Insider Tips: Network Time Protocol

ON THE DOT

Networks often require very accurate timekeeping. The Network Time Protocol provides the time with precision.

BY MARC ANDRÉ SELIG

Any cheap watch will keep better time than the chips in modern computers. There are tricks to improve precision – for example, you can ascertain the deviation and calculate a correcting vector – but you'll never find a real alternative to using the proper tools.

Computers often learn the time from extremely accurate reference clocks. There are many options. A DCF77 receiver connected to the serial or USB port allows a computer to receive the time from a reference clock (Figure 1).

If you need that kind of accuracy in your timekeeping, but you do not want to invest in additional hardware, you can simply poll a time source on the Internet. The protocol you need to do this is the Network Time Protocol (NTP), which can use either TCP or UDP (Port 123) to communicate. The *ntpd* on Linux implements the NTP protocol.

The protocol relies on a number of clever tricks to achieve accurate synchronization within a specific period of time. The reference time is known as UTC (Universal Time/Coordinated). NTP does not sync times over multiple machines, but it syncs the time on a single machine with UTC to the best of its ability.

To do so, NTC uses a hierarchical structure (Figure 2). The idea is that some computers have highly accurate time sources, such as the atomic clock at PTB. Computers that use a clock like this as a direct reference occupy the top level in the hierarchy: stratum 1. There are more levels, known as stratum 2, 3, 4, 5 and so on. You'll find a list of stratum 1 and 2 servers at [1]. Every computer in this system can connect to multiple machines in a stratum with a lower (better) number.

NTP Modes

NTP supports three different modes for passing a time signal from one computer to another. In traditional client/server operations, one computer will poll another to get the time. The client can accept a time signal from a server, but not vice-versa.

Symmetric mode is similar to client/server operations, however, in symmetric mode, the transmission direction can be reversed. If a server's time source goes down, the original client takes over as the server and acts as a source for the

other computers. In other words, the machines are peers.

Figure 3 shows a group of three peers that serve up the time redundantly. This group serves as the back-end for the receiving clients. But typically even larger networks will be fine using asymmetric mode and polling multiple time servers on the Internet.

In the third NTP mode, the servers all broadcast. Each server broadcasts an "alive" packet, telling the clients the IP to send their NTP request to; this saves manual configuration.

The Daemon

Many Linux and Unix systems include the legacy *ntpd*, which uses the */etc/ntp.conf* configuration file. In the simplest case – which is typical of most Linux systems – this file just has the name or IP address of the reference server.

To remove the need for admins to continually update the NTP server their clients use as a time source, the NTP project at [2] runs a DNS alias called *pool.ntp.org*. This resolves to a group of public servers. Listing 1 shows how simple the configuration can be. The daemon just picks one of the three IP addresses that the *ntpd* draws from the file.

Computers should set the time as early as possible in the bootstrap phase, even if they do not have a local NTP daemon. The *ntpd -q* command takes care of this by temporarily launching the NTP server and telling the server to get the correct time. The program then quits automatically. Many distributions use this approach in their boot scripts. Calling *ntpdate* provides similar results, but this tool does not have all the finesse of its bigger sibling. The command is thus generally regarded as obsolete, although it is still used quite frequently.

Time Trouble

Various problems can occur on syncing the time, and *ntpd* has various tricks to avoid them. If the computer has the wrong time, the program needs to avoid syncing too quickly. For example, if a cronjob was running at the time, and *ntpd* decided to put the clock back, the cronjob would run again.

The daemon needs to ensure continuity, providing linear time incrementation without jumping backward or forward. To achieve this, the daemon sets the clock in tiny steps – to be more precise, it increases or shortens the system's timespan for one second by half a millisecond per second, or 0.05 percent. And it keeps on doing so until the computer is back in sync.

Correcting the time by a single second takes 2000 seconds using this approach (that is, more than half an hour.) To set the correct time as quickly as possible, *ntpd* resorts to a compromise. If it notices a deviation of more than 128 milliseconds, it uses stepping to set the time once. It then uses the normal slewing approach to fine-tune the time by millisecond adjustments.

Listing 1: Typical /etc/ntp.conf

```
01 server pool.ntp.org
02 server pool.ntp.org
03 server pool.ntp.org
04 restrict default kod notrap
   nomodify nopeer noquery
```

Figure 1: The Physikalisch-Technische Bundesanstalt in Brunswick, Germany, has a number of atomic clocks. The clocks are accurate to about one to three billionths of a second per day, that is, about one millionth of a second per year. Computers on the Internet can use these clocks as time sources.

Incorrect Time Sources

If you use various Internet servers as time sources, you need some kind of protection against poor timekeeping. An NTP client solves this problem by comparing multiple time servers and finding out which of these servers is closest to UTC. It then syncs with just this one server.

This said, even the best algorithm for this task can return erroneous results from time to time, for example, if several servers are in contention. There is a contingency plan to handle this: if the NTP daemon notices a certain degree of inaccuracy (more than 1000 seconds by default), it will not correct the time.

NTP can handle situations where the network connection or the configured time server is down. In this case, the daemon measures the precision of the system clock and corrects the time based on its findings, even if a reference time is temporarily not available.

Slow Connections

Varying connection latency in the NTP server connection is another potential problem. An ISDN or DSL connection without a heavy load has a latency of a few thousand milliseconds, but if a big upload or download is in progress on this connection, the latency can increase rapidly to multiple seconds in either direction. As the NTP server has no way of ascertaining latency variations, it may set the system clock in quick succession.

ntpd has a special filter to handle this kind of situation, although the filter is

not typically enabled. If you need a workaround for this problem, you can enable the filter by adding the following command *tinker huffpuff 7200* to */etc/ntp.conf*.

As network clients are expected to work without a great deal of configuration effort, automatic NTP configuration would be preferable. There are several solutions for automatic NTP configuration. The least flexible approach is the one adopted by Apple on its Mac computers. If the admin enables the NTP daemon, the daemon uses a fixed, default time server.

The NTP project's time server pool is a far more sensible and simpler alternative. Version 4 of NTP additionally supports autonomous configuration with the daemon using manycast to poll computers on the network and automatically find an appropriate NTP server.

Security

Admins are always worried about network security, and rightly so in the case of NTP. There are basically two different attack scenarios: inside and outside of the protocol.

Working with the protocol, an attacker could supply an incorrect time signal to cover up log entries or even launch a denial of service attack. Administrators should add a *restrict* entry to the configuration file for a simple means of protecting the host (see Listing 1.) The only protection against incorrect time signals is cryptography. *ntpd* provides several options [3].

As *ntpd* typically runs as user ID 0 (root), it can be a potential target outside of the protocol. A buffer overflow would be all it took to gain control of the system. The only solution is to use the *ntpd -q* option, that is, to use a cronjob to make sure that the program is not permanently up.

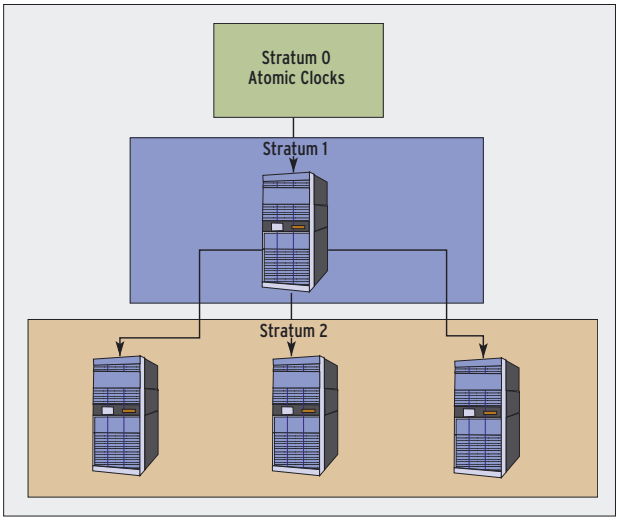

Figure 2: An NTP network is organized hierarchically in multiple levels. Time sources such as atomic clocks reside in stratum 0; computers that reference this kind of clock directly reside in stratum 1. In turn, they provide stratum 2 computers with an accurate time signal.

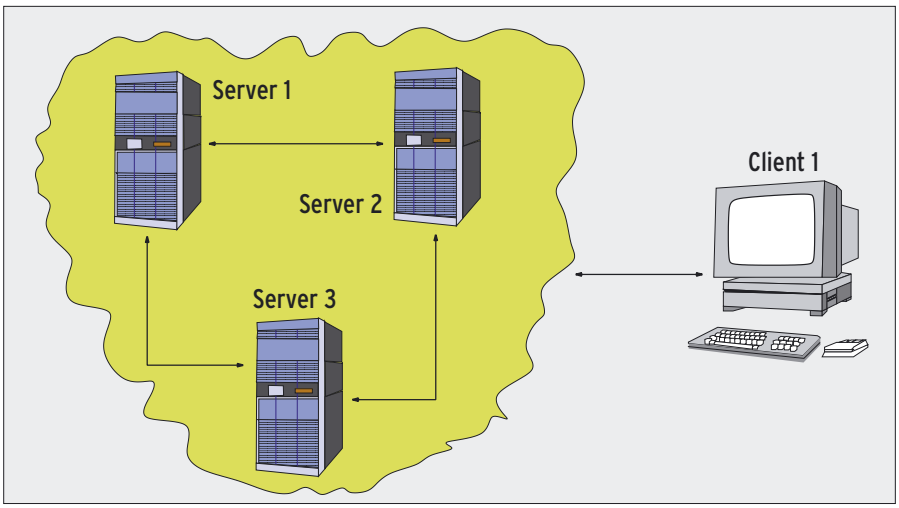

Figure 3: The three NTP servers (on the left) are running in symmetric mode and acting as redundant, mutual time sources. The client (on the right) simply polls one of the servers for the correct time.

Alternatives to NTP

NTP can give you extremely precise timekeeping, and the daemon is easy to manage. However, there are a number of alternative tools with similar functionality. A tool called RDate, for example, uses the time protocol that was standardized in RFC 868 [4]. RDate uses TCP or UDP port 37.

RDate always uses a binary value for the system time rather than a human-readable format. Besides the machine-readable RDate format, there is also the Daytime protocol (RFC 867 [5]), which uses TCP or UDP port 13 to transfer a time string in the clear. This format is useful for troubleshooting and debugging more than anything else.

Both RDate and Daytime mean you have to leave ports open at the firewall. If opening up the ports is not one of your options, your only alternative is to misuse an unblocked port. The HTPDate [6] program simply connects to a HTTP or HTTPS server and grabs the time stamp sent by the server in its HTTP response as a reference time. But if you plan to use HTPDate, you do need to make sure you are using a server with reliable timekeeping. ■

INFO

[1] The NTP.Servers Web: *http://ntp.isc.org/bin/view/Servers/WebHome*

[2] NTP Project: *http://www.ntp.org*

[3] NTP Authentication Support: *http://www.eecis.udel.edu/~mills/ntp/html/authopt.html*

[4] RFC 868 – Time Protocol: *http://www.faqs.org/rfcs/rfc868.html*

[5] RFC 867 – Daytime: *http://www.faqs.org/rfcs/rfc867.html*

[6] HTPDate: *http://www.clevervest.com/htp/*

THE AUTHOR

Marc André Selig spends half of his time working as a scientific assistant at the University of Trier and as an ongoing medical doctor in the Schramberg hospital. If he happens to find time for it, his current preoccupation is programming web based databases on various Unix platforms.