NAME

ntpd - Network Time Protocol (NTP) daemon

SYNOPSIS

ntpd [-aAbdgLmNPqx] [-c *conffile*] [-f *driftfile*] [-g] [-k *keyfile*] [-l *logfile*] [-N high] [-p *pidfile*] [-r *broadcastdelay*] [-s *statsdir*] [-t *key*] [-v *variable*] [-V *variable*] [-T *chroot_dir*] [-U *server_user*] [-x]

DESCRIPTION

The **ntpd** program is an operating system daemon which sets and maintains the system time of day in synchronism with Internet standard time servers. It is a complete implementation of the Network Time Protocol (NTP) version 4, but also retains compatibility with version 3, as defined by RFC-1305, and version 1 and 2, as defined by RFC-1059 and RFC-1119, respectively. **ntpd** does most computations in 64-bit floating point arithmetic and does relatively clumsy 64-bit fixed point operations only when necessary to preserve the ultimate precision, about 232 picoseconds. While the ultimate precision, is not achievable with ordinary workstations and networks of today, it may be required with future gigahertz CPU clocks and gigabit LANs.

HOW NTP OPERATES

The **ntpd** program operates by exchanging messages with one or more configured servers at designated poll intervals. When started, whether for the first or subsequent times, the program requires several exahanges from the majority of these servers so the signal processing and mitigation algorithms can accumulate and groom the data and set the clock. In order to protect the network from bursts, the initial poll interval for each server is delayed an interval randomized over 0-16s. At the default initial poll interval of 64s, several minutes can elapse before the clock is set. The initial delay to set the clock can be reduced using the **iburst** keyword with the **server** configuration command, as described on the Configuration Options page.

Most operating systems and hardware of today incorporate a time-of-year (TOY) chip to maintain the time during periods when the power is off. When the machine is booted, the chip is used to initialize the operating system time. After the machine has synchronized to a NTP server, the operating system corrects the chip from time to time. In case there is no TOY chip or for some reason its time is more than 1000s from the server time, **ntpd** assumes something must be terribly wrong and the only reliable action is for the operator to intervene and set the clock by hand. This causes **ntpd** to exit with a panic message to the system log. The **-g** option overrides this check and the clock will be set to the server time regardless of the chip time. However, and to protect against broken hardware, such as when the CMOS battery fails or the clock counter becomes defective, once the clock has been set, an error greater than 1000s will cause **ntpd** to exit anyway.

Under ordinariy conditions, **ntpd** adjusts the clock in small steps so that the timescale is effectively continuous and without discontinuities. Under conditions of extreme network congestion, the roundtrip delay jitter can exceed three seconds and the synchronization distance, which is equal to one-half the roundtrip delay plus error budget terms, can become very large. The **ntpd** algorithms discard sample offsets exceeding 128 ms, unless the interval during which no sample offset is less than 128 ms exceeds 900s. The first sample after that, no matter what the offset, steps the clock to the indicated time. In practice this reduces the false alarm rate where the clock is stepped in error to a vanishingly low incidence.

As the result of this behavior, once the clock has been set, it very rarely strays more than 128 ms, even under extreme cases of network path congestion and jitter. Sometimes, in particular when **ntpd** is first started, the error might exceed 128 ms. This may on occasion cause the clock to be set backwards if the local clock time is more than 128 s in the future relative to the server. In some applications, this behavior may be unacceptable. If the **-x** option is included on the command line, the clock will never be stepped and only slew corrections will be used.

The issues should be carefully explored before deciding to use the **-x** option. The maximum slew rate possible is limited to 500 parts-per-million (PPM) as a consequence of the correctness principles on which the NTP protocol and algorithm design are based. As a result, the local clock can take a long time to converge to an acceptable offset, about 2,000 s for each second the clock is outside the acceptable range. During this interval the local clock will not be consistent with any other network clock and the system cannot be used for distributed applications that require correctly synchronized network time.

In spite of the above precautions, sometimes when large frequency errors are present the resulting time offsets stray outside the 128-ms range and an eventual step or slew time correction is required. If following such a correction the frequency error is so large that the first sample is outside the acceptable range, **ntpd** enters the same state as when the **ntp.drift** file is not present. The intent of this behavior is to quickly correct the frequency and restore operation to the normal tracking mode. In the most extreme cases (**time.ien.it** comes to mind), there may be occasional step/slew corrections and subsequent frequency corrections. It helps in these cases to use the **burst** keyword when configuring the server.

FREQUENCY DISCIPLINE

The **ntpd** behavior at startup depends on whether the frequency file, usually **ntp.drift** , exists. This file contains the latest estimate of clock frequency error. When the **ntpd** is started and the file does not exist, the **ntpd** enters a special mode designed to quickly adapt to the particular system clock oscillator time and frequency error. This takes approximately 15 minutes, after which the time and frequency are set to nominal values and the **ntpd** enters normal mode, where the time and frequency are continuously tracked relative to the server. After one hour the frequency file is created and the current frequency offset written to it. When the **ntpd** is started and the file does exist, the **ntpd** frequency is initialized from the file and enters normal mode immediately. After that the current frequency offset is written to the file at hourly intervals.

OPERATING MODES

ntpd can operate in any of several modes, including symmetric active/passive, client/server broadcast/multicast and manycast, as described in the Association Management page. It normally operates continuously while monitoring for small changes in frequency and trimming the clock for the ultimate precision. However, it can operate in a one-time mode where the time is set from an external server and frequency is set from a previously recorded frequency file. A broadcast/multicast or manycast client can discover remote servers, compute server-client propagation delay correction factors and configure itself automatically. This makes it possible to deploy a fleet of workstations without specifying configuration details specific to the local environment.

By default, **ntpd** runs in continuous mode where each of possibly several external servers is polled at intervals determined by an intricate state machine. The state machine measures the incidental roundtrip delay jitter and oscillator frequency wander and determines the best poll interval using a heuristic algorithm. Ordinarily, and in most operating environments, the state machine will start with 64s intervals and eventually increase in steps to 1024s. A small amount of random variation is introduced in order to avoid bunching at the servers. In addition, should a server become unreachable for some time, the poll interval is increased in steps to 1024s in order to reduce network overhead.

In some cases it may not be practical for **ntpd** to run continuously. A common workaround has been to run the **ntpdate** program from a **cron** job at designated times. However, this program does not have the crafted signal processing, error checking and mitigation algorithms of **ntpd** . The **-q** option is intended for this purpose. Setting this option will cause **ntpd** to exit just after setting the clock for the first time. The procedure for initially setting the clock is the same as in continuous mode; most applications will probably want to specify the **iburst** keyword with the **server** configuration command. With this keyword a volley of messages are exchanged to groom the data and the clock is set in about a minute. If nothing is heard after a couple of minutes, the daemon times out and exits. After a suitable period of mourning, the **ntpdate** program may be retired.

When kernel support is available to discipline the clock frequency, which is the case for stock Solaris, Tru64, Linux and FreeBSD, a useful feature is available to discipline the clock frequency. First, **ntpd** is run in continuous mode with selected servers in order to measure and record the intrinsic clock frequency offset in the frequency file. It may take some hours for the frequency and offset to settle down. Then the **ntpd** is stopped and run in one-time mode as required. At each startup, the frequency is read from the file and initializes the kernel frequency.

POLL INTERVAL CONTROL

This version of NTP includes an intricate state machine to reduce the network load while maintaining a quality of synchronization consistent with the observed jitter and wander. There are a number of ways to tailor the operation in order enhance accuracy by reducing the interval or to reduce network overhead by increasing it. However, the user is advised to carefully consider the consequenses of changing the poll adjustment range from the default minimum of 64 s to the default maximum of 1,024 s. The default minimum can be changed with the **tinker minpoll** command to a value not less than 16 s. This value is used for all configured associations, unless overriden by the **minpoll** option on the configuration command. Note that most device drivers will not operate properly if the poll interval is less than 64 s and that the broadcast server and manycast client associations will also use the default, unless overriden.

In some cases involving dial up or toll services, it may be useful to increase the minimum interval to a few tens of minutes and maximum interval to a day or so. Under normal operation conditions, once the clock discipline loop has stabilized the interval will be increased in steps from the minumum to the maximum. However, this assumes the intrinsic clock frequency error is small enough for the discipline loop correct it. The capture range of the loop is 500 PPM at an interval of 64s decreasing by a factor of two for each doubling of interval. At a minimum of 1,024 s, for example, the capture range is only 31 PPM. If the intrinsic error is greater than this, the drift file **ntp.drift** will have to be specially tailored to reduce the residual error below this limit. Once this is done, the drift file is automatically updated once per hour and is available to initialize the frequency on subsequent daemon restarts.

THE HUFF-N'-PUFF FILTER

In scenarios where a considerable amount of data are to be downloaded or uploaded over telephone modems, timekeeping quality can be seriously degraded. This occurs because the differential delays on the two directions of transmission can be quite large. In many cases the apparent time errors are so large as to exceed the step threshold and a step correction can occur during and after the data transfer is in progress.

The huff-n'-puff filter is designed to correct the apparent time offset in these cases. It depends on knowledge of the propagation delay when no other traffic is present. In common scenarios this occurs during other than work hours. The filter maintains a shift register that remembers the minimum delay over the most recent interval measured usually in hours. Under conditions of severe delay, the filter corrects the apparent offset using the sign of the offset and the difference between the apparent delay and minimum delay. The name of the filter reflects the negative (huff) and positive (puff) correction, which depends on the sign of the offset.

The filter is activated by the **tinker** command and **huffpuff** keyword, as described in the Miscellaneous Options page.

NOTES

If NetInfo support is built into **ntpd** , then **ntpd** will attempt to read its configuration from the NetInfo if the default ntp.conf file cannot be read and no file is specified by the **-c** option.

Various internal **ntpd** variables can be displayed and configuration options altered while the **ntpd** is running using the **ntpq** and **ntpdc** utility programs.

When **ntpd** starts it looks at the value of **umask** , and if zero **ntpd** will set the **umask** to **022** .

COMMAND LINE OPTIONS

-a	Enable authentication mode (default).
-A	Disable authentication mode.
-b	Synchronize using NTP broadcast messages.
-c	*conffile* Specify the name and path of the configuration file. (Disable netinfo?)
-d	Specify debugging mode. This flag may occur multiple times, with each occurrence indicating greater detail of display.

-D *level* Specify debugging level directly.

-f *driftfile* Specify the name and path of the drift file.

-g Normally, **ntpd** exits if the offset exceeds the sanity limit, which is 1000 s by default. If the sanity limit is set to zero, no sanity checking is performed and any offset is acceptable. This option overrides the limit and allows the time to be set to any value without restriction; however, this can happen only once. After that, **ntpd** will exit if the limit is exceeded. This option can be used with the **-q** option.

-k *keyfile* Specify the name and path of the file containing the NTP authentication keys.

-l *logfile* Specify the name and path of the log file. The default is the system log facility.

-L Listen to virtual IPs.

-m Synchronize using NTP multicast messages on the IP multicast group address 224.0.1.1 (requires multicast kernel).

-n Don't fork.

-N *priority* To the extent permitted by the operating system, run the **ntpd** at a high priority.

-p *pidfile* Specify the name and path to record the **ntpd** 's process ID.

-P Override the priority limit set by the operating system. Not recommended for sissies.

-q Exit the **ntpd** just after the first time the clock is set. This behavior mimics that of the **ntpdate** program, which is to be retired. The **-g** and **-x** options can be used with this option.

-r *broadcastdelay* Specify the default propagation delay from the broadcast/multicast server and this computer. This is necessary only if the delay cannot be computed automatically by the protocol.

-s *statsdir* Specify the directory path for files created by the statistics facility.

-t *key* Add a key number to the trusted key list.

-T *chroot_dir* Chroot the ntpd server process into *chroot_dir*. To use this option you have to copy all the files that ntpd process needs into the chroot directory. This option adds security only if the server also drops root privileges (see -U option).

-U *server_user* Ntpd process drops root privileges and changes user ID to *server_user* and group ID to the primary group of *server_user*.

-v *variable*

-V *variable* Add a system variable listed by default.

-x Normally, the time is slewed if the offset is less than the step threshold, which is 128 ms by default, and stepped if above the threshold. This option forces the time to be slewed in all cases. If the step threshold is set to zero, all offsets are stepped, regardless of value and regardless of the **-x** option. In general, this is not a good idea, as it bypasses the clock state machine which is designed to cope with large time and frequency errors Note: Since the slew rate is limited to 0.5 ms/s, each second of adjustment requires an amortization interval of 2000 s. Thus, an adjustment of many seconds can take hours or days to amortize. This option can be used with the **-q** option.

THE CONFIGURATION FILE

Ordinarily, **ntpd** reads the **ntp.conf** configuration file at startup time in order to determine the synchronization sources and operating modes. It is also possible to specify a working, although limited, configuration entirely on the command line, obviating the need for a configuration file. This may be particularly useful when the local host is to be configured as a broadcast/multicast client, with all peers being determined by listening to broadcasts at run time.

Usually, the configuration file is installed in the **/etc** directory, but could be installed elsewhere (see the **-c** *conffile* command line option). The file format is similar to other Unix configuration files - comments begin with a **#** character and extend to the end of the line; blank lines are ignored.

Confi guration commands consist of an initial keyword followed by a list of arguments, some of which may be optional, separated by whitespace. Commands may not be continued over multiple lines. Arguments may be host names, host addresses written in numeric, dotted-quad form, integers, fbating point numbers (when specifying times in seconds) and text strings. Optional arguments are delimited by **[ ]** in the following descriptions, while alternatives are separated by | . The notation **[...]** means an optional, indefi nite repetition of the last item before the **[...]** .

FILES

/etc/ntp/ntp.conf - the default name of the confi guration fi le

/var/lib/ntp/drift - the default name of the drift fi le

/etc/ntp/keys - the default name of the key fi le

BUGS

ntpd has gotten rather fat. While not huge, it has gotten larger than might be desirable for an elevated-priority **ntpd** running on a workstation, particularly since many of the fancy features which consume the space were designed more with a busy primary server, rather than a high stratum workstation in mind.

AUTHOR

David L. Mills <mills@udel.edu>