Maintaining accurate system time with NTP

Keeping time

Time, to a system administrator is important, especially if it is lunch time, which is only beaten by a nose by the all important system time.

BY COLIN MURPHY

The clocks built on the motherboards of your average PC are not that accurate.

The result of an inaccurate clock is a continual drift away from standard time and is considered unacceptable for machines that provide services that rely on timestamps. Since every file written is tagged with a time stamp we must then conclude that every machine should have accurate system time. Luckily, with the use of *ntp* no one needs to consider accurate timekeeping a luxury.

Standard time

The *Universal Time Coordinated*, UTC for short, is the time standard computers are set to follow, which is the result of a coordinated effort between some of the most accurate clocks on – and off – the planet. Generating a time standard from the movement of the earth was not constant enough, so GMT and times derived from it could not be used to provide microsecond accuracy.

To prevent UTC's gradual drift away from GMT, which could have left us seeing 1200 hours UTC time occurring in the middle of the night according to GMT, a leap second is added or removed from UTC as and when necessary. So, UTC and GMT are not the same thing, but they will be within a second of each other. This is important because a second is a long time in computer terms.

UTC has its home in the US Navel Observatory which passes on the responsibility of telling the time to a collection of servers around the globe, collectively known as *stratum 1* reference clock time servers, but they are usually only made available to serve the next tier of servers, *stratum 2*.

Refer to [1] for a list of such servers, but don't treat it as definitive, these things can go out of date. A good test would be to *ping* the server to double check it is still in operation. You might see if your ISP runs their own server:

```
everyone:~ # ping -c 2
ntp.demon.co.uk
PING ntp.demon.co.uk
(158.152.1.76) from 192.168.1.3
:56(84) bytes of data.
64 bytes from ntp.demon.co.uk
(158.152.1.76): icmp_seq=1
ttl=247 time=11.4 ms
64 bytes from ntp.demon.co.uk
(158.152.1.76): icmp_seq=2
ttl=247 time=10.6 ms
```

Alternatively, look at their web site for a list of servers or ask the ISP directly.

Installation & configuration

The latest version of *ntp* can be found as source tarballs at *http://www.ntp.org/* but it is a stable package, so it is most likely that if you are running an installation from a recent distribution you probably have it installed already or, at the very least, have a copy locally on disc. Should this be the case then you can just load the package in the usual way, using whatever package manager you feel comfortable with.

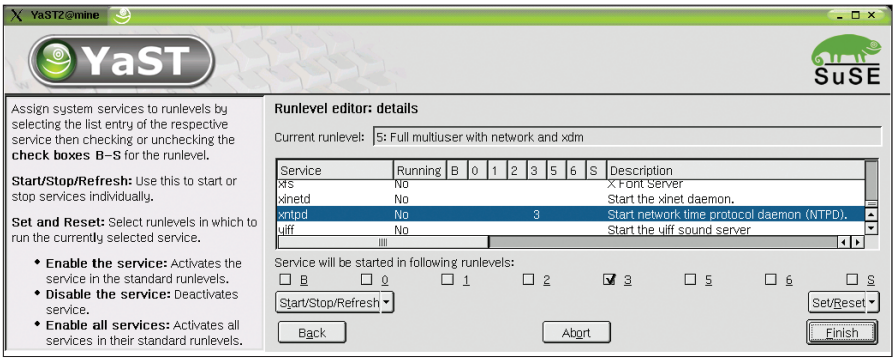

Figure 1: Using Yast to add *ntp* to runlevels

The source code tarball has comprehensive documentation regarding installation, but most people will find that './configure make make install' trinity will be all that is required.

ntp has recently gone through a version change. You may find references to *xntpd* which referes to version 3, version 4 is known as *ntpd*. Sometimes you will find a symlink pointing to *xntp* for backward compatibility. Version 4 gives better accuracy, support for features found in modern kernals like the *nanokernal* and new drivers allowing you to use different types of reference clock.

Starting at boot time

To make the most out of *ntp* it should always be running. It should be started as part of the boot procedure of your machine. How you do this depends on your system, SuSE users can use the Runlevel editor in YaST2 as shown in figure 1, others might choose to edit their *init/rc.d/* scripts by hand.

ntpd will usually be run as a daemon, so as to provide for the client side of the protocol, keeping your local system time accurate. It also provides for the server side as well which means other machine on your network can get time details from this server. When the *ntpd* deamon is started it will call upon its configuration file */etc/ntp.conf*, so it is here that you need to put details of the previously mentioned time servers.

Installing *ntp* will leave you with a well documented */etc/ntp.conf* file to which you should add the IP addresses of the servers you want to use. As a minimum you could get away with just putting a single server in this configuration file, but this really wouldn't be showing *ntp* off to its best ability. You can see a configuration file in listing 1.

Ideally you would want to include 3, maybe 4 servers so that *ntp* can compare the data from each improving the accuracy of its time keeping as it goes. *ntp* is also bright enough to look out for errors in the information it is getting from these servers looking out for values that are way off the mark by suggesting the real time is very much different to your system time. Should this happen it will treat the information coming from this server with contempt.

How wrong is your clock?

Time is supposed to flow in a regular fashion, steady and reliable. If you were setting the time on your system manually by using the 'date' command instead of taking the trouble to set up *ntp*, your system time would jump, in one step, to that new time. This could cause a system problems, especially if time appears to jump backwards, something that could confuse a database.

Thankfully *ntp* doesn't make system time jump in this harsh way, rather it allows the current, incorrect, system time to catch up or fall back until it reaches the correct time, it *slews* the time in a much more controlled way.

Before you kick *ntp* off on it's long term of service, you need to consider how accurate your current system time is, as said, if there is a great difference between your system time and UTC, something in the order of 15 minutes, then *ntp* will just think that there is some error in what it is being told. It might be wise just to correctly set the system time just once before you allow *ntp* to take over, keeping it accurate.

Starting *ntpd* from the command line with the *-q* switch will do just this:

```
every1:~ # ntpd -q
ntp.demon.co.uk
```

This procedure is very similar to using a program called *ntpdate*, which you may even find installed on your system to provide legacy support. *ntpdate* is marked for retirement now that *ntpd* has had the feature added

Now all that need to be done is to start the deamon off properly. Assuming that it has been added to an *init* runlevel then this will happen when you next reboot the machine, or when we call the *init* script from the command line:

```
every1:~ # /etc/init.d/xntpd
restart
Try to get initial date and
time via NTP from
194.117.157.4 done
Starting network time
protocol daemon
(NTPD)                done
```

This would appear to have set things off, but how can you really tell?

Looking at the system time will not help because you have either accurately set it moments before or, if you felt that your system time was close enough you have to wait until *ntp* brings it gently in line with UTC. A query program is supplied just so that we can keep an eye on the activities of *ntp*, see listing 2.

The output from *ntpq -p* shows the servers listed, which *stratum 1* server they refer to and some statistical information. As you can see, none of my *stratum 2* servers refer to the same *stratum 1* server.

The *LOCAL(0)* time server is special and refers to the *LCL* local clock described in */etc/ntp.conf* with the IP address of 127.127.1.0. Should the remote servers fail, or more likely the connection to them fail, then *ntp* will refer to itself as a time server until the situation improves.

Not enough time

When your machine is powered down the battery backed hardware clock has to take over the responsibility of keeping the time, but it is going to be just as inaccurate as the built in system clock. *ntp* helps us here by providing details of how the clock drifts away from the correct time. An offset to account for this drift is updated to */var/lib/ntp/ntp.drift*.

When the machine is powered back up *ntp* can refer to this value using it to offset the hardware clock time to something that should still be accurate enough for *ntp* to carry on with.

Listing 1: An example of */etc/ntp.conf*

```
everyone:~ # cat /etc/ntp.conf
##############################################################################
## /etc/ntp.conf
##############################################################################

## Undisciplined Local Clock. This is a fake driver intended for backup
## and when no outside source of synchronized time is available.
##
server 127.127.1.0              # local clock (LCL)
fudge  127.127.1.0 stratum 10   # LCL is unsynchronized

##
## Outside source of synchronized time
##
server 194.117.157.4    # ntp.blueyonder.co.uk
server 158.152.1.76     # ntp.demon.co.uk
server 130.88.202.49    # ntp2a.mcc.ac.uk

driftfile /var/lib/ntp/ntp.drift # path for drift file

logfile   /var/log/ntp          # alternate log file
```

Listing 2: Confirmation that *ntpd* is running

```
everyone:~ # ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 LOCAL(0)        LOCAL(0)        10 l   15   64  377    0.000    0.000   0.015
+ns2.cableinet.c ntp0.usno.navy. 2 u    9   64  377    9.055    9.409   1.575
*ntp.demon.co.uk ntp1.usno.navy. 2 u   12   64  377   13.366    4.505   0.900
xmaverick.mcc.ac ntp1.ja.net     2 u   22   64  377   14.739  264.194   1.341
```

INFO

[1] Public NTP time servers: *http://www.eecis.udel.edu/~mills/ntp/servers.html*