Ntimed — A NTPD replacement

Poul-Henning Kamp
phk@FreeBSD.org
phk@Varnish.org
@bsdphk
* What is Ntimed going to be
* What is Ntimed right now
* Why did the world need Ntimed
* What’s wrong with NTPD?
Ntimed — what’s the plan?

* Ntimed-client -- "steer my clock"
  Tiny, easily portable, DWIM.
* Ntimed-slave -- "relay time service"
  Lightweight, robust, resilient, policy.
* Ntimed-master -- "primary time service"
  The full monty.
* License = BSD 2-clause
Ntimed-master -- "primary time service"

Target:  Time-nuts, Time-lords &c

Task:  Turn time-machinery (GPS, Atoms, Quasars) into Network time suppliers

Protocols:  NTP (later: PTP)

Size:  < 30KLOC

Status:  Planned
Ntimed-master architecture

* A program for experimental science

* Python for high-level science-bits (see: GNUradio — it works)
  Clock-selection
  Clock-discipline/PLL
  Clock-modelling
  Policy

* Real time and protocol bits in C
  Security, Performance etc.

* Sandboxing
  Refclock code in separate "jail" processes
  Refclocks in any language you like
Ntimed-slave -- "relay time service"

Target:  2-3 per datacenter/ISP/VPN/...

Task:  Import time-service into environment

Protocols:  Outside: NTP  Inside: NTP (PTP ?)

Size:  < 20KLOC

Status:  33% (=ntimed-client)
Ntimed-slave architecture

1 thread -> acquire time
   = Ntimed-client
   Possibly: + more policy controls

1 thread per interface -> deliver time

1 CLI thread for operation/monitoring

"thread" likely "sub-process" for jail/security

Focus: Operations, Statistics & Monitoring
   ie: Spot clients with wrong time.
Ntimed-client -- "steer my clock"

Target: All computers in the world

Task: Put the system right on time

Protocols: NTP (later: PTP)

Size: < 10KLOC

Status: Prerelease
Ntimed-client architecture

Single thread, TODO list scheduling

Components:
  Server management
    DNS, which servers, how many servers.
  Clock Estimation
    Based on triangular pdfs
  Clock steering
    Adaptive PLL
  Kernel Interface
    Get(), Step(), Steer(), Sleep()
  Leap Second mitigation
    If, When, How

Green Computing
Single- vs. Multithreading

I’m a big fan of multithreading
   My other projects are FreeBSD and Varnish

But Ntimed basically does:

```c
while (1) {
    sleep(x);
    send_packet();
    receive_packet();
    do_math();
    if (needed)
        adjust_kernel_clock();
}
```
Ntimed-client security calculus

Privileged Interactions:
  Adjust kernel timescale

Unprivileged interactions:
  Send & Receive UDP packets
  Write logfiles
  Send syslog messages
Ntimed-client attack surface

NTP packets are 48 bytes, fixed format & numerical
  -> no scope for string based exploits

Numbers are in integer format
  -> no scope for IEEE-754 exception exploits

All RX packets discarded, except one reply for each packet we send.
  -> DoS surface/loading is minimal
Sandboxing is not free

Adds complexity
  Create trusted channels between jails

Sandboxes scale badly with portability
  fork(2) + setuid(2) + chroot(2)
  jail(2)
  MAC(2)
  POSIX Acls
  CAPSICUM
  Solaris Privileges
  SELinux
  Windows ?
Ntimed-client is not sandboxed

Cost/Benefit analysis came out negative.
(This decision will be revisited periodically)

If UNIX kernel-timekeeping was file-desc based
  ie: /dev/kernel_time

Ntimed-client could just drop privs after open.
Server Management

DNS, which servers, how many servers

Used servers: Fast poll, unused: slow poll

If DNS returns 10 servers, which do you use?

What happens when DNS response changes?

Incomplete.

Discussions ongoing with pool.ntp.org
Clock Estimation

We have two timescales, we think they are the same

local clock

t=x

remote clock
Clock Estimation

Let's send a packet and ask the other guy

We know: \( t=1 \leq (t=2, t=3) \leq t=4 \)
Clock Estimation

Ok, so that wasn’t so precise...

$t = x$

$t_1 = t_2$

Local ahead

"somewhere in the middle"

$t_3 = t_4$

Local behind
Clock Estimation

Triangular Probability Distribution Functions

local clock

remote clock

$t = x$

Local ahead

Local behind

Probability
Clock Estimation

Lower TTL = sharper TPDF
Clock Estimation

Ask N servers get N replies, do math...
Clock Estimation

![Clock Estimation Graph]

- DK
- US
- NZ
- LAN
Clock Estimation

Best server
Combined estimate
Clock steering

Adaptive PLL

Computer clocks are strange beasts
Not built for timekeeping

Routinely travel in time/space
* VM’s migrating to different hardware
* Suspend/Resume
How time-nuts treat Quartz Crystals

Photos: Steve Smith, G8LMX
How PCs treat Quartz Crystals

Crystal, (5 cents)
How PCs treat Quartz Crystals

Crystal, (5 cents)

100 W variable and unpredictable electrical heater
Clock steering
Kernel Interface

Deliberately kept minimal for portability

Get() -- Tell me the time
    clock_gettime(3) / gettimeofday(3)

Step() -- Set the time (right now!)
    clock_settime(3) / settimeofday(3)

Steer() -- Adjust the rate of time (frequency)
    ntp_adjtime(3)

Sleep() -- Wake me up later
    sleep(3)/usleep(3)
Leap Second Mitigation
If, When, How.
NTP servers are historically bad at this
Limited room for client creativity
”Leap-Smear” is *NOT* a client activity
To authorities responsible for the measurement and distribution of time

UTC TIME STEP
on the 1st of July 2015

A positive leap second will be introduced at the end of June 2015. The sequence of dates of the UTC second markers will be:

- 2015 June 30, 23h 59m 59s
- 2015 June 30, 23h 59m 60s
- 2015 July 1, 0h 0m 0s

The difference between UTC and the International Atomic Time TAI is:

- from 2012 July 1, 0h UTC, to 2015 July 1 0h UTC : UTC-TAI = - 35s
- from 2015 July 1, 0h UTC, until further notice : UTC-TAI = - 36s

Daniel Gambis
Head
Earth Orientation Center of IERS
Observatoire de Paris, France
Leap Second Mitigation

Pondering DNS based ”Bulletin-C service”

$ dig bulletin-c.example.com
bulletin-c.example.com 86400 IN A 244.20.141.253 [...]

1111 + [y×12+m] + [dut1] + [leap] + [crc8]
w=4       w=9       w=7       w=2       w=8

244.8.140.197  -> @ y2015m01  dut1=35  +0
244.20.141.253 -> @ y2015m07  dut1=35  +1
244.8.144.63   -> @ y2015m12  dut1=36  +0

Portable client: Only getaddrinfo(3) needed
Green computing considerations

2014Q2 server sales: 2 million
Assume 100W per server
Assume 25% runs Ntimed-client
Assume Ntimed-client uses 0.1% of resources

\[
2e6 \times 0.25 \times 0.001 = 500 \text{ servers } 100\%
\]

\[
500 \text{ servers } \times 0.1kW = 50 \text{ kW}
\]

\[
50 \text{ kW } \times \frac{1}{2} \text{ year } = 220 \text{ MWh}
\]

\[
200 \text{ Mwh } \approx 110 \text{ t CO}_2 \text{ emissions}
\]
What is Ntimed right now?

Ntimed-client prereleased at github:

https://github.com/bsdphk/Ntimed

Works, but missing:
  Server mgt.
  Leap second mitigation

$ cat *.ch | wc -l
  4669

Written in "Varnish Style":
  Max paranoia (356 lines contains asserts)
  FlexeLint clean
What is Ntimed right now?

Portability:

- Known good: FreeBSD, Various Linuxen
- Known bad: OS/X (kernel support)
- Not quite clear: Solaris, NetBSD, OpenBSD

$ time sh configure
Found bsd.prog.mk, will use it.
Makefile generated, remember to run 'make depend'
0.000u 0.011s 0:00.01 100.0% 0+0k 1+0io 12pf+0w
Why did the world need Ntimed?

Short answer:

HEARTBLEED

Long answer:

Critical FOSS projects are understaffed, overworked, and unable to do a competent job.

Post-HEARTBLEED The Linux Foundation spotted the NTPD project as one of these, and threw some funding at the problem.

... or rather: At Harlan and me.
So why didn’t you just fix NTPD?

I tried, I really tried!

But...

```
$ find . -name '*.[ch]' -print | wc -l
  828
$ find . -name '*.[ch]' -print | xargs cat | wc -l
  363194
```

I spent many weeks trying to find out where to stick the knife in...
NTPD is doomed

I could have renovated NTPD, but it would not be cost or time efficient.

Many advantages to a fresh start:

- Eradicate the many woo-doo workarounds
- Eliminate outdated assumptions
- Lay down good security architecture up front (rather than it being the far end goal)
Is NTPD safe?

Right now? Yes, I think so.

In the long term? No.
What’s wrong with NTPD

Copyright (C) 1970-2014 The University of Delaware

If 1970 is correct: (Harlan ?)
Led Zeppelin IV (1971)
Last Sean Connery James Bond Movie (1971)
Muppet Show didn’t exist for another four years

RFC778 18 april 1981
Indiana Jones: Raiders of the Lost Ark
Das Boot
Suzanne Vega ”Toms Diner”
Jean-Michell Jarre ”Magnetic Fields”
Kraftwerk ”Computer World”
Electric Light Orchestra ”Time”
ABBA ”The Visitors”
It runs on PDP/11 with FUZZBALL OS

Initially it made sense to have one big program
NTPD has grown and grown and grown...
Lots of contributor code with approx 1 user.
Refclocks for stuff eBay has never heard of
It just got out of hand...
How do you even test this?

NTPD used to have a simulation mode.

Could test some of the math.

I tried to resurrect it, but it had been buried in well intentioned changes.

Probably because only Dave and I ever used it...
NTPD was Daves program

And he cares a lot about timekeeping...

That is why I managed to get the "nanokernel" past his review and into NTPD

But he doesn’t care about other stuff...

Which is why none of my other patches made it.
The problem with saints...

Dave failed to arrange a succession as his eye-sight deteriorated.

Harlan Stenn tried to hold the bits together
Created Network Time Foundation
... Which kept NTPD alive and ticking
... on life-support.
A personal note of thanks