Synchronize your network with NTP

NTP is the *Network Time Protocol*, used to synchronize host clocks to one another. Your Slackware distribution comes with NTP preinstalled.

Your reasons for running NTP might include:

- make timestamps in system logs agree with one other, to make sense of events recorded in multiple system logs
- enable software protocols and encryption that depend on accurate time, e.g., Kerberos or PCI
- prevent software build issues caused when your remote filesystem says your source file was modified in the future
- prevent issues with database software that cannot tolerate setting the host clock back in time
- know when to stop hacking and turn on the new *Star Trek: Picard*

Using NTP, your Slackware host can become any of the following:

- An NTP *client*, that corrects its own host clock to match that of another host
- A *standalone* NTP client, that does not match its own clock to another host but does take advantage of NTP's ability to make frequency corrections when the host clock gains or loses time too quickly
- An NTP *server*, that shares its system time with other hosts
- A *primary* NTP server, that gets its system time not from another host but from a hardware clock that has direct access to the Coordinated Universal Time timekeeping process (also known as a *stratum 1 host*)

It is very common to operate NTP as both a server and a client. In other words, your Slackware host can get accurate time from another host on the Internet and then serve that time to hosts on your local network.

> This HOWTO is based on Slackware version 15.0 and the NTP reference implementation version 4.2.8 that comes with it. To check your Slackware version see: “Checking a Slackware Version”. To check your NTP software version, use the ntpd command:
>
> ```
> $ ntpd --version
> ntpd 4.2.8p15@1.3728-o Fri May 21 19:02:16 UTC 2021 (1)
> ```

Access control

NTP uses UDP port 123. Open port 123 in your host firewall if you want to allow other hosts to connect to your host. Open port 123 in your Internet firewall if you want access to Internet time.

NTP uses the `restrict` command in `/etc/ntp.conf` to impose additional restrictions by creating an ACL (*access control list*). The ACL is used by a mini-firewall within NTP itself that drops inbound packets based on options you choose.

Please turn your attention to the ACL pre-supplied by Slackware in `/etc/ntp.conf`:

```
restrict default limited kod nomodify notrap nopeer noquery
restrict -6 default limited kod nomodify notrap nopeer noquery
restrict 127.0.0.1
restrict ::1
```

This ACL prevents NTP from running as either a client (nopeer) or a server (noquery). It drops all packets except requests for basic information (first two lines). It makes an exception for packets that originate from your Slackware host itself (last two lines). This exception is what lets you control your own NTP service using the `ntpq` command.

The second line is redundant and should be deleted.

If you want to use the public NTP server pool, you must add a line to relax the ACL restrictions enough to allow peering with associations. Add a `restrict source` command without the `nopeer` flag, such as:

```
restrict source limited kod nomodify notrap noquery
```

If you want to allow devices on your network (or anywhere you like) to get time from this host, you must add a line to relax these restrictions to permit clients. Add a `restrict` *`address`* command that identifies the device(s) and/or network(s) that are allowed to get time. If your local network is `172.16.0.0/16`, you could add:

```
restrict 172.16.0.0 mask 255.255.0.0 limited kod nomodify notrap nopeer
```

To let you control your NTP service from your maintenance VLAN and not just the host itself, you might want to add the VLAN with no restrictions. Supposing the maintenance VLAN is `172.16.1.0/24`, you could add:

```
restrict 172.16.1.0 mask 255.255.255.0
```

If you want more sophisticated access control than what's described here, for example to encrypt traffic or let you authenticate for administration tasks from any host, look into the secure authentication features of NTP. See: “Authentication Support” at *The NTP Project*.

Diagnostic logging

NTP prefers to use SYSLOG for logging. There is an alternate logging feature in NTP itself that can be used instead. Looking again at the preinstalled Slackware `/etc/ntp.conf`, the alternate logging feature has already been turned on:

```
logfile /var/log/ntp
```

Not recommended. If you use the alternate logging feature, you are also responsible for managing the logfile so that it does not eventually consume all the available space in the filesystem. Not having to do this is one of the big advantages of using SYSLOG.

It is simpler to delete the `logfile` line and use Slackware's preinstalled SYSLOG package. Using SYSLOG, NTP logs warnings and errors to `/var/log/syslog`, and routine status messages to `/var/log/messages`.

If you still want to use the alternate logging feature, be sure to create the empty file and make it writable by the NTP daemon:

```
# touch /var/log/ntp.log
# chown ntp:ntp /var/log/ntp.log
```

Filtering the log

NTP lets you filter certain messages out of the log, based on the message's `class` and `type`. Currently there are four classes defined:

```
clock peer sync sys
```

and four types defined:

```
info events status statistics
```

Because the preinstalled Slackware `/etc/ntp.conf` does not customize the filter, you get out-of-box behavior. NTP will pass messages that are tagged with the `sync` class and drop all messages that are tagged with any other class.

The out-of-box behavior unfortunately filters the message associated with at least one common fatal condition (termination of the ntpd process when the clock offset exceeds its panic threshold).

If you want all available diagnostic messages logged, you should disable all filtering by class or type in `/etc/ntp.conf`:

```
logconfig =allall
```

Statistics gathering

NTP can keep a statistical record of its performance, that you can analyze to check the health of your NTP-managed clock. The preinstalled Slackware `/etc/ntp.conf` already configures the directory path for these statistics:

```
statsdir /var/lib/ntp/stats
```

But to actually collect statistics, you must create the empty directory and make it writable by the NTP daemon:

```
# mkdir /var/lib/ntp/stats
# chown ntp:ntp /var/lib/ntp/stats
```

and add a command to `/etc/ntp.conf` that identifies the statistics you want collected. The most commonly analyzed record is NTP's system clock updates in the `loopstats` file:

```
statistics loopstats
```

There are a total of eight recordtypes that NTP will keep. For information, see: “Monitoring Options” at *The NTP Project*.

At the end of this HOWTO, there is an example of charting the loopstats using the preinstalled Slackware **gnuplot** package.

As with the logfile, if you collect statistics, you are responsible for managing the statistics files so that they do not eventually consume all the available space in the filesystem.

Operating NTP as a client

The preinstalled Slackware `/etc/ntp.conf` already has commands in it that would make NTP a client of the public NTP server pool, just commented out. Here are the relevant lines:

```
#server 0.pool.ntp.org iburst
#server 1.pool.ntp.org iburst
#server 2.pool.ntp.org iburst
#server 3.pool.ntp.org iburst
```

As of NTP 4, this is no longer the recommended way to use the public NTP pool.

You should replace the multiple server commands with a single `pool` command. The command that is equivalent to the lines above is:

```
pool pool.ntp.org
```

Remember that you must also add the `restrict source` command to the ACL as described in an earlier section for this to work.

The NTP Pool Project formerly recommended using country-specific pools in the `server` commands: “you get a bit better result if you use the continental zones … and even better time if you use the country zone”. This is no longer true. They now recommend looking up the global pool `pool.ntp.org`, stating that the global pool “will usually return IP addresses for servers in or close to your country … for most users this will give the best results”.

It's not safe to trust specific individual clocks in the public NTP pool. This is why NTP looks at multiple clocks and compares them before it selects a clock to synchronize with. It's important to configure the clock selection process. Current best practice is to wait until at 3 of 4 public clocks contacted agree about what time it is. Add the command:

```
tos minclock 4 minsane 3
```

It's recommended to set NTP to associate with an odd number of pool clocks, equal to at least *minclock* + 2. If your chosen *minclock* is 4, you can calculate your target number of pool clocks as:

$\boxed{\mathit{minclock} + 2 + 1 = 7}$

(You can use a larger odd number if you wish, but 7 is adequate).

NTP counts every clock you declare explicitly in `/etc/ntp.conf`, plus the pool clocks it discovers, against its *maxclock* parameter. So to come up with the right limit, take the number you just calculated, and add 1 for each explicit clock declaration you have in `/etc/ntp.conf`, *including* your `pool` command, and use that number to set *maxclock*. For example, if you just have the one `pool` command and no other clocks declared, then

$\boxed{\mathit{maxclock} = 7 + 1}$

and you should add the command:

```
tos maxclock 8
```

You can easily double-check your clock associations using the command

```
# ntpq -n -p
```

and verify that the number of pool clocks is what you expected.

The risk that you run if you don't set *minclock*, *minsane*, and *maxclock* properly is that the NTP clock selection algorithm will get it wrong at boot time and give you inaccurate time, or even panic and exit.

Correcting for a fast or slow hardware clock

Any hardware clock runs a few parts per million too fast or too slow. Over time, NTP automatically calculates what this error is and compensates for it. It can also store its calculation in a file that it re-reads when restarted.

The preinstalled Slackware `/etc/ntp.conf` already has the necessary command in it to enable this feature:

```
driftfile /var/lib/ntp/drift
```

Operating NTP as a server

Beyond access control, there is no configuration needed to let your NTP host operate as a server and supply time to your other devices.

In fact, it is a good idea to make one host on your network the primary time server, and configure your other devices to get time from it. This reduces bandwidth on your uplink. Plus it reduces the load on the public NTP pool if you are using it.

If you have client devices that are Slackware hosts, they should not use the `pool` command. Intead they can use the `server` command and identify your primary local time host by IP address. Otherwise, they are configured much like your primary time host.

You might want your other devices to stay synchronized with your primary time host even when your uplink goes down. The way this used to work was by adding your own hardware clock as a sort of “emergency” reference clock that will keep your devices in synch with one another even without an uplink. This is the approach taken by the preinstalled Slackware `/etc/ntp.conf`:

```
server 127.127.1.0
fudge 127.127.1.0 stratum 10
```

Use of this clock driver is no longer recommended.

The local clock driver has been replaced by *Orphan Mode*. The commands above should be changed to:

```
tos orphan 10
```

Orphan mode is also useful in a closed environment, say a high-security installation, where you will not use the NTP pool or an actual UTC hardware clock and only want the devices on a network to agree on the time. It's less less helpful when you want a timely reminder to watch *Star Trek: Picard*.

If you do run in a closed environment, NTP will have no way to calculate drift of your hardware clock against a true clock. You can make your own observations of how fast or slow your hardware clock tends to drift, and manually compensate using the command:

```
tinker freq NNN
```

where *NNN* is the observed frequency error of your hardware clock in parts per million. This is mutually exclusive with the `driftfile` command so you will also have to take that command out.

Startup

At one time it was standard practice to use the `ntpdate` command to make an quick rough adjustment to the system clock and then start `ntpd`, but in this version of NTP that's no longer recommended. It is recommended instead to start `ntpd` as early as possible in the boot sequence and use the `-g` option to set the time.

This is already what Slackware 15.0 is preconfigured to do. To have NTP run at startup, make `/etc/rc.d/rc.ntpd` an executable script:

```
# chmod 755 /etc/rc.d/rc.ntpd
```

and then either reboot or start it manually:

```
# /etc/rc.d/rc.ntpd start
```

The script `/etc/rc.d/rc.ntpd` looks for the file `/run/ntpd.pid` when it is passed the `stop` or `status` option. The command to create this file is already present in `/etc/ntp.conf`:

```
pidfile /var/run/ntpd.pid
```

You may have noticed that the pathnames disagree, and that's mildly infuriating but makes no actual difference because `/var/run` is a symbolic link to `/run`.

System services that should wait to start until the clock is stable can be preceded by the `ntp-wait` command, for example databases. For example, you could conceivably edit the MariaDB section of `/etc/rc.d/rc.M` to read:

```
# Start the MariaDB database:
if [ -x /etc/rc.d/rc.mysqld ]; then
  /usr/sbin/ntp-wait -v
  /etc/rc.d/rc.mysqld start
fi
```

Monitoring NTP

You can review how the pool discovery process is working with the command

```
# ntpq -n -p
```

Here is some sample output:

```
     remote           refid      st t when poll reach   delay   offset
jitter
============================================================================
==
 pool.ntp.org    .POOL.          16 p    -   64    0    0.000   +0.000
```

```
0.000
-50.205.244.37   50.205.244.29    2 u  665 1024  377   20.524   -1.211
0.030
-65.100.46.166   .SOCK.           1 u 1000 1024  377   51.379   -4.051
0.166
+162.159.200.123 10.15.13.87      3 u  966 1024  377    1.644   +0.731
0.197
*108.61.73.243   129.6.15.28      2 u  748 1024  377   40.568   +0.372
0.119
-216.229.0.50    129.7.1.66       2 u  905 1024  377   16.545   -1.035
0.147
+162.159.200.1   10.15.13.87      3 u  556 1024  377    1.565   +0.827
0.076
+69.164.203.231  129.7.1.66       2 u  220 1024  377    0.325   -1.269
0.865
```

In a nutshell, this tells us we have 1 master pool source (type="p") plus 7 peers (type "u"). It also tells us:

- Our system peer (that we are getting our time from) is 108.61.73.243 ("*").
- We have three more peer candidates ("+") in case our system peer goes away.
- NTP is considering pruning three outliers ("-").

The `loopstats` file and other statistics files are described in detail at "Monitoring Options" at *The NTP Project*. Here's a sample and a brief explanation:

```
59760 66558.710 0.000038135 4.882 0.000265925 0.027674 10
59760 68002.750 0.000538317 4.885 0.000305204 0.025906 10
59760 69002.727 0.000199760 4.885 0.000309570 0.024235 10
59760 69437.711 0.000656689 4.886 0.000331590 0.022673 10
59760 71213.750 0.000522794 4.890 0.000313766 0.021244 10
59760 73341.750 0.000484582 4.951 0.000293812 0.029264 10
59760 75485.750 0.000486984 5.011 0.000274837 0.034780 10
59760 76374.727 -0.000069057 5.011 0.000323638 0.032534 10
59760 77579.750 0.000202697 5.012 0.000317617 0.030434 10
59760 79162.710 -0.000183575 5.011 0.000326988 0.028471 10
```

Column 1	Modified Julian Date of the observation
Column 2	Time since midnight (seconds)

You can combine and convert these to a UNIX system value for plotting, as shown below.

Column 3	Difference observed between your system clock and your time source (seconds)
Column 5	Column 3's jitter

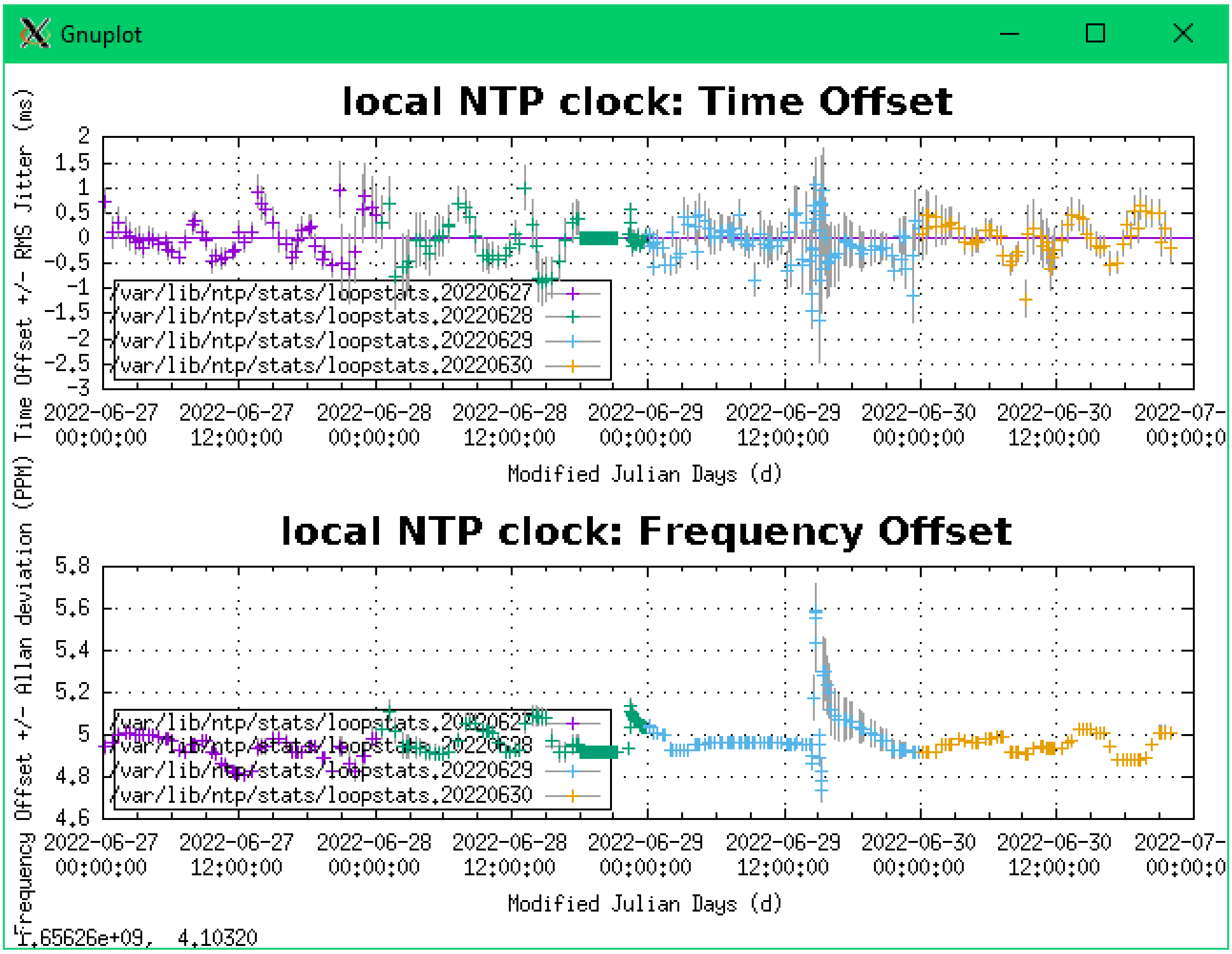
Column 4	Difference between your system clock frequency and the time source frequency (parts per million)
Column 6	Column 4's jitter

Here is a sample **gnuplot** program that charts recent loop statistics.

```
#! /usr/bin/gnuplot -ps
```

```
# Input - four most recent loopstats files
filelist=system("ls -rt /var/lib/ntp/stats/loopstats.* | tail -4")

# Output - X server
set terminal x11

# Multiplot layout
set multiplot layout 2,1

# Settings common to both graphs
set xlabel "Modified Julian Days (d)"
set xdata time
set xtics format "%F\n%T"
set xzeroaxis linetype 1
set grid
set errorbars small linecolor "dark-gray"
set key left bottom box

# Time Offset graph
set title 'local NTP clock: Time Offset' font ',20'
set ylabel "Time Offset +/- RMS Jitter (ms)"
plot [] [] for [filename in filelist] filename \
        using (86400.0*($1-40587)+$2):(1000.0*$3):(1000.0*$5) \
        title filename \
        with yerrorbars pointtype 1

# Frequency Offset graph
set title 'local NTP clock: Frequency Offset' font ',20'
set ylabel "Frequency Offset +/- Allan deviation (PPM)"
plot [] [] for [filename in filelist] filename \
        using (86400.0*($1-40587)+$2):($4):($6) \
        title filename \
        with yerrorbars pointtype 1
```

Sample output:



Sources

- Originally written by Niki Kovacs
- Performance monitoring section contributed by Dominik Drobek
- Rewritten and updated to current best practice by Edward McGuire.

howtos, time, clock, synchronization, author kikinovak, slackware 15.0, author metaed