The Universal Time reference standard has several versions, but these two standards are of interest to network timekeeping.

UT1 is the principal form of Universal Time. Although conceptually it is *Mean Solar Time* at 0° longitude, precise measurements of the Sun are difficult. Hence, it is computed from observations of distant quasars using long baseline interferometry, laser ranging of the Moon and artificial satellites, as well as the determination of GPS satellite orbits. UT1 is the same everywhere on Earth, and is proportional to the rotation angle of the Earth with respect to distant quasars, specifically the *International Celestial Reference Frame* (ICRF), neglecting some small adjustments.

The observations allow the determination of a measure of the Earth's angle with respect to the ICRF, called the *Earth Rotation Angle* (ERA), which serves as a modern replacement for *Greenwich Mean Sidereal Time*). UT1 is required to follow the relationship:

$$\text{ERA} = 2\pi(0.7790572732640 + 1.00273781191135448\ Tu) \text{ radians}$$
$$\text{where } Tu = (\text{Julian UT1 date} - 2451545.0)$$

Coordinated Universal Time (UTC) is an atomic timescale that approximates UT1. It is the international standard on which civil time is based. It ticks SI seconds, in step with *International Atomic Time* (TAI). It usually has 86,400 SI seconds per day, but is kept within 0.9 seconds of UT1 by the introduction of occasional intercalary leap seconds. As of 2012 these leaps have always been positive, with a day of 86,401 seconds.[9]

NTP uses UTC, as distinct from the *Greenwich Mean Time* (GMT), as the reference clock standard. UTC uses the TAI time standard, based on the measurement of 1 second as 9,192,631,770 periods of the radiation emitted by a caesium-133 atom in the transition between the two hyperfine levels of its ground state, implying that, like UTC itself, NTP has to incorporate leap second adjustments from time to time.

NTP is an "absolute" time protocol, so that local time zones—and conversion of the absolute time to a calendar date and time with reference to a particular location on the Earth's surface—are not an intrinsic part of the NTP protocol. This conversion from UTC to the wall-clock time, namely the local date and time, is left to the local host.

Servers and Clients

NTP uses the concepts of *server* and *client*. A server is a source of time information, and a client is a system that is attempting to synchronize its clock to a server.

Servers can be either a *primary server* or a *secondary server*. A primary server (sometimes also referred to as a *stratum 1* server using terminology borrowed from the time reference architecture of the telephone network) is a server that receives a UTC time signal directly from an authoritative clock source, such as a configured atomic clock or—very commonly these days—a GPS signal source. A *secondary server* receives its time signal from one or more *upstream servers,* and distributes its time signal to one of more *downstream servers* and *clients*. Secondary servers can be thought of as clock signal repeaters, and their role is to relieve the client query load from the primary servers while still being able to provide their clients with a clock signal of comparable quality to that of the primary servers. The secondary servers need to be arranged in a strict hierarchy in terms of upstream and downstream, and the stratum terminology is often used to assist in this process.

As noted previously, a stratum 1 server receives its time signal from a UTC reference source. A stratum 2 server receives its time signal from a stratum 1 server, a stratum 3 server from stratum 2 servers, and so on. A stratum n server can peer with many stratum $n - 1$ servers in order to maintain a reference clock signal. This stratum framework is used to avoid synchronization loops within a set of time servers.

Clients peer with servers in order to synchronize their internal clocks to the NTP time signal.

The NTP Protocol

At its most basic, the NTP protocol is a clock request transaction, where a client requests the current time from a server, passing its own time with the request. The server adds its time to the data packet and passes the packet back to the client. When the client receives the packet, the client can derive two essential pieces of information: the reference time at the server and the elapsed time, as measured by the local clock, for a signal to pass from the client to the server and back again. Repeated iterations of this procedure allow the local client to remove the effects of network jitter and thereby gain a stable value for the delay between the local clock and the reference clock standard at the server. This value can then be used to adjust the local clock so that it is synchronized with the server. Further iterations of this protocol exchange can allow the local client to continuously correct the local clock to address local clock skew.

NTP operates over the *User Datagram Protocol* (UDP). An NTP server listens for client NTP packets on port 123. The NTP server is stateless and responds to each received client NTP packet in a simple transactional manner by adding fields to the received packet and passing the packet back to the original sender, without reference to preceding NTP transactions.

Upon receipt of a client NTP packet, the receiver time-stamps receipt of the packet as soon as possible within the packet assembly logic of the server. The packet is then passed to the NTP server process. This process interchanges the IP Header Address and Port fields in the packet, overwrites numerous fields in the NTP packet with local clock values, time-stamps the egress of the packet, recalculates the checksum, and sends the packet back to the client.

The NTP packets sent by the client to the server and the responses from the server to the client use a common format, as shown in Figure 1.

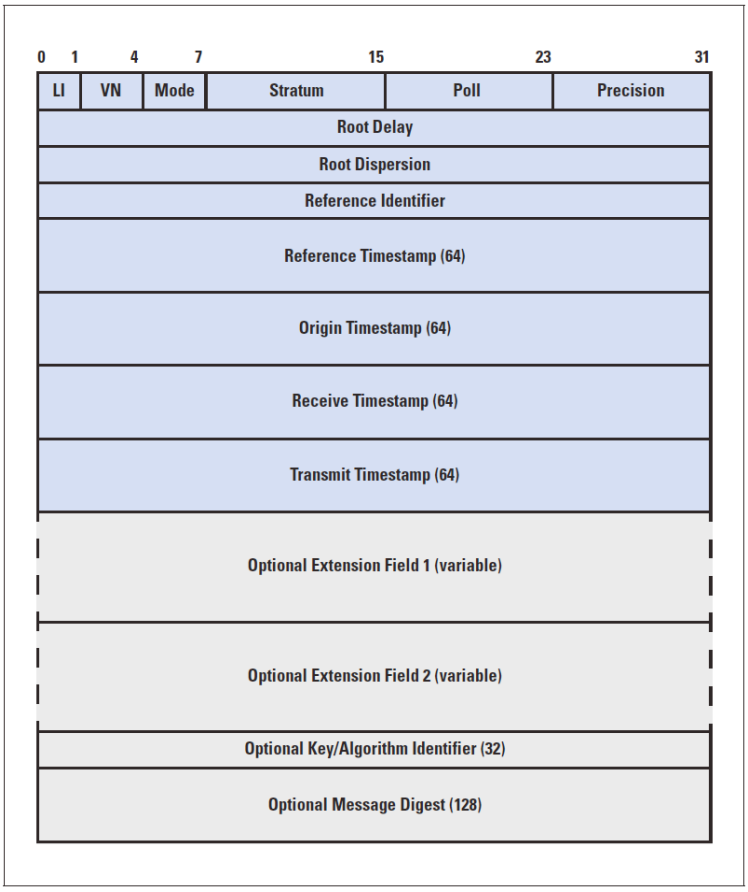

Figure 1: NTP Message Format

The header fields of the NTP message are as follows:

LI Leap Indicator (2 bits)
This field indicates whether the last minute of the current day is to have a leap second applied.
The field values follow:
0: No leap second adjustment
1: Last minute of the day has 61 seconds
2: Last minute of the day has 59 seconds
3: Clock is unsynchronized>

VN NTP Version Number (3 bits) (current version is 4).

Mode NTP packet mode (3 bits)
The values of the Mode field follow:
0: Reserved
1: Symmetric active
2: Symmetric passive
3: Client
4: Server
5: Broadcast
6: NTP control message
7: Reserved for private use

Stratum Stratum level of the time source (8 bits) The values of the Stratum field follow:
0: Unspecified or invalid
1: Primary server

2–15: Secondary server
16: Unsynchronized
17–255: Reserved

Poll Poll interval (8-bit signed integer)$_2$ value of the maximum interval between successive NTP messages, in seconds.

Precision Clock precision (8-bit signed integer)

The precision of the system clock, in $\log_2$ seconds.

Root Delay The total round-trip delay from the server to the primary reference sourced. The value is a 32-bit signed fixed-point number in units of seconds, with the fraction point between bits 15 and 16. This field is significant only in server messages.

Root Dispersion The maximum error due to clock frequency tolerance. The value is a 32-bit signed fixed-point number in units of seconds, with the fraction point between bits 15 and 16. This field is significant only in server messages.

Reference Identifier For stratum 1 servers this value is a four-character ASCII code that describes the external reference source (refer to Figure 2). For secondary servers this value is the 32-bit IPv4 address of the synchronization source, or the first 32 bits of the *Message Digest Algorithm 5* (MD5) hash of the IPv6 address of the synchronization source.

Code	External Reference Source
LOCL	uncalibrated local clock
CESM	calibrated Cesium clock
RBDM	calibrated Rubidium clock
PPS	calibrated quartz clock or other pulse-per-second source
IRIG	Inter-Range Instrumentation Group
ACTS	NIST telephone modem service
USNO	USNO telephone modem service
PTB	PTB (Germany) telephone modem service
TDF	Allouis (France) Radio 164 kHz
DCF	Mainflingen (Germany) Radio 77.5 kHz
MSF	Rugby (UK) Radio 60 kHz
WWV	Ft. Collins (US) Radio 2.5, 5, 10, 15, 20 MHz
WWVB	Boulder (US) Radio 60 kHz
WWVH	Kauai Hawaii (US) Radio 2.5, 5, 10, 15 MHz
CHU	Ottawa (Canada) Radio 3330, 7335, 14670 kHz
LORC	LORAN-C radionavigation system
OMEG	OMEGA radionavigation system
GPS	Global Positioning Service

Figure 2: Reference Identifier Codes (from RFC 4330)

The next four fields use a 64-bit time-stamp value. This value is an unsigned 32-bit seconds value, and a 32-bit fractional part. In this notation the value 2.5 would be represented by the 64-bit string:

```
0000|0000|0000|0000|0000|0000|0000|0010 . |1000|0000|0000|0000|0000|0000|0000|0000
       Integer Part                      |            Decimal Fractional Part
```

The unit of time is in seconds, and the epoch is 1 January 1900, meaning that the NTP time will cycle in the year 2036 (two years before the 32-bit Unix time cycle event in 2038).

The smallest time fraction that can be represented in this format is 232 picoseconds.

Reference Timestamp This field is the time the system clock was last set or corrected, in 64-bit time-stamp format.

Originate This value is the time at which the request departed the client for the server, in 64-bit time-

Timestamp stamp format.

Receive Timestamp This value is the time at which the client request arrived at the server in 64-bit time-stamp format.

Transmit Timestamp This value is the time at which the server reply departed the server, in 64-bit time-stamp format.

The basic operation of the protocol is that a client sends a packet to a server and records the time the packet left the client in the *Origin Timestamp* field (T1). The server records the time the packet was received (T2). A response packet is then assembled with the original Origin Timestamp and the *Receive Timestamp* equal to the packet receive time, and then the *Transmit Timestamp* is set to the time that the message is passed back toward the client (T3). The client then records the time the packet arrived (T4), giving the client four time measurements, as shown in Figure 3.

Timestamp Name	ID	When Generated
Originate Timestamp	T1	time request sent by client
Receive Timestamp	T2	time request received by server
Transmit Timestamp	T3	time reply sent by server
Destination Timestamp	T4	time reply received by client

Figure 3: NTP Transaction Timestamps (from RFC 4330)

These four parameters are passed into the client timekeeping function to drive the clock synchronization function, which we will look at in the next section.

The optional Key and Message Digest fields allow a client and a server to share a secret 128-bit key, and use this shared secret to generate a 128-bit MD5 hash of the key and the NTP message fields. This construct allows a client to detect attempts to inject false responses from a man-in the-middle attack.

The final part of this overview of the protocol operation is the polling frequency algorithm. A NTP client will send a message at regular intervals to a NTP server. This regular interval is commonly set to be 16 seconds. If the server is unreachable, NTP will back off from this polling rate, doubling the back-off time at each unsuccessful poll attempt to a minimum poll rate of 1 poll attempt every 36 hours. When NTP is attempting to resynchronize with a server, it will increase its polling frequency and send a burst of eight packets spaced at 2-second intervals.

When the client clock is operating within a sufficient small offset from the server clock, NTP lengthens the polling interval and sends the eight-packet burst every 4 to 8 minutes (or 256 to 512 seconds).

Timekeeping on the Client

The next part of the operation of NTP is how an NTP process on a client uses the information generated by the periodic polls to a server to moderate the local clock.

From an NTP poll transaction, the client can estimate the delay between the client and the server. Using the time fields described in Figure 3, the transmission delay can be calculated as the total time from transmission of the poll to reception of the response minus the recorded time for the server to process the poll and generate a response:

$$\delta = (T4 - T1) - (T3 - T2)$$

The offset of the client clock from the server clock can also be estimated by the following:

$$\Theta = \tfrac{1}{2}\,[(T2 - T1) + (T3 - T4)]$$

It should be noted that this calculation assumes that the network path delay from the client to the server is the same as the path delay from the server to the client.

NTP uses the minimum of the last eight delay measurements as δ_0. The selected offset, Θ_0, is one measured at the lowest delay. The values (Θ_0,δ_0) become the NTP update value.

When a client is configured with a single server, the client clock is adjusted by a slew operation to bring the offset with the server clock to zero, as long as the server offset value is within an acceptable range.

When a client is configured with numerous servers, the client will use a selection algorithm to select the preferred server to synchronize against from among the candidate servers. Clustering of the time signals is performed to reject outlier servers, and then the algorithm selects the server with the lowest stratum with minimal offset and jitter values. The algorithm used by NTP to perform this operation is *Marzullo's Algorithm*[2].

When NTP is configured on a client, it attempts to keep the client clock synchronized against the reference time standard. To do this task NTP conventionally adjusts the local time by small offsets (larger offsets may cause side effects on running applications, as has been found when processing leap seconds). This small adjustment is undertaken by an *adjtime()* system call, which slews the clock by altering the frequency of the software clock until the time correction is achieved. Slewing the clock is a slow process for large time offsets; a typical slew rate is 0.5 ms per second.

Obviously this informal description has taken a rather complex algorithm and some rather detailed math formulas without addressing the details. If you are interested in how NTP operates at a more detailed level, consult the references that follow, which will take you far deeper into the algorithms and the underlying models of clock selection and synchronization than I have done here.

Conclusion

NTP is in essence an extremely simple stateless transaction protocol that provides a quite surprising outcome. From a regular exchange of simple clock readings between a client and a server, it is possible for the client to train its clock to maintain a high degree of precision despite the possibility of potential problems in the stability and accuracy of the local clock and despite the fact that this time synchronization is occurring over network paths that impose a noise element in the form of jitter in the packet exchange between client and server. Much of today's distributed Internet service infrastructure relies on a common time base, and this base is provided by the common use of the Network Time Protocol.

References and Further Reading

[0] Geoff Huston, "Leaping Seconds," *The Internet Protocol Journal*, Volume 15, No. 3, September 2012.

[1] David L. Mills, "A Brief History of NTP Time: Confessions of an Internet Timekeeper," ACM SIGCOMM, *Computer Communication Review*, Vol. 33, No. 2, pp. 9–12, April 2003, **http://www.eecis.udel.edu/~mills/database/papers/history.pdf**

[2] K. A. Marzullo, "Maintaining the Time in a Distributed System: An Example of a Loosely-Coupled Distributed Service," Ph.D. dissertation, Stanford University, Department of Electrical Engineering, February 1984, **http://en.wikipedia.org/wiki/Marzullo%27s_algorithm**

[3] David L. Mills, "NTP Architecture, Protocol and Algorithms," University of Delaware, **www.eecis.udel.edu/~mills/database/brief/arch/arch.ppt**

[4] Jack Burbank, William Kasch, and David Mills, "Network Time Protocol Version 4: Protocol and Algorithms Specification," RFC 5905, June 2010.

[5] David L. Mills, "Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and

OSI," RFC 4330, January 2006.

[6] **http://www.ntp.org**

[7] **http://www.eecis.udel.edu/~mills/ntp.html**

[8] David Mills, *Computer Network Time Synchronization: the Network Time Protocol on Earth and in Space,* Second Edition, CRC Press, 2011.

[9] **http://en.wikipedia.org/wiki/Universal_Time**

Disclaimer

The above views do not necessarily represent the views or positions of the Asia Pacific Network Information Centre.

Author

Geoff Huston B.Sc., M.Sc., is the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region. He has been closely involved with the development of the Internet for many years, particularly within Australia, where he was responsible for the initial build of the Internet within the Australian academic and research sector. He is author of a number of Internet-related books, and was a member of the Internet Architecture Board from 1999 until 2005, and served on the Board of Trustees of the Internet Society from 1992 until 2001.
www.potaroo.net