Interim Report - Investigation into the Precise Time Protocol

James Cox
Department of Electrical and Electronic Engineering

University of Bath

February 13, 2014

Contents

Acronyms

CSMA/CD Carrier Sense Multiple Access with Collision Detection

GM Grandmaster

GPS Global Positioning System

IEEE Institute of Electrical and Electronic Engineers

LAN Local Area Network

NERC North American Electric Reliability Company

NTP Network Time Protocol

PPS Pulse per Second

PTP Precision Time Protocol

PTPd PTP Daemon

SNTP Simple Network Time Protocol

UTC Coordinated Universal Time

1 Executive Summary

2 Introduction

Synchronising time amongst a number of devices is very important in a wide range of applications. This can either be for implicit time keeping where timing is not referring to a physical clock, but rather on the order of a set of processes to occur. Explicit timing is where the exact time is required, for example in a communications system. The problem is that not every clock can be exactly correct, or it may not be financially feasible to install a very stable and accurate atomic clock in a particular system.

Note that there are two types of clock inaccuracy when trying to synchronise clocks. Firstly they may have initially started at a different time to the others, therefore there is an offset that needs to be corrected. This is called offset correction. The second effect that is noticed is that clocks do not necessarily run at exactly the same speed. Therefore clocks need to be consistantly adjusted, which is called drift correction. Some clock inaccuracy may be tolerable in some circumstances, but in a number of cases a system must be set up to correct for this instability.
There are several industries that would benefit from an accurate timing network. Descriptions of some of the industries are shown below.

Automation Industry
Processes will need to be synchronised exactly, and can only be if their clocks are in sync with one another. If clocks are in sync then processes can be seperated away from communication between each machine and the processing of the control commands [1].

Power Transmission
Time synchronisation is very important in the power transmission industry, case in point in the North American blackout in August 2003 [2]. It made it difficult for the investigation team to be able to sort through the data received when the timestamps were gathered from an inaccurate clock. From the events of this blackout a regulation was put in place to define a minimum absolute accuracy for timestamped data. The adoption of North American Electric Reliability Company (NERC) Standard PRC0181 in 2006 [3] made this law for any recorded substation data in the USA. This requires that timestamped data must be accurate to within 2ms in relation to UTC.

Telecommunications
In telecommunications, timing protocols are being considered when networks need to be syncronised or if mobile base stations need synchronisation pulses.

2.1 Existing Technology

There are currently a few technologies that will deliver time syncronisation to the above industries. These are: Network Time Protocol (NTP), Simple Network Time Protocol (SNTP), synchronisation from Global Positioning System (GPS) satellites, and Pulse per Second (PPS) signals on a seperate channel.

NTP
This is a technology originally designed in 1985 that is used to synchronise clocks over a packet switched network. It is able to achieve synchronisation with Coordinated Universal Time (UTC) within a few millseconds, but can maintain sub-millisecond accuracy on a Local Area Network (LAN) if ideal conditions are met. Errors due to different packet routes or network congestion can decrease this accuracy by 100ms or more. [4]
NTP uses a client-server hierarchy split into ”stratums”. Figure 1 below shows the stratums numbered from 0 to 3.

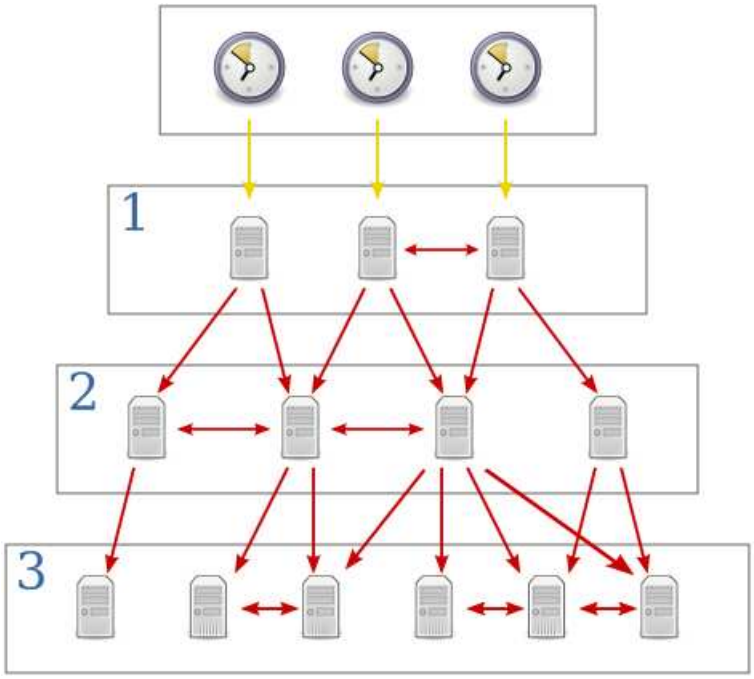

Figure 1: NTP Network Hierarchy [5]

The reference clocks which are in stratum 0 are high precision. Examples of such type clocks are atomic or GPS. The clocks in stratum 2 will base their time off of the clocks in stratum 1. A subset of the clocks in stratum 1 will be used so the overall time is more accurate and robust. Within a stratum clocks may also synchronise for sanity checkup. Any layers below stratum 2 will mirror the same algorithm, and there can be up to 15 layers. Stratum 16 is reserved for clocks that are not synchronised with NTP.
SNTP is a similar protocol but it is complex than NTP as it does not store the state over long periods of time. This is usually used in applications which do not require a high timing accuracy. The SNTP specification can be found in the NTP spec, cited here. [6]

GPS Synchronisation

Time synchronisation can be performed using the high precision clocks found on GPS satellites. Even though this provides a high precision synchronisation, the extra cost and effort may restrict its use to only certain circumstances. [7]

The problem occurs if synchronisation across a packet network is needed but sub-millisecond accuracy is desired. Therefore Precision Time Protocol (PTP) was developed as a successor to the existing NTP standard. Meeting this value of accuracy is very difficult however with a traditional Ethernet network.

When standard switches are used, the packet delay between two nodes is indeterminant. This may be because the packet route from A to B changes depending on network load, or a packet may be held in a switch for an unspecified amount of time whilst working with other data on the network. Therefore this is undesired for PTP when this packet delay must be taken into account when working out the clock offset. Specific timing switches can be used which will prioritise PTP packets, but these may not be available in existing networks or be too expensive to be suitable.

2.2 Current Technology - PTP

Precision Time Protocol (PTP) was developed in 2003 as Version 1 with the intention to build on the existing NTP standard. The Version 1 improvement improved on NTP in a number of different ways. A new PTP standard was introduced in 2008 which again improved on Version 1 with some new features such as boundary clocks. These changes have been tabulated below, Table 1.

Table 1: NTP Vs PTP Version 1 Vs PTP Version 2

Feature	PTP V2 (IEEE1588-2008)	PTP V1 (IEEE1588-2002)
Transparent Clocks	Yes	No
Unicast	Yes	No
Domains	Domain Numbers	Subdomain Name Fields
Clock Quality	Clock Accuracy / Clock Class	Data Field Stratum
Selection Algorithm for best clock	Hierarchical	Selection based
Other Features	Alternate Time Scale Grandmaster Cluster Unicast Masters Alternate Master Path Trace	None

2.3 Project Description

As it can be seen from above, any industry where accurate timing would benefit from PTP, thus determining PTP performance across an network would be useful. Therefore this project will investigate PTP performance across an existing ethernet network. The project specification has been left intentionally open so other work can be carried out time permitting. The university network will be used for this project.

The project will aim to quantify the performance of a PTP system situated on a heavily used Ethernet network. Different packet metrics will be considered when doing this. Other work which may be considered to carry out includes (but not limited to):

- Hardware Slave Clock
- Security of a PTP Network
- PTP Simulation

2.4 Deliverables

The following deliverables have been identified:

- Interim Report
- Packet Metric Scripts
- Final Year Report

2.5 Aims and Objectives

The project can be split into a number of sub goals and objectives. The following goals have been identified:

Learn about PTP and other work in relation to the protocol
This stage would occur at the beginning of the project to understand how PTP works. This is important so work can then be carried out to investigate PTP performance.

Collect PTP Data

In tandem with the above PTP data can be collected. This will be monitoring the performance of PTP across the network as well as how using multiple types of grandmaster/slaves affect the performance. Different clock locations in the network will also be considered.

Implement some packet metric scripts

To be able to understand the performance of the network, some packet metric scripts will be created. A suitable language will be chosen once this part of the project begins.

Determine packet performance using these scripts

Multiple window sizes and types of metric will be used to quantify network performance.

Test Chronos' equipment and provide feedback

As Chronos has provided this project with some equipment, this equipment will also be thoroughly tested and any information gathered can be passed to them once the project is completed.

Based on the goals above and taking into account the deliverables that need to be met for the Individual Design Project, the overall list of tasks is shown below in no particular order

- Research and Learn about PTP
- List all available equipment available
- Set up a small PTP network with one Grandmaster and one Slave clock. Start collecting Data
- Research into different packet metrics that may be used
- Implement these packet metrics in a suitable programming language
- Run these packet metrics with the collected data
- Set a number of slave clocks up throughout the network and run the same packet metric scripts on them.
- Attempt to quantify PTP performance based on these metrics.
- Interim Report
- Final Year Report
- Poster

These tasks have been converted into a Gantt chart which can be found in Figure 2.

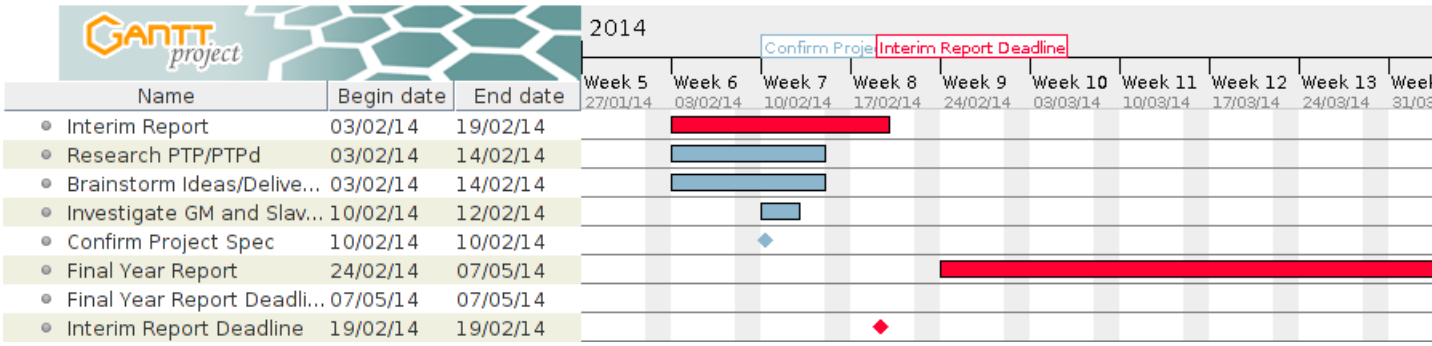

Figure 2: Gantt Chart for Individual Project

The timescales are only estimates, and thus there are some challenges involved if there are delays in any one of these areas. These challenges have been discussed later on in this report.

3 Literature Review

During the first week of the project a number of different Institute of Electrical and Electronic Engineers (IEEE) reports were read and summarised in the logbook. This section of the report will outline some of the documents that were used in order for work to begin. The reports were:

IEEE1588-2008 Specification [?] The 2008 specification for PTP. This was only used as a reference

Prevention of Packet Collisions [8] A journal article describing an algorithm that aims to prevent packet collisions in an Ethernet network.

Definitions and terminology for synchronization in packet networks [?] A standard regarding different packet metrics that could be used in order to try and quantify network delay.

Sub-nanosecond synchronisation [9] A conference paper describing a method of nanosecond accuracy synchonisation

Measurement of Egress and Ingress Delays [10]

3.1 Definitions and Terminology for Synchronisation in Packet Networks

The report defines a number of definitions and terms when dealing with Packet Synchronisation, but the section of interest for this report was Appendix I3 and I4. These can be split into 3 sections: Packet Selection Methods, Packet Metrics without Prefiltering and Packet Metrics with Prefiltering.

3.1.1 Packet Selection Methods

3.1.2 Packet Metrics without Prefiltering

- packet selection methods - explain the use of different packet metrics. - explain the two types of packet metrics - explain each of them briefly - table of pros and cons ?

4 Work Carried out so Far

This section outlines all of the work that has been carried out up until now, as well as any results that have been gathered.

Once the research phase of the project was completed (middle to end of Week 1), work began in implementing some of the packet metric equations that were read in ITU-T G.8260 [**?**]. The Chronos Grandmaster and Slave clocks were also worked on in the first few weeks trying to get them to create a PTP network.

4.1 Packet Metrics

The ITU recommendation document was thoroughly read through and the packet metrics described in that report were implemented. It was decided early on that most, if not all of the packet metrics mentioned in the report would be implemented, and they can then be compared against each other.

The first decision that had to be made was what suitable programming language would be picked for the project. The most suitable language would have to meet the requirements for this section of the project. The following requirements were identified:

- Suitable for reading data, processing the data, and plotting without use of many third party libraries
- Well documented
- Reasonably fast with large data sets

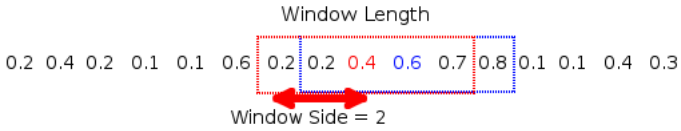

Figure 3: Description of Window and WindowSide

With the above taken into consideration, the decision was to use R. This was primarily because there were other scripts part of PTP Daemon (PTPd) that were written in R, but it was also noted to be a language that would be useful to learn.

Once the language was chosen, the scripts were written based on the approximated equations found earlier in Section 3.2.
Note that git has been used as a version control system for all of these scripts. Both reports written in LaTeX and any scripts created have been version controlled. At time of writing, the following metrics have been written:

1. **TDEV!** (**TDEV!**)
2. minTDEV
3. bandTDEV
4. percentile TDEV

There are other metrics such as cluster TDEV that were looked in to, but the implementation of these will be completed once better understanding of them has been made. This may result in a meeting with some of the engineers in Chronos to discuss these metrics.

4.1.1 General Form

Referring to Equation **??**, this can be implemented in two nested for loops. The quotient can be performed seperately. The function will then return the result.

Before the operation can be performed, the window side length needs to be determined. This is to ensure that when the operation is performed the current index value will be the centre of the mask. The diagram below shows this more clearly in Figure 3.
The full source code for each script will be in the appendix, but any relevant differences will be added to the main body of the report.

4.1.2 TDEV

Because different sample sets can be used, the following input parameters to the function will be: To, n, N and x, with x being the dataset. The full source code can be found in Appendix **??**.

The section of code which is different with TDEV is the section of code in the inner loop. This performs a mean on the three terms. This is shown in Listing **??**.

```
interimStep <- interimStep + mean(x[(i + (2*n)) - windowSide:(i + (2*n)) + windowSide])
    - 2* mean(x[i+n - windowSide : i + n + windowSide]) + mean(x[i - windowSide : i +
    windowSide])
```

Listing 1: TDEV Step

This is the inner section of Equation **??** written in R.

4.1.3 minTDEV

The minTDEV script is very similar to TDEV, but the difference is within the main loop. Instead of calculating the mean of each window the minimum value is taken. The code extract below shows the change. Appendix **??** is the full TDEV source code.

```
interimStep <- interimStep + min(x[(i + (2*n)) - windowSide:(i + (2*n)) + windowSide]) -
    2* min(x[i+n - windowSide : i + n + windowSide]) + min(x[i - windowSide : i +
    windowSide])
```

Listing 2: minTDEV step

4.1.4 bandTDEV

The bandTDEV is based off of the TDEV mentioned previously, but it uses a mean ver a particular band (from a to b) instead of a mean of the entire window. Because a bandMean function was not built in to R, one was created in a seperate source file. This enabled the function to be used in a number of the scripts if required. The bandMean function will not be explicitly discussed, but the full source is shown in Appendix **??**.

The full bandTDEV source code can be found in Appendix **??**, with the one line difference shown below.

```
interimStep <- interimStep + bandMean(x[i + 2*n - windowStep : i + 2*n + windowStep],a,b
    ) - 2 * bandMean(x[i+n - windowStep : i + n + windowStep],a,b) + bandMean(x[i -
    windowStep : i + windowStep],a,b)
```

Listing 3: bandTDEV Step

4.1.5 percentileTDEV

The percentileTDEV function is based very similar to be bandTDEV, except that the lower bad (a) is set to 0. Therefore the code itself is very similar for both metrics. A seperate source file has been created for ease of source code management, but these will be consolidated at a later date.

4.1.6 Overall Script

As multiple metrics will be run on the same source, an overarching script was created to handle the following operations:

- Takes in a text file input of the data sent from PTPd
- Converts this data into a suitable form if necessary
- Calls the relevant packet metric functions
- Displays the data in a suitable form (plot and/or CSV output)
- Handles user input from the command line

The full script can be found in Appendix **??**. All of the functionality above has not been added in to the script, but will be completed in the near future.

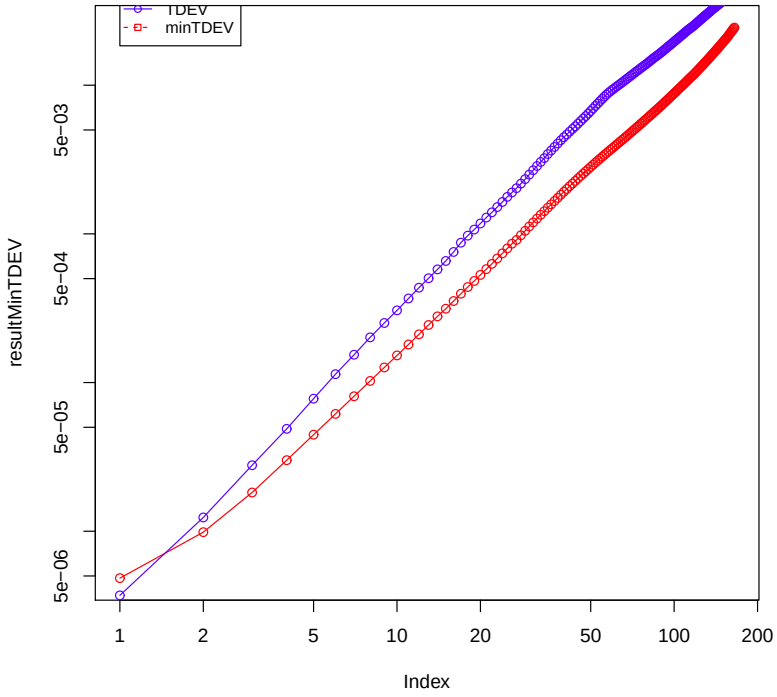

Figure 4: Plot of TDEV and minTDEV for a sample size of 496

4.2 Packet Metric Results

Once the scripts were created, some results were gathered. The data gathering is discussed briefly in a later part of this report.

The dataset available was a 1.2GB file between the Chronos TimePort [**?**] and a software slave. The locations in the network of these clocks were unknown, but most likely it would have been within the EEE department. Any data collected from now onwards will be marked correctly.

The scripts were initially run with the full dataset, but it was quickly realised that the script takes a very long length of time to run in its current state. Therefore a subset of the data was created using the Linux command head. A set of data files were then created, ranging from 50 data points to the maximum.

Note that even though 50 lines of data were collected, the first 4 were truncated. This was to get rid of the headers, two initialisation lines, and a null value.

The plot below (Figure 4) shows the Packet Metric results using the script in Appendix **??**. This was used to generate a TDEV and minTDEV for the first 496 data points.
The raw data output in a tabular form is found in Appendix **??**.

In its current state a lot more data is required to make some sense of these results. The scripts will be amended such that they run much faster (by performing the operation incrementally rather than the total operation on every point), and more data will be collected.

The data was also processed briefly by scanning through the data and noticing if there are any sudden changes or fluctuations of delay. With the initial set of data (not included in this reprt due to it being a large data set), it seems as though the delay is fairly static across the timeframe looked at. This is unusual considering that

standard switches are used rather specific timing switches. More investigation into why this is the case and how the packet metrics described will affect this outcome.

4.3 Chronos Clocks

For this project there are a number of different bits of equipment that can be used for PTP timing. A few days in week 1 were spent working with the Grandmaster TimePort clock from Chronos to see how it worked and what functionality it had. Unfortunately the clock was not able to start the ptp-console during week 2, so more PTP data was not able to be gathered. Chronos will be contacted about this fault and it will be looked into.

4.4 PTP Data Collection

The only data gathered has been from much earlier in Semester 1 and was provided by Dr Robert Watson. It is unknown under what conditions this data was gathered in (ie clock locations on the network) but it provides a suitable basis in which the packet metric scripts can be tested against.

It is expected that a lot more data will be collected during the project.

5 Challenges

- technical challenges - challenges on timing - programming challenges - make a risk table

6 Next Steps

The next steps for the project can be split into a few distinct areas and these will be discussed in turn. The identified next steps for this project are: complete packet metrics, collect data, process the data using the metrics, and any other work.

6.1 Complete Packet Metrics

The plan for this section will be to finish implementing all of the packet metrics mentioned in the paper discussed previously [**?**]. These will be implemented fully and as efficiently as possible. Most likely they will be split into incremental functions rather than the current method at the moment whereby the values are calculated fairly inefficiently. This is important when larger datasets will be used. Another method in resolving this would be to split the workload amongst multiple processes.

6.2 Data Collection

- aims to collect a range of data. + consider different types of clock + different locations + different times of day + different network topologies - describe how each will be carried out - make a plan of how each will be done

6.3 Data Processing

- once data is collected, will process the data - will run each metric for each set up, compare them to each other. - need to understand what each metric does - any other processing techniques (other types of plotting)

6.4 Any other Work

- work for Chronos - cryptographically sign messages ? - *need to ask watson for others*

7 Conclusion

References

[1] D. S. Mohl, "Ptp applications." http://www.ieee1588.com/IEEE1588_PTP_Applications.html, 2010. Accessed: 2014-02-11.

[2] M. R. Bernhard Baumgartner, Christian Riesch, "Ieee 1588/ptp: The future of time synchronization in the electric power industry." https://www.picotest.com/downloads/OTMC100/IEEE_1588_PTP_-_The_Future_of_Time_Synchronization_in_the_Electric_Power_Industry.pdf". Accessed: 2014-02-11.

[3] NERC, "Disturbance monitoring equipment installation and data reporting." http://www.nerc.com/files/prc-018-1.pdf. Accessed: 2014-02-11.

[4] L. Carroll, "Executive summary: Computer network time synchronization." http://www.eecis.udel.edu/~mills/exec.html. Accessed: 2014-02-11.

[5] B. D. Esham, "File:network time protocol servers and clients.svg." {http://en.wikipedia.org/wiki/File:Network_Time_Protocol_servers_and_clients.svg},September2007.newblock Accessed: 2014-02-11.

[6] J. B. D. Mills, J. Martin, "Network time protocol version 4: Protocol and algorithms specification." http://tools.ietf.org/html/rfc5905. Accessed: 2014-02-11.

[7] D. S. Mohl, "Previous solutions." http://www.ieee1588.com/IEEE1588_Previous_Solutions.html. Accessed: 2014-02-11.

[8] C. A, "Preventing the collision of requests from slave clocks in the precision time protocol (ptp)," May 2011.

[9] L. M. . W. T. . S. J. . A. P, "White rabbit: a ptp application for robust sub-nanosecond synchronization," September 2011.

[10] R. C. . M. C. . R. M, "Measurement of egress and ingress delays of ptp clocks," September 2013.

Appendix A TDEV Script

```
#!/usr/bin/env Rscript
#-------------------------------------------------------
#-------------------------------------------------------
#--                 Function Name: TDEV               --
#--                 Name: Time Deviation              --
#--           Input: nTo - position in list           --
#--                  N   - number of samples          --
#--                  x   - vector of samples          --
#--           Output : time deviation                 --
#-------------------------------------------------------
#-------------------------------------------------------
TDEV <- function(To,n, N,x){
# To <- 0.1 #time between samples
# n <- nTo / To #number of samples to current point
  window <- 5 # Set window Size
  windowSide <- (window - 1) / 2 # Set the length of Side of window
  outerStep <- 0
  for (j in windowSide+1:(N-3*n + 1) - windowSide){
    interimStep <- 0
    for (i in j:(n+j - 1)){
```

```
      interimStep <- interimStep + mean(x[(i + (2*n)) - windowSide:(i + (2*n)) +
          windowSide]) - 2* mean(x[i+n - windowSide : i + n + windowSide]) + mean(x[i -
          windowSide : i + windowSide])
    }
    interimStep = interimStep ^ 2
    outerStep = outerStep + interimStep
  }
  outerStep = outerStep / (6 * (N - (3*n) + 1));
  result <- sqrt(outerStep)
  return(result)
}
```

Appendix B minTDEV Script

```
#!/usr/bin/env Rscript
#-------------------------------------------------------
#-------------------------------------------------------
#--                Function Name: minTDEV             --
#--               Name: Minimum Time Deviation        --
#--           Input: nTo - position in list           --
#--                  N   - number of samples          --
#--                  x   - vector of samples          --
#--           Output : time deviation                 --
#-------------------------------------------------------
#-------------------------------------------------------
minTDEV <- function(To,n, N,x){
# To <- 0.1 #time between samples
# n <- nTo / To #number of samples to current point
  window <- 5 # Set window Size
  windowSide <- (window - 1) / 2 # Set the length of Side of window
  outerStep <- 0
  for (j in windowSide+1:(N-3*n + 1) - windowSide){
    interimStep <- 0
    for (i in j:(n+j - 1)){
      interimStep <- interimStep + min(x[(i + (2*n)) - windowSide:(i + (2*n)) +
          windowSide]) - 2* min(x[i+n - windowSide : i + n + windowSide]) + min(x[i -
          windowSide : i + windowSide])
    }
    interimStep = interimStep ^ 2
    outerStep = outerStep + interimStep
  }
  outerStep = outerStep / (6 * (N - (3*n) + 1));
  result <- sqrt(outerStep)
  return(result)
}
```

Appendix C bandMean Scriot

```
#!/usr/bin/env Rscript
#-------------------------------------------------------
#-------------------------------------------------------
#--                Function Name: bandMean            --
#--               Name: Band Mean                     --
#--           Input: window - the samples             --
#--                  a   - lower band                 --
#--                  b   - upper band                 --
```

```
#--            Output : mean of window                --
#-------------------------------------------------------
#-------------------------------------------------------
bandMean <- function(window,a,b){
  sum = 0
  for (i in a:b){
    sum = sum + window(i) # sum window from a to b
  }
  average = sum / (b - a + 1)
  return (average)
}
```

Appendix D bandTDEV Script

```
#!/usr/bin/env Rscript

#-------------------------------------------------------
#-------------------------------------------------------
#--              Function Name: bandTDEV              --
#--             Name: band Time Deviation             --
#--        Input: nTo - position in list              --
#--                N   - number of samples            --
#--                x   - vector of samples            --
#--       Output : band time deviation                --
#-------------------------------------------------------
source("bandMean.r")

bandTDEV <- function(nTo,N,x){
  To <- 0.1 # set minimum step
  n <- nTo / To #number of samples to current
  window <- 15
  windowStep <- (window - 1) / 2 #set side of window
  outerStep <- 0
  a <- 20
  b <- 80
  for (j in 1:(N-3*n + 1)){
    interimStep <- 0
    for (i in j:(n+j - 1)){
      interimStep <- interimStep + bandMean(x[i + 2*n - windowStep : i + 2*n +
          windowStep],a,b) - 2 * bandMean(x[i+n - windowStep : i + n + windowStep],a,b)
          + bandMean(x[i - windowStep : i + windowStep],a,b)
    }
    interimStep = interimStep ^ 2
    outerStep = outerStep + interimStep
  }
  outerStep = outerStep / (6 * (N - 3*n + 1))
  result <- sqrt(outerStep)
  return(result)
}
```

Appendix E Packet Metric Script

```
#!/usr/bin/env Rscript

# -------------------------------------------------------
# -          Script Name: PacketMetric.r                -
```

```
# -        Description: This script will calculate    -
# -        the packet metrics for the given data set. -
# ------------------------------------------------------

source("TDEV.r") #Import TDEV Script
source("minTDEV.r") #Imprt MinTDEV Script
#---------- Import Data into script ----------
arguments  <- commandArgs()
sampleSize <- arguments[6] #Command line args start from index 6

fileName = paste("../PTPData/TestData/SampleSize_", sampleSize, ".txt",sep="")
print(fileName)

print ("Reading CSV Data...")
Data <- read.csv(file = fileName,head = TRUE, sep=",")
print ("CSV Data has been written to Data variable")
delays <- as.matrix(Data[4])
# delays <-sort(delays) Sort if needed
To <- 1/16 #Assume To = 1/16
# ---- Removes Init Messages and the first value
delays = delays[-1]
delays = delays[-1]
delays = delays[-1]
#print (delays)
N <- as.numeric(sampleSize) - 4 #1 for the header, 2 for init, and 1 for the null value

maxn = floor(N / 3)

resultTDEV = matrix(0,maxn)
resultMinTDEV = matrix(0,maxn)

for (i in 1:maxn){

  resultTDEV[i] <- TDEV(To,i,N,delays)
  resultMinTDEV[i] <-minTDEV(To,i,N,delays)
}

#print(resultMinTDEV)
rangeOfValues <- range(0,resultTDEV,resultMinTDEV)
print(rangeOfValues)
#Name pdf file..
outputFileName = paste("../PTPData/Plots/Packet Results - Sample Size - ",N,".eps",sep =
    "")
setEPS()
postscript(outputFileName)
plot(resultMinTDEV,type="o", col="red",log="xy")
lines(resultTDEV,type="o",col="blue")
legend(1,rangeOfValues[2],c("TDEV", "minTDEV"), cex = 0.8,col=c("blue","red"), pch
   =21:22, lty=1:2)

dev.off()
```

Appendix F Table for a sample size of 500