IEEE 1588 Synchronization Over Standard Networks Using the DP83640

National Semiconductor
Application Note 1963
Patrick O'Farrell, David Rosselot
May 14, 2009

1.0 Introduction

National Semiconductor's DP83640 Precision PHYTER® implements time-critical portions of the IEEE 1588 Precision Time Protocol (PTP), allowing high precision IEEE 1588 node implementations. When used with a network consisting of IEEE 1588 capable devices, boundary clocks or transparent clocks, very high precision can be obtained with very simple clock servo algorithms to determine rate adjustments and time corrections. Sophisticated processing is not necessary as only simple averaging or filtering of the protocol measurements is required. When a network consists of devices that are not IEEE 1588 capable, packet delay variation (PDV) is significant. A simple clock servo will not provide a very accurate level of synchronization.

This document describes a method of synchronization that provides much more accurate synchronization in systems with larger PDV. The method described attempts to detect minimum delays, or 'lucky packets'. The method also takes advantage of the DP83640 clock control mechanism to separately control clock rate and time corrections, minimizing overshoot or wild swings in the accuracy of the clock time.

2.0 Background

The IEEE 1588 Precision Time Protocol provides the basic information for the slave to determine both frequency and time offsets relative to the grandmaster clock. The basic algorithm involves measuring the Master-to-slave and Slave-to-master path delays using the Sync and Delay_Req messages respectively.

Figure 1 shows the most basic IEEE 1588 timing diagram.

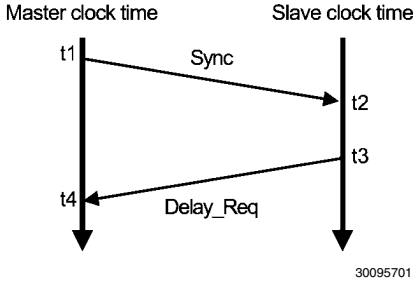

FIGURE 1. Basic PTP Timing Diagram

The Master-to-slave and Slave-to-master delays are:

MSdelay = t2 - t1

SMdelay = t4 - t3

The one-way-delay or meanPathDelay is just the average of these delays:

meanPathDelay = (MSdelay + SMdelay)/2

In the ideal case, the time offset is just:

offset_from_master = MSdelay - meanPathDelay

In a network with network elements (bridges, switches, routers) that include support for IEEE-1588, packet delay variation is essentially negligible. In boundary clock devices, a synchronized clock is maintained on the network element that synchronizes its time and rate to an upstream master, and acts as a master to downstream devices. In transparent clock devices, the packet delay variation is corrected by measuring the residence time for the PTP message as it traverses the device.

In a network with non-1588 capable elements, no compensation is made, resulting in packet delay variation on the order of tens or hundreds of microseconds. These delays become significant and can make any single measurement extremely inaccurate.

Figure 2 shows an MTIE (Maximum Time Interval Error) plot of tests with an 80% traffic condition over a single switch using a basic algorithm with only simple averaging and filtering. As can easily be seen, this provides relatively poor synchronization, with errors as large as 100ms.

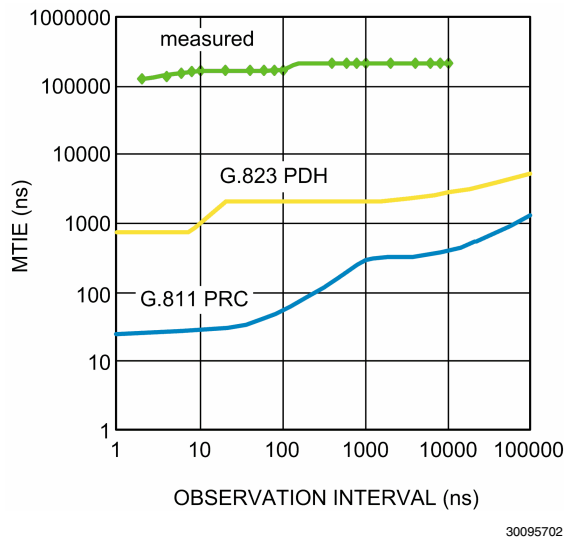

FIGURE 2. MTIE Plot For Basic Algorithm, 80% Traffic Utilization

2.1 PROPOSED ALGORITHM

In a network with non-1588 capable components, the packet delay may range from the minimum physical delay up to the sum of the maximum delays through each device. In practice, there is usually a minimum transmission delay for each device, and therefore a minimum total packet delay from the master to the slave. The basic operation is to attempt to detect the minimum delays, or 'lucky packets', and use the results from these packets to make rate and time corrections. The algorithm is essentially broken down into three stages: meanPathDelay measurement, rate correction, and time correction.

2.1.1 meanPathDelay Measurement

In most networks, the minimum path delay is a relatively constant value. Reconfiguration of the network can cause step changes, but these are usually not very frequent. Thus it is possible to detect the minimum meanPathDelay using a long-term tracking of minimum roundtrip delays (i.e. the full Sync-

Delay_Req computation). The implementation keeps a history of the last N meanPathDelay measurements and finds the minimum value over those measurements:

Min_meanPathDelay(n) = min(meanPathDelay[n+1-N:n]).

Determining the minimum meanPathDelay is critical to doing both the rate correction and time correction.

2.1.2 Rate Correction

Rate correction would typically be done by measuring subsequent sync cycles, and determining the difference between the master measurement of the start of each message versus the slaves measurement of the arrival of each message. This gives a basic ratio of the slave frequency versus the master and can be used to correct for that frequency difference. Since packet delay variation can be significant, this can make any individual rate measurement inaccurate by a significant amount. For example if the sync period is 8 syncs per second, the error might be l00us in 125ms, or close to 1000ppm. If the algorithm were to average all rate measurements, it might require hundreds or thousands of seconds for the rate measurement to converge to a reasonable estimate. Because of the long averaging time, it would also result in a frequency control that could not adapt to short term frequency changes that might occur when using an inexpensive oscillator.

Instead, the proposed algorithm takes advantage of the meanPathDelay measurements to detect low-latency packets and uses only those packets for detecting the rate of frequency offset to the master. If a packet meets the requirements for a good minimum roundtrip delay, the rate is measured by comparing the times since the previous 'good' packet. By using only the low-latency measurements, the convergence time for determining the frequency offset to the master can be reduced significantly. In qualifying 'good' packets, there is a trade-off between quality and quantity. If the qualification is too restrictive, not enough information will be obtained to track frequency changes. If not restrictive enough, the rate calculations may include excessive variation.

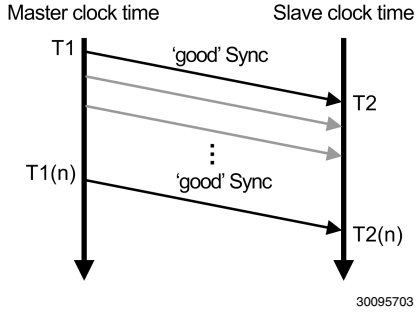

FIGURE 3. Rate Correction Diagram

Figure 3 shows the basic relationship between Sync messages used to determine the rate. From the diagram, the rate ratio would be:

rate_ratio(n) = (T2(n)- T2) / (T1(n)- T1)

In addition, to prepare for the next measurement:

T1 = T1(n), and T2 = T2(n)

Since there still could be some error in measurements, some averaging or filtering of measurements is still required. For simplicity, an exponential moving average, or smoothing function, is used to track the rate. The equation is of the form:

rate_avg(n) = rate_avg(n-1)

+ α(rate_ratio(n) - rate_avg(n-1))

The value of α is typically set to 0.1, but may be increased under certain conditions such as extended periods of increasing or decreasing rate. This allows faster adjustment of the rate under those conditions.

2.1.3 Time Correction

The typical implementation of the time offset determination uses a Sync message to determine an offset versus the master. Some level of averaging or filtering would normally be used to smooth out connections and avoid over correcting for each measurement. For time correction, two different mechanisms were tested to detect and correct time offset.

In the first mechanism, the basic idea is to search for minimum delays. The basic algorithm looks at the minimum Master-to-slave delay over a number of the most recent delays. The time correction may also be limited to prevent over-correction. This algorithm does rely on a larger number of Sync messages than would be required for a IEEE-1588 capable network. Additionally, following a Delay_Req measurement, the algorithm may use either the Master-to-slave delay or the Slave-to-master delay, whichever yields the smaller offset. In cases where one direction of traffic may become congested, the other direction may provide a more accurate measure of the time offset. This method will make a correction each cycle based on the best information it has. This may result in improper corrections if there are no true minimum delay messages received. The cause for this is that the algorithm cannot determine if the measurement error is due to a time offset or to packet delay variation.

The second mechanism for time correction attempts to only use delays to make corrections if they are determined to be actual minimum delay packets. This helps to prevent invalid corrections to the time value. The basic idea is to use both Sync and Delay-Req messages to make time corrections. For Sync messages, if the MSdelay is less than the Min-meanPathDelay, then the measurement indicates there is time offset of at least MSdelay Min-meanPathDelay. In this case, a time correction would be made based on the offset measurement. If the MSdelay is greater than the Min-meanPathDelay, there is no way to tell if the error is due to a time offset or due to PDV, so no correction is made. Similarly for Delay-Req messages, if the SMdelay is less than the Min-meanPathDelay, then the measurement indicates there is a time offset of at least Min-meanPathDelay - SMdelay. Note that this will result in a positive offset detected rather than a negative offset as seen with the MSdelay measurement.

Both implementations makes time corrections by adjusting the PTP clock rate for a period of time. To prevent large fluctuations in rate, each connection is limited in magnitude. This also helps to reduce time interval errors due to rapid corrections of time offsets. In the second mechanism, this is handled by keeping a TimeError value. When a new error is computed due to Sync received or Delay-Resp received, if it indicates a greater offset, TimeError takes the new value. Otherwise TimeError remains unchanged. Based on the TimeError, a limited correction is made and subtracted from TimeError. Thus a measurement of the offset may take multiple corrections before it is completely corrected.

The second mechanism is less likely to make invalid corrections, but may exhibit longer periods without corrections and be more likely to drift based on the error in the rate correction. Overall results are similar, although the second mechanism appears to do better under heavy traffic conditions and multiple switches. Since the second mechanism produced better results, the results section details those results.

3.0 Test Platform

The evaluation platform used for testing the clock servo algorithms is a custom FPGA-based evaluation platform, Analog Launch Pad (ALP). The ALP platform includes a small FPGA which implements a MAC interface, packet buffering, and MDIO management interface for communication with the DP83640 Ethernet physical layer device. The ALP board also includes a USB interface for communication with a host PC. On the host PC, the ALP software runs the PTP protocol, handling building and parsing packets and controlling operation of the PTP hardware in the PHY. The ALP platform incorporates logic and connections to support two independent PHY devices.

The test platform does have limitations in packet and control throughput that limit the number of sync cycles that may be handled. Synchronization works well up to eight per second, but cannot sustain a rate beyond that. Since telecom and other applications require rates on the order of l00 Sync messages per second, this platform cannot provide synchronization on the same level that an embedded platform could provide. Nothing in the DP83640 hardware should limit the device from working in the higher Sync rate environments. The limitations are specific to the evaluation platform.

The ALP platform provides a GUI and a scripting mechanism which supports the Python scripting language. The testing was all done with the PTPv2 protocol and clock servo algorithms running in Python through the ALP GUI.

For the simplest testing, the network consisted of a single HP Procurve switch. Additional traffic was generated using a separate ALP platform set up to send broadcast traffic to the switch to provide a specified percent utilization of the network. Testing was also done against a network consisting of three switches between the Master and Slave.

The PTP Master used an OCXO as its reference clock source. The PTP Slave used an inexpensive TCXO reference.

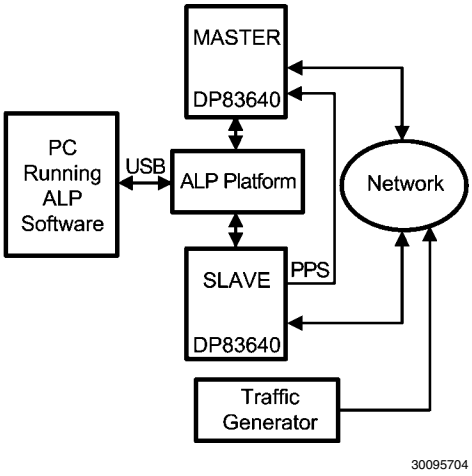

FIGURE 4. Test Platform

4.0 Test Results

The proposed algorithm was used to test performance through a single switch or multiple switches with traffic loads of up to 80%. The master was set to send eight Sync messages per second, while the slave would send a Delay-Resp message for every Sync message. Traffic loading was generated using random size broadcast packets and varying inter-packet gap to generate the specified amount of traffic. The traffic was inserted at an available port on a switch in the test network. Time error data was captured using the Event Timestamp capability of the DPB3640 PTP Master and saved for evaluation. In addition to computing standard deviation, MTIE and TDEV (Time Deviation) plots were generated for each traffic condition. Test durations were a minimum of 4 hours and a maximum of 8 hours.

4.1 SINGLE SWITCH RESULTS

The following figures show the MTIE and TDEV results for a single switch at 20%, 50%, and 80% traffic loading. In addition to the measured results, the plots also show two masks from telecom specifications. The results easily meet the G.823 requirements for the PDH interface, but do not quite meet the G.811 PRC requirements. Further optimization, especially higher Sync rates, would be required to meet the PRC requirements.

TABLE 1. PPS Results For Single Switch Tests

Traffic (% utilization)	Std Dev
20%	13.9ns
50%	15.7ns
80%	28.0ns

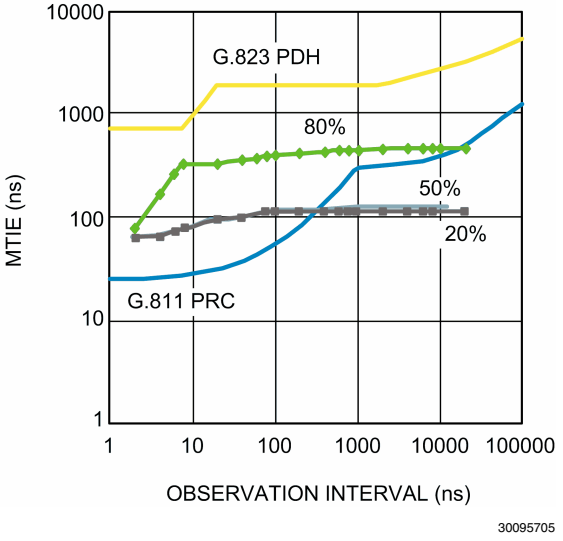

FIGURE 5. MTIE Plot For One Switch Tests

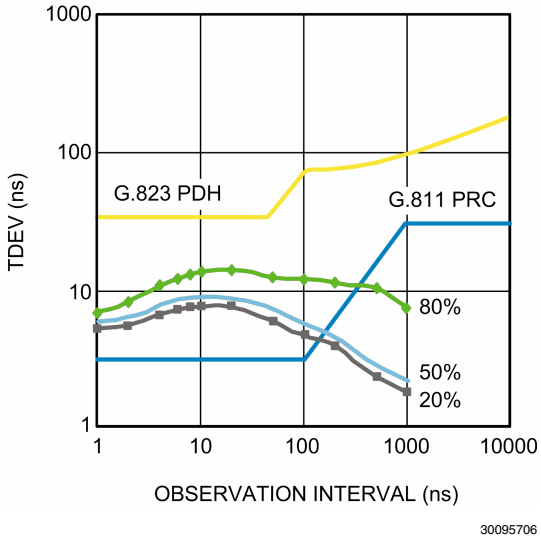

FIGURE 6. TDEV Plot For One Switch Tests

4.2 MULTIPLE SWITCH TESTING

Testing was also done with a 3 switch network consisting of a DLink DES1105 5-port switch, a Linksys SD205 5-port switch, and an HP Procurve 8000M switch. Tests were made at 20% and 50% utilization across all three switches, with the traffic injected at the third switch. As expected, the results were not as good as for a single switch, but still show potential to meet the specifications shown. The following figures show the MTIE and TDEV results for the different traffic conditions.

TABLE 2. PPS Results For Three Switch Tests

Traffic (% utilization)	Std Dev
20%	40.2ns
50%	86.8ns

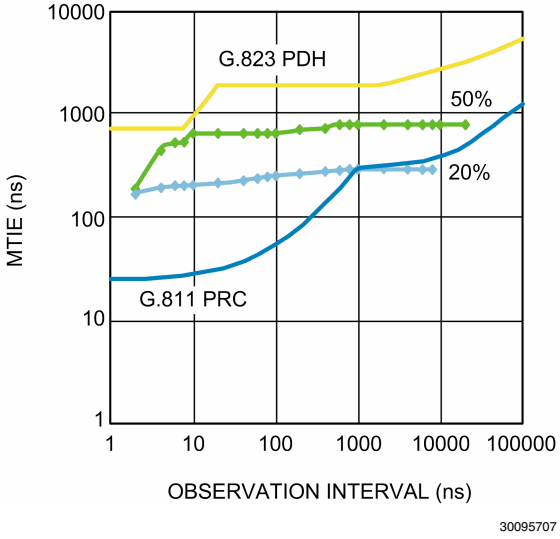

FIGURE 7. MTIE Plot For Three Switch Tests

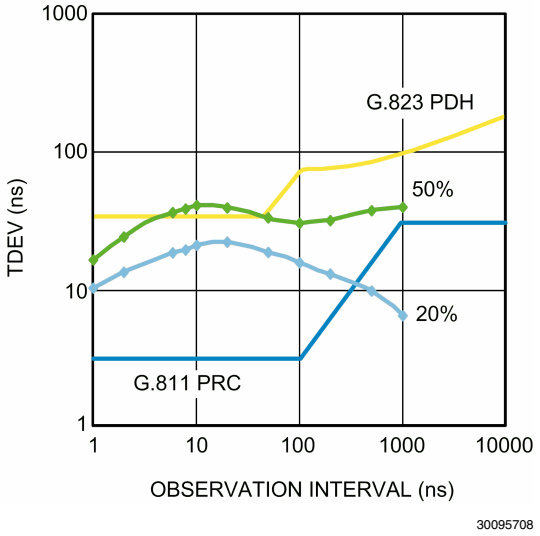

FIGURE 8. TDEV Plot For Three Switch Tests

4.3 OTHER TESTING RESULTS

Although no specific results have been captured, the algorithm appears to respond well to changes in traffic volume within the 0 to 80% range. Beyond 80%, the algorithm still requires some refinement to cope with significantly fewer minimum delay packets.

5.0 Additional Opportunities

While this document describes a working algorithm, there are many possibilities for future development to improve and test the algorithms. The following lists some of the possibilities.

Increase Synchronization Rate. Increasing the synchronization rate requires moving to a platform that can support a higher packet rate, possibly up to 100 syncs/second or more. This would probably require moving to an embedded platform such as the Freescale MCF5234BCCKIT Coldfire platform. Doing this also allows many improvements to the algorithm such as making many delay measurements for each rate or time correction.

Improved Rate Adaption. Some of the largest errors are due to slow response of the algorithm to track changes in frequency of the local oscillator. An improved algorithm to track the rate and make adjustments should improve the overall results.

Testing with Larger Networks. Extend testing to larger networks with more realistic traffic conditions.

Test with VLAN tags. Enabling VLAN tags and allowing for IEEE 802.1Q priority handling of PTP packets based on the VLAN priority field could yield improvements in the results.

6.0 Conclusions

This document describes an algorithm for using the DP83640 Ethernet physical layer device to synchronize clocks across a network with non-IEEE1588 capable devices. The algorithm was not designed as a complete solution across all conditions, but is intended to show the feasibility of such a solution. With only 8 Sync messages per second, the system was able to accurately synchronize across a single heavily loaded switch. With a greater Sync message rate, it is expected that the algorithm will work to a much higher degree of accuracy across larger networks.

7.0 References

IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. (2008) Institute of Electrical and Electronics Engineers, Inc.

DP83640 Precision PHYTER - IEEE 1588 Precision Time Protocol Transceiver. (2008) National Semiconductor Corporation. http://www.national.com/ds/DP/DP83640.pdf

National Semiconductor Ethernet PHYTER - Software Development Guide. (2008) National Semiconductor Corporation.

MCF3254BCCKIT: MCF5234 Business Card Controller with IEEE1588 Precision Time Protocol. Freescale Semiconductor. http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=M5234BCCKIT

Notes

For more National Semiconductor product information and proven design tools, visit the following Web sites at:

Products		Design Support	
Amplifiers	www.national.com/amplifiers	WEBENCH® Tools	www.national.com/webench
Audio	www.national.com/audio	App Notes	www.national.com/appnotes
Clock and Timing	www.national.com/timing	Reference Designs	www.national.com/refdesigns
Data Converters	www.national.com/adc	Samples	www.national.com/samples
Interface	www.national.com/interface	Eval Boards	www.national.com/evalboards
LVDS	www.national.com/lvds	Packaging	www.national.com/packaging
Power Management	www.national.com/power	Green Compliance	www.national.com/quality/green
Switching Regulators	www.national.com/switchers	Distributors	www.national.com/contacts
LDOs	www.national.com/ldo	Quality and Reliability	www.national.com/quality
LED Lighting	www.national.com/led	Feedback/Support	www.national.com/feedback
Voltage Reference	www.national.com/vref	Design Made Easy	www.national.com/easy
PowerWise® Solutions	www.national.com/powerwise	Solutions	www.national.com/solutions
Serial Digital Interface (SDI)	www.national.com/sdi	Mil/Aero	www.national.com/milaero
Temperature Sensors	www.national.com/tempsensors	SolarMagic™	www.national.com/solarmagic
Wireless (PLL/VCO)	www.national.com/wireless	PowerWise® Design University	www.national.com/training

For the most current product information visit us at www.national.com

National Semiconductor Americas Technical Support Center
Email: support@nsc.com
Tel: 1-800-272-9959

National Semiconductor Europe Technical Support Center
Email: europe.support@nsc.com

National Semiconductor Asia Pacific Technical Support Center
Email: ap.support@nsc.com

National Semiconductor Japan Technical Support Center
Email: jpn.feedback@nsc.com