How NTP Works

Daniel Franke
NTPsec security officer

May 1, 2016

What is time? — Philosophical background

Wikipedia being unhelpful: “time is what a clock reads”. How then to abide the notion of an inaccurate clock?

Per the Duhem-Quine thesis: time can only be defined holistically, by the role it plays in a theory of physics.

Time in physics

From Wikipedia, the free encyclopedia

Time in physics is defined by its measurement: time is what a clock reads.[1] In classical, non-relativistic physics it is a scalar quantity and, like length, mass, and charge, is usually described as a fundamental quantity. Time can be combined mathematically with other physical quantities to derive other concepts such as motion, kinetic energy and time-dependent fields. *Timekeeping* is a complex of technological and scientific issues, and part of the foundation of *recordkeeping*.

As our understanding of physics changes, so may our concept of time.

Defining & measuring time

1. State theory of physics as a mathematical formalism. Identify time as a term in that formalism.
2. Translate between map and territory (formalism vs. real world) by identifying formal states that correspond to direct observables, i.e., human perceptual inputs.
3. Identify observable events predicted to occur at uniform time intervals under certain conditions.
4. Define unit of measure as a specified multiple of that interval.
5. Validate theory by repeatedly measuring other things predicted to be fixed; verify that results are consistent.

One definition: **atomic time**

Fundamental unit is the *second*, defined as “the duration of 9,192,631,770 periods of the radiation corresponding to the transition between the two hyperfine levels of the ground state of the caesium-133 atom”.

International Atomic Time (abbrev. TAI for *Temps Atomique International*) is kept as a weighted average of several hundred atomic clocks located in various national laboratories.

Every minute of TAI is precisely 60 seconds, every hour precisely 60 minutes, every day precisely 24 hours.

Another notion: **universal time**

Fundamental unit is the *stellar day*, defined as the time it takes the earth to make one complete rotation on its axis, relative to distant background stars.

One stellar day is *approximately* 86,348 atomic seconds (not 86,400 – that's a solar day), but the ratio is not perfectly fixed, because earth's rotation speed varies due to seismic and tidal forces.

Tracked through astronomical observations by the International Earth Rotation and Reference Systems Service (IERS). Observation procedures are revised every few years.

Various versions of universal time are UT0, UT1, UT1R, UT2. UT0 is the one most closely tied to raw data. Others apply various smoothing and corrections. UT1 is the most widely-used.

Civil time

Civil time is a compromise: **UTC**.

Mostly, it moves in lockstep with TAI.

But, soon after UTC diverges from UT1 by more than half a second, a *leap second* is added (or, theoretically, removed) to UTC. Adding a leap second causes the final minute of June 31 or December 31 to have 61 seconds.

As of the most recent leap second, in June 2015, UTC is 26 seconds behind TAI. UTC is always within 0.9 seconds of UT1, and always differs from TAI by an integer number of seconds.

How consumer devices keep time

Consumer-grade devices that need to keep time (PCs, wall clocks, etc.) typically use a quartz oscillator (abbreviated XO).

Quartz is *piezoelectric*: mechanical stress causes electric potential, and *vice versa*. Put an amplifier across it, creating a positive feedback loop, and you get a nice sine wave at the resonant frequency of the crystal.

By knowing the resonant frequency and counting oscillations, we can keep time with decent precision.

Photo credit: `oomlout.co.uk`

Sources of drift

1. **Manufacturing tolerance** — Typcial low-cost XOs have a rated tolerance of 50 ppm (parts per million). That's over 26 minutes per year!
2. **Aging** — Slow change (over years) in resonant frequency due primarily to outgassing.
3. **Temperature** — Thermal expansion & contraction alter resonant frequency.
4. **Wander** — Seemingly random fluctuation due to everything else we can't measure or control. Imperfections in crystal structure contribute to greater wander.

Dealing with tolerance & aging

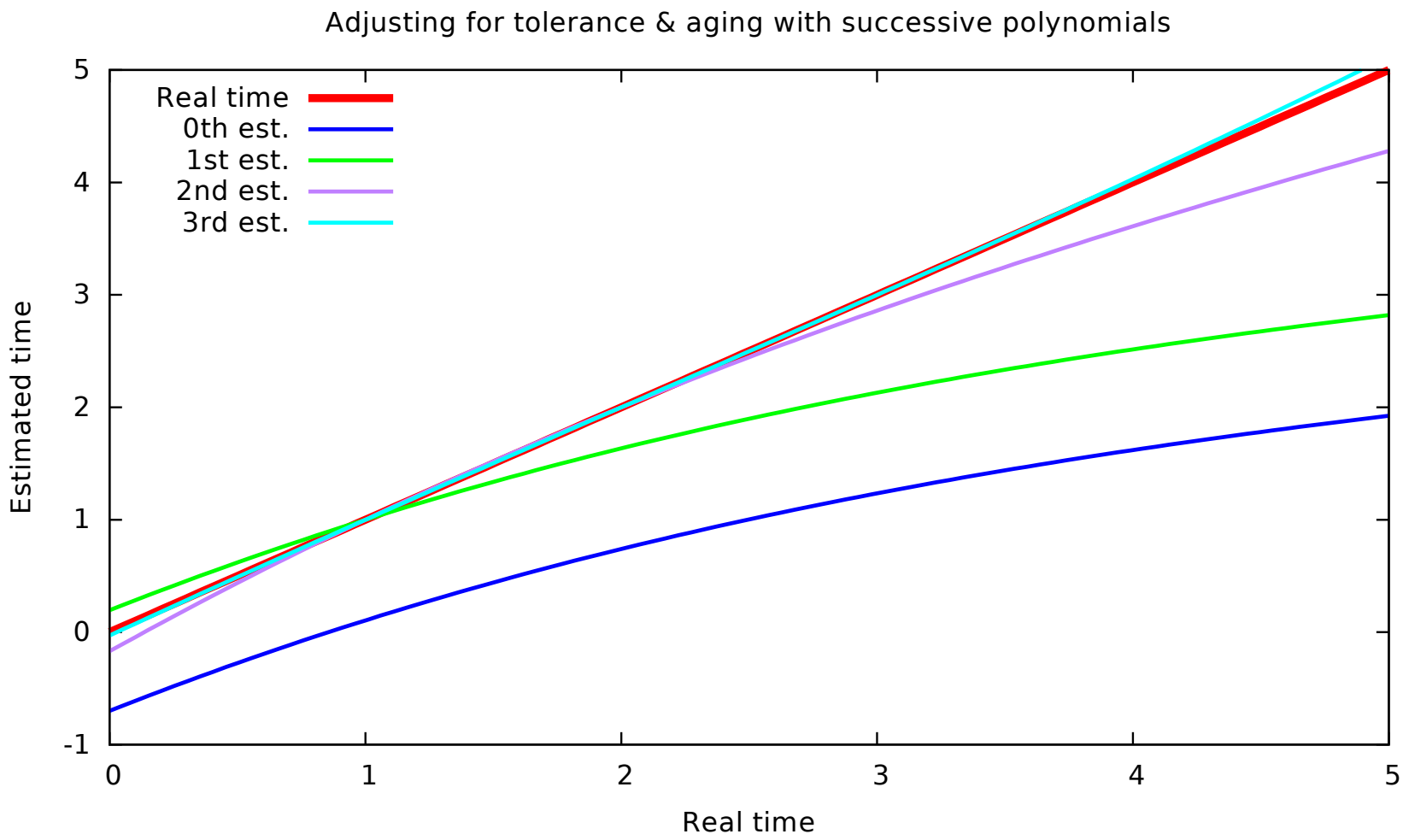
Suppose tolerance and aging are the *only* sources of drift.

Suppose also we have (just) a few opportunities to determine our clock's *exact* offset from a known-accurate reference clock. What can be done?

- **One sample:** set the clock (but then it will drift)
- **Two samples:** determine oscillator frequency
- **Three samples:** get a linear approximation of aging curve
- **Four samples:** refine to a quadratic approximation

Aging is slow enough that a quadratic approximation is likely more than adequate throughout the lifetime of our hardware. So we could take these four samples and then keep our clock accurate indefinitely!

Approxmiation by successive polynomials

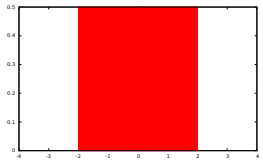
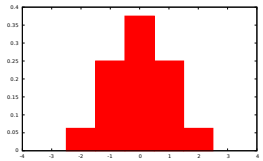
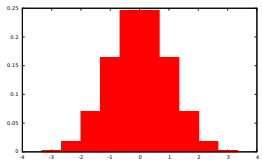
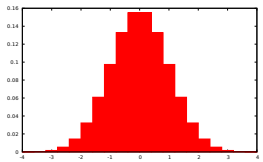
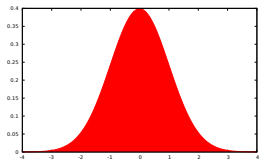


Options for dealing with temperature fluctuation

- Best: don't let it fluctuate! Keep the XO in a temperature-controlled chamber. (OCXO: Oven-controlled crystal oscillator)
- Cut the crystal into a geometry that minimizes the effect of temperature. (TCXO: Temperature-compensated crystal oscillator)
- Monitor temperature and model its effect. (MCXO: microcomputer-compensated crystal oscillator)
- Or, just punt: treat temperature fluctuation as a component of wander.

Random walks

Every $\frac{1}{n^2}$ seconds, flip a coin and step $\frac{1}{n}$ meters forward or back. Where will you end up?

Step size	Time between steps	Position at $t = 1$
± 1	1	
$\pm \frac{1}{2}$	$\frac{1}{4}$	
$\pm \frac{1}{3}$	$\frac{1}{9}$	
$\pm \frac{1}{5}$	$\frac{1}{25}$	
$\pm \epsilon$	ϵ^2	

Wander as a random walk

An oscillator's wander can be modeled as a random walk. The speed at which it wanders is written ω. If the RMS (root-mean-square) difference between the oscillator's speed at the beginning and end of the day is 3 ppm, then

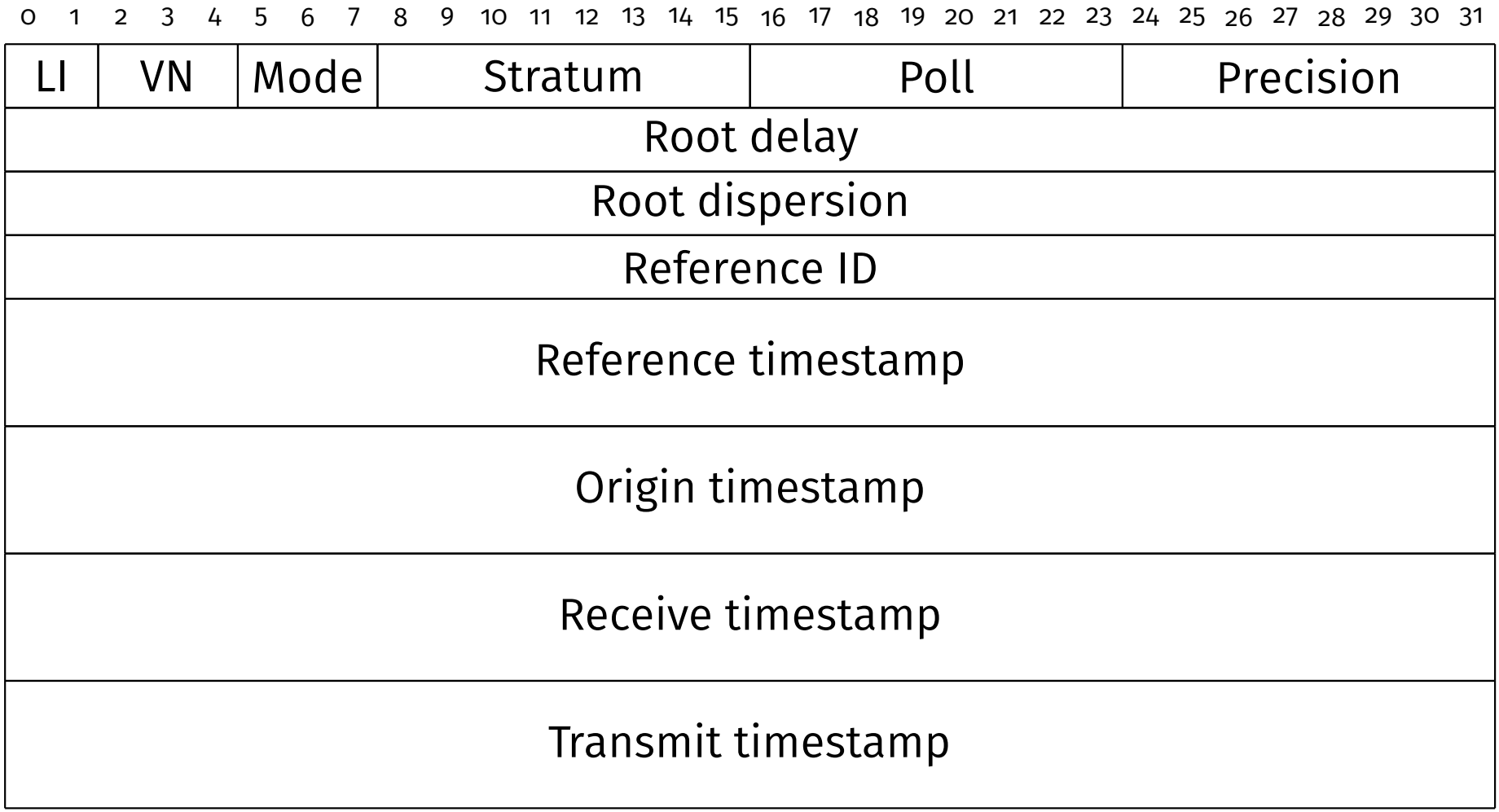
$$\omega = \frac{3\ \text{ppm}}{\sqrt{\text{day}}}.$$

For a lot of calculations it's easier to work with ω^2:

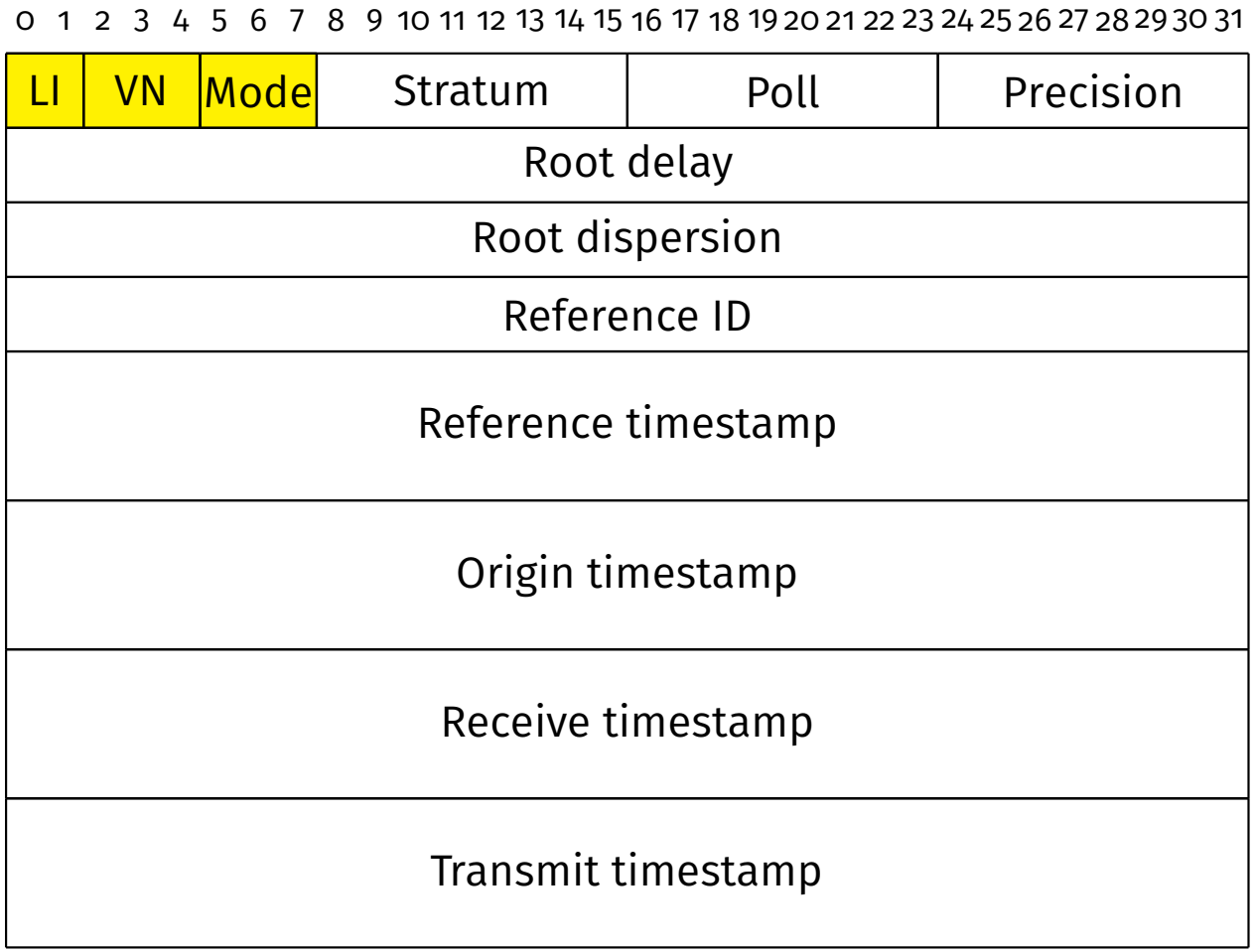
$$\omega^2 = \frac{9\ \text{ppt}}{\text{day}}.$$

ppt is a dimensionless ratio (10^{-12}), so we could write this as 104 aHz (attohertz!) if we so desired.

NTP packet

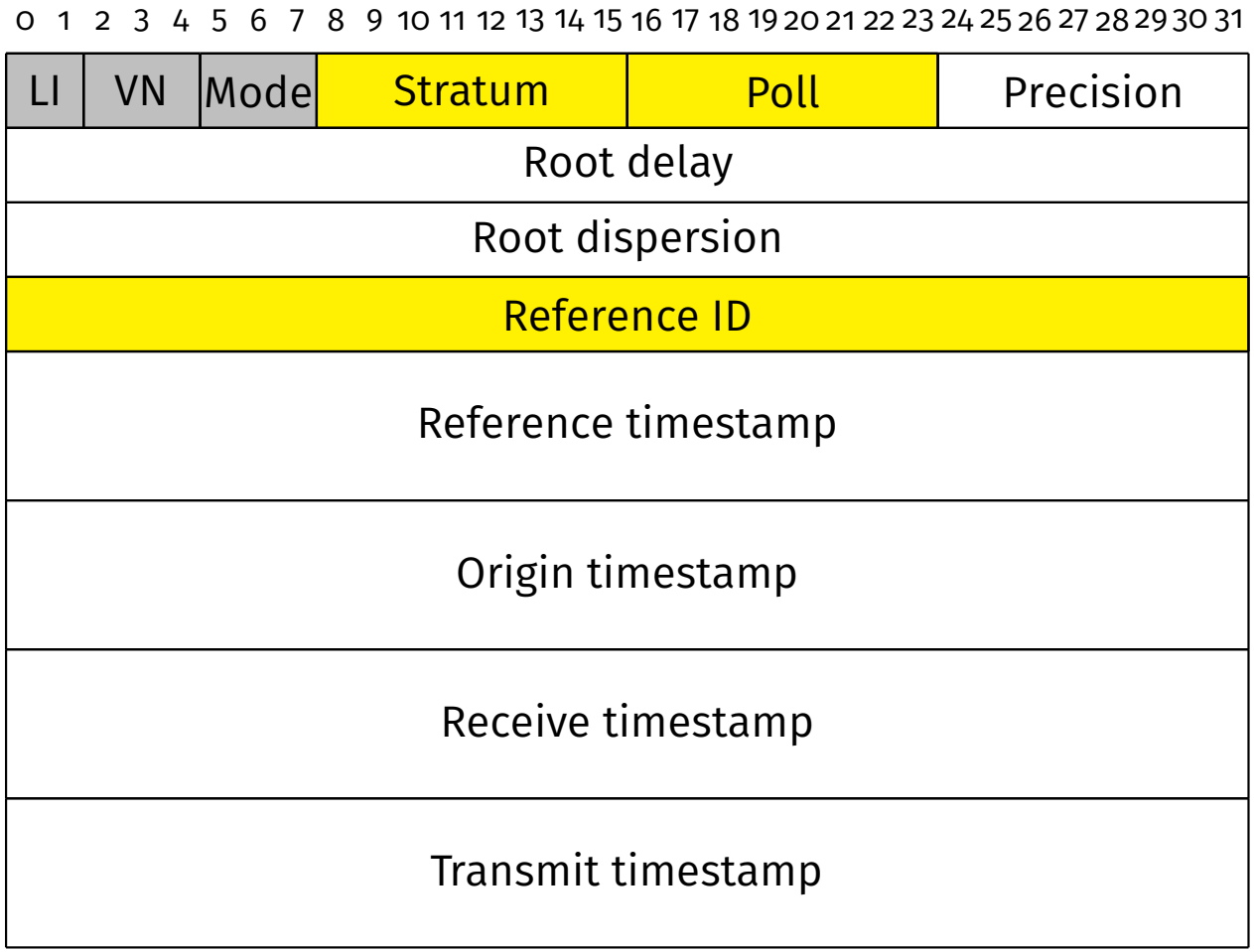

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
LI
VN
Mode
Stratum
Poll
Precision
Root delay
Root dispersion
Reference ID
Reference timestamp
Origin timestamp
Receive timestamp
Transmit timestamp

Uninteresting fields



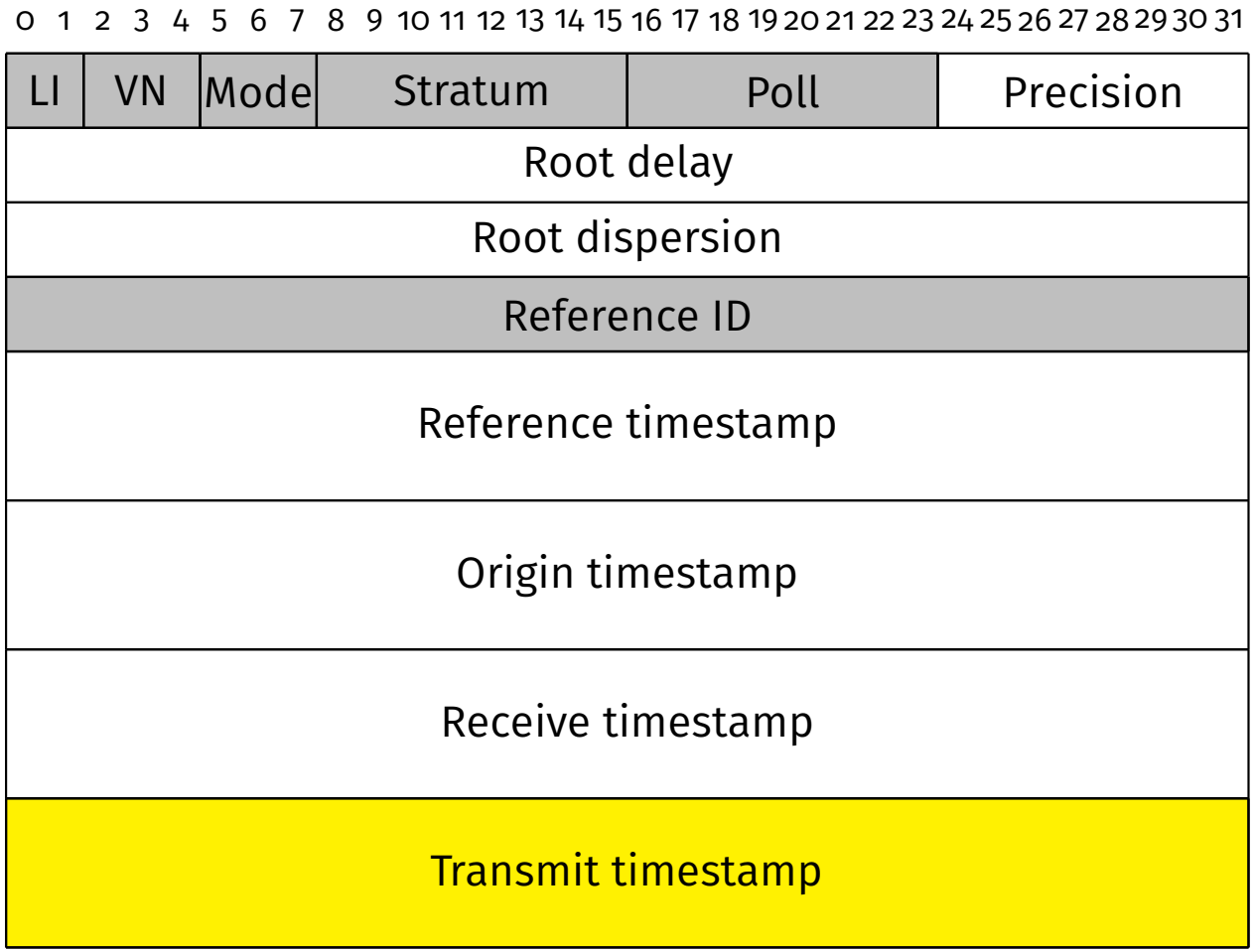
- LI – Notifies of upcoming leap second
- VN – Protocol version (currently 4)
- Mode – 3 for queries, 4 for responses, other values for less-used modes

Uninteresting fields (ctd.)



- Stratum — # of hops away from reference clock
- Poll — Polling interval
- Reference ID — Code identifying reference clock

The stupidest thing which could possibly work



1. Client asks server what time it is
2. Server sends back time in “Transmit timestamp” field
3. Client sets clock from this field

Problem: this ignores network latency

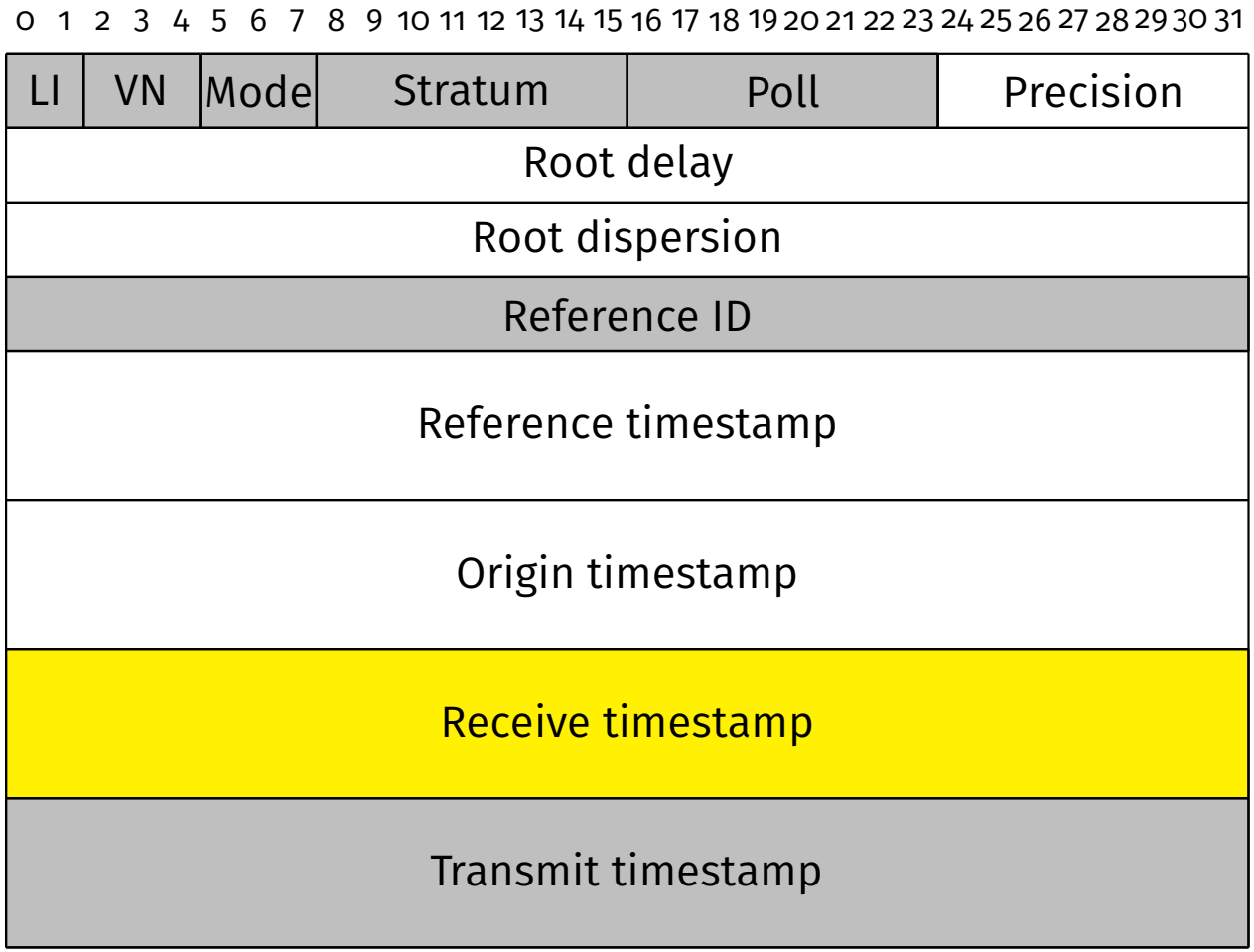
Dealing with network latency

Network latency is usually approximately symmetrical. The time for a packet to travel from host A to host B is approximately the same as the time to travel from host B to host A.

Round-trip latency is the *difference* between two timestamps, so as long as your system oscillator is sane, you can measure it without requiring a correctly-set clock.

So, approximate one-way latency as half of round-trip latency and adjust accordingly.

A slight optimization



Busy servers might not be able to respond right away, so record when the request was received as well as when the response was sent.

NTP's five timestamps

- t_{ref}, reference timestamp — time at which the local clock was last corrected (mainly useful for diagnostics)
- t_1, origin timestamp — time, according to the *client*, when *request was sent*.
- t_2, receive timestamp — time, according to the *server*, when *request was received*.
- t_3, transmit timestamp — time, according to the *server*, when *response was sent*.
- t_4, destination timestmap — time, according to the *client*, when *response was received*.

RTT and time offset

RTT:

$$\delta = (t_4 - t_1) - (t_3 - t_2)$$

Offset:

$$\theta = t_{\text{server}} - t_{\text{client}}$$

$$\hat{\theta} = \frac{(t_2 - t_1) + (t_3 - t_4)}{2}$$

Warning: RFC5905 uses the same symbol θ for both these concepts.

Clock discipline

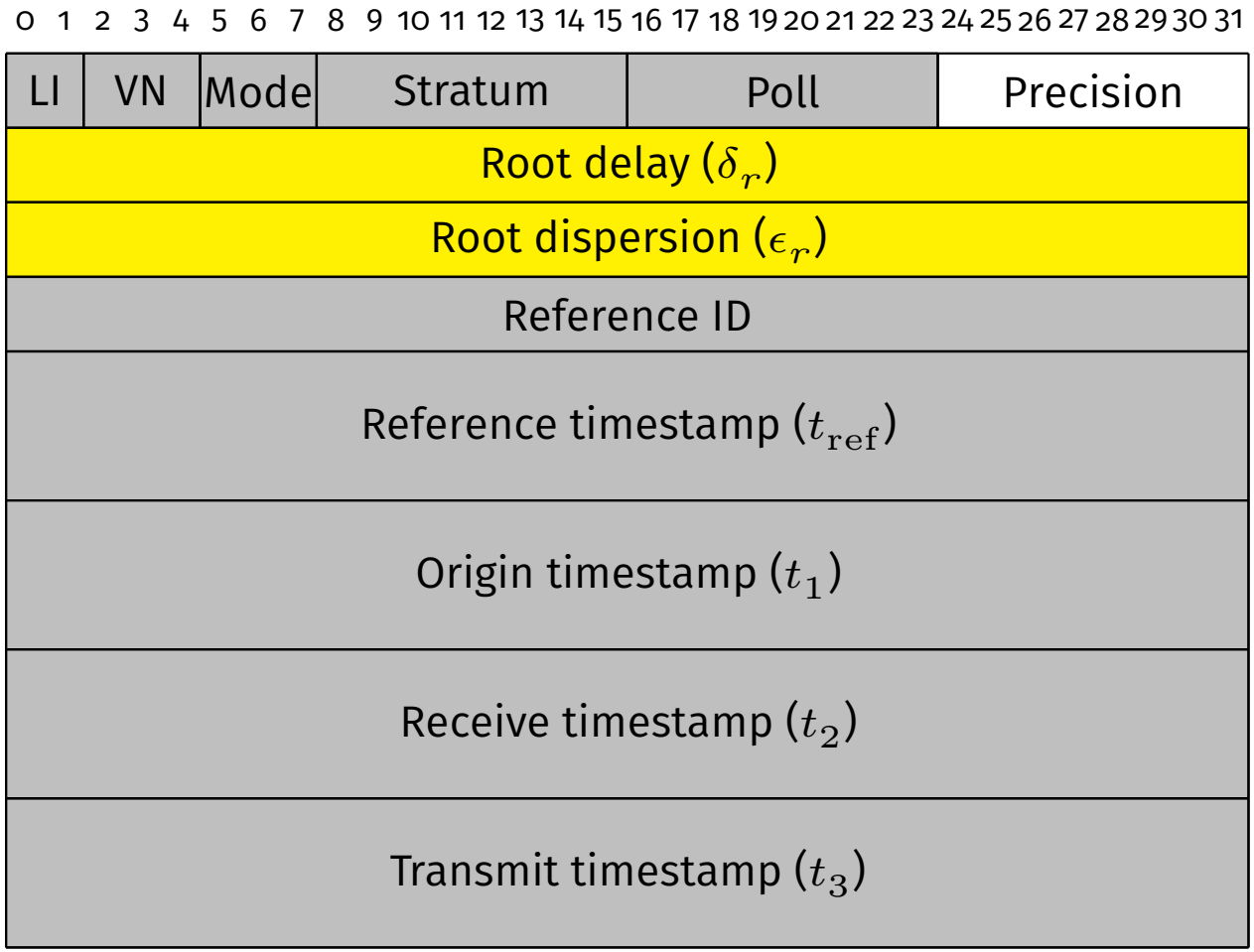
Sudden clock adjustments cause problems for applications. So use a *phase-locked loop* (PLL) to move it gradually.

After obtaining $\hat{\theta}$, set the *residual offset* $\Theta = \hat{\theta}$.

Then continuously adjust the clock speed so that Θ decays exponentially.

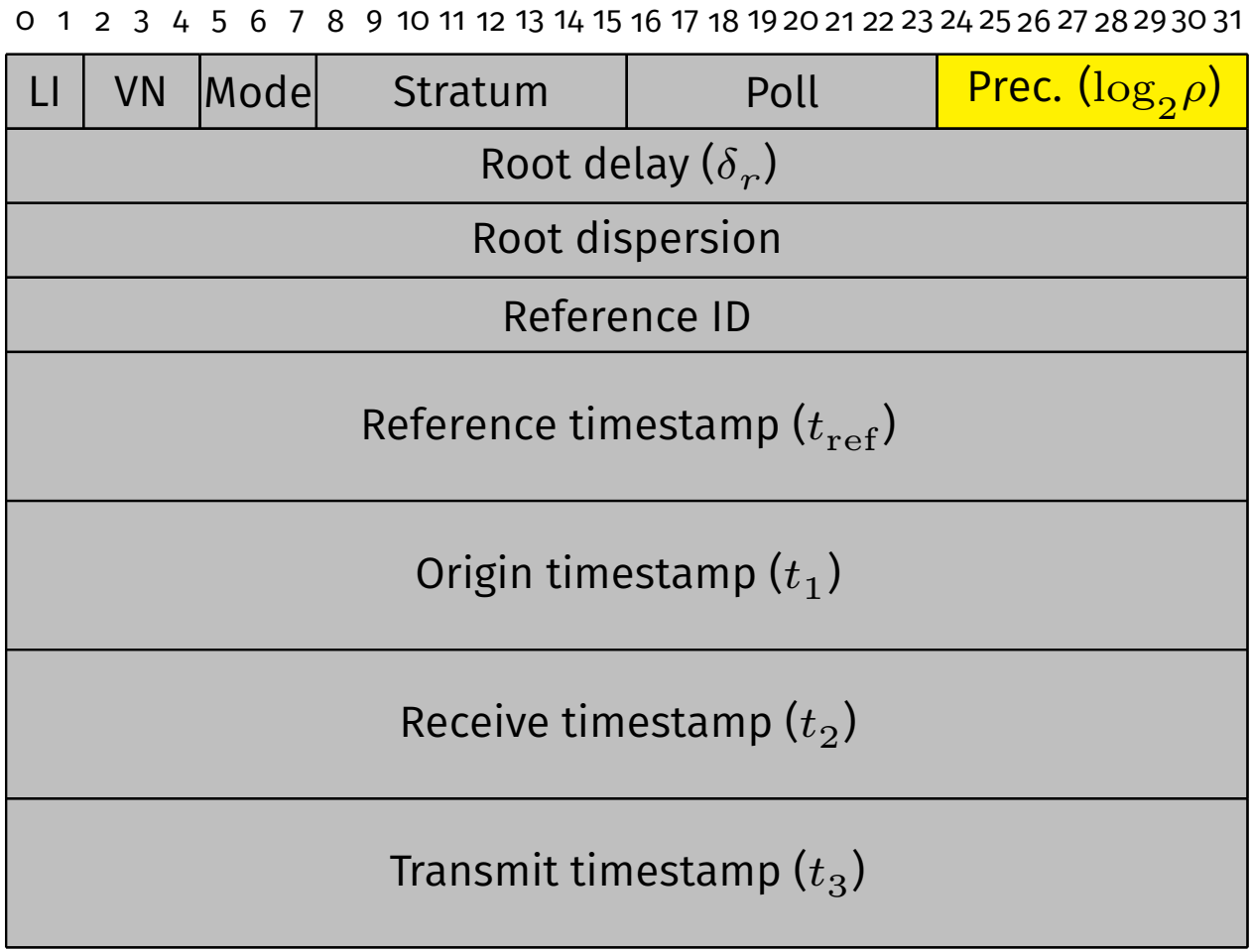
A PLL with a long time constant also helps average together multiple samples.

Root delay & dispersion



- Root delay: Total round trip latency between server and reference clock
- Root dispersion: Server's estimated error vs. reference clock

Precision



- Precision: inherent error in obtaining system clock measurements due to hardware & os delays
- Encoded on wire to nearest power of 2
- Typical values: 250ns – 4µs

Maximum Error

System constant for maximum plausible frequency error:

$$\Phi = 15\text{ppm}$$

Maximum plausible clock measurement error:

$$\epsilon = \rho_{\text{server}} + \rho_{\text{client}} + \Phi(t_4 - t_1)$$

Maximum plausible error $\left|\hat{\theta} - \theta\right|$, aka synchronization distance:

$$\lambda = \frac{\delta}{2} + \epsilon$$

(In practice, $\frac{\delta}{2} \gg \epsilon$)

Maximum Error

System constant for maximum plausible frequency error:

$$\Phi = 15\text{ppm}$$

Maximum plausible clock measurement error:

$$\epsilon = \rho_{\text{server}} + \rho_{\text{client}} + \Phi(t_4 - t_1)$$

Maximum plausible error $\left|\hat{\theta} - \theta\right|$, aka synchronization distance:

$$\lambda = \frac{\delta}{2} + \epsilon$$

(In practice, $\frac{\delta}{2} \gg \epsilon$)

Falseticker rejection

Given several time sources and a $\hat{\theta}$ and λ statistic for each, say that source i is plausible to source j if

$$\hat{\theta}_j - \lambda_j < \hat{\theta}_i < \hat{\theta}_j + \lambda_j.$$

Find a maximal clique of mutually-plausible sources. Members of the clique are called *truechimers*. Those outside it are called *falsetickers* and their data are discarded.

Peer jitter

Given a sequence of n queries to a server, we obtain $\hat{\theta}$ statistics $\hat{\theta}_1, \hat{\theta}_2, \ldots, \hat{\theta}_n$.

The *peer jitter* statistic ψ_p is computed (somewhat oddly) as

$$\psi = \frac{\sqrt{\sum_{i=2}^{n} (\hat{\theta}_1 - \hat{\theta}_i)^2}}{n-1}.$$

The ordinarily-defined network jitter

$$\frac{\sqrt{\sum_{i=1}^{n} \left(\bar{\delta} - \delta_j\right)^2}}{n-1} \quad \text{where} \quad \bar{\delta} = \frac{\sum_{i=1}^{n} \delta_i}{n}$$

should be roughly equal to $\psi\sqrt{2}$.

Selection jitter

Given roughly-simulatenous queries to n different servers, the *selection jitter* ψ_s for the server with index i is defined as

$$\frac{\sqrt{\sum_{j=i}^{n}(\hat{\theta}_i - \hat{\theta}_j)^2}}{n-1}.$$

This is a measure of how much of an outlier that server's $\hat{\theta}$ statistic is compared to those of the other servers.

Clustering

Including outlying servers, even if they make the cut as truechimers, may introduce more noise than signal.

If the worst ψ_s is higher than the best ψ_p, remove that worst server. Recalculate and repeat as necessary.

Total jitter:

$$\psi = \sqrt{\psi_p^2 + \psi_s^2}$$

.

Combining

Root synchronization distance for a server:

$$\Lambda = \frac{\delta + \delta_r}{2} + \epsilon_r + \epsilon + \psi$$

System offset is set to weighted average of all surviving servers:

$$\Theta = \frac{\sum(\Lambda_i^{-1}\hat{\theta}_i)}{\sum \Lambda_i^{-1}}$$

Error estimation

The surviving server with the best ψ_p is the *system peer*. The *system jitter* Ψ is the system peer's ψ.

System root delay:

$$\Delta = \delta + \delta_r$$

System root dispersion:

$$\mathrm{E} = \epsilon_r + \epsilon + \psi + \Phi\tilde{t} + \Theta$$

where $\tilde{t}$ is time elapsed since last query. Root delay & dispersion serve (roughly) as maximum & estimated error.