NTP Precision Time Synchronization

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

From *pogo*, Walt Kelly

Introduction

- Network Time Protocol (NTP) synchronizes clocks of hosts and routers in the Internet
- Well over 100,000 NTP peers deployed in the Internet and its tributaries all over the world
- Provides nominal accuracies of low tens of milliseconds on WANs, submilliseconds on LANs, and submicroseconds using a precision time source such as a cesium oscillator or GPS receiver
- Unix NTP daemon ported to almost every workstation and server platform available today - from PCs to Crays - Unix, Windows, VMS and embedded systems
- Modern workstations can synchronize in principle with a resolution of one nanosecond in time and one nanosecond per second in frequency.
- This requires a comprehensive analysis and redesign of the NTP algorithms and modifications to the Unix kernel software

Needs for precision time

- Distributed database transaction journalling and logging
- Stock market buy and sell orders
- Secure document timestamps (with cryptographic certification)
- Aviation traffic control and position reporting
- Radio and TV programming launch and monitoring
- Intruder detection, location and reporting
- Multimedia synchronization for real-time teleconferencing
- Interactive simulation event synchronization and ordering
- Network monitoring, measurement and control
- Early detection of failing network infrastructure devices and air conditioning equipment
- Differentiated services traffic engineering

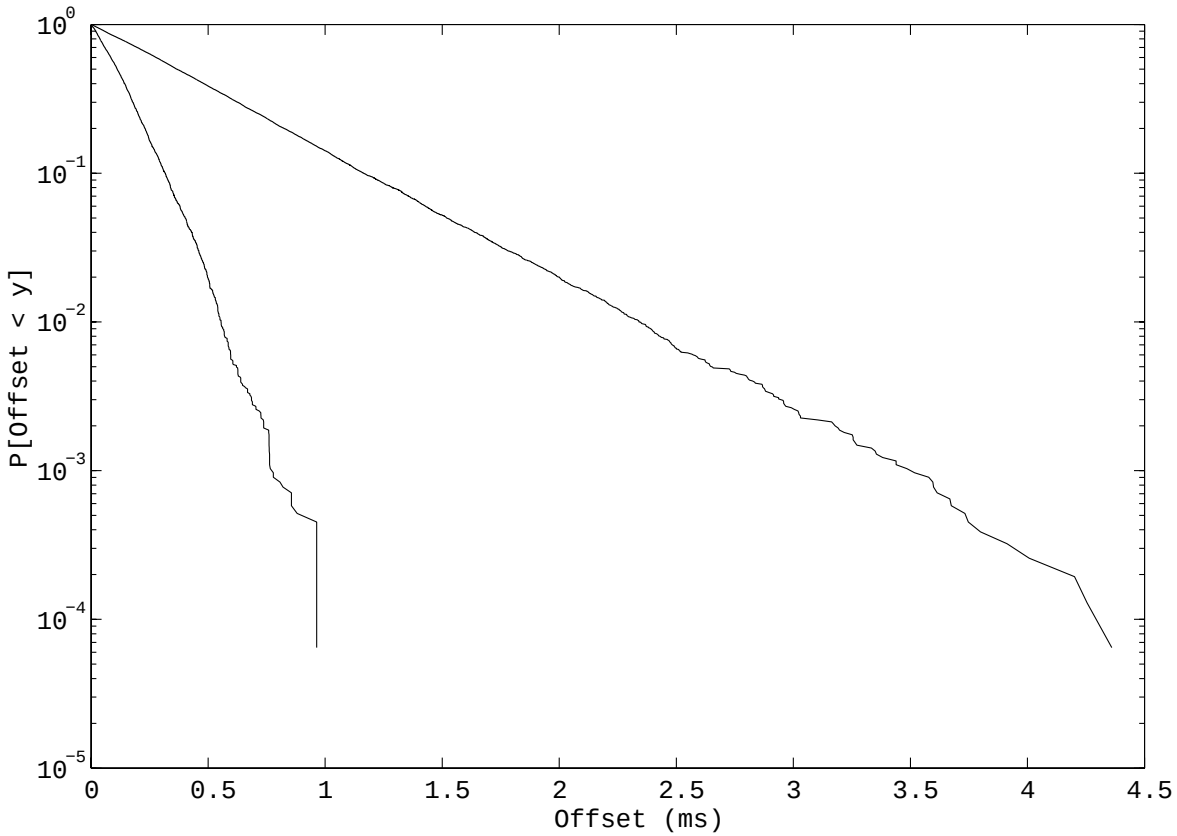
NTP capsule summary

- Primary (stratum 1) servers synchronize to national time standards via radio, satellite and modem
- Secondary (stratum 2, ...) servers and clients synchronize to primary servers via hierarchical subnet
- Clients and servers operate in master/slave, symmetric or multicast modes with or without cryptographic authentication
- Reliability assured by redundant servers and diverse network paths
- Engineered algorithms reduce jitter, mitigate multiple sources and avoid improperly operating servers
- System clock is disciplined in time and frequency using an adaptive algorithm responsive to network time jitter and clock oscillator frequency wander

Precision time performance issues

- Improved clock filter algorithm reduces network jitter
- Reduced hardware and software latencies minimize system errors.
- Engineered cryptographic authentication protocol avoids inline public-key algorithms.
- Optional kernel modifications achieve time resolution to 1 ns and frequency resolution to .001 PPM using NTP and PPS sources.
- Nonlinear hybrid adaptive-parameter (NHAP) clock discipline algorithm optimizes performance with update intervals from 16 seconds to 36 hours.
- Dynamic parameter estimation algorithm optimizes performance over a wide range of network jitter and oscillator wander.
- Temperature stabilized computer clock oscillators.
- Precision timing sources using GPS, LORAN-C and cesium clocks.

Minimize effects of network jitter

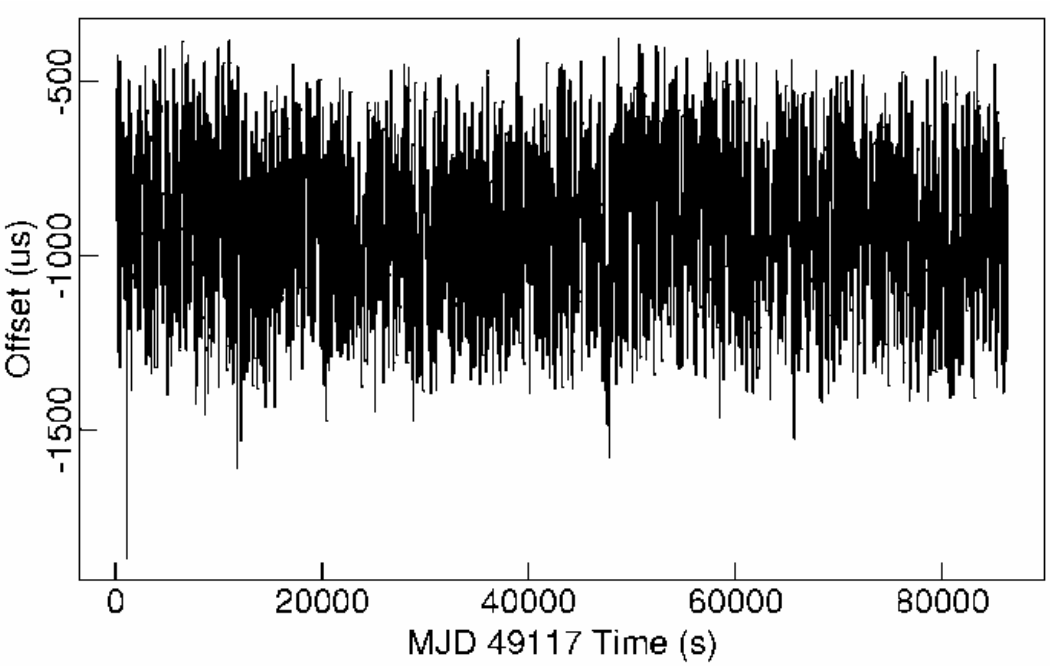


- The traces show the cumulative probability distributions for
  - Upper trace: raw time offsets measured over a 12-day period
  - Lower trace: filtered time offsets after the clock filter

Minimize hardware and software latencies

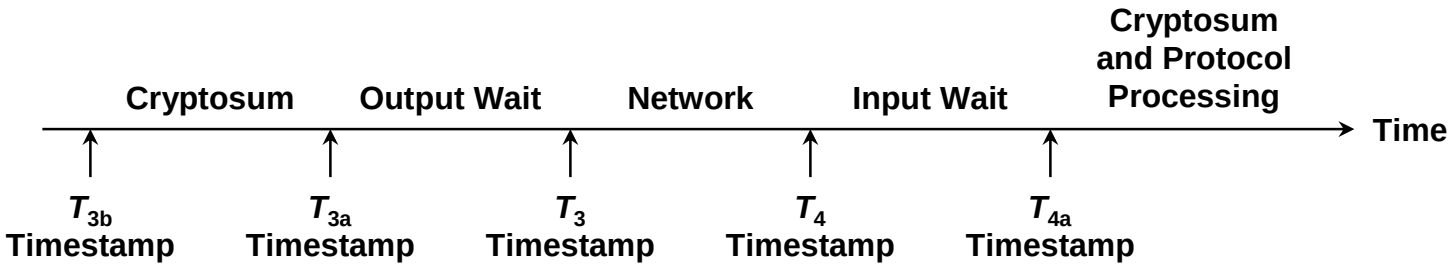
- Minimize jitter due to serial port hardware and driver
  - Capture early timestamps with special streams module/line discipline
  - Disable software driver FIFO delays
- Minimize latencies in the operating system and clock adjustment process
  - Operate at highest processor priority to reduce system queuing delays
  - Decrease the adjustment interval to avoid sawtooth errors
  - Optional protocol modifications to provide pre-cryptosum timestamp
- Minimize cryptographic algorithm variances
  - Use block-cipher algorithm such as DES or MD5 with constant running time independent of message contents and private key
  - Use public-key cryptographic authentication *Autokey* scheme to avoid inline algorithms with large variations in running times

Minimize effects of serial port hardware and driver jitter



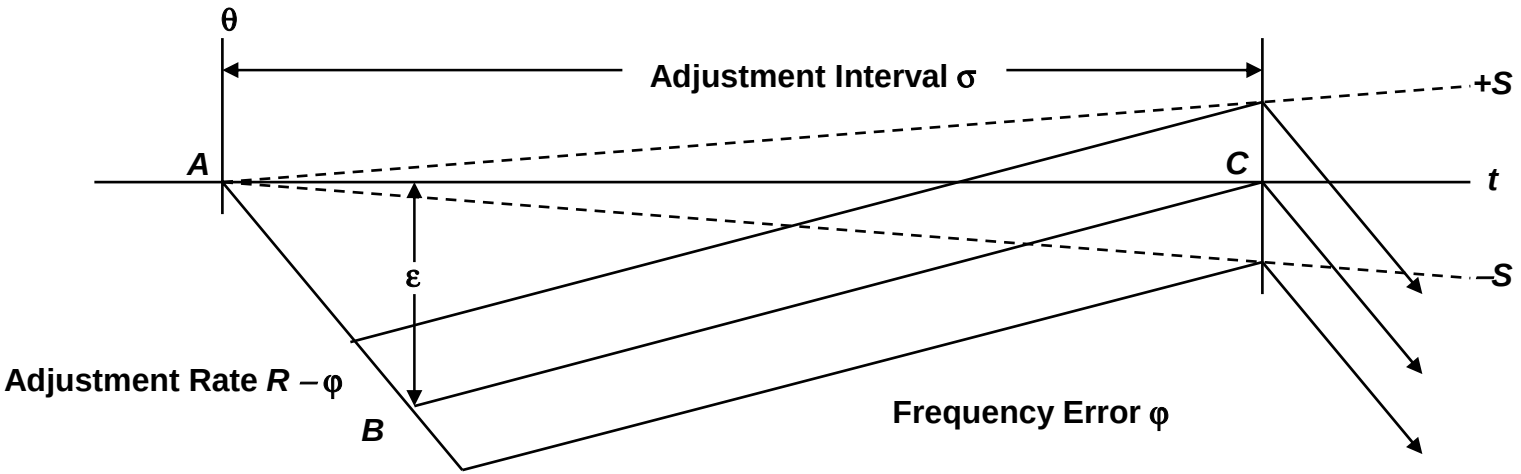
- Graph shows raw jitter of millisecond timecode and 9600-bps serial port
  - Additional latencies from 1.5 ms to 8.3 ms on SPARC IPC due to software driver and operating system; rare latency peaks over 20 ms
  - Latencies can be minimized by capturing timestamps close to the hardware
  - Jitter is reduced using median filter of 60 samples
  - Using on-second format and median filter, residual jitter is less than 50 µs

Minimize latencies in the operating system



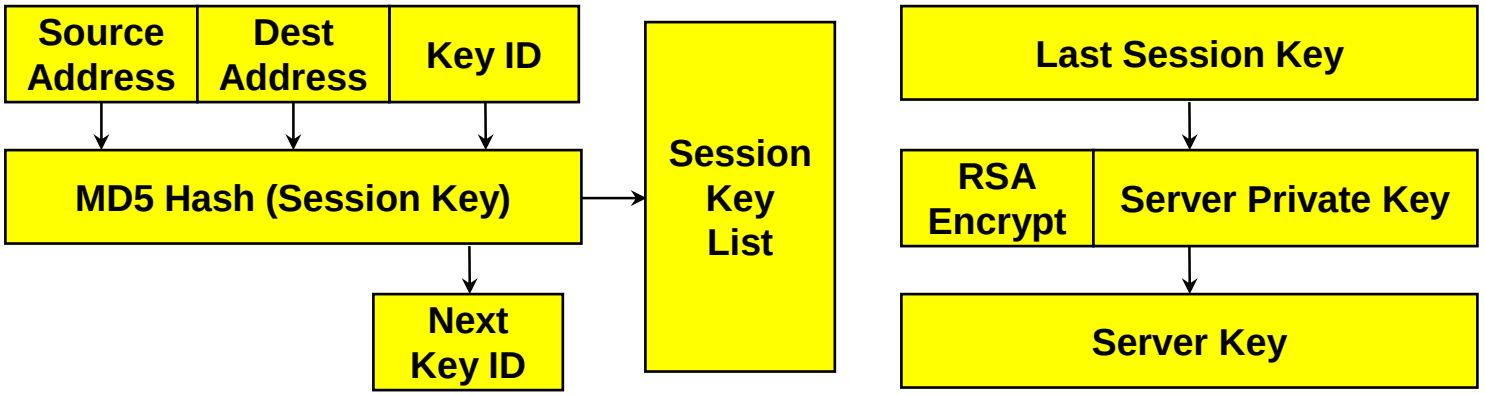
- We want T_3 and T_4 timestamps for accurate network calibration
  - If output wait is small, T_{3a} is good approximation to T_3
  - T_{3a} can't be included in message after cryptosum is calculated, but can be sent in next message; if not, use T_{3b} as best approximation to T_3
  - T_4 captured by most network drivers at interrupt time; if not, use T_{4a} as best approximation to T_4
- Largest error is usually output cryptosum
  - Cryptosum time is about 10 μs - 1 ms for DES, up to 100 ms for modular exponentiation, depending on architecture
  - Block-cipher running time can be measured and predicted fairly well
  - Actual value is measured during operation and calibrated out

Minimize latencies in the clock adjustment process



- Unix `adjtime()` slews frequency at net rate $R - \varphi$ PPM beginning at *A*
- Slew continues to *B*, depending on the programmed frequency offset *S*
- Offset continues to *C* with frequency offset due to error φ
- If $\varepsilon \le x$, then $R \ge \varphi + S$ and $\sigma \le \left(\frac{1}{\varphi} + \frac{1}{R - \varphi} \right) x$
- For ε = 100 μs, φ = 200 PPM, *S* = 200 PPM, this requires *R* ≥ 400 PPM and σ ≤ 1 s
- For best accuracy, must adjust more often than once per second

Avoid inline public-key algorithms: the *Autokey* protocol

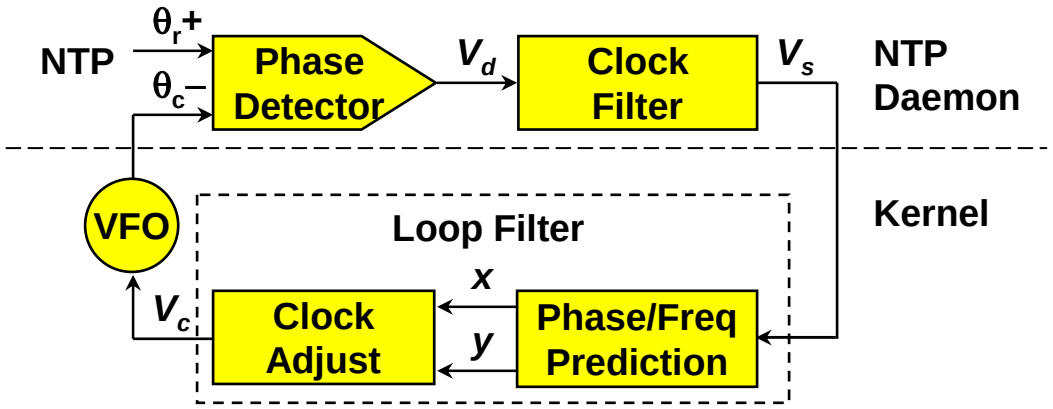


- Server rolls a random 32-bit seed as the initial key ID
- Server generates a session key list using repeated MD5 hashes
- Server encrypts the last key using RSA and its private key to produce the initial server key and provides it and its public key to all clients
- Server uses the session key list in reverse order, so that clients can verify the hash of each key used matches the previous key
- Clients can verify that repeated hashes will eventually match the decrypted initial server key

Kernel modifications for nanosecond resolution

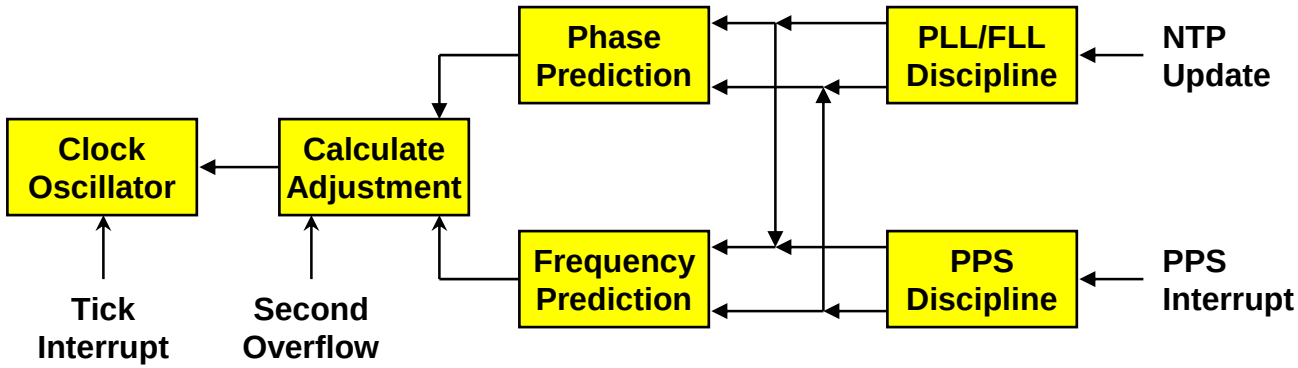
- *Nanokernel* package of routines compiled with the operating system kernel
- Represents time in nanoseconds and fraction, frequency in nanoseconds per second and fraction
- Implements nanosecond system clock variable with either microsecond or nanosecond kernel native time variables
- Uses native 64-bit arithmetic for 64-bit architectures, double-precision 32-bit macro package for 32-bit architectures
- Includes two new system calls `ntp_gettime()` and `ntp_adjtime()`
- Includes new system clock read routine with nanosecond interpolation using process cycle counter (PCC)
- Supports run-time tick specification and mode control
- Guaranteed monotonic for single and multiple CPU systems

NTP clock discipline with nanokernel assist



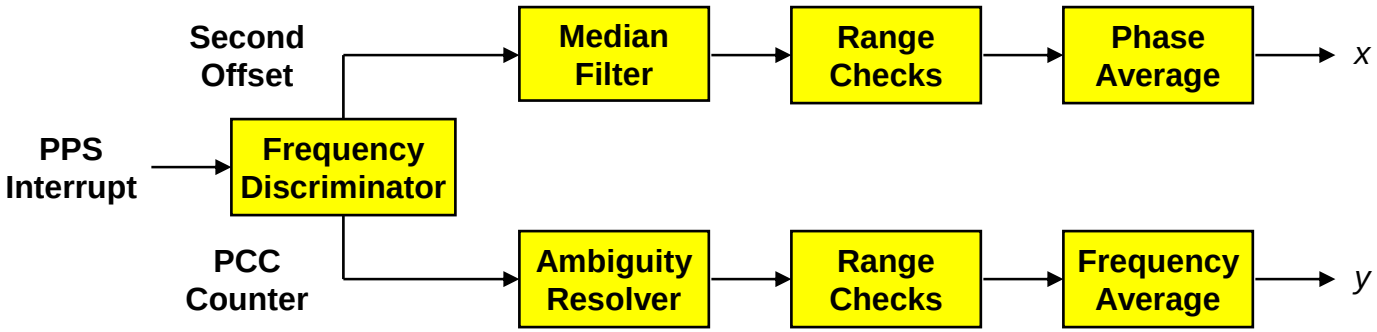
- Type II, adaptive-parameter, hybrid phase/frequency-lock loop disciplines variable frequency oscillator (VFO) phase and frequency
- NTP daemon computes phase error $V_d = \theta_r - \theta_o$ between source and VFO, then grooms samples to produce time update V_s
- Loop filter computes phase *x* and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

Nanokernel clock discipline



- PLL/FLL discipline predicts phase *x* and frequency *y* at averaging intervals from 1 s to over one day
- PPS discipline predicts phase and frequency at averaging intervals from 4 s to 128 s, depending on nominal Allan intercept
- On overflow of the clock second, a new value is calculated for the tick adjustment
- Tick adjustment is added to system clock at every tick interrupt
- Process cycle counter (PCC) used to interpolate microseconds or nanoseconds between tick interrupts

PPS phase and frequency discipline

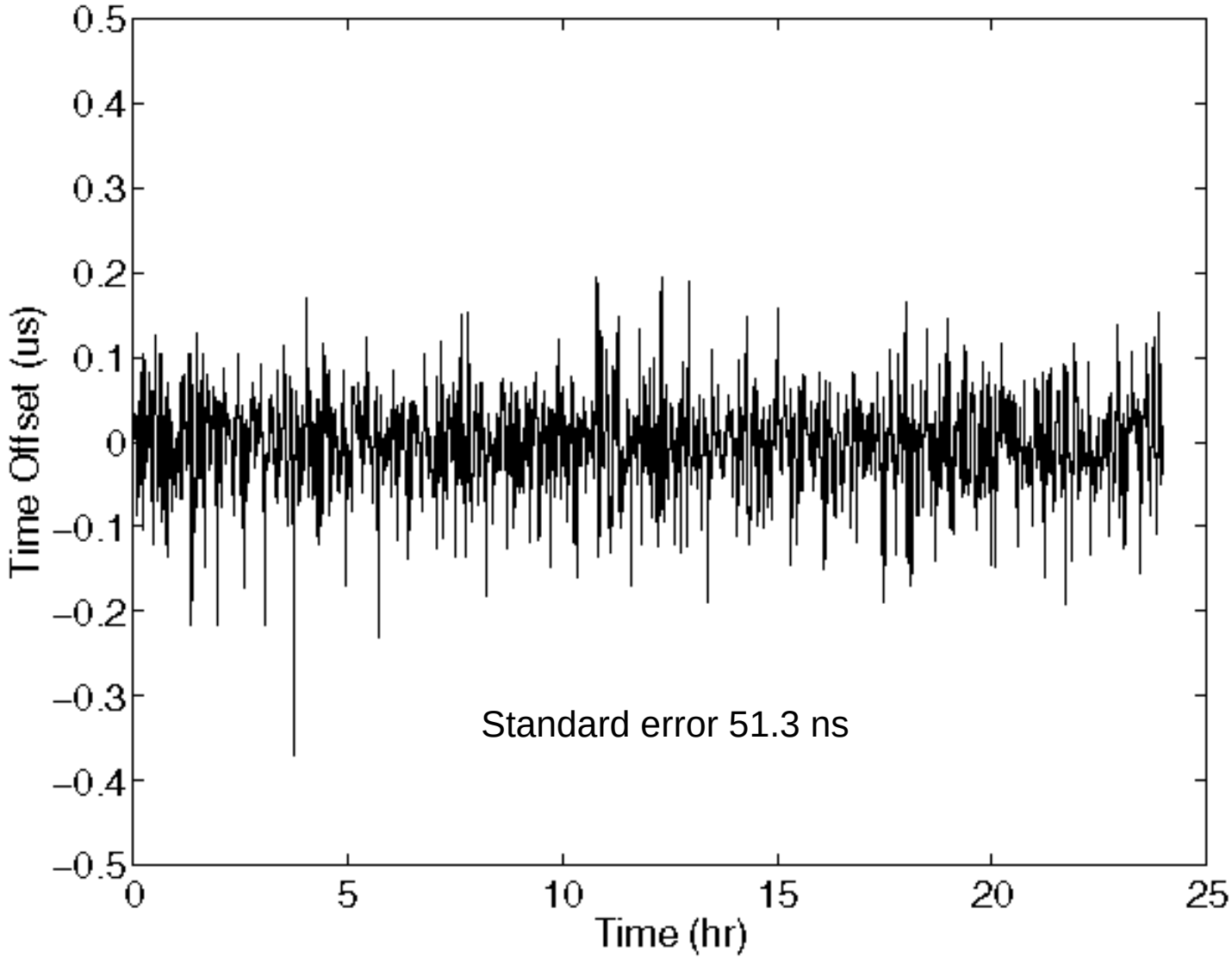


- Phase and frequency disciplined separately - phase from system clock offset relative to second, frequency from process cycle counter (PCC)
- Frequency discriminator rejects noise and incorrect frequency sources
- Median filter rejects sample outlyers and provides error statistic
- Range checks reject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

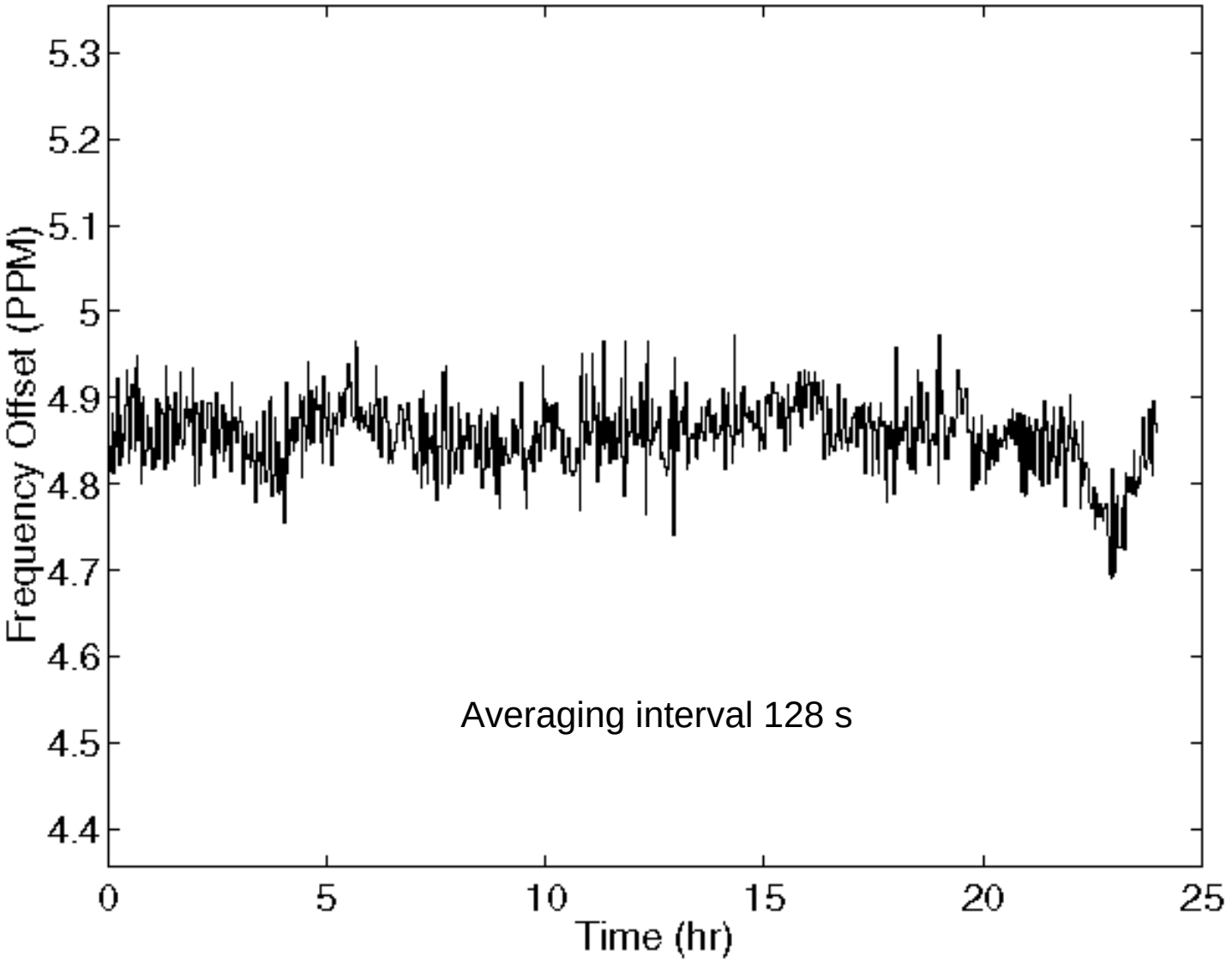
Support for multiple CPU systems

- Each processor has a read-only processor cycle counter (PCC) readable by a special instruction
- The elected master processor increments the kernel clock at each hardware timer interrupt, which occurs at intervals of 1-10 ms
- Once each second, each processor reads the master processor time and estimates the particular CPU frequency, as well as time and frequency offsets of its PCC relative to the kernel clock using an atomic interprocessor interrupt
- When the clock is read on a processor, the current time is computed from the saved master processor time plus an increment computed from the current PCC and estimated time and frequency offsets
- The kernel PLL is adjusted in time and frequency as the result of NTP updates and the PPS signal

Measured PPS time error for Alpha 433



Measured PPS frequency error for Alpha 433



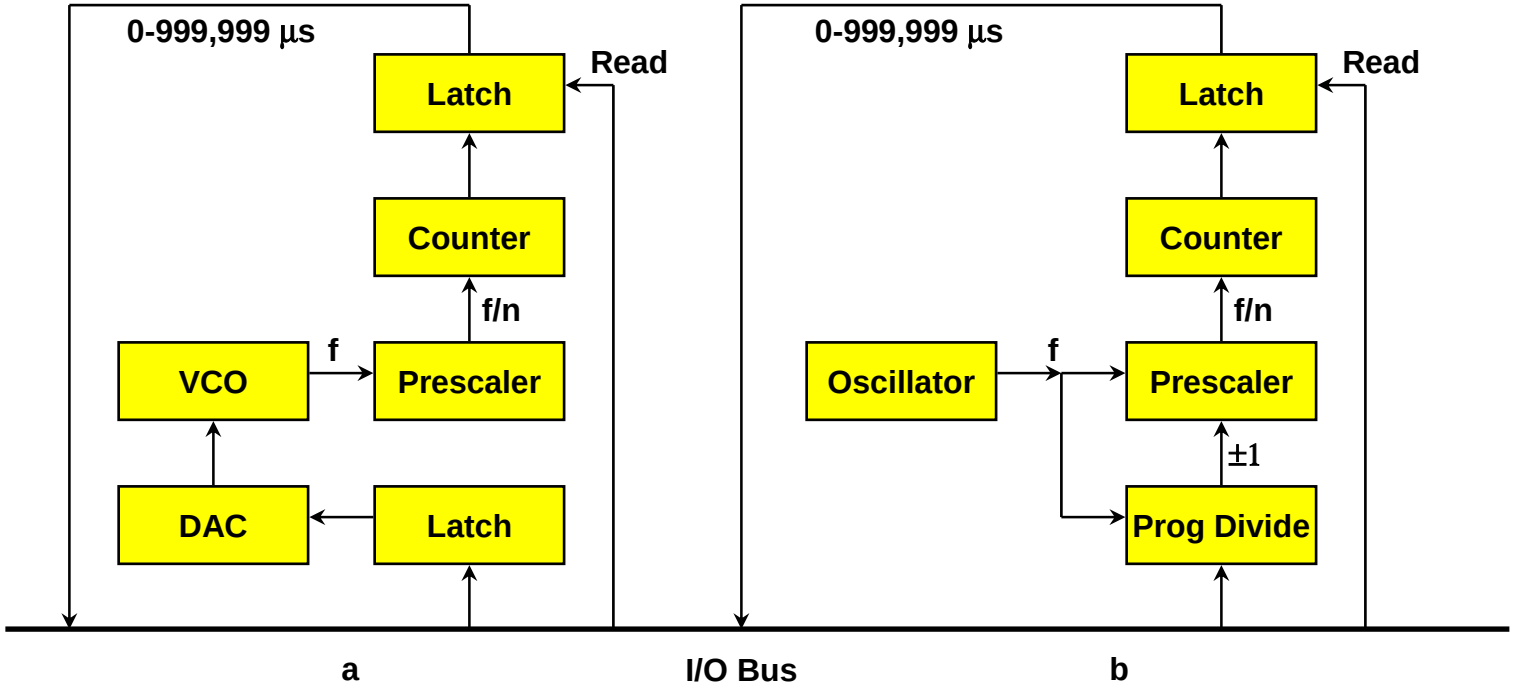
Precision time and frequency sources

- KSI/Odetics TPRO IRIG-B SBus interface
  - Provides direct-reading microsecond clock in BCD format
  - Synchronized to GPS receiver using IRIG-B signal
  - Supported both as an NTP driver and as kernel system clock
  - Stabilizes time to 1 μs and frequency to 0.1 PPM
- Precision oven-stabilized system clock
  - SBus memory-mapped interface
  - Provides direct-reading microsecond clock in Unix timeval format
  - Supported as kernel system clock
  - Stabilizes time via radio or NTP and frequency to .005 PPM
- PPS discipline
  - Driver or kernel interface via modem control line
  - Stabilizes frequency to .001 PPM relative to external 1-PPS source
  - Stabilizes time within 1 μs with seconds numbered by NTP

Gadget Box PPS interface

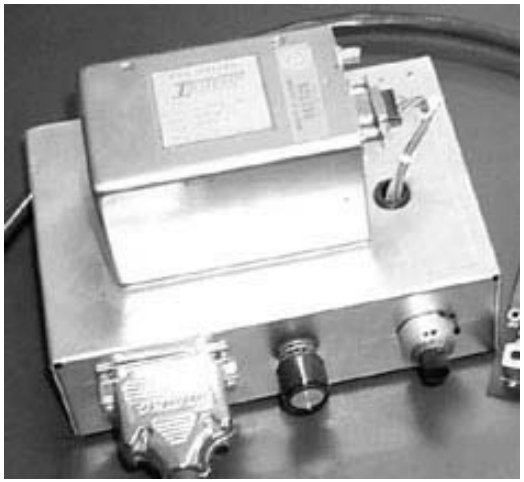
- Used to interface PPS signals from GPS receiver or cesium oscillator
  - Pulse generator and level converter from rising or falling PPS signal edge
  - Simulates serial port character or stimulates modem control lead
- Also used to demodulate timecode broadcast by CHU Canada
  - Narrowband filter, 300-baud modem and level converter
  - The NTP software includes an audio driver that does the same thing

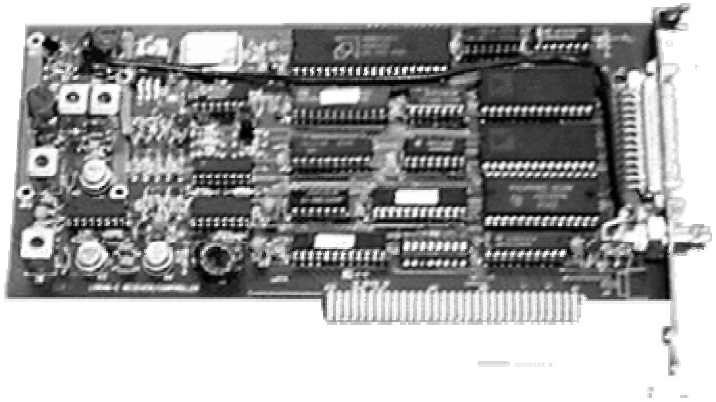
Hardware clock discipline



- Analog (a) and digital (b) frequency compensation methods
- Analog method provides precision frequency control with lowest jitter
- Digital method required with non-adjustable frequency source (e.g., cesium oscillator)
- Both methods implemented for Sun SBus module used as system clock

LORAN-C timing receiver





- Inexpensive second-generation bus peripheral for IBM 386-class PC with oven-stabilized external master clock oscillator
  - Includes 100-kHz analog receiver with D/A and A/D converters
  - Functions as precision oscillator with frequency disciplined to selected LORAN-C chain within 200 ns of UTC(LORAN) and 10^{-10} stability
  - PC control program (in portable C) simultaneously tracks up to six stations from the same LORAN-C chain
- Intended to be used with NTP to resolve inherent LORAN-C timing ambiguity

NTP online resources

- Network Time Protocol (NTP) Version 3 Specification RFC-1305
  - NTPv4 features documented in release notes and reports cited there
- Simple NTP (SNTP) Version 3 specification RFC-2030
  - Applicable to IPv4, IPv6 and ISO CNLS
- List of public NTP time servers (as of November 1999)
  - 93 active primary (stratum 1) servers
  - 109 active stratum 2 servers
- NTP Version 4 implementation and documentation for Unix, VMS and Windows
  - Ported to over two dozen architectures and operating systems
  - Utility programs for remote monitoring, control and performance evaluation
  - Complete documentation in HTML format
- Collaboration resources at
  http://www.eecis.udel.edu/~mills/resource.htm

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