subnet to a given server $i$ are the clock offset $\Theta_i$, roundtrip delay $\Delta_i$ and dispersion $E_i$ inherited by and characteristic of that server.

| Variable | Description |
|---|---|
| $r$ | reading error |
| $\rho$ | max reading error |
| $f$ | frequency error |
| $\varphi$ | max frequency error |
| $\theta$, $\Theta$ | clock offset |
| $\delta$, $\Delta$ | roundtrip delay |
| $\varepsilon$, $E$ | error/dispersion |
| $t$ | time |
| $\tau$ | time interval |
| $T$ | NTP timestamp |
| $s$ | clock divider increment |
| $f_c$ | clock oscillator frequency |

Table 4. Notation Used in Error Analysis

The NTP clock-filter procedure saves the most recent samples $\theta_i$ and $\delta_i$ in the clock filter as described in the NTP specification. The quantities $\rho$ and $\varphi$ characterize the local clock maximum reading error and frequency error, respectively. Each sample includes the dispersion $\varepsilon_i = \rho + \varphi(T_4 - T_1)$, which is set upon arrival. Each time a new sample arrives all samples in the filter are updated with the skew dispersion $\varphi\tau_i$, where $\tau_i$ is the interval since the last sample arrived, as recorded in the variable peer.update. The clock-filter algorithm determines the selected clock offset $\theta$ (peer.offset), together with the associated roundtrip delay $\delta$ (peer.delay) and filter dispersion $\varepsilon_\sigma$, which is added to the associated sample dispersion $\varepsilon_i$ to form the peer dispersion $\varepsilon$ (peer.dispersion).

The NTP clock-selection procedure selects a single peer to become the synchronization source as described in the NTP specification. The operation of the algorithm determines the final clock offset $\Theta$ (local clock), roundtrip delay $\Delta$ (sys.rootdelay) and dispersion E (sys.rootdispersion) relative to the root of the synchronization subnet, as shown in Figure 15. Note the inclusion of the selected peer dispersion and skew accumulation since the dispersion was last updated, as well as the select dispersion $\varepsilon_\xi$ computed by the clock-select algorithm itself. Also, note that, in order to preserve overall synchronization subnet stability, the final clock offset $\Theta$ is in fact determined from the offset of the local clock relative to the peer clock, rather than the root of the subnet. Finally, note that the packet variables $\Delta'$ and E′ are in fact determined from the latest message received, not at the precise time the offset selected by the clock-filter algorithm was determined. Minor errors arising due to these simplifications will be ignored. Thus, the total dispersion accumulation relative to the root of the synchronization subnet is

$$E = \varepsilon + \varphi\tau + \varepsilon_\xi + |\Theta| + E' ,$$

where $\tau$ is the time since the peer variables were last updated and $|\Theta|$ is the initial absolute error in setting the local clock.

The three values of clock offset, roundtrip delay and dispersion are all additive; that is, if $\Theta_i$, $\Delta_i$ and $E_i$ represent the values at peer $i$ relative to the root of the synchronization subnet, the values

$$\Theta_j(t) \equiv \Theta_i + \theta_j(t) , \quad \Delta_j(t) \equiv \Delta_i + \delta_j , \quad E_j(t) \equiv E_i + \varepsilon_i + \varepsilon_j(t) ,$$

represent the clock offset, roundtrip delay and dispersion of peer $j$ at time $t$. The time dependence of $\theta_j(t)$ and $\varepsilon_j(t)$ represents the local-clock correction and dispersion accumulated since the last update was received from peer $i$, while the term $\varepsilon_i$ represents the dispersion accumulated by peer $i$ from the time its clock was last set until the latest update was sent to peer $j$. Note that, while the offset of the local clock relative to the peer clock can be determined directly, the offset relative to the root of the synchronization subnet is not directly determinable, except on a probabilistic basis and within the bounds established in this and the previous section.

The NTP synchronization subnet topology is that of a tree rooted at the primary server(s). Thus, there is an unbroken path from every time server to the primary reference source. Accuracy and stability are proportional to synchronization distance $\Lambda$, defined as

$$\Lambda \equiv E + \frac{\Delta}{2} .$$

The selection algorithm favors the minimum-distance paths and thus maximizes accuracy and stability. Since $\Theta_0$, $\Delta_0$ and $E_0$ are all zero, the sum of the clock offsets, roundtrip delays and dispersions of each server along the minimum-distance path from the root of the synchronization

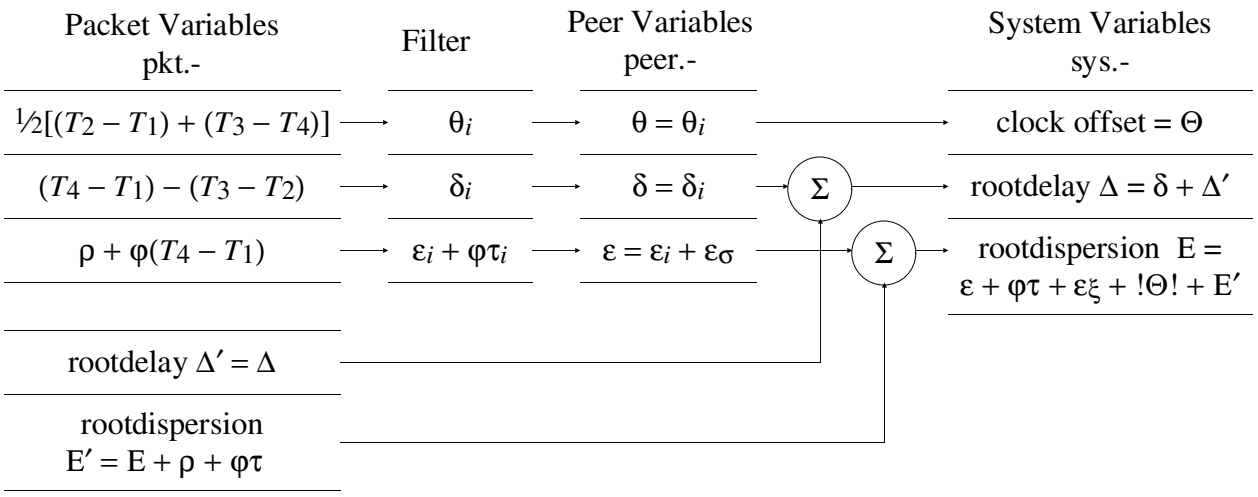


Figure 9. Error Accumulations

An NTP measurement update includes three parts: clock offset θ, roundtrip delay δ and maximum error or dispersion ε of the local clock relative to a peer clock. In case of a primary clock update, these values are usually all zero, although ε can be tailored to reflect the specified maximum error of the primary reference source itself. In other cases θ and δ are calculated directly from the four most recent timestamps, as described in the NTP specification. The dispersion ε includes the following contributions:

1. Each time the local clock is read a reading error is incurred due to the finite granularity or precision of the implementation. This is called the measurement dispersion ρ.

2. Once an offset is determined, an error due to frequency offset or skew accumulates with time. This is called the skew dispersion φτ, where φ represents the skew-rate constant ($\frac{\text{NTP.MAXSKEW}}{\text{NTP.MAXAGE}}$ in the NTP specification) and τ is the interval since the dispersion was last updated.

3 When a series of offsets are determined at regular intervals and accumulated in a window of samples, as in the NTP clock-filter algorithm, the (estimated) additional error due to offset sample variance is called the filter dispersion $\varepsilon_\sigma$.

4. When a number of peers are considered for synchronization and two or more are determined to be correctly synchronized to a primary reference source, as in the NTP clock-selection algorithm, the (estimated) additional error due to offset sample variance is called the selection dispersion εξ.

Figure 7 shows how these errors accumulate in the ordinary course of NTP processing. Received messages from a single peer are represented by the packet variables. From the four most recent timestamps $T_1$, $T_2$, $T_3$ and $T_4$ the clock offset and roundtrip delay sample for the local clock relative to the peer clock are calculated directly. Included in the message are the root roundtrip delay Δ′ and root dispersion E′ of the peer itself; however, before sending, the peer adds the measurement dispersion ρ and skew dispersion φτ, where these quantities are determined by the peer and τ is the interval according to the peer clock since its clock was last updated.

exercise to calculate bounds on clock offset errors as a function of measured delay. Let $T_2 - T_1 = a$ and $T_3 - T_4 = b$. Then,

$$\delta = a - b \quad \text{and} \quad \theta = \frac{a + b}{2} .$$

The true offset of $B$ relative to $A$ is called $\theta_0$ in Figure 14. Let $x$ denote the actual delay between the departure of a message from $A$ and its arrival at $B$. Therefore, $x + \theta_0 = T_2 - T_1 \equiv a$. Since $x$ must be positive in our universe, $x = a - \theta_0 \geq 0$, which requires $\theta_0 \leq a$. A similar argument requires that $b \leq \theta_0$, so surely $b \leq \theta_0 \leq a$. This inequality can also be expressed

$$b = \frac{a + b}{2} - \frac{a - b}{2} \leq \theta_0 \leq \frac{a + b}{2} + \frac{a - b}{2} = a ,$$

which is equivalent to

$$\theta - \frac{\delta}{2} \leq \theta_0 \leq \theta + \frac{\delta}{2} .$$

In the previous section bounds on delay and offset errors were determined. Thus, the inequality can be written

$$\theta - \varepsilon_\theta - \frac{\delta + \varepsilon_\delta}{2} \leq \theta_0 \leq \theta + \varepsilon_\theta + \frac{\delta + \varepsilon_\delta}{2} ,$$

where $\varepsilon_\theta$ is the maximum offset error and $\varepsilon_\delta$ is the maximum delay error derived previously. The quantity

$$\varepsilon = \varepsilon_\theta + \frac{\varepsilon_\delta}{2} = \rho + \varphi_A(T_4 - T_1) + \varphi_B(T_3 - T_2) ,$$

called the peer dispersion, defines the maximum error in the inequality. Thus, the correctness interval $I$ can be defined as the interval

$$I = [\theta - \frac{\delta}{2} - \varepsilon, \theta + \frac{\delta}{2} + \varepsilon] ,$$

in which the clock offset $C = \theta$ is the midpoint. By construction, the true offset $\theta_0$ must lie somewhere in this interval.

### 4.4. Inherited Errors

As described in the NTP specification, the NTP time server maintains the local clock $\Theta$, together with the root roundtrip delay $\Delta$ and root dispersion E relative to the primary reference source at the root of the synchronization subnet. The values of these variables are either included in each update message or can be derived as described in the NTP specification. In addition, the protocol exchange and clock-filter algorithm provide the clock offset $\theta$ and roundtrip delay $\delta$ of the local clock relative to the peer clock, as well as various error accumulations as described below. The following discussion establishes how errors inherent in the time-transfer process accumulate within the subnet and contribute to the overall error budget at each server.

For specific peers $A$ and $B$, where $f_A$ and $f_B$ can be considered constants, the interval containing the maximum error inherent in determining $\delta$ is given by

$$[\min(\varepsilon_4) - \max(\varepsilon_1) - \max(\varepsilon_3) + \min(\varepsilon_2), \max(\varepsilon_4) - \min(\varepsilon_1) - \min(\varepsilon_3) + \max(\varepsilon_2)]$$
$$= [-\rho_A - \rho_B, \rho_A + \rho_B] + f_A(T_4 - T_1) - f_B(T_3 - T_2) \ .$$

In the NTP local clock model the residual frequency errors $f_A$ and $f_B$ are minimized through the use of a type-II phase-lock loop (PLL). Under most conditions these errors will be small and can be ignored. The pdf for the remaining errors is symmetric, so that $\hat{\delta} = <\delta>$ is an unbiased maximum-likelihood estimator for the true roundtrip delay, independent of the particular values of $\rho_A$ and $\rho_B$.

However, in order to reliably bound the errors under all conditions of component variation and operational regimes, the design of the PLL and the tolerance of its intrinsic oscillator must be controlled so that it is not possible under any circumstances for $f_A$ or $f_B$ to exceed the bounds $[-\varphi_A, \varphi_A]$ or $[-\varphi_B, \varphi_B]$, respectively. Setting $\rho = \max(\rho_A, \rho_B)$ for convenience, the absolute maximum error $\varepsilon_\delta$ inherent in determining roundtrip delay $\delta$ is given by

$$\varepsilon_\delta \equiv \rho + \varphi_A(T_4 - T_1) + \varphi_B(T_3 - T_2) \ ,$$

neglecting residuals.

As in the case for $\delta$, where $f_A$ and $f_B$ can be considered constants, the interval containing the maximum error inherent in determining $\theta$ is given by

$$\frac{[\min(\varepsilon_2) - \max(\varepsilon_1) + \min(\varepsilon_3) - \max(\varepsilon_4), \max(\varepsilon_2) - \min(\varepsilon_1) + \max(\varepsilon_3) - \min(\varepsilon_4)]}{2}$$
$$= [-\rho_B, \rho_A] + \frac{f_B(T_3 - T_2) - f_A(T_4 - T_1)}{2} \ .$$

Under most conditions the errors due to $f_A$ and $f_B$ will be small and can be ignored. If $\rho_A = \rho_B = \rho$; that is, if both the $A$ and $B$ clocks have the same resolution, the pdf for the remaining errors is symmetric, so that $\hat{\theta} = <\theta>$ is an unbiased maximum-likelihood estimator for the true clock offset $\theta_0$, independent of the particular value of $\rho$. If $\rho_A \neq \rho_B$, $<\theta>$ is not an unbiased estimator; however, the bias error is in the order of

$$\frac{\rho_A - \rho_B}{2} \ .$$

and can usually be neglected.

Again setting $\rho = \max(\rho_A, \rho_B)$ for convenience, the absolute maximum error $\varepsilon_\theta$ inherent in determining clock offset $\theta$ is given by

$$\varepsilon_\theta \equiv \frac{\rho + \varphi_A(T_4 - T_1) + \varphi_B(T_3 - T_2)}{2} \ .$$

### 4.3. Network Errors

In practice, errors due to stochastic network delays usually dominate. In general, it is not possible to characterize network delays as a stationary random process, since network queues can grow and shrink in chaotic fashion and arriving customer traffic is frequently bursty. However, it is a simple

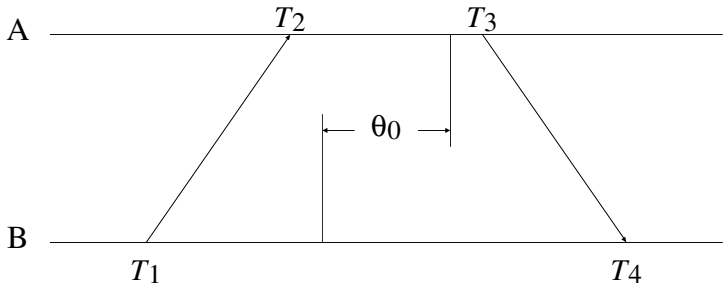

Figure 8. Measuring Delay and Offset

and $p_f(x)$ for $r$ and $f$ respectively. Assuming the clock reading and counting processes are independent, the pdf for $r$ is uniform over the interval $[-\rho, 0]$. With conventional manufacturing processes and temperature variations the pdf for $f$ can be approximated by a truncated, zero-mean Gaussian distribution with standard deviation $\sigma$. In conventional manufacturing processes $\sigma$ is maneuvered so that the fraction of samples rejected outside the interval $[-\varphi, \varphi]$ is acceptable. The pdf for the total timestamp error $\varepsilon(x)$ is thus the sum of the $r$ and $f$ contributions, computed as

$$\varepsilon(x) = \int_{-\infty}^{\infty} p_r(t) p_f(x - t) dt ,$$

which appears as a bell-shaped curve, symmetric about $-\frac{\rho}{2}$ and bounded by the interval

$$[\min(r) + \min(f\tau), \max(r) + \max(f\tau)] = [-\rho - \varphi\tau, \varphi\tau] .$$

Since $f$ changes only slowly over time for any single clock,

$$\varepsilon \equiv [\min(r) + f\tau, \max(r) + f\tau] = [-\rho, 0] + f\tau ,$$

where $\varepsilon$ without argument designates the interval and $\varepsilon(x)$ designates the pdf. In the following development subscripts will be used on various quantities to indicate to which entity or timestamp the quantity applies. Occasionally, $\varepsilon$ will be used to designate an absolute maximum error, rather than the interval, but the distinction will be clear from context.

### 4.2. Measurement Errors

In NTP the roundtrip delay and clock offset between two peers $A$ and $B$ are determined by a procedure in which timestamps are exchanged via the network paths between them. The procedure involves the four most recent timestamps numbered as shown in Figure 6, where the $\theta_0$ represents the true clock offset of peer $B$ relative to peer $A$. The $T_1$ and $T_4$ timestamps are determined relative to the $A$ clock, while the $T_2$ and $T_3$ timestamps are determined relative to the $B$ clock. The measured roundtrip delay $\delta$ and clock offset $\theta$ of $B$ relative to $A$ are given by

$$\delta = (T_4 - T_1) - (T_3 - T_2) \quad \text{and} \quad \theta = \frac{(T_2 - T_1) + (T_3 - T_4)}{2} .$$

The errors inherent in determining the timestamps $T_1$, $T_2$, $T_3$ and $T_4$ are, respectively,

$$\varepsilon_1 = [-\rho_A, 0], \ \varepsilon_2 = [-\rho_B, 0], \ \varepsilon_3 = [-\rho_B, 0] + f_B(T_3 - T_2), \ \varepsilon_4 = [-\rho_A, 0] + f_A(T_4 - T_1) .$$

### 4.1. Timestamp Errors

The standard second (1 s) is defined as "9,192,631,770 periods of the radiation corresponding to the transition between the two hyperfine levels of the ground state of the cesium-133 atom" [ALL74b], which implies a granularity of about $1.1 \times 10^{-10}$ s. Other intervals can be determined as rational multiples of 1 s. While NTP time has an inherent resolution of about $2.3 \times 10^{-10}$ s, local clocks ordinarily have resolutions much worse than this, so the inherent error in resolving NTP time relative to the 1 s can be neglected.

Let $T(t)$ be the time displayed by a clock at epoch $t$ relative to the standard timescale:

$$T(t) = \tfrac{1}{2}D(t_0)[t - t_0]^2 + R(t_0)[t - t_0] + T(t_0) + x(t) ,$$

where $D(t_0)$ is the fractional frequency drift per unit time, $R(t_0)$ the frequency and $T(t_0)$ the time at some previous epoch $t_0$. In the usual stationary model these quantities can be assumed constant or changing slowly with epoch. The random nature of the clock is characterized by $x(t)$, which represents the random noise (jitter) relative to the standard timescale. In the usual analysis the second-order term $D(t_0)$ is ignored and the noise term $x(t)$ modelled as a normal distribution with predictable spectral density or autocorrelation function.

The probability density function of time offset $p(t - T(t))$ usually appears as a bell-shaped curve centered somewhere near zero. The width and general shape of the curve are determined by $x(t)$, which depends on the oscillator precision and jitter characteristics, as well as the measurement system and its transmission paths. Beginning at epoch $t_0$ the offset is set to zero, following which the bell creeps either to the left or right, depending on the value of $R(t_0)$ and accelerates depending on the value of $D(t_0)$.

In this analysis the local clock is represented by a counter/divider which increments at intervals of $s$ seconds and is driven by an oscillator which operates at frequency $f_c = \frac{n}{s}$ for some integer $n$. A timestamp $T(t)$ is determined by reading the clock at an arbitrary time $t$ (the argument $t$ will be usually omitted for conciseness). Strictly speaking, $s$ is not known exactly, but can be assumed bounded from above by the maximum reading error $\rho$. The reading error itself is represented by the random variable $r$ bounded by the interval $[-\rho, 0]$, where $\rho$ depends on the particular clock implementation. Since the intervals between reading the same clock are almost always independent of and much larger than $s$, successive readings can be considered independent and identically distributed. The frequency error of the clock oscillator is represented by the random variable $f$ bounded by the interval $[-\varphi, \varphi]$, where $\varphi$ represents the maximum frequency tolerance of the oscillator throughout its service life. While $f$ for a particular clock is a random variable with respect to the population of all clocks, for any one clock it ordinarily changes only slowly with time and can usually be assumed a constant for that clock. Thus, an NTP timestamp can be represented by the random variable $T$:

$$T = t + r + f\tau ,$$

where $t$ represents a clock reading, $\tau$ represents the time interval since this reading and minor approximations inherent in the measurement of $\tau$ are neglected.

In order to assess the nature and expected magnitude of timestamp errors and the calculations based on them, it is useful to examine the characteristics of the probability density functions (pdf) $p_r(x)$

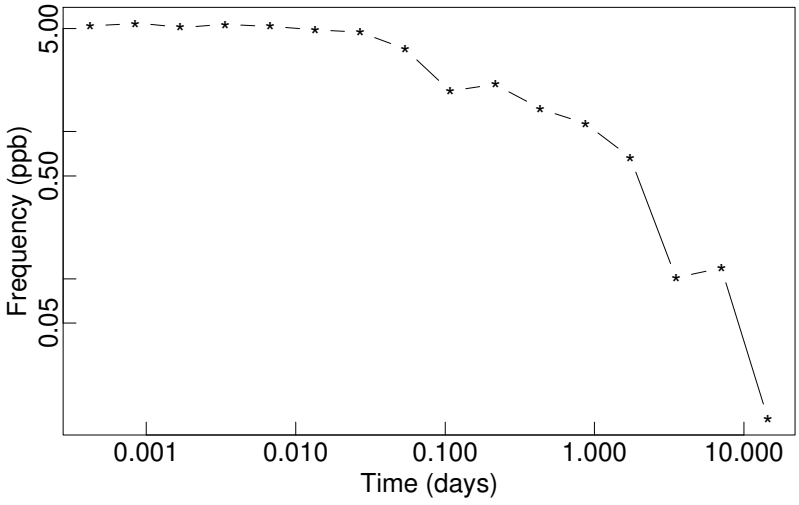

Figure 7. Allan Variance of WWVB Receiver

so that

$$\sigma_y^2(\tau) = \frac{1}{2(N-1)} \sum_{j=1}^{N-1} [y(j+1) - y(j)]^2 .$$

The Allan variance is particularly useful when comparing the intrinsic stability of the oscillators used in different local clocks and timecode receivers. The oscillators are uncoupled from all synchronization sources and left to flywheel for periods up to a month. At intervals of a minute or so the oscillator frequency is measured relative to a precision standard, such as a cesium oscillator or timing receiver. Figure  shows the Alan variance for a Spectracom 8170 WWVB timecode receiver measured at the 1-pps output. This reciever uses an uncompensated crystal oscillator disciplined to the WWVB signal and 60 kHz. The 1-pps output is extracted from the timecode modulation using a counter with a resolution of 100 µs. The resulting stability, averaged over periods up to two hours or so is about $5 \times 10^{-9}$.

## 4. Analysis of Errors and Correctness Principles

This appendix contains an analysis of errors arising in the generation and processing of NTP timestamps and the determination of delays and offsets. It establishes error bounds as a function of measured roundtrip delay and dispersion to the root (primary reference source) of the synchronization subnet. It also discusses correctness assertions about these error bounds and the time-transfer, filtering and selection algorithms used in NTP.

The notation $w = [u, v]$ in the following describes the interval in which $u$ is the lower limit and $v$ the upper limit, inclusive. Thus, $u = \min(w) \le v = \max(w)$, and for scalar $a$, $w + a = [u + a, v + a]$. Table 4 shows a summary of other notation used in the analysis. The notation $< x >$ designates the (infinite) average of $x$, which is usually approximated by an exponential average, while the notation $\hat{x}$ designates an estimator for $x$. The lower-case Greek letters $\theta$, $\delta$ and $\varepsilon$ are used to designate measurement data for the local clock to a peer clock, while the upper-case Greek letters $\Theta$, $\Delta$ and E are used to designate measurement data for the local clock relative to the primary reference source at the root of the synchronization subnet. Exceptions will be noted as they arise.

$$\hat{R}_i(t_0 + \tau) = \hat{R}_i(t_0) + \alpha_i[\hat{R}_i(t_0) - \frac{T_i(t_0 + \tau) - T_i(t_0)}{\tau}] ,$$

where $\alpha_i$ is an experimentally determined weight factor which depends on the estimated frequency error of the $i$th clock. In order to calculate the weight factor $w_i(\tau)$, it is necessary to determine the expected error $\varepsilon_i(\tau)$ for each clock. In the following, braces "|" indicate absolute value and brackets "<>" indicate the infinite time average. In practice, the infinite averages are computed as exponential time averages. An estimate of the magnitude of the unbiased error of the $i$th clock accumulated over the nominal interval $\tau$ is

$$\varepsilon_i(\tau) = |\hat{T}_i(t_0 + \tau) - T_i(t_0 + \tau)| + \frac{0.8 < \varepsilon_e^2(\tau) >}{< \varepsilon_i^2(\tau) >^{1/2}} ,$$

where $\varepsilon_i(\tau)$ and $\varepsilon_e(\tau)$ are the accumulated error of the $i$th clock and entire clock ensemble, respectively. The accumulated error of the entire ensemble is

$$< \varepsilon_e^2(\tau) > = \left[ \sum_{i=1}^{n} \frac{1}{< \varepsilon_i^2(\tau) >} \right]^{-1} .$$

Finally, the weight factor for the $i$th clock is calculated as

$$w_i(\tau) = \frac{< \varepsilon_e^2(\tau) >}{< \varepsilon_i^2(\tau) >} .$$

When all estimators and weight factors have been updated, the origin of the estimation interval is shifted and the new value of $t_0$ becomes the old value of $t_0 + \tau$.

While not entering directly into the above calculations, it is useful to estimate the frequency error, since the ensemble clocks can be located some distance from each other and become isolated for some time due to network failures. The frequency-offset error in $R_i$ is equivalent to the fractional frequency $y_i$,

$$y_i = \frac{\nu_i - \nu_I}{\nu_I}$$

measured between the $i$th timescale and the standard timescale $I$. Temporarily dropping the subscript $i$ for clarity, consider a sequence of $N$ independent frequency-offset samples $y(j)$ ($j = 1, 2, \ldots, N$) where the interval between samples is uniform and equal to $T$. Let $\tau$ be the nominal interval over which these samples are averaged. The Allan variance $\sigma_y^2(N, T, \tau)$ [ALL74a] is defined as

$$<\sigma_y^2(N, T, \tau) > = < \frac{1}{N - 1} \left[ \sum_{j=1}^{N} y(j)^2 - \frac{1}{N} \left( \sum_{j=1}^{N} y(j) \right)^2 \right] > ,$$

A particularly useful formulation is $N = 2$ and $T = \tau$:

$$<\sigma_y^2(N = 2, T = \tau, \tau)> \equiv \sigma_y^2(\tau) = <\frac{[y(j + 1) - y(j)]^2}{2}> ,$$

### 3.1. Development of a Composite Timescale

Consider the time offsets of a number of real clocks connected by real networks. A display of the offsets of all clocks relative to the standard timescale will appear as a system of bell-shaped curves slowly precessing relative to each other, but with some further away from nominal zero than others. The bells will normally be scattered over the offset space, more or less close to each other, with some overlapping and some not. The problem is to estimate the true offset relative to the standard timescale from a system of offsets collected routinely between the clocks.

A composite timescale can be determined from a sequence of offsets measured between the $n$ clocks of an ensemble at nominal intervals $\tau$. Let $R_i(t_0)$ be the frequency and $T_i(t)$ the time of the $i$th clock at epoch $t_0$ relative to the standard timescale. Let $T_{ij}(t)$ be the time difference or offset between clock $i$ and clock $j$, $T_i(t) - T_j(t)$, and "^" designate the associated estimates. Then, an estimator for $T_i$ computed at epoch $t_0$ for epoch $t_0 + \tau$ is

$$\hat{T}_i(t_0 + \tau) = \hat{R}_i(t_0)\tau + T_i(t_0) \ ,$$

neglecting second-order terms. Consider a set of $n$ independent time-offset measurements made between the clocks at epoch $t_0 + \tau$ and let the offset between clock $i$ and clock $j$ at that epoch be $T_{ij}(t_0 + \tau)$, defined as

$$T_{ij}(t_0 + \tau) \equiv T_i(t_0 + \tau) - T_j(t_0 + \tau) \ .$$

Note that $T_{ij} = -T_{ji}$ and $T_{ii} = 0$. Let $w_i(\tau)$ be a previously determined weight factor associated with the $i$th clock for the nominal interval $\tau$. The basis for new estimates at epoch $t_0 + \tau$ is

$$T_j(t_0 + \tau) = \sum_{i=1}^{n} w_i(\tau)[\hat{T}_i(t_0 + \tau) + T_{ji}(t_0 + \tau)].$$

That is, the apparent time indicated by the $j$th clock is a weighted average of the estimated time of each clock at epoch $t_0 + \tau$ plus the time offset measured between the $j$th clock and that clock at epoch $t_0 + \tau$.

An intuitive grasp of the behavior of this algorithm can be gained with the aid of a few examples. For instance, if $w_i(\tau)$ is unity for the $i$th clock and zero for all others, the apparent time for each of the other clocks is simply the estimated time $\hat{T}_i(t_0 + \tau)$. If $w_i(\tau)$ is zero for the $i$th clock, that clock can never affect any other clock and its apparent time is determined entirely from the other clocks. If $w_i(\tau) = 1/n$ for all $i$, the apparent time of the $i$th clock is equal to the average of the time estimates computed at $t_0$ plus the average of the time offsets measured to all other clocks. Finally, in a system with two clocks and $w_i(\tau) = 1/2$ for each, and if the estimated time at epoch $t_0 + \tau$ is fast by 1 s for one clock and slow by 1 s for the other, the apparent time for both clocks will coincide with the standard timescale.

In order to establish a basis for the next interval $\tau$, it is necessary to update the frequency estimate $\hat{R}_i(t_0 + \tau)$ and weight factor $w_i(\tau)$. The average frequency assumed for the $i$th clock during the previous interval $\tau$ is simply the difference between the times at the beginning and end of the interval divided by $\tau$. A good estimator for $R_i(t_0 + \tau)$ has been found to be the exponential average of these differences, which is given by

$$h(i+1) = h(i) + \frac{K_t\,\tau' v_s(i) - h(i)}{K_h}\,.$$

The factor $\tau'$ in the above has the effect of adjusting the bandwidth of the PLL as a function of compliance. When the compliance has been low over some relatively long period, $\tau'$ is increased and the bandwidth is decreased. In this mode small timing fluctuations due to jitter in the network are suppressed and the PLL attains the most accurate frequency estimate. On the other hand, if the compliance becomes high due to greatly increased jitter or a systematic frequency offset, $\tau'$ is decreased and the bandwidth is increased. In this mode the PLL is most adaptive to transients which can occur due to reboot of the system or a major timing error. In order to maintain optimum stability, the poll interval $\rho$ is varied directly with $\tau$.

A model suitable for simulation and parameter refinement can be constructed from the above recurrence relations. It is convenient to set the temporary variable $a = g(i+1)$. At each adjustment interval $\sigma$ the quantity $\frac{a}{K_g} + \frac{f(i+1)}{K_f}$ is added to the local-clock phase and the quantity $\frac{a}{K_g}$ is subtracted from $a$. For convenience, let $n$ be the greatest integer in $\frac{\mu(i)}{\sigma}$; that is, the number of adjustments that occur in the $i$th interval. Thus, at the end of the $i$th interval just before the $i$+1th update, the VCO control voltage is:

$$v_c(i+1) = v_c(i) + [1 - (1 - \frac{1}{K_g})^n]\; g(i+1) + \frac{n}{K_f} f(i+1)\,.$$

Detailed simulation of the NTP PLL with the values specified in Tables 9, 10 and 11 and the clock filter described in the NTP specification results in the following characteristics: For a 100-ms phase change the loop reaches zero error in 39 minutes, overshoots 7 ms at 54 minutes and settles to less than 1 ms in about six hours. For a 50-ppm frequency change the loop reaches 1 ppm in about 16 hours and 0.1 ppm in about 26 hours. When the magnitude of correction exceeds a few milliseconds or a few ppm for more than a few updates, the compliance begins to increase, which causes the loop time constant and update interval to decrease. When the magnitude of correction falls below about 0.1 ppm for a few hours, the compliance begins to decrease, which causes the loop time constant and update interval to increase. The effect is to provide a broad capture range exceeding 4 s per day, yet the capability to resolve oscillator skew well below 1 ms per day. These characteristics are appropriate for typical crystal-controlled oscillators with or without temperature compensation or oven control.

## 3. Clock-Combining Algorithms

A common problem in synchronization subnets is systematic time-offset errors resulting from asymmetric transmission paths, where the networks or transmission media in one direction are substantially different from the other. The errors can range from microseconds on high-speed ring networks to large fractions of a second on satellite/landline paths. It has been found experimentally that these errors can be considerably reduced by combining the apparent offsets of a number of time servers to produce a more accurate working offset. Following is a description of the combining method used in the NTP implementation for the Fuzzball [MIL88b]. The method is similar to that used by national standards laboratories to determine a synthetic laboratory timescale from an ensemble of cesium clocks [ALL74b].

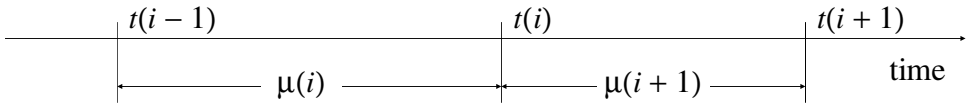


Figure 6. Timing Intervals

| Variable | Value | Description |
|---|---|---|
| μ | | update interval |
| ρ | | poll interval |
| $f$ | | frequency error |
| $g$ | | phase error |

Table 3. Notation Used in PLL Analysis

1, 2 and 3. Note the use of powers of two, which facilitates implementation using arithmetic shifts and avoids the requirement for a multiply/divide capability.

A capsule overview of the design may be helpful in understanding how it operates. The logical clock is continuously adjusted in small increments at fixed intervals of σ. The increments are determined while updating the variables shown in Tables 9 and 11, which are computed from received NTP messages as described in the NTP specification. Updates computed from these messages occur at discrete times as each is received. The intervals μ between updates are variable and can range up to about 17 minutes. As part of update processing the compliance $h$ is computed and used to adjust the PLL time constant τ. Finally, the update interval ρ for transmitted NTP messages is determined as a fixed multiple of τ.

Updates are numbered from zero, with those in the neighborhood of the $i$th update shown in Figure 5. All variables are initialized at $i = 0$ to zero, except the time constant $\tau(0) = \tau$, poll interval $\mu(0) = \tau$ (from Table 10) and compliance $h(0) = K_s$. After an interval $\mu(i)$ ($i > 0$) from the previous update the $i$th update arrives at time $t(i)$ including the time offset $v_s(i)$. Then, after an interval $\mu(i + 1)$ the $i$+1th update arrives at time $t(i + 1)$ including the time offset $v_s(i + 1)$. When the update $v_s(i)$ is received, the frequency error $f(i + 1)$ and phase error $g(i + 1)$ are computed:

$$f(i + 1) = f(i) + \frac{\mu(i) v_s(i)}{\tau(i)^2}, \quad g(i + 1) = \frac{v_s(i)}{\tau(i)}.$$

Note that these computations depend on the value of the time constant $\tau(i)$ and poll interval $\mu(i)$ previously computed from the $i$–1th update. Then, the time constant for the next interval is computed from the current value of the compliance $h(i)$

$$\tau(i + 1) = \max[K_s - |h(i)|, 1] .$$

Next, using the new value of τ, called τ′ to avoid confusion, the poll interval is computed

$$\rho(i + 1) = K_u \tau' .$$

Finally, the compliance $h(i + 1)$ is recomputed for use in the $i$+1th update:

$\sigma = \frac{10^{-3}}{2\ x\ 10^{-4}} = 5$ sec. For the NTP reference model $\sigma = 4$ sec in order to allow for known features of the Unix operating-system kernel. However, in order to support future anticipated improvements in accuracy possible with faster workstations, it may be useful to decrease $\sigma$ to as little as one-tenth the present value.

Note that if $\sigma$ is changed, it is necessary to adjust the parameters $K_f$ and $K_g$ in order to retain the same loop bandwidth; in particular, the same $\omega_c$ and $\omega_z$. Since $\alpha$ varies as the reciprocal of $\sigma$, if $\sigma$ is changed to something other than $2^2$, as in Table 10, it is necessary to divide both $K_f$ and $K_g$ by $\frac{\sigma}{4}$ to obtain the new values.

### 2.5. Adjusting PLL Bandwidth ($\tau$)

A key feature of the type-II PLL design is its capability to compensate for the intrinsic frequency errors of the local oscillator. This requires a initial period of adaptation in order to refine the frequency estimate. The $\tau$ parameter determines the PLL time constant and thus the loop bandwidth, which is approximately equal to $\frac{\omega_c}{\tau}$. When operated with a relatively large bandwidth (small $\tau$), as in the analysis above, the PLL adapts quickly to changes in the input reference signal, but has poor long term stability. Thus, it is possible to accumulate substantial errors if the system is deprived of the reference signal for an extended period. When operated with a relatively small bandwidth (large $\tau$), the PLL adapts slowly to changes in the input reference signal, and may even fail to lock onto it. Assuming the frequency estimate has stabilized, it is possible for the PLL to coast for an extended period without external corrections and without accumulating significant error.

In order to achieve the best performance without requiring individual tailoring of the loop bandwidth, it is necessary to compute each value of $\tau$ based on the measured values of offset, delay and dispersion, as produced by the NTP protocol itself. The traditional way of doing this in precision timekeeping systems based on cesium clocks, is to relate $\tau$ to the Allan variance, which is defined as the mean of the first-order differences of sequential samples measured during a specified interval $\tau$,

$$\sigma_y^2(\tau) = \frac{1}{2(N-1)} \sum_{i=1}^{N-1} [y(i+1) - y(i)]^2 ,$$

where $y$ is the fractional frequency measured with respect to the local timescale and $N$ is the number of samples.

In the NTP local-clock model the Allan variance (called the compliance, $h$ in Table 11) is approximated on a continuous basis by exponentially averaging the first-order differences of the offset samples using an empirically determined averaging constant. Using somewhat ad-hoc mapping functions determined from simulation and experience, the compliance is manipulated to produce the loop time constant and update interval.

### 2.6. The NTP Clock Model

The PLL behavior can also be described by a set of recurrence equations, which depend upon several variables and constants. The variables and parameters used in these equations are shown in Tables

With the parameter values given in Table 10, the Bode plot of the open-loop transfer function $G(s)$ consists of a –12 dB/octave line which intersects the 0-dB baseline at $\omega_c = 2^{-12}$ rad/s, together with a +6 dB/octave line at the corner frequency $\omega_z = 2^{-14}$ rad/s. The damping factor $\zeta = \frac{\omega_c}{2\omega_z} = 2$ suggests the PLL will be stable and have a large phase margin together with a low overshoot. However, if the clock-filter delay $T$ is not small compared to the loop delay, which is approximately equal to $\frac{1}{\omega_c}$, the above analysis becomes unreliable and the loop can become unstable. With the values determined as above, $T$ is ordinarily small enough to be neglected.

Assuming the output is taken at $v_s$, the closed-loop transfer function $H(s)$ is

$$H(s) \equiv \frac{v_s(s)}{\theta_r(s)} = \frac{F_d(s)e^{-Ts}}{1 + G(s)} .$$

If only the relative response is needed and the clock-filter delay can be neglected, $H(s)$ can be written

$$H(s) = \frac{1}{1 + G(s)} = \frac{s^2}{s^2 + \frac{\omega_c^2}{\omega_z \tau}s + \frac{\omega_c^2}{\tau^2}} .$$

For some input function $I(s)$ the output function $I(s)H(s)$ can be inverted to find the time response. Using a unit-step input $I(s) = \frac{1}{s}$ and the values determined as above, This yields a PLL risetime of about 52 minutes, a maximum overshoot of about 4.8 percent in about 1.7 hours and a settling time to within one percent of the initial offset in about 8.7 hours.

### 2.3. Parameter Management

A very important feature of the NTP PLL design is the ability to adapt its behavior to match the prevailing stability of the local oscillator and transmission conditions in the network. This is done using the $\alpha$ and $\tau$ parameters shown in Table 10. Mechanisms for doing this are described in following sections.

### 2.4. Adjusting VCO Gain ($\alpha$)

The $\alpha$ parameter is determined by the maximum frequency tolerance of the local oscillator and the maximum jitter requirements of the timekeeping system. This parameter is usually an architecture constant and fixed during system operation. In the implementation model described below, the reciprocal of $\alpha$, called the adjustment interval $\sigma$, determines the time between corrections of the local clock, and thus the value of $\alpha$. The value of $\sigma$ can be determined by the following procedure.

The maximum frequency tolerance for board-mounted, uncompensated quartz-crystal oscillators is probably in the range of $10^{-4}$ (100 ppm). Many if not most Internet timekeeping systems can tolerate jitter to at least the order of the intrinsic local-clock resolution, called *precision* in the NTP specification, which is commonly in the range from one to 20 ms. Assuming $10^{-3}$ s peak-to-peak as the most demanding case, the interval between clock corrections must be no more than

A good PLL design has a capture range large enough to handle the maximum open-loop VCO frequency error. Ultimately, the PLL capture range is limited by the maximum sampling rate

The ultimate lower bound on stability is the intrinsic jitter and wander of the VCO itself.

In order to achieve the smallest timing error, together with the highest stability, it is necessary to use a large $\tau$; but, as $\tau$ is increased the loop gain $K_v$ must be proportionally reduced to maintain a constant damping factor $\zeta$, as described previously. However, as $K_v$ is reduced, the tracking range is reduced as well. For a type-I PLL the tracking range must not fall below the maximum open-loop VCO frequency error or the PLL may break lock.

### 2.2.2. Type II Phase-Lock Loop

In order to reduce the residual phase error to zero, it is necessary to add another stage of integration, which amounts to the addition of another pole at zero frequency. This provides a means to estimate the frequency error and thus hold the phase offset to zero. However, a PLL with two poles at zero frequency is unstable, since the phase characteristic of the Bode plot approaches 180 degrees as the amplitude characteristic passes through unity gain. Unstability can be avoided through the addition of an additional zero, as shown in the following

$$G(s) = \frac{\omega_c^2}{\tau^2 s^2} \left(1 + \frac{\tau s}{\omega_z}\right),$$

where $\omega_c$ is the crossover frequency (also called loop gain), $\omega_z$ is the corner frequency (required for loop stability) and $\tau$ determines the PLL time constant and thus the bandwidth. While this is a first-order function and some improvement in phase noise might be gained from a higher-order function, in practice the improvement is lost due to the effects of the clock-filter delay, as described below.

Let $F(s)$ be the transfer function of the loop filter, which has yet to be determined. The open-loop transfer function $G(s)$ is the product of these four individual transfer functions:

$$G(s) = \frac{\omega_c^2}{\tau^2 s^2} \left(1 + \frac{\tau s}{\omega_z}\right) = F_d(s) F_s(s) F(s) F_o(s) = 1 e^{-Ts} F(s) \frac{\alpha}{s}.$$

For the moment, assume that the product $Ts$ is small, so that $e^{-Ts} \approx 1$. Making the following substitutions,

$$\omega_c^2 = \frac{\alpha}{K_f} \quad \text{and} \quad \omega_z = \frac{K_g}{K_f}$$

and rearranging yields

$$F(s) = \frac{1}{K_g \tau} + \frac{1}{K_f \tau^2 s},$$

which corresponds to a constant term plus an integrating term scaled by the PLL time constant $\tau$. This form is convenient for implementation as a sampled-data system, as described later.

$$H(s) \equiv \frac{v_s(s)}{\theta_r(s)} = \frac{F_d(s)e^{-Ts}}{1 + G(s)} .$$

If only the relative response is needed and the clock-filter delay can be neglected, $H(s)$ can be written

$$H(s) = \frac{1}{1 + G(s)} = \frac{1}{1 + F(s)\frac{\alpha}{s}} .$$

The PLL behavior is thus completely determined by its open-loop, Laplace transfer function $F(s)$ in the $s$ domain. In the simplest case $F(s)$ can be a constant and the loop will exhibit the classic behavior of the single time-constant (STC) system; however, it is usually desirable to incorporate a low-pass filter in order to reduce incidental noise. In this case the PLL becomes a type-I loop. Since such loops do not provide a way to reduce the residual phase error to zero, an additional integration stage can be introduced, in which case the loop becomes a type-II PLL. The mathematical analysis is different for each type of PLL, as shown in the following sections.

The small-signal acquisition and tracking behavior are determined by the loop gain and time constants established by the loop filter. The large-signal behavior is dominated by the loop gain, together with the numerical scaling of the various quantities. The *capture range* of a PLL is the maximum frequency range over which the PLL will achieve phase-lock, having previously been not in lock. The *tracking range* is the maximum fractional frequency range over which the PLL will remain locked, once phase-lock has been achieved. The *stability* is the largest frequency error.

### 2.2.1. Type I Phase-Lock Loop

A type-I PLL is characterized by a single low-pass filter with transfer function

$$F(s) = \frac{1}{1 + \frac{s}{\omega_L}},$$

where $\omega_L$ is the corner frequency. Substituting in Equation $ and rearranging yields

$$\theta\frac{(s)_o}{\theta}(s)_r = \frac{G(s)}{1 + G(s)} = left\frac{(s^2}{\omega_n^2} + 2\frac{\zeta}{\omega_n} + 1right)^{-1} ,$$

where $\omega_n$ and $\zeta$ are related to the loop gain $\alpha$ and loop-filter corner frequency $\omega_L$:

$$\omega_n^2 = K_v\omega_L \quad \text{and} \quad \zeta = \frac{1}{2}\left(\frac{\omega_L}{\alpha}\right)^{1/2} .$$

For a first-order filter function $\omega_L = \frac{1}{\tau}$, where $\tau$ is the time constant of integration. For a critically damped (Butterworth) system $\zeta$ should be equal to $2^{-1/2}$, which implies that the product $\frac{1}{2\alpha\tau}$ should be unity. This means that, as the time constant of integration is increased, the loop gain must be decreased proportionally in order to maintain the same dynamic characteristics.

| Parameter | Value | Description |
|---|---|---|
| $\alpha$ | $2^{-2}$ | VCO gain |
| $\sigma$ | $2^{2}$ | adjustment interval |
| $\tau$ | 1 | PLL time constant |
| T | $2^{3}$ | clock-filter delay |
| $K_f$ | $2^{22}$ | frequency weight |
| $K_g$ | $2^{8}$ | phase weight |

Table 2. PLL Parameters

| Variable | Description |
|---|---|
| $v_d$ | phase detector output |
| $v_s$ | clock filter output |
| $v_c$ | loop filter output |
| $\theta_r$ | reference phase |
| $\theta_o$ | VCO phase |
| $\omega_c$ | PLL crossover frequency |
| $\omega_z$ | PLL corner frequency |

Table 1. Notation Used in PLL Analysis

In Figure 4 the variable $\theta_r$ represents the phase of the reference signal and $\theta_o$ the phase of the voltage-controlled oscillator (VCO). The phase detector (PD) produces a voltage $v_d$ representing the phase difference $\theta_r - \theta_o$ . The clock filter, if used, functions as a tapped delay line, with the output $v_s$ taken at the tap selected by the clock-filter algorithm. The loop filter, represented by the equations given below, produces a VCO correction voltage $v_c$, which controls the oscillator frequency and thus the phase $\theta_o$.

The open-loop transfer function $G(s)$ is constructed by breaking the loop at point $a$ on Figure 4 and computing the ratio of the output phase $\theta_o(s)$ to the reference phase $\theta_r(s)$. This function is the product of the individual transfer functions for the phase detector, clock filter, loop filter and VCO. With appropriate scaling the phase detector delivers a voltage $v_d(t) = \theta_r(t)$ V/rad, so its transfer function is simply $F_d(s) = 1$. The VCO delivers a frequency change $\Delta\omega = \frac{d\,\theta_o(t)}{dt} = \alpha v_c(t)$, where $\alpha$ is the VCO gain in rad/V-s and $\theta_o(t) = \alpha \int v_c(t)\,dt$. Its transfer function is the Laplace transform of the integral, $F_o(s) = \frac{\alpha}{s}$. The clock filter contributes a stochastic delay due to the clock-filter algorithm; but, for present purposes, this delay will be assumed a constant $T$, so its transfer function is the Laplace transform of the delay, $F_s(s) = e^{-Ts}$. The open-loop transfer function is the product of these four functions

$$G(s) = F_d(s)F_s(s)F(s)F_o(s) = 1e^{-Ts}\,F(s)\,\frac{\alpha}{s}\,.$$

Assuming the output is taken at $v_s$, the closed-loop transfer function $H(s)$ is

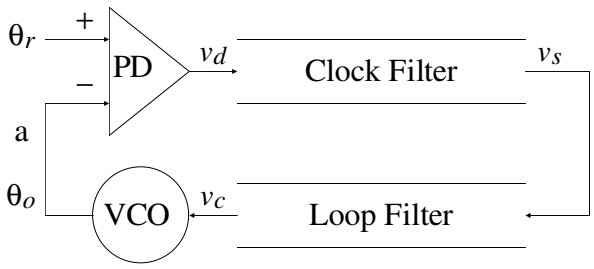


Figure 5. NTP Phase-Lock Loop (PLL) Model

The clock model requires the capability to slew the clock frequency over the range ±100 ppm with an intrinsic oscillator frequency error as great as ±100 ppm. Figure 3 shows the timing relationships at the extremes of the requirements envelope. Starting from an assumed offset of nominal zero and an assumed error of +100 ppm at time 0 s, the line AC shows how the uncorrected offset grows with time. Let σ represent the adjustment interval and *a* the interval AB, in seconds, and let *r* be the slew, or rate at which corrections are introduced, in ppm. For an accuracy specification of 100 μs, then

$$\sigma \le \frac{100\ \mu s}{100\ ppm} + \frac{100\ \mu s}{(r - 100)\ ppm} = \frac{r}{r - 100}.$$

The line AE represents the extreme case where the clock is to be steered −100 ppm. Since the slew must be complete at the end of the adjustment interval,

$$a \le \frac{(r - 200)\ \sigma}{r}.$$

These relationships are satisfied only if $r > 200$ ppm and $\sigma < 2$ s. Using $r = 300$ ppm for convenience, $\sigma = 1.5$ s and $a \le 0.5$ s. For the Unix clock model with *tick* = 10 ms, this results in the value of *tickadj* = 3 μs.

One of the assumptions made in the Unix clock model is that the period of adjustment computed in the *adjtime* call must be completed before the next call is made. If not, this results in an error message to the system log. However, in order to correct for the intrinsic frequency offset of the clock oscillator, the NTP clock model requires *adjtime* to be called at regular adjustment intervals of σ s. Using the algorithms described here and the architecture constants in the NTP specification, these adjustments will always complete.

### 2.2. Mathematical Model of the NTP Logical Clock

The disciplined computer clock can be represented by the feedback-control model shown in Figure 4. The model consists of a phase-lock loop (PLL), which continuously adjusts the clock oscillator to compensate for its intrinsic jitter, wander and drift. A mathematical analysis of this model developed along the lines of [SMI86] is presented in following sections, along with a design example useful for implementation guidance in operating-systems environments such as Unix and Fuzzball. Table 1 summarizes the quantities ordinarily treated as variables in the model. By convention, *v* is used for internal loop variables, θ for phase, ω for frequency and τ for time. Table 2 summarizes those quantities ordinarily fixed as constants in the model. Note that these are all expressed as a power of two in order to simplify the implementation.

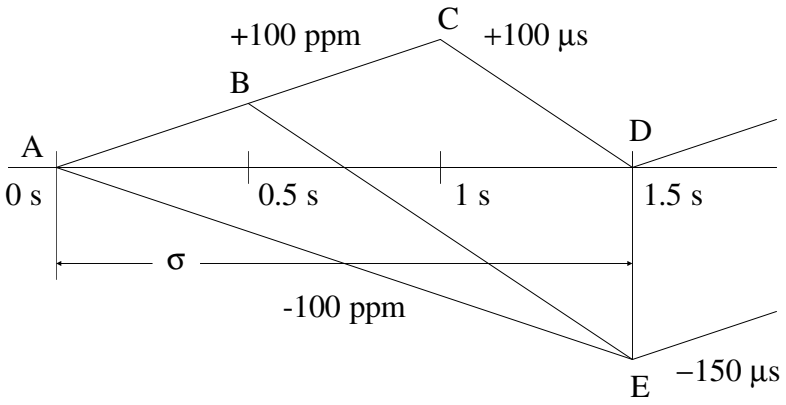


Figure 4. Clock Adjustment Process

to understand the behavior of the Unix clock as controlled by the Fuzzball clock model described above, it is helpful to explore the operations of *adjtime* in more detail.

The Unix clock model assumes an interrupt produced by an onboard frequency source, such as the clock counter and prescaler described previously, to deliver a pulse train in the 100-Hz range. In priniciple, the power grid frequency can be used, although it is much less stable than a crystal oscillator. Each interrupt causes an increment called *tick* to be added to the clock counter. The value of the increment is chosen so that the clock counter, plus an initial offset established by the *settimeofday* call, is equal to the time of day in microseconds.

The Unix clock can actually run at three different rates, one corresponding to *tick*, which is related to the intrinsic frequency of the particular oscillator used as the clock source, one to *tick* + *tickadj* and the third to *tick* − *tickadj*. Normally the rate corresponding to *tick* is used; but, if *adjtime* is called, the argument δ given is used to calculate an interval $\Delta t = \delta \frac{tick}{tickadj}$ during which one or the other of the two rates are used, depending on the sign of δ. The effect is to slew the clock to a new value at a small, constant rate, rather than incorporate the adjustment all at once, which could cause the clock to be set backward. With common values of *tick* = 10 ms and *tickadj* = 5 µs, the maximum frequency adjustment range is $\pm\frac{tickadj}{tick} = \pm\frac{5 \text{ x } 10^{-6}}{10^{-2}}$ or ±500 ppm. Even larger ranges may be required in the case of some workstations (e.g., SPARCstations) with extremely poor component tolerances.

When precisions not less than about 1 ms are required, the Fuzzball clock model can be adapted to the Unix model by software simulation, as described in Section 5 of the NTP specification, and calling *adjtime* at each adjustment interval. When precisions substantially better than this are required, the hardware microsecond clock provided in some workstations can be used together with certain refinements of the Fuzzball and Unix clock models. The particular design described below is appropriate for a maximum oscillator frequency tolerance of 100 ppm (.01%), which can be obtained using a relatively inexpensive quartz crystal oscillator, but is readily scalable for other assumed tolerances.

and hardware counter; however, the oscillator frequency remains constant and the hardware counter produces only a fraction of the total number of bits required by the clock counter. A typical design uses a 64-bit software clock counter and a 16-bit hardware counter which counts the prescaler output. A hardware-counter overflow causes the processor to increment the software counter at the bit corresponding to the frequency $2^N f_p$, where $N$ is the number of bits of the hardware counter and $f_p$ is the counted frequency at the prescaler output. The processor reads the clock counter by first generating a read pulse, which latches the hardware counter, and then adding its contents, suitably aligned, to the software counter.

The Fuzzball clock can be corrected in phase by adding a (signed) adjustment to the software clock counter. In practice, this is done only when the local time is substantially different from the time indicated by the clock and may violate the monotonicity requirement. Vernier phase adjustments determined in normal system operation must be limited to no more than the period of the counted frequency, which is 1 kHz for LSI-11 Fuzzballs. In the Fuzzball model these adjustments are performed at intervals of 4 s, called the *adjustment interval*, which provides a maximum frequency adjustment range of 250 ppm. The adjustment opportunities are created using the interval-timer facility, which is a feature of most operating systems and independent of the time-of-day clock. However, if the counted frequency is increased from 1 kHz to 1 MHz for enhanced precision, the adjustment frequency must be increased to 250 Hz, which substantially increases processor overhead. A modified design suitable for high precision clocks is presented in the next section.

In some applications involving the Fuzzball model, an external pulse-per-second (pps) signal is available from a reference source such as a cesium clock or GPS receiver. Such a signal generally provides much higher accuracy than the serial character string produced by a radio timecode receiver, typically in the low nanoseconds. In the Fuzzball model this signal is processed by an interface which produces a hardware interrupt coincident with the arrival of the pps pulse. The processor then reads the clock counter and computes the residual modulo 1 s of the clock counter. This represents the local-clock error relative to the pps signal.

Assuming the seconds numbering of the clock counter has been determined by a reliable source, such as a timecode receiver, the offset within the second is determined by the residual computed above. In the NTP local-clock model the timecode receiver or NTP establishes the time to within ±128 ms, called the aperture, which guarantees the seconds numbering to within the second. Then, the pps residual can be used directly to correct the oscillator, since the offset must be less than the aperture for a correctly operating timecode receiver and pps signal.

The above technique has an inherent error equal to the latency of the interrupt system, which in modern RISC processors is in the low tens of microseconds. It is possible to improve accuracy by latching the hardware time-of-day counter directly by the pps pulse and then reading the counter in the same way as usual. This requires additional circuitry to prioritize the pps signal relative to the pulse generated by the program to latch the counter.

### 2.1.2. The Unix Clock Model

The Unix 4.3bsd clock model is based on two system calls, *settimeofday* and *adjtime*, together with two kernel variables *tick* and *tickadj*. The *settimeofday* call unceremoniously resets the kernel clock to the value given, while the *adjtime* call slews the kernel clock to a new value numerically equal to the sum of the present time of day and the (signed) argument given in the *adjtime* call. In order

correction to the clock-offset variable, while the clock frequency is adjusted by loading a correction to the DAC latch. In principle, this clock model can be adapted to any precision by changing the number of bits of the prescaler or clock counter or changing the VCO frequency. However, it does not seem useful to reduce precision much below the minimum interrupt latency, which is in the low microseconds for a modern RISC processor.

If it is not possible to vary the oscillator frequency, which might be the case if the oscillator is an external frequency standard, a design such as shown in Figure 10b may be used. It includes a fixed-frequency oscillator and prescaler which includes a dual-modulus *swallow counter* that can be operated in either divide-by-10 or divide-by-11 modes as controlled by a pulse produced by a programmable divider (PD). The PD is loaded with a value representing the frequency offset. Each time the divider overflows a pulse is produced which switches the swallow counter from the divide-by-10 mode to the divide-by-11 mode and then back again, which in effect "swallows" or deletes a single pulse of the prescaler pulse train.

The pulse train produced by the prescaler is controlled precisely over a small range by the contents of the PD. If programmed to emit pulses at a low rate, relatively few pulses are swallowed per second and the frequency counted is near the upper limit of its range; while, if programmed to emit pulses at a high rate, relatively many pulses are swallowed and the frequency counted is near the lower limit. Assuming some degree of freedom in the choice of oscillator frequency and prescaler ratios, this design can compensate for a wide range of oscillator frequency tolerances.

In all of the above designs it is necessary to limit the amount of adjustment incorporated in any step to insure that the system clock indications are always monotonically increasing. With the software clock model this is assured as long as the increment is never negative. When the magnitude of a phase adjustment exceeds the tick interval (as corrected for the frequency adjustment), it is necessary to spread the adjustments over mulitple tick intervals. This strategy amounts to a deliberate frequency offset sustained for an interval equal to the total number of ticks required and, in fact, is a feature of the Unix clock model discussed below.

In the hardware clock models the same considerations apply; however, in these designs the tick interval amounts to a single pulse at the prescaler output, which may be in the order of 1 ms. In order to avoid decreasing the indicated time when a negative phase correction occurs, it is necessary to avoid modifying the clock-offset variable in processor memory and to confine all adjustments to the VCO or prescaler. Thus, all phase adjustments must be performed by means of programmed frequency adjustments in much the same way as with the software clock model described previously.

It is interesting to conjecture on the design of a processor assist that could provide all of the above functions in a compact, general-purpose hardware interface. The interface might consist of a multifunction timer chip such as the AMD 9513A, which includes five 16-bit counters, each with programmable load and hold registers, plus an onboard crystal oscillator, prescaler and control circuitry. A 48-bit hardware clock counter would utilize three of the 16-bit counters, while the fourth would be used as the swallow counter and the fifth as the programmable divider. With the addition of a programmable-array logic device and architecture-specific host interface, this compact design could provide all the functions necessary for a comprehensive timekeeping system.

### 2.1.1. The Fuzzball Clock Model

The Fuzzball clock model uses a combination of hardware and software to provide precision timing with a minimum of software and processor overhead. The model includes an oscillator, prescaler

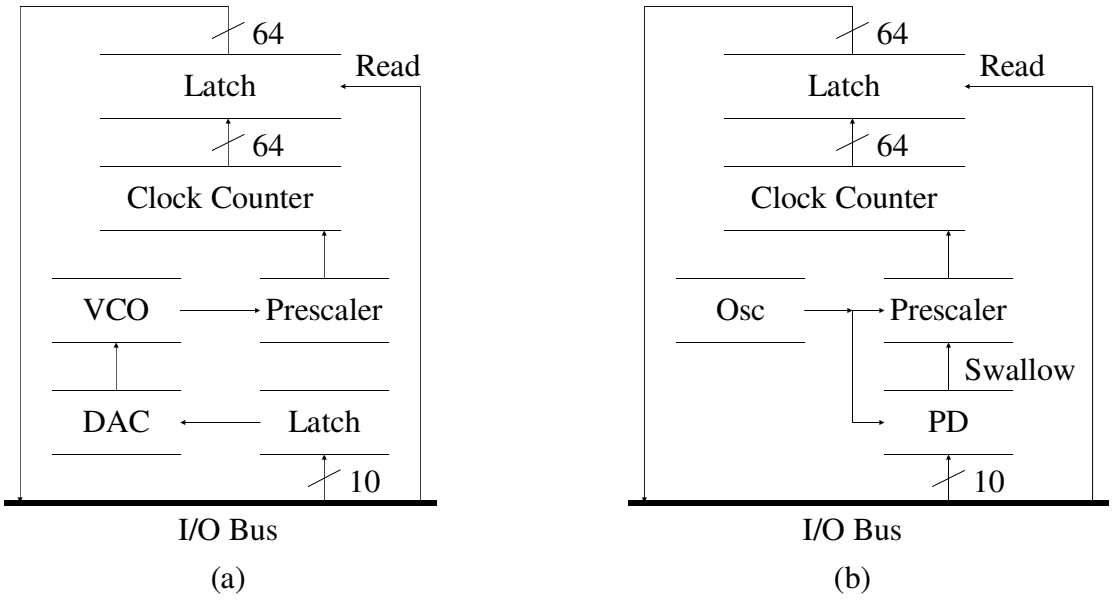


Figure 3. Hardware Clock Models

the oscillator frequency to a standard value, such as 1 MHz or 100 Hz, and a counter, implemented in hardware, software or some combination of the two, which can be read by the processor. For systems intended to be synchronized to an external source of standard time, there must be some means to correct the phase and frequency by occasional vernier adjustments produced by the timekeeping protocol. Special care is necessary in all timekeeping system designs to insure that the clock indications are always monotonically increasing; that is, system time never "runs backwards."

The simplest computer clock consists of a hardware latch which is set by overflow of a hardware counter or prescaler, and causes a processor interrupt or *tick*. The latch is reset when acknowledged by the processor, which then increments the value of a software clock counter. The phase of the clock is adjusted by adding periodic corrections to the counter as necessary. The frequency of the clock can be adjusted by changing the value of the increment itself, in order to make the clock run faster or slower. The precision of this simple clock model is limited to the tick interval, usually in the order of 10 ms; although in some systems the tick interval can be changed using a kernel variable.

This software clock model requires a processor interrupt on every tick, which can cause significant overhead if the tick interval is small, say in the order less 1 ms with the newer RISC processors. Thus, in order to achieve timekeeping precisions less than 1 ms, some kind of hardware assist is required. A straightforward design consists of a voltage-controlled oscillator (VCO), in which the frequency is controlled by a buffered, digital/analog converter (DAC). Under the assumption that the VCO tolerance is $10^{-4}$ or 100 parts-per-million (ppm) (a reasonable value for inexpensive crystals) and the precision required is 100 μs (a reasonable goal for a RISC processor), the DAC must include at least ten bits.

A design sketch of a computer clock constructed entirely of hardware logic components is shown in Figure 2a. The clock is read by first pulsing the read signal, which latches the current value of the clock counter, then adding the contents of the clock-counter latch and a 64-bit clock-offset variable, which is maintained in processor memory. The clock phase is adjusted by adding a

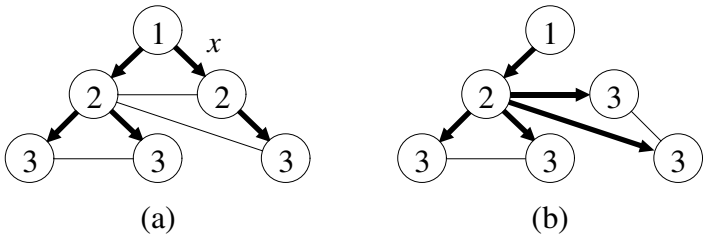


Figure 1. Subnet Synchronization

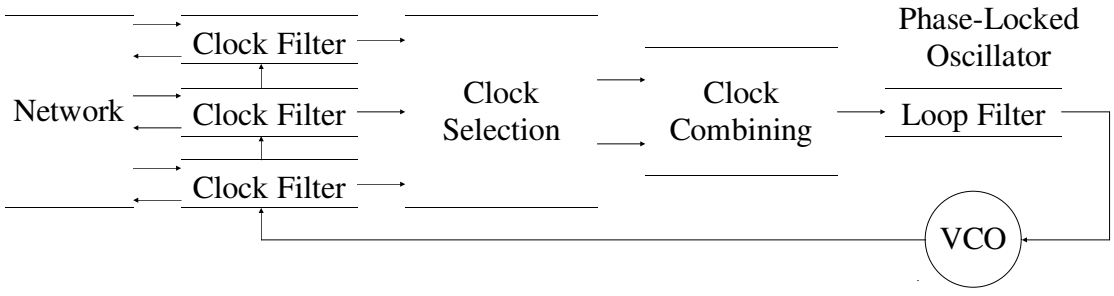


Figure 2. Network Time Protocol

If the network delays from A to B and from B to A are similar, the roundtrip delay δ and clock offset θ of *B* relative to *A* at time $T_i$ are:

$$\delta = a - b \quad \text{and} \quad \theta = \frac{a + b}{2}.$$

Each NTP message includes the latest three timestamps $T_{i-1}$, $T_{i-2}$ and $T_{i-3}$, while the fourth timestamp $T_i$ is determined upon arrival of the message. Thus, both the server and the peer can independently calculate delay and offset using a single bidirectional message stream. This is a symmetric, continuously sampled, time-transfer scheme similar to those used in some digital telephone networks [LIN80]. Among its advantages are that the transmission times and received message orders are unimportant and that reliable delivery is not required.

The peer-selection algorithm determines from among all peers a suitable subset of peers capable of providing the most accurate and trustworthy time using principles similar to those described in [VAS88]. In NTP this is done using a cascade of two subalgorithms, one a version of an algorithm proposed in [MAR85] and the other based on maximum likelihood principles to improve accuracy [MIL91].

The resulting offsets of this subset are first combined on a weighted-average basis using an algorithm similar to that described in [JON83] and then processed by a phase-lock loop (PLL). In the PLL the combined effects of the filtering, selection and combining operations are to produce a phase-correction term, which is processed by the loop filter to control the local clock, which functions as a voltage-controlled oscillator (VCO). The VCO furnishes the timing (phase) reference to produce the timestamps used in all timing calculations.

### 2.1. Computer Clock Models

A computer clock includes some kind of reference oscillator, which is stabilized by a quartz crystal or some other means, such as the power grid. Usually, the clock includes a prescaler, which divides

and algorithms designed to combine them to produce a composite timescale approximating the standard timescale.

## 2. Network Time Protocol

The Network Time Protocol (NTP) is used by Internet time servers and their peers to synchronize clocks, as well as automatically organize and maintain the time synchronization subnet itself. It is evolved from the Time Protocol [POS83] and the ICMP Timestamp Message [DOD81a], but is specifically designed for high accuracy, stability and reliability, even when used over typical Internet paths involving multiple gateways and unreliable networks. The following sections contain an overview of the architecture and algorithms used in NTP. A formal description and error analysis of the protocol is contained in [MIL90b]. A detailed description of the NTP architecture and protocols is contained in [MIL91], while a summary of operational experience and performance is contained in [MIL90a].

NTP and its implementations have evolved and proliferated in the Internet over the last decade, with NTP Version 2 now adopted as an Internet Standard (Recommended) [MIL89] and NTP Version 3 now adopted as a Proposed Standard [MIL90b]. NTP is built on the Internet Protocol (IP) [DOD81b] and User Datagram Protocol (UDP) [POS80], which provide a connectionless transport mechanism; however, it is readily adaptable to other protocol suites. The protocol can operate in several modes appropriate to different scenarios involving private workstations, public servers and various network configurations. A lightweight association-management capability, including dynamic reachability and variable poll-interval mechanisms, is used to manage state information and reduce resource requirements. Optional features include message authentication based on crypto-checksums and provisions for remote control and monitoring.

In NTP one or more primary servers synchronize directly to external reference sources such as radio clocks. Secondary time servers synchronize to the primary servers and others in the synchronization subnet. A typical subnet is shown in Figure a, in which the nodes represent subnet servers, with normal level numbers determined by the hop count from the root (stratum one), and the heavy lines the active synchronization paths and direction of timing information flow. The light lines represent backup synchronization paths where timing information is exchanged, but not necessarily used to synchronize the local clocks. Figure b shows the same subnet, but with the line marked $x$ out of service. The subnet has re-configured itself automatically to use backup paths, with the result that one of the servers has dropped from stratum 2 to stratum 3. In practice each NTP server synchronizes with several other servers in order to survive outages and Byzantine failures using methods similar to those described in [SHI87].

Figure  shows the overall organization of the NTP time-server model, which has much in common with the phase-lock methods summarized in [RAM90]. Timestamps exchanged between the server and possibly many other subnet peers are used to determine individual roundtrip delays and clock offsets, as well as provide reliable error bounds. As shown in the figure, the computed delays and offsets for each peer are processed by the data-filter algorithm to reduce incidental timing noise. As described in the [MIL90b], this algorithm selects from among the last several samples the one with minimum $\delta$ and presents the associated $\theta$ as the output.

Figure  shows how NTP timestamps are numbered and exchanged between peers $A$ and $B$. Let $T_i$, $T_{i-1}$, $T_{i-2}$, $T_{i-3}$ be the values of the four most recent timestamps as shown and let

$$a = T_{i-2} - T_{i-3} \quad \text{and} \quad b = T_{i-1} - T_i.$$

## 1. Computer Clock Modelling and Analysis

In the following discussion the terms *time*, *timescale*, *oscillator*, *clock*, *epoch*, *timestamp, calendar* and *date* are used in a technical sense. Strictly speaking, the *time* of an event is an abstraction which determines the ordering of events in some given frame of reference or *timescale*. An *oscillator* is a generator capable of precise frequency (relative to the given timescale) within a specified tolerance. A *clock* is an oscillator together with a counter which records the (fractional) number of cycles since being initialized with a given value at a given time. The value of the counter at any given time $t$ is called its *epoch* and recorded as the *timestamp T(t)* of that epoch. In general, epoches are not continuous and depend on the precision of the counter.

A *calendar* is a mapping from epoches in some timescale to the year-day *dates* used in everyday life. Since multiple calendars are in use today and sometimes disagree on the dating of the same epoches in the past, the metrology of past and present epoches is an art practiced by historians. However, the ultimate timescale for our world is based on cosmic oscillators, such as the Sun, Moon and other galactic orbiters. Since the frequencies of these oscillators are relatively unstable and not known exactly, the ultimate reference standard oscillator has been chosen by international agreement as a synthesis of many observations of an atomic transition of exquisite stability. The frequency of each heavenly and Earthbound oscillator defines a distinctive timescale, not necessarily always continuous, relative to that of the standard oscillator.

In this paper the *stability* of a clock is how well it can maintain a constant frequency, the *accuracy* is how well its time compares with national standards and the *precision* is to what degree time can be resolved in a particular timekeeping system. The *time offset* of two clocks is the time difference between them, while the *frequency offset* is the frequency difference between them. In this paper reference to simply "offset" means time offset, unless indicated otherwise. The *reliability* of a timekeeping system is the fraction of the time it can be kept connected to the network and operating correctly relative to stated accuracy and stability tolerances.

In order to synchronize clocks, there must be some way to directly or indirectly compare them in time and frequency. In network architectures such as DECnet and Internet local clocks are synchronized to designated *time servers*, which are timekeeping systems belonging to a *synchronization subnet*, in which each server measures the offsets between its local clock and the clocks of its neighbor servers or *peers* in the subnet. In this paper to *synchronize frequency* means to adjust the clocks in the subnet to run at the same frequency, to *synchronize time* means to set them to agree at a particular epoch with respect to Coordinated Universal Time (UTC), as provided by national standards, and to *synchronize clocks* means to synchronize them in both frequency and time.

The International Standard (SI) definition of *time interval* is in terms of the standard second: "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 cesium-133 atom." Let $u$ represent the standard unit of time interval so defined and $v = \frac{1}{u}$ be the standard unit of frequency. The *epoch*, denoted by $t$, is defined as the reading of a counter that runs at frequency $v$ and began counting at some agreed initial epoch $t_0$, which defines the *standard* or *absolute timescale*. For the purposes of the following analysis, the epoch of the standard timescale, as well as the *time* indicated by a clock will be considered continuous. In practice, time is determined relative to a clock constructed from an atomic oscillator and system of counter/dividers, which defines a timescale associated with that particular oscillator. Standard time and frequency are then determined from an ensemble of such timescales