Time Synchronization in Distributed Systems without a Central Clock

by

Takin Tadayon

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2019

© Takin Tadayon 2019

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Time synchronization and a common understanding of the concept of time is an essential and important aspect of modern computing. Among the drawbacks of current and popular methods, such as the Network Time Protocol, (NTP), or the Precision Time Protocol, (PTP), are the need for live synchronization, and the use of a central reference clock.

While these methods are ideal in an environment with constant, uninterruptible, and reliable internet connection, they may prove impracticable in a number of scenarios including Internet of Things (IoT) devices. In cases where the is no access to a central clock, or where the accuracy of two clocks is paramount yet there exists no reliable network access, new methods for synchronization are needed.

Our proposed method utilizes existing hardware, without the need for onerous algorithms, in order to establish time-accuracy between two devices. The method effectively establishes a master-slave relationship between two computers, and endeavors to find the difference in time between the two devices, over a period of time. This reduces the reliance on constant, and on-going connection to a central clock. Further, in cases where methods such as NTP, or PTP may be preferred, our proposed solution can bolster those protocols.

Acknowledgements

I would like to extend my thanks to my supervisor, Dr. Amir Keyvan Khandani, for providing me with the opportunity to be part of the CST group, and for his support and guidance prior to, and throughout, my time as a graduate student I would also like to thank my readers, Drs. Omar Ramahi and Catherine Rosenberg, for their time, and dedication.

I would also like to thank Ms. Neda Mohammadizadeh and Ali Nik-Akhtari for their contributions, assistance, and guidance in all stages of my research.

This research would not have been possible, without the PlanetLab architecture in place. The many nodes and servers associated with the PlanetLab, be it in the University of Monash in Victoria, Australia, ETH servers in Zurich, Switzerland, or here at the University of Waterloo. As such, I wish to extend my appreciation to Dr Ian Goldberg as the administrator for the University of Waterloo's PlanetLab node.

Further, I wish to extend my thanks and gratitude to Drs. Vincent Gaudet, Slim Boumaiza, David Nairn, and Jessica Rossi, Susan Widdifield, Jackie Leach, Lisa Habel, and Karina Gaxiola, and the the reset of Electrical and Computer Engineering Graduate team,for their help, encouragement, and support from the very beginning to the very end of my graduate career. I also thank Dr. Jeff Casello, Rose Vogt, and Chris Read, for their care, and dedication to making graduate students' lives a little easier.

I am grateful for, and forever indebted to, my colleagues, friends, and family, for their support, encouragement, and their overall *da sein*, in every sense of the phrase. I wish to extend specials, and my very humble thanks to Maliheh Ahamdi, Mehrnaz Tabibi, Mozhgan Tabibi, Lily Tarba, Jamie Lyn Stevenson Waugh, Adam Daniel Gomes, Amalia Stavropoulou, Andrea Marrocco, Stanley Ituah, Joshua O'Connell, Wael Mohammad Ali, Eric Schwartz, Nicholas Pham, Nima Abbasi, John Rinehart, Yiling Xu, Alireza Hosseinzadeh, Seyi Gamu, and Laura McCrackin.

Dedication

I dedicate this thesis to my grandmother, my mother, my family, and my friends.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Time is defined to make motion look simple.

John Wheeler

Time, as a concept, is not very useful if not used in relation to another action. Absolute time measurements are not very useful on their own. When an individual asks another, "What time is it?", the answer would not be very useful, or even interesting, if the stated time was not associated with something else, a pending appointment, a measure of the duration of an activity undertaken, or even in relation to the passage of time in relation to the start of the day. Only when used in conjunction with external events do time measurements, and time in general, become pertinent to our everyday lives.

The act of *timekeeping*, or the measurement of the passage of time, itself is only of significance when taken in relation to another timekeeping event, method, or reference. In totality, it is the *passage* of time that is of interest to most individuals, not merely the series of numbers. In fact, the numbers that indicate the current time to most people, are only intelligible, because there is a common, universally agreed upon point of reference.

For example, while completing a form for a catering order in advance of an event, if the only response to the field for time of delivery is the notation *12:43*, it would be considered rather meaningless. There will undoubtedly be follow up questions: *A.M. or P.M.?*, *on what date?*, and perhaps even *where and in what time zone?* (the order might be for sushi from Japan). Similarly, *1544834997* is unintelligible and meaningless to the average individual, where it, in fact, tells you the exact date and time on a computer. The average human will not understand this, but a computer will. To a computer the above numbers mean *Friday, December 14, 2018 7:49:57 PM GMT-05:00*, because to a computer those digits indicate the number of elapsed seconds since midnight January 1, 1970, or *Epoch*, not counting leap seconds.

Another matter of interest is the method by which the time has been measured. Let us consider the catering order once more. If a catering order has indeed been placed for sushi from Japan, we are now dealing with two sources of time. The customer, having placed the order with a specified time of delivery, and the shop having accepted the order, will have a common expectation and understanding of the time of delivery. Though, what if the shopkeeper has an incorrect, or perhaps even a different, concept of time relative to the customer's? Once again, the expected delivery time and date would be quite meaningless if the shopkeeper's watch runs too slow, or if the customer's watch runs too fast.

Above examples may come across as rudimentary and uncomplicated. However, they illustrate the importance of good and proper time-keeping. Many standards and proposals have dealt with the nature of time synchronization including [21], [4] [29], [20], [27], [24], just to name a few. Network Time Protocol is one of the major standards developed to address the challenges involved with time synchronization. When the issue became more acute, and a need for more precise and specialized synchronization methods was called for, a hardware dependent method, the Precision Time Protocol (PTP), was introduced.

This thesis presents a method for time synchronization, which, especially when applied in specific and specialized settings, can be of great benefit. The method may also prove to be supplementary and may be used in conjunction with existing methods. The main contribution of this thesis is as follows. We have proposed a method that will essentially find the difference in time-progression between two computers with sufficient accuracy. This difference in time progression can then be used to predict the time in another computer or to retroactively correct the time on another computer. Devices with limited resources and placed in areas with no or poor connectivity to a reference clock, such as in distributed data-collection efforts, shopping malls, etc. can benefit greatly from this implementation.

Further, time-critical, network-dependent activities could also benefit from this. The measured difference time-progression can be used to interpolate another computer's local time at the time of the critical event, especially in conditions where time-synchronization using traditional means is not feasible, not possible, or not-accessible.

The remainder of this thesis is organized as follows:

Chapter 2 provides the necessary background on topics including time synchronization, and network time synchronization. It also delves into the nature of network synchronization. There is also a summary of the thought process behind the design of the Network Time Protocol. An overview of the pertinent information available in the literature is also provided.

Chapter 3 provides an overview of the processes used in the implementation of the required test. There is also a breakdown of the equipment used for this study.

Chapter 4 introduces the results gathered from the studies. The studies have been conducted using different equipment and computers, and the gathered results from each pertinent results are enclosed.

Chapter 5 summarizes the discoveries and expands on the results obtained from the studies. It also outlines the contributions achieved as a result of this study. It goes on to expand on the implications of these obtained results, and presents potential real-world implementation scenarios.

Finally, Chapter 6 discusses the requisite studies in order to form a complete protocol that has real-world application. It provides an outline for future areas of expansion.

Chapter 2

Background & Literature Review

Communication networks use a variety of tools and methods to ensure the accuracy and fidelity of the data. Such a process is achieved by various methods including, but not limited to, a variety of time synchronization techniques, and comparing and limiting the number of dropped packets. In this section, some pertinent methods and topics are introduced to provide the necessary background. First, some background information is provided, followed by a brief overview of the research done on this topic.

2.1 Background

A brief overview of Time Synchronization, dropped packets, and packet delay is covered in the following section.

2.1.1 Time Synchronization

Time synchronization is a vital component of computer networks. In order for various computers, and/or their dependent peripherals, to communicate with each other, particularly over a synchronous telecommunications network consisting of packet-switched technologies, the individual computers, and their respective peripherals need to ensure that they are synchronized to a common notion of time. Ensuring that all members of a computer network conform to, or at least understand, a uniform notion of time is even more important in any wireless network, for examples Wireless Sensory Network. *Sensor Networks* are specialized sensor networks that can be set up without an existing infrastructure, and as such need to communicate using a common notion of time [29].

Distributed networks are a collection of distinct entities which are spatially separated, and which communicate with one another through a communication network, such as exchanging messages [20]. A system is distributed if the message transmission delay is not negligible compared to the time between events in a single process [20]. Further, distributed systems tend not to have access to a central clock. Imagine hundreds of servers placed across the world and communicating with one another. It is, therefore, and quite frequently in other distributed systems, to obtain a common notion of time, where *time* can mean "either an approximation to real time or simply an integer-valued counter"[30]. Multitudes of real-time systems, from Google Docs' ability to maintain a revision control system to security softwares to simple communication mechanisms, rely on the ability to time-stamp events and determine what event took place first [30]. In a distributed system, it may sometimes be impossible to determine whether event a occurred prior to event b. The "relation '*happened before*' is therefore only a Partial ordering" of the events in a system [20].

2.1.1.1 Partial Ordering

As mentioned, if we say that b followed a, then it is believed that a occurred prior to b. If we are to say that the events in the system in which such events are defined have taken place in a correct fashion, then the events must be done in accordance with specifications that are observable within, and relative to, that system [20]. We can define these events free from the constraints of a strict definition of *real clock* and *physical time* since it is sufficient to identify events on their own, as real clocks are not necessarily accurate and may not keep precise physical time [20].

Let us assume that each system is a collection of processes, and each process consists of a sequence of events, regardless of how one defines processes or events. If a sends a message to b, one may define the entire act of sending a message, from the moment a request to send is made by a to the final act of b rendering the message as one process, and the intermediary actions as events. Conversely, we can continuously break down each action into subgroups to the point of triviality where, for instance, the interrupt request issued by a is considered an event. Regardless of how, or how finely an event is defined, the necessary assumption made is that "the events of a process form a sequence", where a is before b if a occurs before b [20]. In other words "a process is defined to be a set of events with an *a priori* total ordering" [20].

Lamport, in [20], defines the *happened before* relation, indicated using "$\rightarrow$", "on the set of events of a system as the smallest relation" which satisfies the following three conditions:

1. if a and b are events in the same process, and a a comes before b, then $a \rightarrow b$
2. If a is the sending of a message by one process and b is the receipt of the same message by another process, then $a \rightarrow b$.
3. If $a \rightarrow b$ and $b \rightarrow c$ then $a \rightarrow c$. Two distinct events a and b are said to be concurrent if $a \not\rightarrow b$ and $b \not\rightarrow a$.

It is further assumed that $a \not\rightarrow a$, for any event a, implying that"$\rightarrow$" is an irreflexive Partial ordering on the set of all events in a system.

Let us demonstrate with the aid of a *space-time* diagram in Fig. 2.1. Space is given in the horizontal direction, and time is given in the vertical direction, with the time progression indicated from the bottom to the top. Further note that the dots are events in a process (indicated by vertical lines) and messages are described using the curly lines intersecting the vertical lines (processes). In demonstrating the definition, as shown in Fig 2.1, messages can be sent from one process but can be received out of order on the receiving process, e.g. in $q_1 \rightarrow r_4$. Further, if $a \rightarrow b$ then a can causally affect b, and if one doesn't affect *causally* the other, then they can be considered concurrent. Therefore, q_3 and p_3 are concurrent. While q_3 occurs before p_3, since it's placed lower in the diagram, p_3 is not aware of q_3's actions and outcomes until it hears from Process Q at p_4.

2.1.1.2 Logical Clocks and Total Ordering

The notional of a clock need not complicate the discussion on order of precedence, and as previously mentioned, need not involve a discussion on physical time. Lamport in [20] states that it can simply be assumed that a clock is merely a means of indicating precedence between causally linked events; a numbering system for assigning which events precede which other events. Therefore, the numbers assigned can be the time at which an event takes place. Defined more formally, let us designate clock C_i for each process P_i. C_i here is a function that renders a numbers (or time) $C_i \langle a \rangle$ to any event a. Similarly, any function C which assigns a number for any event b, is defined as $C \langle b \rangle = C_j \langle b \rangle$ if for any b in any process P_j.

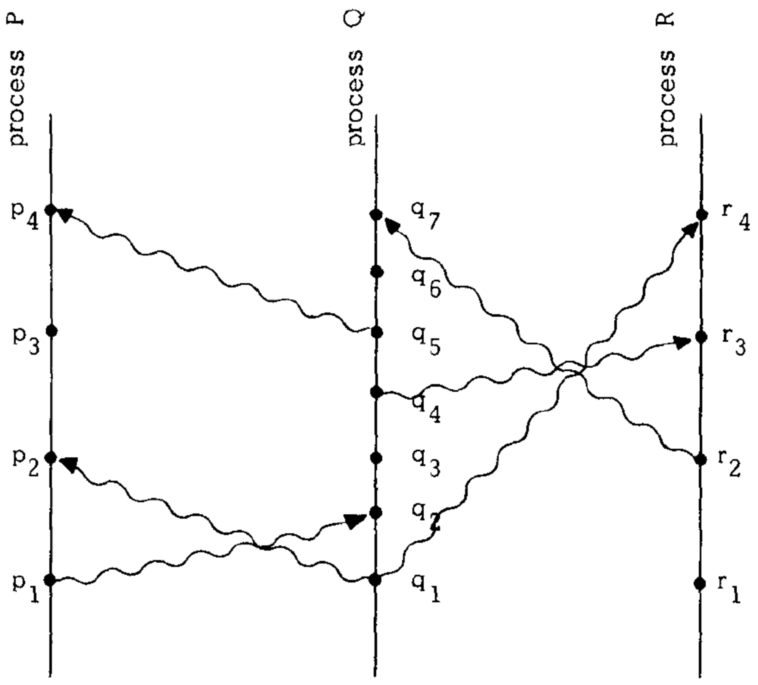


Figure 2.1: Example of Space-Time Diagram[20]

Further, it can naturally be stated that any event a comes before b, then a has occurred in the past, relative to b. Therefore the Clock Condition can be defined as:

Clock Condition. For any events a, b: if $a \to b$, then $C\langle a\rangle < C\langle b\rangle$.

We further need to stipulate two conditions in order to ensure that the Clock Condition is met.

C1. If a and b are events in Process P_i, and a comes before b, then $C_i\langle a\rangle < C_i\langle b\rangle$ [20].
C2. If a is the sending of a message by process P_i, and b is the receipt of that message by process P_j, then $C_i\langle a\rangle < C_j\langle b\rangle$ [20]

Now, by assuming that processes are algorithms, and events are actions taken, commanded, or executed by such algorithms, clocks can be introduced without violating the Clock Condition. Let C_i be the clock for process P_i, where it holds a value during event a representing $C_i\langle a\rangle$ [20]. Note that while C_i may change between events, a change in C_i

does not necessarily indicate a new event. in other words, a conceptual logical clock's time may progress without indicating a change in event status for a particular process. Two rules are proposed in order to ensure that C1 and C2 conditions are met. While ensuring that C1 is met requires no further introductions, meeting the C2 condition requires the introduction of a *timestamp* T_m, representing the time a message m is sent. Each m will contain said timestamp. As a result, the implementation rules are as follows:

IR1. Each process P_i increments C_i between any two successive events. [20].
IR2. a) if event a is the sending of a message m by process P_i then the message m contains a timestamp $T_m = C_i \langle a \rangle$. b)Upon receiving a message m, process P_j sets C_j greater than or equal to its present value and greater than T_m [20].

We can further expand the definition of Partial ordering. Ties and anomalies will undoubtedly occur, and in order to avoid such occurrences, a total ordering is defined, indicated by "$\Rightarrow$". The definition is as follows:
"if a, is an event in the process P_i and b is an event in the process P_j , then $a \Rightarrow b$ if and only if either i) $C_i \langle a \rangle < C_j \langle b \rangle$ or ii) $C_i \langle a \rangle = C_j \langle b \rangle$ and$P_i \prec P_j$" [20]. The "$\prec$" merely enables a priority system.

2.1.1.3 Physical Clock

Assume the following. If Person1 issues command COM1 on computer A to a central server, calls Person2 and instructs her to issue command COM2 on computer B to the same central server, it is possible for the command COM2 to reach the server, and be executed before COM1.

In introducing a *physical clock* Lamport[20] first defines a number of things:

- $\mathscr{I}$: set of all system events
- $\underline{\mathscr{I}}$: set of all events including $\mathscr{I}$, and all other relevant external events.
- $\rightharpoonup$: the "happened before" relation for all $\underline{\mathscr{I}}$. In the above example, the COM1$\rightharpoonup$ COM2, but COM1 $\not\rightarrow$ COM2

We can now define a Strong Clock Condition:

Strong Clock Condition. For any event a, b, in $\mathscr{I}$: if $a \rightharpoonup b$ then $C \langle a \rangle < C \langle b \rangle$.

Note that $\rightharpoonup$ is stronger than $\rightarrow$.

Using the above, we can define a *Physical Clock*, where assuming that $C_i(t)$ is a continuous differentiable function of t – with the exception of isolated discontinuities indicating a clock reset – $dC_i(t)/dt$ is the a rate at which the clock is running at time t [20].

Further, in order to ensure the accuracy of *Physical Clocks* lets us introduce two conditions necessary to guarantee accuracy:

> **PC1**. There exists a constant $\kappa << 1$ such that for all $i : |dC_i(t)/dt - 1| < \kappa$ [20].
> **PC2**. For all $i, j : |C_i(t) - C_j(t)| < \epsilon$ [20].

In other words, physical clocks need to run approximately at the same rate, i.e. $dC_i(t)/dt \approx 1$, and also need to be synchronized, meaning $C_i(t) \approx C_j(t)$.

The Implementation rules can be further specialized in order to address the concerns of the physical clocks.

> **IR1'**. For each i, if P_i, does not receive a message at physical time t, then C_i is differentiable at t, and $dC_i(t)/dt > 0$ [20].
> **IR2'**. a)if P_i sends a message m at physical time t, then m contains a timestamp $T_m = C_i(t)$. b)Upon receiving a message m at time t', process P_j sets $C_j(t')$ equal to maximum $(C_j(t' - 0.T_m + \mu_m)$[20],

where, μ_m is defined as the *minimum delay* of message m that was sent at time t and received at time *t'*. It is further stated that while the rules are formal specifications relating to physical time, each process is only required to know its own clock reading and the timestamp of the messages it has received[20].

Finally, Lamport offers the *Clock Synchronization* theorem. It is offered here verbatim from [20].

Theorem 2.1. *Assume a strongly connected graph of processes with diameter d which always obeys rules IR1' and IR2. Assume that for any message* $m, \mu_m \leq \mu$ *for some constant* μ*, and that for all* $t \geq t_0$*:*

a) PC1 holds

b) There are constants τ *and* ξ *such that every* τ *seconds a message with an* unpredictable delay *less than* ξ *is sent over every arc. Then PC2 is satisfied with* $\epsilon \approx d(2\kappa\tau + \xi)$ *for all* $t \gtrapprox t_0 + \tau d$*, where the approximations assume* $\mu + \xi \ll \tau$*.*

A message *total delay* is defined as $v_m = t' - t$, and *minimum delay* is defined as $\mu \leq 0$, s.t. $\mu_m \geq v_m$. Finally, *unpredictable delay* of a message, as used in the theorem, is defined as $\xi_m = v_m - \mu_m$.

2.2 The Network Time Protocol

As previously noted, there are several, commonly accepted methodologies used in clock and time synchronization. Network Time Protocol is one such method, and it is very prevalent in today's network architectures, and commonly used in most networking systems as a way of ensuring the fidelity of the time on several connected computer node [26].

2.2.1 Definitions

Let us define the elements used to measure, and evaluate the effectiveness and value of the protocol [24].

- *Stability*: Maintaining a constant frequency.
- *Accuracy*: How well a clock time compares with national standards.
- *Precision*: How finely time can be resolved in a particular timekeeping system.
- *Offset*: The time difference between two clocks.
- *Skew*: The frequency difference between two clocks.
- *Reliability*: The fraction of time a timekeeping system can be kept operating and connected in a network.
- *Synchronization subnet*: Series of servers capable of each measuring offset between their own local clocks neighboring servers.
- *Peer*: A neighboring server.
- *Time servers*: Timekeeping systems belonging to a synchronization subnet where local clocks are maintained.
- *Time Synchronization*: To set two cocks to agree at a particular epoch with respect to the Coordinated Universal Time.
- *Frequency Synchronization*: To adjust clocks on a subnet to run at the same frequency.
- *Clock synchronization*: To synchronize two clocks in both frequency and time.

2.2.2 Requirements & Approaches

The internet traffic may be susceptible to routing and load variations, making distributed synchronization a challenging task, since stable local-clock oscillators and multiple offset comparisons over relatively long periods are required in order to establish stable *frequency synchronization* [26]. Frequency synchronization is a particularly challenging task when compared with *time synchronization*, which requires resource redundancies and proper use of selection algorithms.

As such, a set of requirements are prescribed for the NTP protocol [26]. There should be a set of primary reference sources, such as clocks based on atomic clocks or various Global Navigation Satellite System (GNSS), and synchronized according to national standards, need should *continuously* deliver local time based on UTC. Time delivered based on GNSS will be discussed in greater detail. The *continuous* modifier is of note since one of the requirements also calls for the synchronization to occur without interruption in order for the client to receive updated information at a reliable rate to compensate for expected *wanders* in the local clock.

Further, the time servers are required to provide the time with accuracy and precision, regardless of delay variations [26]. The *synchronization subnet* must be "reliable and survival", regardless of network conditions. The software requirements of this protocol should not strain the resources of the host system at any level (minimal), and the software must be Portable. Lastly, the protocol should allow a mechanism for authentication and to prevent security breaches.

At the time of NTP's creation, there were several ways through which it was possible to disseminate and synchronize time to different systems. Using timecode receivers for each system requiring time synchronization is an option, though it is one of the rather more expensive options.

Building on this option, it is also possible to use timecode rebroadcast centers, where the dedicated FM or TV subcarriers rebroadcast the time. The U.S. National Institute of Standards and Technology (NIST) provides the time to public users through the telephone lines using the Automated Computer Time Service (ACTS). The system is only available to users who use an analog mode, which essentially renders the method outdated, and effectively unusable. The system is only capable of providing the date and general time of day. It should not, however, be used for precise measurements.

Further, multiple protocols have been proposed, tested, and used over the years, among them the Daytime Protocol, Time Protocol, ICMP Timestamp message, and IP Timestamp [26]. Unix based systems may have used the *timed daemon*, where a master measures the offsets of a number of slave hosts and sends periodic corrections to each slave [26] [8]. One of the important aspects of NTP is its compensation of effects of statistical delay variations in a network. The aforementioned methods do not have such a mechanism, and which makes precision time distribution in a network environment difficult. This would be in contravention of one of the requirements set out.

Mills in [26], in introducing the NTP, claims that accurate and reliable time synchronization in a network environment can only be achieved through an integrated design. It is further stated that the protocol is hierarchical, and made up of loosely coupled "time servers which exchange periodic update messages maintaining precision timestamps to adjust local oscillator phase and frequency" [26].

2.2.3 Time Standards

Coordinated Universal Time, *UTC*, and International Atomic Time, *TAI*, are two inseparable standards, which effectively make up the foundation of all units of time around the globe. They are also the pillars of NTP. In 1967 a second was defined as the period totaling 9,192,631,770 cycles of the radiation, which is the transition of two energy levels from the ground state of the Cesium 133 atom [4]. The TAI is calculated by tabulating a weighted average of more than 300 atomic clocks.

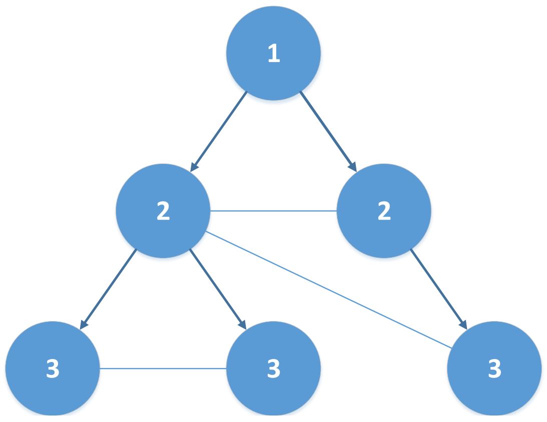
In 1970, the Coordinated Universal Time was designed, by an international group of scientists within the International Telecommunication Union. In 1972, adjustments were made to the clock so that at midnight on the first day of the year, TAI was exactly 10 seconds ahead of UTC. This was done to ensure that the time difference between the two was consistent, and accounted for in whole numbers (including accounting for leap seconds). Further, the UTC was changed so that the *tick* rate would be consistent with TAI.

The NTP data is based on time 0^h on January 1, 1900, without an assumption of prior leap seconds, making the start of UTC at midnight of January 1, 1972, equal to 2,272,060,8000.0 [26]. This is the NTP representation of time using 64-bit unsigned fixed-point numbers, with the first 32 bits indicating the integer part of time, and the last 32 representing the decimal part. The precision of this system is approximately 232 picoseconds,

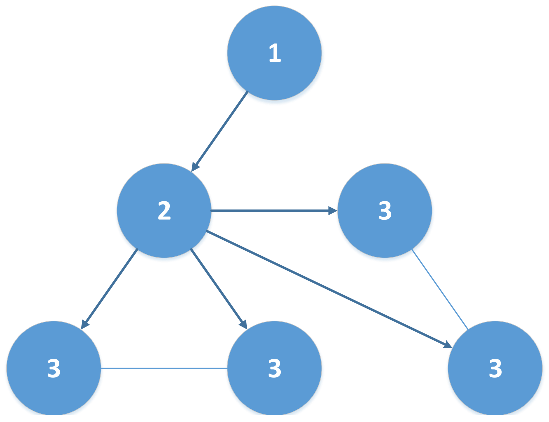
2.2.4 The Network Time Protocol

The system is composed of various primary and secondary time servers, connected to clients. There also exists various redundancies which allow multiple paths between each node, at different levels. Primary time servers are directly connected to, and synced with, primary time sources [26]. There exists a hierarchical configuration, whereby secondary servers derive their time from primary servers, and so forth, with decreasing accuracy as the node level increases with the primary time server at the root of a spanning tree. Each layer, in which the depth increases of the tree increases is called a stratum [25].

Using the existing timecode receivers, and architecture, with the use of propagation-delay corrections, synchronization with the primary server on the order of tens of microseconds can be achieved [25]. The distance metric in the synchronization subnet is determined by, first the stratum, then total round trip path delay to the root, *synchronization distance*, and a variant of the Bellman-Ford distributed routing to determine the minimum-weight spanning tree [26]. *Dispersion*, or the timekeeping quality of a peer, is calculated by a sun of weight offset differences. NTP requires that at least one, and preferably several primary servers be directly connected to external primary time sources. As mentioned, the primary sources are connected to secondary sources, and the secondary sources are connected to the synchronization subnets. Figure 2.2 shows an example of a 6 member subnet.



(a)



(b)

Figure 2.2: Subnet Synchronization

As can be seen in Figure 2.2a, a typical subnet of the NTP synchronization scheme. Each node is a server, with each stratum number indicating the number of hops to the external source. In Figure 2.2a, there are 3 strata present, and as a result, the 3rd stratum, for example, requires 3 hops before reaching the reference time. The darker lines in the figure indicate the synchronization path between the nodes. The lighter lines indicate active exchange of timing and synchronization information, though they are not used. These lines also indicate the possible backup paths.

In Figure 2.2b, we see the case where the connection between two nodes fails. The backup paths are activated. The paths are automatically reconfigured, and the required back up paths are activated. As can be seen, the right-side node 2 is demoted to a stratum 3, and the backup path between the left-side stratum 2, and the right-side stratum 3 is activated, allowing that node to remain a stratum 3, and not be demoted to a stratum 4.

Figures 2.3 and 2.4 show the the organization of the NTP time-server model, as well as the measurement of delay and offset. These topics are discussed in the following section 2.2.4.1

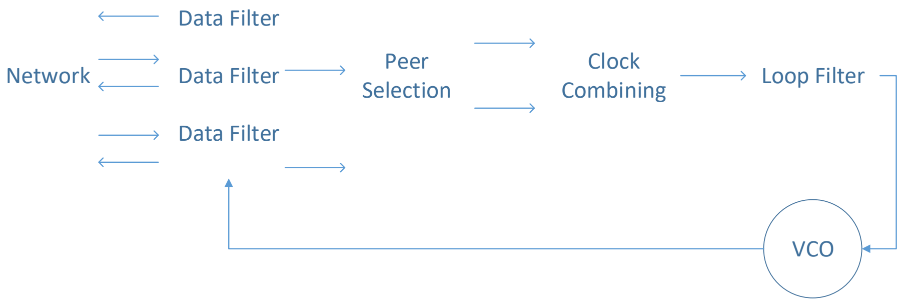


Figure 2.3: Network Time Protocol Flowchart [26]

2.2.4.1 Time and Frequency

A series of packets containing time-stamps are exchange between he various nodes, in order to establish accuracy, error estimates, round-trip delays, and clock offsets [26]. Figure 2.3 shows the overall behavior of the NTP time-server model [26]. How these calculations are determined are as follows. Let there be two nodes A and B.

Let $T_i, T_{i-1}, T_{i-2}, T_{i-3}$, be the time-stamps for the two nodes, and $a = T_{i-2} - T_{i-3}$, $b = T_{i-1} - T_1$ which represent the delay in the travel path and the return path. As a result we have:

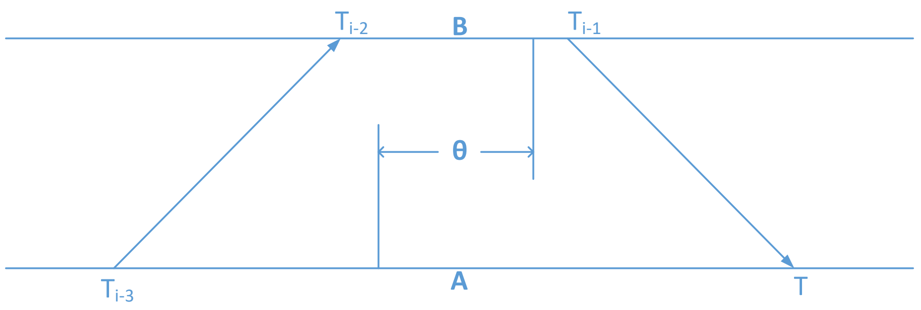


Figure 2.4: Measuring Delay and Offset [26]

$$\delta_i = a - b \text{ and } \theta_i = \frac{(a+b)}{2},$$

where δ_i is defined as the round-trip delay, and θ_i is the offset of the two nodes. As seen in Figure 2.4, θ is true offset, whose calculation is not difficult. If $a \equiv x + \theta$, where x, is a positive value indicating the delay transmission delay between two nodes, where a transmission delay is measured from the moment a packet is transmitted by a node, and received by the other, the relationship between b, and θ will similarly need to be defined as $b \leq \theta$. An inequality, then, can be defined as follows:

$$b = \frac{a+b}{2} - \frac{a-b}{2} \leq \theta \leq \frac{a+b}{2} + \frac{a-b}{2} = a,$$

which, when simplified, is

$$\theta_i - \frac{\theta_i}{2} \leq \theta_i + \frac{\theta_i}{2}.$$

Put another way, "the true clock offset must lie in the interval of size equal to the measured delay and centered about the measured offset"[26].

The NTP messages are received on all nodes containing $T_{i-1}, T_{i-2}, T_{i-2}$ while the T_i is always readily available as the packet capture time. This way all members of the subnet can calculate and verify delay and offset measurements, independently. This allows the architecture to be unconcerned with the order of packet arrivals (all the necessary information are readily available in the packets). The system is also, to a degree, unconcerned with reliable delivery [26]. In the older version, the data filters processed the delays and offsets to increase accuracy and reduce timing noise, while the updated protocol allows clusters to establish these values [26][25]. The filtering and selection algorithms allow the PLL to manage the required phase-corrections. These are processed by the loop filter to control the local clock, acting here as a Voltage-Controlled Oscillator, *VCO*. It is the VCO that provides the phase and time references necessary, to produce the required time-stamps.

2.2.4.2 Modes of Operation

The protocol entities can operate in five general modes of operation, and three service classes. These classes are *Multicast*, *Procedure-call*, and *Symmetric*. The modes of operation are *Symmetric Active*, *Symmetric Passive*, *Client*, *Server*, *Broadcast* [25]. The classes vary based on the number of peers, the direction of synchronization, and retention of state information [26].

In *multicast* class allows for high-speed networks, with many clients attached. It is also intended to be used in an environment where accuracy is not the highest priority. In the *multicast* mode, the server announces that it is willing to provide synchronization to many, but refuses to be synchronized by others, and therefore blocks incoming NTP messages. The computer willing to provide synchronization announces its readiness to provide synchronization, and other computers in client mode can determine to evaluate, and select their source, and receive NTP packets from this source.

In environments where accuracy is of the highest priority, including the use of file-servers, the *procedure-call* class is used. This class is also used in the case of *multicast* failure. This class increases the priority of the synchronization process, as the client sends an NTP request packet to a peer, and the server then returns the message with the necessary time-stamps. Again, a client is indicating that it will be accepting synchronization messages, while a server only provides synchronization but blocks NTP synchronization messages from others.

The *symmetric* class allows for simultaneous synchronization, i.e. receiving and providing synchronization messages from various peers, in accordance with the peer-selection algorithm. The *symmetric active* is intended for high-stratum nodes in the subnet, whereas the *symmetric passive* mode is for low-stratum nodes, with many peers associated with it. The *association* – a loosely coupled connection – is dissolved if the sending node doesn't hear back from the other node, however, the association remains if the sender receives at least one reply message from the recipient [25].

There also exist protocol state variables whose purpose is to preserve time-stamps data.

2.2.4.3 State Variables and Procedures

In the symmetric modes, state variables are persistent and are maintained for each association [26]. In modes other than ones in the symmetric class, the state variables are destroyed as they are not necessary: the state variables are kept until a reply has been sent.

Table 2.1 provides a visual representation of the NTP packet-header. While a complete description of of each field is provided in protocol standard in [25], select fields are described below as outlined in [26], [24], and [25].

Table 2.1: NTP Packet Header [25]

L1	VN	Mode	Stratum	Poll	Precision
Root Delay					
Root Dispersion					
Reference ID					
Reference Timestamp (64 bits)					
Originate Timestamp (64 bits)					
Receive Timestamp (64 bits)					
Transmit Timestamp					
Extension Field 1 (variable)					
Extension Field 2 (variable)					
Key Identifier					
dgst (128)					

Mode, Stratum, Precision. Indicate current operation mode stratum and local clock Precision.

Poll Interval. Maximum interval between NTP messages.

Reference Cock Identifier, Reference Time-stamp. Identifies the type of reference clock, and time of the last update.

Originate Time-stamp. The peer time the last NTP message was originated, copied from the transmit time-stamp field or (T_{i-3}).

Receive Time-stamp. The local time when the latest NTP message was received, or T_{i-1}.

Transmit Time-stamp. The local time when NTP message was transmitted or T_{i-1}.

Root Delay, Root Dispersion. Total round-trip delay and dispersion, respectively, to the reference clock.

The protocol machine maintains the state variables for the above fields, and other state variables such as Filter Register, used by the data-filtering algorithm, Peer time, used to control intervals, and Synchronization Source, among other things.

There are events of note following the expiration of the peer timer. Operator command, or detected system fault, and NTP message transmission are among events that occur following the peer timer expiration. For instance, when the peer timer reaches zero the transmit procedure is initiated, where the peer timer is reset, and the NTP message is sent [26].

When an NTP message is received, the receive procedure is initiated. The association is determined, and established, which may result in a persistent association or a transient one depending on the mode. The raw round-trip delay and clock offset are calculated. Following a series of calculations, and the best peer in a sequence of raw samples is determined using a weighted voting procedure, along with an error estimator, otherwise known as *filter dispersion* [26].

There also exists an update procedure, which is called when a new set of estimates become available [26]. Again, similar to before, a weighted voting procedure determines the best peer. If the best peer is new or has an improved accuracy, a new synchronization source may be recognized. The new error estimator is called *select dispersion.* If there exists a large difference between the local clock and that the time received from the synchronization source, the local clock is reset, all timing information is removed, and the local PC re-selects the synchronization source if required [26].

2.2.5 Filtering and Selection Algorithms

Filtering and selection algorithms are integral to the NTP protocol, which acts to improve the accuracy of estimated delays and offsets between the servers. The complexity of the algorithms depends on the required accuracy, precision, and the statistical properties of the transmission path [24]. It is well known that when dealing with the internet "something somewhere in the system is broken at any given time". There is also virtually no way to guarantee that the connection between one standard location to another will involve a direct path, and a connection will involve multiple hops that can widely vary from one transmission to the next. Hence the algorithms are not one-size-fits-all, and similarly cannot be designed for any particular travel path.

There also exist *convergence* algorithms 'that reduce the statistical outliers, where NTP data-filtering algorithm is an example. Then there are *consistency* algorithms, such as the peer-selection algorithm, whose task it is to find trusted clock subnet.

In the data-filtering algorithm, assuming that the skew between the server and peer clocks is relatively small, we let (δ, θ), where $\delta \text{and} \theta$, represent round-trip delay, and clock offset, respectively, when the path is relatively idle. There is no assumption about the delay distribution due to packet queuing. It is our intention to find an estimator from a sample population over a path with a predetermined interval under normal traffic conditions [26]. The estimator and sample populations are represented as $(\hat{\delta}, \hat{\theta})$and, (δ_i, θ_i), respectively.

Described in [26], observations of packet-switching networks indicated that operations tend to be below the knew of the throughput-delay curve, meaning packet queues are "small with mostly infrequent surges". Further, the probability that packet find a busy path is low due to routing algorithms, and the probability that packets find a busy path in two directions even lower. An observation can be made that the best offset samples can be collected with the lowest delays. By designing a *minimum filter*, where n most recent samples are selected, namely $(\delta_i, \theta_i), ..., (\delta_{i-n+1}, \theta_{i-n+1})$. As a result, the lowest delay sample the estimator.

The minimum filter was tested in various conditions, and using several paths, and it consistently produced lower errors, and, most importantly, the filter lowered maximum error in high network traffic conditions.

2.2.6 Summary

The principles of the NTP design can be summarized as follows, based on the topics covered [26]. There is a hierarchical structure which, based on factors such as precision, accuracy, and robustness, creates a method used for synchronizing multiple computers. The protocol uses connection-less modes (namely User Datagram Protocol (UDP)), to minimize latencies. There exists a method for synchronization that is tolerant of inherent in internet traffic such as dropped packets, and packet re-ordering, among other things, an uses algorithms based on *maximum-likelihood principles*, for selection, and source filtering. There exist redundancies in order to ensure there are back paths, clocks, and sources in the event failures, and selection algorithm for the selection of said sources for reliability and accuracy using a weighted-voting process.

Chapter 3

Methodology

The following is a discussion on the methodologies, and test patterns, and test equipment used for this thesis.

3.1 Planet Lab

The PlanetLab, operated under the PlanetLab Consortium, is a collection of academic, industrial, and government institutions cooperating to support and enhance the PlanetLab overlay network [1]. The Planet-Lab system is made up of a series of computers and servers, located in various locations around the world, which allow researchers to be connected, and have access, to computer systems around the world. This empowers these researchers to conduct live, long-distance experiments, in a real-world environment over the internet. This allows the researchers to observe how their intended, or envisioned, theories and experiments behave in real-time outside of a controlled lab environment, and without the use of simulators.

Further, the motivation for this initiative was to conduct *planetary-scale* tests on network traffic, and also to allow it to be done by various entities as a *community*. It is important to emphasize the community aspect of this, as it allows any member entity to conduct as many experiments necessary, and as extensively as desired, throughout the network. This allows for truly planetary-scale experiments to be done on a system that is nearly fully customizable, and virtually private.

The Planet-Lab network is motivated by, and based on three overarching organizational principles: Distributed Virtualization, Unbundled Management, Chain of Responsibility

[28].

A brief description of each description is as follows:

- *Unbundled Management:*: There are servers that conform to a set of minimum guidelines, but which are operated independently of one another, by the various participating entities.
- *Chain of Responsibility:* The PlanetLab Consortium does not directly own anything, neither the servers, the participants, or the nodes. However, there needs to be a chain of succession, through which, when something goes wrong, errors can be managed, and corrective action taken. Hence, for each *slice* there are *Principal Investigators*, who are ultimately responsible for the researchers working for them, and whose conduct is indirectly supervised by individuals by node *owners*.
- *Distributed Virtualization:* The researchers benefit from the ability to have "multiple points-of-presence". However, in order to prevent interference between the services run by each research group, each group is assigned a *slice*, whose activities are siloed from others'. When interacting with other *nodes*, each slice is able to connect to their own virtual machine, without impacting others' activities.

The PlanetLab structure is very simple. Here we define and outline the structure and major components of the PlanetLab. The topics covered will be pertinent, and limited to, the user-facing environment, and as such, topics and capabilities such as *slice creation service*, or the *node management service*, will not be discussed, though it bears mentioning that these topics are essential to the proper conduct of the PlanetLab environment as a whole, and the adequate and proper function of the various nodes and services outlined hereinafter.

3.1.1 Architectural Components

A *slice* is a layer of abstraction, allowing the creation of a *Virtual machine*. Each *member* of the PlanetLab – where the term member, narrowly defined in this context, is an individual, research group, lab, and/or organization, undertaking a research initiative whose works can collectively be thought of as one isolated entity, and without the need for segregated access for the sub-divided constituents – can only gain access to the PlanetLab network through the creation of a *slice*.

Essentially, a slice is an account, through which one may gain entry to the PlanetLab network. Individual researchers participating on the same topic can each have their own individual accounts on the PlanetLab network, however, they will gain access through one. Think of a slice as a shared lab computer, where every account holder has administrative rights; the computer access is restricted to the account holder, however, once access is granted, the account holders can view, edit, modify, and delete each others' files. Slice creations need to be approved by the Principal Investigator. The research outlined in this thesis was conducted under *uwaterloocst*.

The PlanetLab is made up of *node*, which are one or more servers, capable of hosting one or more *Virtual machine*. Nodes are located at *sites*, which are the physical location of where nodes are. There is no need to delve too deeply into the network requirements of a node, however, suffice to say that there needs to be a way for external clients to reach the node. The node can be just one computer, or a cluster of computers, capable of routing traffic, with enough available physical memory to host a number of virtual machines. The nodes need also be able to store a prescribed amount of data on their hard-drives. Since each slice will need to be able to store its data, and develop, run and install applications on each node, the nodes must have enough space allocated for each slice that resides on it. In reality, there is enough space to upload, and

Since each slice on the node needs isolated access to the node, therefore, as previously stated, the node needs to be capable of instantiating a session by creating a Virtual machine on said machine. Virtual machines are the execution environment in which slices are run in a node [28]. As outlined, Virtual machines are isolate from each other such that:

- the resources used by one Virtual machine does not impact the performance of another Virtual machine
- one Virtual machine cannot eavesdrop on the activities of another Virtual machine.
- one Virtual machine does not have access to the data of another Virtual machine.

Users are able to remotely access the virtual machine and have the ability to run programs, scripts, etc. when logging in. The isolation of the virtual machines, allows the various machines to, as mentioned, create, install, execute, and remove packages and programs, as necessary, without interfering with others'. However, it should be noted that the requirement that there exists an ability to isolate two slices – or two virtual machines, rather – from each other, is not synonymous with blocking all access between the virtual machines. The virtual machines may, in certain scenarios, need to be able to connect to another, a capability that may be provided, if required. This ability extends to all areas of resource allocation, including dynamic memory, and allocated processing power, or network bandwidth.

Each virtual machine is capable of using, managing, and maintaining the required resources of a particular slice, and extend the limitations on a particular slice when necessary, while also ensuring that unused resources are reclaimed, and properly distributed to other slices. On the servers used by during our experiments, for example, it is shown that

```
    /proc/sys/vm/overcommit_memory: 0
    /proc/sys/vm/overcommit_ratio: 50
```

indicating that it is able to commit up to 50% more memory if necessary. However since the `overcommit_memory` options is set to `0` the kernel is able to allocate memory as it sees fit. However, if the virtual machine `overcommit_ratio` option was set to `2`, for example, the memory overcommit ability would have been capped at `50%`.

Each node's capabilities differ from another's. This is evident by how much Random Access Memory is available to our slice on different nodes. For example, on University of Waterloo's own PlanetLab node, *plink*, the *uwaterloo_cst* slice was allocated 4,146,704 KiB, or approximately 4 GB, of physical memory. By contrast, the University of ETH Zürich's *planet2* node, provided our slice with 15,839,832 KiB, or approximately, 16 GB of physical memory.

The virtual machines are implemented using Linux-Vserver, which provides virtualization capability for GNU/Linux-based system, by providing isolation at the kernel-level. The users can access the nodes by connecting via a UNIX shell. Each user must have an SSH key pair in order to gain authenticated access. The keys must be in OpenSSH format. The users upload their public keys to their respective slices on the PlanetLab website, and can subsequently gain authenticated access to the node UNIX shell.

3.1.2 PlanetLab Equipment

The list of nodes affiliated with the *uwaterloo_cst* are as follows:

- University of Waterloo:
  plink.cs.uwaterloo.ca

- Swiss Federal Institute of Technology in Zurich (ETH Zurich):
  planetlab2.inf.ethz.ch

- University of Nevada, Reno (UNR):
  planetlab1.unr.edu

- University of Oregon:
  planetlab1.cs.uoregon.edu
  planetlab2.cs.uoregon.edu
  planetlab3.cs.uoregon.edu
  planetlab4.cs.uoregon.edu

- Shanghai Jiao Tong University
  planetlab-1.sjtu.edu.cn
  planetlab-2.sjtu.edu.cn

- National Education and Research Network (RNP):
  planetlab2.pop-pa.rnp.br

- Information Sciences Institute, University of Southern California (Postel USC):
  planetlab4.postel.org

- Monash University:
  pl1.eng.monash.edu.au

It is important to note that there are over 900 nodes presently available on the PlanetLab's Nodes directory. However, not all of these nodes are available all the time (vindicating Mill's claim that "something somewhere in the system is broken at any given time" [26]). The nodes may fail, be taken offline for maintenance, or simply be unreachable for a variety of other reasons. Unfortunately, and as a result, the capabilities of the servers, as outlined below, are limited to the ones that were reachable at the time of writing this thesis.

On average, all servers which were used for test purposes provided the *uwaterloo_cst* slice with 10 GB of disk quota (hard drive space).

The University of Waterloo's *plink* servers use the 64-bit Intel Xeon Processor E5420s, which have 4 cores, with a base frequency of 2.50 GHz, and 12 MB of L2 cache. As mentioned, *plink* also provide $\sim$ 4 GB of physical memory.

The ETH Zurich's *planetlab2* servers, in comparison, provide up to $\sim$ 16 GB of physical memory. The ETH Zurich servers run on 64-bit Intel Xeon Processor E5620s, which have 4 cores and are capable of multithreading, allowing a total of 8 threads per core. The E5620s have a base frequency of 2.40 GHz, with a maximum Turbo Frequency of 2.66 GHz, 12 MB of SmartCache.

The University of Oregon's *planetlab1-4* servers run with approximately 4 GB of physical memory. These nodes utilize the Intel Xeon Processor X3210s, which have a base frequency of 2.13 GHz with 4 cores, and 8 MB of L2 cache. These processors are not capable of hyper-threading. They have 64-bit instruction sets.

The Shanghai Jiao Tong University's *planetlab-1&2* nodes offer approximately 4 GB of physical memory. The servers run on Intel Xeon Processor E5405s, which are similar to the E5420s, and lie on the same socket type. The E5405s have a base frequency of 2.00 GHz, with 12 MB of L2 cache. They have 4 cores, utilize a capable of 64-bit instruction set, and do not support hyper-threading, or Turbo Boost.

The Monash University's *pl1* servers are powered by Intel Xeon Processor X3330s, with 6 MB of L2 cache. Running at 2.66 GHz base frequency, the X3330s have a 64-bit instruction set, and offer 4 Cores, without hyper-threading technology.

USC's *planetlab4* node offers approximately 4 GB of physical memory. The processor used is 64-bit Intel Core2 Duo Processor E6550. It has 2.33 GHz of processors base frequency, with a 4 MB L2 cache, and 2 cores. It is unable to do perform hyper-threading.

Finally, the RNP's *planetlab2* servers provide the users with nearly 4 GBs of physical memory. The node uses 64-bit Intel Xeon Processor X3323, which has a base frequency of 2.50 GHz. The processor has 6 MBs of L2 cache. The processors do not provide hyper-threading.

Unfortunately, the UNR servers, which were heavily used in the tests performed during our research, were unreachable at the time of the writing of this document. All the node, with the exception of the Shanghai Jiao Tong University's node, had a skeleton, or "thin" version of Fedora Werewolf, release version 8. The kernel is Linux version *2.6.32-20.planetlab.i686.* The Jiao Tong node runs on CentOS, release 6.10. The kernel is version *2.6.32-131.vs230.web10027.xidmask.2.mlab.i686.*

3.2 Lab Equipment

The computers on the originating end, i.e. the Test PCs, consist of 4 desktop PCs. The PCs are named as follows: *aklinux4.eng, aklinux5.eng, akkpc8.eng, akkpc9.eng.* Henceforth, these computers, when identification is required, will be referred to as Test PCs[1-4].

All four computers have the Intel Core2 Duo Processor E6850 as their respective Central Processing Units (CPU). These processors run at 3.00 GHz base frequency, with 4 MB L2 cache. The processors have 2 cores. They have 64-bit instruction sets. They are not able to use hyper-threading,

All computers have ASUSTek *P5K-VM* motherboards. The computers have 4 GiB of physical memory (using 4 memory modules each). The physical memory is clocked at 667MHz, and are 64-bit addressable. Each computer is equipped with two Network Interface Components. One network interface is an onboard Marvell 88E8056 Gigabit Ethernet Controller with Integrated PHY. The network card supports up to 1 Gbits/s data rate, and is clocked at 33 MHz.

The second network interface is the 82574L Gigabit Network Connection manufactured by Intel. The network card is connected via the Peripheral Component Interconnect – due to Motherboard limitations, the Peripheral Component Interconnect-Express variant could not be used. This compromise did not come at the cost of compromised results. This network card supports a data rate of up to 1 Gbits/s. These network cards are IEEE 1588 (also known as Precision Time Protocol) compliant. This enables us to capture the most accurate time possible when sending and receiving data packets. The Precision Time Protocol also provides sub-microseconds accuracy. While regular network cards allow only an accuracy within microseconds (the PlanetLab servers are accurate to only microseconds), this protocol allows synchronization to within hundreds, if not tens of nanoseconds.

The computers have Ubuntu version 17.10 (Artful Aardvark) as their operating system. The kernel is Linux version 4.13.0-46-generic. The *gcc* version on these computers was *gcc* version 7.2.0.

Additionally, a fifth computer is used that is not similar to the other four computers. This is done in order to ensure that we gain a comprehensive and broad understanding of the results. This PC will henceforth be referred to as Test PC5. The Test PC5 has an ASUSTek *CM6731_CM6431_CM6331* motherboard. The Test PC has an Intel Core i7-3770 Processor as its Central Processing Unit, with 4 cores. The processor has a base frequency of 3.40 GHz, with a maximum Turbo frequency of 3.90 GHz, with 8 MBs of SmartCache. The processor has is Intel Turbo Boost Technology 2.0 enabled, and is capable of Hyper-Threading. It also has 64-bit instruction set.

The computer has 8 GiB of physical memory, using 2 memory modules. The physical memory is clocked at 1600MHz, and are 64-bit addressable. Again the computer has two Network Interface Components. One network interface is the onboard RealTek RTL8111 PCI Express Gigabit Controller. The Network card supports up to 1 Gbits/s clocked at 33 MHz. The other network card is again the 82574L Gigabit Network Connection manufactured by Intel, as described in the description for the Test PCs[1-4], above. The operating system, kernel version, and the *gcc* versions are also the same as outlined above.

The summary of the above specifications is provided in 3.1.

3.3 Test-bed Set-up

The computers are connected to the University of Waterloo's network, as administered by the Information Systems and Technology services, via standard Cat5 or Cat6 Ethernet (patch) cables. Each Ethernet cable is no more than 30 feet (∼<10 meters). Due to the packet travel distance between our Devices Under Test (the test computers), and the intended destinations, situated all around the world, the length of the Ethernet cables are negligible.

The computers are assigned MAC address specific static IPs. The computers connect to the outside computers by passing transmitting packets within through the internal intranet. There are about 6-7 hops (typically routers) between each test computer, and the first external *traceroute* hop. The average packet round trip delay time, according to the *traceroute*, 8.596 ms, however, it is possible for the round trip delay figure to be much higher, or much lower at the extrema.

Traceroute is a Linux application which traces the number of *hops* a packet travels before reaching a destination. Each *hop*, represents a Network Layer (layer 3) endpoint, such as a router, or a firewall. As a result, the Data Link Layer (layer 2) stages according to the Open Systems Interconnection (OSI) model – the Link Layer in TCP/IP model– do not show up on *traceroute*. Switches are an example of layer 2 devices. There are examples of high-performance switches which act as layer 3 devices and can be discovered. Also, there are certain vendors that allow MAC address probing for intermediary devices, which can, in fact, discover so-called *transparent* devices. However, these tend to only be applicable to proprietary, and vendor-specific devices.

3.3.1 Method 1

Fig. 3.1 shows the overall description and layout of the test-bed. As can be seen in the figure, the PCs are the starting and the finishing nodes, and traffic is routed through one or multiple nodes, before being sent back to one of the local Test PCs. The intermediary nodes can be one, or multiple PlanetLab nodes. The purpose of the multiple nodes was to ensure that the packets had traveled a substantial distance before having been routed back. This packet-exchange may have happened in three possible ways: *PC 1* to *PC 2*, *PC 2* to *PC 1*, or simultaneous exchange of packets between *PC 1* and *PC 2*.

Table 3.1: Summary of the Specification of Test Computers

Criteria	**Test PC1**	**Test PC2**	**Test PC3**	**Test PC4**	**Test PC5**
Mother Board	P5K-VM	P5K-VM	P5K-VM	P5K-VM	CM6731_CM6431_CM6331
CPU Model	E6850	E6850	E6850	E6850	i7-3770
No. of Cores	2	2	2	2	4
Base Frequency	3.00 GHz	3.00 GHz	3.00 GHz	3.00 GHz	3.40 GHz
Max Frequency	-	-	-	-	3.90 GHz
HT Capable	No	No	No	No	Yes
Instruction Set	64-bit	64-bit	64-bit	64-bit	
Cache	4 MB L2	4 MB L2	4 MB L2	4 MB L2	8MB SmartCache
Physical Memory	4 GiB	4 GiB	4 GiB	4 GiB	8 GiB
Clock Speed	667MHz	667MHz	667MHz	667MHz	1600MHz
NIC_1 Model	Marvell 88E8056	Marvell 88E8056	Marvell 88E8056	Marvell 88E8056	RealTek RTL8111
Capacity	1 Gbits/s	1 Gbits/s	1 Gbits/s	1 Gbits/s	1 Gbits/s
Clock Speed	33 MHz	33 MHz	33 MHz	33 MHz	33 MHz
Intel 1588 Enabled	No	No	No	No	No
NIC_2 Model	Intel 82574L	Intel 82574L	Intel 82574L	Intel 82574L	Intel 82574L
Capacity	1 Gbits/s	1 Gbits/s	1 Gbits/s	1 Gbits/s	1 Gbits/s
Clock Speed	33 MHz	33 MHz	33 MHz	33 MHz	33 MHz
Intel 1588 Enabled	Yes	Yes	Yes	Yes	Yes

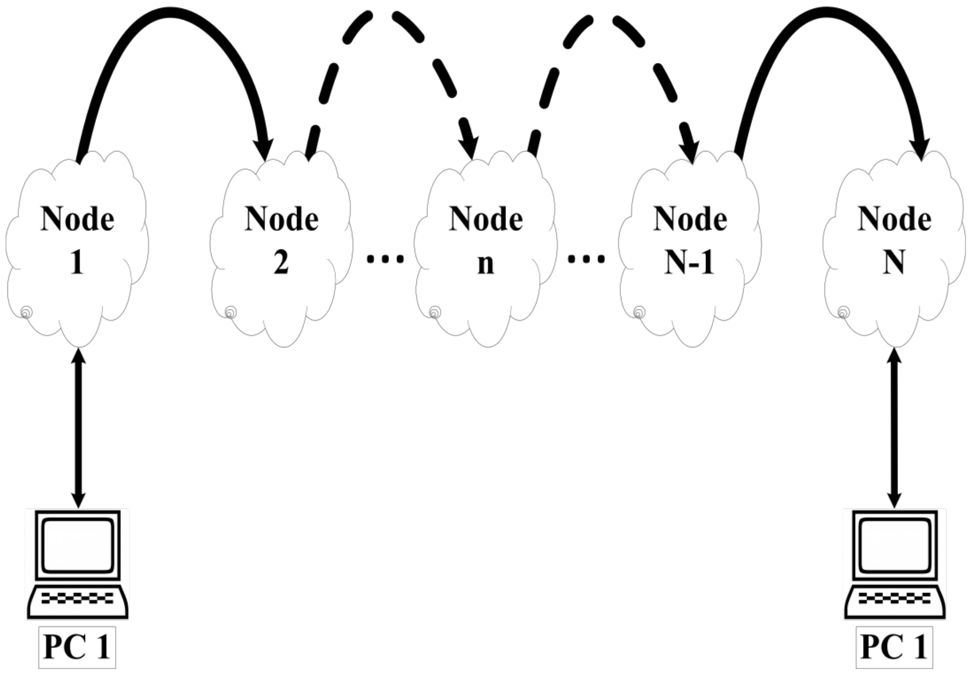


Figure 3.1: Test Environment Setup

The tests involved in this setup, included, but were not limited to, exchange of packets with random or predetermined Payloads (data), and varied payload sizes. The predetermined payloads typically involved a series of incrementally increasing packet identifiers in order to ensure that the sent packets had not been reordered upon receipt.

3.3.2 Method 2

The Method 2 is similar to the previous method, as outlined in 3.3.1. One of the local Test PCs begins to send packets to one of the PlanetLab servers. The packets are sent through multiple intermediary nodes. The destination of these data packets is one of the PlanetLab servers. At the same time, the destination PC is also initiating the transmission of similarly formed packets and sending them to the same local Test PC.

The payloads, similar to the one outlined in 3.3.1, can be varied, or predetermined, and used to either make the packets easily identifiable and to remove ambiguity, or can be random packets simply use to vary the size of the packets.

The purpose of this method was to see whether simultaneous traffic can significantly impair the transmission process, in the form of dropped or delayed packets. Further, it was also envisioned to measure how, and in what manner the travel time would be different depending on the source of the transmission.

3.3.3 Method 3

The most simple test is perhaps the most crucial of the tests. This method does not involve any external routes and is very simple in terms of the transmission set up. A Test PC is connected to another Test PC via an Ethernet Crossover cable. The method of connection is not very important. The crossover cable does not play a significant role. In fact, depending on the type of Network Interface Controller, the network card may be able to handle direct connection between two computers itself using a simple Ethernet cable. The intention is to connect two computers together, directly.

This enables us to measure how the time difference, and/or the delay between the packets change.

3.3.4 Method 4

Understanding time differences and time synchronization is crucial. As a result, one of the tests conducted included two sets of computers. One set consists of two similar test computers, in terms of processing power, memory, and networking capabilities. The other set consists of two personal computers that are not similar in processing power. The point of this method is to ensure that the obtained results were not due to the similarity in the test equipment. Put another way, it is important to verify that the obtained results are reproducible, consistent, and transferable.

By *transferable* here, it is meant that the individual results cannot be result dependant. Not to exhaust the explanation, though it is crucial to point out that the achieved result should, ideally, be reproducible not only in an academic setting, but a real-world setting, and should be commercially applicable.

Chapter 4

Results & Findings

This chapter presents the findings of selective experiments discussed in 3. The main objective has been to study and find the most important aspect of the time phenomenon in network communication, and whether we are able to establish relative time accuracy without the usage of a central clock.

Additional findings are presented in the A, however, they were discounted from the discussions in this thesis, since they were not thought to be central to the narrative presented here.

4.1 Delay over PlanetLab Network

The delay times between various test computers were studied and observed for many months. Presented in Figures 4.1 and 4.2 are only but a selection of a rich collection of delay times collected. Beginning with Figure 4.1a, various packets of various sizes are sent from a local test computer, over the PlanetLab network, and received again at one of the local Test PCs. The path of the majority of the figures demonstrated here are as follows:

$$ak8 \rightarrow plink.cs.uwaterloo.ca \rightarrow planetlab2.inf.ethz.ch \rightarrow plink.cs.uwaterloo.ca \rightarrow ak5,$$

or the reverse of this path.

As can be seen, these tests are done at various times, and different days. Further, it is also observed that the delay times tend to vary depending on the day, or time.

Also of interest is the delay times through these PlanetLab servers. Figures 4.2f and 4.2e show the delay times between the ETH servers in Zurich, and Test PC5. Approximately ten thousand (10000) packets are exchanged between the computer and the server every 10 minutes over 2 days. The two devices simultaneously begin sending packets to the other, where it is received at the point of destination, and its time of arrival is registered. The difference between these two clocks gives us the delay. As is evident from the figure, the delay times tend to change greatly. It further goes to solidify the claim that "errors due to stochastic network delays dominate; however, it is not usually possible to characterize network delays as a stationary random process since network queues can grow and shrink in a chaotic fashion and arriving customer traffic is frequently bursty. [26]."

4.2 Establishing Time Accuracy using Direct Connection

Presented here are the findings where the two computers are connected directly using a cross-over cable without the assistance of a switch, hub, or router. The following subsections will discuss the results between two computers of similar specifications, and dissimilar specifications.

The figures shown are representative of the average of the difference between the latest sample point in time and the first capture point, over the span of each sample point.

If M is the collection of all the times sent by the *master* to the *slave*, and S is the collection of all the corresponding local times on the *slave*, then

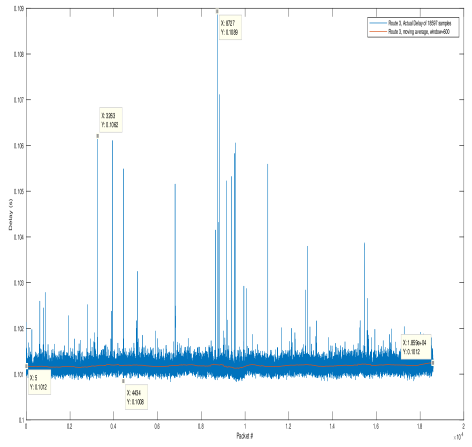
$$Clockdifference = \frac{(s_i - s_1) - (m_i - m1)}{T_i},$$
$$T_i = Ti.$$

Unless indicated otherwise, T, for the purposes of these tests is 30 seconds.

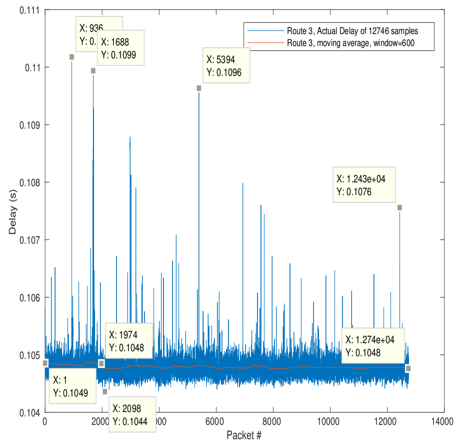
The figures will also show the fluctuations between each sample point, i.e.:

$$fluctuations = \frac{(s_i - s_{i-1}) - (m_i - mi - 1)}{T},$$

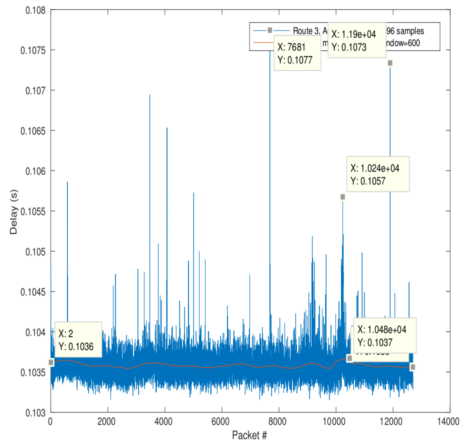
over the same T.



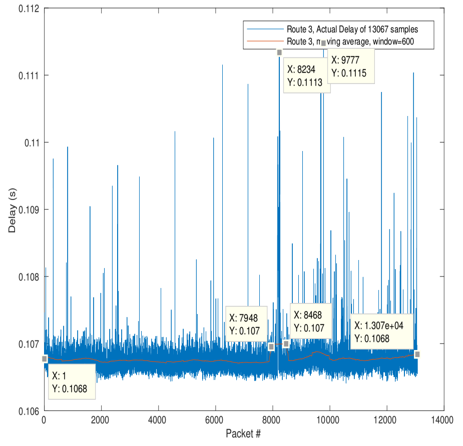
(a) May 22 12:50, PC8→PC5



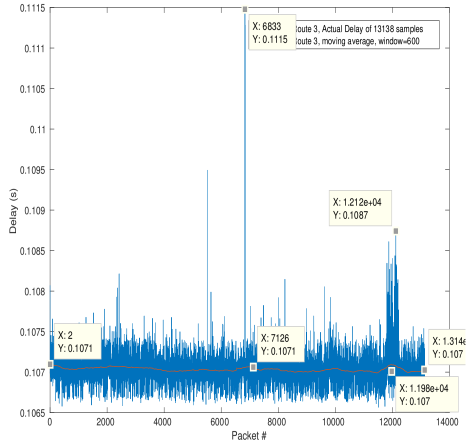
(b) May 22 13:50, PC8→PC5



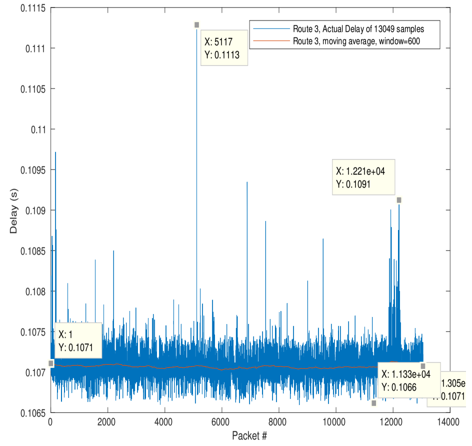
(c) May 22 17:44, PC5→PC8



(d) May 22 18:15, PC5→PC8

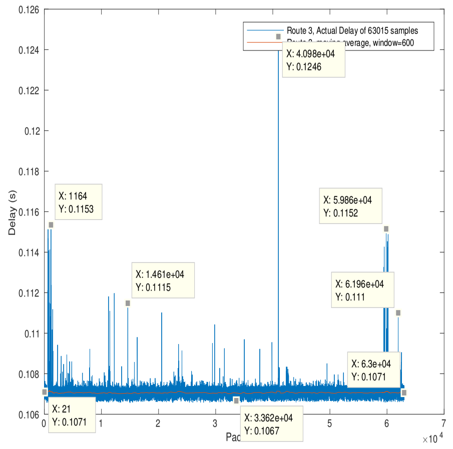


(e) May 28 13:12, PC5→PC8

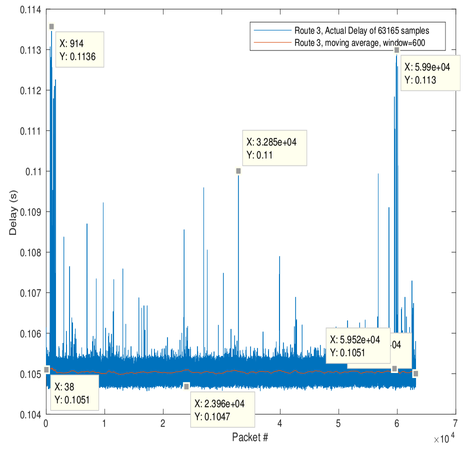


(f) May 28 13:12, PC8→PC5

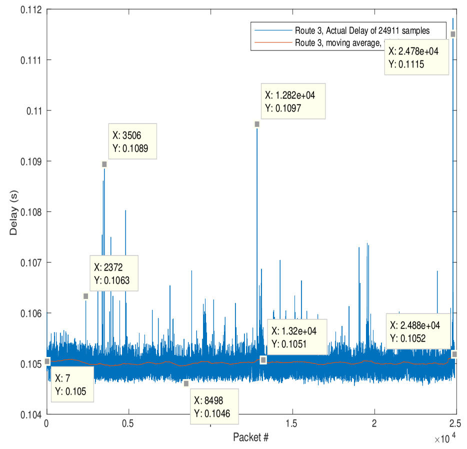
Figure 4.1: PlanetLab Delay Figures captured on May 22 and May 28



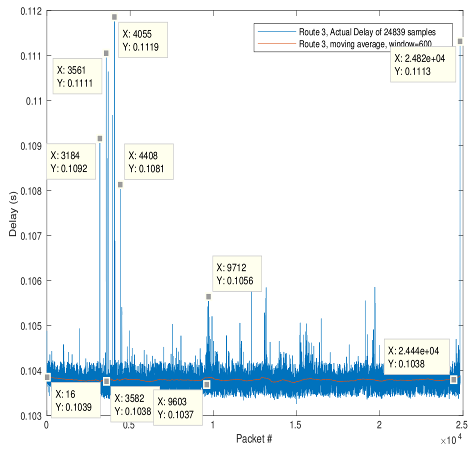
(a) May 28 16:13, PC5→PC8



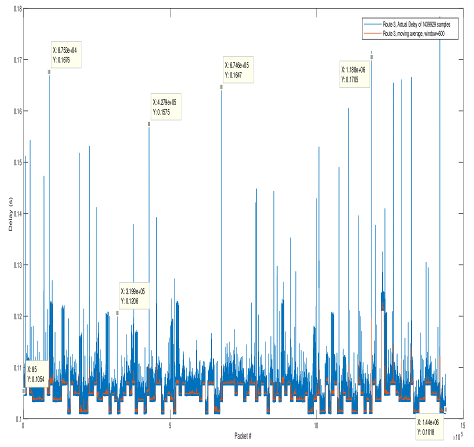
(b) May 28 16:13, PC8→PC5



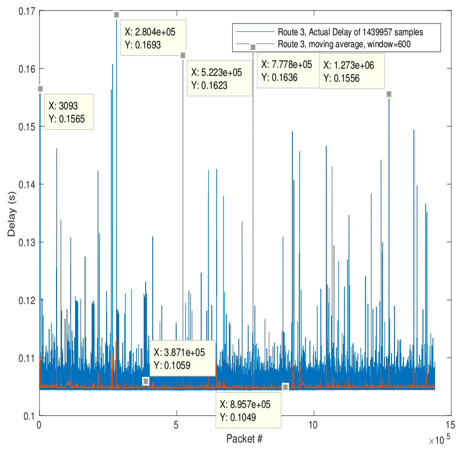
(c) May 29 12:37, PC5→PC8



(d) May 29 12:37, PC8→PC5



(e) June 11 15:40, ETH→PC5



(f) June 11 15:40, PC5→ETH

Figure 4.2: PlanetLab Delay Figures captured on May 28 and May June 11

4.2.1 Similar Computers

Presented here are the findings for two similar test computers. Similar test computers are defined here as two of the 4 primary test computers as outlined in 3.2.

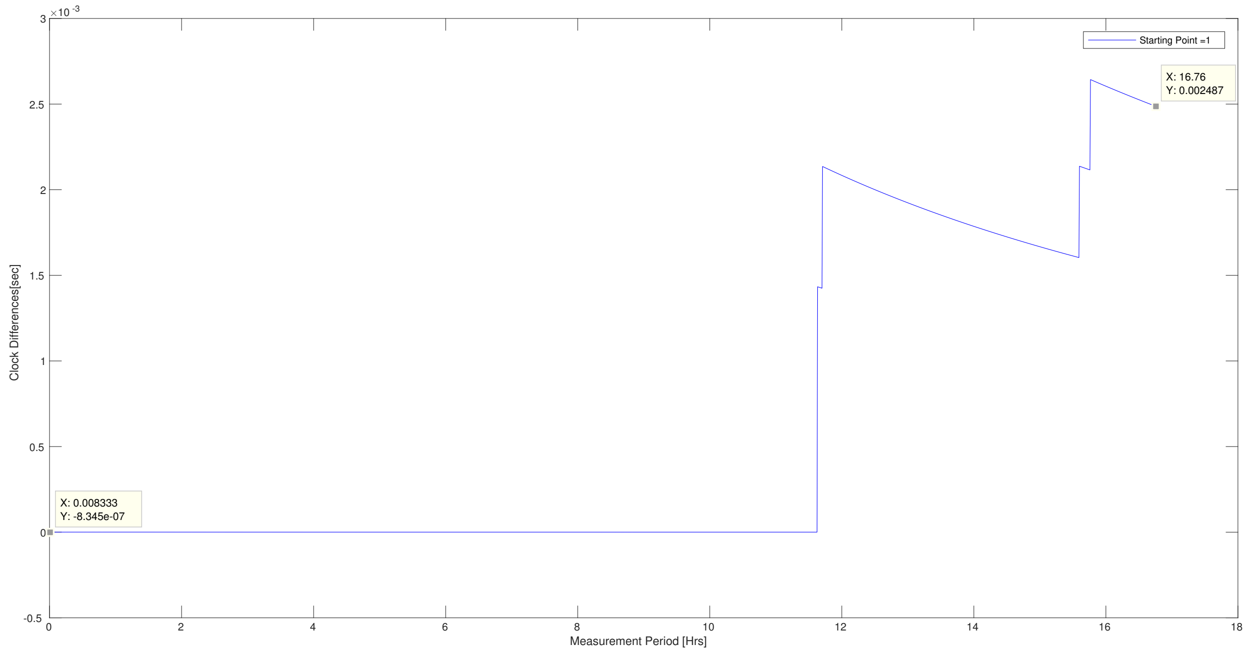
Figure 4.3a shows the results of the experiment conducted with two computers of the same specifications connected directly together. However, it is obvious that this is not the intended result. On closer inspection of Figure 4.3b, which shows the fluctuations between each packet, it is clear that there are 5 dropped packets. Meaning, the receiver never received the packets for whatever reason. The dropped packets have skewed our results, even though the underlying foundation is sound.

Figures 4.4 and 4.5 show the result of the experiment, when the dropped packets are replaced with the average elapsed time between each packet, approximately 30 seconds. The fluctuations shown in 4.5 confirm that the method is consistent with the overall trend and does not have a significant, distorting impact on our results.

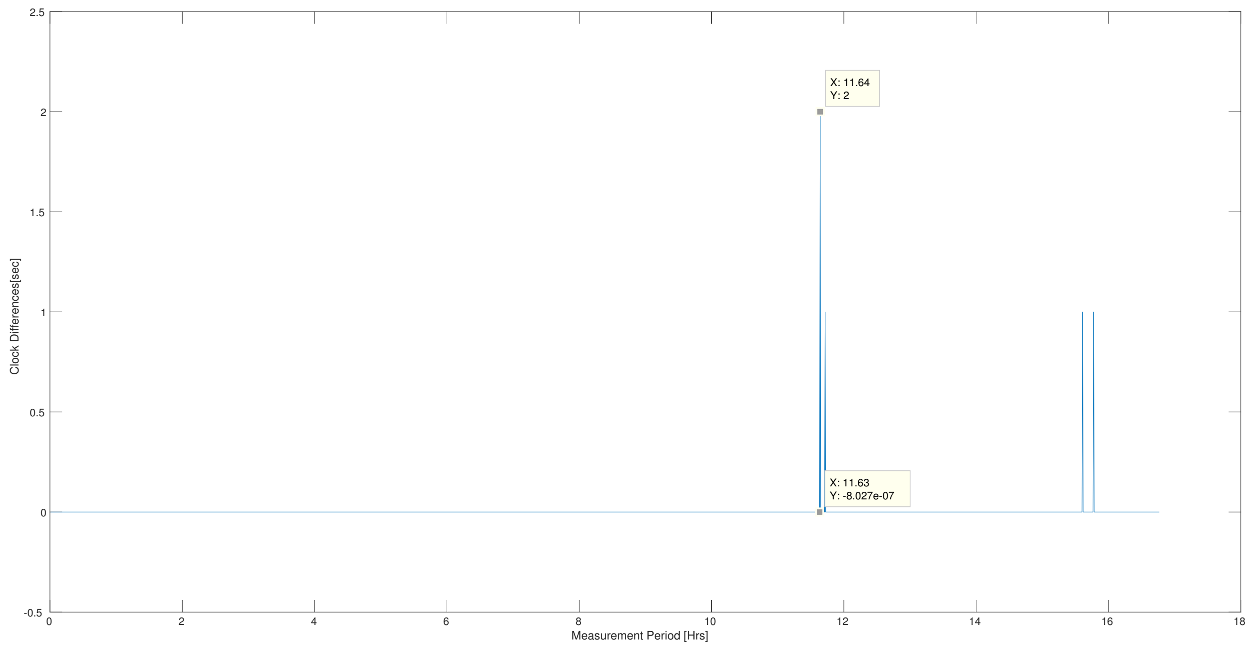
In Figure 4.4, the normalized difference over 1 second based on the last measurement is *3.824e-7*. The calculated mean of the normalized difference of the difference clocks (i.e. mean of the fluctuations) is *3.8224e-7*, with the variance of the same figures being *2.7389e-12*. Further, the results are collected over 2012 sample points, with the period between each sample being 30 seconds, as calculated by the `CLOCK_REALTIME`. The duration of the experiment is approximately 16 hours and 46 minutes

Figure 4.6 show the results of the same experiment, however, with the master-slave relationship reversed between the Test PCs. Here we have the previously transmitting computer receiving the packets, and vice-versa. Figure 4.7 renders the fluctuations between each sent packet. In Figure 4.4, the normalized difference over 1 second based on the last measurement is *-5.545e-7*.

The results are collected over 3602 sample points, with the period between each sample being 30 seconds. The duration of the experiment is approximately 30 hours. The calculated mean of the normalized difference of the difference clocks (i.e. mean of the fluctuations) is *-5.543e-7*, with the variance of the same figure being *1.9272e-12*.



(a) Similar Computers: 2012 Samples normalized average (last-first)



(b) Similar Computers: 2012 Samples Fluctuations

Figure 4.3: Similar Computers: 2012 Direct Connection aklinux5→aklinux4

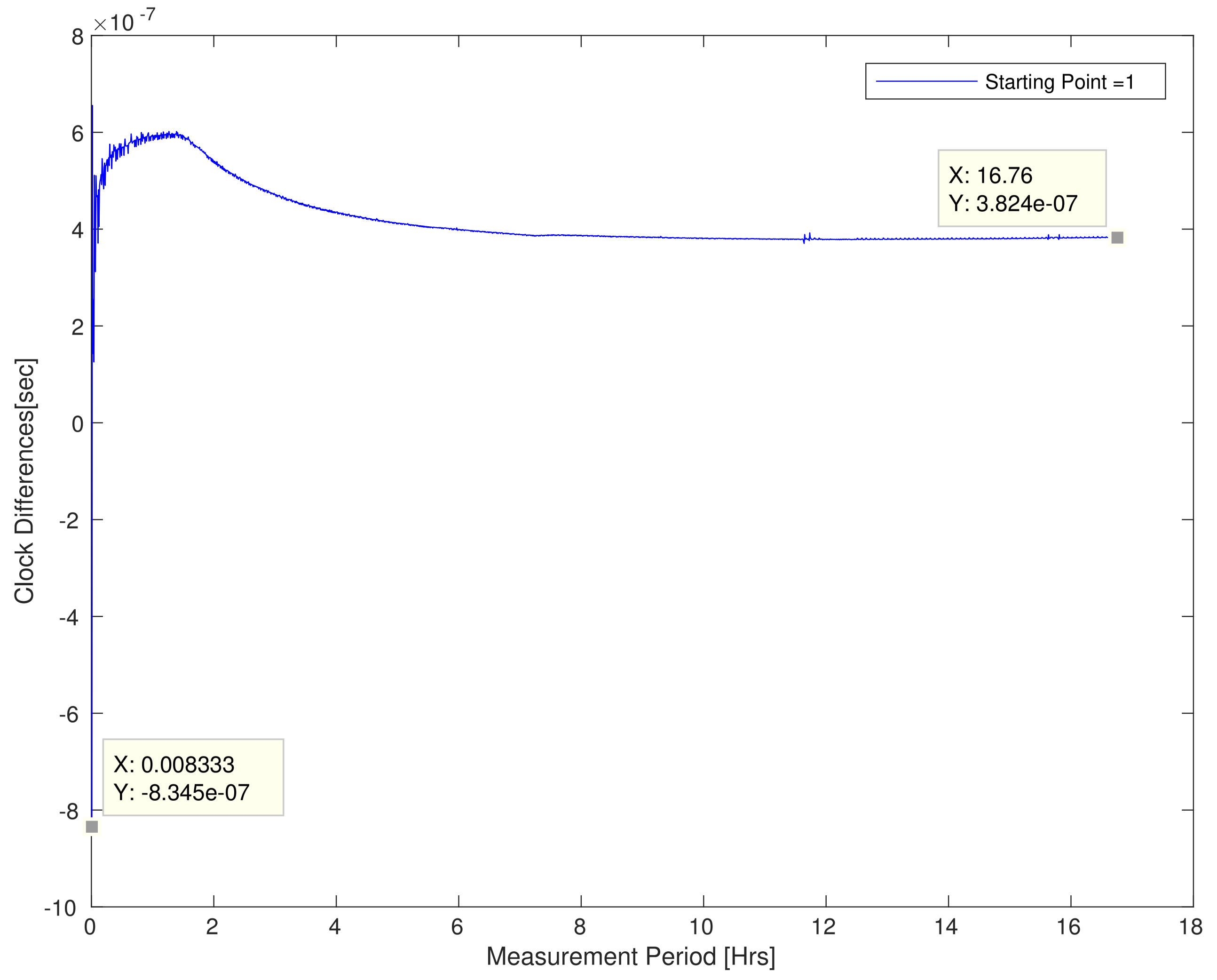


Figure 4.4: Similar Computers: 2012 Samples normalized average Corrected (last-first) aklinux5→aklinux4

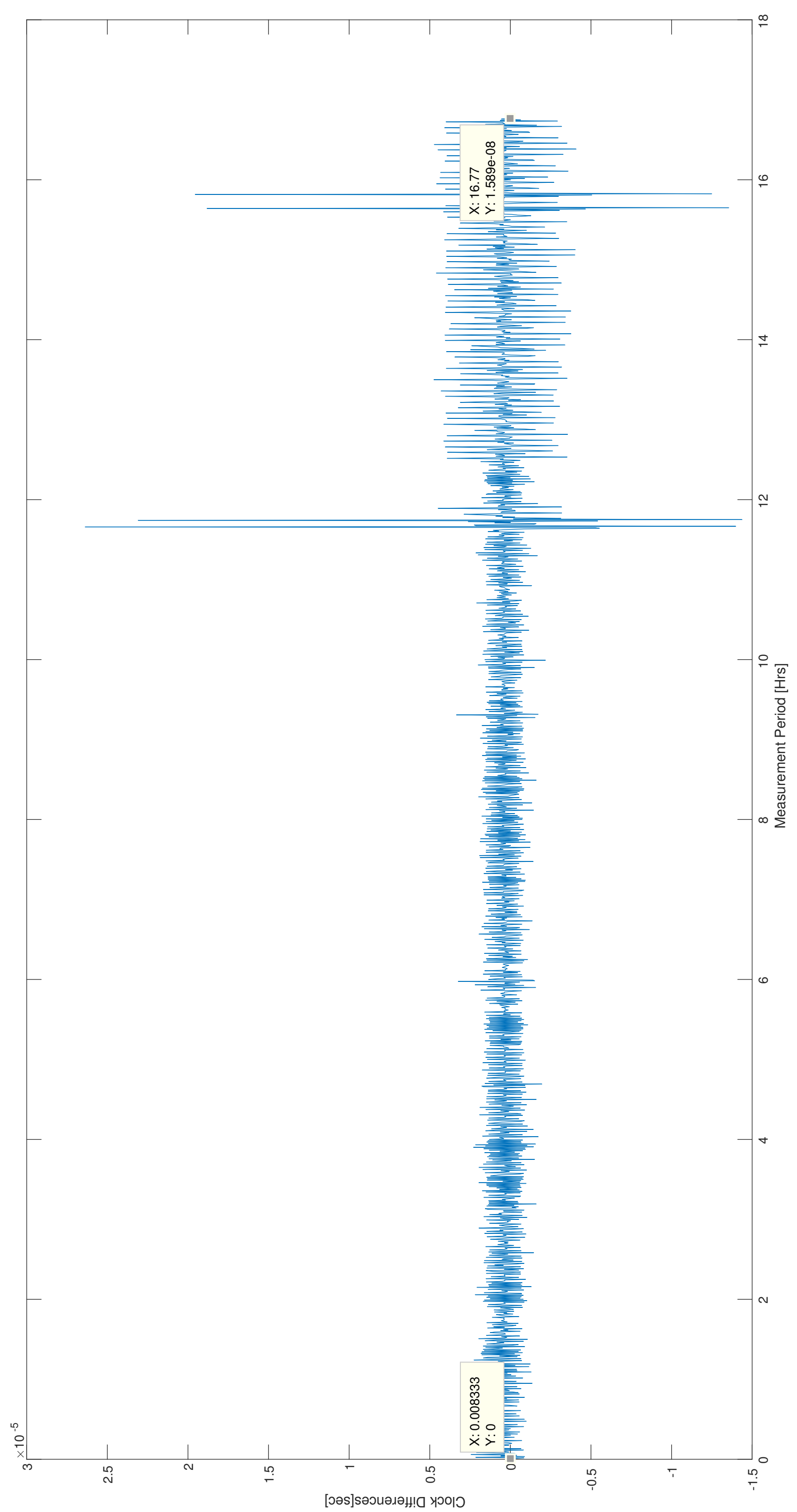


Figure 4.5: Similar Computers: 2012 Samples Fluctuations Corrected $(i - (i - 1))$ aklinux5→aklinux4

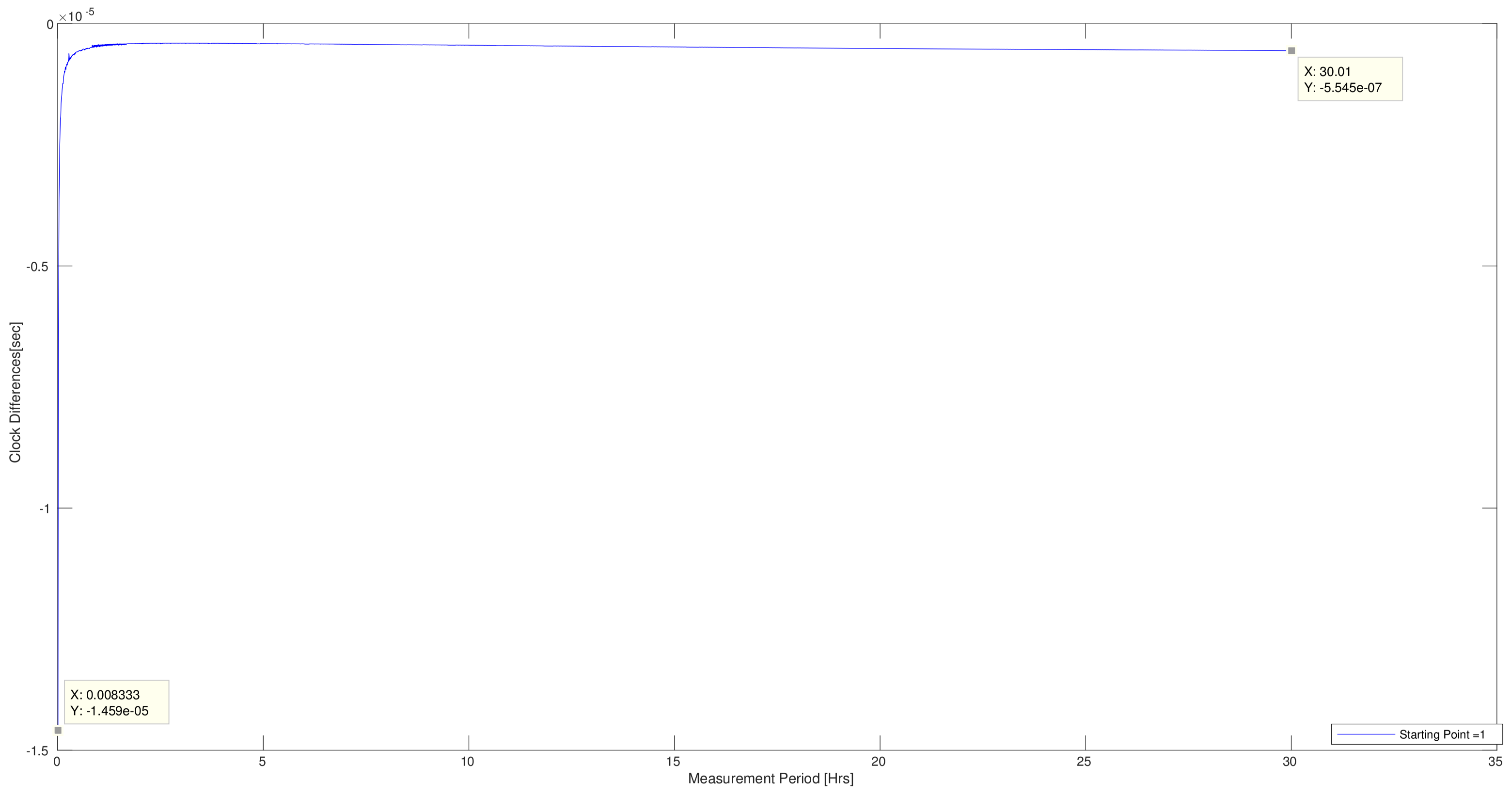


Figure 4.6: Reversed Similar Computers: 3602 Samples normalized average (last-first) aklinux4→aklinux5

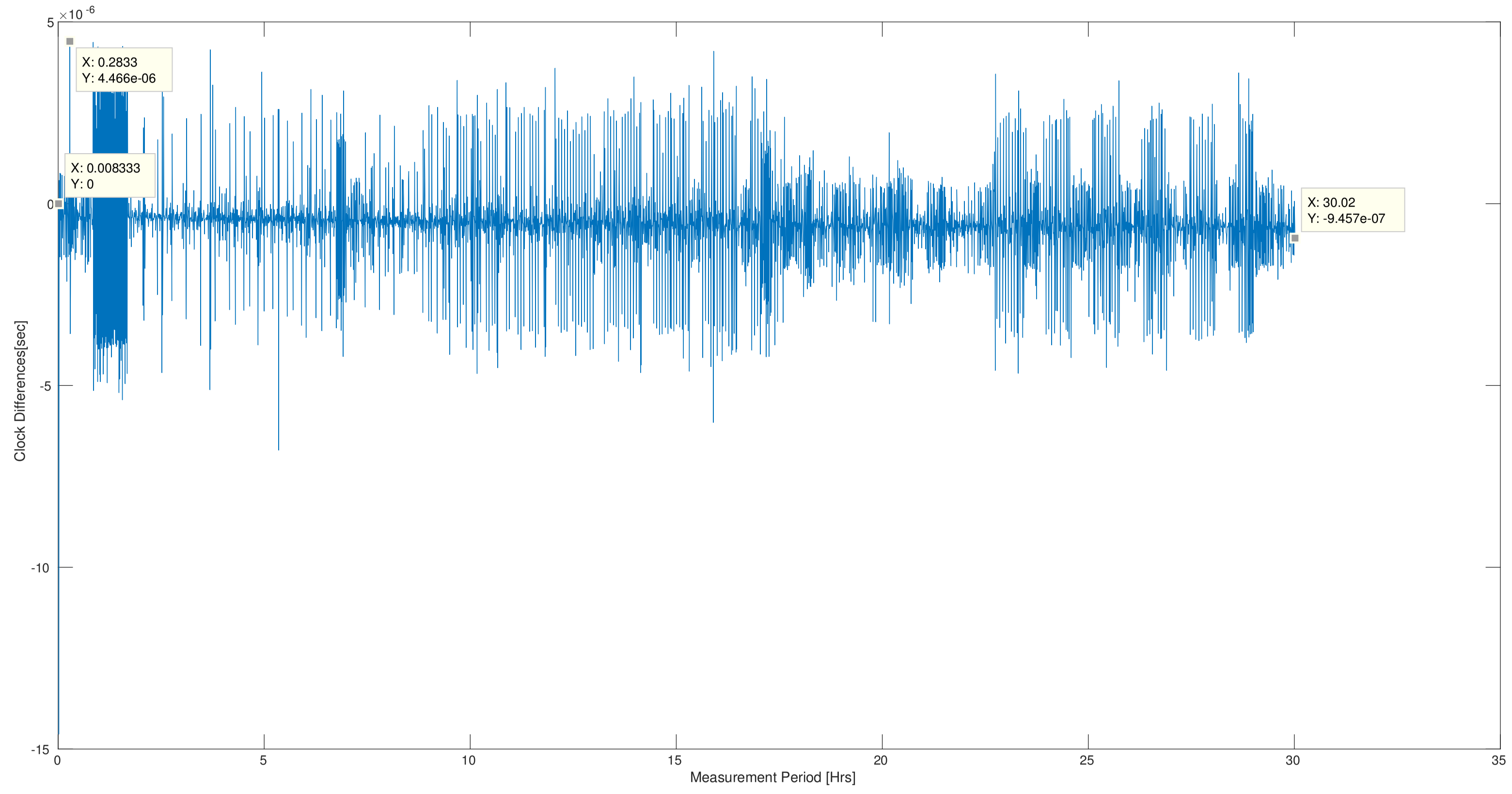


Figure 4.7: Reversed Similar Computers: 3602 Samples Fluctuations $(i-(i-1))$ aklinux4→aaklinux5

4.2.2 Dissimilar Computers

Presented here are the findings for two dissimilar test computers. Dissimilar test computers are defined here as one of the 4 primary test computers as outlined in 3.2, connected to the computer outlined in 3.2, which does not have the same hardware components as the 4 primary test computers. The dissimilar computer, in comparison to the other 4 primary test computers will, henceforth, be referred to as the Non-Standard (NS) computer.

Figure 4.8 shows the results of the experiment conducted with two computers that are connected together using a direct connection with the help of a cross-over cable. Figure 4.9 displays the fluctuations in delay between each packet exchange.

In Figure 4.8, the normalized difference over 1 second based on the last measurement is *-1.703e-5*. The calculated mean of the normalized difference of the difference clocks (i.e. mean of the fluctuations) is *-1.702e-5*, with the variance of the same figures being *7.2919e-13*. Further, the results are collected over 2557 sample points, with the period between each sample being 30 seconds, as calculated by the `CLOCK_REALTIME`. The duration of the experiment is approximately 21 hours and 18 minutes.

Figure 4.10 show the results of the same experiment, however, with the master-slave relationship reversed between the Test PCs. Here we have the previously transmitting computer receiving the packets, and vice-versa. Figure 4.11 renders the fluctuations between each sent packet. In Figure 4.10, the normalized difference over 1 second based on the last measurement is *1.723e-5*.

The results are collected over 3972 sample points, with the period between each sample being 30 seconds. The duration of the experiment is approximately 33 hours. The calculated mean of the normalized difference of the difference clocks (i.e. mean of the fluctuations) is *1.722e-05*, with the variance of the same figure being *3.255e-12*.

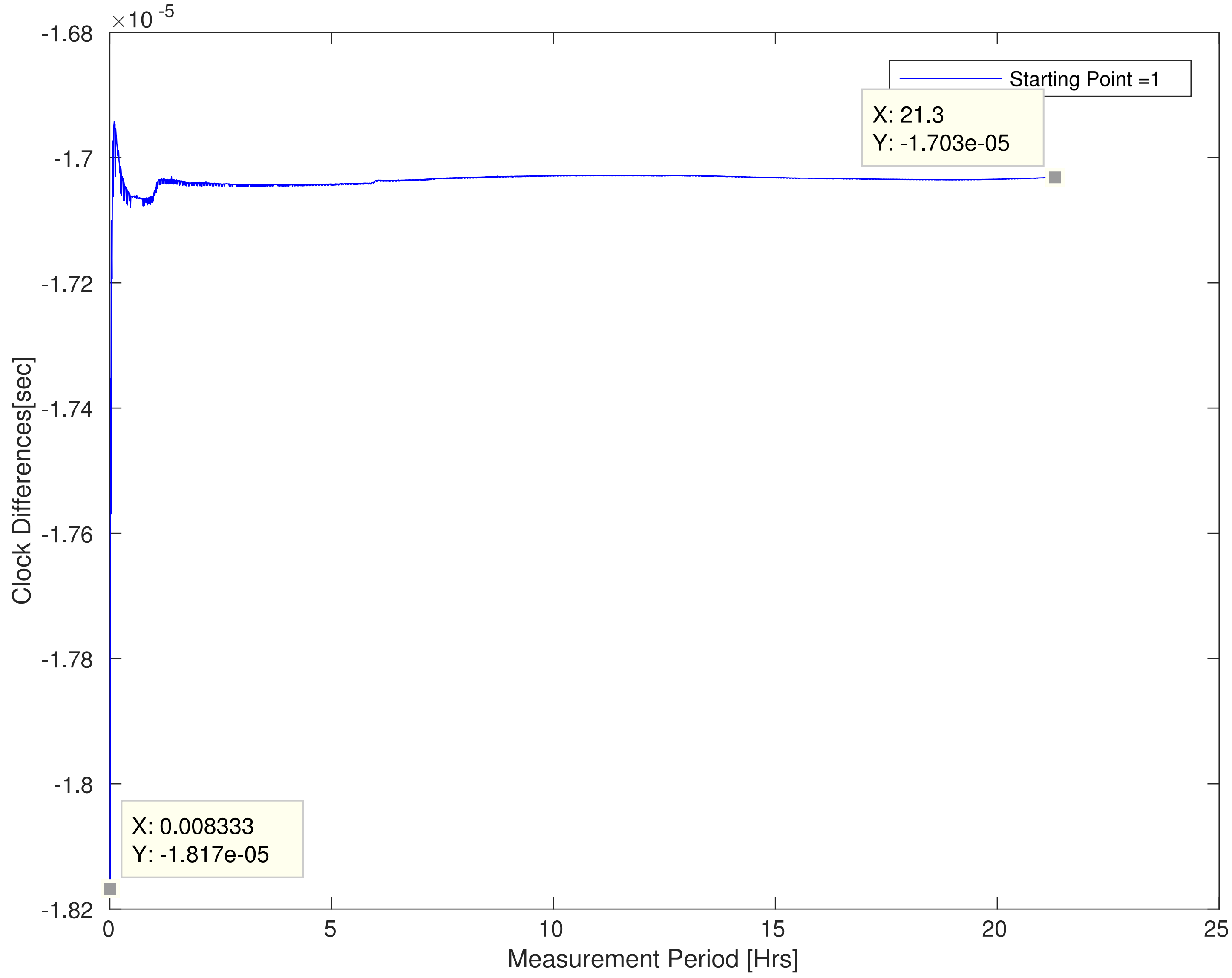


Figure 4.8: Dissimilar Computers: 2557 Samples normalized average (last-first) NS→akkpc9

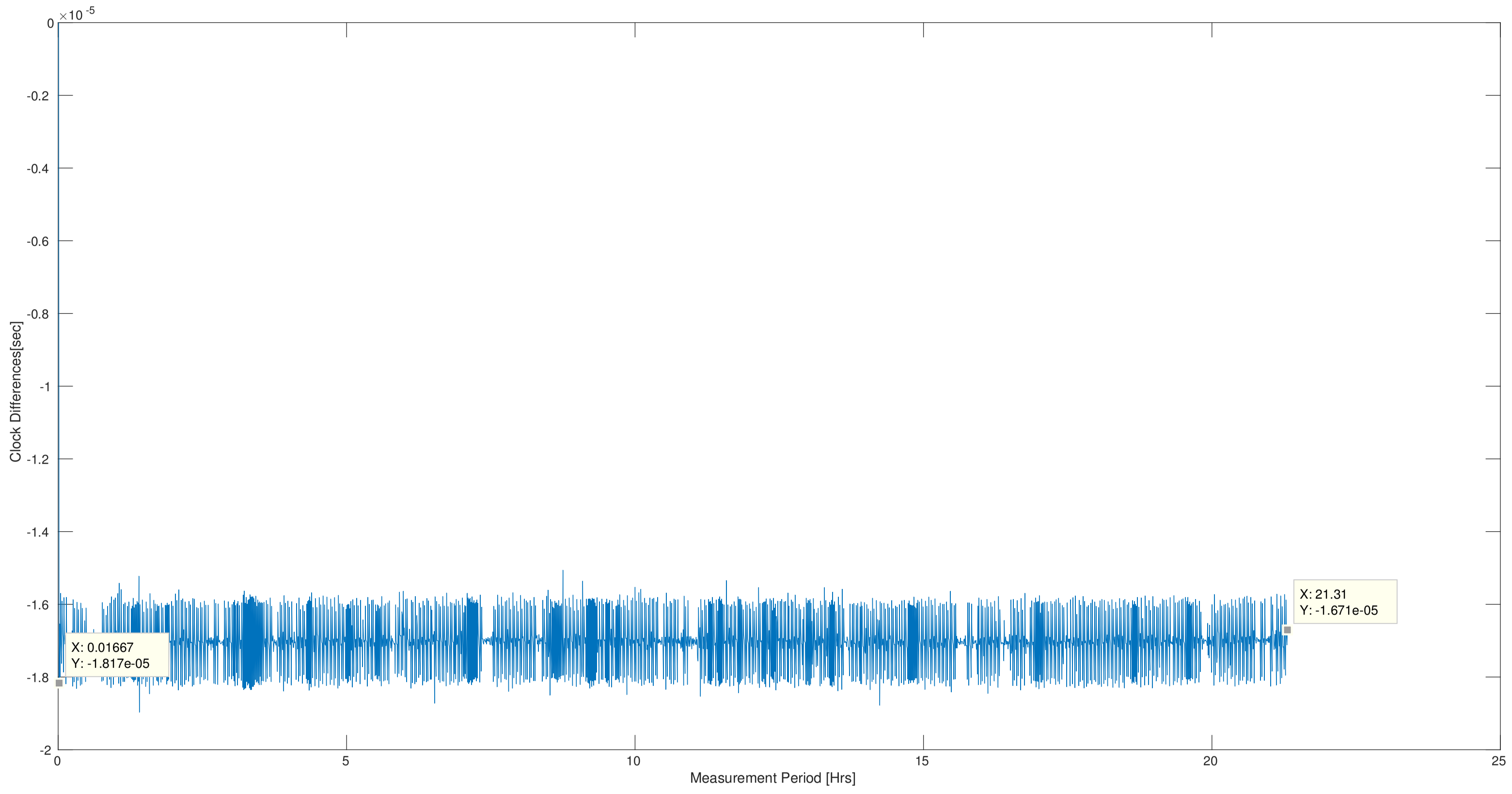


Figure 4.9: Dissimilar Computers: 2557 Samples Fluctuations $(i - (i - 1))$ NS→akkpc9

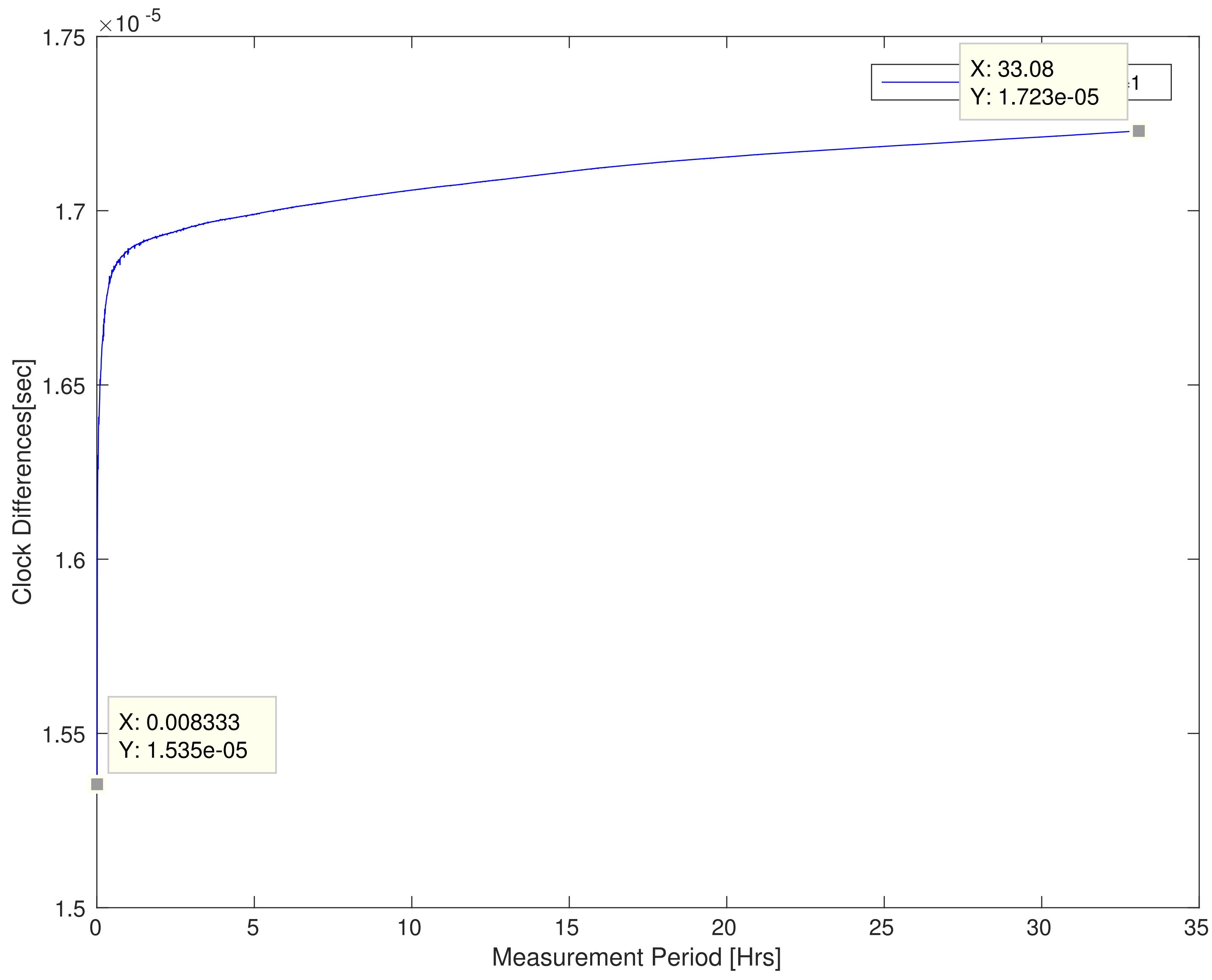


Figure 4.10: Reversed Dissimilar Computers: 3972 Samples normalized average (last-first) akkpc9→NS

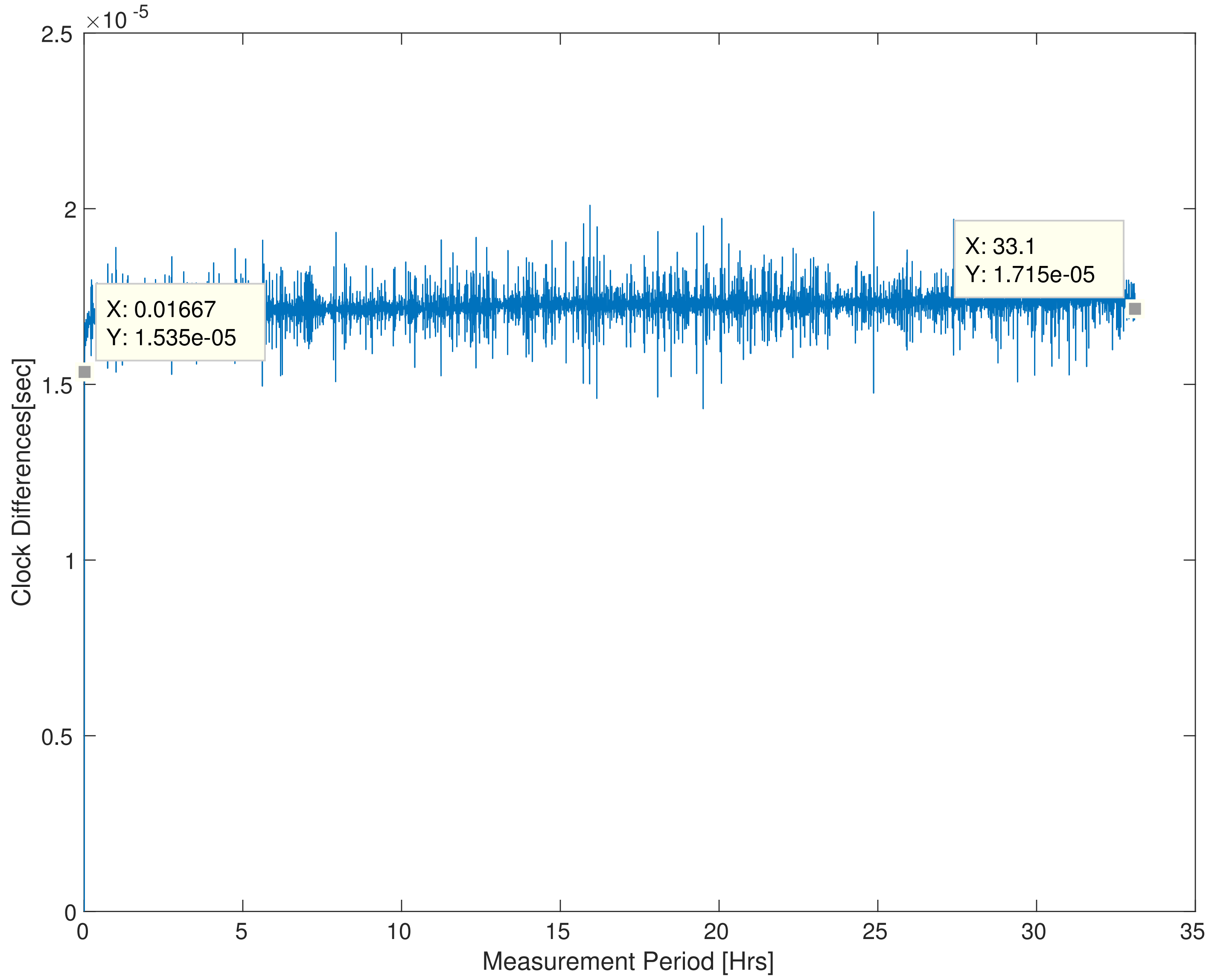


Figure 4.11: Reversed Dissimilar Computers: 3972 Samples Fluctuations $(i-(i-1))$ akkpc9→NS

4.3 Establishing Time Accuracy Over the Network: UW Network

The following tests were completed within the University of Waterloo network. As discussed previously, the tests are varied between two similar computers, and also two dissimilar computers. Further, a simple `traceroute` was used to see how many hops the computers needed to make, and it was indicated that the two computers encountered no significant delay in their traffic. This means the traffic would have only traveled through level-2 devices under the OSI model, such as network switches.

4.3.1 Similar Computers

The following are the results of similar Test PCs connected over the UW network, starting with Figure 4.12, which displays the difference of the difference between the slave's last received packet and its first, and the master's last sent packet and its first, average and normalized over 1 second, and traced over time, as described in 4.2.1. Figure 4.13 displays the fluctuations in delay between each packet exchange.

As can be seen in Figure 4.12, the normalized difference over 1 second based on the last measurement is *-6.544e-7*. The calculated mean of the normalized difference of the difference clocks (i.e. mean of the fluctuations) is *-6.539e-7*, with the variance of the same figures being *5.538e-17*. Further, the results are collected over 1509 sample points, with the period between each sample being 30 seconds, as calculated by the `CLOCK_REALTIME`. The duration of the experiment is approximately 12 hours and 35 minutes.

Figure 4.14 show the results of the same experiment, however, with the master-slave relationship reversed between the Test PCs. Here we have the previously transmitting computer receiving the packets, and vice-versa. Figure 4.15 renders the fluctuations between each sent packet. In Figure 4.14, the normalized difference over 1 second based on the last measurement is *6.55e-7*.

The results are collected over 400 sample points, with the period between each sample being 30 seconds. The duration of the experiment is approximately 3 hours and 20 minutes. The calculated mean of the normalized difference of the difference clocks (i.e. mean of the fluctuations) is *6.533e-7*, with the variance of the same figure being *5.701e-13*.

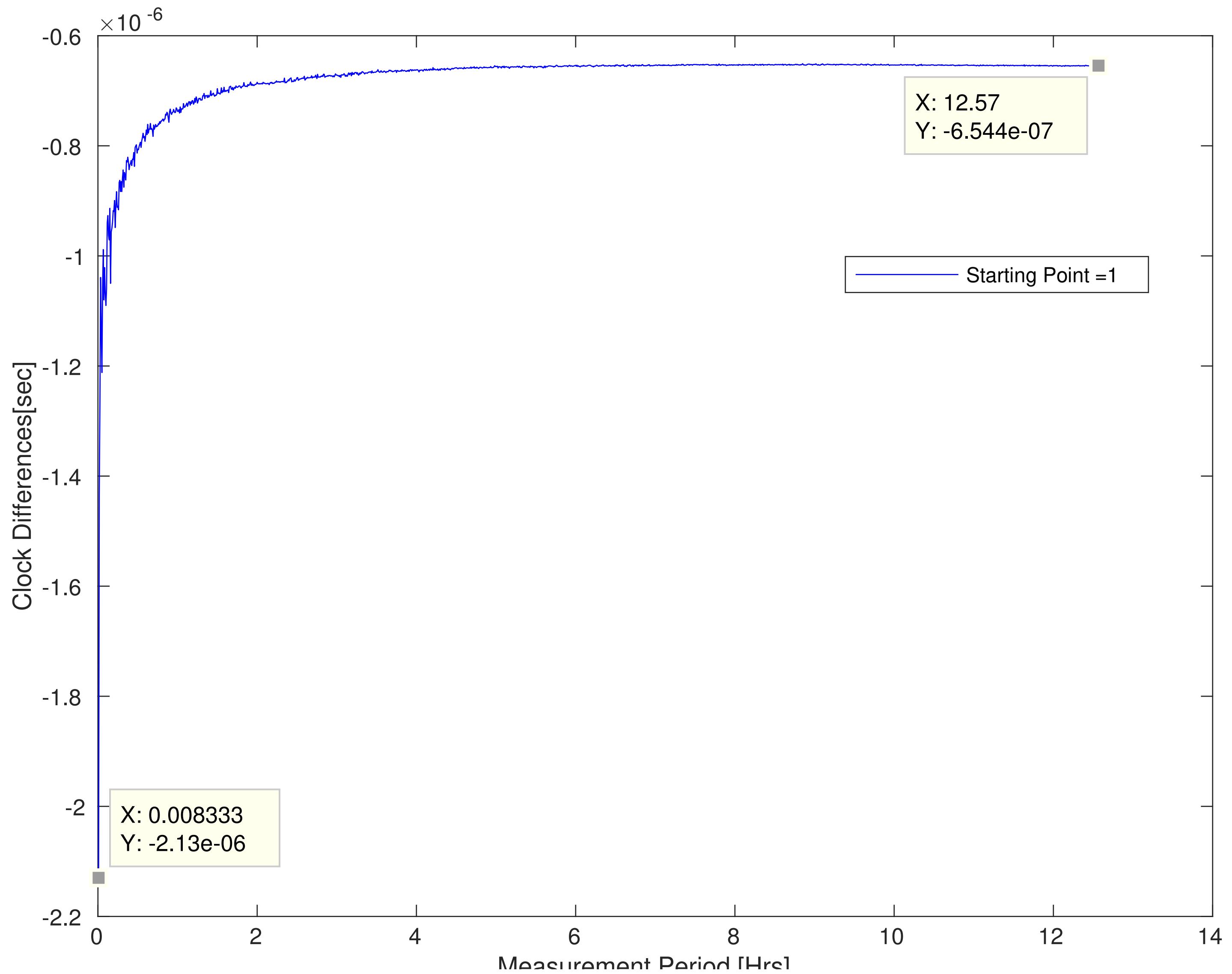


Figure 4.12: Similar Computers Over the UW Network: 1509 Samples normalized average (last-first) aklinux4→aklinux5

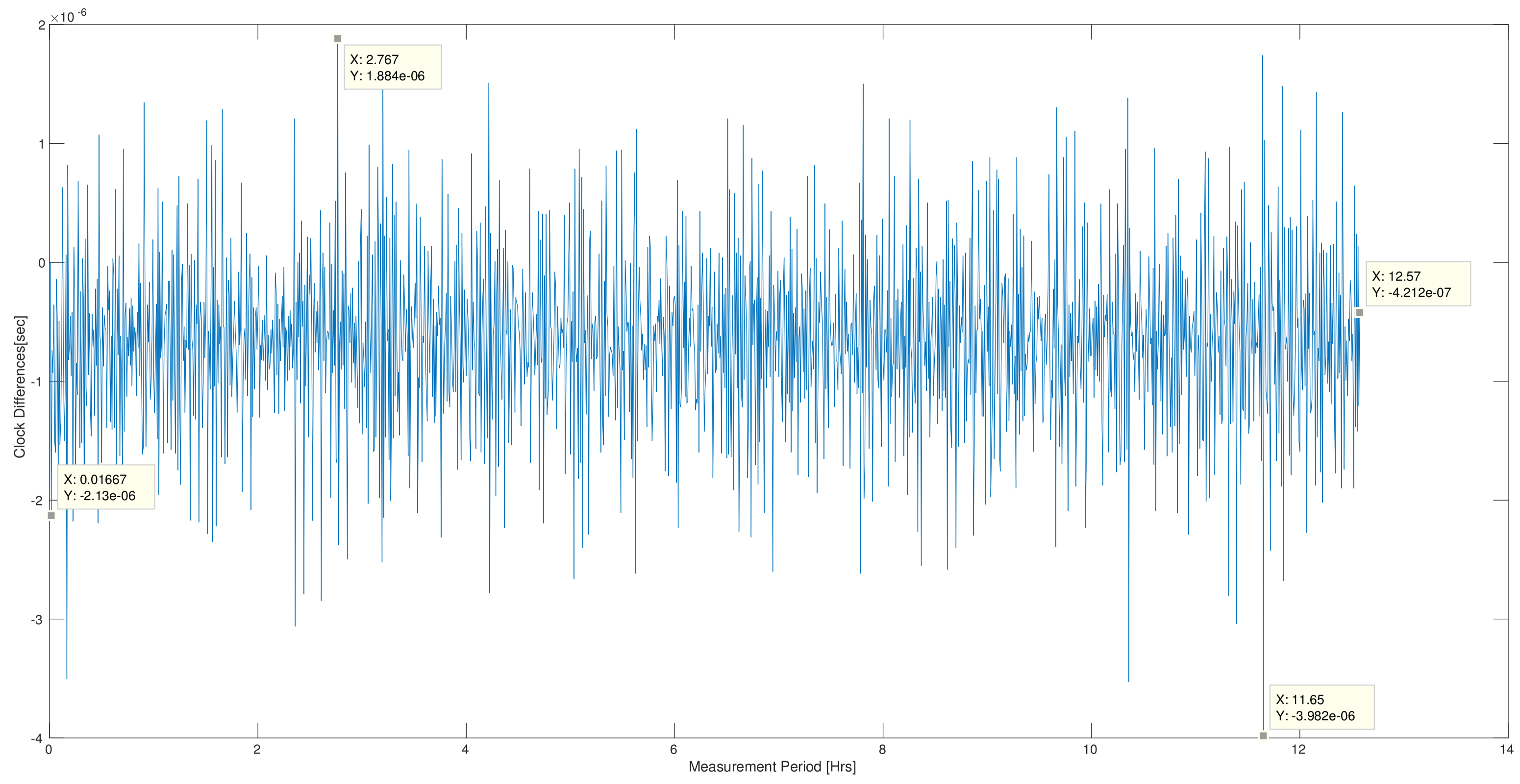


Figure 4.13: Similar Computers Over the UW Network: 1509 Samples Fluctuations $(i - (i - 1))$ aklinux4→aklinux5

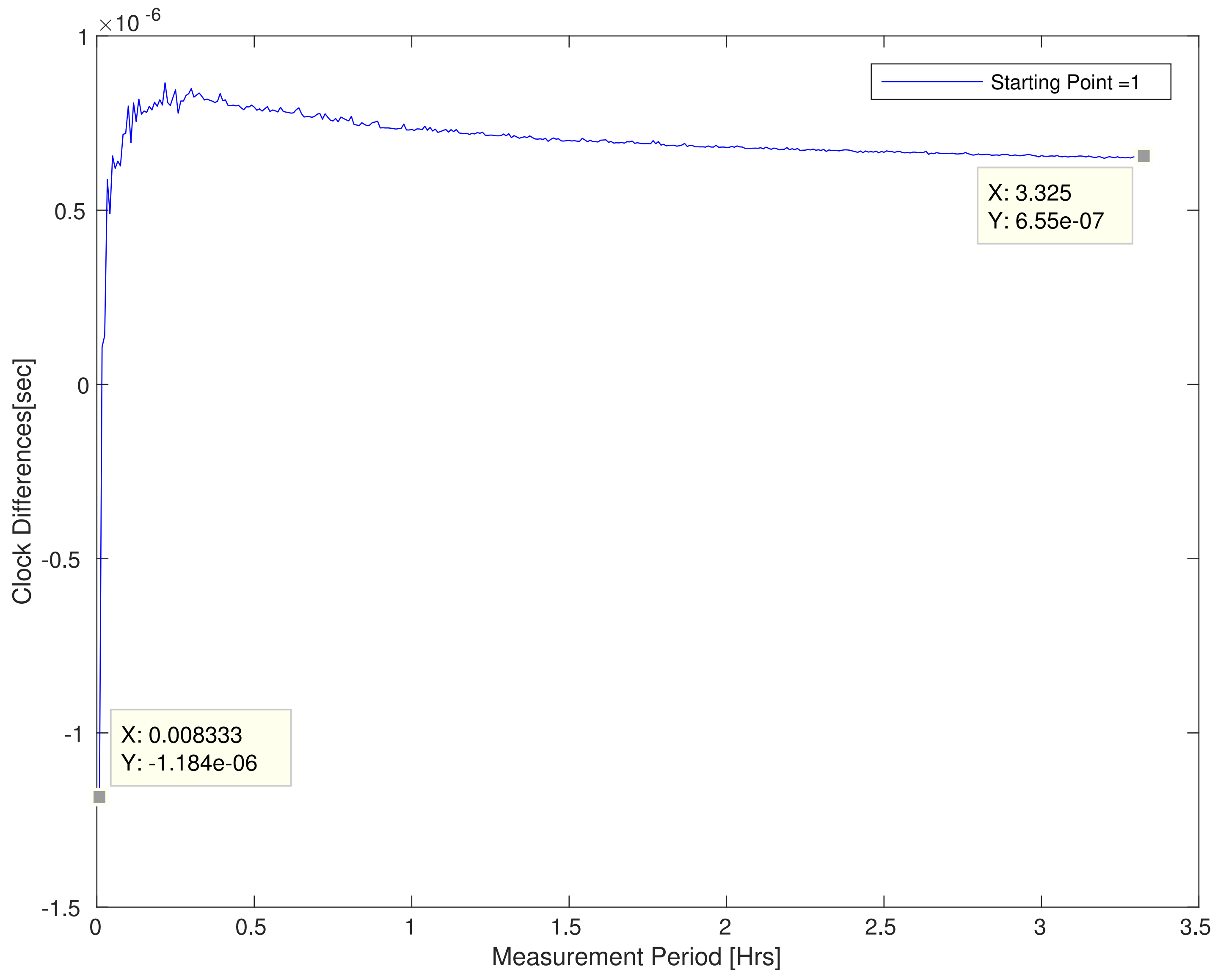


Figure 4.14: Similar Computers Over the UW Network Reversed: 400 Samples normalized average (last-first) aklinux5→aklinux4

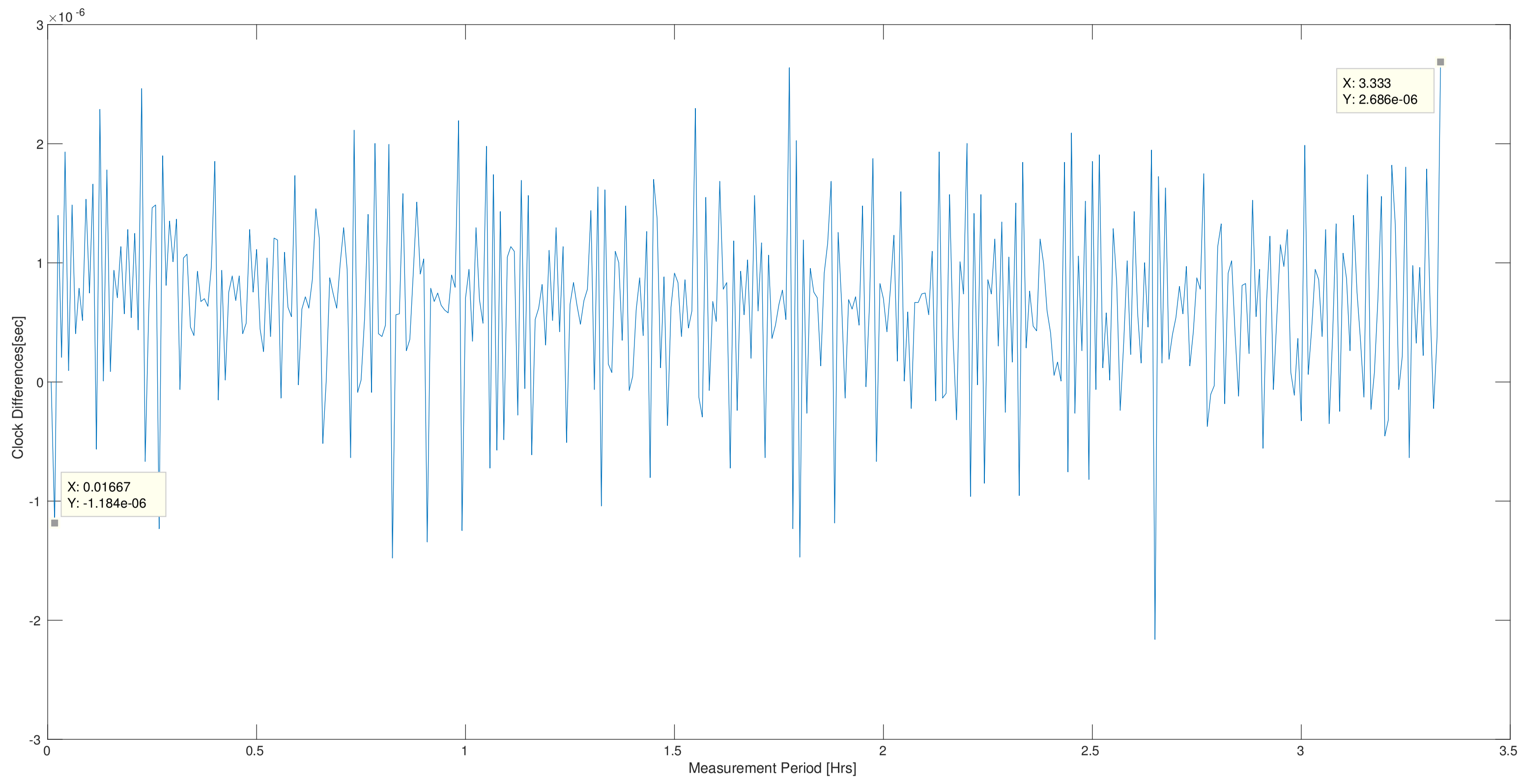


Figure 4.15: Similar Computers Over the UW Network Reversed: 400 Samples Fluctuations $(i-(i-1))$ aklinux5→aaklinux4

4.3.2 Dissimilar Computers

The following are the results of dissimilar Test PCs connected over the UW network, starting with Figure 4.16, per the method outlined in 4.2.1. Figure 4.17 displays the fluctuations in delay between each packet exchange.

As can be seen in Figure 4.16, the normalized difference over 1 second based on the last measurement is *1.748e-5*. The calculated mean of the normalized difference of the difference clocks (i.e. mean of the fluctuations) is *1.747e-5*, with the variance of the same figures being *4.174e-13*. Further, the results are collected over 1526 sample points, with the period between each sample being 30 seconds, as calculated by the `CLOCK_REALTIME`. The duration of the experiment is approximately 12 hours and 40 minutes.

Figure 4.18 show the results of the same experiment, however, with the master-slave relationship reversed between the Test PCs. Here we have the previously transmitting computer receiving the packets, and vice-versa. Figure 4.19 renders the fluctuations between each sent packet. In Figure 4.18, the normalized difference over 1 second based on the last measurement is *-1.731e-5*.

The results are collected over 280 sample points, with the period between each sample being 30 seconds. The duration of the experiment is approximately 2 hours and 20 minutes. The calculated mean of the normalized difference of the difference clocks (i.e. mean of the fluctuations) is *-1.725e-4*, with the variance of the same figure being *1.717e-12*.

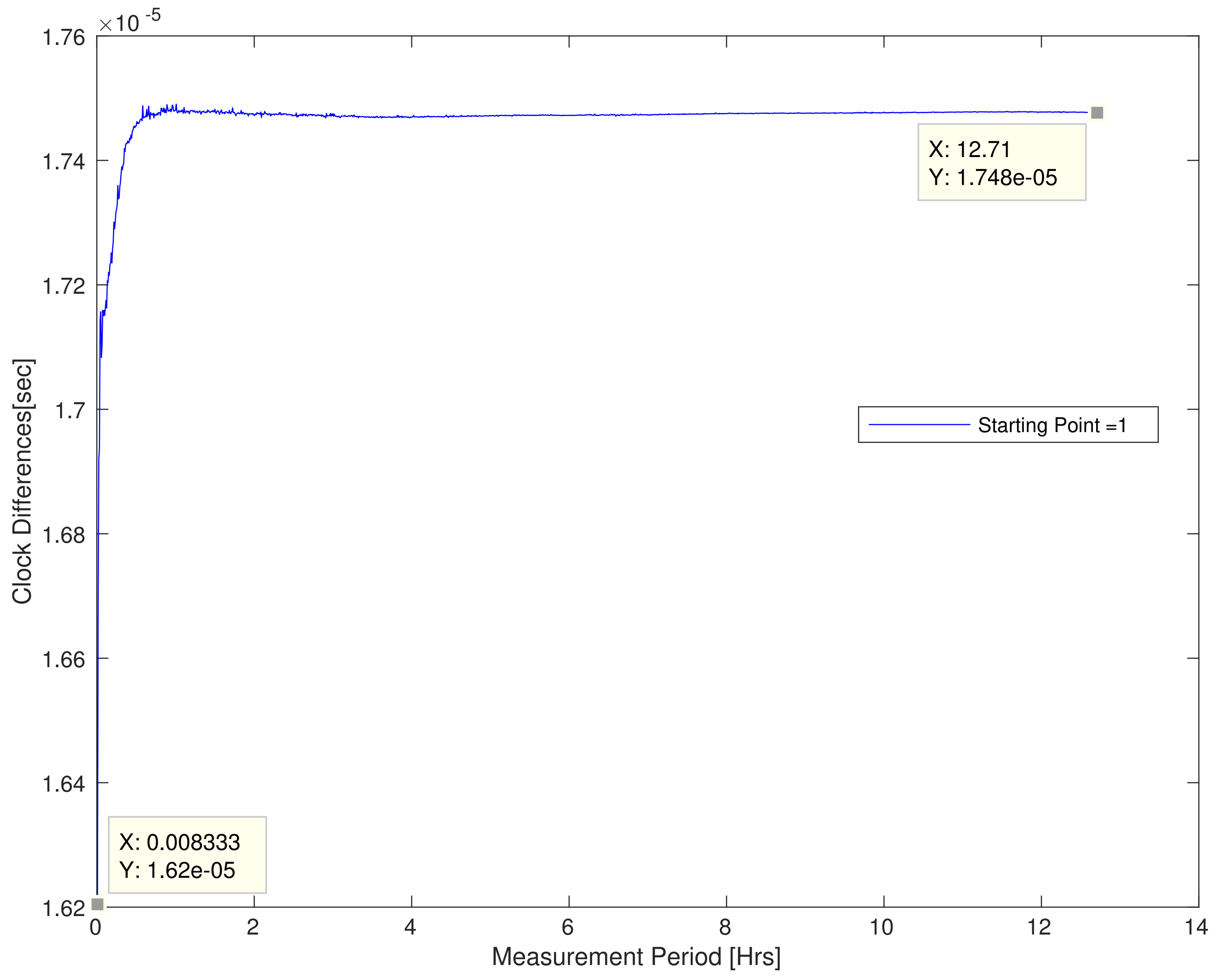


Figure 4.16: Dissimilar Computers Over the UW Network: 1526 Samples normalized average (last-first) akkpc9→NS

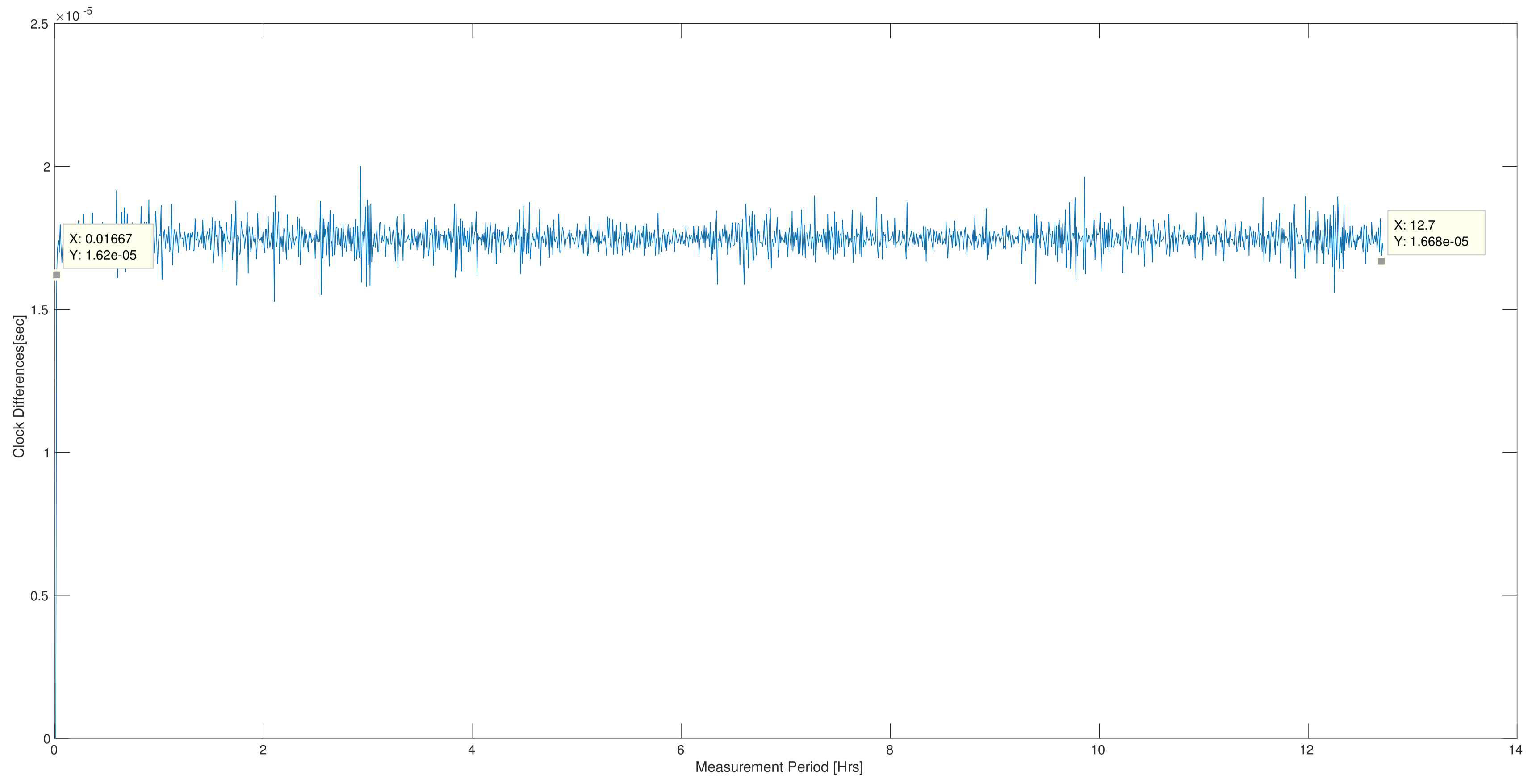


Figure 4.17: Dissimilar Computers Over the UW Network: 1526 Samples Fluctuations $(i - (i - 1))$ akkpc9→NS

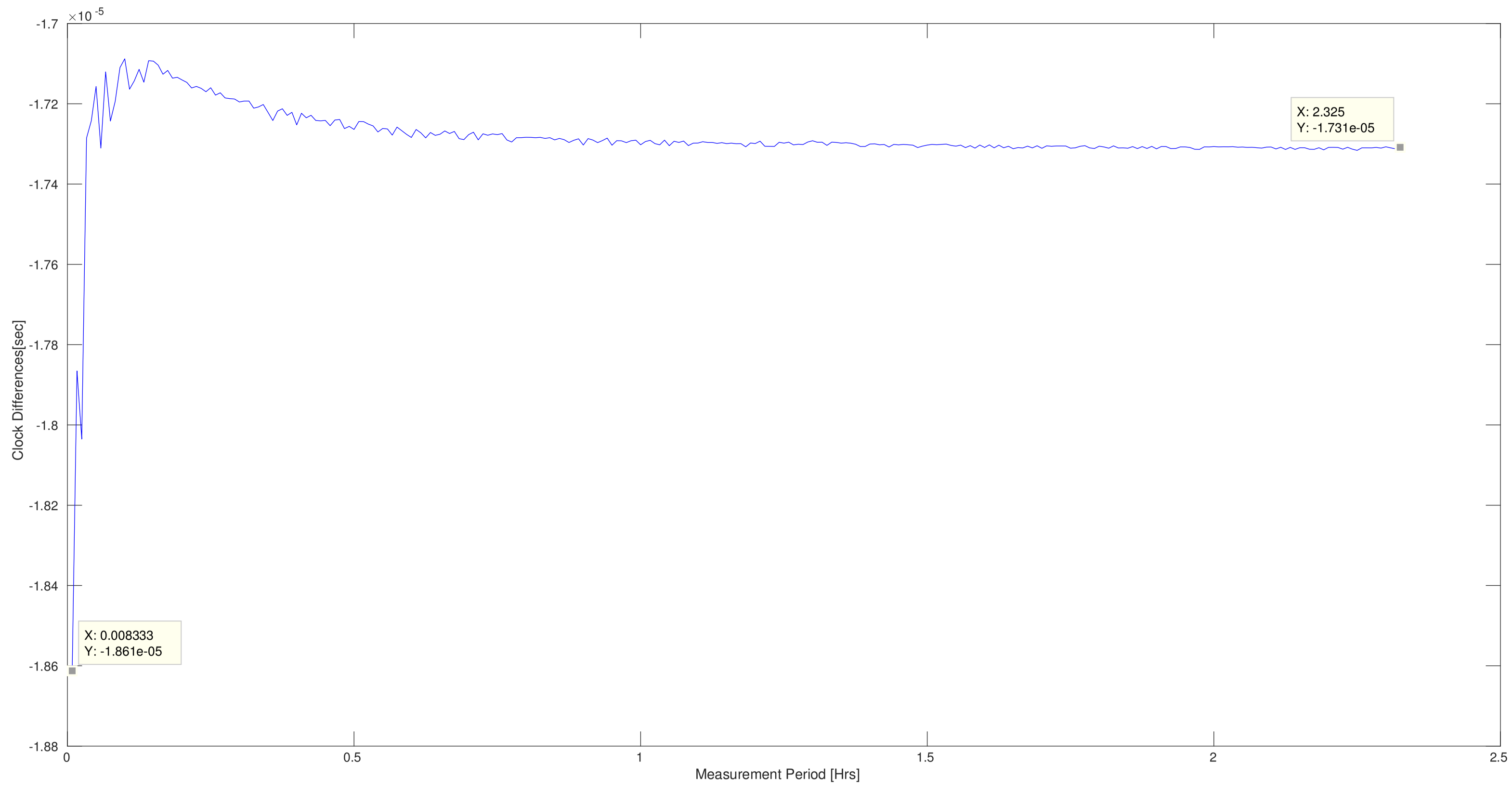


Figure 4.18: Dissimilar Computers Over the UW Network Reversed: 280 Samples normalized average (last-first) akkpc8→akkpc9

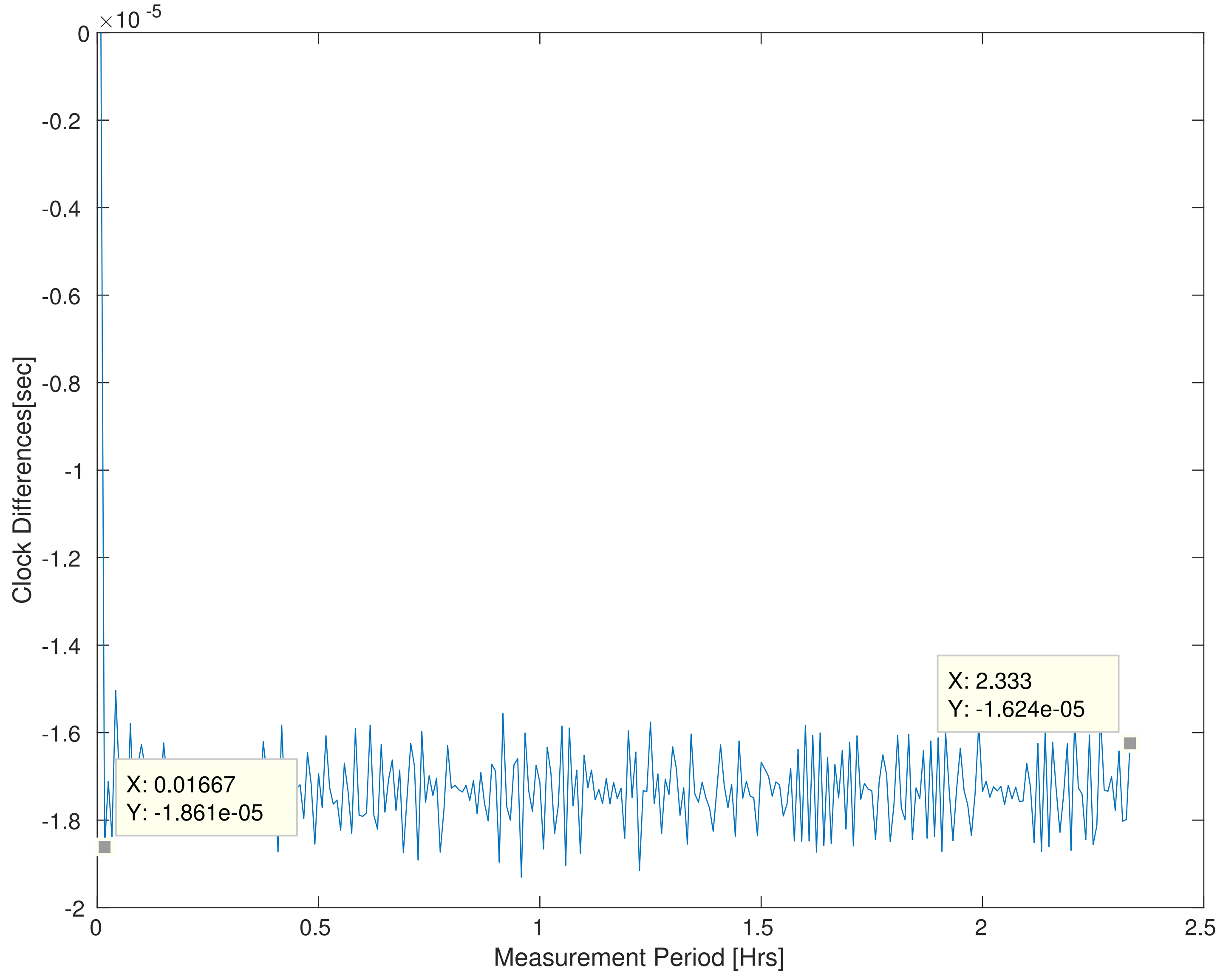


Figure 4.19: Dissimilar Computers Over the UW Network Reversed: 280 Samples Fluctuations $(i-(i-1))$ akkpc8→akkpc9

4.4 Establishing Time Accuracy Over the PlanetLab Network with 1 Node: University of Oregon

This section presents the experiments conducted using the PlanetLab network to connect to the PlanetLab servers at the University of Oregon in Eugene, OR. The tests are varied between two similar computers, and also two dissimilar computers. Further, a simple `traceroute` was used to see how many hops the computers needed to make, and there are about 30 hops between Test PCs, and their destination. The two routes are also different. The breakdown of these paths are shown in B

4.4.1 Similar Computers

The following are the results of similar Test PCs connected over the PlanetLab network, starting with Figure 4.20, which displays the difference of the difference between the slave's last received packet and its first, and the master's last sent packet and its first, average and normalized over 1 second, and traced over time, as described in 4.2.1. Figure 4.21 displays the fluctuations in delay between each packet exchange.

As can be seen in Figure 4.20, the normalized difference over 1 second based on the last measurement is *3.802e-7*. The calculated mean of the normalized difference of the difference clocks (i.e. mean of the fluctuations) is *-3.789e-7*, with the variance of the same figures being *1.064e-9*. Further, the results are collected over 292 sample points, with the period between each sample being 30 seconds, as calculated by the `CLOCK_REALTIME`. The duration of the experiment is approximately 2 hours and 25 minutes.

Figure 4.22 show the results of the same experiment, however, with the master-slave relationship reversed between the Test PCs. Here we have the previously transmitting computer receiving the packets, and vice-versa. Figure 4.23 renders the fluctuations between each sent packet. In Figure 4.22, the normalized difference over 1 second based on the last measurement is *-6.381e-7*.

The results are collected over 333 sample points, with the period between each sample being 30 seconds. The duration of the experiment is approximately 2 hours and 45 minutes. The calculated mean of the normalized difference of the difference clocks (i.e. mean of the fluctuations) is *-6.362e-7*, with the variance of the same figure being *9.483e-10*.

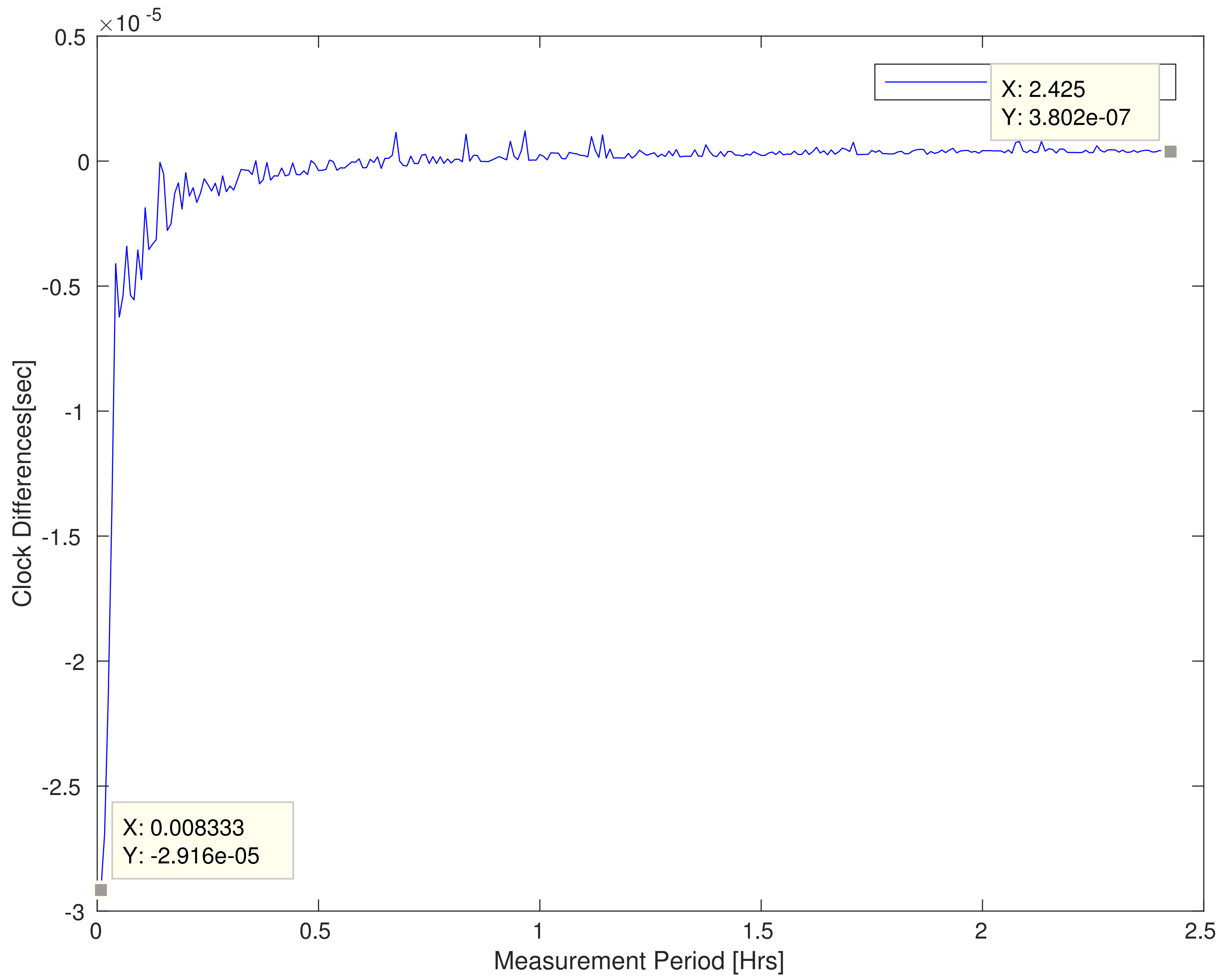


Figure 4.20: Similar Computers Over the PL Network: 292 Samples normalized average (last-first) aklinux5→UOregon→UW→aklinux4

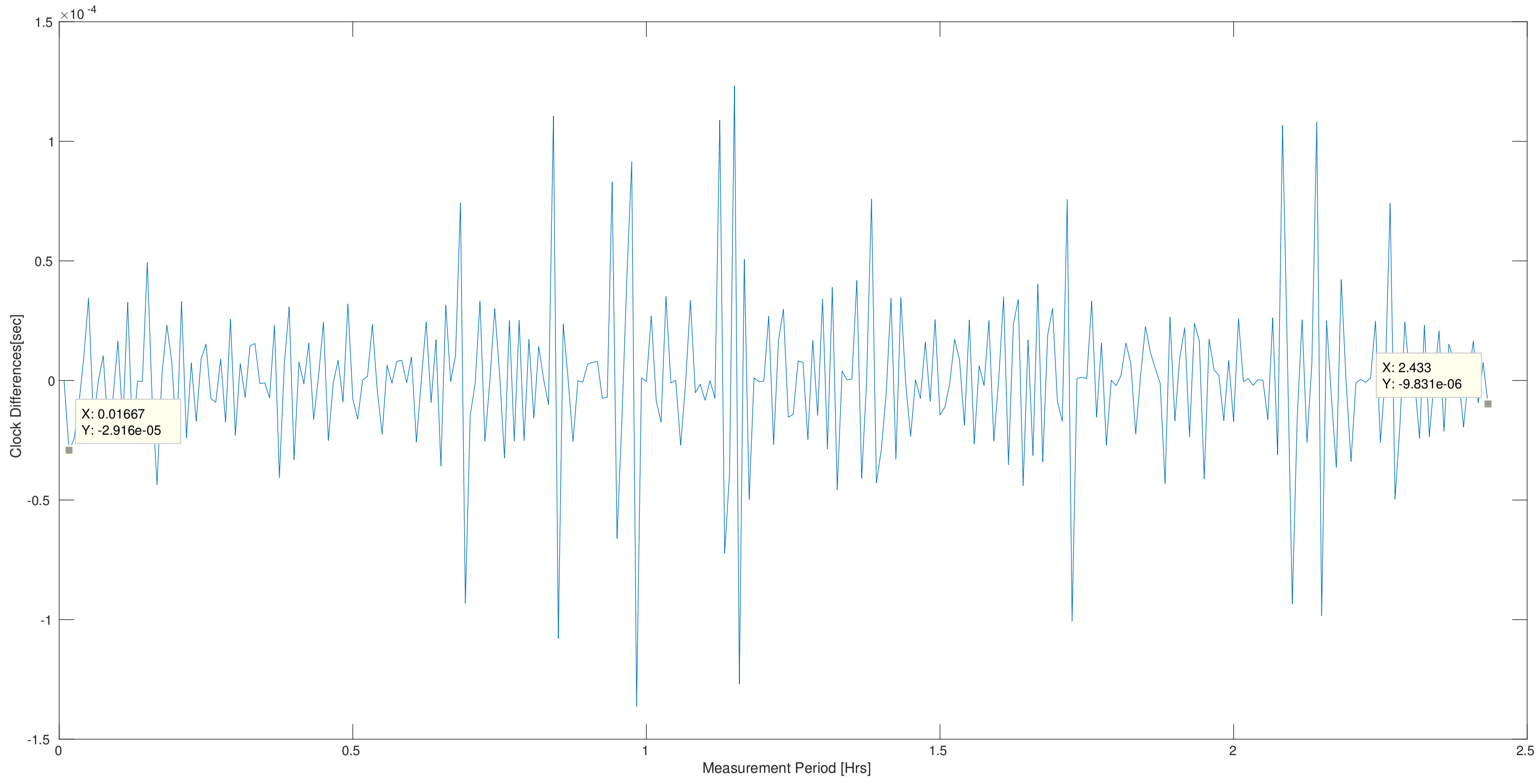


Figure 4.21: Similar Computers Over the PL Network: 292 Samples Fluctuations $(i - (i - 1))$ aklinux5→UOregon→UW→aklinux4

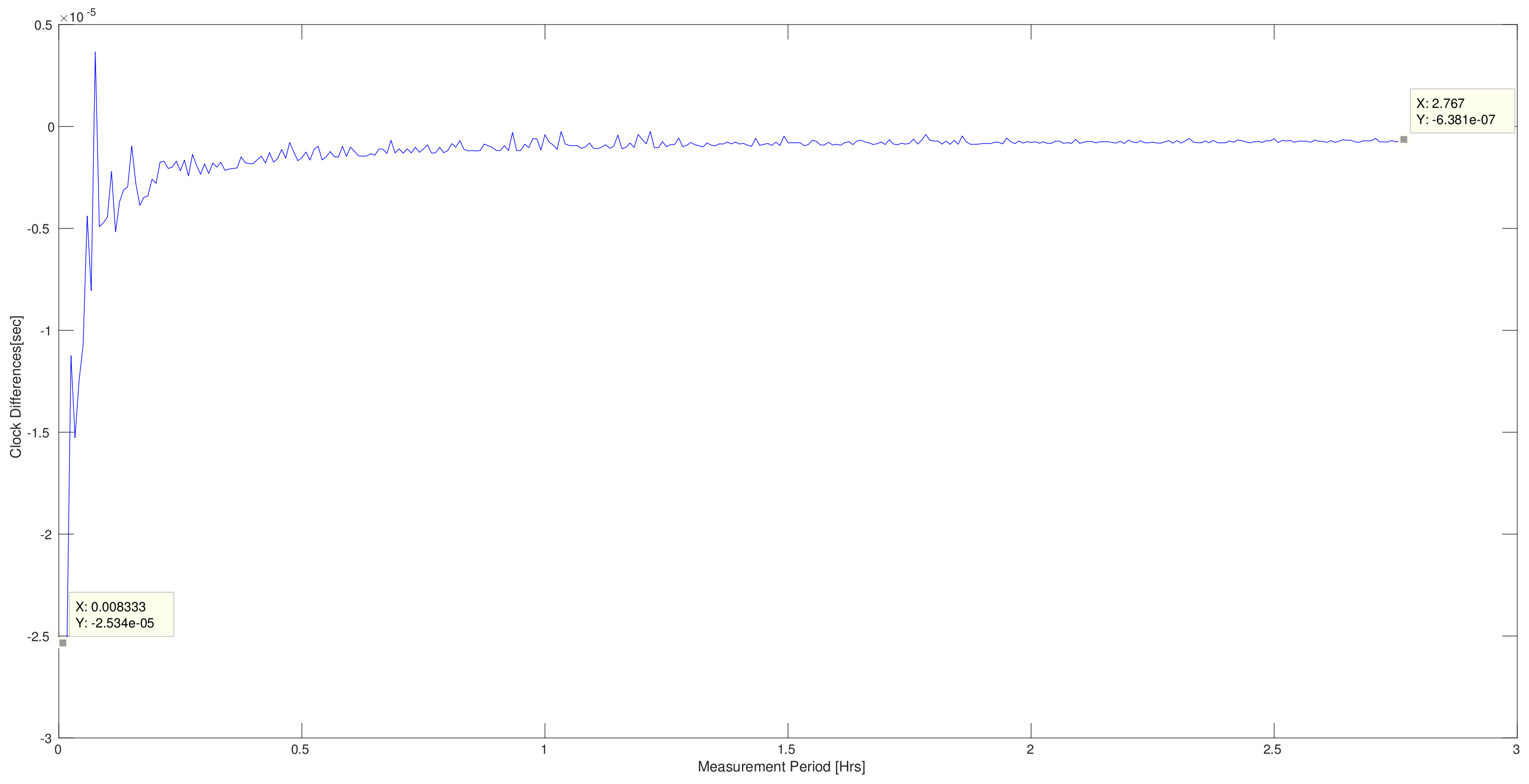


Figure 4.22: Similar Computers Over the PL Network Reversed: 333 Samples normalized average (last-first) aklinux4→UOregon→UW→aklinux5

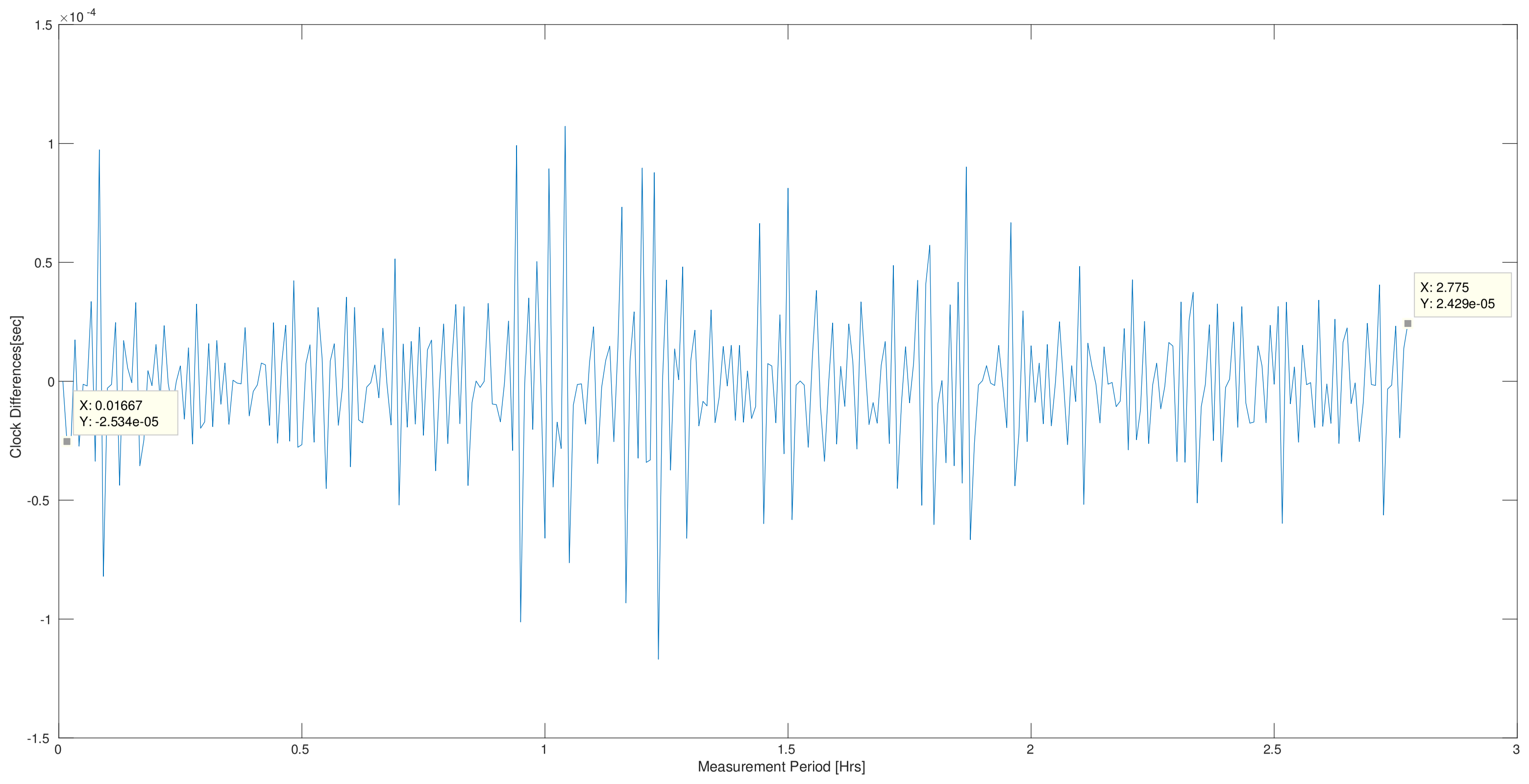


Figure 4.23: Similar Computers Over the PL Network Reversed: 333 Samples Fluctuations $(i - (i - 1))$ aklinux4→UOregon→UW→aklinux5

4.4.2 Dissimilar Computers

Continuing the experiments over the PlanetLab network Figure 4.24 shows the results of the first of experiments involving the dissimilar computers with a connection to the University of Oregon PlanetLab servers, and calculated per the method outlined in 4.2.1. Figure 4.25 displays the fluctuations in delay between each packet exchange.

As can be seen in Figure 4.24, the normalized difference over 1 second based on the last measurement is *-1.755e-5*. The calculated mean of the normalized difference of the difference clocks (i.e. mean of the fluctuations) is *-1.748e-5*, with the variance of the same figures being *1.105e-9*. Further, the results are collected over 280 sample points, with the period between each sample being 30 seconds, as calculated by the `CLOCK_REALTIME`. The duration of the experiment is approximately 2 hours and 20 minutes.

Figure 4.26 show the results of the same experiment, however, with the master-slave relationship reversed between the Test PCs. Here we have the previously transmitting computer receiving the packets, and vice-versa. Figure 4.27 renders the fluctuations between each sent packet. In Figure 4.26, the normalized difference over 1 second based on the last measurement is *1.725e-5*.

The results are collected over 342 sample points, with the period between each sample being 30 seconds. The duration of the experiment is approximately 2 hours and 20 minutes. The calculated mean of the normalized difference of the difference clocks (i.e. mean of the fluctuations) is *1.720e-5*, with the variance of the same figure being *8.272e-10*.

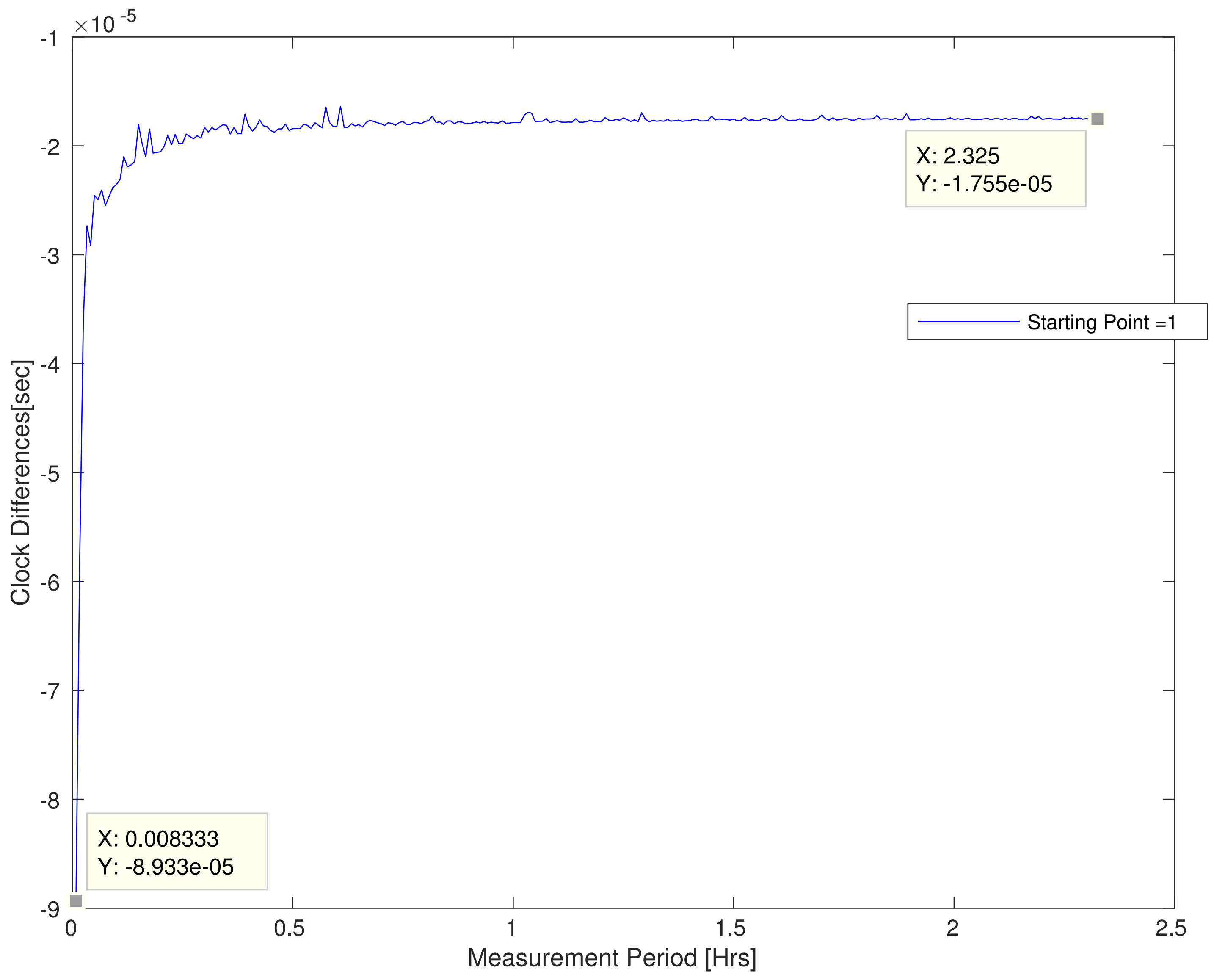


Figure 4.24: Dissimilar Computers Over the PL Network: 280 Samples normalized average (last-first) akkpc8→UOregon→UW→akkpc9

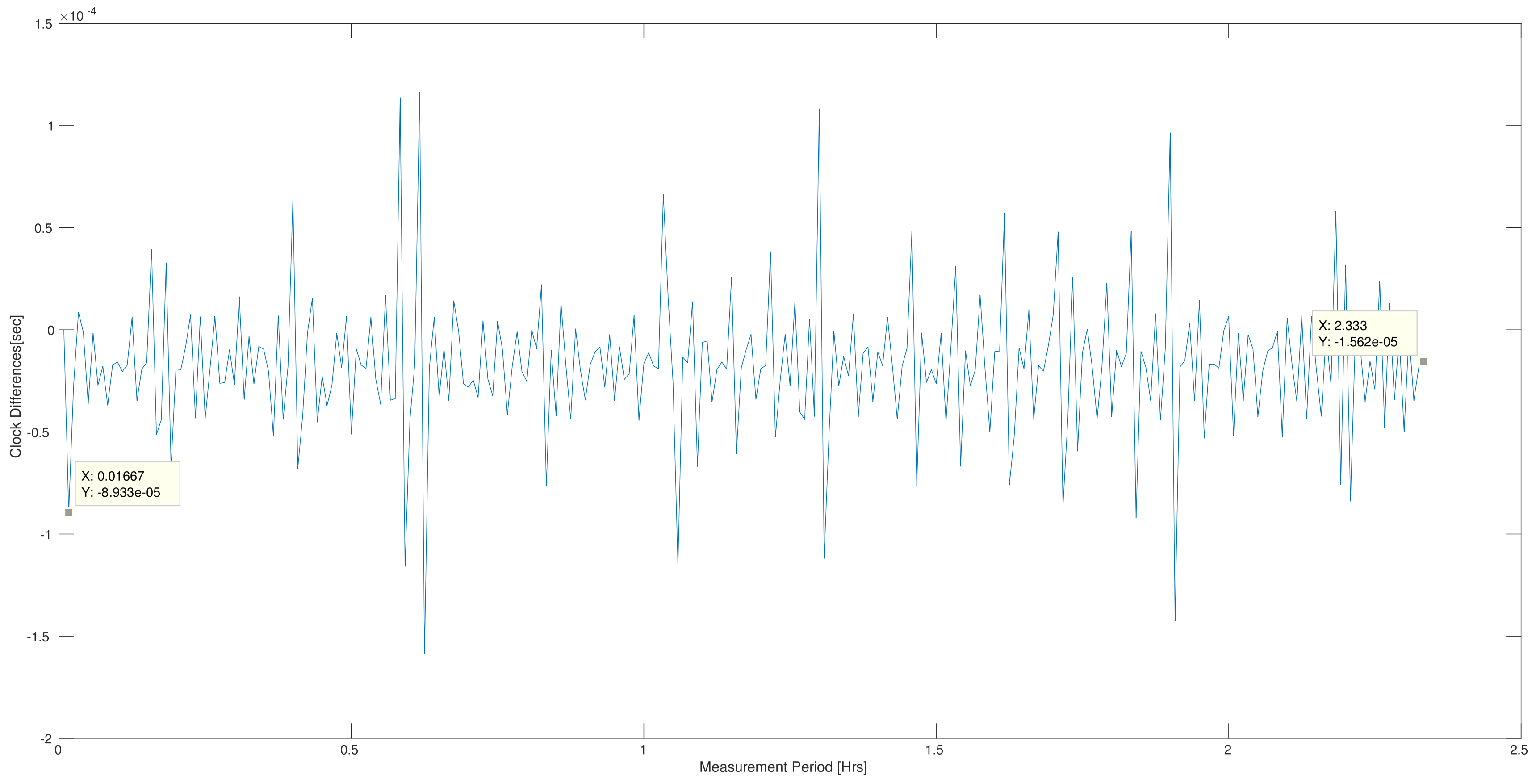


Figure 4.25: Dissimilar Computers Over the PL Network: 280 Samples Fluctuations $(i - (i - 1))$ akkpc8→UOregon→UW→akkpc9

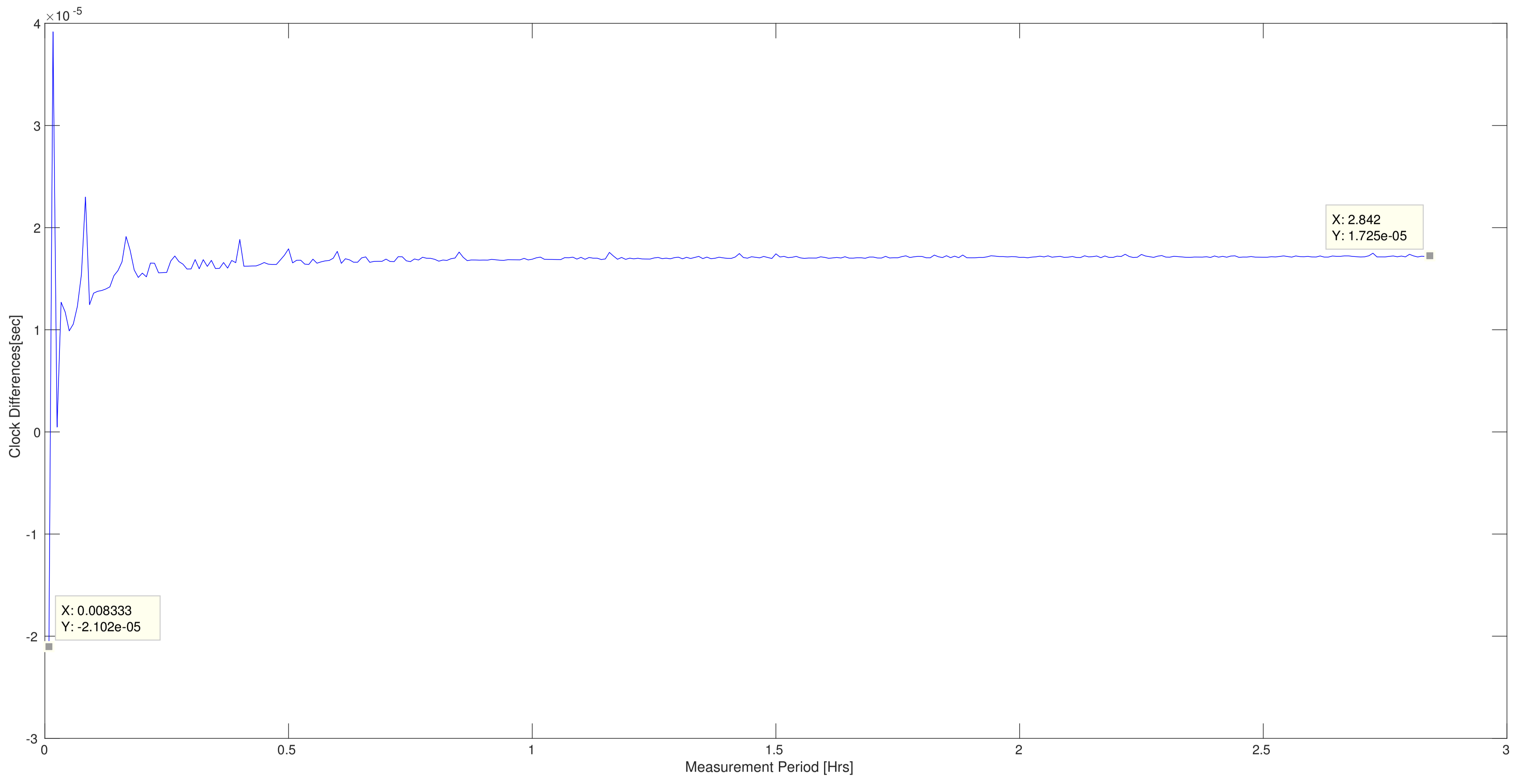


Figure 4.26: Dissimilar Computers Over the PL Network Reversed: 342 Samples normalized average (last-first) akkpc9→UOregon→UW→akkpc8

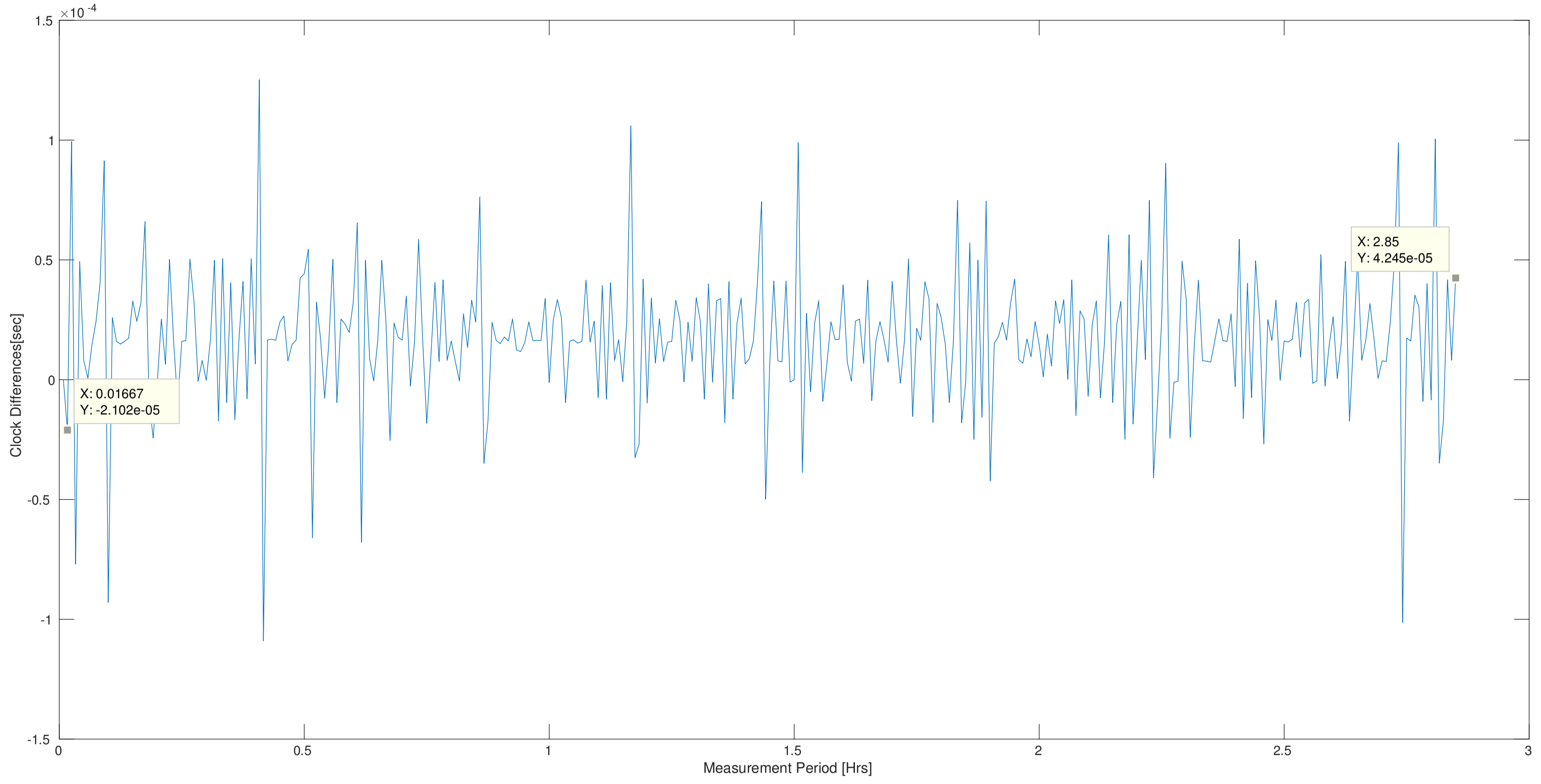


Figure 4.27: Dissimilar Computers Over the PL Network Reversed: 342 Samples Fluctuations $(i-(i-1))$ akkpc9→UOregon→UW→akkpc8

4.5 Establishing Time Accuracy Over the PlanetLab Network with 2 Nodes: University of Oregon, Monash Universty

This section, similar to Section 4.4, presents the experiments conducted using the PlanetLab network. Here we are utilizing the PlanetLab servers at the University of Oregon in Eugene, OR., and Monash University in Melbourne, Australia. The computers are connected to the servers at one of the two locations and then rerouted to other location, in order to extend the travel time. Essentially, all packets that arrive at one of the servers, are forwarded to the subsequent server. The tests are varied between two similar computers, and also two dissimilar computers. Further, a simple `traceroute` was used to see how many hops the computers needed to make, and there are nearly 46 hops between Test PCs, and their destination. The paths taken by each computer are different from one another. The breakdown of these paths are shown in B

4.5.1 Similar Computers

The following are the results of similar Test PCs connected over the PlanetLab network, starting with Figure 4.28, which displays the difference of the difference between the slave's last received packet and its first, and the master's last sent packet and its first, average and normalized over 1 second, and traced over time, as described in 4.2.1. Figure 4.29 displays the fluctuations in delay between each packet exchange.

As can be seen in Figure 4.28, the normalized difference over 1 second based on the last measurement is *-7.226e-07*. The calculated mean of the normalized difference of the difference clocks (i.e. mean of the fluctuations) is *-7.191e-7*, with the variance of the same figures being *1.268e-9*. Further, the results are collected over 211 sample points, with the period between each sample being 30 seconds, as calculated by the `CLOCK_REALTIME`. The duration of the experiment is approximately 1 hour and 45 minutes.

Figure 4.30 show the results of the same experiment, however, with the master-slave relationship reversed between the Test PCs. Here we have the previously transmitting computer receiving the packets, and vice-versa. Figure 4.31 renders the fluctuations between each sent packet. In Figure 4.30, the normalized difference over 1 second based on the last measurement is *6.155e-7*.

The results are collected over 573 sample points, with the period between each sample being 30 seconds. The duration of the experiment is approximately 4 hours and 50 minutes.

The calculated mean of the normalized difference of the difference clocks (i.e. mean of the fluctuations) is *6.144e-7*, with the variance of the same figure being *1.88e-9*.

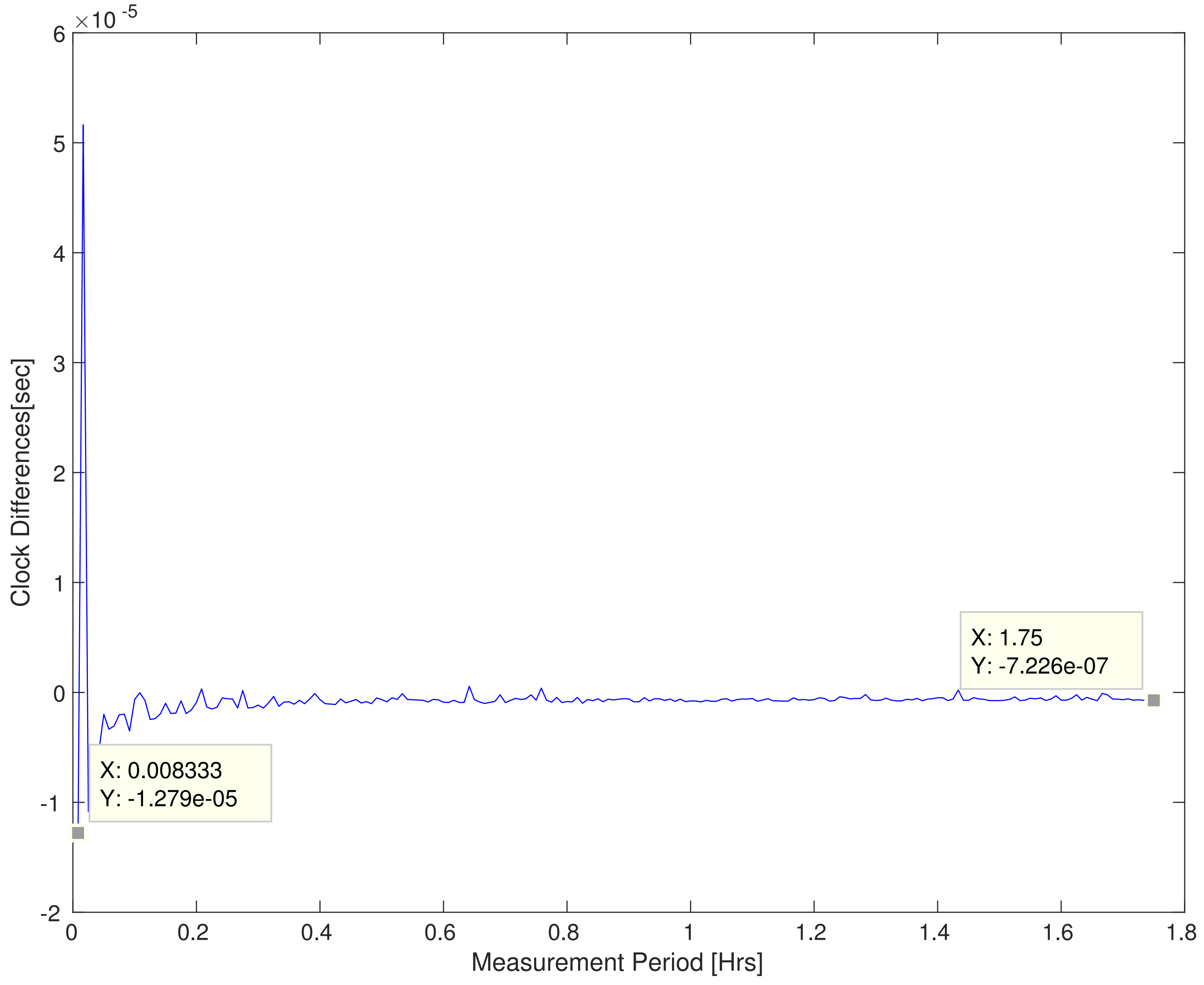


Figure 4.28: Similar Computers Over the PL Network: 211 Samples normalized average (last-first) aklinux4→Monash→UOregon→UW→aklinux5

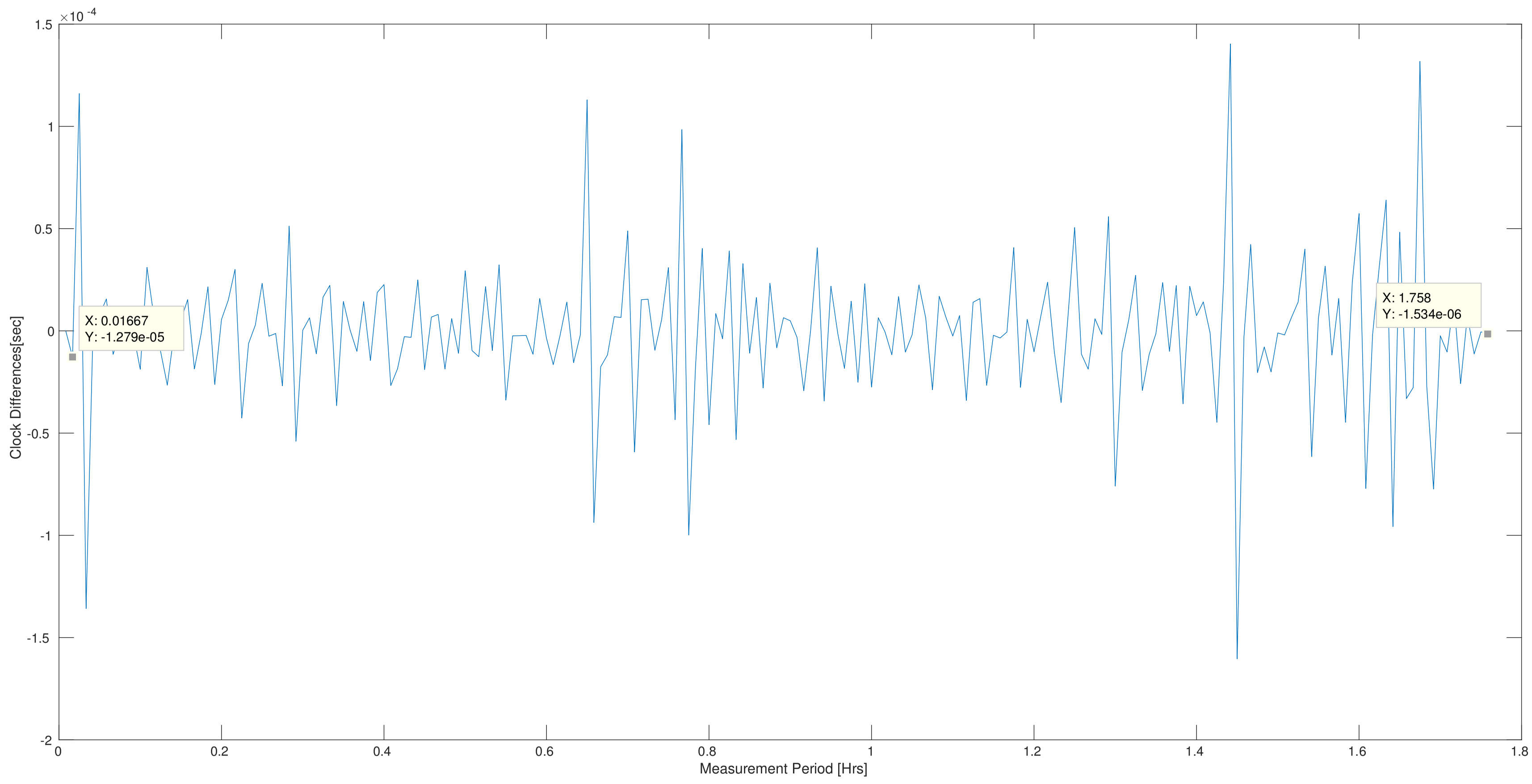


Figure 4.29: Similar Computers Over the PL Network: 211 Samples Fluctuations $(i - (i - 1))$ aklinux4→Monash→UOregon→UW→aklinux5

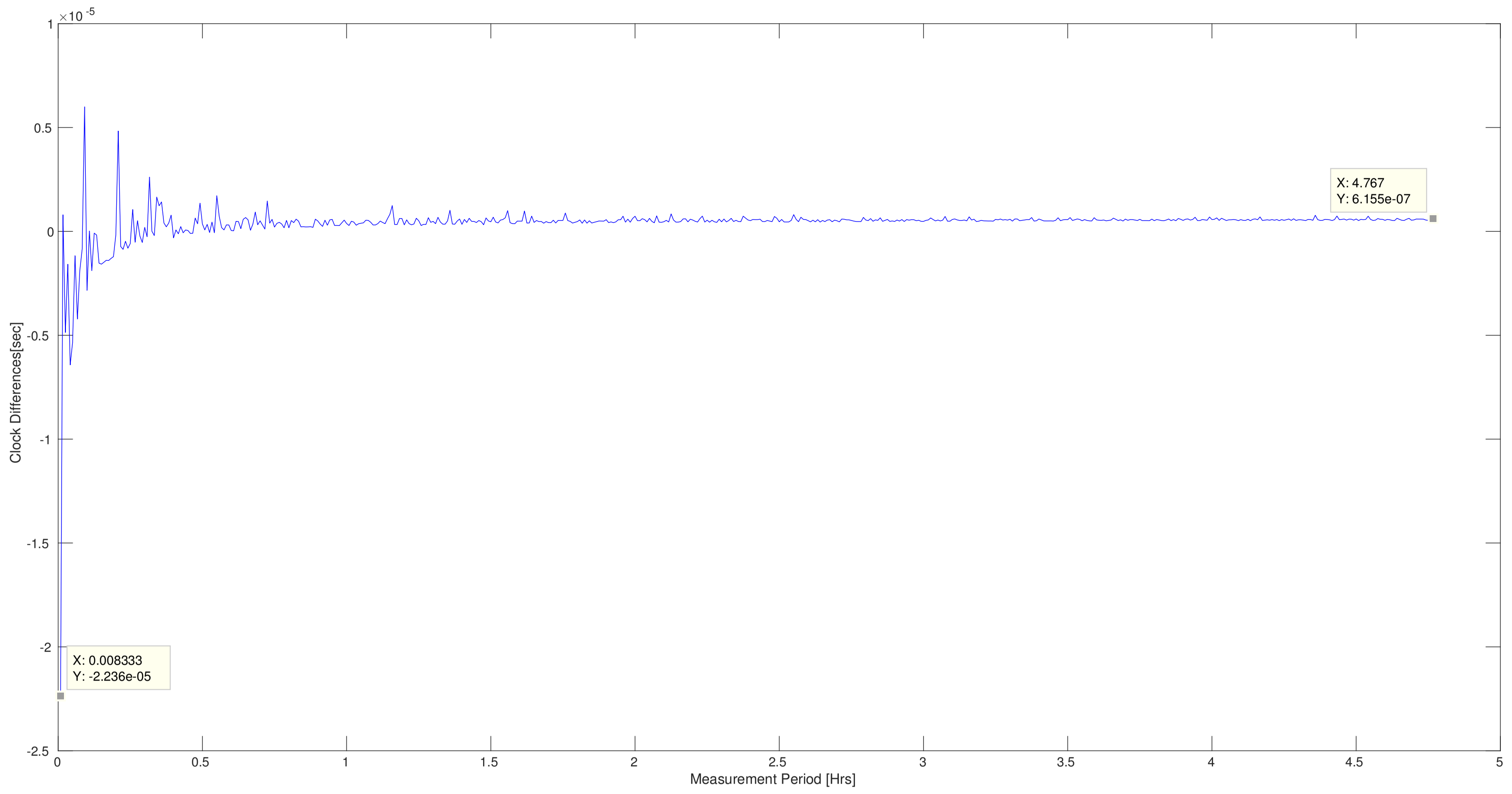


Figure 4.30: Similar Computers Over the PL Network Reversed: 573 Samples normalized average (last-first) akkpc5→UOregon→Monash→UW→akkpc4

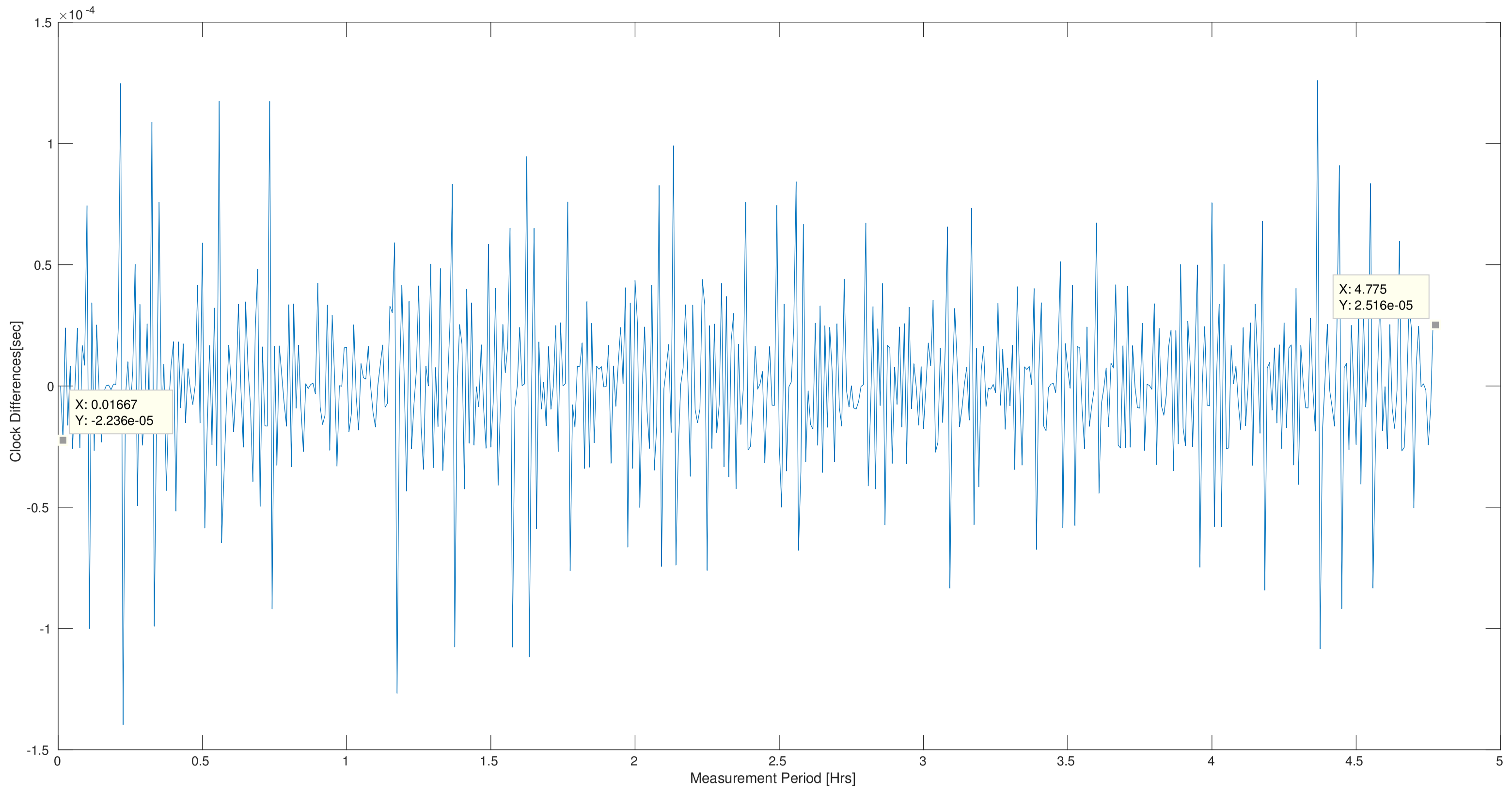


Figure 4.31: Similar Computers Over the PL Network Reversed: 573 Samples Fluctuations $(i - (i - 1))$ akkpc5→UOregon→Monash→UW→akkpc4

4.5.2 Dissimilar Computers

The following are the results of similar Test PCs connected over the PlanetLab network, starting with Figure 4.28, which displays the difference of the difference between the slave's last received packet and its first, and the master's last sent packet and its first, average and normalized over 1 second, and traced over time, as described in 4.2.1. Figure 4.29 displays the fluctuations in delay between each packet exchange.

Continuing the experiments over the PlanetLab network Figure 4.32 shows the results of experiments as outlined in 4.5 though involving the dissimilar computers, and calculated per the method outlined in 4.2.1. Figure 4.33 displays the fluctuations in delay between each packet exchange.

As can be seen in Figure 4.32, the normalized difference over 1 second based on the last measurement is *1.715e-5*. The calculated mean of the normalized difference of the difference clocks (i.e. mean of the fluctuations) is *1.707e-5*, with the variance of the same figures being *1.604e-9*. Further, the results are collected over 217 sample points, with the period between each sample being 30 seconds, as calculated by the `CLOCK_REALTIME`. The duration of the experiment is approximately 1 hour and 50 minutes.

Figure 4.34 show the results of the same experiment, however, with the master-slave relationship reversed between the Test PCs. Here we have the previously transmitting computer receiving the packets, and vice-versa. Figure 4.35 renders the fluctuations between each sent packet. In Figure 4.34, the normalized difference over 1 second based on the last measurement is *1.725e-5*.

The results are collected over 531 sample points, with the period between each sample being 30 seconds. The duration of the experiment is approximately 2 hours and 20 minutes. The calculated mean of the normalized difference of the difference clocks (i.e. mean of the fluctuations) is *1.720e-5*, with the variance of the same figure being *8.272e-10*.

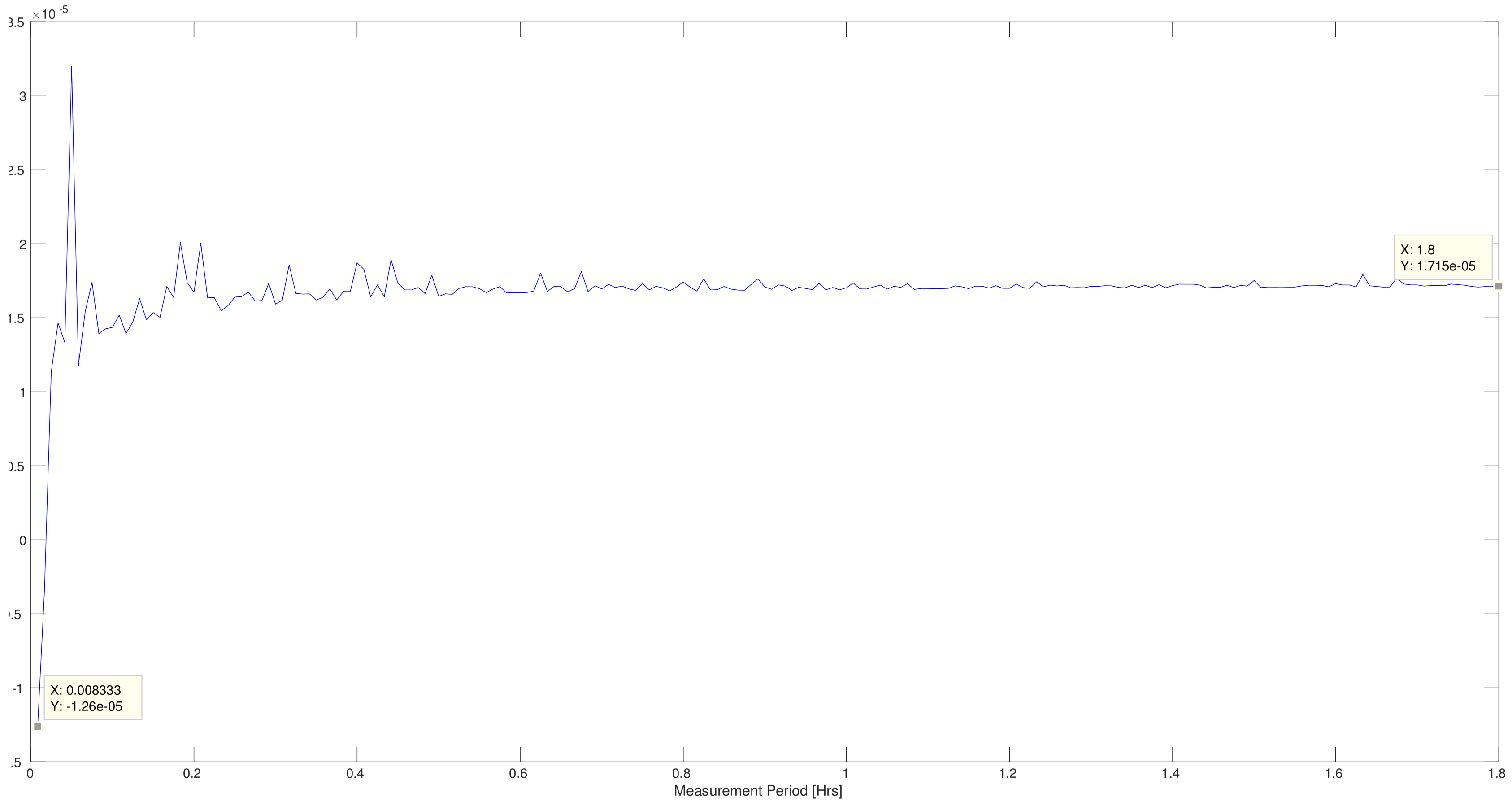


Figure 4.32: Dissimilar Computers Over the PL Network: 217 Samples normalized average (last-first) akkpc9→Monash→UOregon→UW→akkpc8

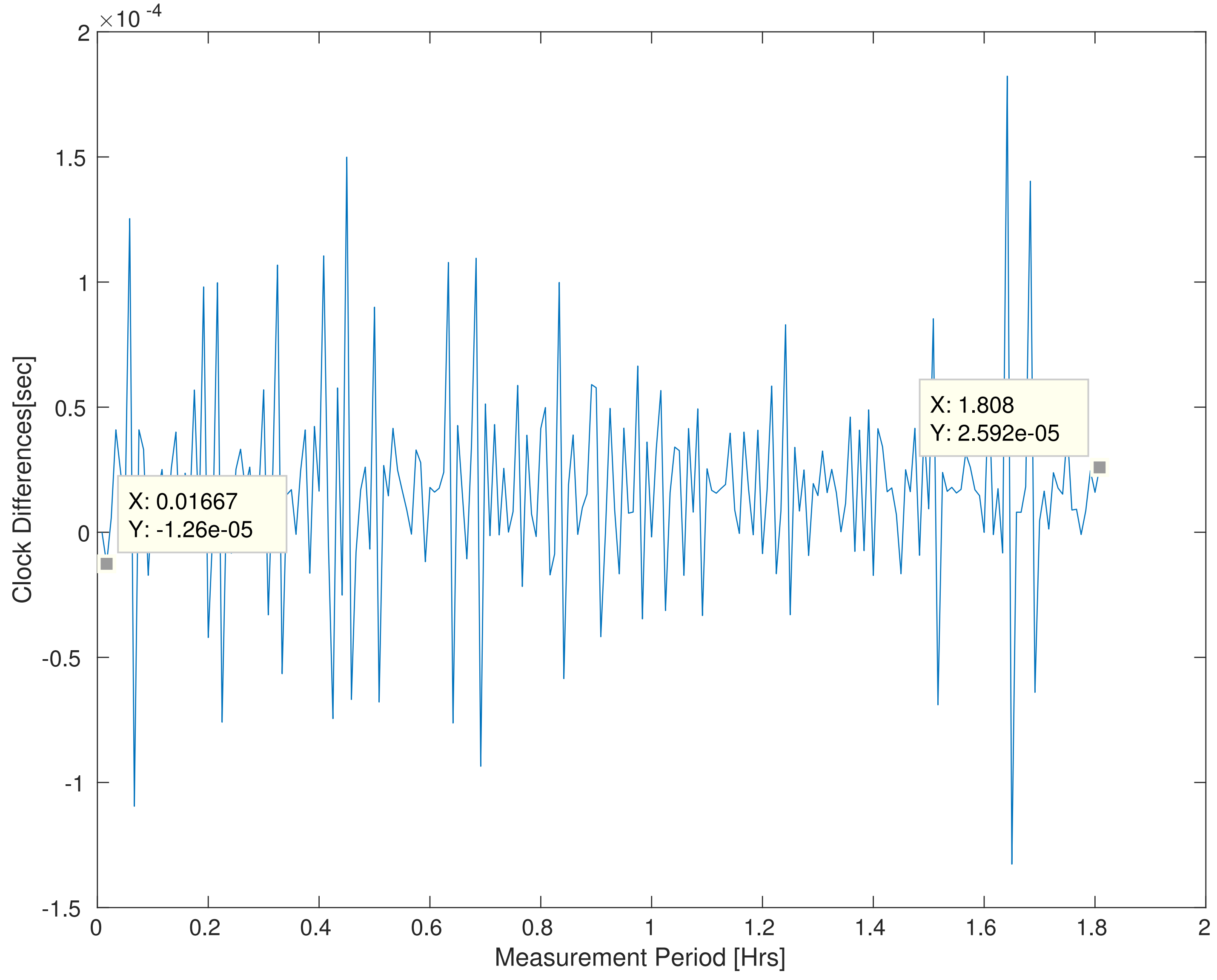


Figure 4.33: Dissimilar Computers Over the PL Network: 217 Samples Fluctuations $(i - (i - 1))$ akkpc9→Monash→UOregon→UW→akkpc8

4.6 Summary

Table 4.1 provides a summary of the values differences between test computers normalized to 1 second. Further, provides a breakdown of the number of samples gathered per experiment. The table summarizes the Normalized Difference (ND) between the difference of the master's first and last samples, and slave's.

Table 4.1: Summary of normalized values in difference between the Test PCs in per second

Test	**Similar DUTs**		**Dissmilar DUTs**	
	ND (/sec)	**Samples**	**ND (/sec)**	**Samples**
Direct	3.824e-7	2012	-1.703e-5	2557
Direct Reverse	-5.545e-7	3602	1.723e-5	3972
UW	-6.544e-7	1509	1.748e-5	1526
UW Reverse	6.55e-7	400	-1.731e-5	280
PLab: 1 Node	3.802e-7	292	-1.755e-5	280
Plab: 1 Node Rev	-6.381e-7	333	1.725e-5	342
PLab: 2 Nodes	-7.191e-7	211	1.715e-5	217
PLab: 2 Nodes Rev	6.155e-7	573	1.725e-5	531

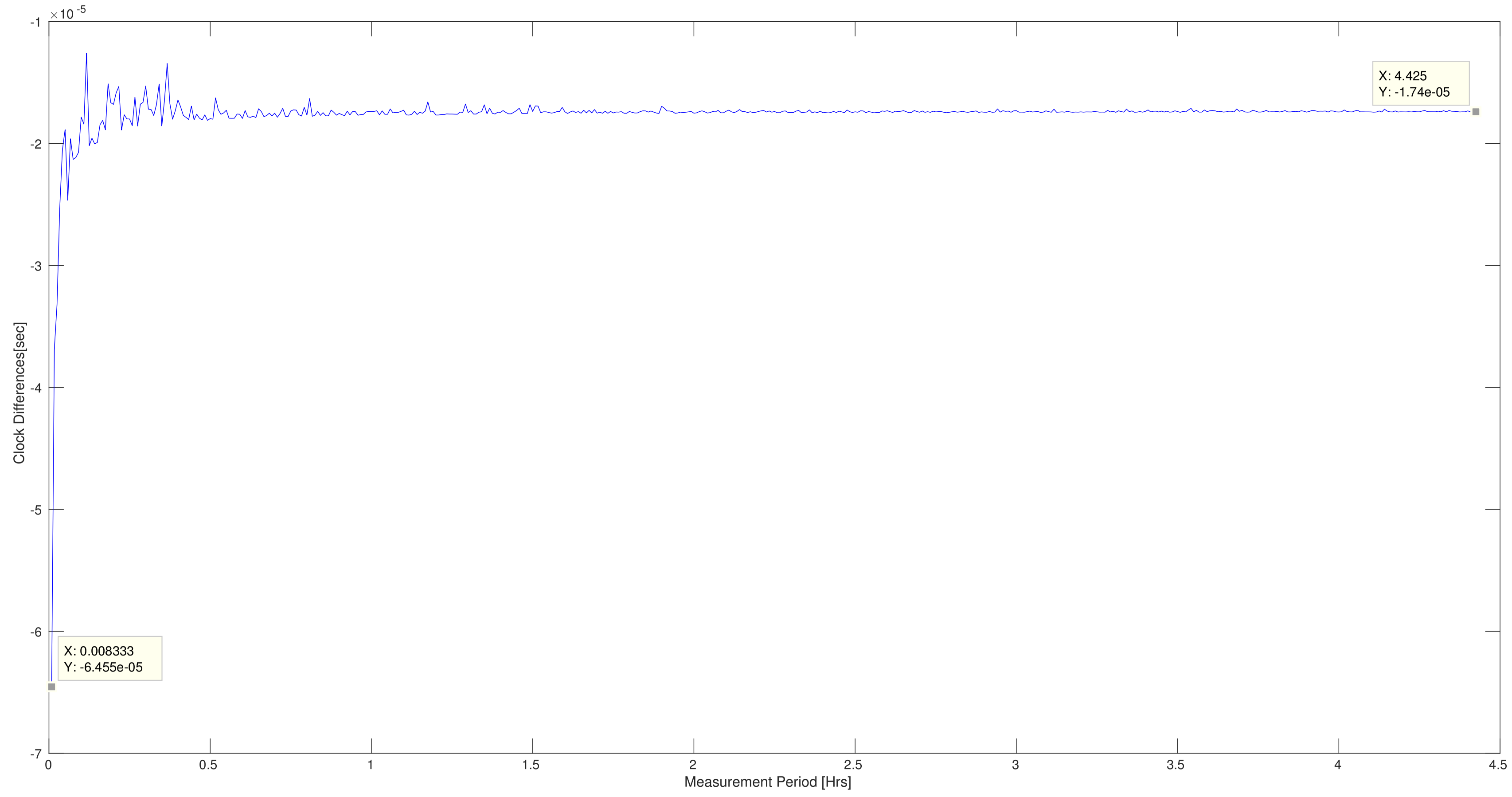


Figure 4.34: Dissimilar Computers Over the PL Network Reversed: 531 Samples normalized average (last-first) akkpc8→UOregon→Monash→UW→akkpc9

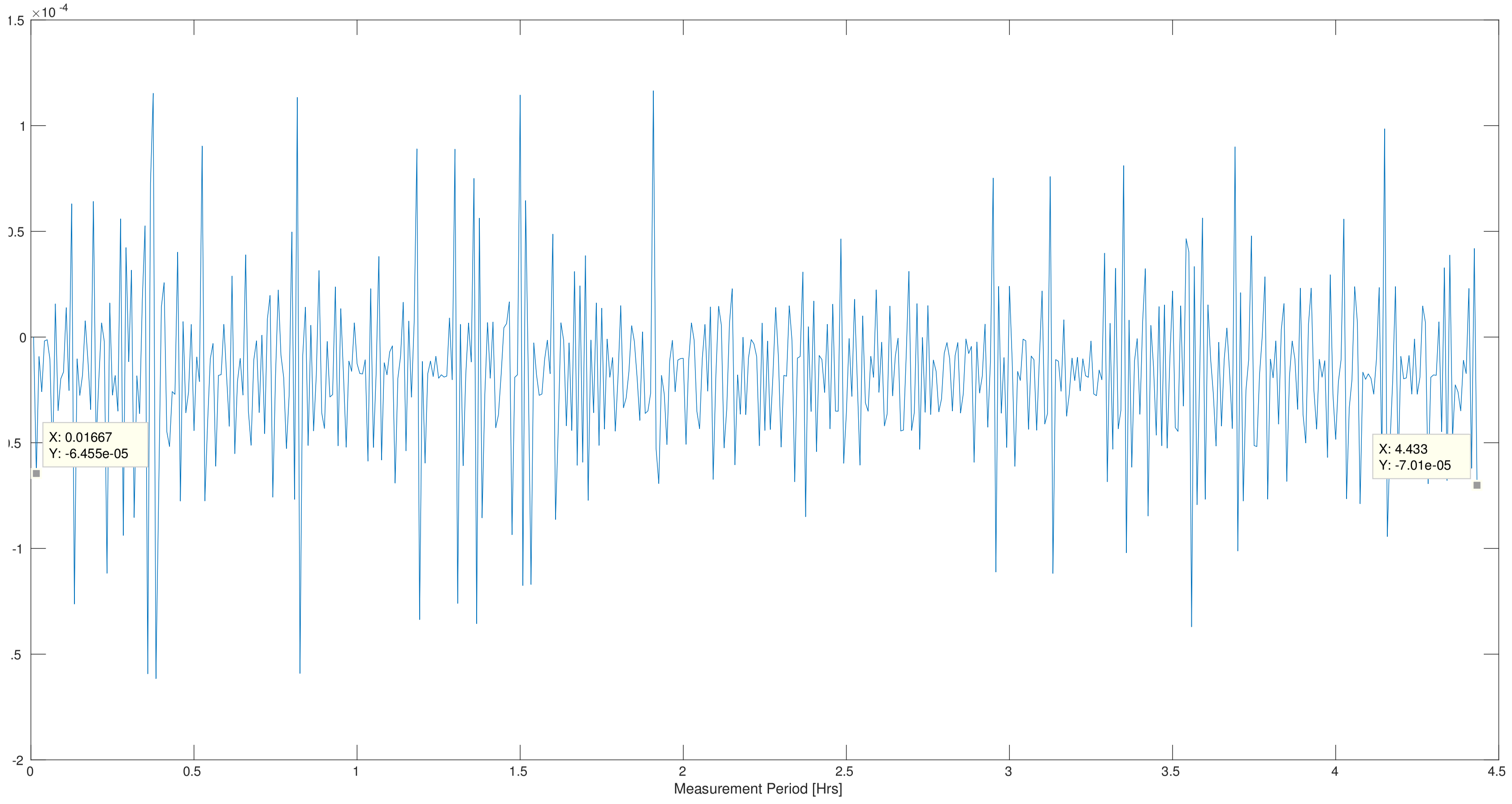


Figure 4.35: Dissimilar Computers Over the PL Network Reversed: 531 Samples Fluctuations $(i-(i-1))$ akkpc8→UOregon→Monash→UW→akkpc9

Chapter 5

Discussion and Conclusion

This section will provide a discussion on the results rendered hereinbefore.

5.1 Discussion

Network Time Protocol, is a comprehensive clock-synchronization method, complete with an Internet Engineering Taskforce standard, and multiple open-source implementations, including, but not limited to *chrony*, and a Request For Comments proposal already pending for an extension to the 4th version of the NTP [27]. The protocol's strengths reside in its ability for multiple fall-backs and contingency time-servers, security considerations and authentication protocols, and, above all, simplicity in implantation. The design can also be described as highly resilient against errors. While it is generally true that as the stratum level increases so does the degree of inaccuracy of the local clock compared to the reference clock, the lower-stratum clients can synchronize against multiple same or lower strata servers, and choose the most stable time source.

The setup procedure is highly simplified, especially if there exists unrestricted access to the Internet. The client need simply install the client software, provide the necessary administrative privileges, and the entire process becomes, essentially, *Plug and Play* (PnP). In specialized settings, or where modifications are of necessity, the adjustments are simple and objectively straightforward; the system administrator need only enter the sources of time, as required or desired, within a given framework or intranet.

In cases where a central reference clock is unavailable, or where the infrastructure is required to be siloed, or perhaps segregated from an external network, there is a protocol in place to appoint one time-server as the reference point and to create the subsequent strata. This allows for the intranet to be synchronized within their own framework, and allow for external correction once connected to a true external reference clock. This ability is desired in cases where there is an intermittent or non-ideal connection between the stratum 1 time-servers, and the reference clocks.

The centralized synchronization with the explicit use of a reference clock, and its inherent hierarchical design, while considered as strengths in design considerations, may, in fact, prove to be the Achilles heel of this protocol in certain specialized implementations. While, all things considered, these implementation scenarios are not the true, intended purpose of this, and other similar protocols, especially with their inherent centralized synchronization mechanism, it does leave room for improvement and further research.

Let us consider some scenarios where a centralized synchronization method may not be ideal, or practicable. There are many examples, from the highly integrated and interconnected, to the sparse and decentralized applications, where a centralized and hierarchical time-synchronization method can be undesirable, or at times outright impractical. In applications that involve monetary transactions, ranging from the mundane, such as in a shopping mall, to to the sophisticated such as High-Frequency Trading (HFT), timing is crucial, if not critical, in the execution of a transaction or a trade. While a common understanding or definition of time is a shared necessity in all scenarios involving monetary transactions, real-time time-synchronization is not the only way to achieve this common understanding of shared time.

If we consider the example of High-Frequency Trading for instance, there are numerous clients feeding into a centralized clearinghouse. The clearinghouses, then, prioritize the trades and resolve discrepancies between their clients, and, in turn, process their trades through their pertinent trading exchanges. While it is presumably true that most clients, and indeed most clearing houses have sophisticated mechanisms, and the network traffic bandwidth for processing these transactions, network delays, time-synchronization breakdowns, and dropped packets are a reality of Internet network traffic. As a result of such scenarios, it would be prudent to institute other means of reaching a common understanding of shared time.

As an example for decentralized applications, let us consider Internet of Things (IoT) devices. IoT devices, especially those used for data collection in research, may go days, weeks, or even months and years without being synchronized with a central time server. This may be due to a plethora of reasons; energy conservation in sparsely situated miniature probes is an issue of concern in large, real-world data collection efforts. Another reason may be the remoteness of these devices. Lack of access to active, continuous time-synchronization is a legitimate concern and may prove detrimental in research.

5.2 Contribution and Proposed Implementation

System time, used here in a general system and architecture agnostic manner, can be derived from several sources, including the Time Stamp Counter, crystal oscillator on a Real Time Clock (RTC) Integrated Circuit (IC), and the High Precision Event Timers. Therefore, the progression of time, and time-keeping in general, can vary from device to device, depending on hardware limitations, and software implementations, as well as fall-back procedures in place in cases of failure. As a result, time progresses in a non-universal method, especially if used in cross-platform settings, with non-uniform, non-standard hardware.

We have shown here that a reasonable and credible drift factor can be attained. We have succeeded in demonstrating that two computers can establish how fast one's clock progresses compared with another's. While it has long been clear that it is difficult to synchronize two system clocks, we have been able to demonstrate that the drift factor itself can be measured to a reliable degree, and utilized in synchronizing the two systems absent a reliable connection between the two. While modern systems do utilize the wander factor, it is not actively used in synchronization.

The implementation of this method enables users of this method to calculate the correct time at the time of time-stamping (or after the operation has completed) relative to another computer. The idea can be described thus: two computers can measure their relative time progression differences. Both systems will retain these measured time wander. The non-primary system, or slave system, synchronizes its local time with the primary, or master, system. The systems are subsequently disconnected from one another. The master can check in with the slave from time to time to re-synchronize, collect the data, etc. However, the point remains that at this point both the master *and* the slave is capable of approximating, with good accuracy, the local time of the other computer. If the slave computer is collecting data and wishes to time-stamp the data based on the local time of the master system, it can find out the elapsed time since disconnection, and apply the drift factor based on the measurement. Similarly, the master computer can, at time of data collection or upon successful reestablishing connection with the slave, ex post facto.

5.3 Conclusion

In this thesis, we have studied the behavior of the local clocks of different computers and established that the local times of two systems can vary. We have further established that the factor by which two computer's local times varies can be measured with great accuracy. Finally, we have shown that this factor is relatively stable. We have hypothesized that these methods can be utilized to predict, and subsequently correct, with a high degree of reliability, the time of an associated computer. It should be noted that while the benefits of this method may be minimal on the small scale, when applied on large scales, can be of great worth.

Active time-synchronization of hundreds, if not thousands, of systems, is difficult and arduous work. Since in the applications envisioned, there are many systems in need of time-synchronization relative to a central system, with scant or no active connectivity, and where active-synchronization may fail or be delayed to many factors, this method can be of great benefit. The method offered here also shows that the devices that would use this method can benefit from lower system overhead. The lower system over-head, then frees up the limited resources, be it bandwidth, memory, energy, to be used where they would be of greater use.

Chapter 6

Future Work

The method, and the underlying research, presented here create a solid foundation for expansion and further development. Here we outline the recommended work for any further research and development.

First, the methods outlined here need to be tested further on different platforms in order to ensure that the method would be functional in a cross-platform setting. It is important to note that, especially if specialized field testing settings are to be considered, the available hardware may not permit for the presence of an RTC on the platform. The same tends to apply to hand-held devices.

As a corollary of the above, any test much also factor in newer processor architectures. As an example, the impact of multi-core devices with turbo boost functionalists must be tested in order to see if any difference can be seen in the performance. These factors may skew the results.

Second, further research needs to be conducted on the impact of the surrounding environment on the outcome of such methods. Temperature, humidity, and other environmental factors can have detrimental consequences on the operation of a computer system, regardless of the presence of piezoelectric elements such as those used in RTC oscillator circuits.

Third, any real-world implementation of this method needs to consider network travel time, and factor this delay into the calculation. The method will further need to be tested within broader, more complex, network structures. Any robust time synchronization method must be robust enough to handle issues such as dropped packets, packet reordering, network path failures, and further tested on varied network load and usage conditions.

Lastly, a cost-benefit analysis should be conducted with respect to the system overhead, and performance impacts in comparison to such implementations as Network Time Protocol, and Precision Time Protocol. This should be tested in real-time time-stamp correction, as well as corrections.

References

[1] About. https://www.planet-lab.org/consortium. Accessed: 2018-10-23.

[2] Intel 82574l gigabit ethernet controller. https://ark.intel.com/products/32209/Intel-82574L-Gigabit-Ethernet-Controller?q=82574L. Accessed: 2018-10-30.

[3] clock_gettime(3): clock/time functions - Linux man page, Nov 2018. [Online; accessed 27. Nov. 2018].

[4] Timekeeping and clocks faqs, May 2018.

[5] Byron E Blair. *Time and frequency dissemination: an overview of principles and techniques*, volume 140, page 233314. Citeseer, 1974.

[6] Danny Dolev, Joseph Y Halpern, and H Raymond Strong. On the possibility and impossibility of achieving clock synchronization. *Journal of Computer and System Sciences*, 32(2):230–250, 1986.

[7] J. Elson and D. Estrin. Time synchronization for wireless sensor networks. In *Proceedings 15th International Parallel and Distributed Processing Symposium. IPDPS 2001*, pages 1965–1970, April 2001.

[8] IBM. timed daemon.

[9] Intel Corporation. Intel core i7-3770 processor. https://ark.intel.com/products/65719/Intel-Core-i7-3770-Processor-8M-Cache-up-to-3-90-GHz-. Accessed: 2018-12-02.

[10] Intel Corporation. Intel core2 duo processor e6550. https://ark.intel.com/products/30783/Intel-Core-2-Duo-Processor-E6550-4M-Cache-2-33-GHz-1333-MHz-FSB-?q=E6550. Accessed: 2018-10-29.

[11] Intel Corporation. Intel xeon processor e5405. https://ark.intel.com/products/33079/Intel-Xeon-Processor-E5405-12M-Cache-2-00-GHz-1333-MHz-FSB-?q=E5405. Accessed: 2018-10-29.

[12] Intel Corporation. Intel xeon processor e5420. https://ark.intel.com/products/33927/Intel-Xeon-Processor-E5420-12M-Cache-2-50-GHz-1333-MHz-FSB-. Accessed: 2018-10-28.

[13] Intel Corporation. Intel xeon processor e5620. https://ark.intel.com/products/47925/Intel-Xeon-Processor-E5620-12M-Cache-2-40-GHz-5-86-GT-s-Intel-QPI-. Accessed: 2018-10-28.

[14] Intel Corporation. Intel xeon processor x3210. https://ark.intel.com/products/28033/Intel-Xeon-Processor-X3210-8M-Cache-2-13-GHz-1066-MHz-FSB-?q=X3210. Accessed: 2018-10-29.

[15] Intel Corporation. Intel xeon processor x3323. https://ark.intel.com/products/35336/Intel-Xeon-Processor-X3323-6M-Cache-2-50-GHz-1333-MHz-FSB-?q=X3323. Accessed: 2018-10-29.

[16] Intel Corporation. Intel xeon processor x3330. https://ark.intel.com/products/35432/Intel-Xeon-Processor-X3330-6M-Cache-2-66-GHz-1333-MHz-FSB-?q=X3330. Accessed: 2018-10-29.

[17] Intel Corporation. Intel core2 duo processor e6850. https://ark.intel.com/products/30785/Intel-Core-2-Duo-Processor-E6850-4M-Cache-3-00-GHz-1333-MHz-FSB-?q=E6850. Accessed: 2018-10-30.

[18] Intel Corporation. *IA-PC HPET (High Precision Event Timers) Specification. Revision: 1.0a*, October 2004.

[19] Intel Corporation. *Intel 64 and IA-32 Architectures Software Developers Manual. Volume 3B: System Programming Guide, Part 2*, September 2016.

[20] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, 1978.

[21] Leslie Lamport and P Michael Melliar-Smith. Synchronizing clocks in the presence of faults. *Journal of the ACM (JACM)*, 32(1):52–78, 1985.

[22] Marvell Technology Group. Yukon gigabit ethernet controller with integrated phy. https://datasheet.octopart.com/88E8056-A2-NNC1C000-Marvell-datasheet-17899617.pdf. Accessed: 2018-10-30.

[23] Mazi Esfahani, Milad. Amshare: Distributed frequency synchronization of small cell base transceiver stations utilizing commercial am wireless signals. Master's thesis, 2017.

[24] David Mills. Network time protocol (version 3) specification, implementation and analysis. Technical report, 1992.

[25] David Mills, Jim Martin, Jack Burbank, and William Kasch. Network time protocol version 4: Protocol and algorithms specification. Technical report, 2010.

[26] David L Mills. Internet time synchronization: the network time protocol. *IEEE Transactions on communications*, 39(10):1482–1493, 1991.

[27] T. Mizrahi and D. Mayer. Network time protocol version 4 (ntpv4) extension fields. RFC 7822, RFC Editor, March 2016.

[28] Larry Peterson, Steve Muir, Timothy Roscoe, and Aaron Klingaman. PlanetLab Architecture: An Overview. Technical Report PDN–06–031, PlanetLab Consortium, May 2006.

[29] Surendra Rahamatkar, Ajay Agarwal, and Narendra Kumar. Analysis and Comparative Study of Clock Synchronization Schemes in Wireless Sensor Networks. 2010.

[30] Barbara Simons. An overview of clock synchronization. In *Fault-Tolerant Distributed Computing*, pages 84–96. Springer, 1990.

Glossary

Clock Condition If any event a comes before b, then a has occurred in the past, relative to b. Therefore, for any events a, b: if $a \to b$, then $C\langle a\rangle < C\langle b\rangle$. 7

Coordinated Universal Time The universal time agreed upon and implemented in 1970 and 1972, respectively, and currently defined by ITU Recommendation ITU-R TF.460-6. It is not adjusted for daylight savings time 12

Crossover cable A simple patch cable where the bi-directional data wires are switches on either side in order to facilitate the direct connection between two Network Cards. These cables were required in order network cards, however, the new generation of NICs are typically capable of handling direct connections without the need for specialized cables. 30

Distributed networks A collection of distinct entities which are spatially separated, and which communicate with one another through a communication network, 5

Epoch Universally agreed upon start of time in all computing systems beginning at midnight January 1, 1970. 2

High-Frequency Trading High speed securities trades that are effectuated using highly complex computer algorithms, and are often associated with high-speed computers, high volumes, and high securities turn-over rates, etc. 79

International Atomic Time In 1967 a second was defined as the period totaling 9,192,631,770 cycles of the radiation, which is the transition of two energy levels from the ground state of the Cesium 133 atom. The TAI is calculated by tabulating a weighted average of more than 300 atomic clocks. 12

Network Time Protocol A major standards developed to address the challenges involved with time synchronization. 2, 10, 78, 83

node one or more servers, capable of hosting one or more virtual machine 22

Partial ordering It is sometimes not possible to determine whether a happened before b, due to a number of externalities. As a result, in relation to an event, it is possible to determine that some events did indeed take place prior to another, though the rest are indeterminable. 5, 6, 8

Payload The *data* portion a sent packet, with the header section, or other meta-data excluded 29

PlanetLab A collection of computers, and servers, located in various locations around the world which enable researchers in academic, industrial, and government institutions to be connected, and have access, to systems around the world in order to conduct research on the nature of the internet, and connectivity. 20

Portable Portability is ability to transfer a computer program from one system to another without a need for modification. A program that is system, or OS agnostic is portable. 11

Sensor Network Specialized sensor networks that can be set up without an existing infrastructure, and as such need to communicate using a common notion of time. 4

slice A slice is a layer of abstraction, allowing the creation of a Virtual machine. 21

Virtual machine The emulation of a computer architecture which provides complete usability, and functionality. It further enables several environments to exist on one machine without interfering with one another, essentially placing the different environments into silos. 21, 22

Voltage-Controlled Oscillator An electronic oscillator whose oscillation frequency is controlled by a voltage input. 15

Wireless Sensory Network Similar to the Sensory Network, though connected without the use of hard connections, such as ethernet 4

Abbreviations

ACTS Automated Computer Time Service 11

GNSS Global Navigation Satellite System 11

IC Integrated Circuit 80

IoT Internet of Things 79

RTC Real Time Clock 80, 82

UDP User Datagram Protocol 19

List of Symbols

$\underline{\mathscr{I}}$ set of all events including $\mathscr{I}$, and all other relevant external events 8

$\mathscr{I}$ set of all system events 8

$\rightharpoonup$ the “happened before” relation for all $\underline{\mathscr{I}}$ 8

$\longrightarrow$ the *happened before* relation on the set of events of a system as the smallest relation which satisfies the 6

$\Rightarrow$ total ordering 8

APPENDICES

Appendix A

Additional Results

Here we are presenting the results of additional testing for similar test computers conducted over the network.

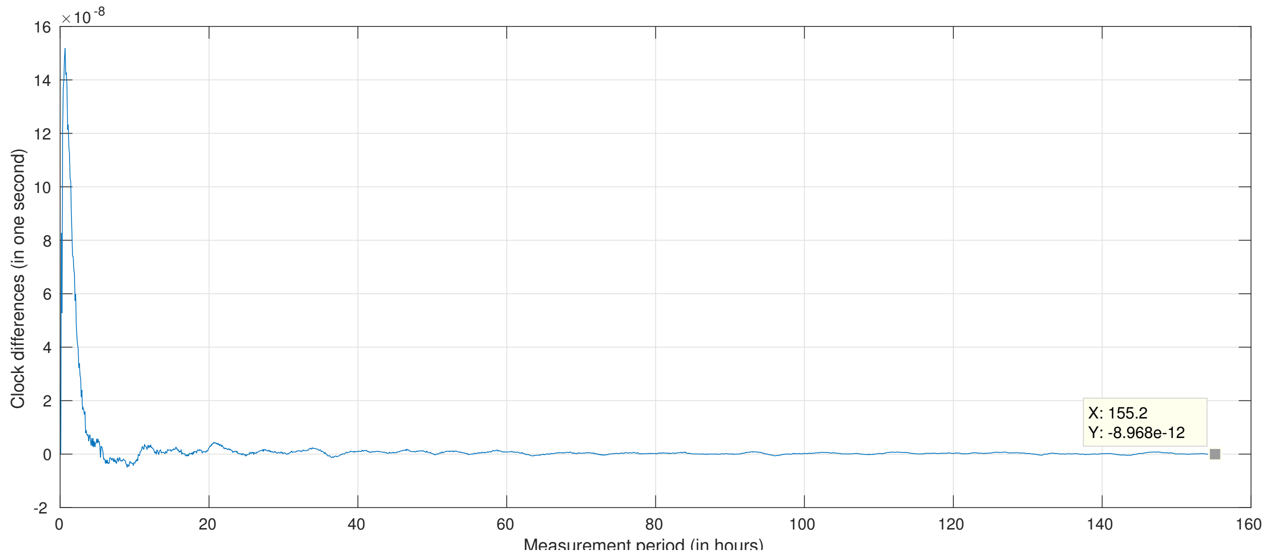


Figure A.1: The Time Accuracy between two Similar Test Computers

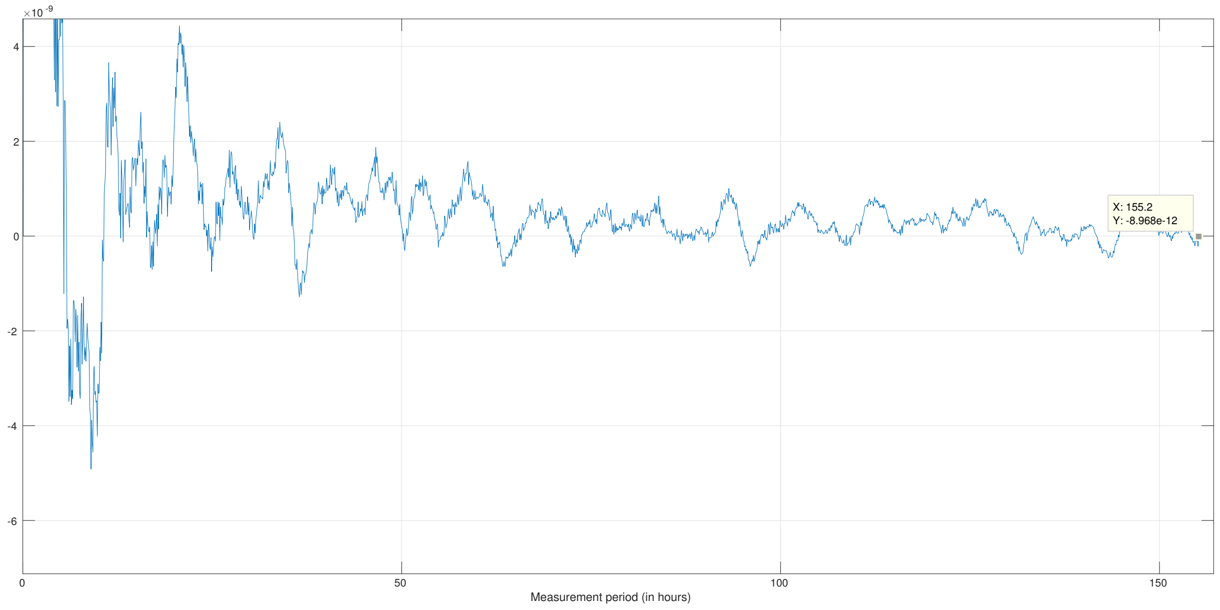


Figure A.2: The Close up of Time Accuracy between two Similar Test Computers in A.1

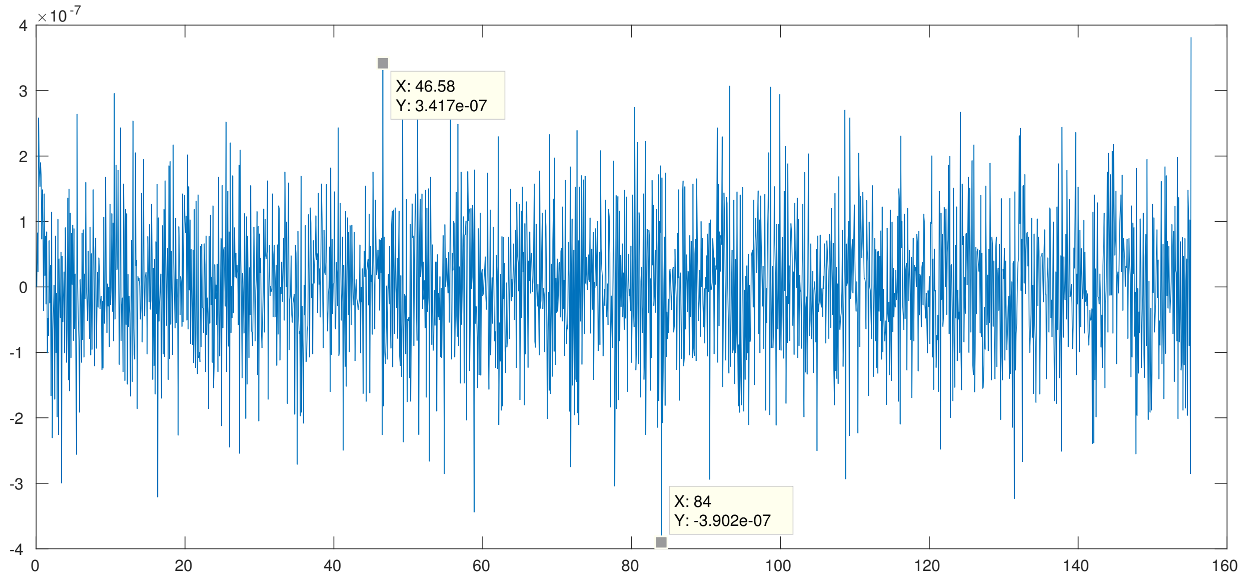


Figure A.3: The Fluctuation in Time Accuracy between two Similar Test Computers in A.1

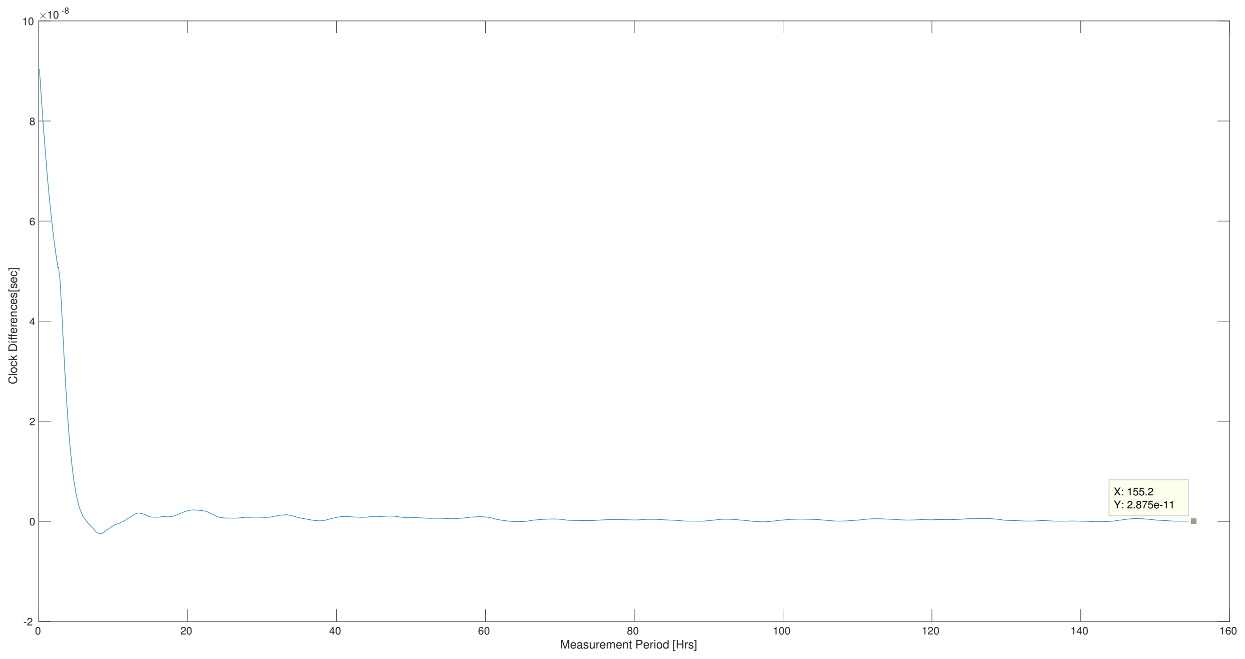


Figure A.4: Simple moving average with 60 sample window applied to A.1

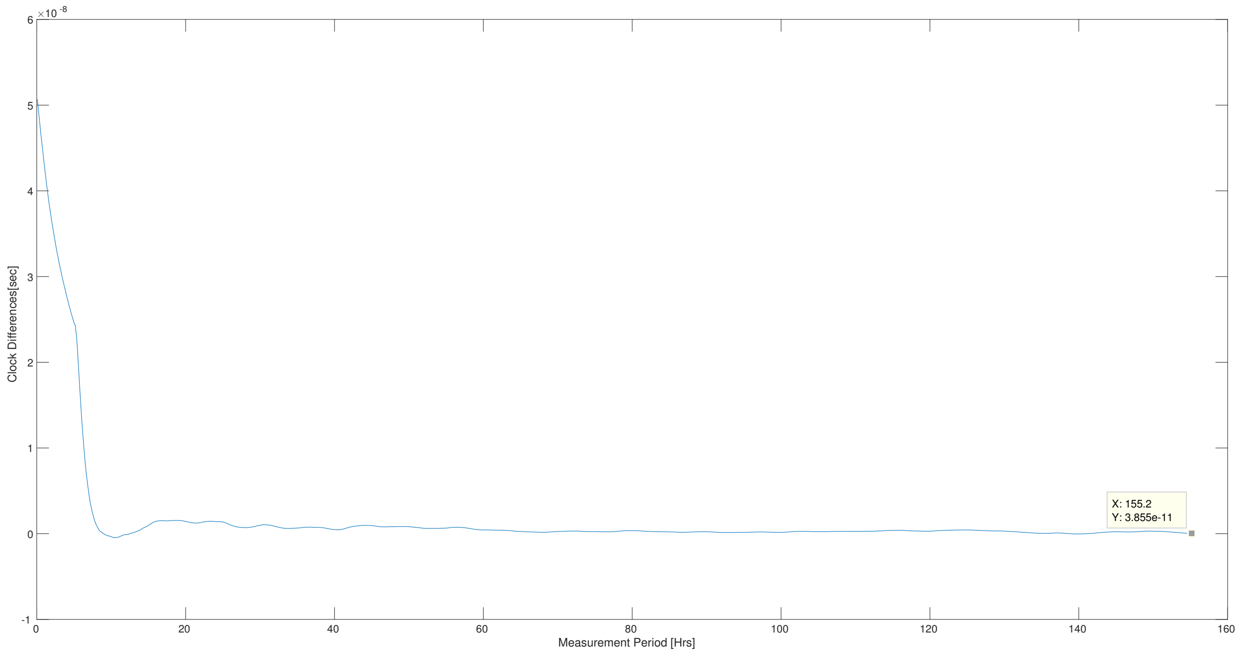


Figure A.5: Simple moving average with 120 sample window applied to A.1

Appendix B

Trace Route Results

Here, we present the result of *traceroute* measurements to see how many hops exist between nodes of importance.

B.1 *Traceroute* from PC9 to UW PlanetLab Server

```
traceroute to plink.cs.uwaterloo.ca (129.97.74.12), 30 hops max, 60 byte
 ↪  packets
 1  ceit-exsw01-ecenet.uwaterloo.ca (129.97.90.1)  0.885 ms  0.935 ms
  ↪  0.926 ms
 2  v490-eng-rt-e2.ns.uwaterloo.ca (172.16.32.193)  4.116 ms  4.145 ms
  ↪  4.170 ms
 3  te4-3-dist-rt-mc.ns.uwaterloo.ca (172.18.7.17)  0.621 ms  0.705 ms
  ↪  0.794 ms
 4  v490-mc-cs2.ns.uwaterloo.ca (172.16.15.2)  4.898 ms  4.924 ms  4.932
  ↪  ms
 5  v490-dc-cs2.ns.uwaterloo.ca (172.16.3.6)  7.436 ms  7.423 ms  7.442
  ↪  ms
 6  plink.cs.uwaterloo.ca (129.97.74.12)  0.218 ms  0.231 ms  0.296 ms
```

B.2 *Traceroute* from plink in UW→ ETH

```
traceroute to planetlab2.inf.ethz.ch (192.33.90.67), 30 hops max, 60 byte
packets
 1  dc3558-cs2-csnet.uwaterloo.ca (129.97.74.1)  6.243 ms  6.240 ms
  ↪  6.227 ms
 2  te2-7-dist-rt-phy.ns.uwaterloo.ca (172.16.3.5)  0.693 ms  0.680 ms
  ↪  0.896 ms
 3  xe1-0-0-u11-dist-sa-mc-trust.ns.uwaterloo.ca (172.31.0.149)  0.643 ms
  ↪  0.612 ms  0.606 ms
 4  te2-12-dist-rt-mc-global.ns.uwaterloo.ca (172.31.0.161)  1.590 ms
  ↪  1.577 ms  1.310 ms
 5  te2-16-cn-rt-mc.ns.uwaterloo.ca (172.16.31.117)  0.791 ms  0.770 ms
  ↪  0.993 ms
 6  gi0-0-1-ext-rt-rac.ns.uwaterloo.ca (172.16.31.109)  1.476 ms  3.750
  ↪  ms  3.927 ms
 7  216.191.167.37 (216.191.167.37)  2.410 ms  2.626 ms  2.666 ms
 8  199.212.160.218 (199.212.160.218)  2.956 ms  2.949 ms  2.932 ms
 9  ae2.zayo.mpr2.tor1.ca.zip.zayo.com (64.125.14.230)  2.975 ms  2.961
  ↪  ms  2.981 ms
10  ae13.cr1.lga5.us.zip.zayo.com (64.125.30.208)  84.372 ms  84.342 ms
 ↪  84.181 ms
11  ae5.cs1.lhr11.uk.eth.zayo.com (64.125.29.127)  84.166 ms  84.137 ms
 ↪  84.168 ms
12  ae0.cs1.lhr15.uk.eth.zayo.com (64.125.29.119)  84.404 ms  84.145 ms
 ↪  84.431 ms
13  ae2.cs1.ams10.nl.eth.zayo.com (64.125.29.16)  84.425 ms  84.364 ms
 ↪  84.206 ms
14  ae3.er1.ams1.nl.zip.zayo.com (64.125.31.105)  84.162 ms  84.128 ms
 ↪  88.907 ms
15  swiCE1-100GE-0-3-0-1.switch.ch (80.249.208.33)  97.135 ms  97.660 ms
 ↪  97.133 ms
16  swiCE4-100GE-0-0-0-0.switch.ch (130.59.36.6)  97.416 ms  97.390 ms
 ↪  97.387 ms
17  swiBE3-100GE-0-1-0-1.switch.ch (130.59.37.145)  99.637 ms  99.610 ms
 ↪  99.326 ms
```

```
18  swiBF1-100GE-0-0-0-1.switch.ch (130.59.39.78)  99.570 ms  99.558 ms
 ↪  99.530 ms
19  swiEZ3-100GE-0-1-0-0.switch.ch (130.59.37.6)  101.263 ms  101.259 ms
 ↪  101.220 ms
20  swiEZ1-P3.switch.ch (130.59.36.33)  100.947 ms  136.371 ms *
21  rou-open-net-switch.ethz.ch (192.33.92.161)  101.654 ms  101.650 ms
 ↪  101.094 ms
22  planetlab2.ethz.ch (192.33.90.67)  100.877 ms  100.818 ms  100.846 ms
```

B.3 *Traceroute* from plink in UW→ Monash

```
traceroute to pl1.eng.monash.edu.au (130.194.252.8), 30 hops max, 60 byte
 ↪  packets
 1  dc3558-cs2-csnet.uwaterloo.ca (129.97.74.1)  4.602 ms  4.614 ms
  ↪  4.602 ms
 2  te2-7-dist-rt-phy.ns.uwaterloo.ca (172.16.3.5)  0.829 ms  0.816 ms
  ↪  0.804 ms
 3  * xe1-0-0-u11-dist-sa-mc-trust.ns.uwaterloo.ca (172.31.0.149)  0.720
  ↪  ms *
 4  te2-12-dist-rt-mc-global.ns.uwaterloo.ca (172.31.0.161)  1.447 ms
  ↪  1.386 ms  1.377 ms
 5  te2-16-cn-rt-mc.ns.uwaterloo.ca (172.16.31.117)  1.117 ms  1.104 ms
  ↪  1.080 ms
 6  te0-0-2-0-ext-rt-mc.ns.uwaterloo.ca (172.16.32.149)  1.556 ms  1.439
  ↪  ms  3.703 ms
 7   (66.97.28.65)  1.642 ms  1.635 ms  1.451 ms
 8   (66.97.16.109)  4.934 ms  4.930 ms  4.739 ms
 9   (66.97.16.26)  4.705 ms  4.702 ms  4.941 ms
10  toro1rtr1.canarie.ca (205.189.32.41)  4.480 ms  4.480 ms  4.455 ms
11  wnpg1rtr1.canarie.ca (205.189.32.180)  25.708 ms  25.705 ms  25.678
 ↪  ms
12  wnpg2rtr1.canarie.ca (205.189.32.60)  25.924 ms  25.901 ms  25.864 ms
13  clgr2rtr1.canarie.ca (205.189.32.176)  40.111 ms  40.064 ms  40.086
 ↪  ms
```

```
14  vncv1rtr1.canarie.ca (205.189.32.174)  51.029 ms  51.031 ms  50.994 ms
15  vctr3rtr1.canarie.ca (205.189.32.198)  52.715 ms  52.709 ms  52.681 ms
16  sttl1rtr1.canarie.ca (205.189.32.182)  54.698 ms  54.670 ms  54.699 ms
17  aarnet-1-is-jmb-776.lsanca.pacificwave.net (207.231.241.149)  79.230 ms  79.145 ms  79.140 ms
18  et-1-2-1.pe1.a.koa.aarnet.net.au (113.197.15.86)  125.099 ms  125.100 ms  142.142 ms
19  et-1-0-0.pe1.tkpa.akl.aarnet.net.au (113.197.15.84)  213.301 ms  213.301 ms  213.110 ms
20  et-0-1-0.199.pe1.wnpa.akl.aarnet.net.au (113.197.15.70)  213.123 ms  213.097 ms  213.102 ms
21  et-2-1-0.pe1.sxt.alxd.nsw.aarnet.net.au (113.197.15.76)  213.828 ms  213.618 ms  213.301 ms
22  113.197.15.158 (113.197.15.158)  213.308 ms  213.347 ms  213.340 ms
23  xe-1-1-0.pe1.eskp.nsw.aarnet.net.au (113.197.15.199)  214.310 ms  216.095 ms  214.336 ms
24  et-5-3-0.pe1.wmlb.vic.aarnet.net.au (113.197.15.9)  225.808 ms  225.830 ms  225.795 ms
25  et-0-1-0.pe1.nbpk.vic.aarnet.net.au (113.197.15.27)  226.030 ms  226.098 ms  226.095 ms
26  xe-5-0-6-211.pe1.nbpk.vic.aarnet.net.au (138.44.64.221)  226.323 ms  226.361 ms  226.322 ms
27  * * *
28  * * *
29  pl1.eng.monash.edu.au (130.194.252.8)  226.379 ms  226.307 ms  226.340 ms
```

B.4 *Traceroute* from plink in UW→ UOregon

```
traceroute to planetlab4.cs.uoregon.edu (128.223.8.114), 30 hops max, 60
 ↪  byte packets
 1  dc3558-cs2-csnet.uwaterloo.ca (129.97.74.1)  5.603 ms  5.580 ms
  ↪  5.566 ms
 2  te2-7-dist-rt-phy.ns.uwaterloo.ca (172.16.3.5)  0.543 ms  0.772 ms
  ↪  0.779 ms
 3  xe1-0-0-u11-dist-sa-mc-trust.ns.uwaterloo.ca (172.31.0.149)  0.503 ms
  ↪  0.488 ms  0.457 ms
 4  te2-12-dist-rt-mc-global.ns.uwaterloo.ca (172.31.0.161)  1.440 ms
  ↪  1.429 ms  1.649 ms
 5  te2-16-cn-rt-mc.ns.uwaterloo.ca (172.16.31.117)  0.914 ms  0.862 ms
  ↪  0.883 ms
 6  te0-0-2-0-ext-rt-mc.ns.uwaterloo.ca (172.16.32.149)  1.572 ms  1.689
  ↪  ms  4.196 ms
 7  unallocated-static.rogers.com (72.142.108.181)  1.693 ms  1.890 ms
  ↪  2.223 ms
 8  24.156.146.189 (24.156.146.189)  7.947 ms  7.693 ms  7.668 ms
 9  9044-cgw01.wlfdle.rmgt.net.rogers.com (209.148.230.45)  4.895 ms
  ↪  4.932 ms  4.922 ms
10  209.148.230.26 (209.148.230.26)  24.721 ms 64.71.241.110
 ↪  (64.71.241.110)  24.691 ms  (209.148.237.5)  20.143 ms
11  eeq-exchange.tr01-asbnva01.transitrail.net (206.126.236.45)  20.128
 ↪  ms  20.190 ms  20.181 ms
12  ae-20.4079.rtsw.clev.net.internet2.edu (162.252.70.129)  25.906 ms
 ↪  25.920 ms  25.891 ms
13  ae-1.4079.rtsw.eqch.net.internet2.edu (162.252.70.131)  25.885 ms
 ↪  25.959 ms  25.935 ms
14  ae-2.4079.rtsw.chic.net.internet2.edu (162.252.70.132)  26.172 ms
 ↪  26.164 ms  25.904 ms
15  ae-3.4079.rtsw.kans.net.internet2.edu (162.252.70.141)  36.871 ms
 ↪  36.954 ms  36.918 ms
16  ae-5.4079.rtsw.salt.net.internet2.edu (162.252.70.145)  56.724 ms
 ↪  56.692 ms  56.692 ms
17  lo-0.8.rtsw.sunn.net.internet2.edu (64.57.20.225)  71.838 ms  71.632
 ↪  ms  71.621 ms
```

```
18  198.32.165.200 (198.32.165.200)  82.605 ms  82.594 ms  82.554 ms
19  * * *
20  * * *
21  * * *
22  planetlab4.cs.uoregon.edu (128.223.8.114)  82.871 ms  82.667 ms
 ↪  82.653 ms
```

B.5 *Traceroute* from UOregon → Monash

```
traceroute to pl1.eng.monash.edu.au (130.194.252.8), 30 hops max, 60 byte
 ↪  packets
 1  vl-8.uonet2-gw.uoregon.edu (128.223.8.3)  0.319 ms  0.288 ms  0.273
  ↪  ms
 2  * * *
 3  10.252.10.250 (10.252.10.250)  0.651 ms * *
 4  10.252.253.130 (10.252.253.130)  0.962 ms 10.252.253.129
  ↪  (10.252.253.129)  1.407 ms 198.32.165.125 (198.32.165.125)  12.042
  ↪  ms
 5  198.32.165.125 (198.32.165.125)  12.800 ms  12.450 ms  12.434 ms
 6  aarnet-2-is-jmb-776.sttlwa.pacificwave.net (207.231.241.4)  43.743 ms
  ↪  198.32.165.125 (198.32.165.125)  11.998 ms
  ↪  aarnet-2-is-jmb-776.sttlwa.pacificwave.net (207.231.241.4)  43.101
  ↪  ms
 7  et-2-0-0.pe1.a.hnl.aarnet.net.au (113.197.15.200)  95.247 ms  95.986
  ↪  ms aarnet-2-is-jmb-776.sttlwa.pacificwave.net (207.231.241.4)
  ↪  43.608 ms
 8  et-2-1-0.4070.rtsw.losa.net.internet2.edu (162.252.70.71)  19.325 ms
  ↪  et-2-1-0.pe1.sxt.bkvl.nsw.aarnet.net.au (113.197.15.98)  188.319 ms
  ↪  188.406 ms
 9  aarnet-2-is-jmb-776.sttlwa.pacificwave.net (207.231.241.4)  43.688 ms
  ↪  43.255 ms et-2-3-0.pe1.mcqp.nsw.aarnet.net.au (113.197.15.144)
  ↪  191.587 ms
10  et-0-3-0.pe1.eskp.nsw.aarnet.net.au (113.197.15.3)  189.831 ms
 ↪  189.882 ms et-2-3-0.pe1.mcqp.nsw.aarnet.net.au (113.197.15.144)
 ↪  188.724 ms
```

```
11  et-2-1-0.pe1.sxt.bkvl.nsw.aarnet.net.au (113.197.15.98)  188.326 ms
 ↪  et-0-3-0.pe1.eskp.nsw.aarnet.net.au (113.197.15.3)  189.870 ms
 ↪  et-2-3-0.pe1.mcqp.nsw.aarnet.net.au (113.197.15.144)  188.792 ms
12  et-0-3-0.pe1.eskp.nsw.aarnet.net.au (113.197.15.3)  190.154 ms
 ↪  189.857 ms et-0-1-0.pe1.nbpk.vic.aarnet.net.au (113.197.15.27)
 ↪  201.905 ms
13  et-5-3-0.pe1.wmlb.vic.aarnet.net.au (113.197.15.9)  201.284 ms
 ↪  et-0-3-0.pe1.eskp.nsw.aarnet.net.au (113.197.15.3)  190.698 ms
 ↪  et-0-1-0.pe1.nbpk.vic.aarnet.net.au (113.197.15.27)  201.490 ms
14  * * *
15  * xe-5-0-6-211.pe1.nbpk.vic.aarnet.net.au (138.44.64.221)  201.908 ms
 ↪  et-0-1-0.pe1.nbpk.vic.aarnet.net.au (113.197.15.27)  201.554 ms
16  xe-5-0-6-211.pe1.nbpk.vic.aarnet.net.au (138.44.64.221)  202.099 ms *
 ↪  *
17  * * *
18  pl1.eng.monash.edu.au (130.194.252.8)  201.934 ms  201.915 ms
 ↪  201.892 ms
```

B.6 *Traceroute* from UOregon → plink in UW

```
traceroute to plink.cs.uwaterloo.ca (129.97.74.12), 30 hops max, 60 byte
 ↪  packets
 1  vl-8.uonet1-gw.uoregon.edu (128.223.8.2)  0.352 ms  0.347 ms  0.341
  ↪  ms
 2  * * *
 3  10.252.9.246 (10.252.9.246)  0.374 ms * *
 4  10.252.253.129 (10.252.253.129)  1.643 ms 10.252.10.254
  ↪  (10.252.10.254)  0.985 ms  0.976 ms
 5  198.32.165.253 (198.32.165.253)  11.984 ms 10.252.9.246
  ↪  (10.252.9.246)  0.916 ms 10.252.10.246 (10.252.10.246)  0.863 ms
 6  ae-2.4079.rtsw.salt.net.internet2.edu (162.252.70.154)  26.138 ms
  ↪  26.125 ms 10.252.10.254 (10.252.10.254)  1.043 ms
 7  ae-2.4079.rtsw.salt.net.internet2.edu (162.252.70.154)  26.391 ms
  ↪  26.350 ms ae-3.4079.rtsw.chic.net.internet2.edu (162.252.70.140)
  ↪  56.925 ms
 8  ae-3.4079.rtsw.chic.net.internet2.edu (162.252.70.140)  57.242 ms
  ↪  ae-5.4079.rtsw.kans.net.internet2.edu (162.252.70.144)  45.881 ms
  ↪  ae-3.4079.rtsw.chic.net.internet2.edu (162.252.70.140)  57.173 ms
 9  * ae-5.4079.rtsw.kans.net.internet2.edu (162.252.70.144)  45.836 ms
  ↪  lo-0.8.rtsw2.eqch.net.internet2.edu (64.57.29.130)  57.093 ms
10  * ae-3.4079.rtsw.chic.net.internet2.edu (162.252.70.140)  56.902 ms
 ↪  57.081 ms
11  45-cgw01.ym.rmgt.net.rogers.com (209.148.235.90)  67.536 ms
 ↪  209.148.233.89 (209.148.233.89)  79.783 ms  79.779 ms
12  * 44-cgw01.ym.rmgt.net.rogers.com (209.148.235.106)  67.553 ms
 ↪  209.148.230.54 (209.148.230.54)  81.545 ms
13  24.156.146.198 (24.156.146.198)  82.549 ms 209.148.233.89
 ↪  (209.148.233.89)  79.646 ms  80.130 ms
14  24.156.146.198 (24.156.146.198)  82.722 ms  82.407 ms 209.148.233.89
 ↪  (209.148.233.89)  79.753 ms
15  209.148.230.54 (209.148.230.54)  79.976 ms  82.216 ms 24.156.146.218
 ↪  (24.156.146.218)  82.693 ms
16  24.156.146.198 (24.156.146.198)  82.573 ms  82.893 ms  82.509 ms
17  * * 24.156.146.218 (24.156.146.218)  82.514 ms
18  * * *
19  * * *
20  * * *
21  * * *
22  plink.cs.uwaterloo.ca (129.97.74.12)  83.232 ms  83.186 ms  83.186 ms
```

B.7 *Traceroute* from Monash → plink in UW

```
traceroute to plink.cs.uwaterloo.ca (129.97.74.12), 30 hops max, 60 byte
 ↪  packets
 1  west-gw1-v252.net.monash.edu.au (130.194.252.253)  0.914 ms  0.912 ms
  ↪  1.140 ms
 2  core-gw1-e2-2.net.monash.edu.au (130.194.29.111)  0.789 ms  0.866 ms
  ↪  0.838 ms
 3  * * *
 4  gw1.xe-5-0-6-211.pe1.nbpk.vic.aarnet.net.au (138.44.64.220)  0.764 ms
 ↪  0.754 ms  0.738 ms
 5  et-7-1-0.pe1.wmlb.vic.aarnet.net.au (113.197.15.26)  1.478 ms  1.467
 ↪  ms  1.457 ms
 6  et-1-3-0.pe1.eskp.nsw.aarnet.net.au (113.197.15.8)  12.877 ms  12.789
 ↪  ms  12.782 ms
 7  et-0-1-0.pe1.mcqp.nsw.aarnet.net.au (113.197.15.2)  14.015 ms  14.029
 ↪  ms  14.019 ms
 8  et-1-1-0.pe1.sxt.bkvl.nsw.aarnet.net.au (113.197.15.145)  14.647 ms
 ↪  14.641 ms  14.615 ms
 9  et-0-0-0.pe1.a.hnl.aarnet.net.au (113.197.15.99)  107.619 ms  107.646
 ↪  ms  107.638 ms
10  et-2-1-0.bdr1.a.sea.aarnet.net.au (113.197.15.201)  158.964 ms
↪  158.975 ms  158.959 ms
11  207.231.240.21 (207.231.240.21)  172.149 ms  173.762 ms  173.743 ms
12  vctr3rtr1.canarie.ca (205.189.32.183)  173.732 ms  173.896 ms
↪  173.884 ms
13  vncv1rtr1.canarie.ca (205.189.32.199)  175.819 ms  175.651 ms
↪  175.644 ms
14  clgr2rtr1.canarie.ca (205.189.32.175)  186.523 ms  186.542 ms
↪  186.440 ms
15  wnpg2rtr1.canarie.ca (205.189.32.177)  200.870 ms  200.859 ms
↪  201.026 ms
16  wnpg1rtr1.canarie.ca (205.189.32.61)  201.008 ms  201.003 ms  200.931
↪  ms
17  toro1rtr1.canarie.ca (205.189.32.181)  222.184 ms  222.239 ms
↪  222.326 ms
18  205.189.32.40 (205.189.32.40)  222.642 ms  222.780 ms  222.701 ms
19   (66.97.16.25)  223.334 ms  223.238 ms  223.253 ms
20   (66.97.16.110)  226.254 ms  226.258 ms  226.491 ms
21   (66.97.28.66)  226.064 ms  226.116 ms  226.141 ms
22  * * *
23  * * *
24  * * *
25  plink.cs.uwaterloo.ca (129.97.74.12)  226.508 ms  226.472 ms  226.484
↪  ms
```