Clock Models, Metrics, and Testing

Reid McGaughey
Hardware Engineer
Cisco

Presented at the ODVA
2014 Industry Conference & 16th Annual Meeting
March 11-13, 2014
Phoenix, Arizona, USA

Abstract:

An introduction to clock models, metrics for gauging a clock's performance, and the various test setups for collecting these metrics.

Keywords:

PTP, CIP Sync, Max|TE|, MTIE, TDEV

Body:

1.0 Introduction

This paper is an introduction to clock models, metrics, and testing for the ODVA community. This paper is not the first primer on this topic, and interested readers should see [1] for an excellent primer on timekeeping. This paper assumes the reader is familiar with Precision Time Protocol (PTP) and the various clock types: Grandmaster Clocks (GMC), Boundary Clocks (BC), Transparent Clocks, and Ordinary Clocks. For more information on PTP, please see the IEEE 1588-2008 specification [2]

2.0 Clock Models

This paper only considers the End to End delay mechanism, but these same models could be extended to the Peer to Peer delay mechanism as well. While many ODVA members work on processes that do not have closed form solutions, clocks and oscillators have been analyzed in great detail and do have closed form models [3]. Consequently, this paper will be using state space models for further analysis, since the state space model simplifies the modeling of multi-input multi-output systems. This model is not the only way to represent clocks and oscillators, and the implementer is free to use another model, if the model is more suitable. This section describes the plant models for the various clock types, but does not discuss controllers for those plants. This section only considers the plant for the local clock for boundary clocks and transparent clocks, rather than creating chainable models.

2.1 The System Model

The simplest implementation for a clock feeds a frequency reference from an oscillator into a counter, like Figure 1.

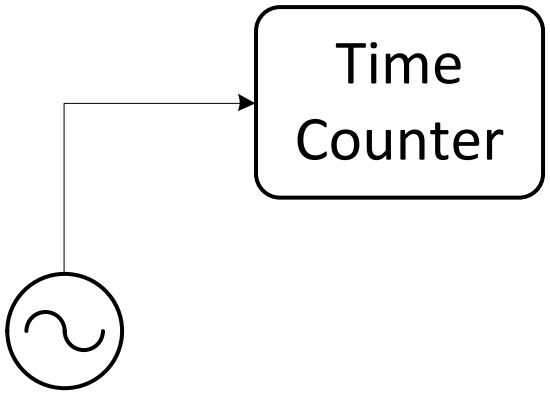

Figure 1

The time counter integrates the edges coming from the oscillator. Ignoring rollover, this structure creates a monotonic clock, where each bit of the counter represents a discrete chunk of time. For example, if the oscillator's frequency is nominally 125 MHz, then each bit of the counter would represents 8 ns of time. At this point, the clock's output is only determined by the differential equations that govern the oscillator and the time counter.

This clock structure follows a second order differential equation, similar to the differential equation for one dimensional position. The clock will have some initial time (position), some initial rate (velocity), and the rate of the clock may be drifting as well (acceleration). Combining all of the effects together yields equations 1 and 2.

$$Rate(t) = Rate_0 + Drift(t) * t \quad Eq.1$$

$$Time(t) = Time_0 + Rate(t) * t + 0.5 * Drift(t) * t^2 \quad Eq.2$$

Summarizing the behavior of equations 1 and 2:
-A constant drift causes the rate to change linearly.
-A constant rate causes the time to change linearly.
-A constant drift term causes the time to change quadratically.

Transforming the equations above into a discrete time state model, we get equations 3 and 4. Please note in these equations (and everywhere else in this paper), the symbol τ represents the sample interval (e.g. 1 second).

$$x[k] = \begin{bmatrix} Time \\ Rate \\ Drift \end{bmatrix} \quad Eq.3$$

$$x[k+1] = \begin{bmatrix} 1 & \tau & 0.5\tau^2 \\ 0 & 1 & \tau \\ 0 & 0 & 1 \end{bmatrix} x[k] \quad Eq.4$$

2.2 The Actuators

Now that we have the state equations for a clock, we can add actuators to manipulate the clock.

2.2.1 Rate Adjust

Many PTP implementations provide a mechanism to adjust the rate of the clock. These mechanisms may use a Numerically Controlled Oscillator (NCO), which adds or removes edges from the oscillator input into the Time Counter. Other implementations, adjust the value that is added to the Time Counter for every edge from the oscillator. In either case, the both actuators adjust the rate of the clock. Figure 2 shows the updated clock model with the rate adjust actuator. For this model, the NCO will be modeled as an integrator.

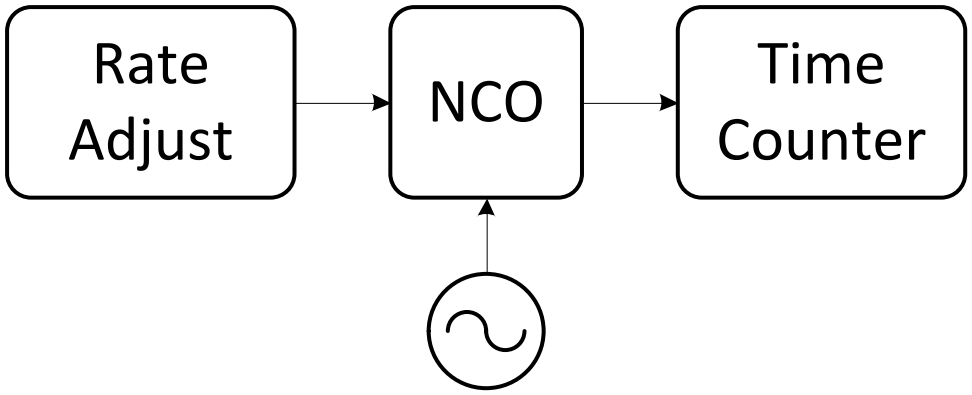

Figure 2

How does adjusting the rate affect the system? Adjusting the rate will instantly set the rate to a new value. However, the rate adjustment does come with a side effect; the rate adjustment may cause the time to change as well, since a rate change causes time to grow with a different linear slope. See equations 5 and 6 for the representation for this actuator.

$$u[k] = [Rate\ Adjust] \quad Eq.5$$

$$x[k+1] = \begin{bmatrix} 1 & \tau & 0.5\tau^2 \\ 0 & 1 & \tau \\ 0 & 0 & 1 \end{bmatrix} x[k] + \begin{bmatrix} \tau \\ 1 \\ 0 \end{bmatrix} u[k] \quad Eq.6$$

This side effect can actually be a benefit, since a rate adjustment can actually monotonically adjust the time state. However, this control mechanism can only slowly make large changes to the time state with reasonable rate adjustments. As some of you may have noticed, this system is not completely controllable, since there is no way to adjust the drift. Fortunately, the drift term is bounded for real oscillators.

2.2.2 Time and Rate Adjust

Since a clock is mostly digital logic, we can enhance our clock with a second actuator that instantaneously updates time. This is the actuator setup recommended in the in Volume 1 of the CIP Networks Library [4]. The controller has the ability to adjust the time of the clock directly, and adjust the rate of the clock. Figure 3 shows the clock model updated with the time adjust actuator.

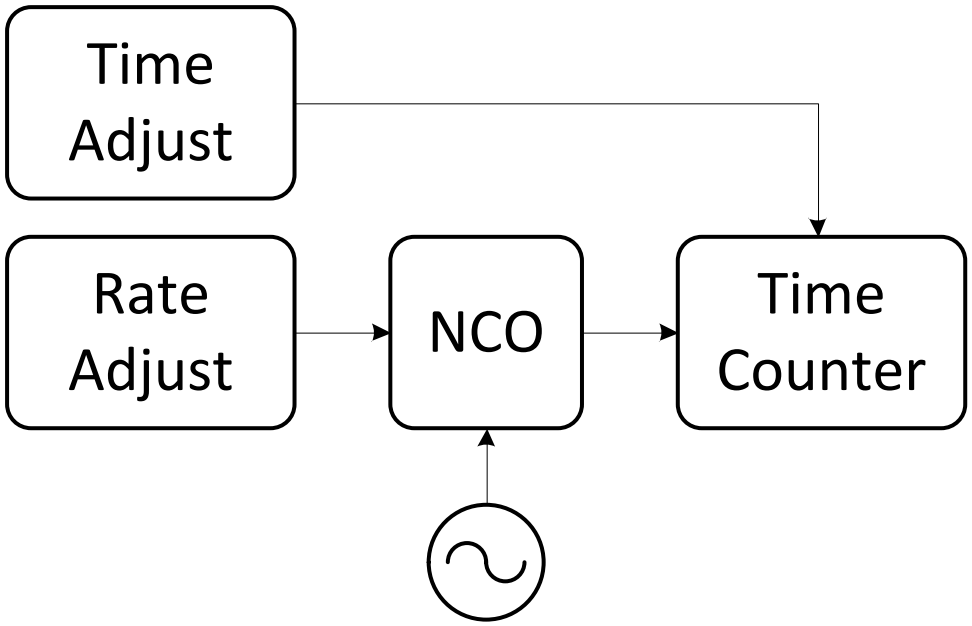

Figure 3

Please see equations 7 and 8 for the mathematical model of the new actuator setup.

$$u[k] = \begin{bmatrix} Time\ Adjust \\ Rate\ Adjust \end{bmatrix} \quad Eq.7$$

$$x[k+1] = \begin{bmatrix} 1 & \tau & 0.5\tau^2 \\ 0 & 1 & \tau \\ 0 & 0 & 1 \end{bmatrix} x[k] + \begin{bmatrix} 1 & \tau \\ 0 & 1 \\ 0 & 0 \end{bmatrix} u[k] \quad Eq.8$$

This actuator has the nice benefit that it can make large adjustments to the time state very quickly. However, the Time Adjust actuator must be used with care, since it may cause the clock to be non-monotonic. If your application does not need a monotonic clock, or can tolerate brief windows where the clock is not monotonic, then there is no concern with using the time adjust actuator. This implementation has the additional benefit that it can always fall back to a rate only adjust, so all implementations should implement both actuators for flexibility. Again, the system is not completely controllable, since there is no way to adjust the drift. Fortunately, the drift term is bounded for real oscillators.

2.3 References and Sensors

Now that we have an adjustable clock, we want to compare our local clock against a reference clock and adjust our local clock to match the reference clock. Precision Time Protocol (PTP), and CIP Sync, uses two-way time transfer (TWTT) to compare the local clock with the master clock. Please see appendix A for a more detailed discussion of TWTT.

The end to end delay mechanism (E2E) of PTP combines the references and the local sensors into two messages, the Sync Message and the DelReq message. The Sync message generates a reference timestamp from the master clock (T1) and measures the local clock with a timestamp (T2) after some delay. Likewise, the DelReq message measures the local clock with a timestamp (T3) and provides a reference timestamp from the master (T4) after some delay. Equation 9 shows the reference timestamps grouped into the reference vector, and equation 10 shows the local timestamp measurements grouped into the sensor vector.

$$r[k] = \begin{bmatrix} T1 \\ T4 \end{bmatrix} = \begin{bmatrix} Reference\ Time \\ Reference\ Time \end{bmatrix} \quad Eq.9$$

$$y[k] = \begin{bmatrix} T2 \\ T3 \end{bmatrix} = \begin{bmatrix} Local\ Time + Delay \\ Local\ Time - Delay \end{bmatrix} \quad Eq.10$$

As mentioned above, our sensors actually include a delay, however, our existing clock model does not contain have a 'Delay' state, so we must add a state to the model. Equation 11 shows the enhanced state vector for TWTT.

$$x[k] = \begin{bmatrix} Time \\ Freq \\ Drift \\ Delay \end{bmatrix} \quad Eq.11$$

Equation 12 shows the enhanced model with actuators. Like drift, the delay state is also uncontrollable, but the delay is bounded in real systems.

$$x[k+1] = \begin{bmatrix} 1 & \tau & 0.5\tau^2 & 0 \\ 0 & 1 & \tau & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x[k] + \begin{bmatrix} 1 & \tau \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} u[k] \quad Eq.12$$

2.3.1 Boundary Clock or Slave Clock

Boundary clocks and Ordinary clocks use both the Sync messages and the DelReq messages to recover time from the reference clock. Our output matrix maps the time and delay states to the sensors in equation 13.

$$y[k] = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & -1 \end{bmatrix} x[k] \quad Eq.13$$

Every model has its limitations, and it is worth noting that local timestamps do not have any significance without their corresponding references. The T1 timestamp must be the reference for the T2 timestamp, and the T4 timestamp must be the reference for T3. The controller must use these timestamps to calculate the time error and the delay, and then make appropriate adjustments to the clock.

This system is completely observable, but it is worth noting that the sensors do not directly measure the time, the rate, the drift or the delay; but these states may be estimated. Rather than using an observer, some implementations collect several samples, and then calculate the delay, the time error, the first difference of the time error, and the second difference of the time error on those samples, effectively measuring all of the states.

2.3.2 Transparent Clock

A transparent clock only has access to the data generated by the Sync messages, so the output matrix only has one row. Unfortunately, the TC system is not observable since the TC cannot distinguish between the delay and the time error, see equation 14. As an approximation, a TC can assume the delay is zero, and use One Way Time Transfer (OWTT) to roughly synchronize to the master clock. The TC could also ignore time altogether and only rate lock to the GMC by using the first difference of the OWTT Time Error.

$$y[k] = [1 \quad 0 \quad 0 \quad 1]x[k] \quad Eq.14$$

2.4 Disturbances

With a few exceptions, most of the clock is a deterministic digital system that does not experience any disturbances. This section will discuss the disturbances that do affect the clock, particular implementers should be aware of the environmental sensitivities of the oscillator and packet delay variation,

2.4.1 Oscillator's Environmental Conditions

The base oscillators for clocks are very sensitive to environmental conditions, especially thermal changes [5]. Unless the clock's application can tolerate a more expensive oscillator such as a Temperature Compensated Crystal Oscillator (TCXO) or an Oven Controlled Crystal Oscillator (OCXO), temperature sensitivity will be the limiting factor for the clock's precision. A linear temperature ramp can cause a constant drift, which as this paper discussed

earlier causes a quadratic time error. Other factors such as vibration, pressure, and aging affect the oscillators as well.

2.4.2 Oscillator Noise

In additional to the error caused by the environmental conditions, oscillators exhibit random noise. Oscillators typically exhibit a combination of random walk noise (Brownian noise, red noise or $1/f^2$ noise), flicker noise (1/f noise), and white noise [3]. Because of the random walk and flicker noise, clocks frequently exhibit slow long term deviations that cause rate disturbances and drift disturbances. The white noise mainly disturbs the time state.

2.4.3 Packet Delay Variation

After temperature variation, packet delay variation (PDV) will be the second limiting factor for a PTP implementation. Since the sensors measure both the delay and the time error, delay variation may look like clock movement. Unfortunately, PDV is non-stationary, non-Gaussian, zero-mean, and wide band. It is possible to filter PDV, but filtering PDV typically requires complex algorithms and stable oscillators. Fortunately, PDV is only an issue if non-PTP networking elements are in the hierarchy, so applications that cannot tolerate the additional time error should avoid using non-PTP networking elements.

2.4.4 Complete Disturbance Model

Equations 15 and 16 show the disturbances grouped into the disturbance vector w, and how they affect the system.

$$w[k] = \begin{bmatrix} Osc.Noise \\ Osc.Noise \\ Environment + Osc.Noise \\ PDV \end{bmatrix} \quad Eq.15$$

$$x[k+1] = \begin{bmatrix} 1 & \tau & 0.5\tau^2 & 0 \\ 0 & 1 & \tau & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x[k] + \begin{bmatrix} 1 & \tau \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} u[k] + w[k] \quad Eq.16$$

3.0 Clock Metrics

Now that we have discussed the clock models, we will take a look at the metrics traditionally used to characterize clocks. While many of these metrics are standard statistical tools, a few of these metrics are unique to clocks. The new metrics were created to help characterize the noise sources that affect clocks and oscillators. A few of these metrics may appear intimidating, but purpose built timing measurement systems will handle all of the mathematical heavy lifting, so we can consider the metrics from a qualitative perspective.

3.1 Time Error Sequences

Time Error (TE) is defined as the local clock's time minus the reference clock's time:

$$TE = Time_{local} - Time_{ref} \quad Eq.17$$

To analyze a clock's performance, we need to collect a sequence of N TE samples with a sample interval τ_0. For example, if the test setup uses compares the slave clock's 1 PPS output with the master clock's 1 PPS output, the sample interval τ_0 is 1 second. A sequence of N TE samples is referred to as $\{x(n\tau_0)\}$, where 'n' represents the index. Once we have a large enough sequence of TE samples, we can begin statistical analysis of the clock. Typically, a test setup needs to collect around 12,000 samples for a statistically significant result [6].

3.2 The Histogram

A classical statistical method for estimating the probability distribution, the histogram provides a good indication of the spread of the clock data when the clock is locked. The histogram will typically produce a Gaussian shape, but a Gaussian shape does not necessarily indicate the clock noise is white. The histogram will typically include information about the peak to peak error, the mean, and the standard deviation. The histogram does not provide any insight into the short term or long term stability of the clock, nor does the histogram provide any insight when the clock's reference has been removed. Consequently, additional metrics to study clocks were created [1] [3].

3.3 Maximum Absolute Time Error

The Maximum Absolute Time Error metric (Max|TE|) finds the worst case negative or positive time error from a sequence of time error samples.

$$max|TE| = \max_{1 \le n \le N} |\{x(n\tau_0)\}| \qquad Eq.18$$

The max|TE| metric is important for qualifying the single worst case time error of a clock, but the max|TE| metric does not provide a lot of additional insight into the stability of the clock.

3.4 Maximum Time Interval Error

The Maximum Time Interval Error, or MTIE, calculates the maximum peak to peak variation for a given window size. The MTIE calculation is repeated for multiple window sizes and the MTIE value vs. window size is plotted on a log-log plot. Equation 19 shows the MTIE calculation where 'n' represents the window size.

$$MTIE(n\tau_0) = \max_{1 \le k \le N-n} \left[\max_{k \le i \le k+n} x_i - \min_{k \le i \le k+n} x_i \right], n = 1,2, \dots, N-1 \qquad Eq.19$$

Maximum Time Interval Error provides two important insights into a clock. MTIE will grow to infinity if the clock is not locked, and the MTIE growth rate provides insight into the stability of the clock without a reference. Additionally, MTIE shows how quickly a clock's error can change in a given time window. For example, MTIE produces a different graph for a clock that can go from the most negative error to the most positive error in a single sample, versus a clock that takes many samples to cover that same distance. In SONET applications, the MTIE metric was also used to correctly size FIFOs between network elements. Since quick and large changes in the timing relationship between two clocks would cause buffer overruns or underruns.

3.5 Time Deviation

The Time Deviation metric, or TDEV, is effectively a combination of the standard deviation and a Fourier analysis. The TDEV metric calculates the RMS time error for samples separated by a given window size. The TDEV calculation is repeated for multiple window sizes and the TDEV value vs. window size is plotted on a log-log plot. Equation 20 shows the TDEV calculation where 'n' represents the window size.

$$TDEV(n\tau_0) = \sqrt{\frac{1}{6n^2(N-3n+1)} \sum_{j=1}^{N-3n+1} \left[\sum_{i=j}^{n+j-1} (x_{i+2n} - 2x_{i+n} + x_i) \right]^2}, n = 1,2, \dots, integer\left(\frac{N}{3}\right) \qquad Eq.20$$

The TDEV metric shows how stable a clock is for a given window size. For example, a short term unstable, long term stable reference, like GPS, will have a large TDEV for small window sizes, but a small TDEV for large window sizes. Inversely, a short term stable long term unstable reference, like a crystal oscillator, will have a small TDEV for small window sizes, but a large TDEV for large window sizes. Knowing the TDEV properties of a clock helps the implementer recognize why a clock is failing the timing performance requirements, as well. If the clock fails the performance requirements at small window sizes, then the servo loop is too fast or the clock needs a better timestamp unit. If the clock fails the performance requirement at large tau, the servo loop is too slow or the clock needs a better oscillator. TDEV can also be used to compare the relative stability of two unsynchronized clocks, but that property of the metric may not be useful for the industrial market.

3.6 Example

As an example, we will analyze the following sequence of time error measurements. This time series was artificially created using a statistical clock model for demonstration purposes.

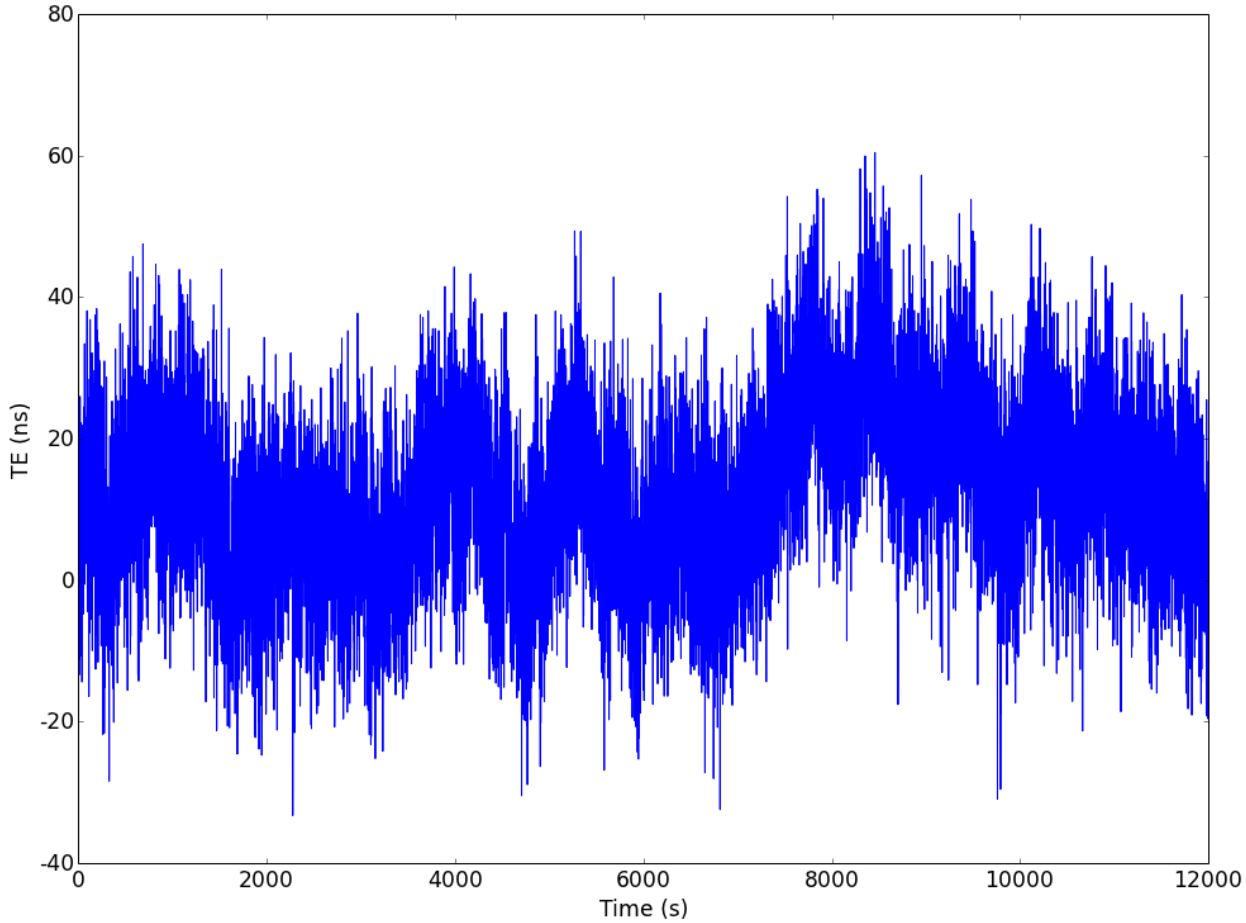

Figure 4

The clock error appears to be bounded between approximately -35 ns and + 65 ns, and the clock clearly has high and low frequency components. If we take a look at the clock's histogram, we see a roughly Gaussian distribution with a mean of 12.4 ns and a standard deviation of 12.7 ns.

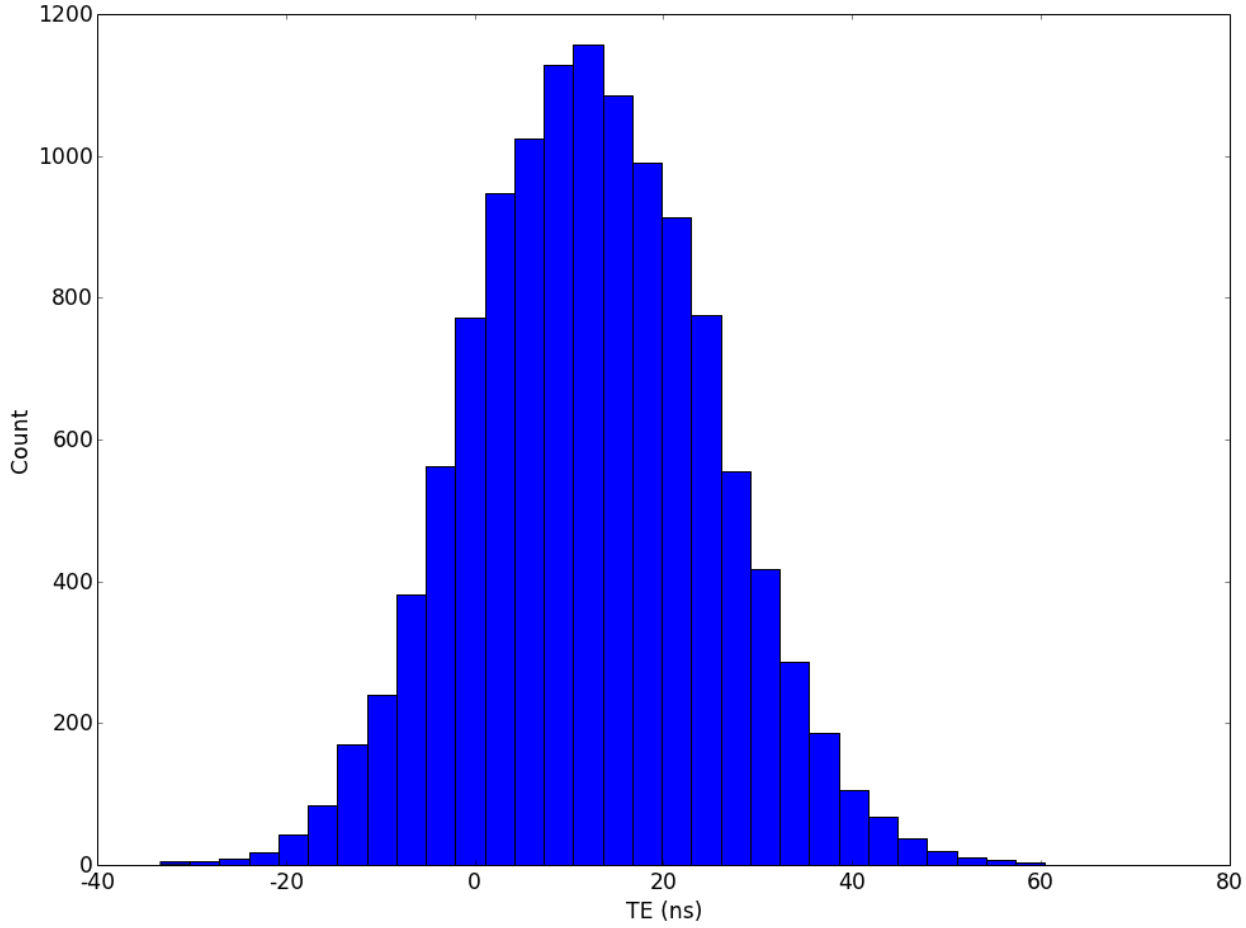

Figure 5

However at this point, we have not fully characterized the stability of the clock. This final graph plots Max|TE|, MTIE, and TDEV on one log-log plot:

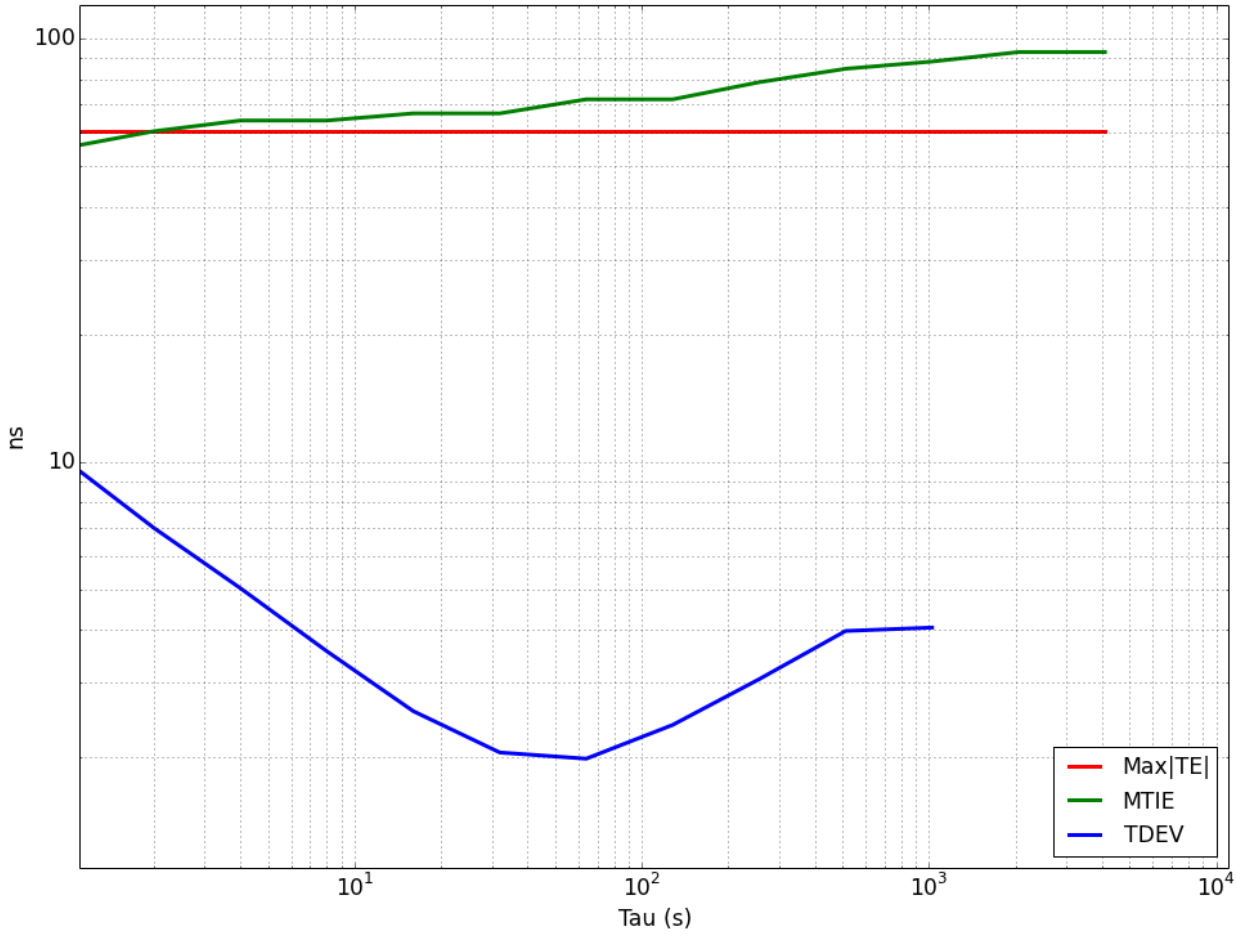

Figure 6

Interpreting the Max|TE| line (red) is fairly easy; the Max|TE| line shows that the maximum absolute error was roughly 60ns. The MTIE (green) and TDEV (blue) lines do require a little interpretation. This graph shows the smallest windows on the left and the largest windows on the right. So the MTIE data point farthest to the left shows that from a one sample to the very next sample, the time error can change roughly 55 ns. The MTIE data point farthest to the right shows that if two samples are separated by roughly 4000 seconds, the time error can change up to 90 ns. Finally the TDEV graph shows how stable the clock is for various time intervals. Starting at the left of the graph, the TDEV graph shows the clock is the least stable for short windows. As the window sizes increases from 1 second to about 30 seconds, the stability of the clock improves and then flattens out. Finally, as the windows get bigger than 60 s, TDEV starts to grow again showing the clock becoming more unstable as the windows grow larger.

4.0 Test and Measurement

Now that we have discussed the metrics, let us look at the test and measurement setups for collecting the Time Error sequences.

4.1 PPS Signals and Oscilloscope

In the he most popular test setup for slave clocks today, the master's 1 PPS signal and the slave's 1 PPS signal are fed into the oscilloscope. The scope is set for infinite persistence and set to trigger on the rising edge of the master's clock. The slave clock's 1 PPS edge will be painted on the screen. If the oscilloscope supports math functions, the oscilloscope can generate a histogram and calculate data like the peak to peak error, the mean, and the standard deviation. If the oscilloscope does not support math functions, the peak to peak error can be measured, and the mean

error roughly approximated. The test setup must compensate for the length of the coax cables, or a bias will be introduced into the measurement. As mentioned before, thermal stability will typically be the limiting factor for a clock's performance, so the Device under Test (DUT) should be tested an environment chamber that generates the worst case applicable thermal ramp. If the clock will be used in a large PTP deployment, the clock should be connected to a representative network to ensure that the clock works in a hierarchy. Otherwise, the network could be a simple Ethernet cable.

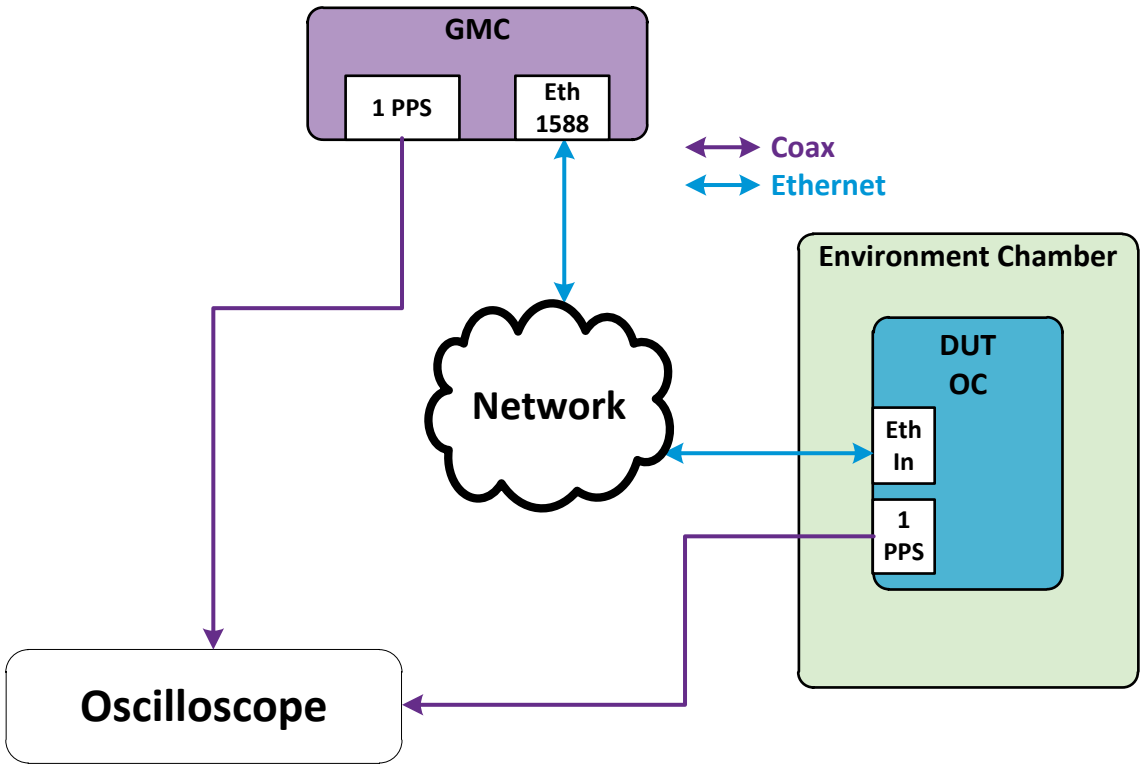

Figure 7

4.2 Purpose Built Timing Test Equipment

Purpose built timing test equipment support all of the metrics described above and bring a host of other features that allow you to unit test a clock. It is hard to overstate the value of unit testing, especially since many of the system level test setups are complicated and expensive. A good unit test can prove out the timing before going into the system test setup, which will save time and improve the efficiency of the system test setup.

4.2.1 Network Emulator

Since PTP is a time distribution protocol, most clocks will always be connected to a chain of other clocks that add error to the timing signals that come from the master. Building a large network and consistently generating the worst case impairments is very expensive and very difficult. Both of the purpose built PTP testers support network emulation, which allows the tester to easily and consistently produce some of the worst case network conditions such as packet drops, packet delay variation, network reconfigurations, or correction field error. The network emulator is not just useful for PTP; the emulator can be used to generate these same conditions for other CIP traffic as well.

4.2.2 Wander Generators

Purpose built timing equipment can also output clocks that replace the DUT's on board oscillator. The test equipment allows the tester to control the wander of the output clocks to emulate the worst case oscillator drift, as an alternative to a temperature chamber. This feature is not as important for DUTs that use regular crystal oscillators

(XOs); since an XO's worst case can be easily reproduced. However, it is more difficult to produce the worst case drift for TCXOs/OCXOs, in which case the jitter/wander generator simplifies the worst case test scenarios.

4.2.3 PPS Signal Wander Measurer
Similar to the oscilloscope test, except a purpose built wander measurer supports all of the metrics described above, and the jitter/wander measurer can plot the time error over time just like an oscilloscope plots voltage over time.

4.2.4 Packet Time Error Measurer
For Transparent Clocks or Boundary Clocks, the primary concern is the quality of time being delivered to the next clock in the chain. Purpose built timing equipment can analyze the time error of the PTP packets, and perform all of the same features that are available to the PPS wander measurement tools.

4.2.5 OC Test Setup
Figure 8, shows a typical setup used to test Ordinary Clocks. The timing test equipment is time and frequency locked to the GMC using a 1 PPS and 10 MHz signal. The timing messages flow from the GMC to the test tool, which emulates the worst case network, then passes the timing data on to the OC. Based on the impaired timing data the OC tries to recover time and outputs a 1 PPS clock. The test equipment measures the time error of the slave's 1 PPS signal versus the GMC's time. The DUT is in an environment chamber to generate the worst case thermal transients.

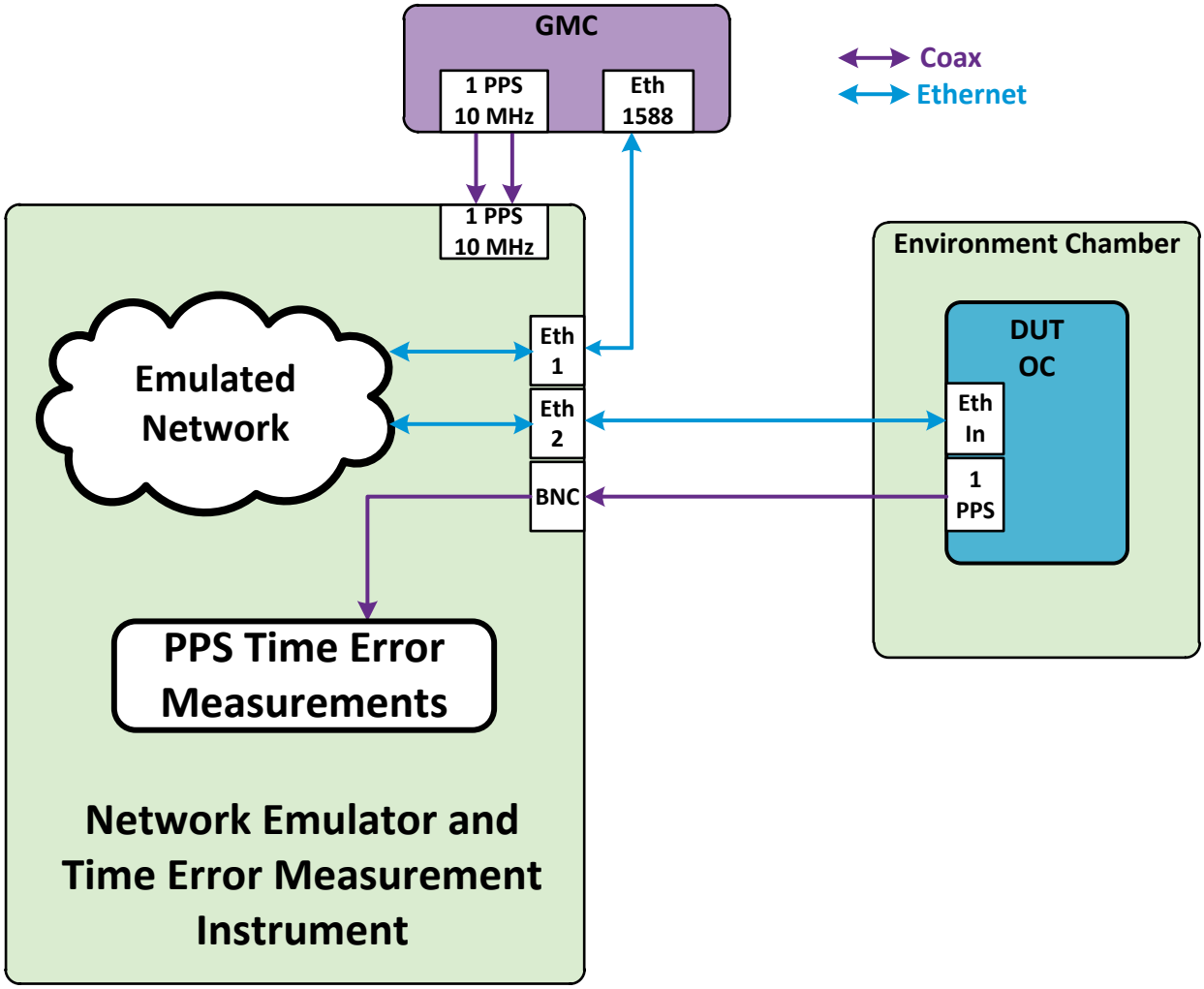

Figure 8

4.2.6 BC/TC Test Setup

Figure 9 shows a typical setup used to test boundary clocks and transparent clocks. The timing test equipment is time and frequency locked to the GMC using a 1 PPS and 10 MHz signal. The timing messages flow from the GMC to the test tool, which emulates the worst case network, then passes the timing data on to the BC or TC. Based on the impaired timing data the BC/TC tries to recover time and forwards time out the second port. The test equipment measures the time error of the BC's or TC's PTP packets versus the GMC's time. The DUT is in an environment chamber to generate the worst case thermal transients. Figure 9 shows a Reference OC in the picture, but the reference OC is optional.

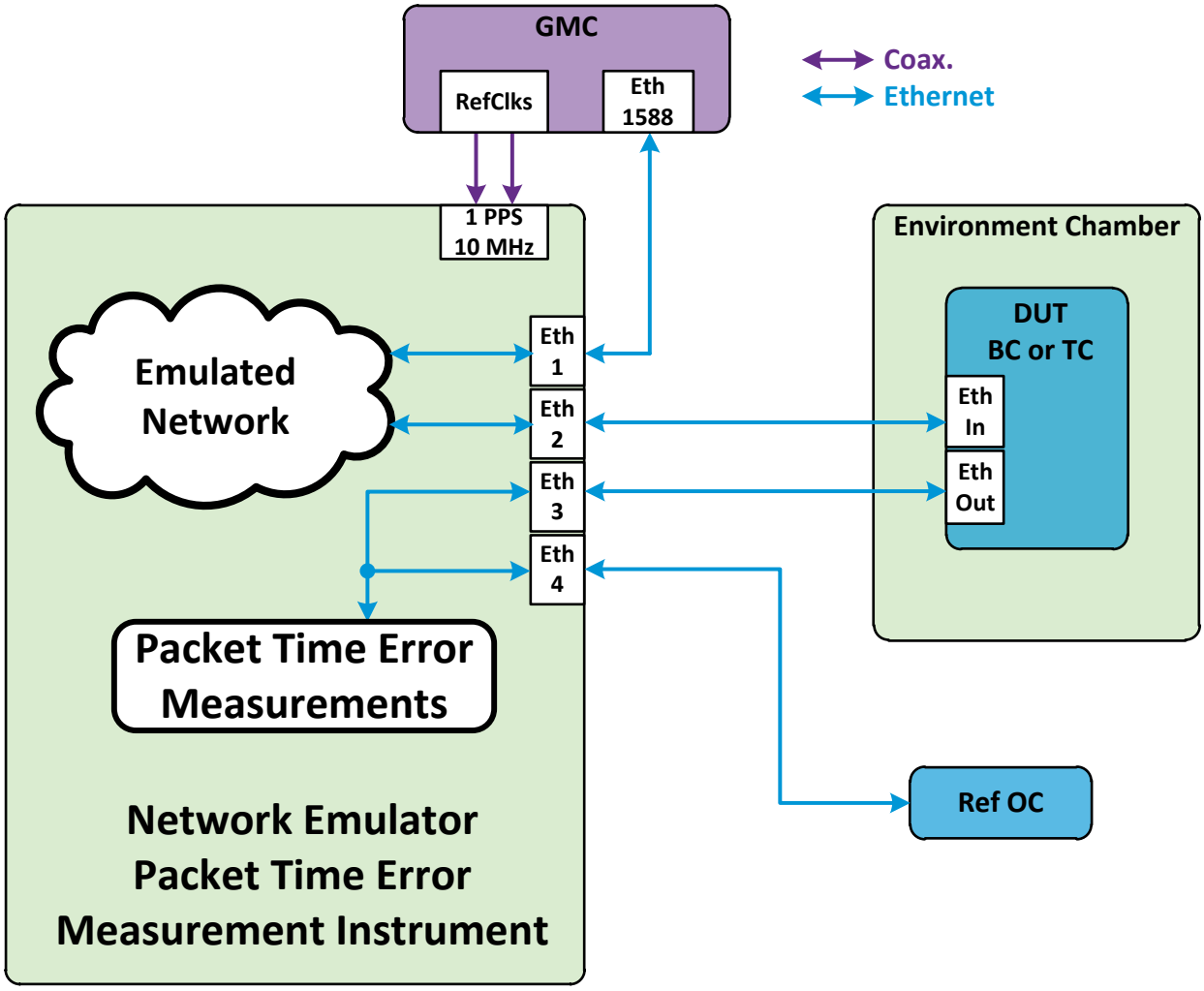

Figure 9

5.0 Summary

We have discussed the implementation and plant models for clocks, and how PTP interacts with these models. As part of the model discussion, we have discussed the disturbances that affect clocks, especially thermal drift and PDV. Clocks have noise processes other than additive white Gaussian noise, and metrics that provide insight into those noise processes provide further insight into the clock. Finally, we discussed the importance of unit testing and how purpose built timing test equipment can improve the development process.

Appendix A: Two Way Time Transfer Mechanics

This appendix provides an overview of Two Way Time Transfer (TWTT) mechanics. In one way time transfer (OWTT), a master clock sends his time to a slave clock. When the slave clock receives the master's time, the slave clock compares the local time to the master's time and corrects for the difference.

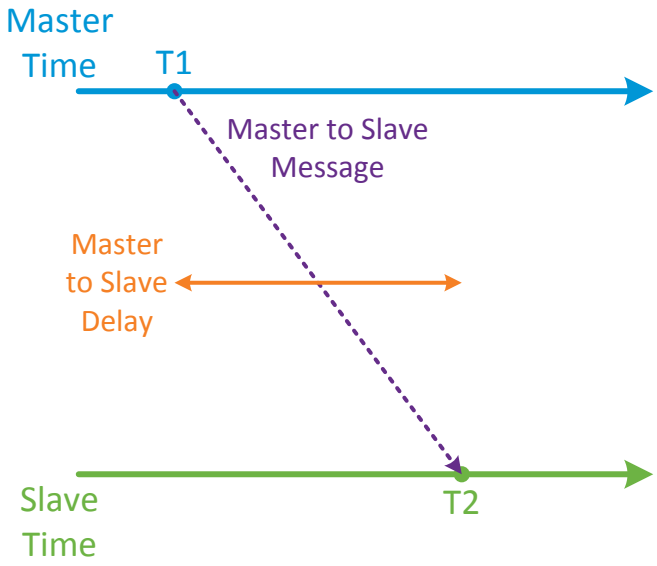

Figure 10

The T1 timestamp is generated when the Master sends a message to the slave, and the T2 timestamp is generated when the slave receives the message from the master.

$T2 = T1 + Delay + Offset$

Unfortunately, sending the master's time to the slave will take some finite amount of time. The transmission latency is known as the delay for time transfer protocols. If the delay between the master and slave is known, or insignificant, then the two clocks will be synchronized. However, if the delay is unknown and significant, then the two clocks will be out of sync, by an unknown amount.

To solve this problem, TWTT adds a return message from the slave to the master, and uses the additional data generated by the return message to approximately measure the delay and the time difference. However, the solution requires lots of assumptions and does not perfectly compensate for the delay.

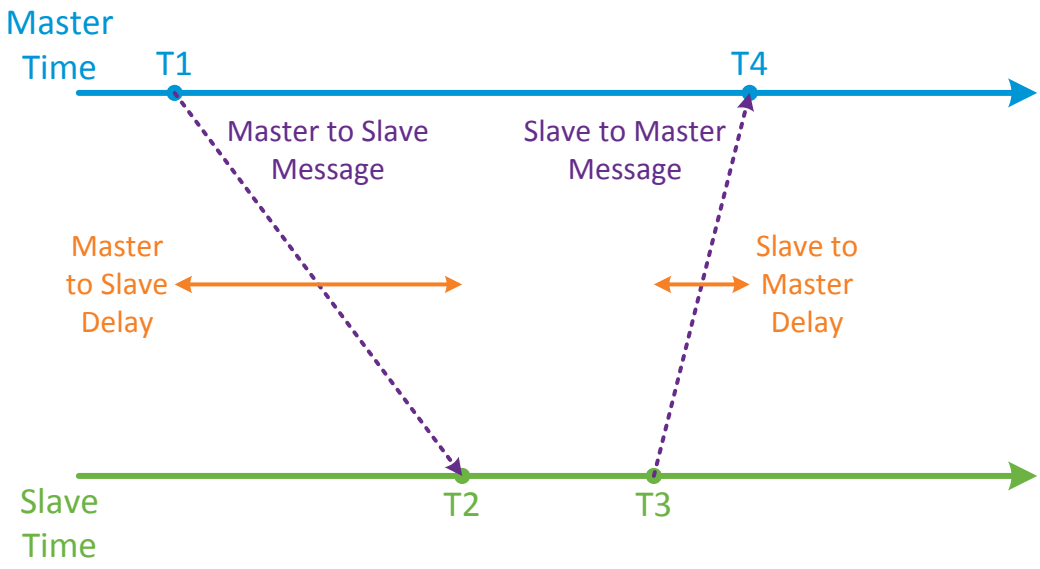

Figure 11

The T1 timestamp is generated when the Master sends a message to the slave, and the T2 timestamp is generated when the slave receives the message from the master.

The T3 timestamp is generated when the Slave sends a message to the master, and the T4 timestamp is generated when the master receives the message from the slave.

What is the relationship between these timestamps?

$$T2 = T1 + Delay_1 + Offset_1$$

$$T4 = T3 + Delay_2 - Offset_2$$

At this point we have two equations with four known values (T1, T2, T3, and T4), and four unknown values (Delay1, Delay2, Offset1, Offset2). Basic algebra tells us it is impossible to determine the delay or the offset with our current number of unknown variables and equations. However, if we assume the two delays are equal, and the two offsets are equal, then our equations reduce to two equations and two unknowns. Which we know can be solved algebraically.

$$Delay_1 = Delay_2 = Delay$$

$$Offset_1 = Offset_2 = Offset$$

$$T2 = T1 + Delay + Offset$$

$$T4 = T3 + Delay - Offset$$

$$Delay = \frac{\big((T2 - T1) + (T4 - T3)\big)}{2}$$

$$Offset = \frac{\big((T2 - T1) - (T4 - T3)\big)}{2}$$

What happens if our assumptions are not true?

If any of the assumptions are violated, the slave will have some time error. The time error may be static or dynamic depending on how the assumptions are violated.

References:

[1] D.W. Allan, N. Ashby, C. Hodge; Science of Timekeeping; Hewlett-Packard Application Note 1289; 1997.

[2] IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems; IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002) ; July 24 2008.

[3] D.B. Sullivan, D.W. Allan, D.A. Howe, and F.L. Walls; Characterization of Clocks and Oscillators; NIST Tech Note 1337; 1990.

[4] CIP Networks Library, Volume 1, Common Industrial Protocol, Edition 3.13, November 2012.

[5] R. Bechmann; Frequency-temperature-angle characteristics of AT-type resonators made of natural and synthetic quartz; Proc. IRE, Vol. 44, November 1956; pp. 1600-1607.

[6] Timing requirements of slave clocks suitable for use as node clocks in synchronization networks; ITU-T Recommendation G.812 (2004).

****