MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Verner David**	Personal ID number:	**475054**
Faculty / Institute:	**Faculty of Information Technology**		
Department / Institute:	**Department of Computer Systems**		
Study program:	**Informatics**		
Specialisation:	**Computer Systems and Networks**		

II. Master’s thesis details

Master’s thesis title in English:

Evaluation of PTP and White Rabbit protocols implementations

Master’s thesis title in Czech:

Porovnání implementací protokolů PTP a White Rabbit

Guidelines:

Bibliography / sources:

Name and workplace of master’s thesis supervisor:

RNDr. Ing. Vladimír Smotlacha, Ph.D. Cesnet z.s.p.o., Zikova 4, Praha 6

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **20.11.2023** Deadline for master's thesis submission: **09.05.2024**

Assignment valid until: ______________

RNDr. Ing. Vladimír Smotlacha, Ph.D.	prof. Ing. Pavel Tvrdík, CSc.	doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the master’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

© ČVUT v Praze, Design: ČVUT v Praze, VIC

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master’s thesis

Evaluation of PTP and White Rabbit protocols implementations

Bc. David Verner

Department of Computer Systems
Supervisor: RNDr. Ing. Vladimír Smotlacha, Ph.D.

May 9, 2024

Acknowledgements

In first place I would like to thank my supervisor, RNDr. Ing. Vladimír Smotlacha, Ph.D. for his invaluable advice and guidance. I would also like to thank CESNET for providing me with necessary equipment and facilities. Last thanks goes out to my family and friends for their patience and support in this uneasy situation

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Prague on May 9, 2024 . . . .

Czech Technical University in Prague
Faculty of Information Technology
© 2024 David Verner. All rights reserved.
This thesis is school work as defined by Copyright Act of the Czech Republic. It has been submitted at Czech Technical University in Prague, Faculty of Information Technology. The thesis is protected by the Copyright Act and its usage without author's permission is prohibited (with exceptions defined by the Copyright Act).

Citation of this thesis

Verner, David. *Evaluation of PTP and White Rabbit protocols implementations.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2024.

Abstrakt

Tato diplomová práce se soustředí na síťové protokoly pro přenos času, jmenovitě Precision Time Protocol a White Rabbit. Nejprve byly protokoly studovány a také byly prozkoumány jejich možnosti. Poté byly analyzovány dostupné PTP zařízení, jak koncové tak přepínače, z hlediska jejich schopností. Toto sloužilo jako základ pro experimenty, ve kterých byla změřena a vyhodnocena kvalita těchto zařízení. Podobné experimenty byly také provedeny s White Rabbit zařízeními. Další část práce se věnovala kalibraci systému White Rabbit. Její výsledky byly také změřeny a vyhodnoceny. Jelikož tato kalibrace nepokrývá všechny případy, byla navržena nová metoda kalibrace, která umožňuje kalibrovat přenos po páru vláken. Nakonec byl vytvořen monitorovací systém pro zařízení White Rabbit, který je nyní nasazen na síti sdružení CESNET.

Klíčová slova Počítačová síť, Přenos času, NTP, PTP, White Rabbit, Síťový protokol

Abstract

This master thesis focuses on network protocols for time transfer, namely Precision Time Protocol and White Rabbit. First, the protocols were studied and their capabilities and options explored. After that a group of PTP end devices PTP switches was evaluated based on their capabilities. This served as a basis for experimental testing, when the performance of said devices was measured and evaluated. Similar experiments were also conducted with White Rabbit devices. The option to calibrate White Rabbit devices was also explored and evaluated. Since the official calibration guide does not cover every use case, a new method to calibrate connections using unidirectional pairs of fibre was proposed. Finally a monitoring system for White Rabbit was developed, that is now used in CESNET network for monitoring deployed White Rabbit Switches.

Keywords Computer Network, Time transfer, NTP, PTP, White Rabbit, Network Protocol

Contents

List of Figures

List of Tables

Introduction

Synchronization of clocks in computers and network devices is an important part of computer networks. In the beginning, the primary goal of time synchronization was to set all clocks on a network to the same time, since back then oscillators were not as precise and computer clocks frequently drifted out of alignment.

While protocols like Network Time Protocol (NTP) have been in use since the 1980s, their accuracy isn't adequate for today's requirements of distributed systems. Those systems can be distributed databases, with high rate of requests and where order of those requests is important, industrial control systems, which require precise coordination of multiple devices for successful operation or distributed measurement systems, which consist of many nodes, sometimes kilometres away from each other and sometimes require timestamps accurate to 500 picoseconds, like in the Large Hadron Collider in CERN.

It's possible to use Global Navigation Satellite Systems (GNSS) like GPS, GLONASS or Galileo to synchronize those devices, but installing them at every node is cost prohibitive. Sometimes GNSS cannot be used, for example when it's impossible to route an antenna from the device to get clear view of the sky, when the device is underground or underwater. In good conditions, GNSS can achieve precision of 1 nanosecond.

Using GNSS to synchronize every server in a datacenter isn't feasible either, so a different network or a bus would have to be constructed to deliver clock signal to all the servers, usually in the form 10 Mhz and 1 PPS, however time of day information would still have to be acquired using a different method, like NTP for example. Since this method of PPS distribution doesn't account for propagation delay, the delay introduced by the cables that deliver this signal would have to be measured and compensated for in end devices.

Precise clock synchronization in network is the goal behind Precision Time Protocol (PTP), which can operate on standard Ethernet network, however for best results, it's still necessary to use special network cards and switches with PTP support. This hardware allows PTP to achieve sub microsecond accuracy while it’s still possible to carry regular network traffic on the same network. Similar to NTP, it provides automatic delay calculation. The accuracy of PTP allows it to be used instead of GNSS.

PTP is highly configurable protocol and not all manufacturers of PTP enabled devices implement all features it provides. Furthermore it can be implemented both in software of the computer it’s running on or in hardware directly in the PTP interface. Its performance can vary greatly considering all of this, it guarantees only sub-microsecond precision, however certain implementations can reach precision below 10 nanoseconds.

White Rabbit improves on PTP, mainly by matching the frequency of all clocks on its network, this allows it to achieve sub nanosecond precision. White Rabbit surpasses GNSS when it comes to precision and accuracy and is nowadays used in many research projects that require precise time synchronization.

This thesis evaluates these protocols, namely PTP and White Rabbit and provides comparison of their different implementation by various manfacturers.

Chapter 1

Time Transfer Protocols

Time Transfer Protocols have the goal of synchronizing clocks on different computers. This chapter introduces selected protocols, that are currently used in the world for this purpose.

1.1 Synchronization problem

Clock synchronization is a problem, whose objective is to coordinate independent clocks so they all have the same time scale. This means that the difference in the phase offsets of their clock pulses over time are as small as possible.

There are two ways to achieve this, based on the property of the clock that's being adjusted.

First one relies on periodic phase offset adjustment. Clocks exchange their current phase offset and are adjusted accordingly. Since their frequencies differ, the clocks will drift apart again. They then require constant synchronization to keep the phase offset difference low.

Second option is to also adjust the frequency of the clocks, so they match. It can be done either by sending multiple timestamps and disciplining the target oscillator based on those samples or sending target frequency over the physical connection between the two clocks (for example 10MHz signal over cable or carrier signal in radio transmission). It is impossible to align the phases of the signal this way unless the delay between the clocks is known, so phase correction is still necessary, However, assuming path delay to remain constant over time, the target frequency matches the one at the source, which eliminates the clock drift.

It's also possible to modify the frequency of one clock and make it drift into alignment with the other clock, and match their frequencies after that, that eliminates the need for phase adjustment

1.2 Time and frequency transfer

There are many ways to exchange time information between multiple devices. These can be generally split into two categories: one-way and two-way.

One-way techniques rely on a master sending out a signal containing time and frequency information and slaves using that information to set their internal clock. There is no communication from the slaves to the master. Using this technique, propagation delay of said signal is usually unknown and therefore cannot be compensated for. Nowadays most popular one-way methods of synchronization are radio signals like DCF77, used in most radio clocks sold in Europe, and GNSS (Global Navigation Satellite Systems). Since GNSS (for example GPS) uses many masters (satellites), each carrying its own atomic clock, the slave can calculate and eliminate the propagation delay using information from 4 or more satellites.

Two-way methods do two one-way transmissions, one in each direction. Using those two transmissions, it is possible to calculate the round-trip delay between the master and slave and therefore measure the offset between the two clocks. Calculation of the offset and one-way propagation delay assumes that the delays in each direction are equal, which might not always be the case, especially in protocols running over computer packet networks. Therefore different protocols were designed to eliminate any variable delay sources and only leave the propagation delay, which remains constant over time. Such protocols are the subject of this thesis.

Frequency transfer is more straightforward, it can either be done with special connection with a clock pulse (Typically 10MHz) or it can be transferred using the carrier signal of the connection (SyncE). Certain methods support only synchronization (NTP, PTP), other only syntonization (Direct connection of clocks, SyncE) and some can do both (DCF77, White Rabbit).

1.3 Network Time Protocol

Network Time Protocol (NTP) [1] is currently the most used protocol for clock synchronization in computer networks. It uses two-way packet exchange to measure the clock offset.

NTP works on a client-server model, where one client requests data from many servers and synchronizes its clock based on the data received. One NTP server can use other servers as a reference for its clock. The servers are thus organized in a hierarchical structure. Each server is assigned a number based on how many steps removed it is from a primary clock. This number is called stratum. Servers directly connected to primary clock are stratum 1, the ones synchronized to such servers are stratum 2, etc. Although the definition goes up to stratum 16, until clock is marked as unsynchronized, in reality, only servers with stratum 1 or sometimes 2 serve as global time sources and servers with stratum up to 3 or 4 as local NTP servers for private networks, all higher strata clocks are not considered a good sources, since the error they accumulated over many hops from the master clock is too high.

Each client has a list of servers, that can provide time synchronization. After it polling all servers on that list, it calculates which server provides the best timestamps by using intersection algorithm to compare received data from all of them. It typically achieves precision in the realm of tens of ms over the internet or a large network. On local networks, it can even achieve sub millisecond precision.

1.3.1 Synchronization

To account for propagation delay from master to slaves, NTP implements a two way synchronization method by sending one packet each way. The communication is started by a slave by sending a NTP request to the server. [2]

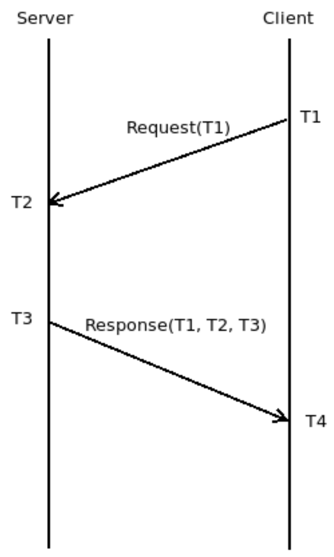


Figure 1.1: Diagram of NTP communication

- Slave sends a NTP request at T1 time, the local time of the request is embedded into the packet.
- The server receives this packet at time T2 and saves this timestamp.
- The server sends a response to the client at time T3, timestamps of all previous events (T1, T2 and T3) are included in this packet.
- Client receives the packet at time T4 and saves all 4 timestamps.

Now the client has all information it needs to calculate the offset of its clock from the clock provided by the server

$$offset = \frac{(T_2 - T_1) + (T_3 - T_4)}{2}\,. \tag{1.1}$$

Round trip delay can also be calculated in a similar manner.

$$delay = (T_4 - T_1) - (T_3 - T_2)\,.$$

1.4 Disadvantages

NTP is good enough for synchronization of regular personal computers, but it has many shortcomings. Mainly the assumption on path delay between the application on server and client side being the same in both directions. This delay consists of 3 parts.

- Propagation Delay - How long does it take for either an electrical signal or a light pulse to travel through the wire/fibre.
- Queuing Delay - Delay caused by traffic congestion and by buffering in network interfaces, both in end devices and switches/routers.
- Processing Delay - Delay caused by the application itself, the time it takes from creation of the request to its encapsulation and actual relay of the packet to the network interface and vice versa, the time it takes from receiving the packet to reading the timestamp in the receiving application. Routing and switching on the path between client and server also contributes to this delay.

It can be assumed that propagation delay is symetrically distributed between both directions, if the request and response packets take the same path through the network. But both queueing and processing delay are unpredictable and NTP then cannot guarantee accurate calculation of the offset. Upper bound of inaccuracy is therefore $delay/2$, which accounts for the case, when round-trip delay is caused by one direction and the other one is instant. When the requirements for accuracy are higher, other methods need to be considered.

1.5 Precision Time Protocol

Precision Time Protocol [3] (PTP or IEEE 1588) was designed to replace NTP where higher level of accuracy was needed. It still uses normal packet=switched network to communicate. However, unlike NTP, PTP uses a master-slave model, with one, so called, Grandmaster clock (GM) on the network serving as a source of truth for all clocks that are being synchronized. This clock regularly sends messages with current time to all of its peers and those slave clocks only initiate communication to measure the round-trip time and calculate the clock offset between the master and themselves. [4]

1.5.1 Synchronization

PTP tries to improve on the synchronization model used by NTP by reducing uncertainty in packet delay. Since NTP does not use special hardware and only operates using software, it cannot predict how much will its packets get delayed either in the source or end device itself or on the path to their target. In order to eliminate any delays caused by packet residency, it introduces a way to timestamp packets either at the exact moment of the egress from the device or ingress to it. This requires special network adapters, which have a timestamping module located between the media access control (MAC) module and the physical layer (PHY), that is capable of inspecting packets that are passing through and noting the time of their passage. This information is either sent to the target device in another packet called Follow up, or if the hardware allows it, it can be put into the original packet as it is being processed by the interface.

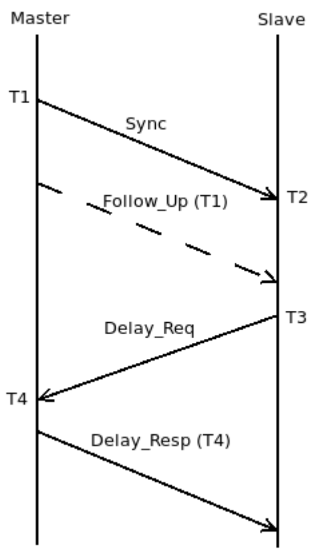


Figure 1.2: Diagram of PTP communication

The synchronization works as described by illustrated 1.2, please note that this is just one of two delay measurement principles that PTP provides, as will be discussed later.

Master initiates the process by sending a Sync message and noting down the time of its egress from the device T_1. When a slave receives this message, it notes down time of its arrival T_2. The master then sends a Follow up message containing T_1. After receiving the Follow up packet, the slave knows the offset of the two clocks with an error caused by path delay. Slave then needs to send a request to the master, in order to calculate the time difference in the other direction. This request is also timestamped, but the time is just stored in the slave (for calculation, master does not need such infromation). Upon receiving the packet, master again timestamps this event and sends the timestamp back to the slave in Delay_resp message. This way slave obtains the same 4 timestamps as in NTP, so the calculation of clock is the same as described by equation 1.1.

The main difference from NTP lies in the hardware timestamping, allowing these calculations to be much more precise, making the synchronization more stable, since the propagation delay should be almost constant over time and equal in both directions.

This shows the synchronization only for devices with a direct connection between them. The biggest difference between PTP and NTP however lies in how they compensate for the delay that happens in other devices on the path between the master and the slave. NTP does not differentiate how many network devices, such as switches, server and client have between them, thus introducing delay asymmetry into the calculation due to unpredictable switching or routing in the device and packet queuing in NICs. PTP on the other hand provides support for switches with hardware timestamping that can precisely measure residency time and update the timestamps in PTP messages, effectively eliminating all mentioned uncertainties. [5]

1.5.2 Clock types

Precision Time Protocol defines 3 types of clocks based on the function they serve in a PTP network. [6]

1.5.2.1 Ordinary Clock

Ordinary clocks (OC) represent standard PTP end devices with only one PTP enabled port. They can be configured either as a slave or a master. When configured as a master, they can rely on their inner clock for distribution, but more often than not, they also support external references, such as GPS receiver or PPS and 10 MHz input from a more precise source, like an atomic clock for example. A slave clock is getting timing information from its selected master and adjusts its internal clock accordingly.

1.5.2.2 Transparent Clock

Transparent clocks (TC) serve as switches in a PTP network. They have multiple ports, to which different PTP devices are connected. They neither synchronize their clock nor provide synchronization information, they only forward PTP packets to their destination like a regular switch would. In addition to that, transparent clock measures the time each PTP packet spends traversing the switch and updates timestamp of the message with data corrected for this delay, thus behaving transparently for PTP. A standard network switch can also be used in place of a TC, but since it cannot do timestamp correction, the precision of time transfer will be severely reduced.

When operating a PTP network that incorporates TCs, you have two options for delay calculation.

First one is that the Delay_Req message is passed from the slave all the way to the master and each TC it passes through timestamps it both on ingress and egress and when master sends Delay_Resp back, TCs correct the timestamp to compensate for residency time. This is know as an End to End (E2E) delay calculation.

The other option is Peer to Peer (P2P). When using P2P, each PTP device on the network calculates its offset from all neighboring devices. Every path is then measured in both directions, in case a Sync packet takes a different path through the network, for example when a grandmaster changes. Since delay of every path in the network is known, it can just be added up on the way of the Sync packet to get the total delay for the slave. Notice how the client does not request anything from the master directly. This method helps in network resiliency, since it avoids an issue when Sync and Delay_req packets would take different paths which would result in miscalculation of total network delay, it does not matter which path the Sync packet chooses, the delay will be always calculated correctly. Another benefit of using P2P lies in rapid adaptation to network topology changes, since no further calculations are required when a device fails and the message is forced to take a different path. Drawback of P2P is that cannot be used on a network that contains non-PTP switches, since those switches will not respond to Peer Delay Request messages.

1.5.2.3 Boundary Clock

Boundary clocks (BC) are placed between PTP network segments. They have multiple PTP ports, of which, one is acting as a slave, synchronizing the Boundary clock's oscillator to an upstream master, and the rest are master ports. The BC will share its time with devices connected to these ports. Since the BC is communicating with downstream devices directly and only synchronizes itself with its master, usage of BCs in large networks can alleviate network traffic towards the GM and thus prevents overloading it, which could lead to less precise synchronization.

1.5.3 One-step and two-step

PTP network can operate in two modes based on method of communication of precise timestamps of packet egress. Either one-step, where the correct timestamp is already in the Sync packet, or two-step, where the egress timestamp

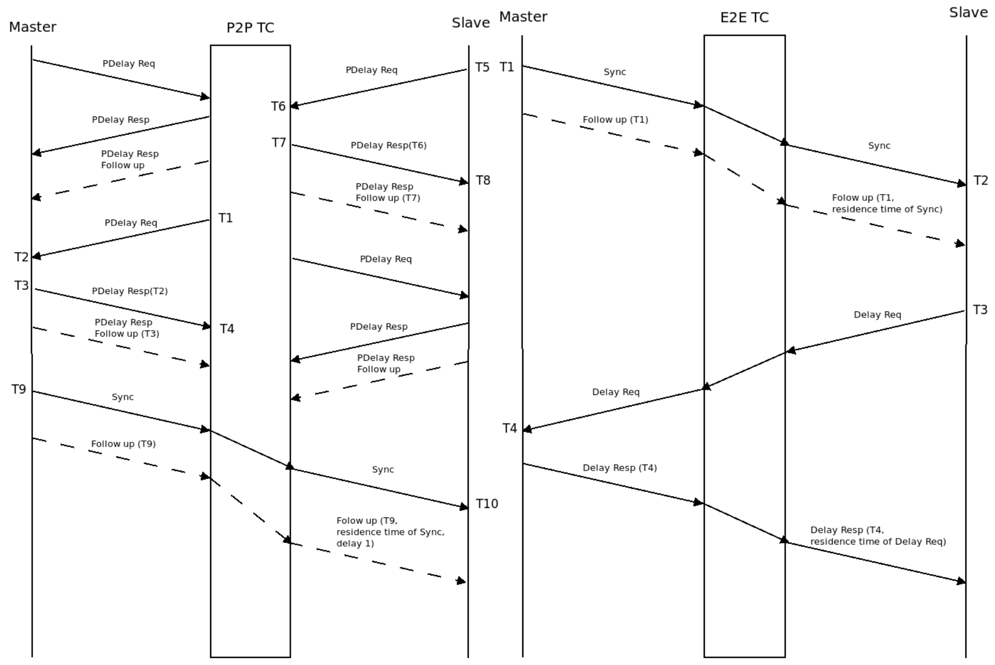


Figure 1.3: Comparison of E2E and P2P communication

is sent in a separate packet, called a Follow up. Each of these methods offers some pros and cons. One-step is usually simpler to implement, since for Full HW acceleration, follow up packets would have to be generated directly in the PHY, which operates in a "one byte in, one byte out" mode. One method to circumvent this obstruction is to generate a dummy Follow up packet using CPU and update the timestamp of this packet in the PHY. The biggest disadvantage of using one-step is its physical limitation when it comes to higher network speeds. At gigabit speeds, it takes 51 nanoseconds to send 64 bytes (the size of PTP message), but at 10 Gbps speed it is only 5 ns and there is not enough time to get the timestamp of the packet, calculate the delay and write the data back into the packet while it is being processed by the network interface. In this thesis, we will be only using a gigabit network, so we can disregard this fact, but it is nonetheless an important factor to consider when building a PTP network. [7]

1.5.4 Best Master Clock Algorithm

To ensure best possible synchronization at all times between nodes in a PTP network, there can only be one master clock, unlike in NTP, where the selection and averaging of timescales is done client-side. This clock is selected using so called Best Master Clock Algorithm (BMCA). BMCA is a distributed algorithm, which makes it very robust in case of any failure in the network.

Each clock that attempts to be a master sends Announce messages with following information about itself.

- **GM priotity 1** - User set value, serves as an override for automatic clock evaluation.
- **GM clock class** - Status of the clock (Synchronized to external reference, lost its reference but still in holdover, unsynchronized etc).
- **GM clock accuracy** - Self reported accuracy of the master clock.
- **GM clock variance** - Value calculated based on Allan variance 3.3.5 of the clock.
- **GM priority 2** - User set value, tie breaker in case of identical clock quality.
- **Clock identity** - Unique clock ID, tiebreaker in case of identical quality and priority.
- **Port number** - Tie breaker for messages coming from different ports of the same BC.
- **Steps removed** - Tie breaker for messages coming from the same port of a BC but taking different paths through the network.

Each clock that receives these messages evaluates them in order they are listed here and chooses its state and its upstream master clock. If a slave receives multiple of these messages, it chooses the master with the best quality. If a device attempts to be a master and only receives announce messages from a worse clock, it continues to send its announce messages to all other clocks. If it does not receive a message from a better candidate during so called Qualification timeout, it becomes the master for its network and starts sending Sync packets to all slaves. Once it receives announce packet from a better master, it stops the Sync and Announce transmission and changes its state to either slave or passive, depending on its configuration as shown in Figure 1.4. Once the delay between announce messages reaches configured announce timeout, it may start to announce itself once again and attempts to become the new master.[8]

Since every clock evaluates the information contained in announce messages, the algorithm always finishes and always elects the same master in each PTP network in finite time. This makes it very robust, because it can handle failure of multiple clocks and adapts quickly to any changes in the network topology.

Please note that clock in this sense is meant as a PTP instance on a port. So in case of a BC, each port configured as a master is a master clock candidate for the network it is connected to. BC synchronizes its internal clock from a

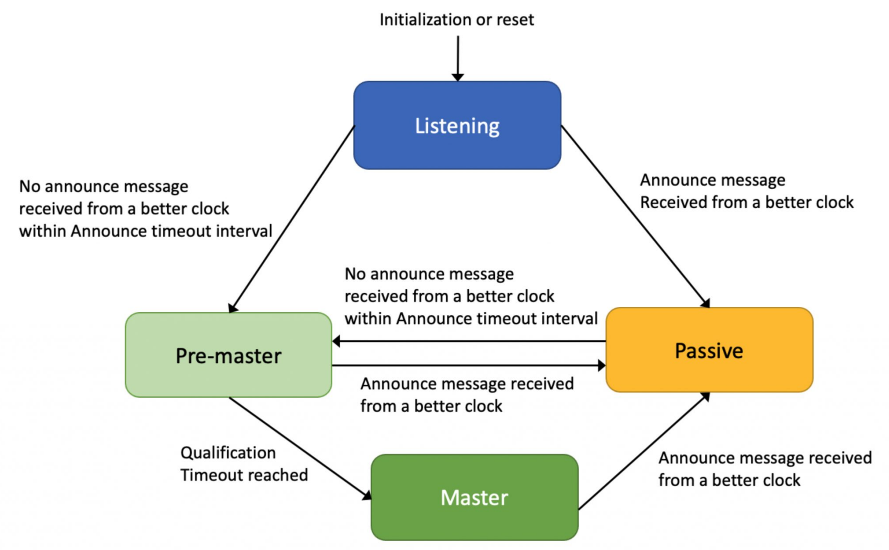

Initialization or reset
Listening
No announce message received from a better clock within Announce timeout interval
Announce message Received from a better clock
No announce message received from a better clock within Announce timeout interval
Pre-master
Passive
Announce message received from a better clock
Qualification Timeout reached
Master
Announce message received from a better clock

Figure 1.4: Diagram of BMCA state machine for master-only ports, taken from [9]

master connected to its slave port and advertises this timescale on all of its master ports. Each of those ports is then in master state, if no better master clocks are found on their respective network, or in passive state, if the BC received an Announce message from a better master candidate then itself on that specific port.

1.5.5 Message types

The PTP uses various messages sent over the network to synchronize clocks and manage itself. These messages can be split into two categories.

Event messages, as the name suggests represent an event in time, they therefore require precise timestamping to either note the time for themselves or to relay timing information about the event to other nodes. These messages are:

- **Sync** - Periodic broadcast of a timing information from elected master. It can contain either precise timestamp of the Sync event, in case of one-step operation, or just an approximate one for two-step operation. This is indicated by a flag in the PTP packet header.

- **Delay_req** - Request from a slave to calculate its precise offset from its master. The contents are not precise, both egress from slave and the ingress to master is precisely timestamped.

- **Pdelay_req** - Similar to the normal Delay_req, but does not propagate through Transparent clocks. Again the time of egress and ingress is timestamped.
- **Pdelay_resp** - Precise timestamp of the ingress mentioned above. Since the two devices participating in a P2P delay calculation do not exchange a Sync message, this message also needs to be timestamped and the timestamp sent in a Follow_up packet (in case of two-step) in order to calculate the delay between the two nodes.

General messages do not require precise timestamping, however, they can carry the timestamp of an event message.

- **Announce** - Info about a master candidate that advertises itself to the PTP network for BMCA.
- **Follow_up** - Packet containing precise timestamp of a Sync message. Only used in two-step mode
- **Delay_resp** - Message containing time of ingress of a Delay_req message to the master device.
- **Pdelay_resp_follow_up** - Precise timestamp of the sending of a Pdelay_resp message.
- **Management**
- **Signaling**

1.5.6 PTP packet

PTP packet consists of two parts, PTP header and the timestamp itself. PTP header contains information about the options selected for current configuration of the protocol, as well as data specific for that particular message.

Listing 1.1: structure of PTP packet parsed by Wireshark

```
Precision Time Protocol (IEEE1588)
    0000 .... = majorSdoId: Unknown (0x0)
    .... 1000 = messageType: Follow_Up Message (0x8)
    0000 .... = minorVersionPTP: 0
    .... 0010 = versionPTP: 2
    messageLength: 44
    domainNumber: 0
    minorSdoId: 0
    flags: 0x0000
    correctionField: 0.690994 nanoseconds
    messageTypeSpecific: 0
```

```
ClockIdentity: 0x64fb81fffe2f7785
SourcePortID: 2
sequenceId: 64897
controlField: Follow_Up Message (2)
logMessagePeriod: 0
preciseOriginTimestamp (seconds): 1707258129
preciseOriginTimestamp (nanoseconds): 407789450
```

Following is a brief description of selected values contained in the message.

At the start of the header, there is generic information about the message, such as message type, message length and protocol version, Be mindful that PTPv2 is not backwards compatible with the first version.

Flag is a set of bits that indicate properties of the protocol used. For example whether the communication is unicast or multicast, or one-step or two-step.

Clock identity is a universally unique identifier, that can be traced back to the clock. Clock identity is usually equal to MAC address of the network interface the clock is attached to. In this case, it is the ID of the clock that sent the packet which was timestamped.

PTP Domains allow separation of groups of PTP devices on the same physical network. It acts similar to VLANs, only clocks with the same Domain Number can interact with each other. When clock in one PTP domain receives a timing message from another domain, the packet is discarded. Allowed domain numbers range from 0 to 255, with 0 being the default value.

Correction Field accounts for any measured delay that happened on the path. It is the field that TCs modify when packet passes through them. In P2P operation, the measured peer delay is accumulated in this field. In two-step operation, Sync packet has 0 correction and everything is done through the Follow_up packet.

PreciseOriginTimestamp is, as the name suggests, a precise timestamp of a Sync or Pdelay_resp packet. This information is either contained in the packet itself in one-step mode or in a respective Follow up packet when using two-step mode.

1.5.7 PTP profiles

Since PTP found its way into many different applications across many different fields, the need to modify the protocol to suit the needs of those applications arose. Therefore in PTPv2, defined in standard IEEE 1588-2008, support for different PTP profiles was added. PTP profile is defined in IEEE 1588-2008 as “The set of allowed Precision Time Protocol (PTP) features applicable to a device”. Profile can either configure variables of PTP, such as, but not limited to:

- Intervals and timeouts of messages (Sync, Announce, Delay Request...),

- Priorities of devices in BMCA,
- Communication protocol (IPv4/6 or IEEE 802.3 Ethernet),
- Delay calculation (P2P or E2E),
- One-step or Two-step operation.

These parameters can be configured in standard PTP, but enforcing their values in this way ensures compatibility across all devices using that profile without any need for additional configuration. However, PTP profiles also provide extension to the normal PTP operation, which allows for usage in areas where the IEEE 1588 standard is not sufficient. Extensions modify behaviour of the core mechanics of PTP. Let us take one of the most popular profiles as an example, the ITU-T G.8265.1 Telecom Profile, used in base stations of telecommunication networks.

- Alternate BMCA - New parameters to be checked can be introduced or the algorithm can be totally changed to adhere to application requirements, for example allowing more grandmasters.
- New clock types with modified behaviour - Those include Telecom Slaves that can have multiple grandmasters selected and the slave than chooses a grandmaster to get its timescale from.
- Forced unicast communication - Standard PTP supports both multicast and unicast communication, the Telecom Profile forces unicast communication where slaves need to request synchronization from master.

Telecommunication networks are very rigid, meaning almost no network topology changes are happening. Therefore the alternate BMCA acts very static, with masters being decided by the first priority field. Because of this, the communication between slaves and masters can happen using unicast, with slaves requesting synchronization first, using Signalling messages, after receiving Announce message and the Master initiating the synchronization process using a Sync message as usual. Another feature of Telecom Profile is the lack of support of Transparent Clocks. So all switching on the way from master to the slave is done through Boundary Clocks, which should yield better results as transparent clock accumulate time error throughout the network, since they do not contain as precise oscillators. [10] This also ensures higher degree of redundancy, when in a case of a grandmaster failure, the boundary clock can temporarily step in as the time source ensuring that the clocks that are below the BC in the network hierarchy all have the same time scale. Since TCs are not not used in this profile, P2P delay calculation method would not yield any benefits over E2E and therefore is not allowed in order to make network communication simpler.

1.6 White Rabbit

White Rabbit (WR) is a PTP profile originally developed by CERN for synchronization of sensors in their particle accelerator. Since then, it outgrew its original purpose and is now an international project to achieve precise time synchronization.

Since it is a profile, it is based on PTP, but comes up with a few improvements in order to increase synchronization precision all the way below 1 nanosecond. The two biggest ones are syntonization (frequency transfer) and even more precise delay compensation, including fiber asymmetry in the case of longer connections, called Link Delay Model, described in 1.6.2. Unlike PTP, WR ensures that all nodes on the network are running their clocks with the same frequency.

Similar to the Telecom Profile WR only defines devices that are ordinary or boundary clocks. Therefore it is only using E2E delay calculation between each switch. White Rabbit profile (WRPTP) also only allows two-step communication.

1.6.1 Syntonization

As mentioned above, both NTP and PTP are protocols for time transfer, not frequency and when a client machine wants to match the frequency of its clock with the master, it needs to do so based on timestamps received from a master clock. This adjustment cannot be completely deterministic, so WR improves on this solution by introducing frequency transfer, also called syntonization. In order to adhere to base Ethernet specifications, frequency is transfered in the physical layer, using Synchronous Ethernet (SyncE). Normally, Ethernet interfaces produce their own frequency for any given connection to be used as a carrier signal. SyncE sources that frequency from a different clock, the oscillator in WR switch in this case. On the other end of the connection, the frequency is recovered from the bitstream to a clock signal and the clock of that peer is adjusted using a phase-locked loop (PLL). The WR protocol uses frequency of 125Mhz. Since all clocks on the network are running at the exact same frequency, constant time synchronization events are no longer necessary to avoid clock drift. WR then uses PTP only for initial synchronization and after that keeps the protocol running at a reduced message rate for redundancy sake, but those packets are not necessary. The syntonization occurs before the initialization of PTP transmission, WR protocol introduces a handshake called WR Link Setup that makes sure the clocks are running with the same frequency and phase offset before the first Sync message, the handshake is shown in figure 1.5.

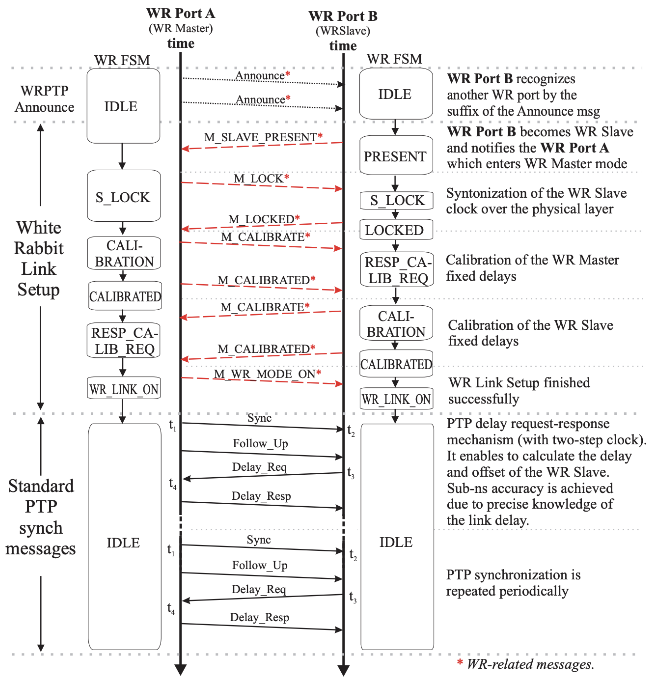

WR Port A
(WR Master)
time
WR Port B
(WRSlave)
time
WR FSM
WR FSM
WRPTP Announce
IDLE
Announce*
Announce*
IDLE
WR Port B recognizes another WR port by the suffix of the Announce msg
M_SLAVE_PRESENT*
PRESENT
WR Port B becomes WR Slave and notifies the WR Port A which enters WR Master mode
White Rabbit Link Setup
S_LOCK
M_LOCK*
S_LOCK
Syntonization of the WR Slave clock over the physical layer
M_LOCKED*
LOCKED
CALI-BRATION
M_CALIBRATE*
RESP_CA-LIB_REQ
Calibration of the WR Master fixed delays
CALIBRATED
M_CALIBRATED*
M_CALIBRATE*
CALI-BRATION
Calibration of the WR Slave fixed delays
RESP_CA-LIB_REQ
M_CALIBRATED*
CALIBRATED
WR_LINK_ON
M_WR_MODE_ON*
WR_LINK_ON
WR Link Setup finished successfully
Standard PTP synch messages
IDLE
Sync
Follow_Up
Delay_Req
Delay_Resp
IDLE
PTP delay request-response mechanism (with two-step clock). It enables to calculate the delay and offset of the WR Slave. Sub-ns accuracy is achieved due to precise knowledge of the link delay.
PTP synchronization is repeated periodically
* WR-related messages.

Figure 1.5: White Rabbit handshake, taken from [11]

1.6.2 Link Delay Model

In order to achieve sub-nanosecond accuracy, White Rabbit protocol further improves the mitigation of delay asymmetry done by PTP. It separates those asymmetries into two categories.

- **Fixed delays** - Delays in the device itself, they do not change with setup changes.
- **Variable delays** - Delays in the fiber between two WR nodes.

Example of fixed delays are delays happening in FPGA, path delay on the PCB or delay caused by processing of data in the SFP. These delays are easy to compensate for, since they are constant, so they only need to be measured once and the device calibrated. This is also done in WR Link Setup, same as syntonization.

To keep the variance in path delay as predictable as possible, WR network utilizes fiber connections exclusively. Copper connections can be used for management of the switch, but copper SFPs do not support SyncE and encode everything using their own clock, which makes them unsuitable for WR. When using a pair of single mode fibers, one for each direction, there is no guarantee of them being the same length. According to [12], light travels though a fiber at approximately 67% of c or $200000 km/s$. Every 20cm of difference in length then causes 1 ns difference in propagation delay, which is unacceptable for WR. Because of this, it is recommended to use a single fiber for both directions, which is achieved using 1000BASE-BX10 bi-directional transceivers. BX uses different wavelength for each direction of the signal, namely 1310 and 1490 nm. Biggest cause of variable delay is then chromatic dispersion. Each wavelength of light travels at a slightly different speed and those small differences can cause significant error when taking into account the precision and distance between nodes WR operates with. As mentioned in [13], such difference of wavelengths can result in a time difference of approximately 1500 ps/km, with great variance depending on the fiber used. According to [14], G652 fiber which is recommended for use in WR networks achieves chromatic dispersion of 1486 ps/km. The WR devices therefore need to be calibrated to account for the difference in propagation delay in the fiber.

Angle of refraction, and therefore chromatic dispersion, are also dependent on temperature and other physical properties of the fiber itself, so it is important to calibrate the fiber in the conditions it is going to be used in. Another source of variable delay is SFP temperature. Differences in temperature can cause slight shift in the wavelength of light produced by the laser, which contributes to the chromatic dispersion, which further increases uncertainty. Luckily the temperature of SFPs usually does not change once the device reaches operating temperature, making calibration simpler. [15]

Chapter 2

Devices and Their Capabilities

All the devices here were kindly provided by CESNET for the purpose of this thesis.

2.1 Time and frequency counter

To precisely measure the phase offset of two clocks, a time counter was used, namely a Standford Research Systems SR620. [16]. It offers 25 picosecond one-shot time resolution, which is good enough to verify that the tested protocols behave as advertised. Unfortunately WR in ideal conditions achieves similar accuracy of time transfer those measurements will be skewed by the noise of the time counter.

2.2 PTP Grand Master

When running experiments with end devices and switches, it was important to keep the timesource consistent, to eliminate as much variance as possible. It was necessary to use a high quality PTP grand master with good oscillator that can provide sufficient stablity. A Meinberg microSync RX412 [17] was used as the grand master for all the tests, unless mentioned otherwise.

2.3 Reference clock

An atomic clock was used as the main time source for the PTP master. Namely it was the Symmetricom 5071A [18], a Caesium based clock, which has been in production for decades and is considered an industry standard nowadays. It achieves $5 * 10^{-13}$ accuracy, which makes it ideal time source for our experiments, since all tested technologies are less accurate.

	ISO/OSI Layer	Clock type	One-step/Two-step
Intel I210	2, 3	OC, BC, TC	2
Solarflare SF432	3	OC, BC	1, 2
Meinberg PTP270PEX	2, 3	OC	1, 2

Table 2.1: Comparison of PTP NICs

2.4 PTP devices

Following devices were the subjects of experiments, their performance was measured.

2.4.1 End devices

3 different network interface cards (NICs) from 3 different manufactures were provided for examination and measurement. The cards were installed in 3 separate computers with otherwise identical configuration.

- **CPU** - Intel Celeron 847 1.1 GHz, 2 cores
- **Motherboard** - ASUS J1800I-C
- **Memory** - 2GB
- **Boot drive** - 500GB Seagate HDD
- **Operating System** - Debian 10

All cards are using PCIe and were therefore connected to that slot on the motherboard.

As we can see in table 2.1, the implementation of PTP in various switches varies a lot and just purchasing PTP devices without any prior research might not yield the result one is looking for.

2.4.1.1 Intel I210

Intel I210-T1 network adapter is one of the most accessible ways to get to PTP enabled hardware. It's a gigabit PCIe card with one RJ45 ethernet port that is capable of hardware timestamping. It's fully compatible with LinuxPTP project, an open source PTP implementation made for, as the name suggests, Linux operating system.

LinuxPTP implements the PTP protocol according to IEEE 1588-2008 standard, it consists of three separate daemons, that are required for complete functionality of PTP.

- **ptp4l** - Base application, that is responsible for all PTP communication as well as management of the hardware clock. It implements all 3 PTP clock modes (BC, TC, OC). Since the Intel card used contains only one port, to use it for BC and TC, multiple cards need to be installed and connected either directly, or through the system clock.
- **phc2sys** - A utility that synchronizes PTP hardware clock with the system clock of the host machine and vice versa. It can be used to synchronize local clock to the clock of PTP in case of use in a slave ordinary clock. Or the system clock can be used as the time scale to be shared by PTP when the device is configured as a grandmaster.
- **ts2phc** - ”Timestamp to Physical Clock” is used to recover clock pulse from the PTP device when supported. This was the case of the Intel cards, they had software programmable pins, that could be configured as PPS and 10 MHz output, this clock signal was then used for measurement. Alternatively, those pins can also be used as a input from a precise time source when used as a grandmaster clock. In the case of Intel I210-T1 those pins are present only as headers on the PCB of the card, which means they aren’t easily accessible and they had to be connected using a probe instead of regular cables.

2.4.1.2 Meinberg PTP270PEX

While the card from Intel is a generic network interface with PTP capabilities, PTP270PEX from Meinberg is a dedicated PTP ordinary clock designed for time transfer, it is not capable of facilitating regular network traffic. Since it’s an ordinary clock, it also contains only one PTP port, however it cannot serve as a BC or TC, when using multiple units, unlike the Intel card. Apart from the ethernet port, the card also contains a RS232 connection, which can be used as an output for the clock signal, selected pins of the RS232 connector can be assigned to either PPS or 10Mhz signal. Unlike Intel I210, the whole PTP stack is being run directly on the device, without the need for any software on the host computer. In fact, when the intention is to use the clock output, the PCIe connection serves only as a power source, however it’s used for checking device’s state and configurating it, using `mbgstatus` and `mbgctrl` utilities respectively. Meinberg also includes `mbgsvcd` daemon, that is used for transmission of timestamps between the card and the host for synchronizing its clock.

2.4.1.3 SolarFlare SF432

SolarFlare SF432 is a 10 Gb/s card with two SFP+ interfaces. Interestingly, it doesn’t support PTP on Layer 2, therefore it was only used for experiments using IPv4 for communication. It doesn’t support transparent clock mode

	ISO/OSI Layer	Clock type	One-step/Two-step
Hirschmann MM3/MM23	2, 3	BC, TC	1,2
Cisco Catalyst 9300	2, 3	BC, TC	2
Planet IP40	2, 3	BC, TC	2

Table 2.2: Comparison of PTP Switches

either, even though it has two network ports. If it needs to be used as a switch, it has to be configured as a boundary clock. It also has a direct connection to the clock circuit, which can serve as either an input or an output, it uses a SSMB coaxial connector, which is a lot easier to work with than the ones in the other two NICs. While SF432 also relies on software implementation of PTP, it isn't supported by linuxPTP, instead `sfptpd` had to be used, a linux PTP daemon created directly by Solarflare.

2.4.2 Switches

As described in 1.5.2, PTP requires support in network switches for the best accuracy. All PTP switches used here and serve the role of a transparent clock. Over all the implementation of PTP in switches is much more consistent than in NICs, as can be seen in table 2.2.

2.4.2.1 Linksys LGS116P

Linksys is a regular consumer switch without any PTP support, it therefore only supports E2E delay calculation. It was used for a baseline measurements to compare it to PTP enabled switches.

2.4.2.2 Hirshmann MM3 and MM23

Two industrial switches intended to be mounted on a DIN rail. They only support Fast Ethernet, it was therefore possible to saturate the link between them, so they were ideal for testing, how does PTP cope with overloaded network.

2.4.2.3 Planet IP40

Planet IP40 is industrial switch of current generation, apart from 4 RJ45 ports, it also features 2 SFP ports for longer distance transfer.

2.4.2.4 Cisco Catalyst 9300

Catalyst 9300 is a enterprise grade switch with PTP support. It supports multicast two-step operation on both layer 2 and layer 3. It's capable of as an TC as well as BC. Its PTP capabilities are locked behind a license, without the license it acts as a regular switch for PTP traffic and therefore would not yield any benefits.

2.5 White Rabbit Devices

Only one device model that supports White Rabbit was available, White Rabbit Switch. Three pieces of this switch were provided, so experiments with WR synchronization were still possible.

2.5.1 White Rabbit Switch

White Rabbit Switch (WRS) is the backbone of a WR network. It is entirely open source, including the hardware, software and firmware. At its core it's a gigabit network switch with 18 SFP ports, which can transport both timing data as well as regular network traffic. It serves the function of either a boundary clock or a grand master. Apart from the 18 WR capable and management ports, it contains 10 Mhz and PPS input and output. Input ports are used to send signals from a precise upstream clock to the switch when acting as a grand master. Output port is used to recover clock signal from a switch configured as a boundary clock. [19]

Since the base version is completely open source, its license allows for modifications. One of the most popular ones is the WRS Low Jitter made by Seven Solutions. It made several improvements over the base version. The most notable being a more stable clock using a oven controlled crystal oscillator (OCXO). Its crystal sits in a small oven with precise temperature control, so it's much more resilient to frequency changes due to the change in ambient temperature.

2.5.1.1 Shortcomings

Even though the White Rabbit Switch is a complete solution for precise time and frequency transfer, that is widely used, it is clear that it is still a research project and it lacks some capabilities that would be necessary for commercial deployment.

- Management GUI - WRS provides a simple WebGUI for switch management, but its usage is not recommended by the creators, all the management is therefore done using a CLI over ssh. Because of this, it is not easy to monitor the state of WR system, especially when running multiple switches in a network.

- Redundant power supply and power management - WRS only contains one power supply unit (PSU), so it is susceptible to both its failure and failure of the power circuit it is connected to.

- Backup timing source - While the White Rabbit standard allows boundary clocks to have a backup grandmaster selected. The switch does not allow to use external signal as a timing source when configured as boundary clock. In the same manner, grandmaster clocks cannot switch over to a boundary clock mode in case of external reference failure. These errors have to be detected manually and the switch reconfigured, but this means a downtime of multiple minutes in the best case scenario. In the worst case, this error goes undetected and the whoel downstream network becomes unsynchronized to the source time scale.

Some of these issues, mainly the redundant PSU are to be solved in 4th version of the switch (WRS-v4) [20], however as of writing this thesis, this version is not yet available and is expected to release at the end of 2024.

2.5.1.2 White Rabbit Switch Z16

All of the problems mentioned above are currently solved by a commercial implementation of the White Rabbit protocol, in a White Rabbit Switch Z16 made by Safran [21]. It features not only redundant, but also hot-swappable PSUs as well as fans, which make it ideal to use in a application, where keeping the downtime as low as possible is crucial. It's also fully configurable through a modern web interface. In terms of redundant timing sources, it offers multiple inputs organized in a cascade. When one source fails, the one which is configured next in order takes over. However the first timing source is only used in case the backup source fails. This prevents frequent switching between sources, which can lead to inaccuracies. This failover is fully automatic and takes only around 30 seconds, until the oscillator in the switch becomes locked to the new source clock.

Chapter 3

Measurements and Results

This chapter explains different experiments that were conducted during the project, as well as results of those experiments. All the experiments were done in a netowrk laboratory provided by CESNET.

3.1 Clock offset measurement

The difference between clocks of the two devices will be compared using Time and Frequency counter. This counter will be connected to PPS output of the clocks and will be measuring their phase offset. All of the tested devices have the PPS output low on default and raise the voltage every second, so rising edge of the signal was used to trigger the time counter. The only exception is the Intel card, which is the other way around, falling edge was used instead.

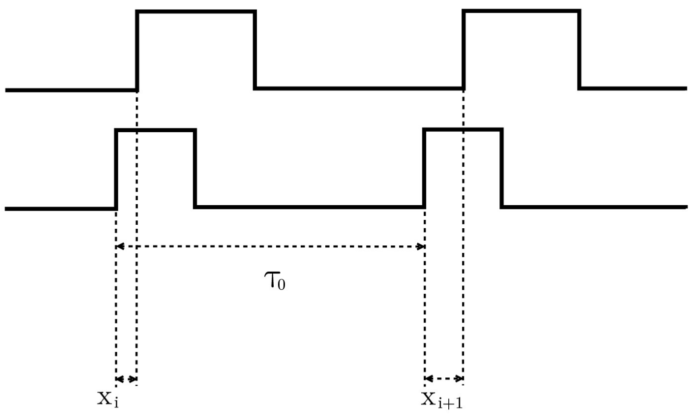


Figure 3.1: Phase offset measurement

Please note that it was not always possible to use cables of the exact same length, so some phase offset error was introduced this way. However even when some accuracy was lost this way, the precision of the synchronization remains unchanged and can be trusted.

3.2 Data Gathering and Processing

The data was gathered from SR620 using a serial connection. After each experiment, some preprocessing of the data was needed. In order to set the sample rate of the time counter to one second, only positive differences between samples were possible to be measured. This means, that when the pulse of the slave occured before on from reference clock, instead of measuring time difference x_i, $1 - x_i$ was measured. In order to compensate for this error, 1 second was subtracted from all samples whose value is greater than 0.5 seconds.

3.3 Data Evaluation

After filtering the data, they need to be evaluated. Several metrics exist for analyzing stability of frequency, which is the primary goal of time transfer.

3.3.1 Time Error

Time error $TE(t)$ or $x(t)$ is the measurement of immediate synchronization error in time t:

$$TE(t) = x(t) = T(t) - T_{ref}(t)$$

T is the time function of the tested clock and T_{ref} is the time function of the reference clock. $x(t)$ represents the phase offset of the two clock at time t.

In this case, time error is represented as a series of sampled values $\{x_i\}$, such that

$$x_i = x(t_0 + (i-1)\tau_0)$$

where t_0 is the time of the start of the measurement and τ_0 is the sampling rate, which was set at one second for all conducted tests. [22]

3.3.2 Time Interval Error

Time Interval Error (TIE) indicates, how much has the offset changed over time. It is not a function of time, but a function of the size of the observation interval τ. TIE interval of size τ starting in time t_0 is defined as:

$$TIE_{t_0}(\tau) = TE(t_0 + \tau) - TE(t_0)$$

or, for the measured data

$$TIE_{t_0}(\tau) = x_{N_\tau} - x_0$$

where N_t is the number of samples in the interval of size t.

$$N_t = t/\tau_0 + 1$$

In this case $N_\tau = \tau/\tau_0 + 1$, or the number of samples in observation interval. Effectively it is the difference between the last and first value of the observation interval. [22]

3.3.3 Maximum Time Interval Error

Maximum Time Interval Error (MTIE) is industry standard metric for evaluating clock stability. MTIE is also function of the size of the observation interval τ. This observation interval is moved through the measured data points and each time, the peak to peak variance is observed, the value of MTIE for that τ is the maximum of all those observations. Its mathematical definition for size of observation interval τ and duration of measurement T is:

$$MTIE(\tau, T) = \max_{j=1}^{N_T - N_\tau + 1} \left(\max_{i=j}^{N_\tau + j - 1} (x_i) - \min_{i=j}^{N_\tau + j - 1} (x_i) \right)$$

where N_T is the total number of samples and N_τ is again the number of samples in the observation interval.[23]

MTIE is a sensitive metric, meaning when there is one offset that is significantly higher than the rest, it will hide all other values and can skew evaluation of the clock stability. It is therefore important to carefully filter the measured data and identify and remove any outliers, that were likely caused by an error in measurement.

It's important to note, that both TIE and MTIE represent precision of the synchronization, not accuracy, they do not correspond with the absolute offset of the two clocks, just the stability of that offset.

3.3.4 Standard Variance

Standard variance is defined as:

$$s^2 = \frac{1}{N-1} \sum_{i=1}^{N} (y_i - \bar{y})^2$$

where N is the number of samples, y_i are the fractional frequency values ($\Delta f/f$), and $\bar{y}$ is the average frequency. It's not suitable for measurements of frequency stability, because it does not converge for some types of FM noise. [24]

3.3.5 Allan Variance

Allan variance is similar to standard variance and is also used to measure fractional frequency fluctuations, but unlike standard variance, it converges for most types of noise, which makes it much more suitable for measuring frequency stability. [25]

$$\sigma_y^2(\tau) = \frac{1}{2(M-1)} \sum_{i=1}^{M-1} [y_{i+1} - y_i]^2$$

3.3.6 Time Deviation and Time Variance

Time Deviation (TDEV) is the square root of Time Variance (TVAR)

The definition of TVAR is:

$$\sigma_x^2(\tau) = (\frac{\tau^2}{3}) \cdot Mod\sigma_y^2(\tau)$$

where $Mod\sigma_y^2(\tau)$ is modified Allan variance.

TDEV is then defined as:

$$\sigma_x(\tau) = (\frac{\tau}{\sqrt{3}}) \cdot Mod\sigma_y(\tau)$$

On a log-log plot, TDEV has the similar slope to MDEV (modified Allan deviation), but it is transposed by +1 a normalized by $\sqrt{3}$. [24]

TDEV is the most used metric to calulate freqency stability, it was therefore selected as the main metric to compare measured data.

3.4 Stable32

While it is possible to calculate both MTIE and TDEV manually or by creating a script to process gathered data, there already are computer programs created for that purpose. One of the most popular ones is Stable32 developed by William J. Riley [26].

Stable32 is a complex tool for frequency stability analysis, it can:

- Parse data from a file.
- Filter outliers based on deviation.
- Calculate different metrics to evalutate frequency stability.
- Plot calculated data to a chart

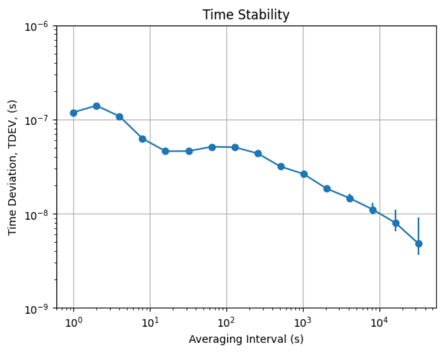
For our purpose, Stable32 will be only used for calculating either $MTIE$ or $TDEV$, visualization of those metrics will be done using Matplotlib, which provides more variability and allows for easier comparison, since the ranges of the different plot can be set manually and produced images look better overall.

3.5 PTP

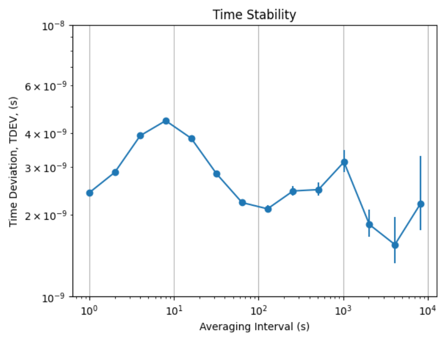
This section explaines all the different experiments conducted using IEEE 1588 for time synchronization.

3.5.1 Direct connection

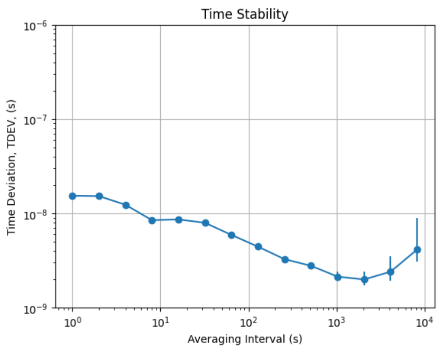
For baseline experiments for each card, they were connected directly to the PTP grandmaster and the phase offset of the PTP clock compared to the reference was measured.



(a) Intel I210



(b) Solarflare SF432



(c) Meinberg PTP270PEX

Figure 3.2: Direct connection to PTP grandmaster

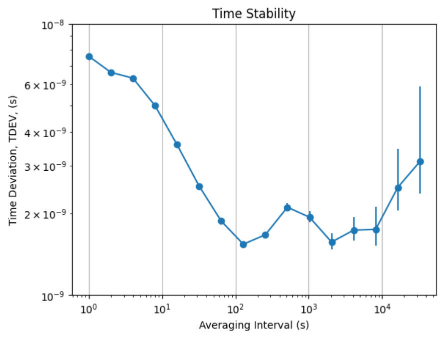
3.5.2 Transparent clock

For transparent clock measurements, one of the provided network switches was placed between the PTP grandmaster and end device. The phase offset between those two devices was again measured.

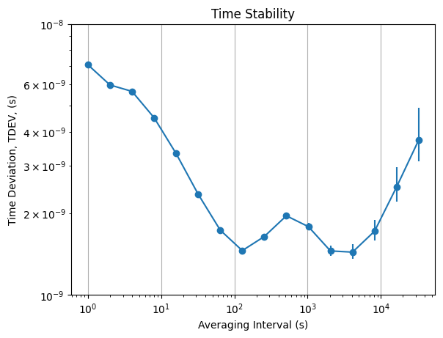
3.5.2.1 Cisco

Since the switch by Cisco is an enterprise switch, the PTP capabilities are not the primary focus. Its configuration is not straightforward and at the time of conducting the experiments was not documented very well. It took 2 months of communication with technical support, until PTP started working, this doesn't make it ideal for installation in environments that don't already have Cisco infrastructure with support staff for large scale operations.

While troubleshooting PTP over UDP, measurements were taken to compare it against layer 2.



(a) Layer 2



(b) Layer 3

Figure 3.3: TDEV of Cisco switch using different transport layers

As it is clear from the graphs in figure 3.3, both of these options behaved performed almost exactly in the same way.

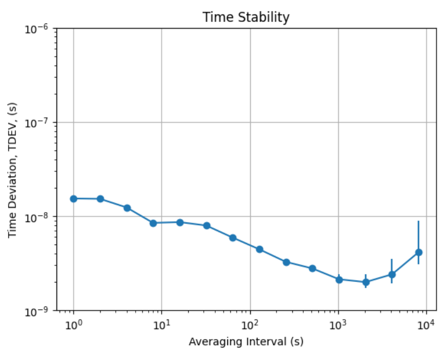
3.5.2.2 Hirschmann

Since two Hirschmann switches were available, it was possible to connect them in a series and form a PTP cascade, to see how a number of TCs on a path influence the quality of synchronization. The card from Meinberg was used in these cases.

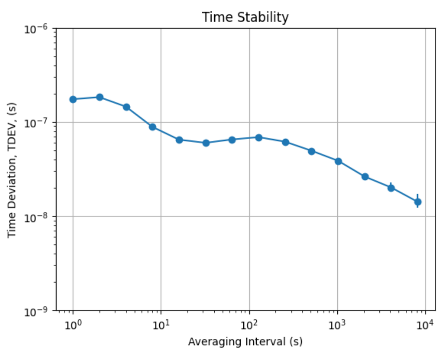
Interestingly, precision of synchronization only when going from direct connection to one TC. Adding another switch into the cascade did not affect the precisiou whatsoever. When comparing average of the offset in table 3.1, all 3 options performed very similar in terms of accuracy of the synchronization.

Configuration	**Average Phase Offset**
Direct Connection	331.67 ns
1 Hirschmann switch	351.81 ns
2 Hirschmann switches	342.97 ns

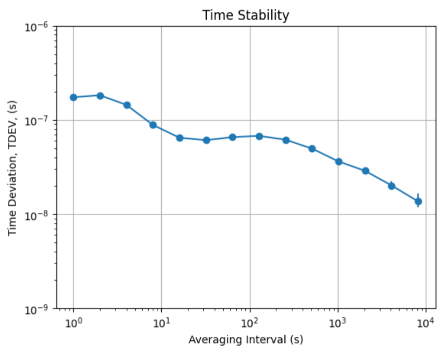
Table 3.1: Average phase offset of PTP cascade



(a) Direct connection



(b) One Hirschmann TC on the path



(c) Two Hirschmann TCs on the path

Figure 3.4: Direct connection to PTP grandmaster

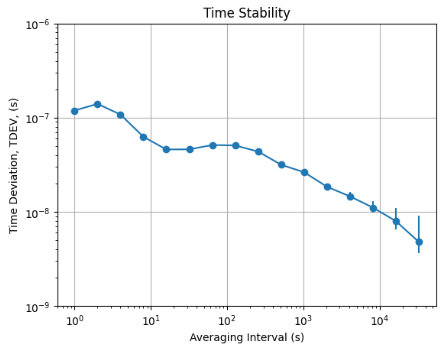
Even though the synchronizatin quality was not affected when chaining multiple transparent clocks, overall Hirschmann switches performed worse than Cisco.

3.5.2.3 Planet

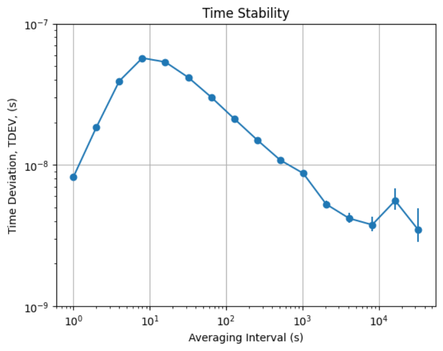
Thanks to its ease of configuration, TC by Planet was used to compare how all 3 cards perform when they are connected through a transparent clock. TDEV of these results are shown in figure

3.5.2.4 No PTP support

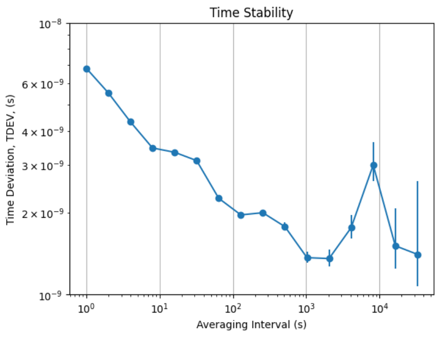
Next to the PTP enabled switches, there was also one regular network switch provided, it can be used to not only illustrate the benefits of using transparent clocks in a network, but its performance can be measured against a transparent clock with timestamping disabled.



(a) Meinberg PTP270PEX

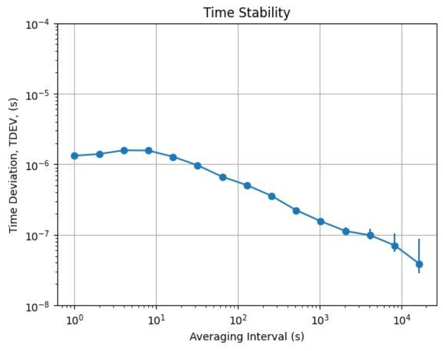


(b) Solarflare SF432

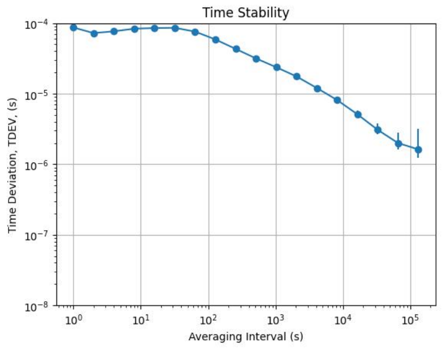


(c) Intel I210

Figure 3.5: All cards connected through Planet IP40



(a) Non-PTP switch Linksys



(b) Cisco with PTP disabled

Figure 3.6: Time deviation of switches without PTP

3.5.3 Fully SW implementation

This section is focused on linuxPTP project along with NICs from Intel. Together, they can provide a complete PTP solution for a fraction of the cost of commercial products, however their implementation is not as robust as dedicated PTP hardware and they are harder to manage with PTP stack split between multiple daemons. While it might be enough for a small research project, it is not suitable for industrial or enterprise applications.

3.5.3.1 Direct connection

This time the grandmaster was used in freerunning mode, so it was distributing the time scale of its internal oscillator. Because of this, it was needed to extract the clock pulse from the internal clock for it to be used as reference clock. One option to do so would be by using interrupts on the CPU of the system running the PTP stack and send a pulse to serial port this way. However this was not used in the end, because the noise introduced by different processing delays when creating that pulse was bigger than the phase offset and therefore render all the samples useless. The other option that was used in the end was to use the connection on the Intel card itself. LinuxPTP provides a *ts2phc* utility that can take signal from the PTP clock on the card and send it to one of the software defined pins.[27] SR620 can then be connected to said pin using a probe and time could be measured this way. The same method was used when connecting the slave device to time counter.

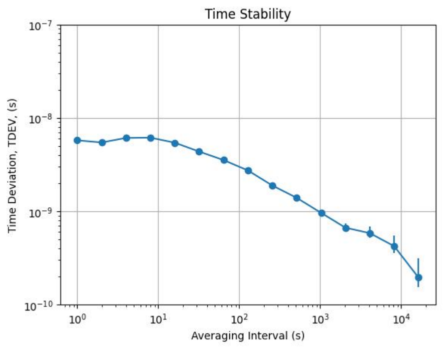


Figure 3.7: Time deviation of Direct connection to SW grandmaster

Interestingly, the graph of time deviation shows better results than the direct connections in previous section. This could be caused by the simpler setup, that eliminates inaccuracies caused by synchronizing the GM to the source clock.

3.5.3.2 Boundary Clock

Intel i210 cards along with linuxPTP provide the option to use two or more network cards in a boundary clock configuration. The internal clock of a host is synchronized with the information received on a card configured as slave and is distributed further to master cards. This solution is cheaper than commercial boundary clocks.

While linuxPTP also supports transparent clock, which would be the better option to use in a device without a precise clock, it does not work the same way as boundary clock mode, instead of using the clock of the system to transfer the time scale between the two cards, it requires connecting the pins of two cards with cables. Therefore it is expected that BC will be less accurate over time compared to direct connection

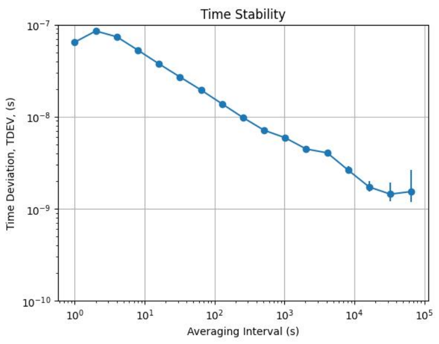


Figure 3.8: Time deviation of connection to SW grandmaster using a SW BC

This assumption was confirmed by figure 3.8 when compared to figure 3.7. Boundary clock was one order of magnitude less accurate than direct connection. However it was still providing sub microsecond precision. When comparing the average phase offset of the two configurations, they were closely matched 85 and 67 nanoseconds for the direct connection and boundary clock respectively.

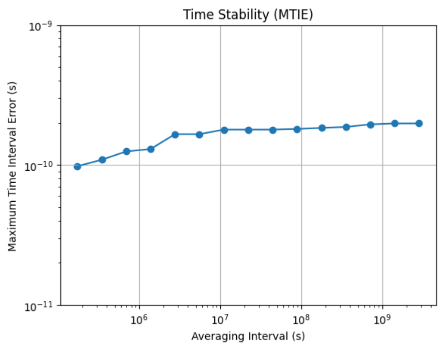
3.6 WR

White Rabbit switches contain PPS output, which can be used for measuring the time offset between slave and a master. This does not play a role in stability of the synchronization, since that signal will only be offset by propagation delay in the wire from reference clock to master switch. All experiments in this section were done this way.

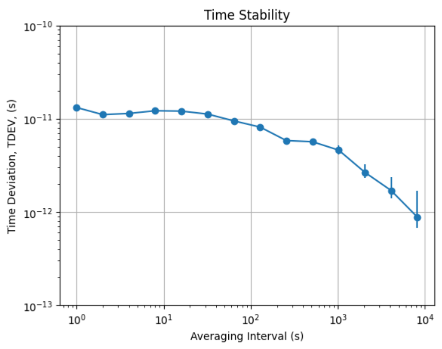
3.6.1 Direct Connection

Similar to PTP experiments in previous section, first a baseline needed to be established. Two WRSs were connected and their offset was measured. For higher accuracy of measurement, instead of using the source clock as timing reference for measurements, the output of WRS in a GM role was used. After connecting the switches using a 2 meter long G652 fibre patch cable.

A benefit of WR in contrast of the regular PTP is immediate phase lock thanks to WR Link Setup. It was therefore not necessary to wait until the offset became stable.



(a) MTIE



(b) TDEV

Figure 3.9: Direct WR connection using fibre patch cable

As it is clear from the shown graphs, White Rabbit works as advertised and achieves sub nanosecond precision in this setup.

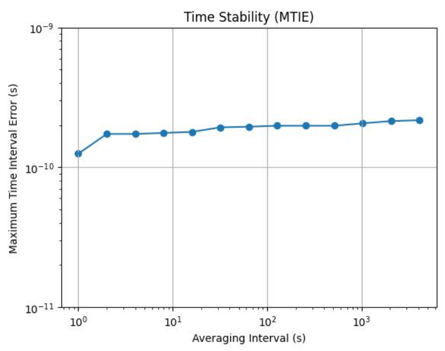
3.6.2 Connection Through a Switch

For the second test, third switch was used in a cascade, similar to PTP tests. Phase offset of the border devices was measured for comparison with direct connection.

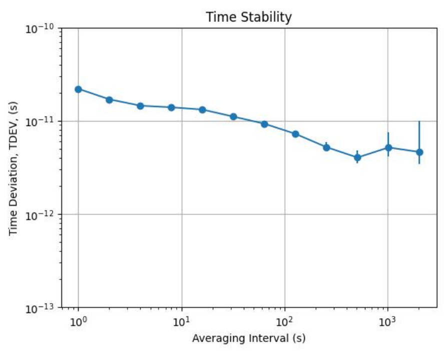
Because of SyncE, all three clock in WR switches were running with the same frequency, the stability of the transmission therefore was not affected. Even without calibration, WRS provided excellent precision over 1 hop, which is still well below the guaranteed nanosecond. However, there is slight drop in stability over longer intervals, as can be seen on the graphs 3.9 and 3.10, namely for τ above 10^3.

3.6.3 Long Distance Transfer

One of the main purposes of White Rabbit is synchronization of clocks over long distances, where PTP is not viable and time transfer is usually done using GNSS. BX transceivers have a physical limitation of 10km, their laser is not powerful enough to carry signal further. So another technology has to be employed to use WR for longer connections. WR Good Practice Guide [28] recommends using Dense Wavelenth Division Multiplexing (DWDM).



(a) MTIE

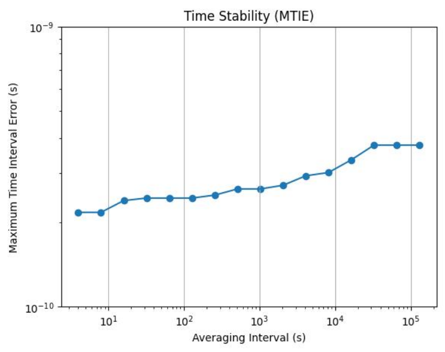


(b) TDEV

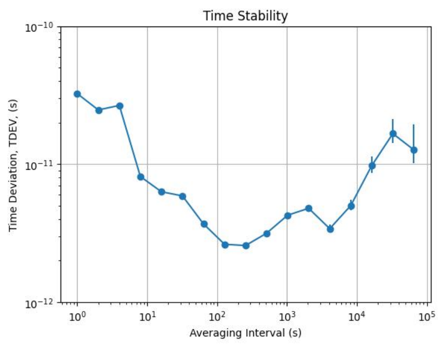
Figure 3.10: Connection over one WR BC

DWDM SFPs allow transmission for up to 120km kilometres and since they use narrower band, they are much easier to amplify, which makes them ideal for long distance transfers. Added benefit of DWDM is much lower chromatic dispersion, because different available wavelengths are much closer than in the case of BX, the smallest difference between two channels is only 0.8 nm, compared to standard 180 nm in case of BX.

Two separate experiments using DWDM were conducted, first one in laboratory environment, spools of optic fiber with total length of 150km were used to emulate long distance fiber path. Optical amplifiers were placed in between the spools to make the signal strong enough. Results of this experiment are shown in figure 3.11



(a) MTIE

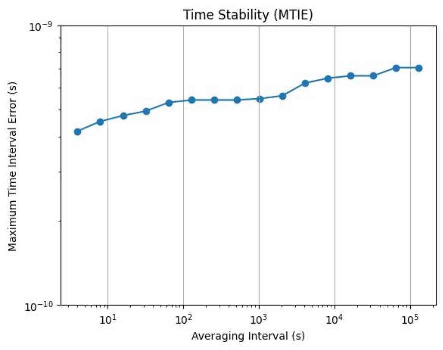


(b) TDEV

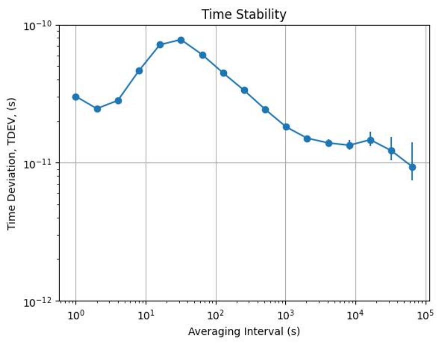
Figure 3.11: MTIE and TDEV for DWDM connection in a lab

The other one used an already deployed fiber optic network connection from CESNET laboratory in Prague to Neratovice and back. This connection also used amplification.

As you can see in figures 3.11 and 3.12, the connection in lab provided more precise synchronization for τ between $10^1 s$ and $10^4 s$.



(a) MTIE



(b) TDEV

Figure 3.12: MTIE and TDEV for DWDM connection in a real network

3.6.4 PTP in White Rabbit Switches

White Rabbit Switches not only support White Rabbit PTP profile, but also default IEEE 1588-2008. In this profile, the main benefits of White Rabbit are not in use, so worse performance is expected. This option can be used, when there is a timing network backbone consisting of WR devices and some end devices only support PTP. These results are again a bit skewed since the phase offset was measured directly between grandmaster and slave devices, however, this should not have an effect on either TDEV or MTIE calculations.

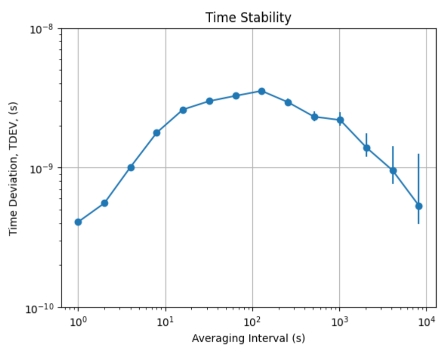
3.6.4.1 Direct Connection

For this experiment, one WRS was setup as a GM and other one as BC. One port on each device was configured for raw PTP over ethernet. Similar to previously tested devices, it was necessary to wait for the switches to become synchronized and their offset stable, since they were not using Link Setup to negotiate their offset. Even though the switches were exchanging timing information and one became synchronized to the other one based on the measured offset over time, the slave switch was still reporting in *wr_mon* that it was free running, because it relies on WR messages to determine whether the clock is locked or not.

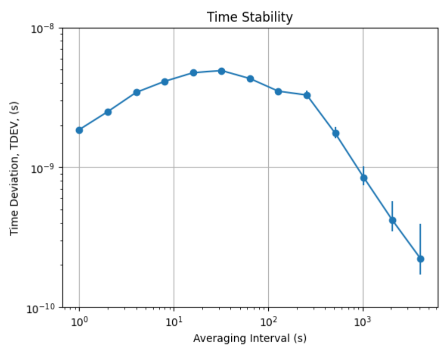
The graphs in figure 3.13 show that even when connected through the switch, the stability of the transmission is affected only in the short term, however this still yields excellent short term results compared to regular PTP solutions, most likely thanks to high quality OCXOs used in Low Jitter WRSs.

3.6.4.2 Connection Through a Switch

The same procedure was taken, as when using WR for synchronization of 3 switches in series. Since WRS does not support TC mode, the middle switch



(a) Direct PTP connection



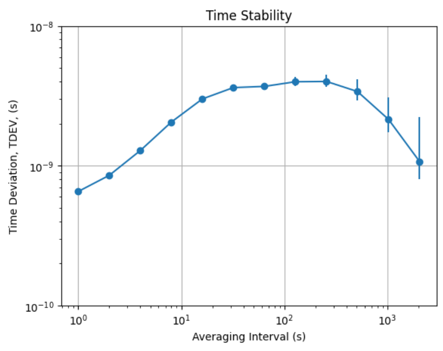
(b) Connection through one BC

Figure 3.13: TDEV of clock synchronized using PTP in WRSs

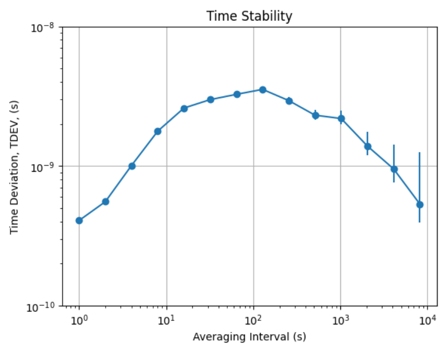
was also configured in BC mode, with two ports switched from WR profile to raw PTP.

3.6.4.3 Copper and Fiber Transceivers

Since White Rabbit contains SFP ports, the difference between copper and fiber connection was measured as well. As mentioned in 1.6.2, copper SFPs cannot be used to carry timing information for WR protocol since they do don't support use of an external clock for data encoding. They can be used for PTP synchronization however.



(a) Copper connection



(b) Fiber connection

Figure 3.14: TDEV or WRSs connected using copper and fiber respectively

It is shown in figure 3.14 that over short distances, copper cables don't make a difference. Unfortunately, with 100m being the length limit of Cat5e ethernet cables, it is not possible to create long enough connection, where propagation delay would make a significant impact.

Chapter 4

Monitoring and Calibration

This chapter talks about additional work, that was done for support of WR protocol. First is a solution for White Rabbit monitoring of own design. This chapter also mentions White Rabbit calibration, both the official method to calibrate fiber asymmetry in single fiber transfers and also proposed method, that can be used to calibrate WR connections using pair of unidirectional fibers.

4.1 White Rabbit Monitoring

Since the intention is to deploy the White Rabbit Switches on a country wide network, the need to monitor their state arose. The switches themselves provide a CLI utility called `wr_mon` to report state of its clock and ports to the user. This is a sufficient solution when there is a need to debug something while operating a couple of switches, but not for a large number of devices. White Rabbit Switches provide SNMP support exactly for that purpose. Following monitoring stack was used.

4.1.1 SNMP

Simple Network Management Protocol (SNMP) is a protocol developed for monitoring and configuration of network devices. White Rabbit Switches don't implement its 3rd version, so they lack support of user based authentication and encryption. Instead, they are protected by so called community strings. There are two types of these strings, public and private, they serve as a password to SNMP values, public one to read the value and private to change it. Unfortunately those strings are exchanged between the agent and the host unencrypted, so the whole system is susceptible to man in the middle attacks This is less of an issue when using SNMP only for monitoring and it will most likely be ran on a separate private management network. However it is still a factor worth considering.

4.1.1.1 Management Information Base

All of the data is organized in a tree structure defined by a Management Information Base (MIB). This structure defines the location, format, description and possible values of the data. The tree structure starts with a root and each subtree represents a category of data, values themselves are stored in leaves of the tree. Each node in the tree is identified by a unique OID (object identifier), OID is a sequence of numbers in which each number corresponds to a layer of the tree. Alternatively the numbers can be represented by strings to make configuration simpler, for example value in a tree can be represented either as `.1.3.6.1.4.1.96.100.7.5.1.5` or

```
iso.org.dod.internet.private.enterprise.cern.wrSwitchMIB.
wrsExpertStatus.wrsPtpDataTable.wrsPtpDataEntry.
wrsPtpServoStateN
```

Following is an example of a definition of a leaf that represents an enumerated value, taken from the MIB of the WR Switch.

```
wrsPtpServoStateN OBJECT-TYPE
SYNTAX INTEGER {
            uninitialized(0),
            syncNsec(1),
            syncSec(2),
            syncPhase(3),
            trackPhase(4),
            waitOffsetStable(5),
            standardPTP(99)
}
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
        "Numeric representation of servo state"
::= { wrsPtpDataEntry 7 }
```

To avoid conflict and keep the OIDs universally unique, subtrees are assigned to various organizations, that then manage that subtree and assign subtrees of that subtree further.

4.1.2 Data storage

Since all monitoring data will consist of numbers with timestamps, the easiest and most performant option to store the data will be a time series database (TSDB). TSDBs are databases optimized for storing measurements and events over time, they also offer some specific features, such as aggregation of old data. Compared to conventional relational database, data older than a set threshold are aggregated to less precise data points, this can be done many times. For example with default collection every minute, data older than one month can be reduced to a data point every 10 minutes, data older than a year every hour etc. One of the most popular time series databases is InfluxDB. Ultimately Victoria Metrics was chosen as the back end for monitoring WRSs. Victoria Metrics is a complete monitoring solution, which runs InfluxDB TSDB at its core. [29]

4.1.3 Data Collection

The data offered by the WRSs over SNMP needs to be periodically queried, processed and uploaded to the TSDB. It's possible to use REST API of InfluxDB directly or use a library to create a script for data collection. However the people behind InfluxDB also created a data collection agent called Telegraf. [30]

Telegraf is a simple utility written in Go, that offers many plugins both for input and output, for our purpose we will use SNMP input plugin with influxdb_v2 output plugin. We will also need to setup a execd plugin that runs a python script which converts measured sfp rx and tx power from micro watts to dBm, which couldn't be done by Telegraf itself unfortunately. A simple ping plugin, which checks whether the switches are online or not, was configured as well.

4.1.4 Data visualization

After the data is collected, we still need a way to visualize the data. The most popular application for that is Grafana.[31] Grafana is a highly customizable tool that can generate charts and other visualizations using the data provided by various data sources. Those visualizations are organized into so called dashboards, so all the data is easily accessible. Grafana was also chosen because it's already in use by CESNET for monitoring other devices, so it would be more convenient to have access to all the data in one place. Grafana doesn't support Victoria Metrics directly, but since it Prometheus Query Language (PromQL), it can be added as a Prometheus data source. PromQL is a simple query language specifically for time series databases. `wr_ptp_clock_type{url="$wr_url"}` returns data from `wr_ptp_clock_type` table where `url` matches variable `wr_url`. This allowed for displaying information about one specific switch, selectable in a drop down menu. Since the Victoria Metrics support Float64 as the only data type, values of enums were stored as such, we then need to map those values back to strings according to the definition in MIB, for example the servo stated shown in the excerpt above.

Example screenshot of the interface created in grafana is shown on figure 4.1

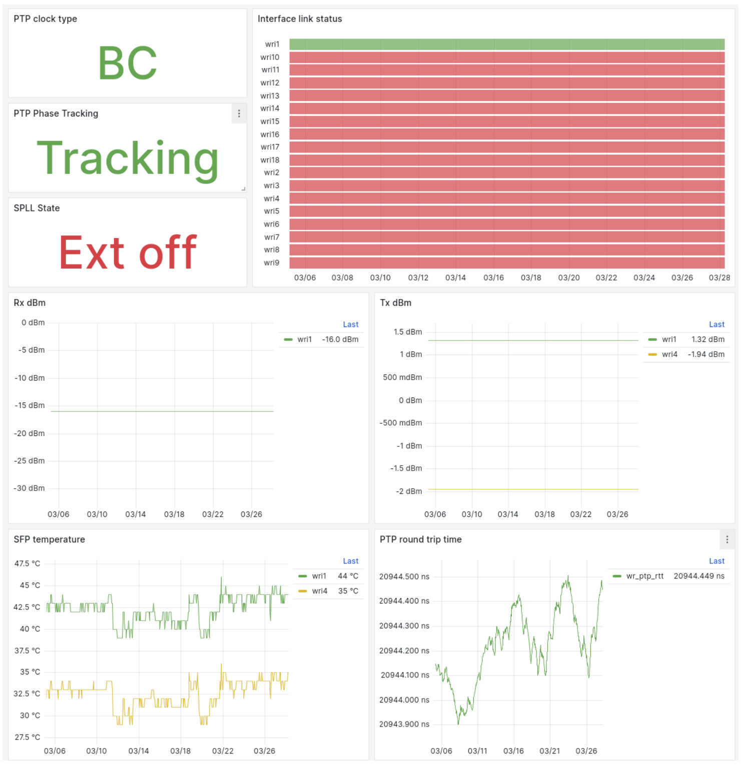


Figure 4.1: White Rabbit Switch represented in Grafana

4.2 White Rabbit Calibration

To take advantage of the Link Delay Model described in 1.6.2, White Rabbit Switches need to be calibrated.

For White Rabbit calibration, a fiber cable with known latency and asymmetry is needed, measuring a reference wire will therefore be the first step of the calibration process. The calibration process is explained in [32]. This document was the main source for this section. However, the WRS software has gone through significant changes since this guide was published, so it had to be adapted for the current version.

4.2.1 Fiber latency measurement

Latency of the reference fiber can be measured using round trip packet delay from master to slave and back. However when measuring this delay, it not only includes fiber latency, but also fixed delays in master and slave, as well as their bitslide values. Fixed delays are

- Δ_{TXM} - Transmit delay of master,
- Δ_{RXM} - Receive delay of master,

- Δ_{TXS} - Transmit delay of slave,
- Δ_{RXS} - Receive delay of slave,

Bitslide is defined as “Delay caused on both sides by aligning the recovered clock signal to the inter-symbol boundaries of the data stream.”

The complete master to master delay can be expressed as:

$$delay_{MM} = \Delta_{TXM} + \Delta_{RXM} + \varepsilon_S + \Delta_{TXS} + \Delta_{RXS} + \varepsilon_S + \delta_{MS} + \delta_{SM} \tag{4.1}$$

However bitslide values (ε_S and ε_M are known and reported by the WR system, so they can be subtracted from this equation for easier calculations.

$$delay'_{MM} = delay_{MM} - \varepsilon_S - \varepsilon_M \tag{4.2}$$

For fiber latency calibration, two fiber cables of different length are used, one is short cable (f_1), only used for calibration, the other one is up to multiple kilometres long and will be used in the deployed WR network (f_2), this is the cable that needs to be measured.

The latency of fiber f_2 was then measured:

1. Two White Rabbit switches were configured as a grandmaster and slave respectively and a connection between them using f_1 was made.
2. After synchronization, round trip delay reported by `wr_mon` utility was noted ($delay_{MM1}$). According to the guide bitslide values were also supposed to be shown by `wr_mon`, that was not the case however. Bitslide was read through the web application, that was only temporarily enabled for this purpose, the bitslide value is reported by Endpoint tool. [33] Values measured were $delay_{MM1} = 1051512ps, \varepsilon_S = 11669ps, \varepsilon_M = 11645ps$
3. The same process was repeated with f_2 to obtain values of $delay_{MM2}$, ε_{S2} and ε_{M2}. Namely $delay_{MM2} = 72943799ps, \varepsilon_S = 5717ps, \varepsilon_M = 5786ps$
4. Both fibers were connected using an adapter and another measurement was made using this cable. This way $delay_{MM3}$, ε_{S3} and ε_{M3} were measured. $delay_{MM3} = 72964408ps, \varepsilon_S = 118ps, \varepsilon_M = 87ps$
5. Bitslide values can be subtracted from delays as described by 4.2. This equation was used to calculate $delay'_{MM1}$, $delay'_{MM2}$ and $delay'_{MM3}$ from measured values.
6. The round trip latency of both of the fibers can be calculated.

$$\delta_1 = delay'_{MM3} - delay'_{MM2} = 32007ps$$

$$\delta_2 = delay'_{MM3} - delay'_{MM1} = 91936032ps$$

4.2.2 Fiber Asymmetry

In previous section, the round trip asymmetry of the fiber used, however, as described by equation 4.1, the latency in the fiber consists of two parts, each representing one direction.

$$\delta = \delta_{SM} + \delta_{MS}$$

In order to calibrate WR switches, the asymmetry of the fiber needs to be calculated. Fiber asymmetry α is defined as

$$\alpha = \frac{\delta_{MS} - \delta_{SM}}{\delta_{SM}}$$

.

WR calibration process calculates fiber asymmetry by measuring phase offset of two WR switches, first interconnected using f_1 and then also f_2. Since fixed delays are identical for both setups, the only difference between those two measurements will be caused by fiber asymmetry. Cable f_1 is deemed short enough, that its asymmetry does not affect the synchronization in any meaningful way.

$$\alpha = \frac{2(skew_{PPS2} - skew_{PPS1})}{\frac{1}{2}\delta_2 - (skew_{PPS2} - skew_{PPS1})}$$

Skew in the context means phase offset between slave WR and master WR ($skew_{PPS} = t_S - t_M$). δ_2 is round trip propagation delay of cable f_2 calculated in previous step.

In this case $skew_{PPS1}$ was measured at $71ps$ and $skew_{PPS2}$ at $4639ps$, this was the total delay caused by chromatic dispersion. The fiber asymmetry was then calculated at $\alpha = 2.54003 * 10^{-4}$, which differs from the default asymmetry of $2.6787 * 10^{-4}$.

Value α was entered into the fiber cable database on the white rabbit switch using `wrs_nconfig` utility and this fiber was selected for the used port on both devices.

4.2.3 SFP calibration

After calibrating the fiber used in WR network, phase offset of the two switches should remain constant independently of the length of the cable itself. There are still delays happening in each of the switches on transmission and reception. For this calculation, an assumption of delay symmetry between both switches, as well as Rx and Tx directions on said switch. It is therefore necessary to match setups of both switches as precisely as possible. Both switches were the same model, specifically the Low Jitter version of White Rabbit switch made by Seven Solutions, which ensured that both delays caused by PCB traces will be equal, similarly, both switches had the same bitstream uploaded in their respective FPGA. Also both SFPs on either end of the connection need to be the same, BSFP-GE43-20D and BSFP-GE34-20D were used for down-link and up-link respectively. Those transceivers are the exact same model and differ only in the wavelengths used, this means that they should be closely matched in terms of delays.

Since both devices are the same, it is safe to assume that their delays are equal:

$$\Delta_{RXM} + \Delta_{TXM} = \Delta_{RXS} + \Delta_{TXS}$$

The total delay caused by Rx and Tx latencies in both switches is:

$$\Delta_{RXM} + \Delta_{TXM} + \Delta_{RXS} + \Delta_{TXS} = delay'_{MM1} - \delta_1$$

Delay caused by the master device is then:

$$\Delta_{RXM} + \Delta_{TXM} = (delay'_{MM1} - \delta_1)/2$$

It would be really complicated to measure Rx and Tx separately, for example measuring delay between electrical impulse entering the SFP and the laser producing light, or delays inside the FPGA. It's therefore assumed, that the WR device has no asymmetry between its input and output.

$$\Delta_{RXM} = \Delta_{TXM}$$

Both of those values are then equal to quarter of the fixed delay

$$\Delta_{RXM} + \Delta_{TXM} = (delay'_{MM1} - \delta_1)/4$$

After calculating the delays of the SFP, they were entered into SFP database using `wrs_nconfig` tool.

After this step, a calibrator device was created, which can now be used to calibrate the delays in other WR devices.

4.2.4 Calibration of unidirectional fibers

Even though White Rabbit Good Practice Guide [28] recommends using a single fiber for both upstream and downstream connection, those connections are not always available in real networks. So for long distance transmissions using already deployed fiber infrastructure, a pair of unidirectional fibers has to be used. As mentioned in 1.6.2, pair of fibers are troublesome to use, because each of the fibers has different length. This could lead to differences in a order of microseconds over long distances, which is unacceptable by White Rabbit standards. This difference in fiber length causes a difference in propagation delay, which need to be accounted for. Unfortunately, WR Calibration Guide [32], does not provide any information on this topic.

Following method was therefore proposed for calculating the difference in delay of the two fibers.

4.2.4.1 Requirements

Since already deployed fiber is being used, WR devices on either end of the connection cannot be connected to time counter to measure their offset. Therefore a precise time source has to be located at both locations to be used as a reference clock for measurements. This time source has to provide better precision than WR, so it is possible to evaluate it, while GNSS receiver with a calibrated OCXO can be used, an atomic clock would be a better option. Of course to perform the measurements, a time counter to measure the phase offset has to be present at the slave side of the connection as well.

Before the measurements can performed, the WR devices has to be configured. The remote one is connected to precise 1PPS and 10 Mhz input and is in grandmaster mode, local one is in BC mode. It's important that transceivers that use the same wavelength for both directions. Also make sure that α of the fiber used is set to 0 in databases in both WR nodes, since the two directions of communication are using the same wavelength, no asymmetry due to chromatic dispersion is present.

4.2.4.2 Calibration Process

Take look at figure 1.2 and equation 1.1. If the path symmetry requirement is satisfied, one way path delay is:

$$\Delta_{SM} = \Delta_{MS} = \frac{(T_4 - T_1) - (T_3 - T_2)}{2}$$

,

where Δ_{SM} is path delay from slave to master and Δ_{MS} is path delay from master to slave.

Offset of the two clocks is:

$$\theta_0 = \frac{(T_2 - T_1) + (T_3 - T_4)}{2}$$

Let us assume that path delay is increased in one direction by δ, without loss of generality assume that it is the path from slave to master, so:

$$\Delta_{SM} = \Delta_{MS} + \delta$$

The offset than changes by half of this error:

$$\theta_A = \theta_0 - \frac{\delta}{2}$$

When the fiber are swapped, the error affects the other direction of communication, so:

$\Delta_{MS} = \Delta_{SM} + \delta$

Phase offset of the slave clock than becomes:

$$\theta_B = \theta_0 + \frac{\delta}{2}$$

Subtracting the two equations gives us the formula for calculating δ:

$$\theta_B - \theta_A = \delta$$

The process to measure and calculate δ is as follows:

1. Connect the two switches and let them synchronize. Use `wr_mon` to check that slave is reporting `TRACK_PHASE` against the master.

2. Measure the phase offset of the slave clock and reference clock θ_A.

3. Swap the uplink and downlink fiber on both ends and wait for synchronization of the WR nodes.

4. Measure the phase offset of the slave and reference clocks θ_B. Make sure that the interval between the measurements is as small as possible (maximum of 2-3 minutes). Otherwise the frequency offset of the two clocks would drift apart when there is a frequency offset between the the two clocks.

5. Extrapolate the two datasets, so their timestamps overlap, this provides us with a way to measure their difference compensated for any frequency drift. For correct extrapolation, each of the measured samples need to be timestamped to account for the pause in data collection.

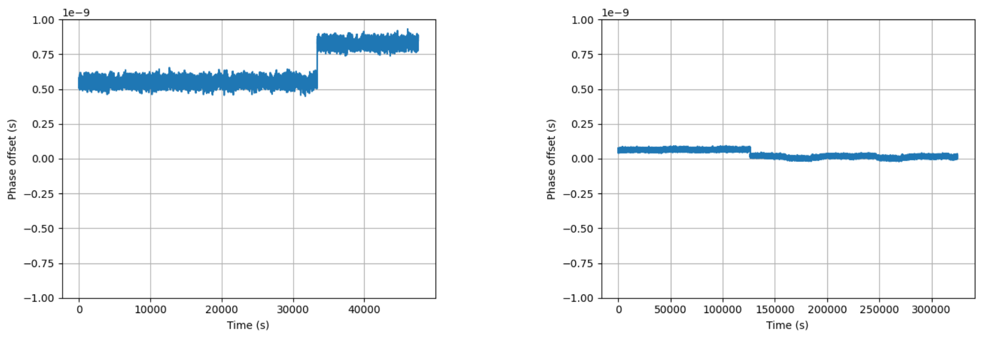
6. Select an overlapping point of the extrapolated datasets and calculate δ. Timestamps which is picked does not matter, since the frequency offset would remain constant over the two measurements, so when the data is plotted to a chart, both datasets have the same slope.

4.2.4.3 Drawbacks

This method will provide precise compensation of the delay of the WR link when using a pair of unidirectional fibers, but compared to calibration of single fibre applications, it has its drawbacks. Unlike single fiber calibration, it measures the absolute offset of the two paths, so it cannot react to changes in total path length. This is not an issue when running WR on private infrastructure, however more often than not, those path consist of at least some rented parts, where there is no guarantee of the path staying the same. Therefore a constant monitoring solution needs to be deployed, this can be done with the setup described in previous section. When the fiber path changes, the delay changes as well and therefore the clock offset measured by the time counter. The samples are then checked for any sudden jumps in their values. If that happens, the calibration process needs to be repeated.

4.2.5 Calibration results

As mentioned in section 4.2, 7km long fibre was used for calibration purposes. The fibre from previous experiments could not be used, because BX transceivers have a limit of 10 kilometres. Before the calibration itself, baseline measurements were done to evaluate the improvement. Out of the box WRS comes preconfigured with an α for a generic G652 fiber. This α was close to the real asymmetry of the fiber used, since time difference between short and long fibers was only 250 picoseconds. As stated in 1.6.2, chromatic dispersion in G652 cable should produce delay of $1486ps/km$ in one way transmission [14], so half that for two-way communication, a 7 km cable should therefore have a delay of $\frac{7 \cdot 1486}{2} = 5201ps$. When running two White Rabbit Switches without any compensation for chromatic dispersion, the offset between the switches was $4860ps$, which signifies dispersion of $1337ps/km$, when ignoring fixed delays. This indicates, that the actual α of the two cables was slightly different, calibration would therefore improve the accuracy of the synchronization, even though it was already under one nanosecond. The calibration process is described in section 4.2.



(a) Uncalibrated WRSs (b) Calibrated WRSs

Figure 4.2: Phase offset over time of uncalibrated and calibrated WR switches

Figure 4.2 shows the difference between uncalibrated and calibrated WR connection, note that the cables were switched in the middle of the experiment to showcase the difference due to chromatic dispersion. After calibration, the precision increased to around 10 ps for the longer cable and 60 ps for the shorter one. Most importantly, the different cables were much closer to each other in terms of absolute delay, which matters when transmitting to multiple slaves with different distances from the master.

Two separate spools of fiber cable were used, they were linked together for the total length of 7km. It was important to use cable short enough, that its link budget is enough for the BX transcievers and they can get the singal from one end to the other.

Since the precision of the counter is only 25 ps, it is not possible to judge the accuracy of the two clocks, because the dispersion of the measured data is below that. However it is clear, that calibration improved stability of that synchronization.

Interesting thing to note is that when connected through the long fiber, the offset slowly fluctuated by about 15 picoseconds every day, this was not present when using the short patch cable. This was likely caused by temperature differences in the lab between day and night, which changes slight shift in chromatic dispersion. This showcases one of the ways a fiber optic cable can be used along with precise timing as a sensor.

Chapter 5

Conclusion

In this thesis, different protocols for time synchronization were studied and measured.

In the first chapter, I got familiar with time synchronization problem and its issues that need to be solved. I then learned about protocols that try to solve these problems when performing time synchronization over computer networks, namely Precision Time Protocol and White Rabbit, and their different options and use cases.

After getting familiar with Precision Time Protocol I verified compatibility of provided devices, only a subset of possible configurations was compatible. I measured precision and accuracy of multiple configurations, in order to figure out which PTP implementation gives the best results.

Next part was focused on White Rabbit protocol and the benefits it has over regular PTP. First I tested the performance of White Rabbit several different networks. After doing initial tests, I performed calibration of WR switches. This calibration method was proven to be working and yielding good results, it will be used in future WR deployments. I also proposed new calibration method for calibrating White Rabbit Switches connected using a fiber pair, which is needed for CESNET network.

Finally I developed a monitoring system for White Rabbit Switches. This system is now used in trial mode in CESNET network to monitor their White Rabbit Switches.

Bibliography

1. MILLS, David. *Network Time Protocol (Version 1) Specification and Implementation.* [Internet Requests for Comments]. RFC Editor, 1988-04. RFC. RFC Editor. Available also from: `https://www.rfc-editor.org/rfc/rfc1059`.
2. FOUNDATION, Network Time. How does it work? [online]. 2022 [visited on 2024-04-09]. Available from: `https://www.ntp.org/ntpfaq/ntp-s-algo/`.
3. IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. *IEEE Std 1588-2019 (Revision ofIEEE Std 1588-2008).* 2020. Available also from: `https://ieeexplore.ieee.org/document/9120376`.
4. MICROSEMI CORPORATION. IEEE 1588 Technology. [N.d.]. Available also from: `https://www.microsemi.com/company/technology/ieee-1588-technology#how-1588-works`.
5. FOUNDATION, Network Time. IEEE 1588 Precision Time Protocol (PTP) [online]. 2022 [visited on 2024-04-09]. Available from: `https://www.ntp.org/reflib/ptp/`.
6. COLLINS, Danielle. What is IEEE 1588 and why is it important for Industrial Ethernet networks? 2019. Available also from: `https://www.motioncontroltips.com/what-is-ieee-1588-and-why-is-it-important-for-industrial-ethernet-networks/`.
7. PANNELL, Don. To 1-Step or Not to 1-Step. *IEEE 802.1 Meeting.* 2015. Available also from: `https://www.ieee802.org/1/files/public/docs2015/ASRev-pannell-To-1-step-or-not-0315-v1.pdf`.
8. DOUGLAS, Arnold. BMCA deed dive: part 1 [online]. 2022 [visited on 2024-04-18]. Available from: `https://blog.meinbergglobal.com/2022/02/01/bmca-deep-dive-part-1/`.

9. DOUGLAS, Arnold. BMCA deed dive: part 2 [online]. 2022 [visited on 2024-04-22]. Available from: `https://blog.meinbergglobal.com/2022/03/11/bmca-deep-dive-part-2/`.

10. FROST, Tim. The PTP Telecom Profiles for Frequency, Phase and Time Synchronization [online]. 2013 [visited on 2024-04-22]. Available from: `https://www.microsemi.com/document-portal/doc_view/133481-ptp-telecom-profiles-for-frequency-phase-and-time-synchronization`.

11. WŁOSTOWSKI Tomasz; Daniluk, Grzegorz. WHITE RABBIT: SUB-NANOSECOND SYNCHRONIZATION FOR EMBEDDED SYSTEMS. 2011. Available also from: `http://www.elpromatime.com/wp-content/uploads/2018/10/PTTI-2011-White-Rabbit.pdf`.

12. COFFEY, Joseph. Latency in optical fiber systems. *Commscope* [online]. 2017 [visited on 2024-04-19]. Available from: `https://www.commscope.com/globalassets/digizuite/2799-latency-in-optical-fiber-systems-wp-111432-en.pdf`.

13. JANSWEIJER, P.P.M.; PEEK, H.Z. Measuring propagation delay over a 1.25 Gbps bidirectional data link. *National Institute for Subatomic Physics* [online]. 2010 [visited on 2024-04-18]. Available from: `https://www.nikhef.nl/pub/services/biblio/technicalreports/ETR2010-01.pdf`.

14. ITU. Optical interfaces for coarse wavelength division multiplexing applications. *SERIES G: TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS* [online]. 2018 [visited on 2024-05-09]. Available from: `https://www.itu.int/rec/T-REC-G.695-201807-I/en`.

15. LI, Hongming; GONG, Guanghua; PAN, Weibin; DU, Qiang; LI, Jianmin. Temperature Effect and Correction Method of White Rabbit Timing Link. *IEEE TRANSACTION ON NUCLEAR SCIENCE* [online]. 2014 [visited on 2024-04-12]. Available from: `https://arxiv.org/pdf/1406.4223`.

16. *SR620 — Time interval / frequency counter* [online]. Standford Research Systems [visited on 2024-05-04]. Available from: `https://www.thinksrs.com/products/sr620.html`.

17. *microSyncRX : Powerful IEEE 1588 PTP Grandmaster and NTP Server* [online]. Meinberg Funkuhren GmbH [visited on 2024-05-04]. Available from: `https://www.meinbergglobal.com/english/products/multipurpose-synchronization-system.htm`.

18. *5071A Cesium Primary Time and Frequency Standard* [online]. Microchip [visited on 2024-05-04]. Available from: `https://www.microchip.com/en-us/products/clock-and-timing/components/atomic-clocks/atomic-system-clocks/cesium-time/5071a`.

19. DANILUK, Grzegorz; VAN DER BIJ, Erik. White Rabbit Switch Hardware [online]. [N.d.] [visited on 2024-04-18]. Available from: `https://ohwr.org/project/wr-switch-hw/wikis/home`.

20. VAN DER BIJ, Erik; LIPINSKI, Maciej. White Rabbit Switch Hardware version 4 [online]. [N.d.] [visited on 2024-05-09]. Available from: `https://ohwr.org/project/wr-switch-hw-v4/wikis/home`.

21. *White Rabbit Z16* [online]. Safran [visited on 2024-05-02]. Available from: `https://safran-navigation-timing.com/product/white-rabbit-z16/`.

22. BREUER, Jan. Synchronizace času v distribuovaných heterogenních měřicích a řídicích systémech [online]. 2016 [visited on 2024-05-09]. Available from: `https://dspace.cvut.cz/handle/10467/65508`.

23. BREGNI, Stefano; MACCABRUNI, Stefano. Fast Computation of Maximum Time Interval Error by Binary Decomposition. *IEEE Transactions on Instrumentation and Measurement* [online]. 2000 [visited on 2024-04-24]. Available from: `https://bregni.faculty.polimi.it/papers/mtiefast.pdf`.

24. RILEY, W. J. Handbook of Frequency Stability Analysis. *NIST Special Publication* [online]. 2008 [visited on 2024-04-24]. Available from: `https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=50505`.

25. DANIELSON, Magnus. Time deviation. 2018. Available also from: `https://en.wikipedia.org/wiki/Time_deviation`.

26. *Stable32* [online]. HTS and IEEE UFFC [visited on 2024-04-30]. Available from: `http://www.stable32.com/`.

27. COCHRAN, Richard. *Accessing the SDPs on the i210* [online]. 2022. [visited on 2024-05-03]. Available from: `https://github.com/richardcochran/i210-rework/blob/master/i210-rework.org`.

28. DIERIKX, Erik; XIE, Yan. White Rabbit Good Practice Guide [online]. 2019 [visited on 2024-05-03]. Available from: `https://ohwr.org/project/white-rabbit/wikis/uploads/7df19b6a4d0e90bf6d7b8ae32b3b32c4/WR_Good_Practice_Guide.pdf`.

29. *VictoriaMetrics: Simple and Reliable Monitoring for Everyone* [online]. VictoriaMetrics [visited on 2024-04-18]. Available from: `https://www.victoriametrics.com/`.

30. *Telegraf* [online]. InfluxData [visited on 2024-04-18]. Available from: `https://www.influxdata.com/time-series-platform/telegraf/`.

31. *Grafana: The open observability platform* [online]. Grafana Labs [visited on 2024-04-18]. Available from: `https://www.grafana.com`.

32. DANILUK, Grzegorz. White Rabbit calibration procedure [online]. 2015 [visited on 2024-05-02]. Available from: `https://white-rabbit.web.cern.ch/documents/WR_Calibration-v1.1-20151109.pdf`.

33. KVĚTOŇOVÁ, Šárka. KALIBRACE SYSTÉMU WHITE RABBIT. 2020.

Appendix **A**

Acronyms

BC Boundary Clock

BMCA Best Master Clock Algorithm

E2E End to End

GM Grandmaster

GNSS Global Navigation Satellite System

MAC Media Access Control

MIB Management Information Base

NIC Network Interface Card

NTP Network Time Protocol

OC Ordinary Clock

OID Object Identifier

P2P Peer to Peer

PPS Pulse Per Second

PTP Precision Time Protocol

SFP Small Form-factor Pluggable

SNMP Simple Network Management Protocol

SyncE Synchronous Ethernet

TC Transparent Clock

TDEV Time Deviation

TIE	Time Error
TSDB	Time-series Database
WRPTP	White Rabbit PTP profile
WRS	White Rabbit Switch
WR	White Rabbit

Appendix B
CD contents

```
readme.txt ........................... Description of media on this CD
src
  latex ................................... Source code of LaTeX
  data .............................. Raw measured data  .1 text
  thesis.pdf ................................ Compiled PDF text
```