Final Year Project : Interim Report - Investigation into the Precision Time Protocol

James Cox
Department of Electrical and Electronic Engineering

University of Bath

February 26, 2014

Abstract

The ability to synchronise clocks accurately across an Ethernet link or network is becoming increasingly important. Specific protocols such as the Precision Time Protocol (PTP) have therefore been established to try and improve timing accuracy. This report outlines the initial work undertaken into the investigation of PTP performance, discusses some existing white papers on the subject, as well as detailing the work carried out so far on implementing the packet metrics. Finally the report sets out the work that will be attempted over the next three months as well as highlighting possible risks and mitigation strategies.

Contents

Acronyms

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

1 Introduction

Synchronising time between several devices is very important in a number of different applications such as power transmission or the telecommunications industry. There are two main types uses of time synchronisation: implicit or explicit time keeping. Implicit timing does not refer to a physical clock. Instead it is used to maintain the order of a set of processes. Explicit timing is where the exact time is required to a high degree of accuracy, for example in a telecommunications system where accurate timestamps are required. The reason for using time synchronisation networks is that not all clock can be exactly correct, or it may not be financially feasible to install a stable and accurate atomic clock in some applications. An example of this would be the Global Positioning System (GPS) where four satellites are used to correct for receiver clock inaccuracies.

There are two types of clock inaccuracies when synchronising clocks. Firstly they may have started at a different time relative to the others. Adjusting for this time difference is called offset correction. The second effect is that clocks do not necessarily run at exactly the same speed. Therefore clocks need to be continuously 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. Some examples of these are:

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 also be separated away from communication between each machine and the processing of the control commands. [2]

Power Transmission
Time synchronisation is very important in the power transmission industry. An example of a situation where timing would have mitigated an event from occurring is the North American blackout in August 2003 [3]. 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 the North American Electric Reliability Company (NERC) Standard PRC018-1 in 2006 [4] made it a requirement for any substation in the USA to log data to a minimum accuracy. The timestamped data must be accurate to within 2ms relative to Coordinated Universal Time (UTC).

Telecommunications
In telecommunications, timing protocols are considered when networks need to be synchronised or if mobile base stations need synchronisation pulses. With the increase in GPS jamming, systems such as 4G must rely on other timing methods in case GPS is affected.

1.1 Existing Technologies

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

NTP
This is a technology originally designed in 1985 and is used to synchronise clocks over a packet switched network. It is able to achieve synchronisation with UTC within a few milliseconds, 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. [5]

NTP uses a client-server hierarchy split into ”Stratums”. Figure 1 on the next page shows the Stratums numbered from 0 to 3.

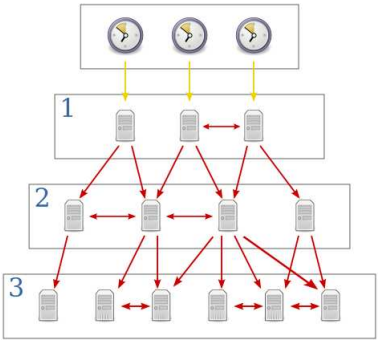


Figure 1: NTP Network Hierarchy [6]

The reference clocks located in Stratum 0 are high precision such as atomic clocks or they use GPS synchronisation. The clocks in Stratum 2 will base their time off of the clocks in Stratum 1. A number of the clocks in Stratum 1 will be used for time synchronising each clock in Stratum 2. This is done so that the time is more accurate and robust. Within a Stratum, clocks may also synchronise for sanity checking to ensure that all clocks within a Stratum are accurate between each other. 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 [7].

SNTP is used in applications which do not require a high timing accuracy. It does this by ignoring drift values. Therefore it is recommended that SNTP is only used in the higher Stratums [8]. The SNTP specification is part of the NTP specification, cited here [9].

GPS Synchronisation

Time synchronisation can be performed using 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 [10]. As this method is heavily reliant on the GPS system for synchronisation, this method would be susceptible to any GPS jamming attempts.

PPS Synchronisation

The final method of synchronisation is using a dedicated channel and a one pulse per second signal which synchronises all the clocks connected to that medium.

Issues arise when synchronising time over a packet switched network where sub millisecond accuracies are required. PTP was developed as a successor to the existing NTP standard which aims to reach sub millisecond accuracies. 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 indeterminate. 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. Therefore this is undesired for PTP as 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.

1.2 Current Technologies

1.2.1 Description of PTP

PTP uses a similar Master-Slave hierarchy of NTP, but it does not use the Stratum method. Instead it uses domains which separate out PTP synchronisation networks. The master clock for the domain will broadcast out the current time to all of the other clocks on the network using a multicast message. In IEEE1588-2008 this can occur up to one message every 100 milliseconds.

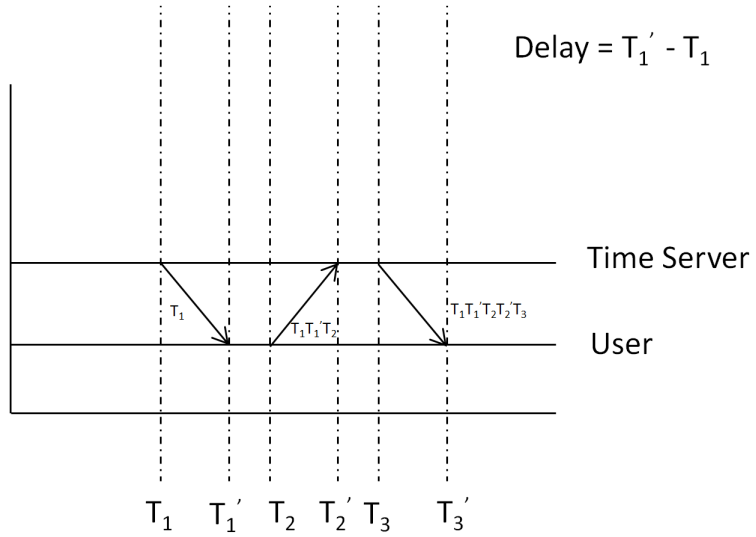


Figure 2: PTP Timing Diagram (based on [11])

The list below shows the basic steps that PTP follows [12]:

1. Broadcast begins at T_1 where the master sends a *sync* message to all clocks on the domain.
2. Each slave clock takes a note of when the message was received using their local clock. This timestamp is labelled $T_1^{'}$.
3. An optional *follow_up* message may be sent that includes an accurate timestamp of T_1. This step occurs if the master clock does not have the capability to create an accurate timestamp when sending the *sync* message.
4. In order for the slave to synchronise with the master, the round trip delay needs to be known. Therefore a *delay_req* message is sent by the slave clock at time T_2.
5. The master will respond to this message with a *delay_resp* message. This timestamp is called $T_2^{'}$

At this point T_1, $T_1^{'}$, T_2 and $T_2^{'}$ are now known.

If we define d as the transit time, and $\widetilde{o}$ as the constant offset between the two clocks:

$$T_1^{'} - T_2 = \widetilde{o} + d \tag{1}$$

$$T_2^{'} - T_2 = -\widetilde{o} + d \tag{2}$$

If we rearrange Equation 2 for d :

$$d = T_2^{'} - T_2 + \widetilde{o} \tag{3}$$

Substituting Equation 3 into 1:

$$T_1^{'} - T_1 = \widetilde{o} + T_2^{'} - T_2 + \widetilde{o} \tag{4}$$

$$T_1^{'} - T_1 - T_2^{'} + T_2 = 2\widetilde{o} \tag{5}$$

$$\widetilde{o} = \frac{T_1^{'} - T_1 - T_2^{'} + T_2}{2} \tag{6}$$

The offset is now known and can be adjusted for. The following assumptions have been made to create the calculations above.

1. Message exchange occurs over a short period of time that the delay is assumed to be constant.
2. Transit time is symmetrical (i.e. time from Master to Slave is the same as Slave to Master).
3. Both the slave and the master can measure the transmit and receive times of messages accurately (ignoring clock drift).

1.2.2 Comparison with NTP

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

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

Feature	NTP	IEEE1588-2002	IEEE1588-2008
Time System	UTC	TAI	TAI - can choose epoch
Transparent Clocks	No	No	Yes
Unicast	No	No	Yes
Domains	Subdomain Name Fields	Subdomain Name Fields	Domain Numbers
Clock Quality	None	Data Field Stratum	Clock Accuracy / Clock Class
Selection Algorithm for best clock	Unknown	Selection Based	Hierarchical
Unique Features	None	Noise Reduction	Alternate Time Scale Grandmaster Cluster Unicast Masters Alternate Master Path Trace

1.3 Project Description

From the above, determining PTP performance across a network would be beneficial to any industry that requires accurate timing. The basic operation of PTP relies on the symmetrical transit times which might not be true in the case of packet switched Ethernet Networks.

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: Hardware Slave Clock, Security of a PTP Network, or PTP Simulation.

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.

1.4 Aims and Objectives

The aims of the project are:

- To develop a set of packet metric implementations
- Quantify PTP performance on the university network

- Generalise results and make a recommendation on switch types / ideal networks for PTP
- A stretch target: Look into methods in making the technology more secure

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 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

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. The reports were:

IEEE1588-2008 Specification [12] The 2008 specification for PTP. This was only used as a reference.

Definitions and terminology for synchronization in packet networks [1] A standard regarding different packet metrics that could be used in order to try and quantify network delay.

Prevention of Packet Collisions [13] A journal article describing an algorithm that aims to prevent packet collisions in an Ethernet network.

Sub-nanosecond synchronisation [14] A conference paper describing a method of nanosecond accuracy synchronisation.

Because the main focus on the project is packet metrics, only the ITU packet recommendation [1] will be discussed in this literature review.

2.1 Definitions and Terminology for Synchronisation in Packet Networks [1]

This paper defines a number of definitions and terms when dealing with packet synchronisation. The areas of interest in this report were packet metrics which are found in Appendix I3 and I4. These can be split into three sections: Packet Selection Methods, Packet Metrics without Pre-filtering, and Packet Metrics with Pre-filtering.

2.1.1 Packet Selection Methods

There are two main methods of selecting packets when calculating a packet metric: either using a selection technique at the same time as the packet metric calculation, or as a pre-processing technique before the metric calculation is performed.

Packet selection, when integrated with the calculation, is very useful when the behaviour of a network is to be determined with respect to its packet delay variation. This is because it provides a generic method that is independent to a particular slave clock implementation [1]. This packet selection method is also known as a Class B metric.

The other method uses a pre-processing technique which preselects packets from a time window. By doing this the process will average out any inconsistencies in the delays, thus resembling a clock running in steady state. Therefore this method is more suitable when trying to specify network limits. This is known as a class A metric).

There are four examples of packet selection methods that are mentioned in the recommendation report. These are: Minimum Packet Selection Method, Percentile Packet Selection Method, Band Packet Selection Method and Cluster Range Packet Selection Method. Each of these will be implemented and are briefly discussed in the Work Carried out so far section of this report.

2.1.2 Packet Metrics without Pre-filtering

The first packet method technique discussed is TDEV. It is used to specify network wander limits for timing signals and can also be used for packet data. TDEV can be applied to both integrated and pre-processed packet selection methods. The implementation equations are quoted in the reference. The approximation equations were used when implementing the functions.

2.1.3 Packet Metrics with Pre-filtering

The other method is using pre-filtering before the metric is calculated. An averaging function is applied to the set of data, but care must be taken to not over-filter the input. This filtered packet sequence can then applied to the metrics mentioned previously in the report. Prefiltered metrics are useful as they can help specify network limits.

2.2 Project Plan

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. This list of tasks was then added to a Gantt chart (See Appendix A) to make sure time has been allocated to certain tasks correctly. Each task has been explained below in a bit more detail. The project deliverables have been listed first.

Log Book This is a continuous log book of work kept throughout the project. It has been used as both a place to document the work completed as well as a notebook of things to remember for the next day. This is due at the same time as the final year report.

Interim Report A report due at the end of the third week. Will mainly information about what the project is about, what work has currently been completed, and what further work will be done. There is also a section on risks and challenges to the project. The information written in the logbook will help with this task.

Final Year Report Report which details all of the work completed throughout the project. The logbook and the interim reports are two prerequisites for this task. Enough time needs to be allocated to this task in order for it to be completed on time and to allow time for proofreading and feedback.

Poster A poster to summarise work completed. Plots are preferred over tables at this stage. A short presentation to project assessors will also be given.

Read about Packet Metrics This task will occur in the first week of the project to understand what sort of packet metrics would be used. Notes will be taken in the logbook.

Read about Protocol This task will only take a few days, but it is an important stage as a solid understanding of the protocol is needed to then be able to complete the later tasks.

Decide on a suitable language A programming language must be decided early, as the language may have to be learnt or revised before work can begin.

Decide on suitable metrics A list of suitable metrics needs to be made before the implementation stage can begin.

Implement metrics This is where the bulk of the programming will occur. Initial testing will be performed on a small set of data when a metric has been completed, but the rest of the testing will happen later.

Test metrics The testing phase of all of the metrics will begin here. The code will be profiled so the scripts run in a reasonable amount of time.

Run tests on sample data During the testing, sample data will also be used as inputs to the script to gather some initial results.

Set up Equipment In parallel with the reading stage, the hardware clocks will be set up to run a PTP network.

Run tests on Chronos Equipment Once the equipment is ready, some data will be collecting using the Chronos equipment.

Run tests on software clocks Preferably at the same time but on a different domain, other testing using purely software clocks can then begin.

Run tests with mix equipment Finally a mixture of equipment can be used to collect data.

Other Tests Any other tests necessary will be performed at this stage.

Run Scripts Metric data will be calculated. A prerequisite to this stage is that all of the metric scripts have been completed and run sufficiently fast.

Other Work and Buffer This is a one week buffer to allow for any other work that might need to be completed and to allow for some extension work if time permits.

As much work as possible will be performed in parallel, such as implementing the metrics and collecting any data. Note that there be days away from university to go visit Chronos or away on BUCS pool tournaments, so this must also be taken into account when making the plan.

3 Work Carried out so Far

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 covered in ITU-T G.8260 [1]. The Chronos Grandmaster [15] and Slave clocks [16] were also worked on in the first few weeks trying to get them to create a PTP network.

3.1 Packet Metrics

The International Telecommunication Union (ITU) recommendation document was thoroughly read through and the packet metrics described in that report were implemented. It was decided early on that all of the packet metrics mentioned in the report would be implemented, and they can be compared against each other.

The first decision that had to be made was what suitable programming language would be picked for the project. It would have to meet the requirements for this section of the project. The following requirements were that it had to be suitable for reading data, processing the data, and plotting without use of many third party libraries. The language had to be reasonably fast with large datasets, or a way to move time critical sections of the code into other faster languages.

With the above taken into consideration, the decision was to use R (also known as GNU S) [17]. This was primarily because there were other scripts part of PTP Daemon (PTPd) that were written in R, but it would also be useful to learn the language. R is also well suited to plotting graphs and is well versed with statistical analysis of datasets.

For version control, Git has been used for all scripts and reports. The source repository for this project can be found on Github [18]. At time of writing, the following metrics have been written: TDEV, minTDEV, bandTDEV, percentileTDEV, MAFIE, MATE (and all of their derivatives). Only the first four have been tested, with some other preliminary work in trying to speed the process by porting the inner loop into C.

There are other metrics such as cluster TDEV that were looked at, 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.

3.1.1 General Form of the Script

The first version will have each packet metric type implemented within its own script. Based off of the equations given in the ITU recommendation report, by having each metric in its own script would lead to duplication of code. This approach was done just to get the metrics working and to have then all standalone. The code can then be refactored and organised in a better way.

Referring to the equations in the report [1], the main form of the script can be implemented in a single for loop (or equivalent vector operation). The quotient can then be performed separately. 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. It is also important that the window does not contain values over the ends of the dataset. The diagram below shows this more clearly in Figure 3.

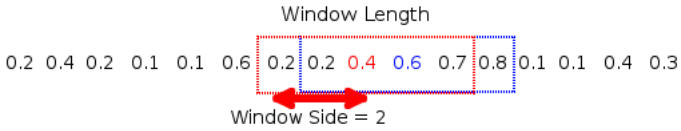


Figure 3: Description of Window and WindowSide

As operations are being performed on a window, array slicing can be used when available in the language specification. This would be the most efficient way of working with sub-arrays, but they would need to implemented in languages that do you not have this functionality.

The input parameters for most of the functions will be: T_o,n, N and x, with x being the dataset. The metrics that rely on a band will also contain a and b.

3.1.2 Time Deviation (TDEV)

TDEV performs a mean on each of the window positions for all of the values in the array. The inner loop step can be seen below in Listing 1. The full source code for the TDEV script is in Appendix B.

```
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

3.1.3 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 in the window will be taken. This would mean that this script should run quicker than the mean. This however will depend on the implementations of both the mean and the minimum functions. The code extract below (Listing 2) shows the change with Appendix C displaying the full minTDEV 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

3.1.4 BandTDEV

The bandTDEV is based off of the TDEV mentioned previously, but it uses a mean over a particular band (from a to b) instead of a mean of the entire window, in effect using a smaller window. As a bandMean function was not built in to R, one was created in a separate 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 D.
The full bandTDEV source code can be found in Appendix E, 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

3.1.5 PercentileTDEV

The percentileTDEV function is based very similar to be bandTDEV, except that the lower band (a) is set to 0. This means that this will take a percentage of the total window size.

```
interimStep <- interimStep + bandMean(x[i + 2*n - windowStep : i + 2*n + windowStep],0,b
    ) - 2 * bandMean(x[i+n - windowStep : i + n + windowStep],0,b) + bandMean(x[i -
    windowStep : i + windowStep],0,b)
```

Listing 4: percentileTDEV Step

As it can be seen the only change is that a has been replaced with integer literal 0.

3.1.6 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 script.
- 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 F. All of the functionality above has not been added in to the script, but will be completed in the near future.

3.1.7 C Implementations

During the initial phases of testing, the scripts were taking much longer than initially anticipated to run. It was found that this was due to how R handles for loops. Instead of this, some C implementations of TDEV and minTDEV were produced and their operation times were compared against each other.

When this was done however it was noted that both scripts were producing slightly different results. Because of this

3.2 Packet Metric Results

Once the scripts were created, some results were gathered. During Week 2 of the project the Timeport was working sporadically, so metric calculations were performed on the data initially provided.

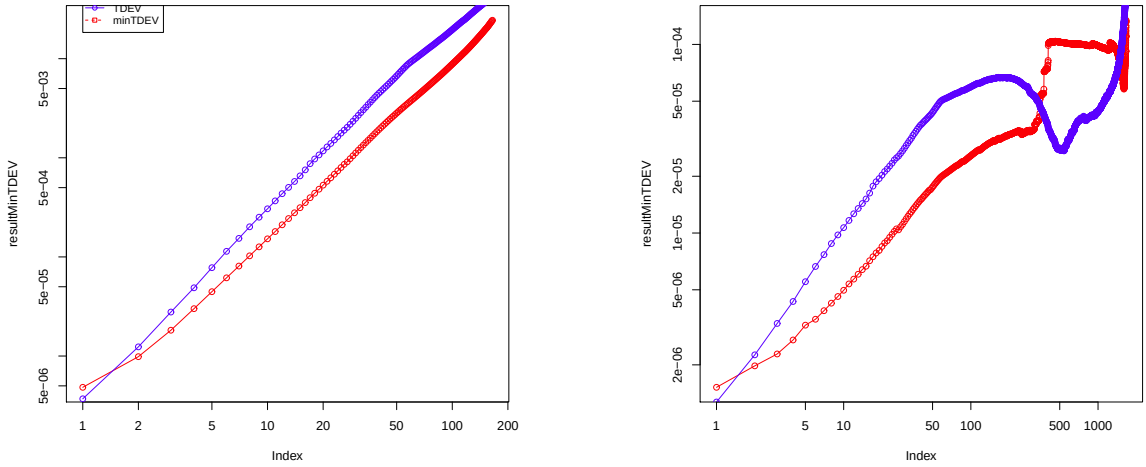
The dataset available was a 1.2GB file between the Chronos TimePort [15] and a software slave (PTPd running on a Ubuntu machine). The locations in the network of these clocks were unknown, but most likely it would have been within the EEE department. Any data collected from now onwards will be marked 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 (over a minute for only 1000 data points). 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 (around 10 000 000).

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

Note that in all of the current tests the sample of the dataset has been taken from the start. Because of network stabilisation this may not be a suitable method in the long run, so data samples from other parts of the data set will be taken at a later date.
The plot below (Figure 4) shows the Packet Metric results using the script in Appendix F. This was used to generate a TDEV and minTDEV for the first 500 and 5000 data points.



(a) Sample Size - 500

(b) Sample Size - 5000

Figure 4: TDEV and minTDEV Plot with different sample sizes

It is hard to draw any initial conclusions from this data set as other metrics need to be run. The data was 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 report), it seems as though the delay is fairly static across the timeframe looked at. This needs to be investigated much further as that mean that time critical switches are not required in a typical PTP network.

3.3 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. Data will be collected from Week 4 onwards with the Grandmaster and any slave clocks that are available.

3.4 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.

4 Challenges

The challenges have been split into the two main sections of the project: data collection and packet metrics. The table below shows the risk of each of these challenges and what their mitigation strategies would be.

Table 2: Risks to the Project

Risk	Description	Severity	Likelyhood	Mitigation
Data Collection				
Time to Collect Data	Is there enough time to collect the data required?	2	2	Start collecting data early in the project
Number of permutations of set ups	Are there enough different clock types and network layouts to be able to draw a conclusion from them?	3	2	Try as many combinations of clocks as possible. Start early
Hardware clock Failures	A hardware clock failing would push the project completion date back	4	3	Have software clocks on standby in case. Speak to Chronos if needed
Network Restrictions	Will the network be restricted?	4	2	Speak with BUCS ahead of time.
Packet Metrics				
Not enough Metrics	Not enough metrics to get a decent result	2	1	Research into other metrics. Speak to Chronos
Metrics not efficient	Implementation of metrics are not efficient	2	3	Start calculating metrics early

As it can be seen above the majority of the risks above relate to making sure that the project is started as early as possible and that both data collection and packet metric implementations can be worked on in parallel.

The biggest challenge for this project is trying to project how long it will take to implement and test the scripts. Therefore the script implementations have occured as early as possible in the project plan. Implementation and testing occurs concurrently. In addition to this as the different packet metrics are related in some way to one another, the implementation and testing time will decrease after the first type of script has been produced.

5 Next Steps

The next steps for the project can be split into a few distinct areas. The identified next steps for this project are: complete packet metrics, collect data, process the data using the metrics, and any other work. The other work that might be included is cryptographically signing PTP packets.

Complete Packet Metrics

The plan for this section will be to finish implementing all of the packet metrics mentioned in the paper discussed previously [1]. 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.

Refactoring Code

The other programming related task will be to refactor the code so that it runs more efficiently. This will be done in a few different ways: collating the scripts together into one or run the metric scripts themselves in C,

but call them from R.

The first method would be the easier one of the two to implement, but the issue may be that looping in R is the bottle neck. Therefore a C script will be written for one of the metrics and profiling of the code will be performed

Data Collection

Throughout the project more PTP data will be collected. A range of sets of data will be taken, which include: different clock types, different clock locations, different times of day, different clock topologies.

The following tests will be performed:

- Chronos Grandmaster / Chronos Slave
- Chronos Grandmaster / Software Slave
- Chronos Grandmaster / Beaglebone Slave
- Software Grandmaster / Chronos Slave
- Software Grandmaster / Software Slave
- Software Grandmaster / Beaglebone Slave

Each of the above will be tested in a few locations throughout the university and at different times of day.

An initial test on similarities between clock performances will be made. If they end up being very similar, only one set up will be used. This can then be tested in different locations on the network.

Data Processing

Once data has been collected, it will be processed and analysed. For each set of data each metric will be computed. Note that one important step will be to understand what each metric does and what it means in terms of PTP performance. These metrics can then be compared against each other and plotted. Other plotting techniques may also be used.

6 Conclusion

In conclusion, an introduction into this project has been given, with a list of the reports that were read initially. The aims and objectives of the project have been clearly outlined. The work that has been completed thus far has been discussed with the next steps also broadly outlined. Finally risks to the project have been identified and the next steps also highlighted.

References

[1] ITU, "Definitions and terminology for synchronization in packet networks," *SERIES G: TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS : Packet over Transport aspects Quality and availability targets*, 2012.

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

[3] 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.

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

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

[6] 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.

[7] D. L. Mills, "Executive summary: Computer network time synchronization." http://www.eecis.udel.edu/~mills/exec.html. Accessed: 2014-02-19.

[8] M. R. CLOCKS, "What is the difference between ntp and sntp?." http://www.meinbergglobal.com/english/faq/faq_37.htm. Accessed: 2014-02-17.

[9] 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.

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

[11] RadhaKrishna.Arvapally, "Ieee1588 communication mechanism." http://en.wikipedia.org/wiki/File:IEEE1588_1.jpg. Accessed: 2014-02-15.

[12] W. P. N. C. S. W. Group, "1588-2008 - ieee standard for a precision clock synchronization protocol for networked measurement and control systems." http://standards.ieee.org/findstds/standard/1588-2008.html, 2008. Accessed: 2014-02-13.

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

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

[15] Chronos, "Ctl4540 timeport." TimePort Datasheet.

[16] Chronos, "Syncwatch standalone." Chronos Slave Clock Datasheet.

[17] R. Project, "The r project for statistical computing." http://www.r-project.org/. Accessed: 2014-02-18.

[18] J. Cox, "Finalyearproject - ptp." https://github.com/jac50/FinalYearProject. Accessed: 2014-02-18.

Appendix A Gantt Table and Chart

	Task Name	Duration (days)	Start	Finish	Predecessors
A	**Deliverables**	72	Mon 03/02/14	Tues 13/05/14	
1	Log Book	65	Mon 03/02/14	Fri 02/05/14	
2	Interim Report	9	Fri 07/02/14	Wed 19/02/14	B, 1
3	Final Year Report	17	Mon 14/04/14	Tues 06/05/14	B,C,D,1,2
4	Poster	5	Wed 07/05/14	Tues 13/05/14	1,3
B	**Preliminary Reading**	**7**	**Mon 03/02/14**	**Tues 11/02/14**	
1	Read up on PTP	7	Mon 03/02/14	Tues 11/02/14	
2	Reading about Metrics	7	Mon 03/02/14	Tues 11/02/14	
C	**Packet Metrics**	**35**	**Mon 10/02/14**	**Fri 28/03/14**	
1	Decide on Language	1	Mon 10/02/14	Mon 10/02/14	
2	Decide on Metrics	1	Mon 10/02/14	Mon 10/02/14	
3	Implement Metrics	24	Mon 11/02/14	14/03/14	1,2
4	Test all Metrics	33	Wed 12/02/14	Fri 27/03/14	3
5	Run some tests on sample Data	8	Wed 12/02/14	Fri 21/02/14	3
D	**Data Collection**	**45**	**Mon 03/02/14**	**Tues 04/04/14**	
1	Set up Equipment	5	Mon 03/02/14	Fri 07/02/14	
2	Run Tests w/ Chronos Equipment	10	Mon 24/02/14	Fri 07/03/14	1
3	Run Tests w/ Software	5	Mon 10/03/14	Fri 14/03/14	1
4	Run Tests with w/ Mix	5	Mon 17/03/14	Fri 21/03/14	1
5	Other Tests	10	Mon 24/03/14	04/04/14	1
6	Run scripts on Results	30	Mon 24/02/14	Fri 04/04/14	2,3,4,5
E	**Other Work and Buffer**	**12**	**Mon 07/04/14**	**Fri 18/04/14**	

	Task Mode	Task Name	Duration	Start	Finish
1		**Deliverables**	**71 days**	**Tue 04/02/14**	**Tue 13/05/14**
2		Log Book	64 days	Tue 04/02/14	Fri 02/05/14
3		Interim Report	9 days	Fri 07/02/14	Wed 19/02/14
4		Final Year Report	17 days	Mon 14/04/14	Tue 06/05/14
5		Poster	5 days	Wed 07/05/14	Tue 13/05/14
6		**Preliminary Reading**	**7 days**	**Mon 03/02/14**	**Tue 11/02/14**
7		Read about possible Packet Metrics	7 days	Mon 03/02/14	Tue 11/02/14
8		Read up on the PTP Protocol	7 days	Mon 03/02/14	Tue 11/02/14
9		**Packet Metrics**	**35 days**	**Mon 10/02/14**	**Fri 28/03/14**
10		Decide on a suitable language	1 day	Mon 10/02/14	Mon 10/02/14
11		Decide on suitable packet metrics	1 day	Mon 10/02/14	Mon 10/02/14
12		Implement all packet metrics	24 days	Tue 11/02/14	Fri 14/03/14
13		Test all metrics	33 days	Wed 12/02/14	Fri 28/03/14
14		Run some tests on the sample data	8 days	Wed 12/02/14	Fri 21/02/14
15		**Data Collection**	**45 days**	**Mon 03/02/14**	**Fri 04/04/14**
16		Set up Equipment	5 days	Mon 03/02/14	Fri 07/02/14
17		Run Tests with Chronos Equipment	10 days	Mon 24/02/14	Fri 07/03/14
18		Run Tests with software only	5 days	Mon 10/03/14	Fri 14/03/14
19		Run Tests with mix equipment	5 days	Mon 17/03/14	Fri 21/03/14
20		Other Tests	10 days	Mon 24/03/14	Fri 04/04/14
21		Run Scripts on Results	30 days	Mon 24/02/14	Fri 04/04/14
22		Other Work and Buffer	5 days	Mon 07/04/14	Fri 11/04/14

Figure 5: Gantt Chart

Appendix B 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 (i in windowSide+1:(N-3*n + 1) - windowSide){
    interimStep <- 0
    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])
    print(c("First Average in R", mean(x[(i + (n) - windowSide) : (i + (n) + windowSide)
        ])))
    interimStep = interimStep ^ 2
    outerStep = outerStep + interimStep
  }
  outerStep = outerStep / (6 * (N - (3*n) + 1));
  result <- sqrt(outerStep)
  return(result)
}
```

Appendix C 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 (i in windowSide+1:(N-3*n + 1) - windowSide){
    interimStep <- 0
      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 D bandMean Script

```
#!/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 E 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 (i in 1:(N-3*n + 1)){
    interimStep <- 0
      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 F 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("../LaTeXScripts/generateLatexTable.r")
source("TDEV.r") #Import TDEV Script
source("minTDEV.r") #Import MinTDEV Script
source("TDEVAllMethods.r")
dyn.load("TDEV.so")

#---------- 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)
RawResult = c(0,0,0,0)
resultTDEV = matrix(0,maxn)
resultTDEVC = matrix(0,maxn)
resultMinTDEV = matrix(0,maxn)
resultbandTDEV = matrix(0,maxn)
resultpercentTDEV = matrix(0,maxn)
a <- 20
b <- 80


for (i in 1:maxn){
  ptm <- proc.time()
  resultTDEV[i] <- TDEV(To, i,N,delays)
  resultMinTDEV[i] <-minTDEV(To,i,N,delays)

  temp <- .C("TDEV",To,as.integer(i),as.integer(N),delays,result = double(1))
```

```
51
52    resultTDEVC[i] <-as.double(temp['result'])
53    #RawResult = TDEVAll(To,i,N,delays,a,b)
54    #resultTDEV[i] = RawResult[1]
55    #resultMinTDEV[i] = RawResult[2]
56    #resultbandTDEV[i] = RawResult[3]
57    #resultpercentTDEV[i] = RawResult[4]
58    #print(paste("Result: ------------------", resultTDEV[i]))
59     print (paste("Iteration", i,"complete in Time:", "..." ))
60  }
61  #print(resultTDEV)
62  #print(resultTDEVC)
63  #print(resultMinTDEV)
64  #print(resultMinTDEV)
65  #rangeOfValues <- range(0,resultTDEV, resultMinTDEV,resultbandTDEV,resultpercentTDEV)
66  #print(rangeOfValues)
67  #Name pdf file..
68  outputFileName = paste("../PTPData/Plots/Packet Results - Sample Size - ",N,".eps",sep =
        "")
69  setEPS()
70  postscript(outputFileName)
71  plot(resultMinTDEV,type="o", col="red",log="xy")
72  lines(resultTDEV,type="o",col="blue")
73  lines(resultbandTDEV,type="o",col="green")
74  lines(resultpercentTDEV,type="o",col="orange")
75  #legend(1,rangeOfValues[2],c("TDEV", "minTDEV","bandTDEV","percentTDEV"), cex = 0.8,col=
      c("blue","red","green","orange"), pch=21:22, lty=1:2)
76
77  dev.off()
78  #Create a CSV output file
79  result <- matrix(0,ncol = 3, nrow = maxn)
80  result[,1] <- seq(1,maxn)
81  result[,2] <- resultTDEV
82  result[,3] <- resultTDEVC
83  #result[,4] <- resultMinTDEV
84  #result[,5] <- resultbandTDEV
85  #result[,6] <- resultpercentTDEV
86  #generateLatex(result,c("Index","TDEV", "minTDEV"), "Raw results of 500 samples for TDEV
        and minTDEV", "table:500sample")
87
88  print(result)
```