Final Year Report - Investigation into the Precision Time Protocol

James Cox
Department of Electrical and Electronic Engineering

University of Bath

March 20, 2014

Abstract

Abstract goes here

Contents

1 Introduction **4**
- 1.1 How PTP Works . . . 5
- 1.2 Project Description . . . 5

2 Literature Review **5**
- 2.1 Lit Review for Packet Metric . . . 6
- 2.2 Other Relevant Reading . . . 6

3 Project Methods **6**

4 Data Collection **6**
- 4.1 Hardware - Timeport [**?**] . . . 6
  - 4.1.1 Description . . . 6
  - 4.1.2 How to Set Up the TimePort . . . 7
- 4.2 Hardware - Chronos Syncwatch [**?**] . . . 7
  - 4.2.1 Description . . . 7
  - 4.2.2 How to Set Up the SyncWatch . . . 8
  - 4.2.3 Hardware - Beaglebone Black [**?**] . . . 8
  - 4.2.4 Description . . . 8
  - 4.2.5 How to Set up the Beaglebone Black . . . 8
  - 4.2.6 Sending Data to the Local Machine . . . 10
- 4.3 Software - PTPd . . . 11
- 4.4 Data Collection Overview . . . 11
- 4.5 Locations for Clocks . . . 12
- 4.6 Test Sheets . . . 12
- 4.7 Testing Schedule . . . 12

5 Packet Methods **13**
- 5.1 Choosing a suitable language . . . 13
- 5.2 General Packet Metric Implementation . . . 15
  - 5.2.1 TDEV and TDEV derivatives . . . 15
  - 5.2.2 MATIE and MAFE derivatives . . . 15
  - 5.2.3 Overall Packet Script . . . 15
  - 5.2.4 Testing . . . 15
  - 5.2.5 Optimisation . . . 15
- 5.3 Calculating Results . . . 15
  - 5.3.1 Other Utility Scripts . . . 15

6 Results **15**

7 Discussion **15**

8 Milestones **15**

9 Conclusion **15**

10 Acknowledgements **15**

Appendices **15**

Appendix A Gantt Chart and Table **15**

Appendix B Chronos CTL4540 TimePort Specification **16**

Appendix C TimePort Documentation **18**

Appendix D Introduction **18**

Appendix E Scope **18**

Appendix F Diagram of a TimePort **18**

Appendix G How to set the TimePort up as a Grandmaster **18**

Appendix H Problems and Solutions **18**

Appendix I Conclusion **18**

Appendix J Test Sheet Example **18**

Appendix K Test Sheet Summary Sheet **19**

Appendix L Main Packet Metric Script **19**

Appendix M TDEVAllMethods **19**

Appendix N MATIEAllMethods **19**

Acronyms

CSAC

CSMA/CD Carrier Sense Multiple Access with Collision Detection

GM Grandmaster

GPS Global Positioning System

IEEE Institute of Electrical and Electronic Engineers

ITU International Telecommunication Union

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

TDEV Time Deviation

UTC Coordinated Universal Time

List of Figures

1 Chronos TimePort Labelled Diagrams 7
2 Chronos Syncwatch Labelled Diagrams 8
3 Labelled BeagleBone Black 8
4 SSH to Beaglebone 9
5 Rsync Example 10
6 Gantt Chart 15

List of Tables

1 Flags used for PTPd 11
2 Hardware Summary 11
3 List of Criteria for the Language Options 14
4 Language Options Ranking Tables 15

1 Introduction

In several applications, maintaining a high level of time accuracy is very important. This may either be a physical timestamp or that there is a need to synchronise process occuring at specific times. Some examples of these applications are: telecommunications, the automation industry, or the power transmission industry.

There are currently only a few methods to realise this, which depends heavily on the application. For example, if there is space and the budget, a high accuracy atomic clock (CSAC) could be used. If this is not possible,

then a Global Positioning System (GPS) receiver could be used and time can then be synchronised to the accurate clocks onboard the GPS satellites.

This leads to some issues when heavily relying on GPS for time synchronisation. There would be catastrophic consequences in some applications if timing is not kept accurate. The issues relate to the ease of jamming of a **GPS receiver!** (**GPS receiver!**). [reference]

Due to the jammability of GPS, there must be either be a backup solution or an alternative method of time synchronisation in order for accurate timing to still be possible.

One way of realising this is by using a distributed timing system, such as Network Time Protocol (NTP) or Precision Time Protocol (PTP). This would allow for nodes to be able to synchronise their clocks with a much more accurate time source without having to rely on GPS

1.1 How PTP Works

Explain PTP Briefly

1.2 Project Description

This project aims to investigate PTP performance on a heavily used Ethernet network, and to attempt to quantify PTP performance using packet metrics.
There are also some deliverables as part of the Final Year Project. These include: an interim report, a log book, a final year report and a poster. These deliverables, along with the sub tasks involved in order to complete them have been detailed in the Gantt chart, as seen in Appendix A.

The following project objectives 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 on a network.

Collect PTP Data
: In parallel 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.

2 Literature Review

With the following objectives and tasks in mind, a literature review was performed with some suitable documentation: mainly packet metric related, but also on cryptographically signing PTP packets.

The reports below will be discussed in some detail:

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.

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

2.1 Lit Review for Packet Metric

Lit review for packet metric

2.2 Other Relevant Reading

There was other relevant reading performed in the first week of the project to do with cryptography and how packet collisions can be prevented. - cryptography ?

3 Project Methods

Based on the objectives mentioned previously, the project can be split into three distinct sections:

Data Collection
This part of the project will involve collecting PTP timing data on the university network. It will consist of using a number of different clock types and locations on the network.

Packet Methods
This section will mainly involve the implementations of the packet metric scripts based on the referenced report above. Focus on the implementation will be made in this section rather than the metrics themselves.

Calculating/Analysing Results
Once the metrics have been implemented fully, there needs to be some supplementary scripts written to process some of this data.

4 Data Collection

The first step to perform with this part of the project is to work out what hardware is available. The following hardware was identifed as being available to use for the duration of this project.

- Hardware Grandmaster - Chronos TimePort [**?**]
- Hardware Slave - Chronos Syncwatch [**?**]
- Hardware Slave - Beaglebone Black []
- Software Grandmaster - PTP Daemon (PTPd)
- Software Slave - PTPd

4.1 Hardware - Timeport [?]

4.1.1 Description

The Chronos CTL4540 Timeport is a low powered portable device that is able to maintain its time to a high accuracy when disconnected from a synchronisation source. It is able to maintain accuracy within a couple hundred nanoseconds without needing to be connected to GPS. It also has an internal LiPo battery. This enables the device to be used to transport and measure time.

(a) Chronos TimePort Outside

Figure 1: Chronos TimePort Labelled Diagrams

With the above features in mind, it is thus suited for a number of markets, including the power industry and telecommunication network operators. It can also be used to correct for any time errors caused by any cabling or equipment.

Typical methods of doing this would involve using a Caesium atomic clock [REF] or setting up a GPS attenna and connecting this to some other equipment. The TimePort is best suited over these two operaitons because it is much lower power and much more transportable than an atomic clock. It also removes the requirement of GPS equipment.

Appendix **??** shows the full specifications of the CTL4540 TimePort. Below are a few labelled photos of the clock.
The difference between the release TimePort and the TimePort that will be used in this project is that the firmware on the TimePort is bleeding edge. With that in mind time needs to be allocated to allow for any issues that the clock may have. The university has close links with Chronos thus it should be straightforward to either get our issues solved or to receive a new TimePort.

This clock will mainly be kept in the same position on the network and will acct as a Grandmaster.

In terms of documentation there is not much available for this device apart from some emails sent between Chronos and Dr Robert Watson. Therefore Appendix **??** shows some documentation put together for my own use during this project. The documentation includes details on how to interface with the clock and a list of basic commands.

To access the device it needs to be accessed locally over a USB to Serial connection. SSH is unavailable as the control port has not been implemented yet.

4.1.2 How to Set Up the TimePort

- notes in logbook. - will complete this section Thursday. - screenshots too

4.2 Hardware - Chronos Syncwatch [?]

4.2.1 Description

The Chronos Syncwatch is a hardware slave clock used to synchronise time in a number of different applications. It operates in all of the current synchronisation technologies. SyncE, ESMC, PTPv2, 1PPS+TOD, 1PPS, Frequencies(64k-200MHz), T1 & E1 protocols and interfaces are supported.

It can be used on both legacy and modern Ethernet/IP networks. It can simultaneously operate on a number of the protocols above. It can also operate in both local and remote modes.

It is a small modular device with a simple user interface. It also integrates with with Symmetricom's TimeMonitor software.

The device markets include telecommunications, TV and radio broadcasting, and the power industry.

The table shown in Appendix **??** details the Syncwatch specifications. The figures below show labelled diagrams of the inside and outside of the Chronos Syncwatch.

(a) Chronos Syncwatch Outside

(b) Chronos Syncwatch Inside

Figure 2: Chronos Syncwatch Labelled Diagrams

Figure 3: Labelled BeagleBone Black

This product is similar to the TimePort in the fact that there isn't much documentation around for it. Therefore Appendix **??** shows the documentation written up for the Syncwatch.

The Syncwatch will be mainly kept in the upstairs Level 3 Communications lab as it is a larger device. As this device is a release product, all of the ports are working. Therefore the syncwatch can be set up via SSH or using the program. This is all explained in the documentation in Appendix **??**.

4.2.2 How to Set Up the SyncWatch

- notes in logbook. - will complete this section Thursday. - screenshots too

4.2.3 Hardware - Beaglebone Black [?]

4.2.4 Description

The Beaglebone Black is a hardware device but it is running a software PTP Daemon (called PTPd).

In terms of hardware capabilities it has an ARM Cortex A-8 processor with 512MB of DDR3 RAM. It runs a cut down version of Linux called Angstrom Linux. It has Ethernet connectivity and runs off of a 5V DC supply.

As it runs Linux and can be connected to the network, an SSH server has been set up on it with a static IP address. This made it easy to start the PTP daemon.

Below (Figure 3 is a labelled picture of the BeagleBone Black.
The Beaglebone will be a useful device to use as a slave clock because of its portability. It would be able to be placed anywhere on the network without any disruption to that particular lecture room or lab space.

4.2.5 How to Set up the Beaglebone Black

When the project was started, Robert Watson had the BeagleBone Black working with PTPd already, so there was only some work to be done in order to automate the process.

To set up the Beaglebone Black:

1. Plug in the Beaglebone Black to the 5V adapter.
2. Plug in the Ethernet cable
3. Once the Beaglebone boots you can then access the device over SSH.

Type:

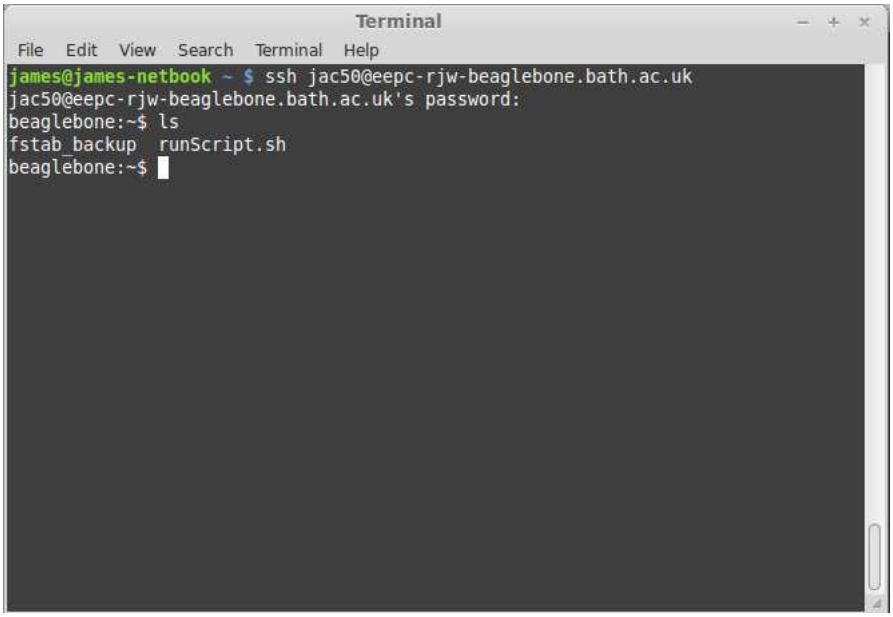

Figure 4: SSH to Beaglebone

```
ssh jac50@eepc-rjw-beaglebone.bath.ac.uk
```

to log in, replacing jac50 with the username on the device. The users on the device were eerjw, jac50, and root.

The screenshot below (Figure 4) demonstrates this. The ls command was typed to show that the connection was successful.
Once SSH'd into the device, then the device can be accessed like any other linux machine. Note however that there is a restricted command set.

When the BeagleBone boots, it was required that the SD card used to store the test data on would be automounted and that the PTP daemon automatically runs.

Several attempts in trying to automount the SD card using conventional means such as adding in an entry to fstab were attempted, but this did not work.

The method in getting round this was by creating a script in /etc/init.d. Any script located in that folder will automatically be loaded once the device boots. The PTP daemon was also run from this same script. The script would also need to automatically name the data file or the data would be overwritten every time the device was turned on. A convention of *timeport_YYYY_MM_DD.txt* was decided.

The full bash script is shown below, in Code Extract 1

```
1 #!/bin/bash
2 mount /dev/mmcblk1p1 /mnt/sd
3 date=$(date +"%Y_%m_%d")
4 sudo /home/eesrjw/ptpd2 -i eth0 -C -S -g -d 17 -V > /mnt/sd/eesrjw/timeport_$date.txt
```

Code Extract 1: Bash Script in init.d for Beaglebone Black

Line 1 is the bash shebang which lets the operating system know that the following script is written in bash. The second line mounts the SD card to the correct location. In this case the SD card is device mmcblk1p1, and the mount location is /mnt/sd/.

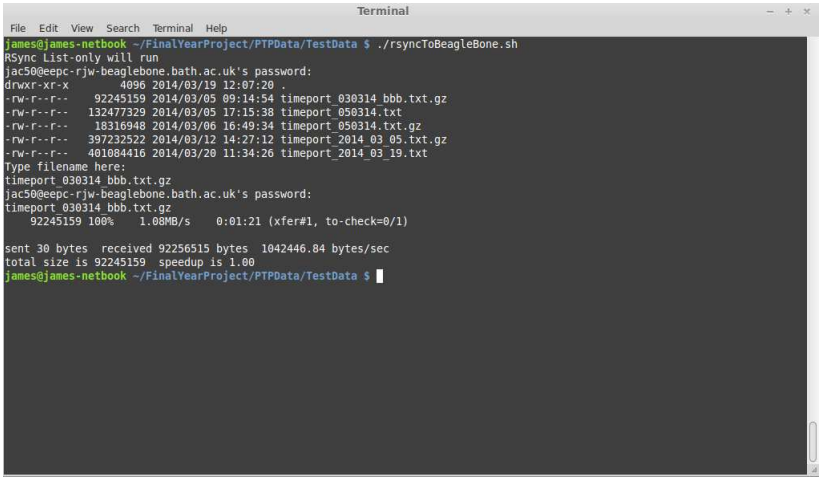

Figure 5: Rsync Example

The third line defines a date variable in Year_Month_Day format. The forth line runs the PTPd2 daemon. As the script is not saved in the PATH variable, the full path to the script is used. The flags will be discussed in more detail in the later section as similar flags will be used.

Once the script above is run (or if the PTPd2 script is run on its own from the terminal), the output is stored in the text file mentioned on Line 5.

Once the test is completed, the script can be killed by using *kill* – 9 on ptpd2. The final step is to transfer the text file from the beaglebone to the local machine ready for packet metrics to be run on it.

4.2.6 Sending Data to the Local Machine

There are two ways to retrieve the data from the Beaglebone: either pulling out the SD card and using an SD card reader to transfer the text file, or remotely using a utility such as rsync.

It was decided that as all other commands are sent to the device remotely, that a short rsync script will be made. Code Extract 2 below is the rsync script used.

```
1 #!/bin/bash
2 echo "RSync List-only will run"
3 rsync --list-only jac50@eepc-rjw-beaglebone.bath.ac.uk:/mnt/sd/eesrjw/ ./
4 echo "Type filename here: "
5 read fileName
6 rsync -v --progress jac50@eepc-rjw-beaglebone.bath.ac.uk:/mnt/sd/eesrjw/i\$fileName ./
    NotSorted
```

Code Extract 2: Rsync Script

The script above prompts the user to type in the filename. It lists the files in the correct directory on the beaglebone in case the user does not know the correct file name. Line 6 then performs the sync operation, using the verbose and the progress flag.

The only issue with this script is that it prompts the user twice for the password. As this script was not run very often it was not an issue. If it was however more research would have been done to see if that could be fixed. A screenshot below (Figure **??** shows the rsync script transferring across a gzipped data file.
It was important to gzip the file beforehand or the transfer would have taken a lot longer. The rate of data collection is around RATE OF DATA HERE

4.3 Software - PTPd

The final type of clock that can be used is a software daemon called PTPd (or sometimes PTPd2). It is a program written in C that meets most aspects of the IEEE1588 specification. PTPd2 meets the changes made in the 2008 standard.

In-depth code analysis of the script will not be provided in this report. Instead the different flags that may be used for this project will be tabulated below (Table 1).

Table 1: Flags used for PTPd

Flag Name	Flag Letter V2.3.0 or above	Old Flag Letter	Description
Interface	i	b	Network Interface to use
Domain	d	i	PTP Domain Number
Foreground	C	C	Run program in foreground
Verbose	V	None	Run in Verbose mode
No Clock Adjust	n	t	Do not adjust the local clock
Slave only mode	s	g	Set PTPd as Slave

PTPd can be used as both a slave and a grandmaster. As there is already a dedicated grandmaster, PTPd will be used mainly as a slave.
The script call has already been given for the beaglebone. The code extract below has been used for the PTPd_Netbook

```
1 ./ptpd2 -C -S -g -i 17 -t | tee /home/james/FinalYearProject/PTPData/TestData/TimePort-
    To-Soft-Test4/RawData.txt
```

The difference in the call to ptpd2 above is that tee has also been used so the data is displayed both on the screen and sent to the text file. This script can be run on any linux computer with root access. Note that the above script in Extract 4.3 is already run as a root user. The other method to do this would be to run the script with sudo.

4.4 Data Collection Overview

As the majority of the devices above will be controlled remotely via SSH, it would be useful for all of them to be on static IPs assigned by the university computing services. All devices were able to get a static IP with a domain forwarding in the format eepc-rjw-nameofdevice.bath.ac.uk. This is a local address which isn't forwarded outside of the university network. The summary table below shows what hardware is available, based on class, and IPs for the PTP port and the control port.

Table 2: Hardware Summary

Clock ID	Name	Type	PTP IP	Control IP	MAC Address
001	Chronos TimePort	Hardware GM	TimePort	USB over Serial	...
002	Chronos SyncWatch	Hardware	syncwatchptp	syncwatch	...
003	BeagleBone Black	Hardware	beaglebone	beaglebone	...
004	PTPd_Desktop	Software	...	...	...
005	PTPd_Netbook	Software	...	...	...

The clock ID was used with internal documentation to know which clock was used where. Note that the IPs listed are just part of the full name. To access one of them on the university network, add the prefix eepc-rjw- and the suffix .bath.ac.uk.

4.5 Locations for Clocks

To get a varied set of data points, it was decided to collect data at a number of locations throughout the network.

4.6 Test Sheets

As there will be quite a few tests performed during the project, and it is important to note times and locations of each test, a test sheet has been created using LibreOffice Calc. An example of a test sheet is found in Appendix **??**

Each test will have the following:

Test ID Each test gets a unique ID number. The number increments for every test performed.

Test Name A general name for the test. This usually consists of the GM clock type, the slave clock type, and the number associated with that type of test.

Test Date The date at which the test was performed in ISO 8601 format.

File Name The file name for the test file. If the test has to be stopped for any reason, a new file is made with a number at the end. The file name is typically RawData.txt.

Directory The directory where the test data is stored.

Clock Type Each clock will have its clock type listed. The clock types have been mentioned earlier in this report.

Clock Name The name of the clock. This is a unique identifer in case multiple clocks of the same type and model are used.

Clock Model The model of the clock.

Start Time The time that the test started. This is as accurate as possible so network data can be correlated with it.

End Time The time that the test finishes. If the test is stopped prematurely but started up again, the final end time is noted here, but the intermediate start and stop time is listed in the comments section.

Network Activity An average network activity for the day (low, medium, high). This is used to correlate delay spread with network activity.

Test Description Brief description of why the test was performed and what the expected outcome of the test is.

Comments Any comments can be noted here. Start/Stop times, or if any issues come up will be noted here.

All of the test sheets will not be shown in this report. Instead a summary sheet of all the tests was produced. The summary sheet will include the Test name, date, directory, and start and end times of the tests. This will show up later in the report.

4.7 Testing Schedule

This part of the project will run in parallel with the implementation stage of the packet metrics, as this does not rely on them being completed. The tests that are to be completed will include:

- Hardware to Hardware
- Hardware to Software

- Hardware to Beaglebone
- Different locations
- Different Times

An explicit testing schedule has not been produced. Instead there are week blocks allocated in the gantt chart for data collection.

5 Packet Methods

This section of the project is the bulk of the programming development. Packet metrics will be created in a suitable language and will be run against the data collected in the previous section.

A range of packet metrics were chosen to be implemented from the report detailed in the literature review section of this report. The following packet metrics will be implemented:

- TDEV
- minTDEV
- percentileTDEV
- bandTDEV
- MATIE
- MAFE

Note that both MATIE and MAFE can have different packet selection methods (min, percentile, or band), so there may be more than the above implemented in the final script.

5.1 Choosing a suitable language

The first decision to make for this section of the project was to choose a suitable programming language. Based on the languages that would be suitable for a task such as this, the following languages were identified: R, C, Matlab, or Python.

The requirements that the language must meet in order to be suitable for the project are listed below.

Note that parts of this section has been copied from the Individual Technical Report for the Third Year Group Business and Design Project as there are some parts that are applicable.

REQ1- Familiarity with the Language
Spec: Used for a sufficient length of time
If the language was very familiar the development time of the scripts would be quicker. This extra time may be acceptable however if there is a much better language ssuited for the task.

REQ2 - Well Documented
Spec: Not Applicable
The majority of modern high level languages are well documented, with some online resources better than others. The language must be well documented so **[REQ7]** can be met. This will also make it easier if **[REQ1]** has not been met fully as it would be easier to learn the language with good documentation.

REQ3 - Plotting Functionality
Spec: Sufficient plotting functionality available
Does the language support complex plotting as standard or are external libraries required?

REQ4 - External libraries already available

Spec: All available

Some external libraries may be needed in case some specific functionality is required. Examples of external libraries that will be required are a command line argument tool and logging functionality.

REQ5 - Speed

Spec: Performs the metrics in a reasonable length of time

The metrics should be able to run on a relatively large dataset in a reasonable length of time. As it is unknown how long the scripts will take, this reasonable length of time will be decided later. If need be optimisation can be made to make the scripts faster.

REQ6 - Linux Compatibility

Spec: Can be developed under Linux

As the rest of the development will be using a Linux Mint netbook, it is a preference for the language to be suitable for a Linux development environment.

To decide on the best solution, a set of ranking criteria was created as well as a ranking table. The table below shows the criteria that the above languages were compared against.

Table 3: List of Criteria for the Language Options

Criterion	Description	Requirement	Weight	Highest - 5	Lowest - 0
Familiarity with the Language	Is the engineer familiar with the language syntax and style?	**[REQ1]**	9	Developed a few large projects.	No familiarity
Plotting functionality	What plotting functionality is available for the language	**[REQ3]**	7	Lots of plotting functionality	All plotting functions would need to be written from scratch
External Libraries available	Are all of the libraries available to complete the project?	**[REQ4]**	7	All of the required libraries are available.	Minimal library support.
Speed	Is the chosen language going to be fast enough for the application?	**[REQ5]**	6	Fast enough.	Not fast at all. Needs careful programming to make as efficient as possible.
Linux Compatibility	Is the language compatible in a linux development environment?	**[REQ6]**	6	Yes, it is compatible.	No. Windows Only
Development Time	How long would it take to develop the first program	**None**	7	Less than a month	Longer than 3 months
Documentation	Is the language mature enough to have a full set of documentation?	**[REQ2]**	6	Yes. The language has clear and concise for all of the documentation.	Very limited.

Table 4: Language Options Ranking Tables

Programming Language				
Ranking Criteria	Weight	Language		
		C++	Matlab	Python
Familiarity with the Language	9	27	36	45
Form Libraries Easy to use	8	24	24	32
Portability	8	40	16	32
External Libraries Available	7	28	21	28
Multithreading Support	7	14	0	28
Development Time	7	21	28	28
Documentation	6	18	18	30
Total Figure of Merit		151	115	195

5.2 General Packet Metric Implementation

5.2.1 TDEV and TDEV derivatives

5.2.2 MATIE and MAFE derivatives

5.2.3 Overall Packet Script

5.2.4 Testing

5.2.5 Optimisation

5.3 Calculating Results

*explain workflow of collecting data

5.3.1 Other Utility Scripts

explain what utility scripts otherwise not mentioned above that will be used.
Awk Data Parser

6 Results

*explain results

7 Discussion

8 Milestones

9 Conclusion

10 Acknowledgements

Appendix A Gantt Chart and Table

Figure 6: Gantt Chart

Appendix B Chronos CTL4540 TimePort Specification

IPPS	200 nanoseconds over 8 hours (±10 °C temp change)
Holdover	100 nanoseconds ove 4 hours (±10 °C temp change)

Inputs

+5V DC:	MiniB USB
GPS antenna:	SMA
Ethernet (PTP and SNTP/NTP):	RJ45 10/100
Ethernet (management):	RJ45 10/100
1PPS (phase 2):	BNC

Outputs

1PPS:	BNC
Frequency 1: 2.048 MHz, 10 MHz	BNC G.703
Frequency 2: 2.048 MHz, 10 MHz	BNC G.703
IRIG-B:	BNC
RS232:	9 way D-Type 9600 band
RS442:	15 way D-Type 9600 band
Ethernet (PTP and SNTP/NTP) (Max 10 clients):	RJ45 10/100
Ethernet (management):	RJ45 10/100

Environmental

Operating Temperature:	0 °C to +50 °C
Maintain holderover tolderance down to:	−10 °C for 15 minutes
Storage temperature:	−20 °C to +80 °C

Physical

Size:	190 x 57 x 170mm (WxHxL)
Weight:	1150g

Appendix C TimePort Documentation

Appendix D Introduction

Appendix E Scope

Appendix F Diagram of a TimePort

Appendix G How to set the TimePort up as a Grandmaster

Appendix H Problems and Solutions

Appendix I Conclusion

Appendix J Test Sheet Example

Test: Timeport_to_Software Test One				
Test Name:	TimePort_To_Software Test One			
Test ID:	001			
Test Date	2014-02-27			
File Name:	RawData.txt			
Directory:	./PTPData/TimePort_To_Software_Test1			
Start Time:	1037			
End Time:	2200			
Clock #1 Type:	Hardware		**Clock #2 Type:**	Software
Clock #1 Name:	TimePort_1		**Clock #2 Name:**	PTPd_Netbook
Clock #1 Model:	TimePort		**Clock #2 Model:**	PTPd
Clock #1 Location:	Watson's Office		**Clock #2 Location:**	2E 2.13
Network Activity:	Normal			
Test Description:	An initial test to collect data to supplement the example data already received.			
Comments	1342: Data seems to be collecting fine. 3hrs20mins: 45MB			

Appendix K Test Sheet Summary Sheet

Test Number	Directory	Master	Slave	Location Master	Location Slave	Sta
001	27/02/14	TimePort-To-Software-Test1	TimePort_1	PTPd_Netbook	2E ..	2E
Finished						
002	28/02/14	TimePort-To-Software-Test2	TimePort_1	PTPd_Netbook	2E ..	2E
Finished						
003	28/02/14	TimePort-To-Software-Test3	TimePort_1	PTPd_Desktop	2E ..	2E
In Progress						
004	03/03/14	TimePort-To-Software-Test4	TimePort_1	PTPd_Netbook	2E ..	2E
Finished						
005	03/03/14	TimePort-To-Software-Test5	TimePort_1	PTPd_Netbook	2E ..	Lib
Finished						
006	03/03/14	TimePort-To-Beaglebone-Test1	TimePort_1	Beaglebone_1	2E ..	2E
Finished						
007	04/03/14	TimePort-To-Software-Test6	TimePort_1	PTPd_Netbook	2E ..	2E
Finished						
008	05/03/14	TimePort-To-Beaglebone-Test2	TimePort_1	Beaglebone_1	2E ..	2E
In Progress						
009	05/03/14	TimePort-To-Software-Test7	TimePort_1	PTPd_Netbook	2E ..	2E
In Progress						

Appendix L Main Packet Metric Script

Appendix M TDEVAllMethods

Appendix N MATIEAllMethods