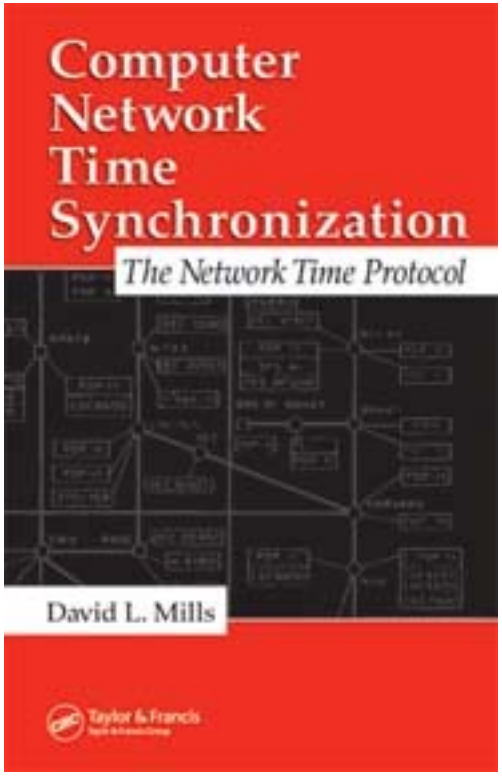
Computer Network Time Synchronization: the Network Time Protocol

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



Published by CRC Press, 2006, 304 pp.

Introduction

- Network Time Protocol (NTP) synchronizes clocks of hosts and routers in the Internet.
- NIST estimates 10-20 million NTP servers and clients deployed in the Internet and its tributaries all over the world. Every Windows/XP has an NTP client.
- NTP 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.
- NTP software has been ported to almost every workstation and server platform available today - from PCs to Crays - Unix, Windows, VMS and embedded systems, even home routers, wifis and UPSes.
- The NTP architecture, protocol and algorithms have been evolved over the last 25 years to the latest NTP Version 4 described in this and related briefings.

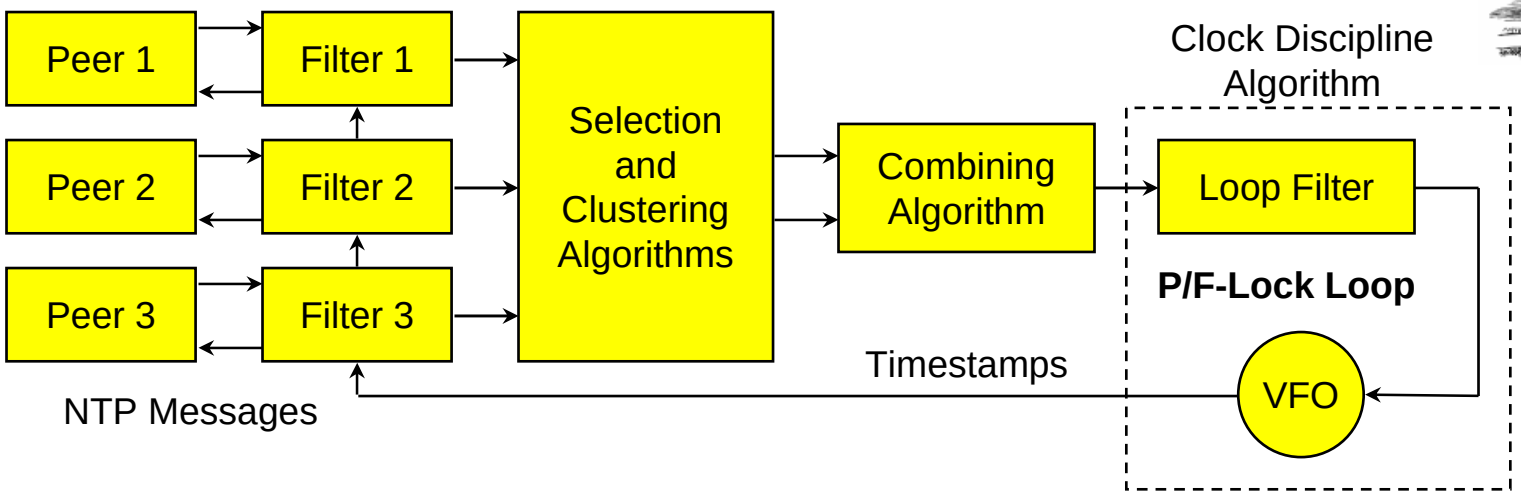
The Sun never sets on NTP

- NTP is argueably the longest running, continuously operating, ubiquitously available protocol in the Internet
  - USNO and NIST, as well as equivalents in other countries, provide multiple NTP primary servers directly synchronized to national standard cesium clock ensembles and GPS
  - Over 230 Internet primary serversare in Australia, Canada, Chile, France, Germany, Isreal, Italy, Holland, Japan, Norway, Sweden, Switzerland, UK, and US.
- Well over a million NTP subnets all over the world
  - National and regional service providers BBN, MCI, Sprint, Alternet, etc.
  - Agencies and organizations: US Weather Service, US Treasury Service, IRS, FAA, PBS, Merrill Lynch, Citicorp, GTE, Sun, DEC, HP, etc.
  - Private networks are reported to have over 10,000 NTP servers and clients behind firewalls; one (GTE) reports in the order of 30,000 NTP workstations and PCs.
  - NTP has been in space, on the sea floor, on warships and in every continent, including Antarctica, and planned for the Mars Internet.

Needs for precision time

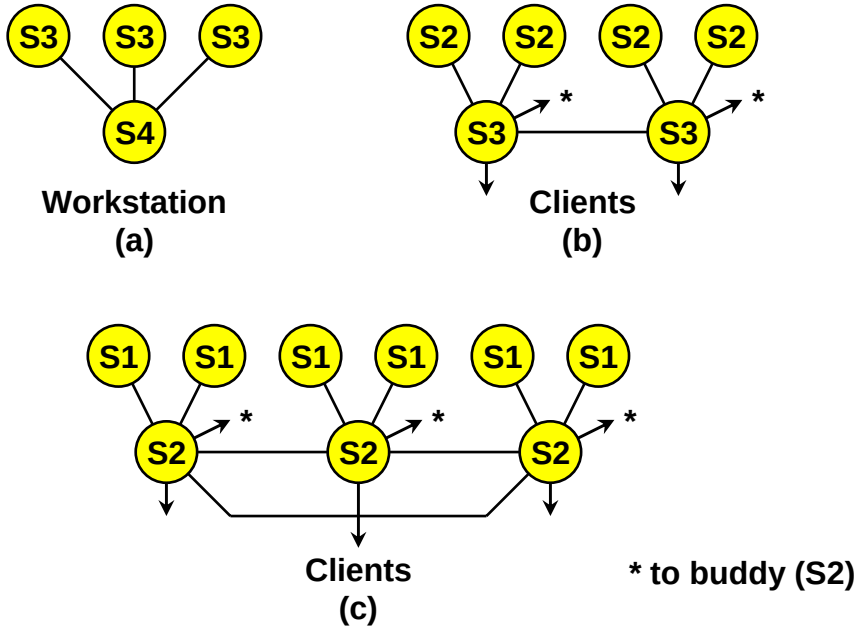
- Distributed database transaction journalling and logging
- Stock market buy and sell orders
- Secure document timestamps (with cryptographic certification)
- Aviation traffic control and position reporting
- Radio and TV programming launch and monitoring
- Intruder detection, location and reporting
- 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
- Distributed network gaming and training

NTP architecture overview



- Multiple servers/peers provide redundancy and diversity.
- Clock filters select best from a window of eight time offset samples.
- Intersection and clustering algorithms pick best *truechimers* and discard *falsetickers.*
- Combining algorithm computes weighted average of time offsets.
- Loop filter and variable frequency oscillator (VFO) implement hybrid phase/frequency-lock (P/F) feedback loop to minimize jitter and wander.

NTP subnet configurations



- (a) Workstations use multicast mode with multiple department servers.
- (b) Department servers use client/server modes with multiple campus servers and symmetric modes with each other.
- (c) Campus servers use client/server modes with up to six different external primary servers and symmetric modes with each other and external secondary (buddy) servers.

Goals and non-goals

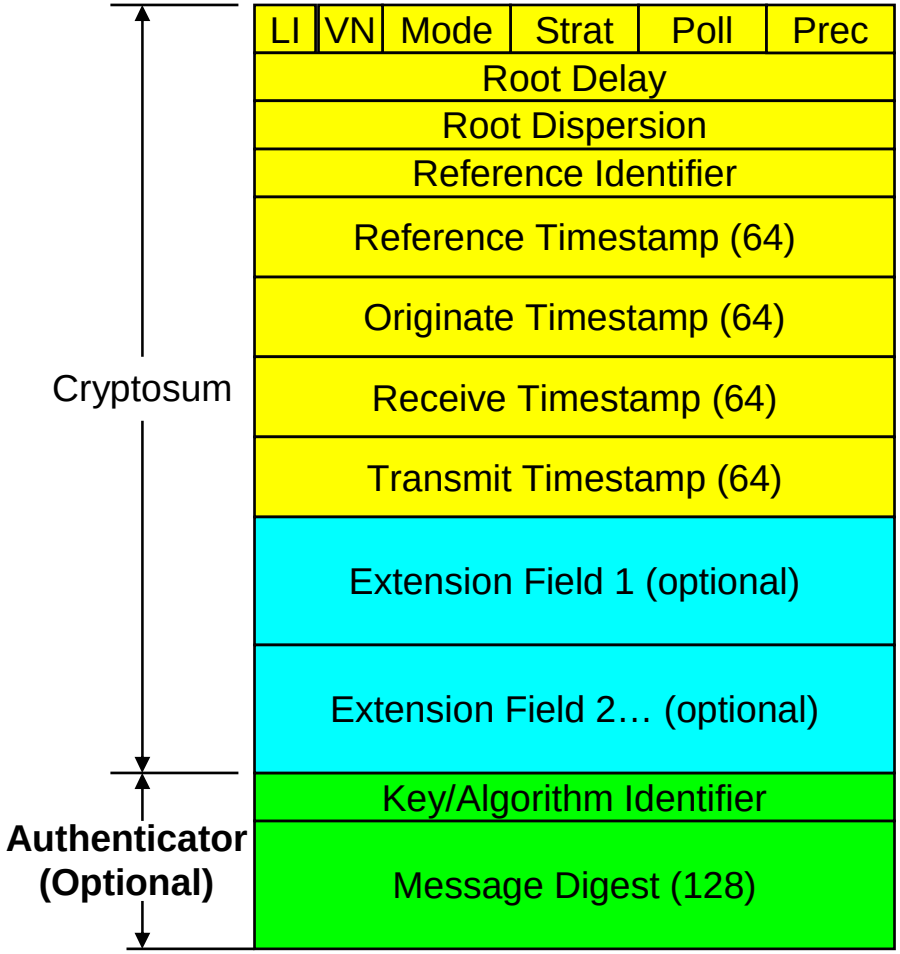
- Goals
  - Provide the best accuracy under prevailing network and server conditions.
  - Resist many and varied kinds of failures, including two-face, fail-stop, malicious attacks and implementation bugs.
  - Maximize utilization of Internet diversity and redundancy.
  - Automatically organize subnet topology for best accuracy and reliability.
  - Self contained cryptographic authentication based on both symmetric key and public key infrastructures and independent of external services.
- Non-goals
  - Local time – this is provided by the operating system.
  - Access control - this is provided by firewalls and address filtering.
  - Privacy - all protocol values, including time values, are public.
  - Non-repudiation - this can be provided by a layered protocol if necessary.
  - Conversion of NTP timestamps to and from other time representations and formats.

Evolution to NTP Version 4

- Current Network Time Protocol Version 3 has been in use since 1992, with nominal accuracy in the low milliseconds.
- Modern workstations and networks are much faster today, with attainable accuracy in the low microseconds.
- NTP Version 4 architecture, protocol and algorithms have been evolved to achieve this degree of accuracy.
  - Improved clock models which accurately predict the time and frequency adjustment for each synchronization source and network path.
  - Engineered algorithms reduce the impact of network jitter and oscillator wander while speeding up initial convergence.
  - Redesigned clock discipline algorithm operates in frequency-lock, phase-lock and hybrid modes.
- The improvements, confirmed by simulation, improve accuracy by about a factor of ten, while allowing operation at much longer poll intervals without significant reduction in accuracy.

NTP protocol header and timestamp formats

NTP Protocol Header Format (32 bits)



LI	leap warning indicator
VN	version number (4)
Strat	stratum (0-15)
Poll	poll interval (log2)
Prec	precision (log2)

NTP Timestamp Format (64 bits)

Seconds (32)	Fraction (32)

Value is in seconds and fraction since 0^h 1 January 1900

NTP v4 Extension Field

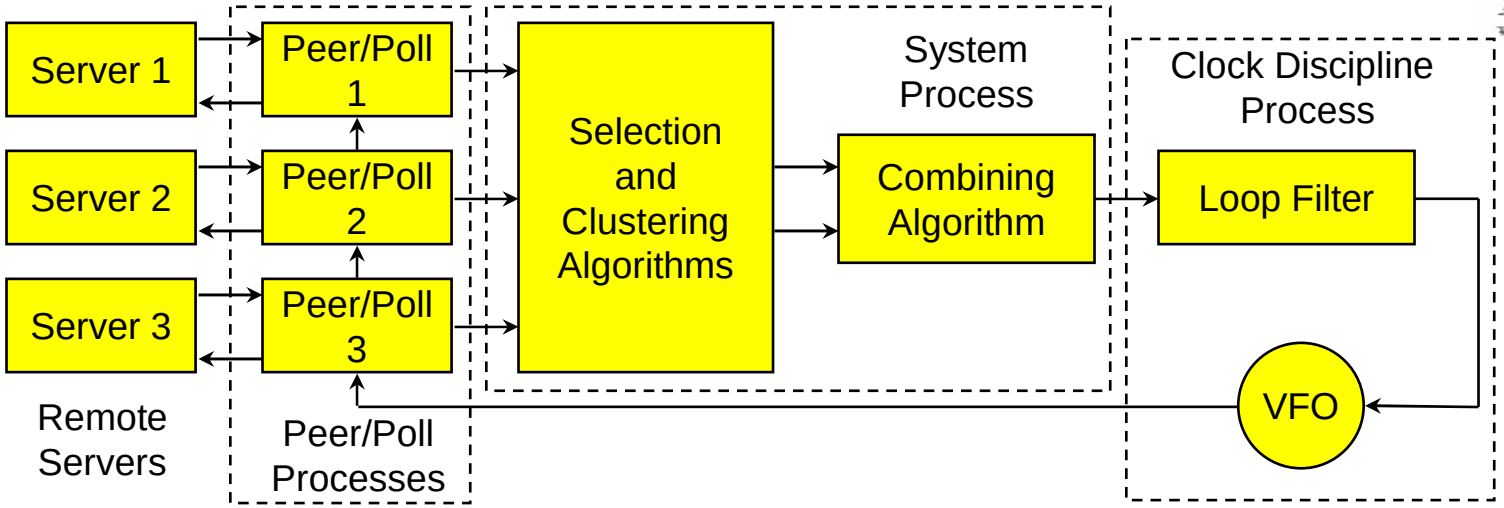
Field Type	Length
Extension Field (padded to 32-bit boundary)	

Last field padded to 64-bit boundary

NTP v3 and v4
NTP v4 only
authentication only

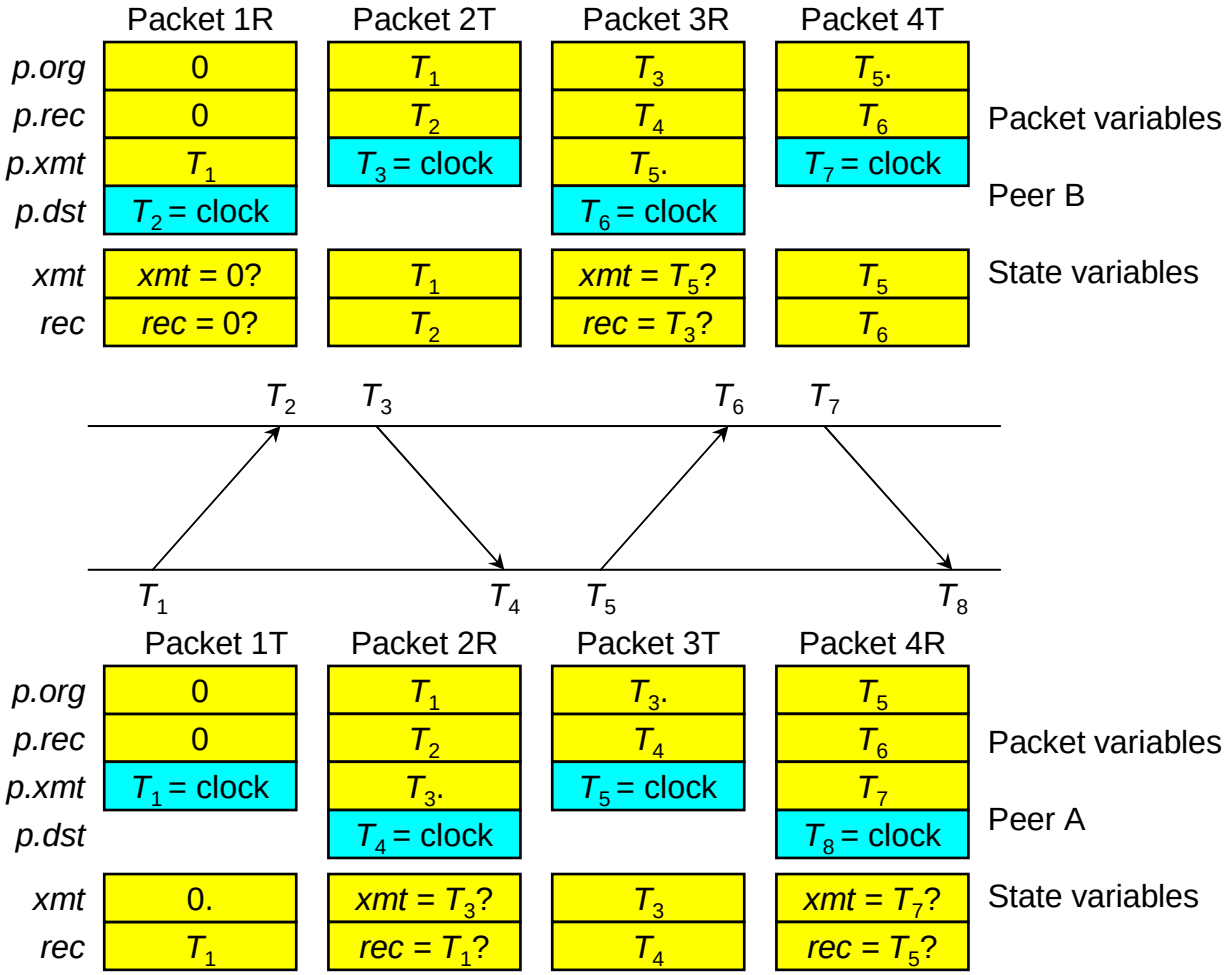
Authenticator uses MD5 cryptosum of NTP header plus extension fields (NTPv4)

NTP process decomposition

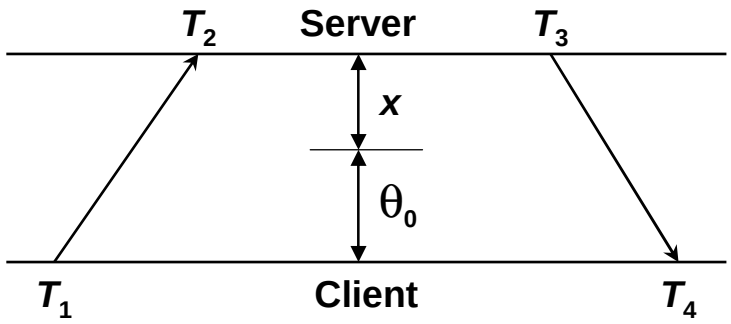


- Peer process runs when a packet is received.
- Poll process sends packets at intervals determined by the clock discipline process and remote server.
- System process runs when a new peer process update is received.
- Clock discipline process runs at intervals determined by the measured network phase jitter and clock oscillator (VFO) frequency wander.
- Clock adjust process (VFO) runs at intervals of one second.

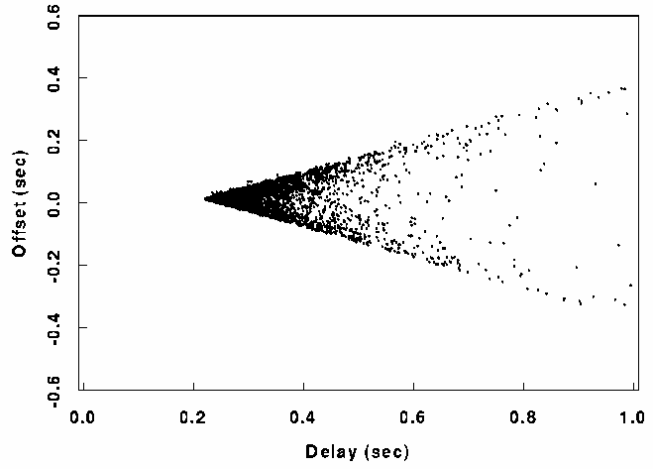
NTP peer protocol



Clock filter algorithm

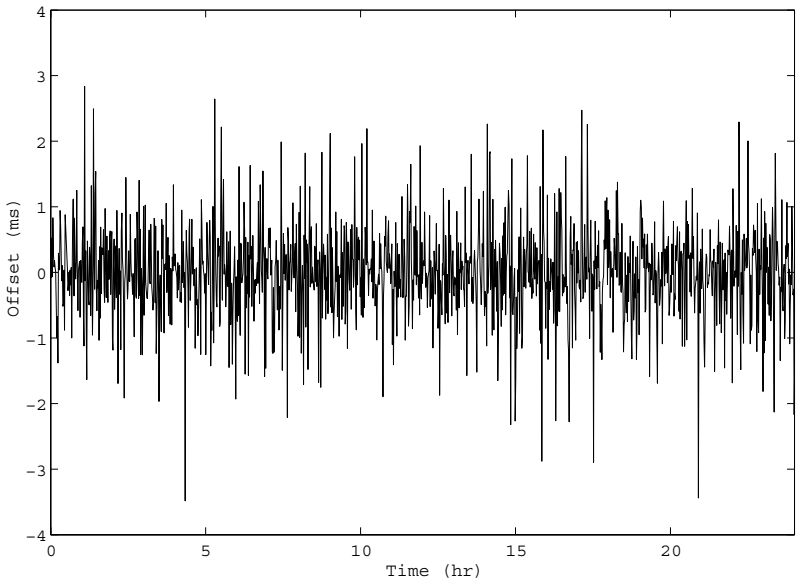


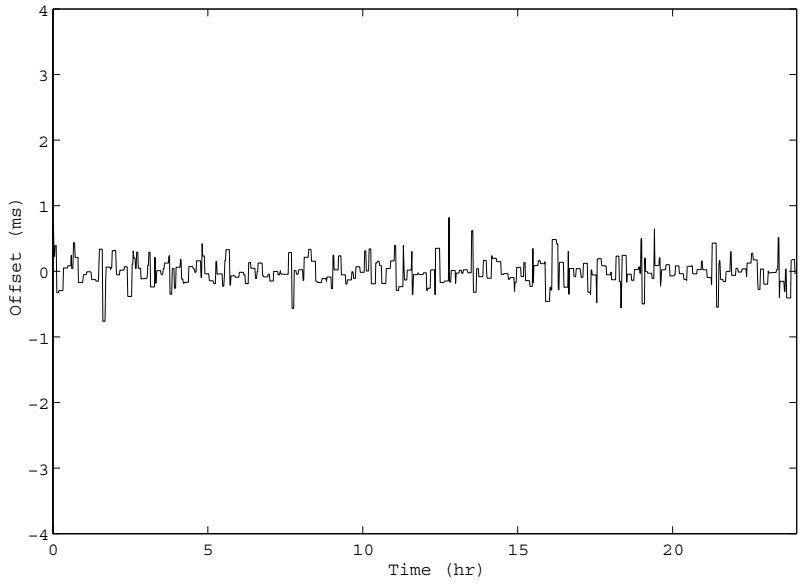
$$\theta = \tfrac{1}{2}[(T_2 - T_1) + (T_3 - T_4)]$$
$$\delta = (T_4 - T_1) - (T_3 - T_2)$$



- The most accurate offset θ_0 is measured at the lowest delay δ_0 (apex of the wedge scattergram).
- The correct time θ must lie within the wedge $\theta_0 \pm (\delta - \delta_0)/2$.
- The δ_0 is estimated as the minimum of the last eight delay measurements and (θ_0, δ_0) becomes the peer update.
- Each peer update can be used only once and must be more recent than the previous update.

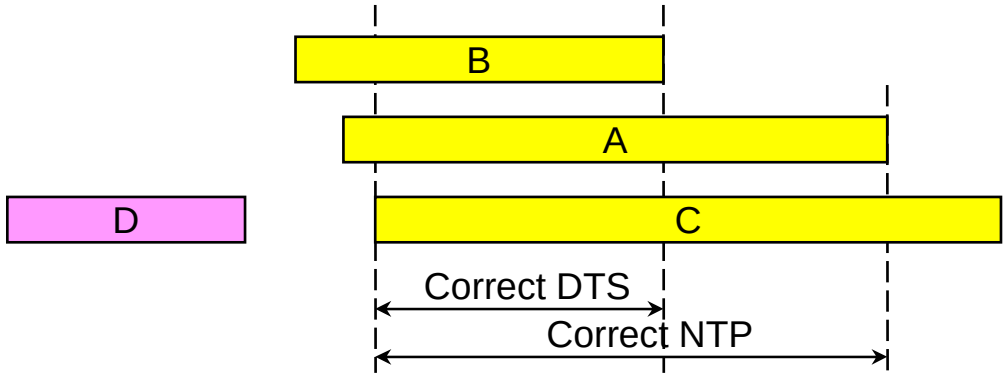
Clock filter performance





- Left figure shows raw time offsets measured for a typical path over a 24-hour period (mean error 724 μs, median error 192 μs)
- Right graph shows filtered time offsets over the same period (mean error 192 μs, median error 112 μs).
- The mean error has been reduced by 11.5 dB; the median error by 18.3 dB. This is impressive performance.

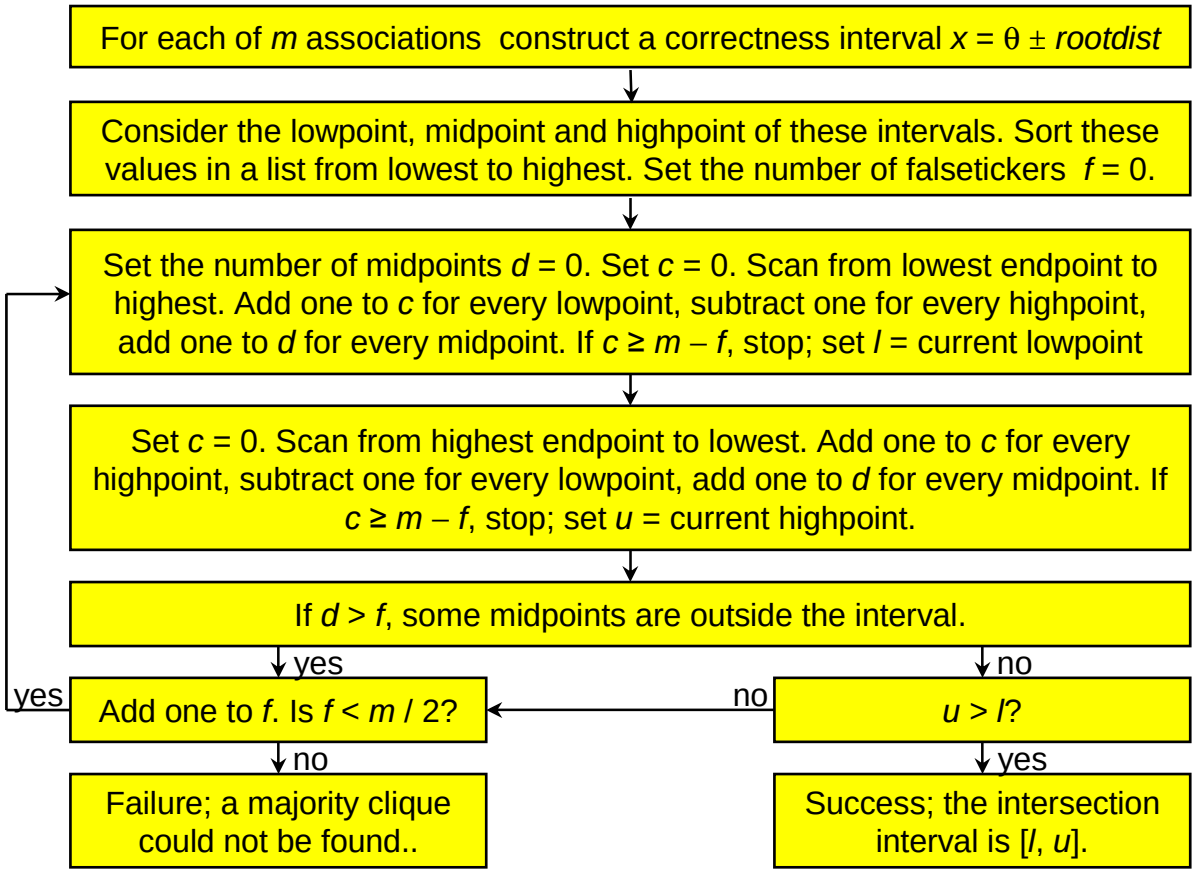
Clock select principles



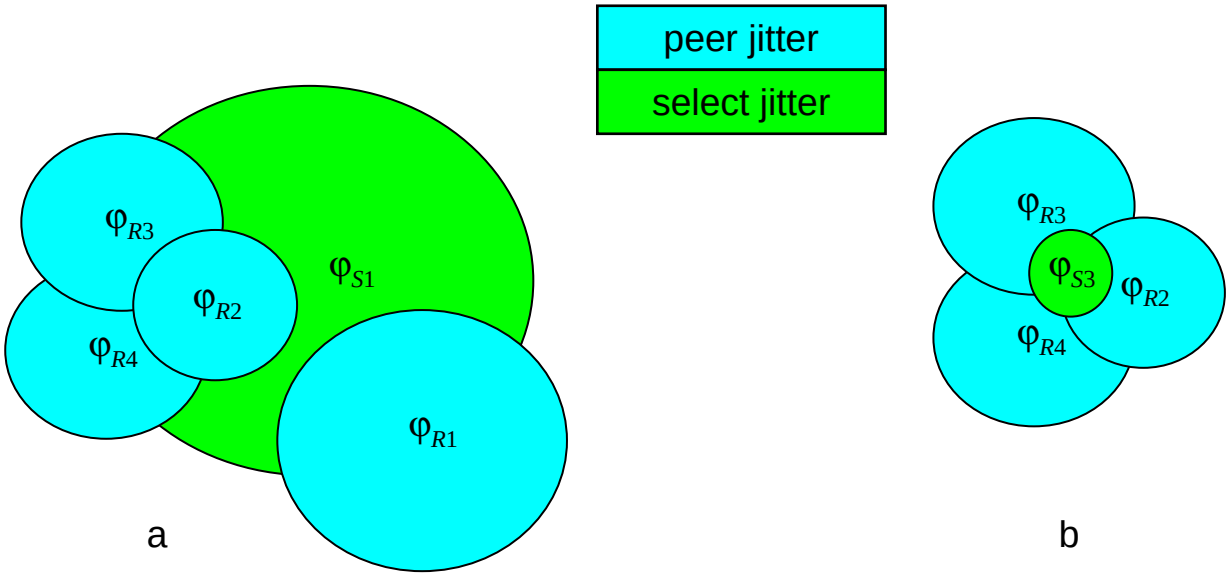
correctness interval = q - l £ q_0 £ q + l
m = number of clocks
f = number of presumed falsetickers
A, B, C are truechimers
D is falseticker

- The correctness interval for any candidate is the set of points in the interval of length twice the synchronization distance centered at the computed offset.
- The DTS interval contains points from the largest number of correctness intervals, i.e., the intersection of correctness intervals.
- The NTP interval includes the DTS interval, but requires that the computed offset for each candidate is contained in the interval.
- Formal correctness assertions require at least half the candidates be in the NTP interval. If not, no candidate can be considered a truechimer.

Clock select algorithm

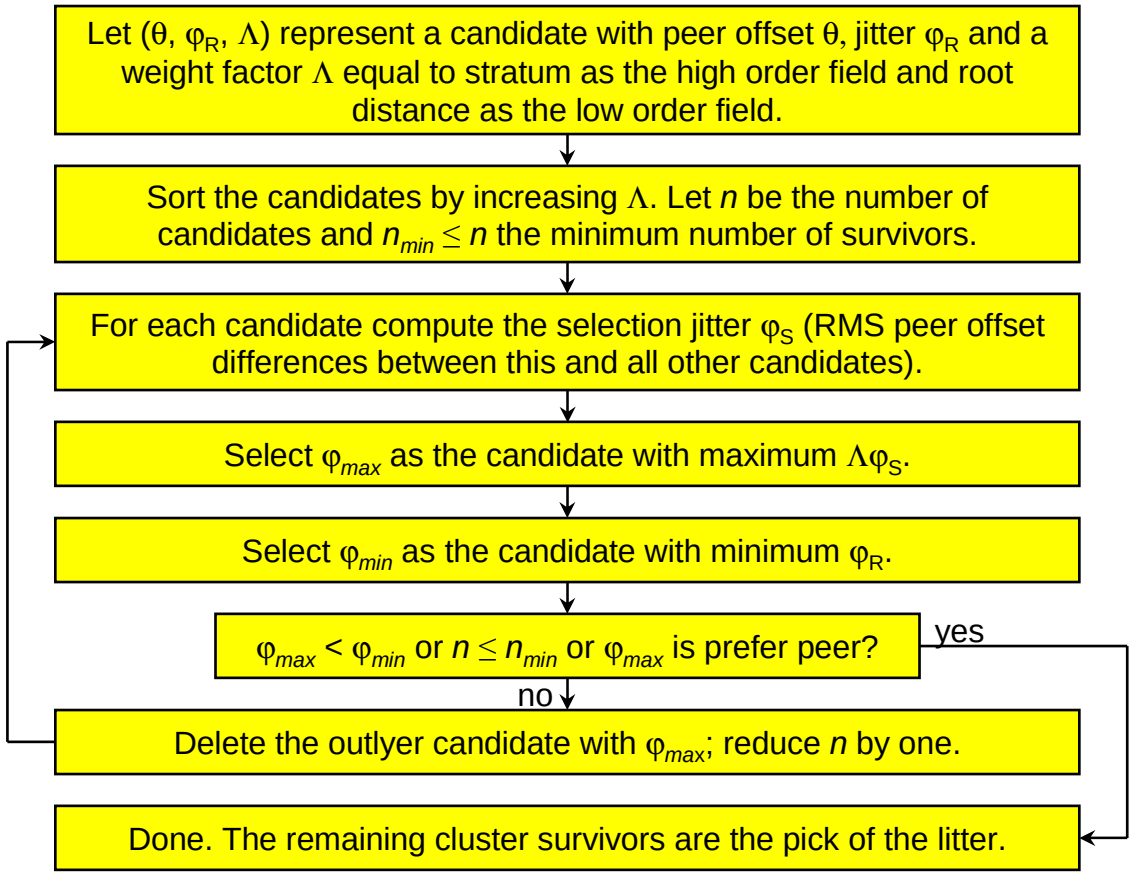


Cluster principles

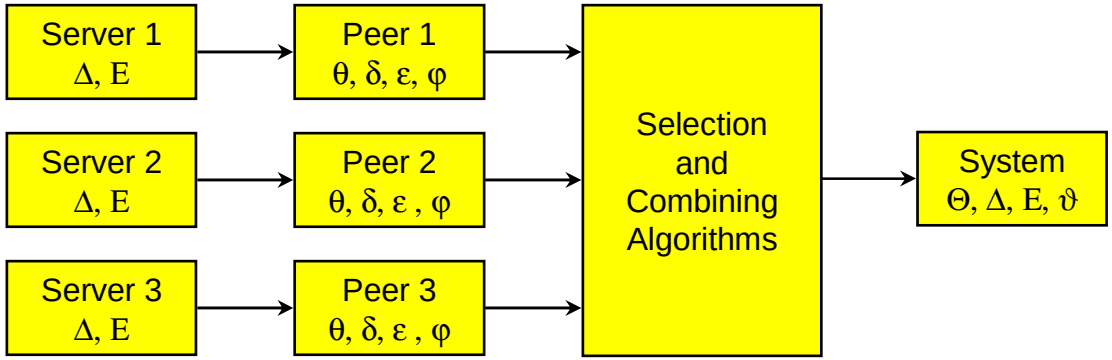


- Candidate 1 is further from the others, so its select jitter φ_{S1} is highest.
- (a) $\varphi_{max} = \varphi_{S1}$ and $\varphi_{min} = \varphi_{R2}$. Since $\varphi_{max} > \varphi_{min}$, the algorithm prunes candidate 1 to reduce select jitter and continues.
- (b) $\varphi_{max} = \varphi_{S3}$ and $\varphi_{min} = \varphi_{R2}$. Since $\varphi_{max} < \varphi_{min}$, pruning additional candidates will not reduce select jitter. So, the algorithm ends with φ_{R2}, φ_{R3} and φ_{R4} as survivors.

Cluster algorithm

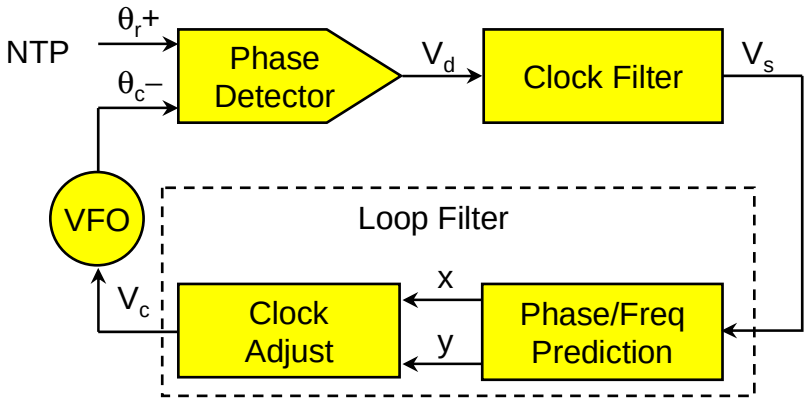


NTP dataflow analysis



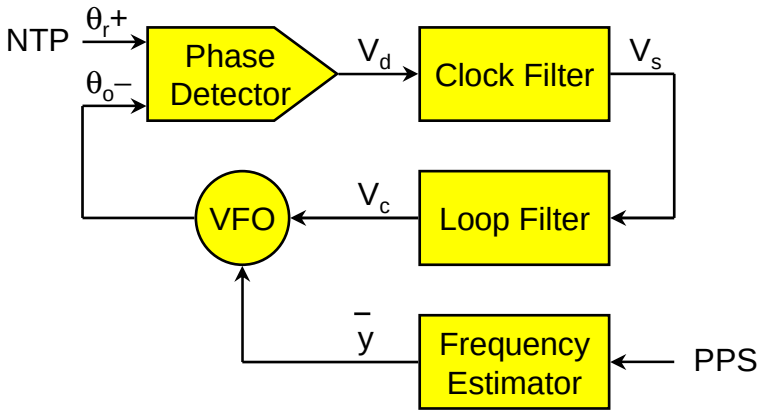
- Each server provides delay Δ and dispersion E relative to the root of the synchronization subtree.
- As each NTP message arrives, the peer process updates peer offset θ, delay δ, dispersion ε and jitter φ.
- At system poll intervals, the clock selection and combining algorithms updates system offset Θ, delay Δ, dispersion E and jitter ϑ.
- Dispersions ε and E increase with time at a rate depending on specified frequency tolerance ϕ.

Clock discipline algorithm



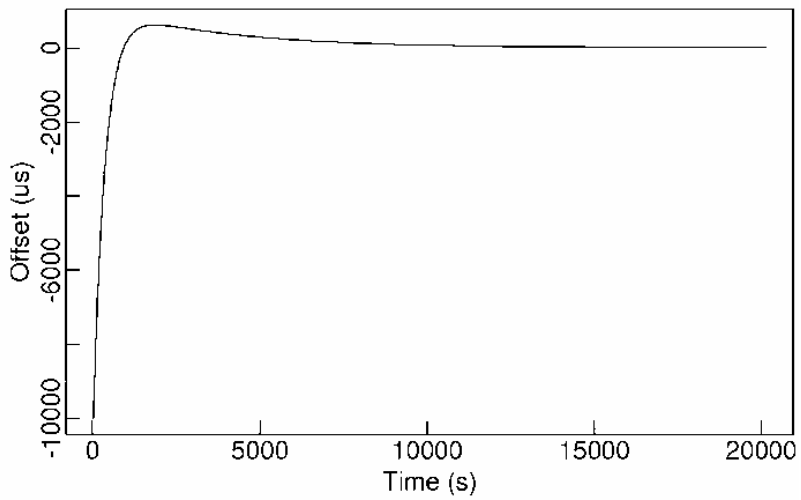
- V_d is a function of the phase difference between NTP and the VFO.
- V_s depends on the stage chosen on the clock filter shift register.
- *x* and *y* are the phase update and frequency update, respectively, computed by the prediction functions.
- Clock adjust process runs once per second to compute V_c, which controls the frequency of the local clock oscillator.
- VFO phase is compared to NTP phase to close the feedback loop.

NTP clock discipline with PPS steering

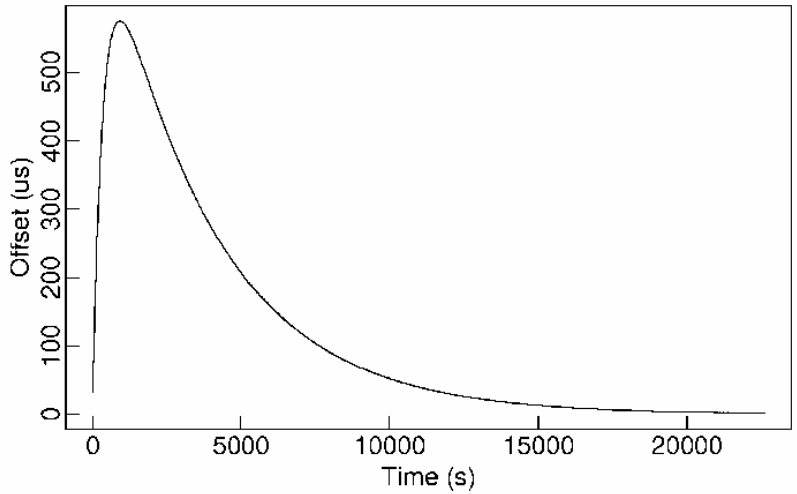


- NTP daemon disciplines variable frequency oscillator (VFO) phase V_c relative to accurate and reliable network sources.
- Kernel disciplines VFO frequency *y* to pulse-per-second (PPS) signal.
- Clock accuracy continues to be disciplined even if NTP daemon or sources fail.
- In general, the accuracy is only slightly degraded relative to a local reference source.

Traditional approach using phase-lock loop (PLL)



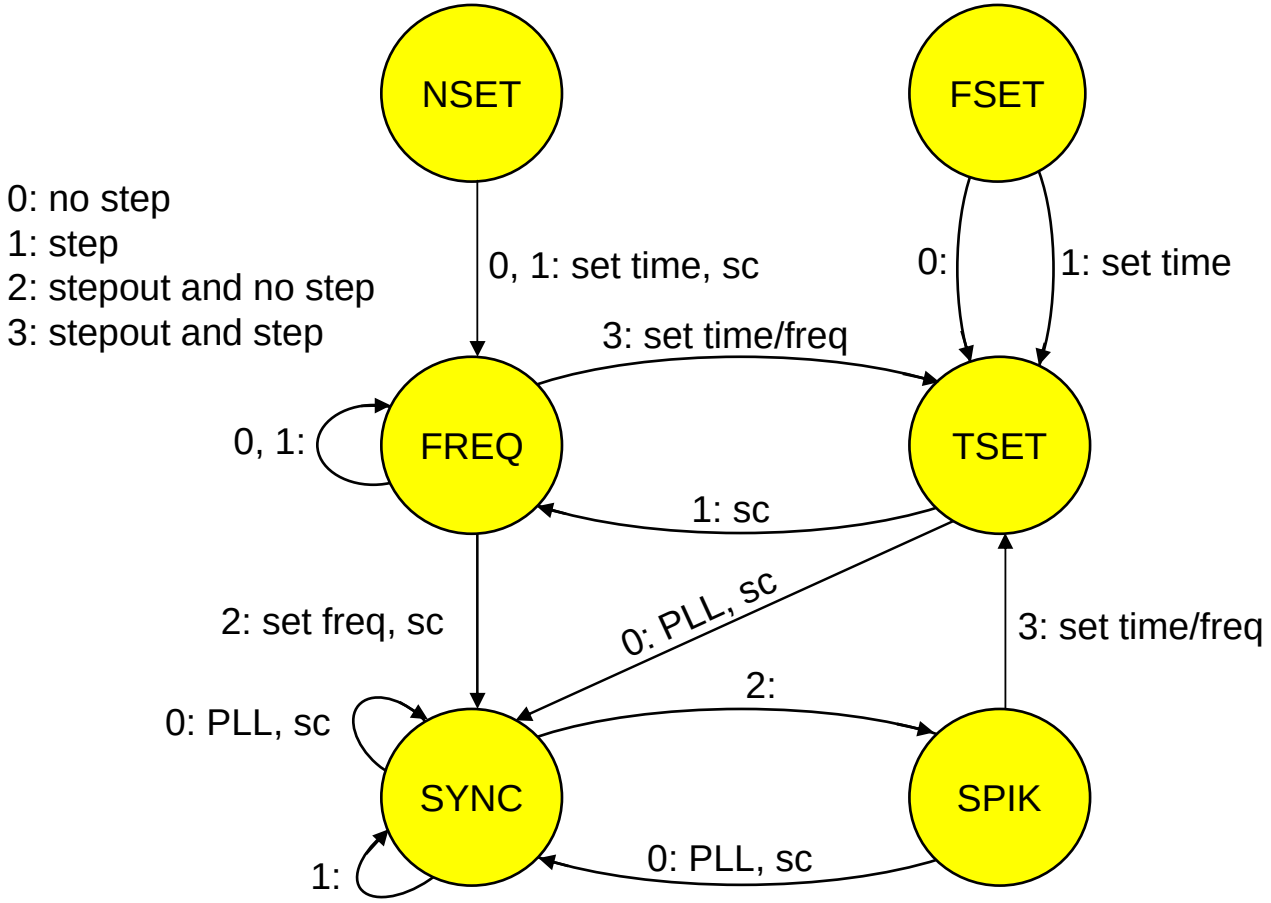
Response to 10-ms Phase Step



Response to 2-PPM Frequency Step

- Left graph shows the impulse response for a 10-ms time step and 64-s poll interval using a traditional linear PLL.
- Right graph shows the impulse response for a 5-PPM frequency step and 64-s poll interval.
- It takes too long to converge the loop using linear systems.
- A hybrid linear/nonlinear approach may do much better.

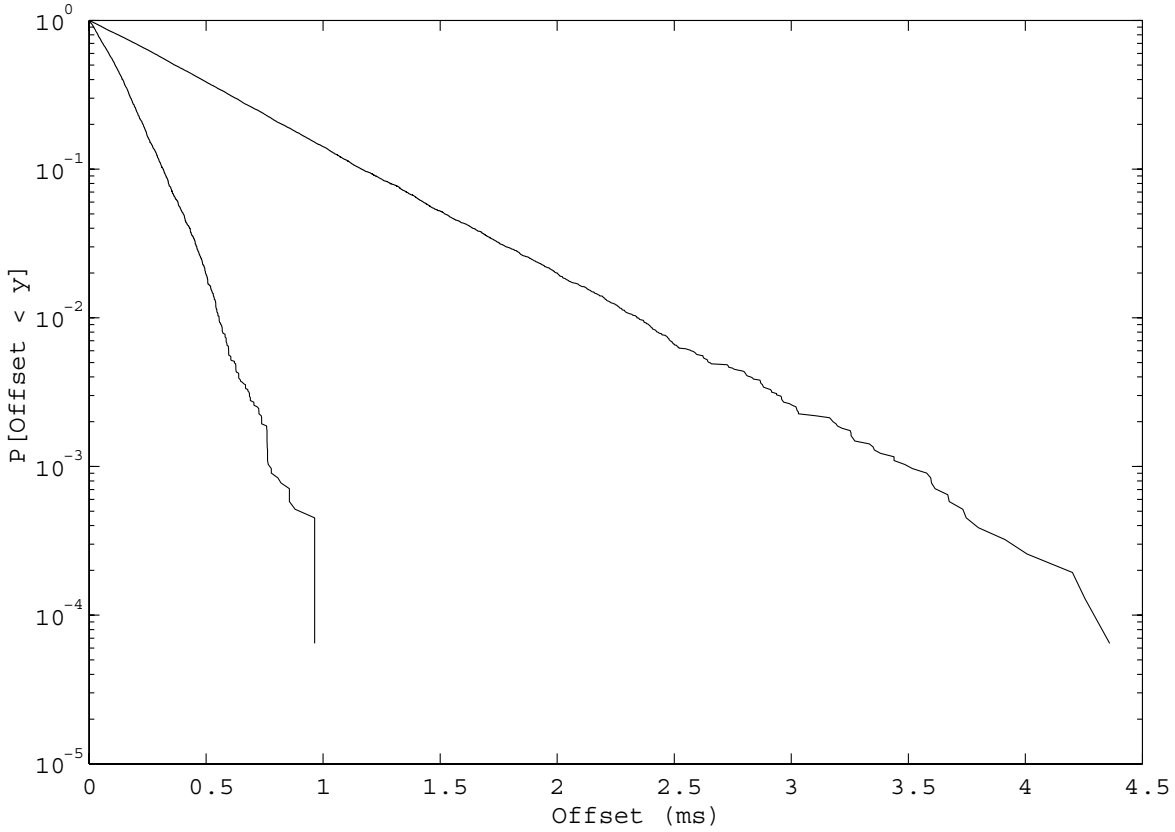
Clock state machine transition function



NTP enhancements for precision time

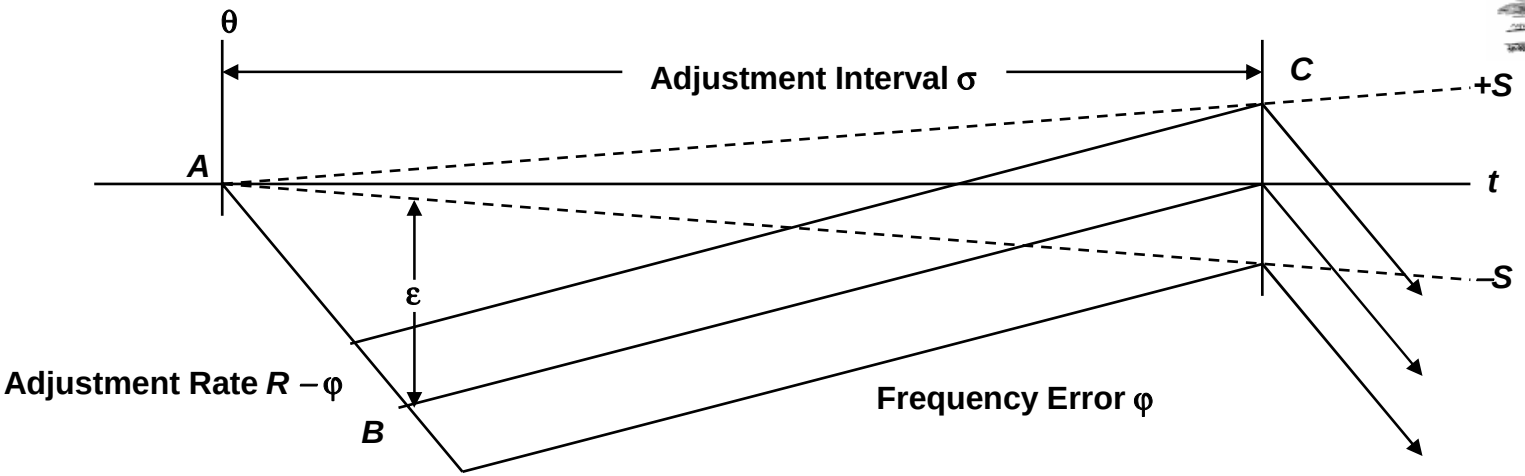
- Precision time kernel modifications
  - Time and frequency discipline from NTP or other source
  - Pulse-per-second (PPS) signal interface via modem control lead
- Improved computer clock algorithms
  - Hybrid phase/frequency clock discipline algorithm
  - Message intervals extended to 36 hours for toll telephone services
  - Improved glitch detection and supression
- Precision time and frequency sources
  - PPS signal grooming with median filter and dynamic adaptive time constant
  - Additional drivers for new GPS receivers and PPS discipline
- Reduced hardware and software latencies
  - Serial driver modifications to remove character batching
  - Early timestamp/PPS capture using line disciplines
  - Protocol modifications for multiple primary source mitigation

Minimize effects of network jitter



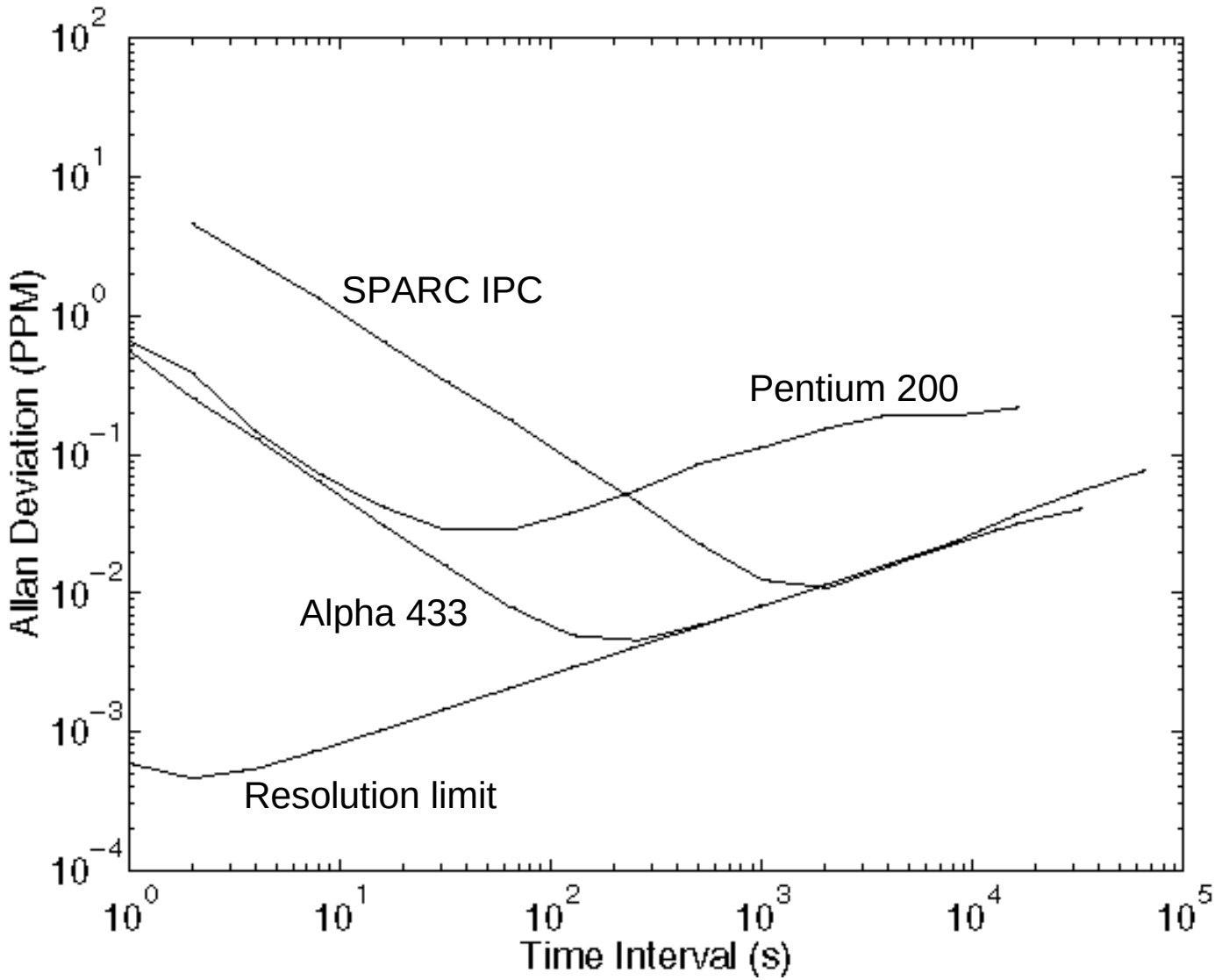
- The traces show the cumulative probability distributions for
  - Upper trace: raw time offsets measured over a 12-day period
  - Lower trace: filtered time offsets after the clock filter

Unix time adjustment primitive

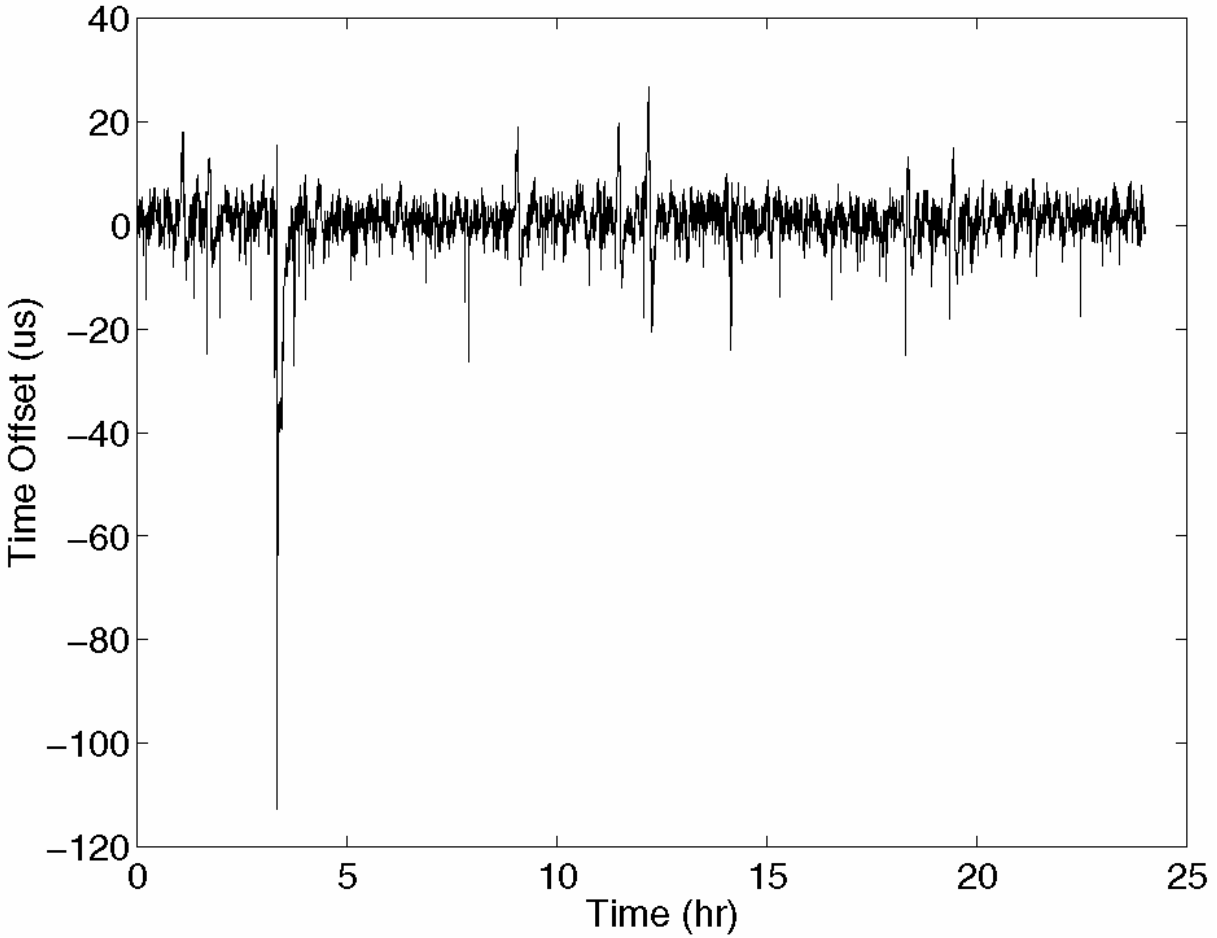


- The discipline needs to steer the frequency over the range ±*S*, but the intrinsic clock frequency error is φ
- Unix `adjtime()` slews frequency at rate *R* – φ PPM beginning at *A*
- Slew continues to *B*, depending on the programmed frequency steer
- Offset continues to *C* with frequency offset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

Computer clock modelling

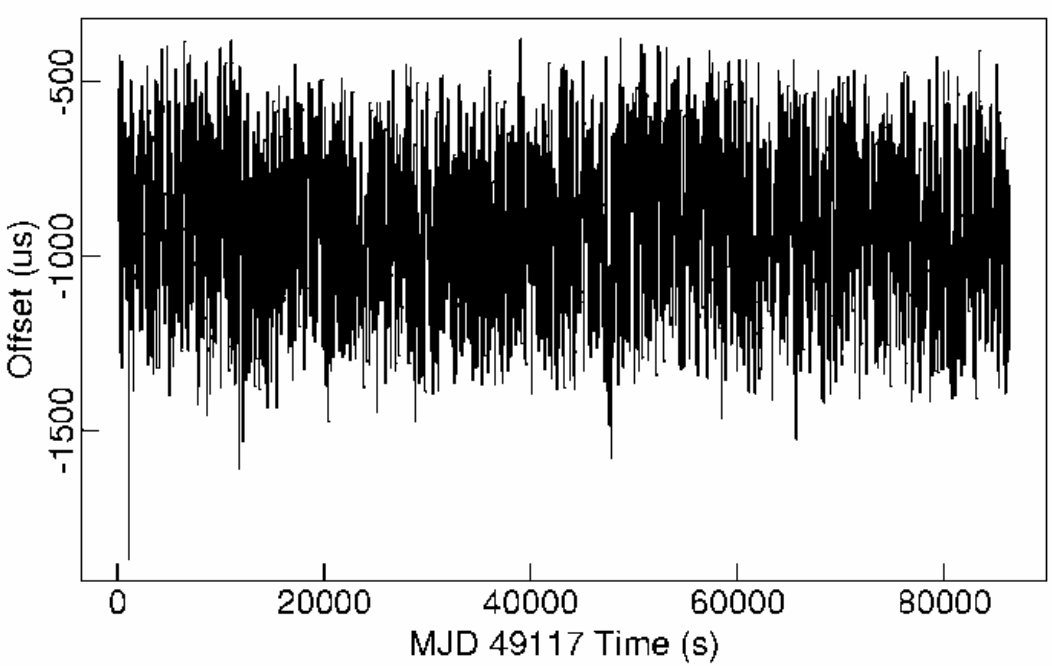


PPS time offset characteristic for Rackety



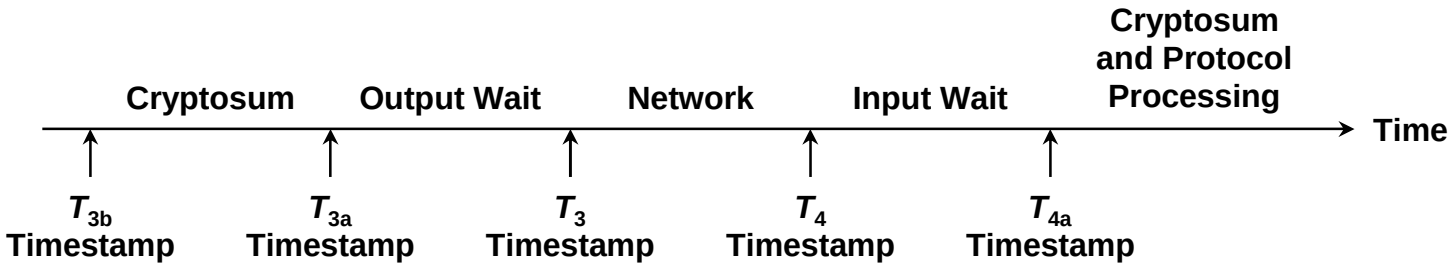
- Jitter is presumed caused by interrupt latencies on the Sbus
- Large negative spikes reflect contention by the radios and network

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/trimmed-mean filter of 60 samples
  - Using on-second format and filter, residual jitter is less than 50 µs

Minimize latencies in the operating system

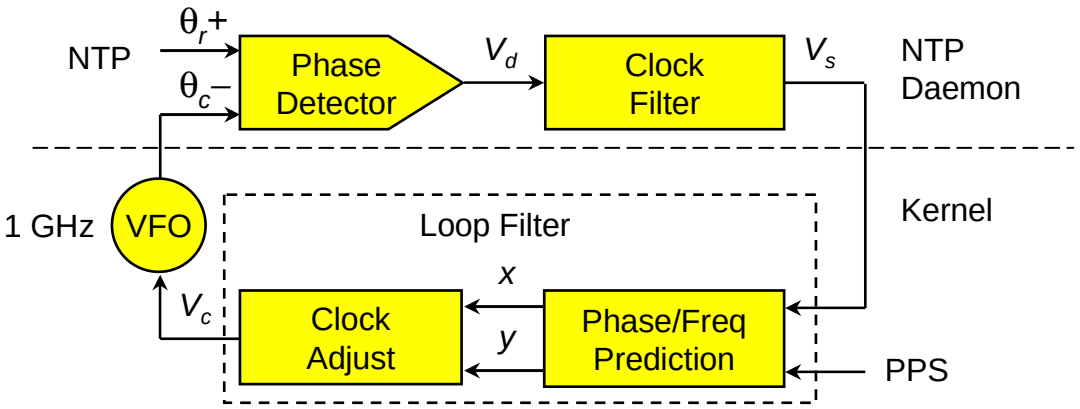


- We want T_3 and T_4 timestamps for accurate network calibration
  - If output wait is small, T_{3a} is good approximation to T_3
  - T_{3a} can't be included in message after cryptosum is calculated, but can be sent in next message; if not, use T_{3b} as best approximation to T_3
  - T_4 captured by 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

Kernel modifications for nanosecond resolution

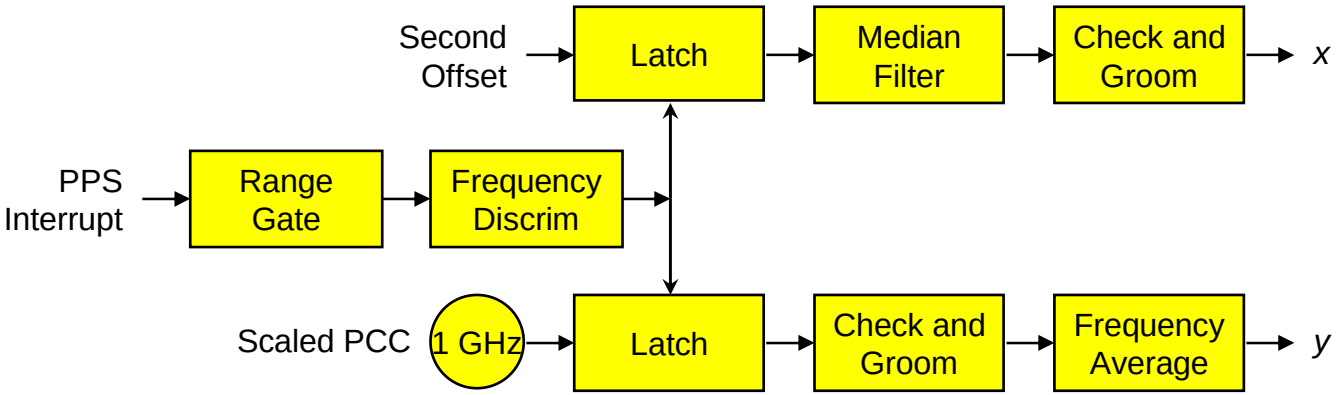
- *Nanokernel* package of routines compiled with the operating system kernel
- Represents time in nanoseconds and fraction, frequency in nanoseconds per second and fraction
- Implements nanosecond system clock variable with either microsecond or nanosecond kernel native time variables
- Uses native 64-bit 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



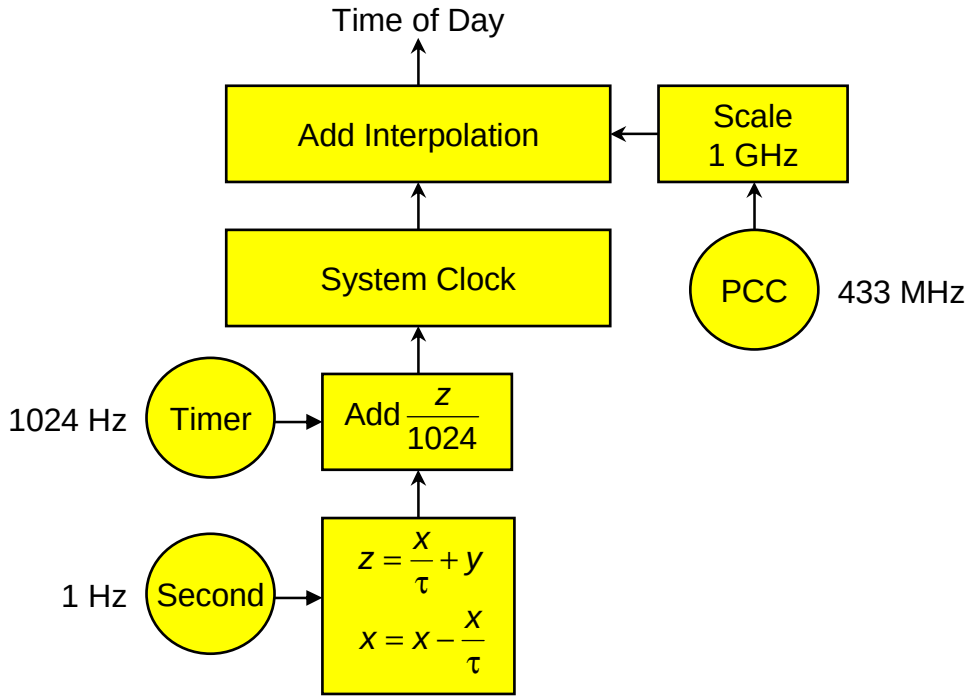
- Type II, adaptive-parameter, hybrid phase/frequency-lock loop disciplines variable frequency oscillator (VFO) phase and frequency
- NTP daemon computes phase error $V_d = \theta_r - \theta_o$ between source and VFO, then grooms samples to produce time update V_s
- Loop filter computes phase *x* and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

PPS phase and frequency discipline



- Phase and frequency disciplined separately - phase from system clock second offset, frequency from processor cycle counter (PCC)
- Frequency discriminator rejects noise and invalid signals
- Median filter rejects sample outlyers and provides error statistic
- Check and groom rejects popcorn spikes and clamps outlyers
- Phase offsets exponentially averaged with variable time constant
- Frequency offsets averaged over variable interval

Nanosecond clock

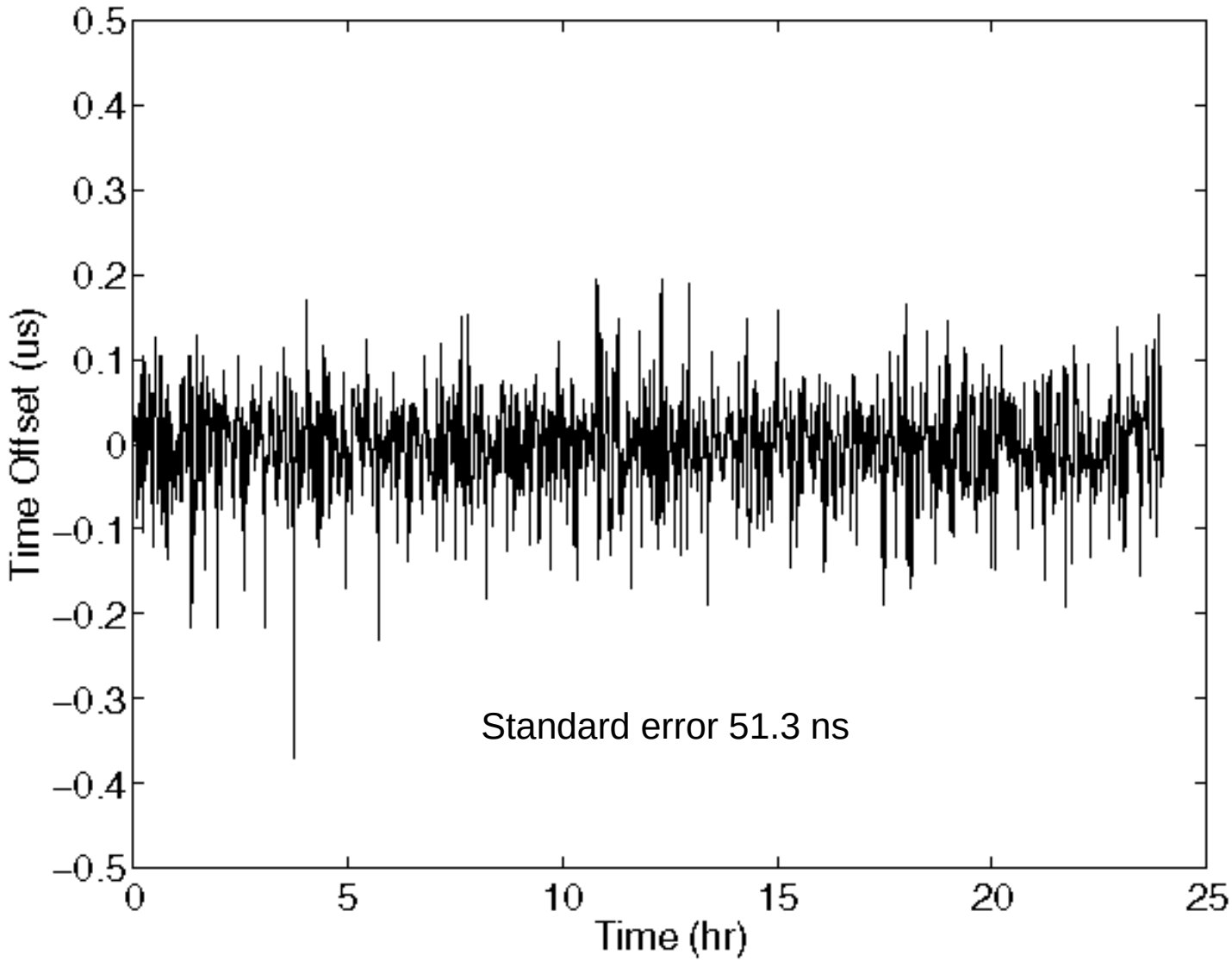


- Phase *x* and frequency *y* are updated by the PLL/FLL or PPS loop.
- At the second overflow increment *z* is calculated and *x* reduced by the time constant.
- The increment is amortized over the second at each tick interrupt.
- Time between ticks is interpolated from the PCC scaled to 1 GHz.

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

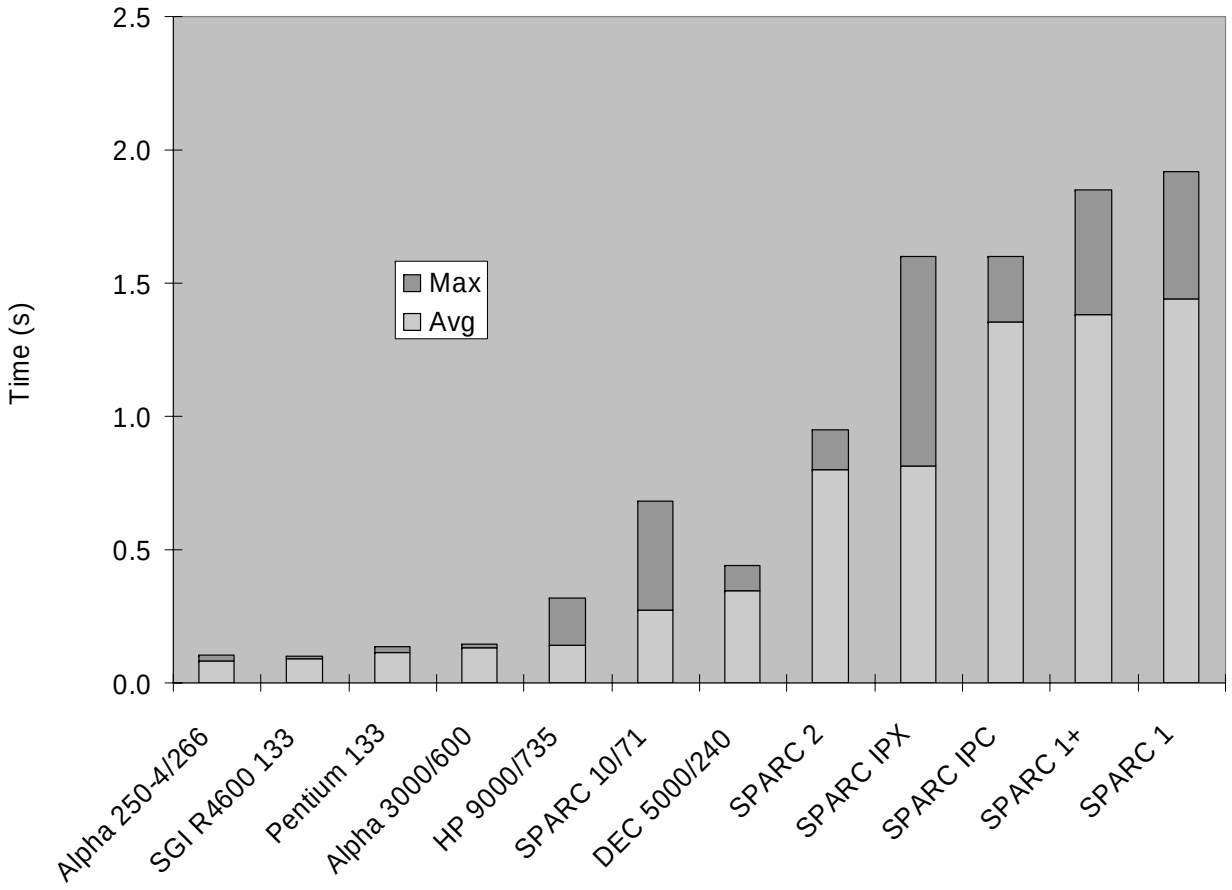
Measured PPS time error for Alpha 433



Symmetric key and public key cryptography

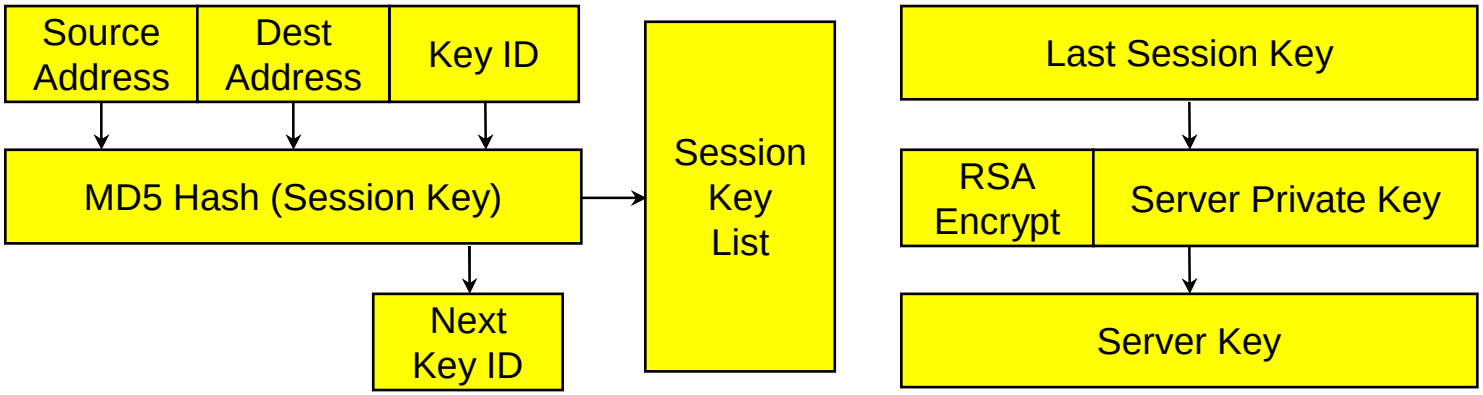
- Public key cryptography
  - Encryption/decryption algorithms are relatively slow with highly variable running times depending on key and data
  - All keys are random; private keys are never divulged
  - Certificates reliably bind server identification and public key
  - Server identification established by challenge/response protocol
  - Well suited to multicast paradigm
- Symmetric key cryptography
  - Encryption/decryption algorithms are relatively fast with constant running times independent of key and data
  - Fixed private keys must be distributed in advance
  - Key agreement (Diffie-Hellman) is required for private random keys
  - Per-association state must be maintained for all clients
  - Not well suited to multicast paradigm

MD5/RSA digital signature computations



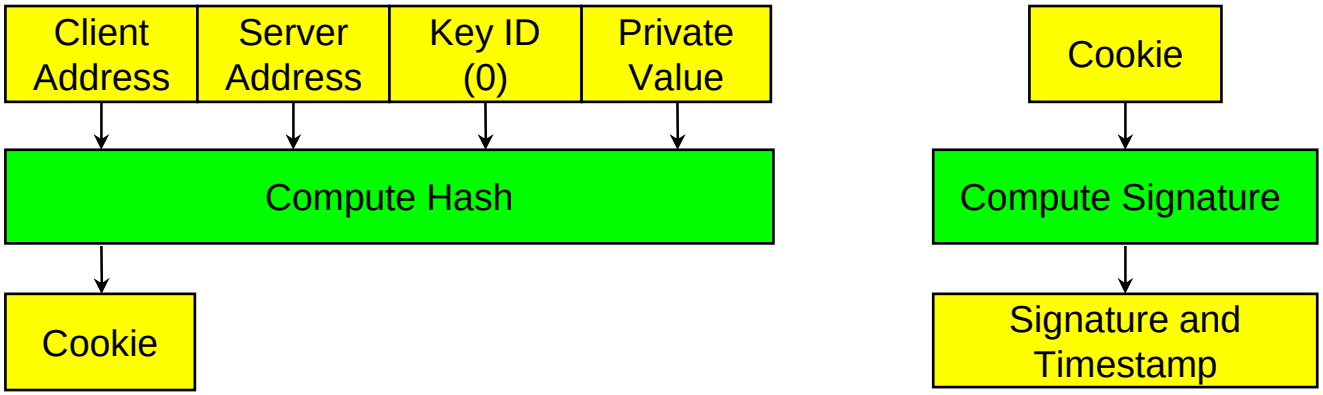
- Measured times (s) to construct digital signature using RSAREF
- Message authentication code constructed from 48-octet NTP header hashed with MD5, then encrypted with RSA 512-bit private key

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



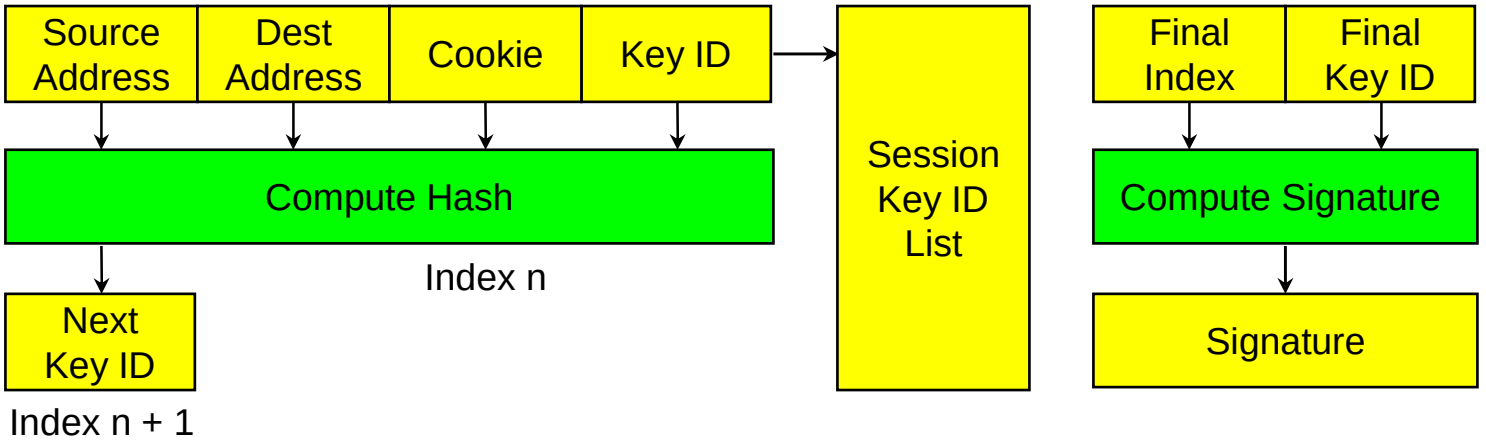
- Server rolls a random 32-bit seed as the initial key ID
- Server generates a session key list using repeated MD5 hashes
- Server encrypts the last key using RSA and its private key to produce the initial server key and provides it and its public key to all clients
- Server uses the session key list 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

Computing the cookie



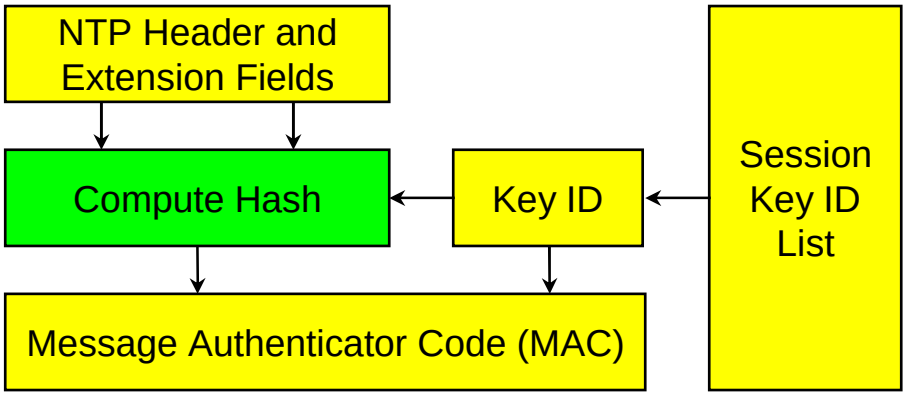
- The server generates a cookie unique to the client and server addresses and its own private value. It returns the cookie, signature and timestamp to the client in an extension field.
- The cookie is transmitted from server to client encrypted by the client public key.
- The server uses the cookie to validate requests and construct replies.
- The client uses the cookie to validate the reply and checks that the request key ID matches the reply key ID.

Generating the session key list



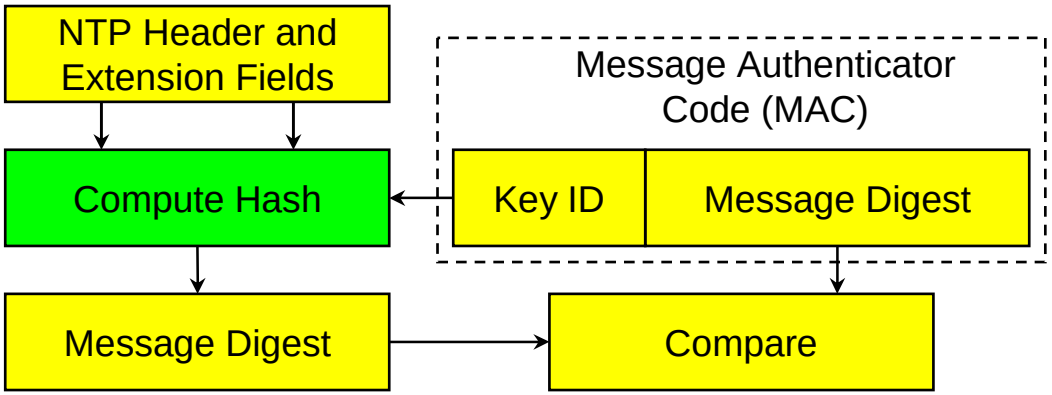
- The server rolls a random 32-bit seed as the initial key ID and selects the cookie. Messages with a zero cookie contain only public values.
- The initial session key is constructed using the given addresses, cookie and initial key ID. The session key value is stored in the key cache.
- The next session key is constructed using the first four octets of the session key value as the new key ID. The server continues to generate the full list.
- The final index number and last key ID are provided in an extension field with signature and timestamp.

Sending messages



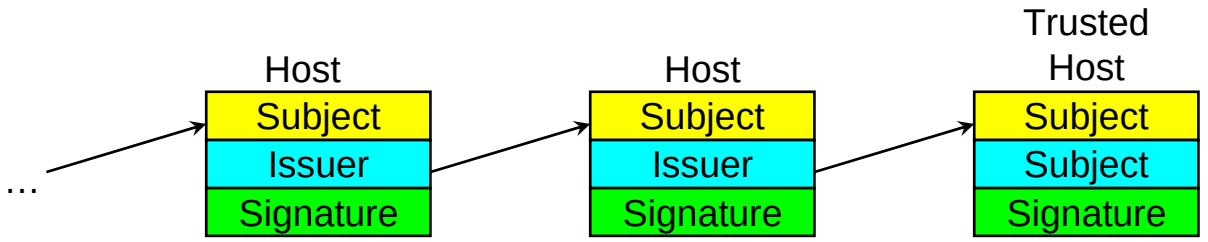
- The message authenticator code (MAC) consists of the MD5 message digest of the NTP header and extension fields using the session key ID and value stored in the key cache.
- The server uses the session key ID list in reverse order and discards each key value after use.
- An extension field containing the last index number and key ID is included in the first packet transmitted (last on the list).
- This extension field can be provided upon request at any time.
- When all entries in the key list are used, a new one is generated.

Receiving messages



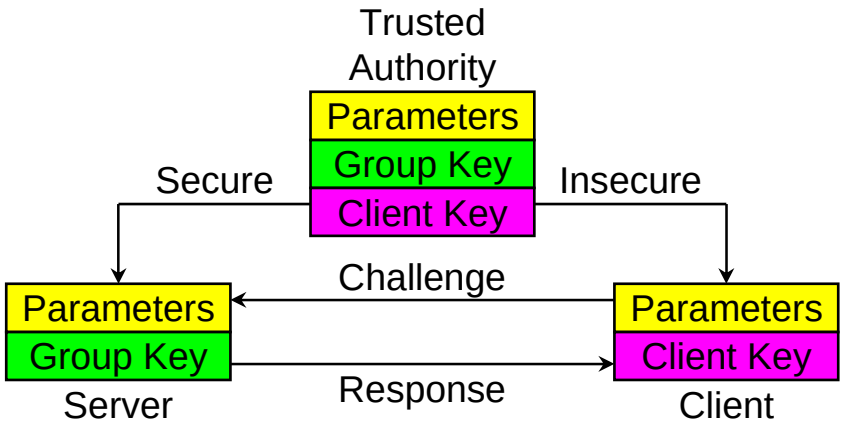
- The intent is not to hide the message contents, just verify where it came from and that it has not been modified in transit.
- The MAC message digest is compared with the computed digest of the NTP header and extension fields using the session key ID in the MAC and the key value computed from the addresses, key ID and cookie.
- If the cookie is zero, the message contains public values. Anybody can validate the message or make a valid message containing any values.
- If the cookie has been determined by secret means, nobody except the parties to the secret can validate a message or make a valid message.

Trusted certificate (TC) identity scheme



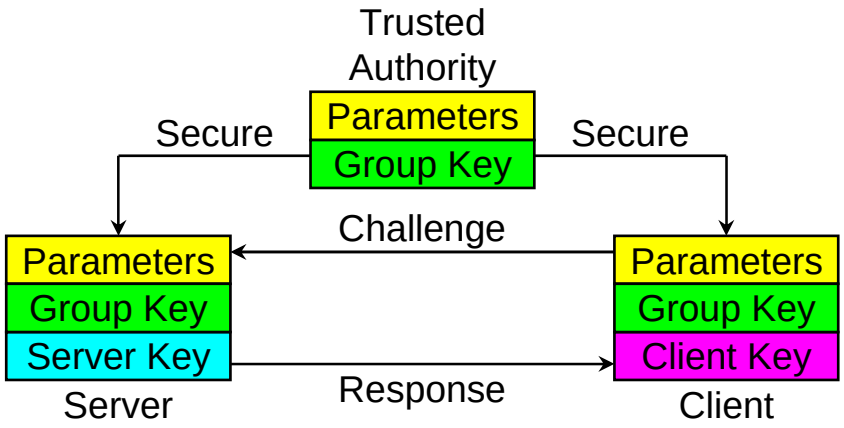
- Each certificate is signed by the issuer, which is one step closer on the trail to the trusted host.
- The trusted host certificate is self-signed and self-validated.
- This scheme is vulnerable to a middleman masquerade, unless an identity scheme is used.
- The identity scheme, if used, has the same name as the trusted host subject name.

Schnorr (IFF) identity scheme



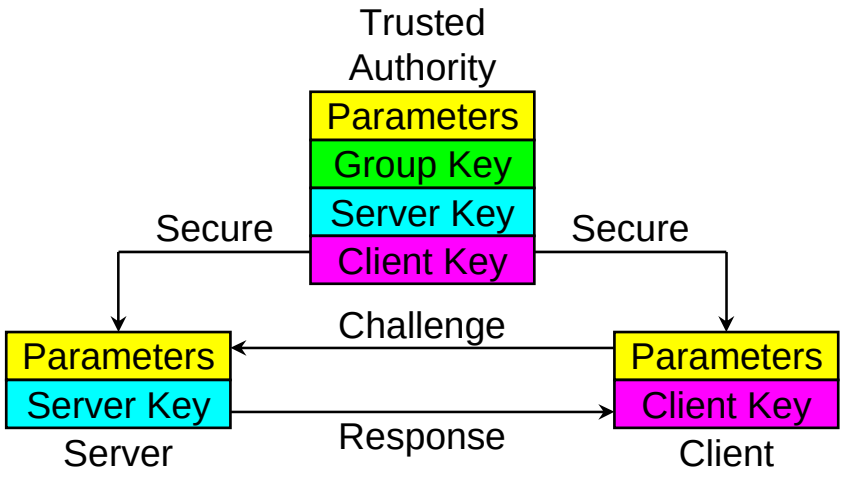
- TA generates the IFF parameters and keys and transmits them by secure means to all servers and clients.
- Only the server needs the group key; the client key derived from it is public.
- IFF identity exchange is used to verify group membership.

Guillou-Quisquater (GQ) scheme



- TA generates the GQ parameters and keys and transmits them by secure means to servers and clients.
- Server generates a GQ private/public key pair and certificate with the public key in an extension field.
- Client uses the public key in the certificate as the client key.
- GQ identity exchange is used to verify group membership.

Mu-Varadharajan (MV) scheme



- TA generates MV parameters, group key, server key and client keys.
- TA transmits private encryption and public decryption keys to all servers using secure means.
- TA transmits individual private decryption keys to each client using secure means.
- TA can activate/deactivate individual client keys.
- The MV identity exchange is used to verify group membership.

Further information

- NTP home page 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 home page 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
- Udel FTP server: ftp://ftp.udel.edu/pub/ntp
  - Current NTP Version software, documentation and support
  - Collaboration resources and junkbox
- Related projects http://www.eecis.udel.edu/~mills/status.htm
  - Current research project descriptions and briefings