Clock Synchronization

It is important that the system clocks in the Linux workstations of the RFPK Software Team be closely synchronized with the clock in the Linux server, whitechuck, because the CVS application on that machine as well as the backup process depends on the comparison of timestamps on files residing on different machines. This tutorial explains how to configure your workstation so that it stays synchronized within a fraction of a second with that on whitechuck.

Table of Contents

Network Time Protocol

The Network Time Protocol (NTP) allows computers connected to the Internet to keep their clocks synchronized. Certain host computers equipped with accurate clocks make the NTP service available to other, less well endowed, hosts. A machine with an accurate clock which is offering an NTP service is known as a *level one* NTP server. Other machines, which may not be equipped with accurate clocks but which nevertheless have accurate time due to the fact that they are synchronized to one or more *level one* NTP servers, can offer an NTP service over the Internet as *level two* NTP servers, and so on.

The host bigben.cac.washington.edu is a *level one* NTP server equipped with a clock that gets its time from GPS satellites. The RFPK Linux server, whitechuck, is synchronized to bigben and several other *level one* NTP servers. This document describes the way to configure your RedHat Linux 8.0 workstation to synchronize to whitechuck.

There are two reasons for synchronizing our workstations to whitechuck rather than to bigben:

- Reduce the load on bigben.
- Synchronize to the machine that we really want to be in sync with. Even if whitechuck should go out of sync with the *level one* servers because of Internet problems, for example, our CVS and backup processes would still work well because they would be comparing timestamps on files on our workstations with those on whitechuck.

Configuring RFPK Software Development Workstations

In this section, we will edit several configuration files. To do this, you will need root privilege.

We will start by putting the file `/etc/ntp.conf` under local source control:

```
su -
cd /etc
```

```
ci -l ntp.conf
```

Edit `ntp.conf` with your favorite editor, setting the default behavior to *nomodify* rather than *ignore*.

```
restrict default ignore

 should be changed to

restrict default nomodify
```

Next specify that whitechuck is a trusted time server by making this change:

```
 after

# restrict mytrustedtimeserverip mask 255.255.255.255 nomodify notap noquery
# server mytrustedtimeserverip

 insert the following pair of lines

restrict whitechuck.rfpk.washington.edu nomodify notrap noquery
server whitechuck
```

After making the above changes, use the **rcsdiff** command to check your work:

```
rcsdiff ntp.conf
```

Finally, we need to recreate the `/etc/ntp/step-tickers` file, which is a list of time servers queried by your system when it boots.

```
cd /etc/ntp
echo whitechuck.rfpk.washington.edu > step-tickers
echo time.nist.gov >> step-tickers
```

Daemon Setup

We need to start the ntp daemon, if it is not already running, and make sure that it is started automatically whenever the system reboots. As root, input the following commands:

```
/etc/rc.d/init.d/ntpd restart
chkconfig --level 2345 ntpd on
```

You should now check that your ntp daemon is working. Wait a few seconds after restarting ntpd, then issue this command:

```
ntpq -p
```

You should see output that looks something like this:

```
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 LOCAL(0)        LOCAL(0)        10 l   63   64  377    0.000    0.000   0.008
*whitechuck.rfpk bigben.cac.wash  2 u  192  256  377    0.684    0.075   0.024
```

I believe that this shows that there are two time servers, the dummy local server to be used in case the network is down, and whitechuck. It shows that whitechuck gets its time from bigben, whitechuck is a level 2 server, it is remote, it was last queried 192 seconds ago, it is queried every 256 seconds, a measure of its estimated "reachability" is 377, there is a delay of 0.684 milli-seconds between the time it is queried and the time the response arrives, an offset of 0.075 milli-seconds was applied in order to synchronize the clocks and an estimate of the random error is 0.024 milli-seconds.

If you do not get comparable numbers (for example, the jitter is 0.000) there is something wrong.

Configuring the RFPK Time Server on Whitechuck

The RFPK time server running on whitechuck needs to be configured somewhat differently than those running on workstations.

Just as for workstations, the default should be edited to read

```
restrict default nomodify
```

The list of trusted servers, should read as follows:

```
restrict bigben.cac.washington.edu nomodify notrap noquery
restrict time.nist.gov nomodify notrap noquery
restrict tick.uh.edu nomodify notrap noquery
restrict tick.usno.navy.mil nomodify notrap noquery

server bigben.cac.washington.edu
server time.nist.gov
server tick.uh.edu
server tick.usno.navy.mil
```

Unlike the workstation case, the server must be opened to clients. Just following the commented CLIENT NETWORK section, add the following

```
restrict 128.95.35.0 mask 255.255.255.0 notrust nomodify notrap
```

which enables service requests coming from any of the 256 IP addresses starting 128.95.35, which is the range encompassing the IPs for workstations located in the lab at AERL 241.

Additional lines, similar to the one above, can be added to accommodate RFPK machines with IP addresses that fall into other address ranges.

Configuring Time Service for the Cluster

The gateway node should be configured to be the same as whitechuck, except that bigben.cac.washington.edu should be replaced by whitechuck.rfpk.washington.edu and the range of client IP addresses should be "192.168.1.0 mask 255.255.255.0".

Each of the other nodes should be configured as workstations, except that the trusted server specification should be the following:

```
restrict 192.168.1.1 mask 255.255.255.255 nomodify notrap noquery
server 192.168.1.1
```