The Nanokernel

David L. Mills
University of Delaware
http://www.eecis.udel.edu/~mills
mailto:mills@udel.edu

Sir John Tenniel; *Alice's Adventures in Wonderland,*Lewis Carroll

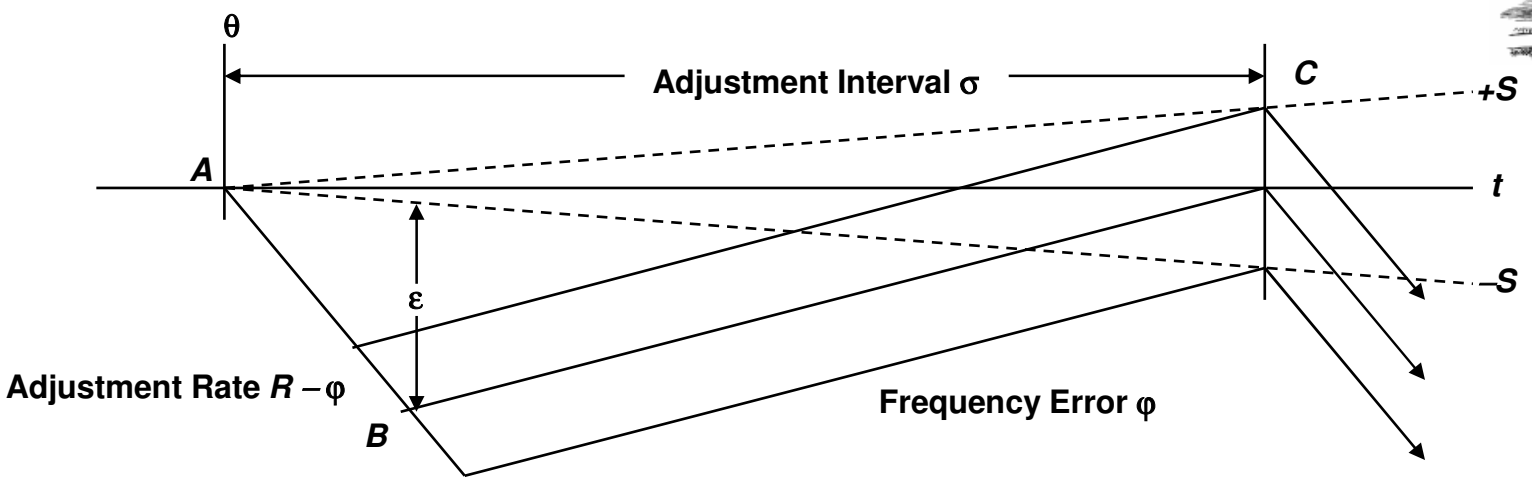
Going faster and faster

- Observing that
  - PC and workstation processor clock rates approaching and exceeding 1 GHz are becoming widely deployed
  - 10-Mb Ethernets have been replaced by 100-MHz Ethernets and these are now being replaced by 1-GHz Ethernets
  - 1.5-Mb Internet links have been replaced by 43-Mb links and these are being replaced by 155-Mb OC-3 links and soon 2.4-Gb OC-48 links
- And further that
  - Timekeeping expectations have been typically tens of milliseconds on a WAN and low milliseconds on a LAN
  - Timekeeping ambitions for some applications are in the tens of microseconds on a modern workstation
- We conclude that
  - Accuracy expectations can be considerably improved by moving the clock discipline code from user space to the kernel and, where available, precision synchronization sources

Evolution to NTP Version 4

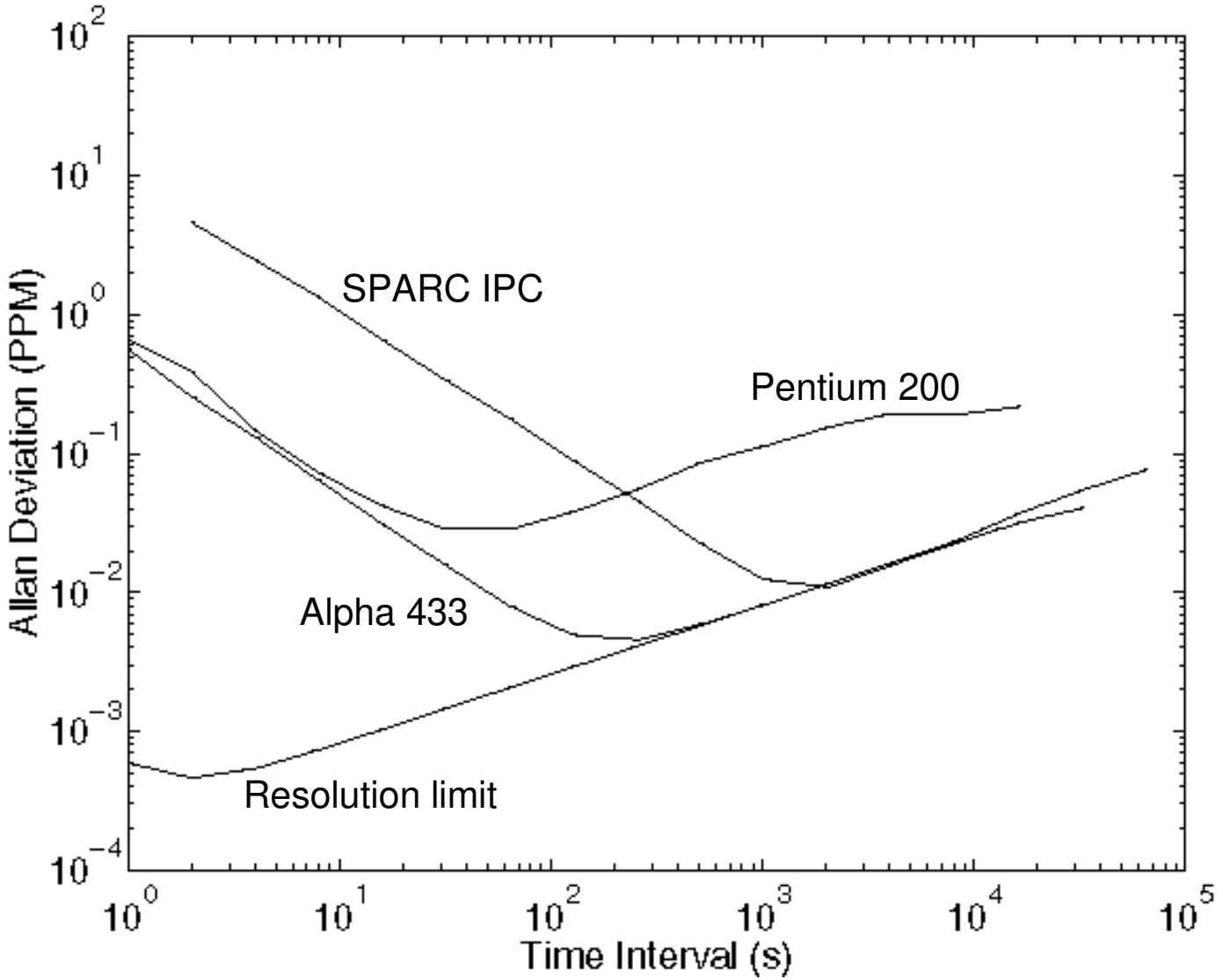
- NTP Version 3 was first used in 1992 in a rambunctious Internet of congested links and 25-MHz workstations
- NTP Version 4 architecture, protocol and algorithms have been evolved for gigabit networks and gigahertz computers
  - Improved clock models which accurately predict the phase and frequency noise for each synchronization source and network path
  - Engineered algorithms which reduce the impact of delay spikes and oscillator wander while speeding up initial convergence
  - Redesigned clock discipline algorithm which can operate in frequency-lock, phase-lock and hybrid modes
- But, the Unix time adjustment primitive prevents further improvement to the ambition of a few microseconds

Unix time adjustment primitive



- The discipline needs to steer the frequency over the range ±*S*, but the intrinsic clock frequency error is φ
- Unix `adjtime()` slews frequency at rate *R* – φ PPM beginning at *A*
- Slew continues to *B*, depending on the programmed frequency steer
- Offset continues to *C* with frequency offset due to error φ
- The net error with zero steering is ε, which can be several hundred μs

Computer clock modelling

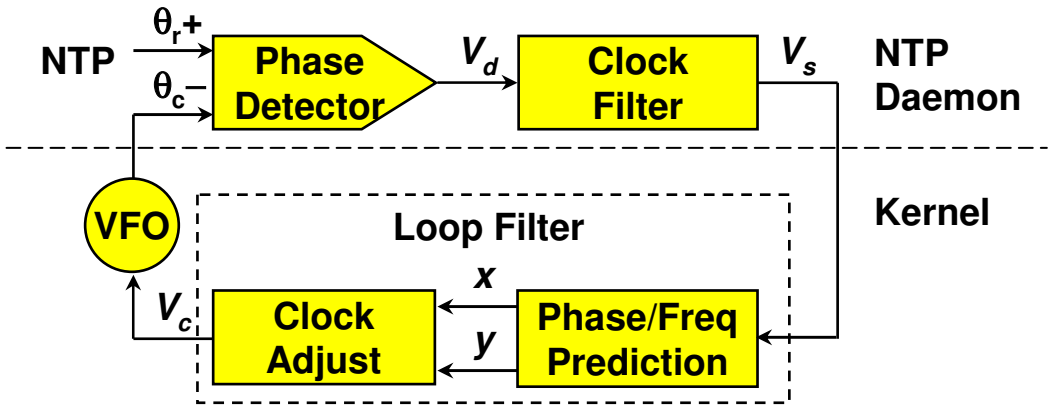


Allan intercepts compared

System	Resolution	Precision	Stability	*x* Intercept	*y* Intercept	Range *
SPARC IPC	1000 ns	1000 ns	good	2000 s	.01 PPM	600 - 5000 s
Pentium 200	1 ns	5 ns	poor	50 s	.03 PPM	10 - 300 s
Alpha 433	1 ns	2.3 ns	good	200 s	.005 PPM	50 - 2000 s
Resolution limit	1 ns	1 ns	good	2 s	.0004 PPM	1 - 10 s

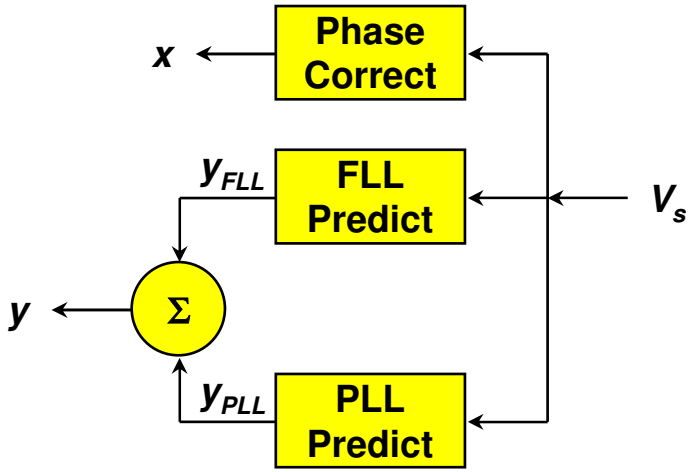
* For stability no worse than twice *y* intercept

NTP kernel clock discipline



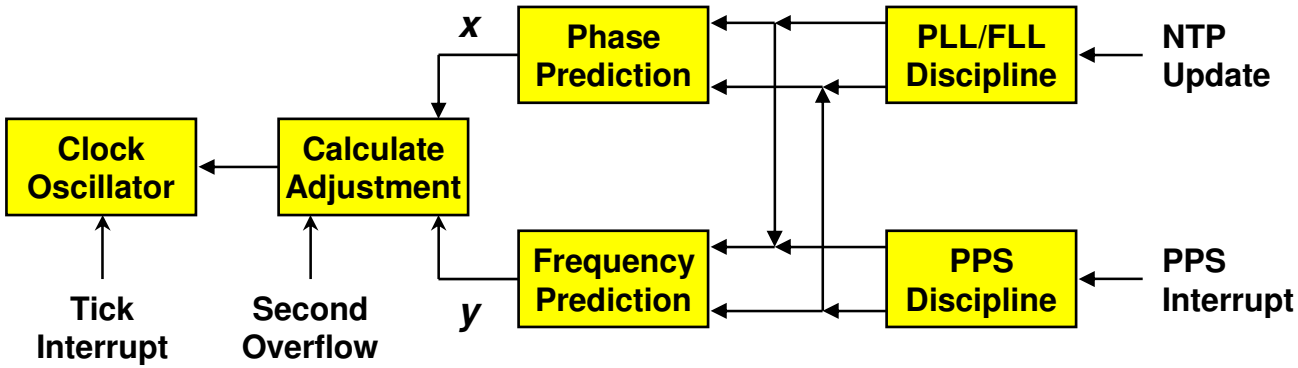
- Type II, adaptive-parameter, hybrid phase/frequency-lock loop disciplines variable frequency oscillator (VFO) phase and frequency
- NTP daemon computes phase error $V_d = \theta_r - \theta_o$ between source and VFO, then grooms samples to produce time update V_s
- Loop filter computes phase *x* and frequency *y* corrections and provides new adjustments V_c at 1-s intervals
- VFO frequency is adjusted at each hardware tick interrupt

FLL/PLL prediction functions



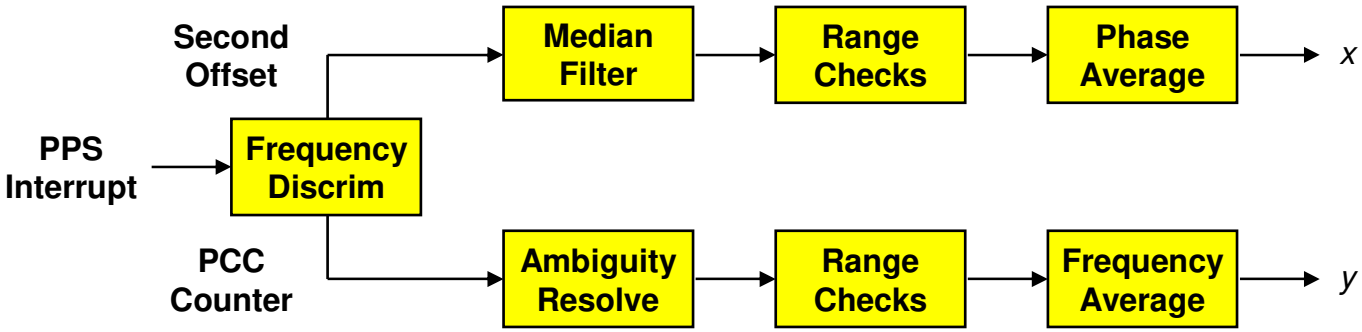
- V_s is the phase offset produced by the clock filter algorithm
- x is the phase correction computed as a fraction of V_s
- y_{FLL} is the frequency adjustment computed as the average of past frequency offsets
- y_{PLL} is the frequency adjustment computed as the integral of past phase offsets
- y_{FLL} and y_{PLL} are combined according to weight factors determined by poll interval and Allan deviation characteristic

Nanokernel architecture



- PLL/FLL discipline predicts phase *x* and frequency *y* at measurement intervals from 1 s to over one day
- PPS discipline predicts phase and frequency at averaging intervals from 4 s to 256 s, depending on nominal Allan intercept
- On overflow of the clock second, a new value is calculated for the tick interrupt adjustment
- Tick adjustment is added to the system clock at every tick interrupt
- Process cycle counter (PCC) used to interpolate microseconds or nanoseconds between tick interrupts

PPS phase and frequency discipline

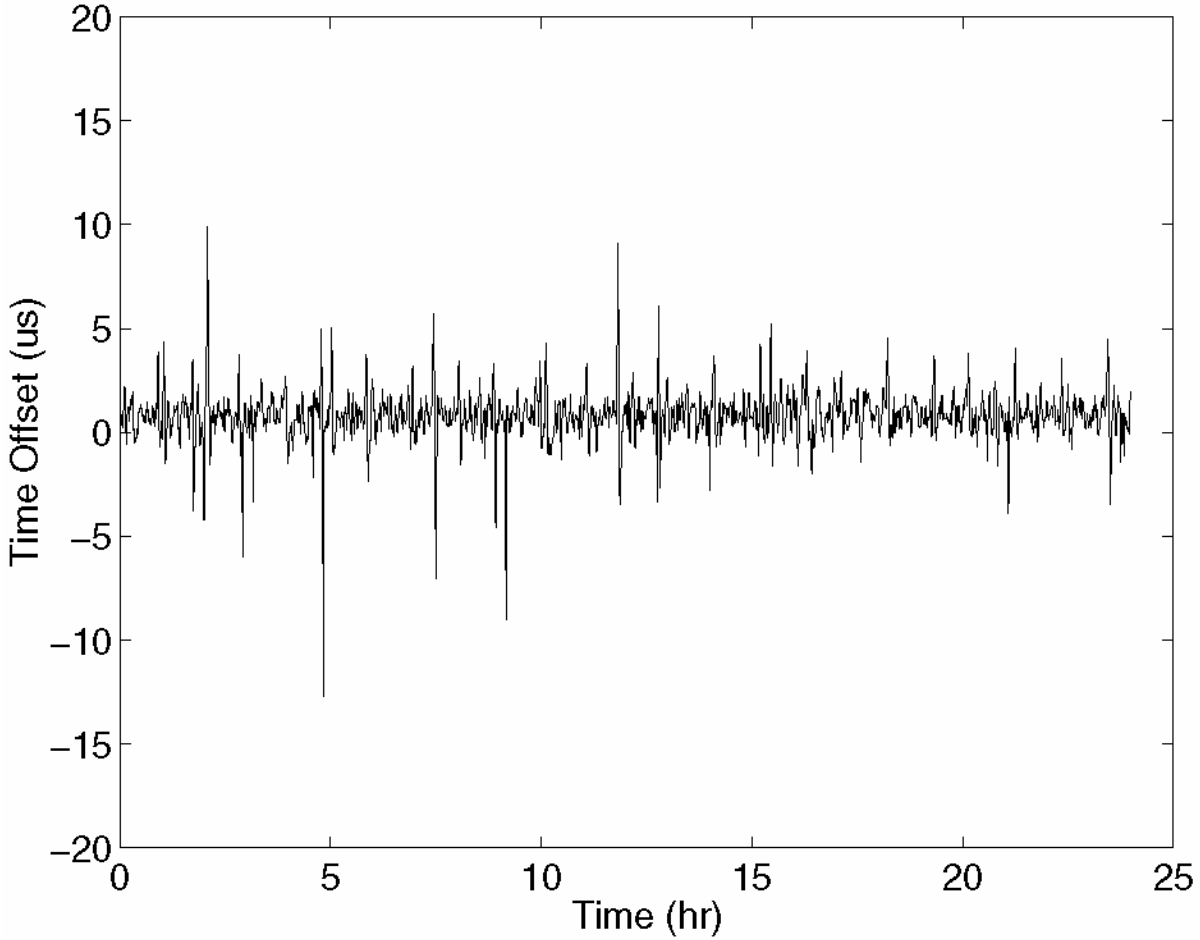


- Phase and frequency disciplined separately - phase from system clock offset relative to second, frequency from process cycle counter (PCC)
- Frequency discriminator rejects noise and incorrect frequency sources
- Median filter rejects sample outlyers and provides error statistic
- Range checks reject popcorn spikes in phase and frequency
- Phase offsets exponentially averaged with variable time constant
- Frequency offsets averaged over variable interval

Experimental results with PPS discipline

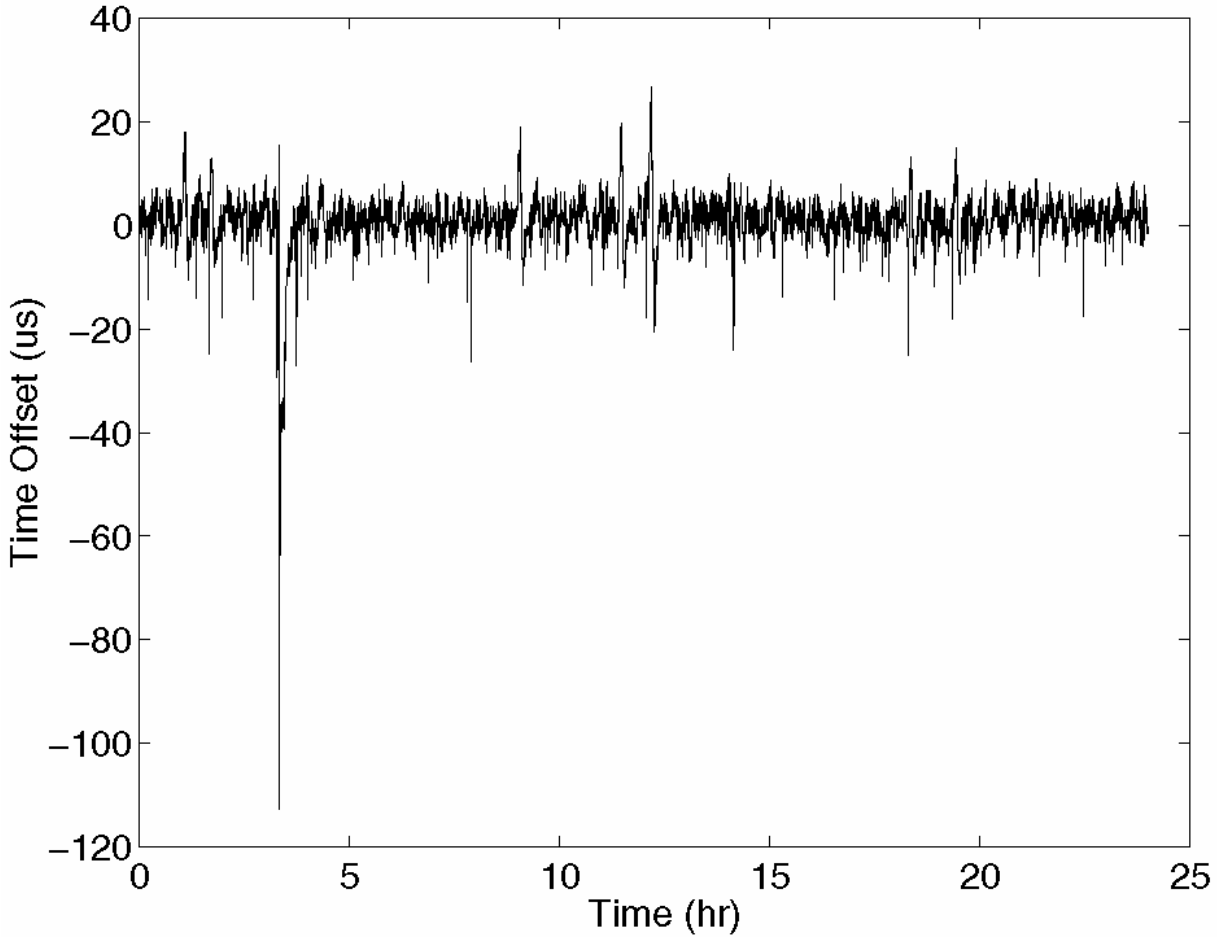
- Hepzibah is a 400-MHz Pentium workstation with a GPS receiver
  - The PPS signal is connected via parallel port and modified driver
- Rackety is a 25-MHz SPARC IPC dedicated NTP server with dual redundant GPS receivers and dual redundant WWVB receivers
  - This machine has over 1000 clients causing a load of 15 packets/sec
  - The PPS signal is connected via serial port and modified driver
- Churchy is a 433-MHz Alpha workstation with a GPS receiver
  - This machine uses a SAW oscillator presumed spectrally pure
  - The PPS signal is connected via serial port and modified driver
- All machines accessed the PPS signal from a GPS receiver and a level converter where necessary
- Experiments lasted one day with data collected by the NTP daemon

PPS time offset characteristic for Hepzibah



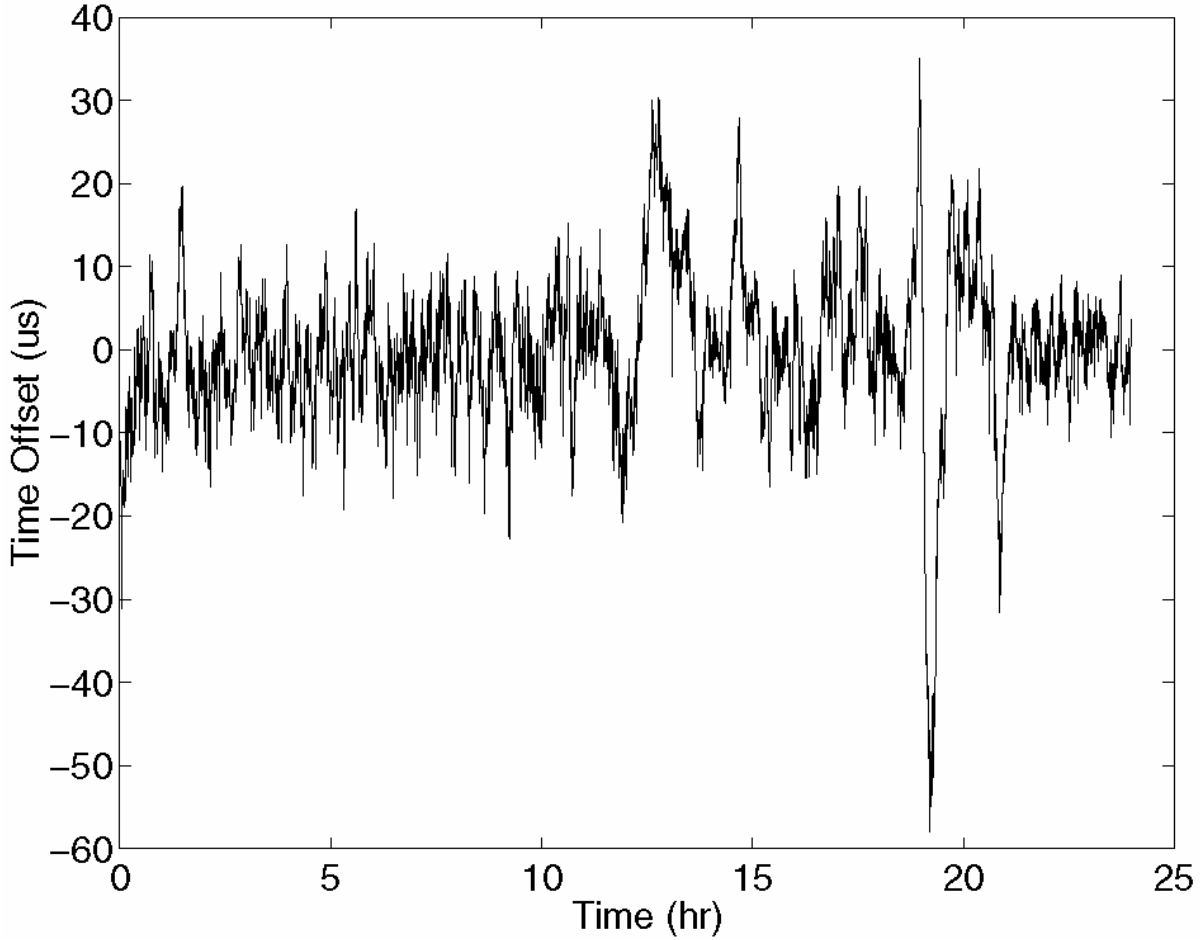
- Jitter is presumed caused by interrupt latencies on the ISA bus
- We need to explain why the spikes are both positive and negative

PPS time offset characteristic for Rackety



- Jitter is presumed caused by interrupt latencies on the Sbus
- Large negative spikes reflect contention by the radios and network

PPS time offset characteristic for Churchy



- Jitter is presumed caused by interrupt latencies on the PCI bus
- High flicker noise may be due to SAW phase noise and no PLL

Present status

- The kernel modifications have been implemented for FreeBSD, Linux, Alpha and SunOS
- The current version is in current FreeBSD and Linux public distributions
- An older version, but with resolution only to the microsecond, is in current Alpha and Solaris licensed products
- A standard PPS application program interface (PPSAPI) has been proposed to the IETF and supported in all kernels

Further information

- Network Time Protocol (NTP): http://www.ntp.org/
  - Current NTP Version 3 and 4 software and documentation
  - FAQ and links to other sources and interesting places
- David L. Mills: http://www.eecis.udel.edu/~mills
  - Papers, reports and memoranda in PostScript and PDF formats
  - Briefings in HTML, PostScript, PowerPoint and PDF formats
  - Collaboration resources hardware, software and documentation
  - Songs, photo galleries and after-dinner speech scripts
- FTP server ftp.udel.edu (`pub/ntp` directory)
  - Current NTP Version 3 and 4 software and documentation repository
  - Collaboration resources repository
- Related project descriptions and briefings
  - See “Current Research Project Descriptions and Briefings” at http://www.eecis.udel.edu/~mills/status.htm