A logo with text and blue swirls

Description automatically generated

**AALBORG UNIVERSITY**

**AI AND SOUND (AIS)**

**Documentation on Wired PTP Synchronization over Network for Audio Streaming**

**Muhammed Fatih Koc**

**AALBORG, DENMARK**

**JULY 2023**

Table of Contents

[Introduction 4](#_Toc141437881)

[1.The Precision Time Protocol 5](#_Toc141437882)

[2.Synchronizing Time with LinuxPTP 5](#_Toc141437883)

[3.How gPTP Synchronization Works 6](#_Toc141437884)

[4. Utilizing PTP for Clock Synchronization in Networked Devices 8](#_Toc141437885)

[4.1. Clock synchronization importance 8](#_Toc141437886)

[4.2. Network Requirements to Achieve Sub-100 Nanosecond Synchronization 9](#_Toc141437887)

[4.3. Ethernet Switches 9](#_Toc141437888)

[5. System Requirement: Network Connection 9](#_Toc141437889)

[6. Exploring PTP Clocks in Network Environments 10](#_Toc141437890)

[7. System Setup: Implementation of PTP Synchronization with LinuxPTP and Hardware Components 10](#_Toc141437891)

[8. Overview of the synchronized multi-PC system 12](#_Toc141437892)

[9. LinuxPTP: High-Precision Clock Synchronization for Linux Systems 13](#_Toc141437893)

[10. Explanation of code of setup in lab 14](#_Toc141437894)

[10.1. Installing Linux PTP 14](#_Toc141437895)

[10.2. PTP Setup 14](#_Toc141437896)

[Bibliography 22](#_Toc141437897)

List of Figures

[Figure 1. Packets transfer between slave and master in PTP 7](#_Toc141437898)

[Figure 2. IP address of the PC that attached with new NIC card 10](#_Toc141437899)

[Figure 3. Network Setup for PTP synchronization 11](#_Toc141437900)

[Figure 4. PTP Sychronization Block Diagram 12](#_Toc141437901)

[Figure 5. Basic Idea of the synchronous multi PC setup 13](#_Toc141437902)

[Figure 6. PTP version and Linux PTP library 14](#_Toc141437903)

[Figure 7. Ptp Sychronization Output master/slave between 2 PCs 16](#_Toc141437904)

[Figure 8 Distribution of Master offset on 8 PC separately 17](#_Toc141437905)

[Figure 9. Time offset between the PHC and the system clock: system clock sync 17](#_Toc141437906)

[Figure 10. PHC Offset Variation Across Samples 19](#_Toc141437907)

[Figure 11. PHC Offset Variation Across Samples with 8 client PCs 20](#_Toc141437908)

[Figure 12. Distribution of PHC offset 20](#_Toc141437909)

[Figure 13 Distribution of PHC offset on 8 PC separately 21](#_Toc141437910)

# Introduction

There is an increase in the demand for high quality audio transmission and ideal network performance. In this document, we concentrate on assessing Work Package 1's (WP1) success criteria in relation to a full-band sound zone system. The aim is to demonstrate dependable and a low latency audio streaming to several wireless speakers using a network with strong transmission capabilities. By confirming specified success criteria (SC) relating to data transmission speeds, latency, jitter, packet losses, and audio signal synchronization, the project hopes to reach a number of milestones. The demos will be performed to demonstrate the full-band sound zone system's capability.

In this report, we investigated the use of Precision Time Protocol (PTP) network synchronization to provide time synchronization among many devices. Time synchronization is critical in many applications, such as audio streaming and so for, where accurate timing is required for smooth operations and coordination. The IEEE 1588 standard defines PTP as a robust approach for synchronizing clocks in distributed systems with nanosecond accuracy. We want to obtain highly precise and tightly synchronized watches among a network of devices by using PTP synchronization.

**SUBJECT AREA**: PTP Synchronization over Network

**KEYWORDS:** PTP, Networking, audio streaming

# 1.The Precision Time Protocol

Understanding that each computer has a clock that shows the current time is essential for keeping the time accurate on various devices. A technique for continuously converting the system time to a universal time is necessary for synchronization. Such protocols include the Network Time Protocol (NTP) and the (PTP) Precision Time Protocol [[1]](#rf1). It is important to keep in mind in our work while considering the requirement for clock synchronization that the amount of time discrepancies is more significant than how closely the receiver clocks adhere to real time.

In order to ensure that all clients hear the audio signals at once, clock synchronisation is essential when sending audio signals to numerous clients. However, it is crucial to make clear that the synchronisation in this context relates to the reproduction/playback of the audio data, not its receipt at specific loudspeakers. Every audio data packet sent over the air has a time-stamp attached that tells the receiver when to play that particular packet. All loudspeakers need to agree on an exact sense of absolute time in order for playback to be synchronised. In order for each loudspeaker to accurately read and output the audio data at the same time, the clocks inside each one must be exactly synchronised.

PTP highlights high-precision situations by using a time representation format that includes seconds and nanoseconds in its packets. As a result, PTP holds its operating environment to high standards. The standard makes ethernet the recommended environment, suggesting a preference for shorter network lengths, in order to obtain the optimum performance. Although PTP can operate exclusively in software mode, specialized hardware is required for it to function at its peak. Examples of these instruments include network interface controllers, routers, and switches.

PTP operates over ethernet and works by using a master clock that sends periodic messages to slave clocks, which then adjust their own clocks to match the master clock. The PTP clock is a clock device that implements the Precision Time Protocol (PTP) specified in the IEEE 1588 standard. The PTP clock is typically used as a master clock to synchronize the time of other devices connected over a network, known as slave devices [[2].](#rf2)

The system clocks of slave devices can be synchronized using the PTP clock, which offers a very precise time reference. This is accomplished by the slave devices receiving PTP messages over the network, which contain timing data that enables the slave devices to modify their system clocks appropriately.

# 2.Synchronizing Time with LinuxPTP

The NTP and PTP are two protocols for synchronizing clocks across computer networks. While correct time is typically taken for granted when interfacing with equipment, these protocols are critical for all types of facilities. Precise and linked time displays on clocks and computers assist employees in industrial firms/other settings in keeping track of their operations. PTP has sub-microsecond accuracy, whereas NTP has millisecond accuracy. As a result, in this work, we will use the PTP synchronization protocol [[3].](#rf3)

A mechanism for precisely synchronizing computers over a local area network (LAN) is provided by the PTP, which is described in the IEEE-1588 standard [[4].](#rf4) On a network specifically set up for IEEE -1588, PTP can provide sub-microsecond level or even nanosecond time precision with additional hardware interface. Our target is to deploy the hardware timestamping to achieve the nanosecond level time precision.

The linuxptp [[13](#rf13)] provides some tools to carry out time synchronization such as ptp4l daemon is implemented that synchronizes the PTP Hardware Clock (PHC) from the network interface card (NIC). With the help of this linuxptp, Linux users can implement the PTP in accordance with IEEE standard 1588. PTP operates over ethernet and works by using a master clock that sends periodic messages to slave clocks, which then adjust their own clocks to match the master clock. The ptp4l daemon in linuxptp is specifically designed to synchronize the PHC on the network interface card (NIC) to achieve precise time synchronization using the PTP. However, the NIC needs to offer hardware PTP capabilities for this synchronisation to function.

Hardware PTP support is essential because it allows the NIC to timestamp PTP packets at the hardware level, which significantly improves accuracy. The ptp4l daemon can use this feature to synchronise the PHC directly when the NIC supports hardware PTP, resulting in more accurate and dependable time synchronisation in the PTP network. The ptp4l can still operate in software mode if the NIC does not have hardware PTP functionality, however the synchronisation might not be as precise as with hardware-based timestamping.

Time synchronization is a crucial aspect of Time-Sensitive Networking (TSN), ensuring precise coordination among devices [[5].](#rf5) IEEE 802.1AS defines this synchronization and commonly known as the Generalized Precision Time Protocol (gPTP). gPTP is a modified version of the Precision Time Protocol (PTP) derived from IEEE 1588, designed primarily for time-sensitive applications. gPTP optimizes PTP for these specific demands by simplifying and restricting it.

The Grand Master (GM) role is one important feature added by gPTP. Any node in a gPTP domain can act as the GM. Which node becomes the GM is determined by the Best Master Clock Algorithm (BMCA). The BMCA is an algorithm that can assist you in determining which clock to utilize as the source of timing on your network. The GM is in charge of supplying clocking information to all other nodes in the gPTP domain, ensuring network synchronization.

The ptp4l synchronizes the PTP hardware clock from the NIC. It ensures network time stamps accuracy by aligning the PHC with the PTP master clock. Another Linux PTP daemon, phc2sys, synchronizes the PHC with the System clock. It ensures that the system time is consistent with the hardware clock that is PTP synchronized.

# 3.How gPTP Synchronization Works

The PTP's server and client modes are like the synchronization messages defined by the protocol that are sent between a master and slave clock. Time is given by the master, and the slave synchronizes with the master. GMs are masters who have their clocks synced to a time source like GPS. GM is the master clock that represents as the primary time source for all other clocks on the network. The best master clock (BMC) technique enables many masters to agree upon the optimal clock for the network in addition to the messages. In our experiment, we deliberately configured the system to only have one master PC, leaving the functions of the other PCs unchanged. By doing this, we are hardcoding the grandmaster role to the master PC, which is a particular clock in the network. This strategy may be appropriate in some circumstances where it is known that a particular clock has the most precise and reliable time source.

A minimum of one master and one slave are needed for clock synchronization on the LAN. A single master can synchronize with several slaves, in this work we initially tested synchronization with two slaves and increased the number of slaves to 8 for sound zone project requirements. Slave clocks modify their local clocks using synchronization messages from the master clock. The master and slave clocks record exact timestamps and the packets are transferred between slave and master in PTP as depicted in the [figure 1](#f1). The network latency needed to synchronize the slave to the master is calculated using these timestamps. Between the slave clock and the master clock, four timestamps are recorded. The timestamps are necessary to calculate the slave offset. The common names for the timestamps are T1, T2, T3, and T4.

A diagram of a program

Description automatically generated

Figure 1. Packets transfer between slave and master in PTP

It is necessary to compute the delay paths from master to slave and slave to master. First, distinguish between a master and a slave as: The initial timestamp is T1. It is the precise time when the sync message was sent by the master. This timestamp is given in the following message because the time was sampled by T1 when the sync message was transmitted over the ethernet port. The second timestamp is T2. It specifies the precise moment when the slave got the sync message. The difference between master and slave can be computed as soon as T1 and T2 are accessible at the slave: T2 - T1 = difference between master and slave.

Determine the distinction between slave and master: The third timestamp is T3. This is the actual moment of the data delay request from the slave. The fourth timestamp is T4. This is the precise time that the master receives the delay request message. The difference between slave and master can be detected as soon as T3 and T4 are available at the slave. T4 - T3 = difference between slave and master. Once the difference between master and slave is accessible at the slave, the one-way delay can be calculated as follows: The one-way delay is equal to the sum of the master and slave disparities. The master-slave difference, one-way delay, or ((T2 - T1) - (T4 - T3)) / 2 is the offset, which is used to correct the slave clock.

# 4. Utilizing PTP for Clock Synchronization in Networked Devices

The clocks that are synchronized as part of the PTP synchronization are the clocks of the networked devices that are part of the PTP network. Devices often use internal clocks for a variety of tasks, such as timestamping network packets, monitoring system performance, and running programs that require accurate timing. When we address utilizing PTP to synchronize clocks, we imply synchronizing the internal clocks of networked devices with a common reference time source, such as GPS or an atomic clock. Throughout the synchronization process, the local clocks of each device are modified to remove any mismatch and put them in line with the reference clock source. The purpose of this synchronization is to ensure that every PTP network device operates with an accurate sense of time, which is critical for a variety of application like audio streaming. By synchronizing their clocks using PTP, devices can interact more successfully and carry out operations with more accuracy and precision.

It is possible to synchronize the system clock with a PTP grandmaster clock using the well-known Linux PTP daemon ptp4l. The general procedure for using ptp4l to synchronize the system clock is as follows: At first installing ptp4l on a Linux system requires using the package manager that is appropriate for distribution. Then creating a configuration file for ptp4l that contains the IP address of the PTP grandmaster clock to configure ptp4l. The configuration file is found in the directory AAU\_Audio\_Sync/src/ptp/config\_files. Next step is lauching ptp4l by running the proper command for the distribution to launch ptp4l. For instance, the following command can be used to launch ptp4l with the configuration file:

sudo ptp4l -f /etc/ptp4l.conf -i <interface-name> by replacing <interface-name> with the name of the network interface you want to use for PTP synchronization.

Lastly, monitoring clock synchronization which is the clock synchronization by using the ptp4l -m command, which displays the offset between the system clock and the PTP grandmaster clock. The offset should be very small if synchronization is successful. The ptp4l -m command shows the offset among the system clock and the PTP grandmaster clock, allowing you to keep track of clock synchronization. In the event of successful synchronization, the offset ought to be very close to 0.

## 4.1. Clock synchronization importance

Because it ensures that all clients play back, clock synchronization is crucial for a multichannel streaming system. This is particularly crucial when audio signals are being utilized for current application. The audio signals will not be in sync if the clocks on the clients are not synchronized, resulting in distorted and challenging to understand sounds. Clock synchronization is also important in audio signal transfer for clients to properly sequence the audio packets they receive.

## 4.2. Network Requirements to Achieve Sub-100 Nanosecond Synchronization

Obtaining sub-100 nanosecond timing on a local area network requires a setup that is fully IEEE-1588 compliant. The three main components are: a GPS Grandmaster clock, an Ethernet switch and a PTP slave. All components must support hardware timestamping. The master and slaves are implemented with PTP supported for synchronization in this work. As future work, we could have switch with PTP support as well to further reduce the latency.

## 4.3. Ethernet Switches

Standard ethernet switches and IEEE-1588 enabled ethernet switches are two different types of ethernet switches. Before delivering packets, a typical ethernet switch briefly caches them. The packet's storing time is non-deterministic and reliant on network demand, which causes fluctuation in packet delay. Even when the master and slave clocks enable hardware timestamping, the main cause of conventional ethernet switches' poor time synchronization is the packet delay variance [[6].](#rf6) Improved synchronization between the master and slave can be achieved by using a transparent clock or boundary clock, which also protects the master and slave from the effects of packet delay variation. When it comes to time, high-speed low-latency switches are categorized as standard switches. Under light network loads, high-speed low latency store and forward switches can generate very accurate and consistent synchronization. As a future development the example PTP supported router [[7]](#rf7) may be used for lower the clock timing.

# 5. System Requirement: Network Connection

There are several ways to check if two machines are connected to the same network, first is to check the IP addresses: You can use the command "ifconfig" on Linux to check the IP address of each machine. If the first three octets of the IP address are the same for all machines, then they are likely on the same network which shown on the [figure 2](#f2).

Or we can use the "ping" command to check if one machine can reach the other. For example, on the sender machine, you can type "ping <receiver-ip>" and see if you receive a response. If you do, then the machines are connected to the same network.

A screenshot of a computer

Description automatically generated

Figure 2. IP address of the PC that attached with new NIC card

# 6. Exploring PTP Clocks in Network Environments

A PTP clock is a hardware or software system that can synchronize its time with other clocks on a network utilizing the PTP. Depending on the device, a PTP clock may be software running on a computer or other hardware. A system clock is a clock built into a computer or other electronic device that is used to keep track of the system's time. The system clock is used to time-stamp events and is used by the operating system and applications to schedule tasks. PTP Hardware Clock is a clock with hardware that supports the PTP protocol is called a PTP hardware clock. It is used in a PTP network to keep accurate time by synchronizing its clock with the master clock. A master clock is a clock that has been recognized as the main time source of the network. All other clocks in the network must receive accurate time information from it. The master clock in a PTP network is usually the clock with the highest accuracy and the most reliable time source [[8].](#rf8)

# 7. System Setup: Implementation of PTP Synchronization with LinuxPTP and Hardware Components

One of the primary functions of Time Sensitive Network (TSN) is time synchronization, and Linux's linuxptp software implements the PTP in accordance with IEEE standard 1588 will be discuss and implement in this work. The PTP Hardware Clock (PHC) of Linux PTP can be synchronized with the network interface card (NIC) using the ptp4l daemon, which is provided as a tool for time synchronization. The main objective of this configuration is to synchronize the computers' internal clocks accurately. Boundary Clock (BC) and Ordinary Clock (OC) are implemented by the ptp4l. To assemble a networking card, the essential work requires a PCIe slot that can accommodate a NIC card. In this experiment each PC that was attached to the Asus Router had an Intel Ethernet Controller I210-T1 NIC card installed to verify time synchronization. Each PC is fitted with a NIC, which is essential to the synchronization process. The Linux PTP software's PHC may synchronize with the system's network interface thanks to the NIC.

Since we initially used laptops, we had to use NIC cards that were not as accurate as those we had with desktops. All 8 computers are running the same version of Ubuntu, 22.04 [[9],](#rf9) and are DELL OptiPlex 9020 branded. For linux command during the work some tutorial has been taken [[10].](#rf10) The one PC is used as master and slave and had 8 clients in total attached the asus router [[11].](#rf11) The Dell OptiPlex 9020 desktop PCs are equipped with two PCI Express x16 slots, each offering a speed of 16GB/s. These slots provide support for installing additional hardware components, such as the Asus card and the I210 NIC [[12].](#rf12) In this instance, the router, an ASUS RT-AX88U gaming router, serves as the network's central hub. It makes it easier for the client PCs and the master PC to communicate and sync. With the availability of these expansion slots, users have the flexibility to enhance their system's capabilities by adding compatible devices. An IP address is given to each PC on the network to facilitate communication. PTP synchronization messages and precise time information can be exchanged between the master PC and the clients thanks to the NIC cards and the router's network capabilities. We have built up a system that provides exact time synchronization between the eight client PCs and the master PC by configuring this network configuration and implementing PTP synchronization with the designated hardware and software components as seen in the [figure 3.](#f3)

A group of computer monitors on a desk

Description automatically generated

Figure 3. Network Setup for PTP synchronization

The PTP is based on a straightforward master slave synchronization principle as indicated in [figure 1](#f1). There are two phases in the normal execution of PTP. The first step for clock synchronization in the PTP system is establishing the master slave synchronization hierarchy, check the time synchronization support on the relevant port, use tool in the linuxptp package (ptp4l) that is an implementation of the PTP, then starting ptp4l [[13](#rf13)] as a service and to use ptp4l with hardware time stamping capable drivers and NICs in addition must provide the network interface, new messages with offsets will be printed periodically as seen in the diagram of PTP Sychronization in the [figure 4](#f4). All computers must support the hardware that PTP protocol. We connected an Intel Ethernet Controller I210-T1 NIC card to each desktop in order to support this claim.

A diagram of a computer process

Description automatically generated

Figure 4. PTP Sychronization Block Diagram

To maintain perfect synchronization, the network must have a PTP master clock that consistently communicates time information to the PTP hardware clocks on the PCs. All machines must have the ptp4l application installed and set to use the same PTP master clock. The LinuxPTP is software implementation of the PTP, also known as IEEE 1588. PTP is a protocol used to audio stream systems to synchronize clocks across a network in our work. In a PTP network, the master clock is responsible for providing correct times to all others clocks on the network and serves as the network's authoritative time source. The master clock is typically chosen for its high precision and dependability as a time source. The master clock PTP provides a time signal and sends it to the network's clocks, allowing them to rapidly adjust to the master clock's time.

Depending on the size and complexity of the PTP network, one or more PTP master clocks may be present. At first we started to build the setup with 2 slave PC and 1 master, as next target the slave numbers will be extened to the 8 slaves for our streaming audio sound technology project. A system for choosing the best master clock from a pool of candidates is included in the PTP protocol. This mechanism takes into account network topology, accuracy, and stability. Once a master clock has been determined, it distributes time information to all other clocks in the network, enabling synchronization to a single point in time.

# 8. Overview of the synchronized multi-PC system

The synchronized multiple PC setup consists of two types of hardware: The audio is transmitted by the source device and received by the sink device. The source device is in charge of generated audio transfer across the network to each connected sink device. The sink device's responsibilities include receiving audio and playing it back in sync with the other sink devices. In the next chapter, the audio streaming part will be discussed however in this section the ptp synchronization will be taken into account as indicated earlier.

The essential principle of the synchronized multiple PC system is illustrated in the diagram in the [figure 5,](#f5) with a single source device and several receiver devices. The source device captures audio from an audio program and distributes it to the sink devices via the network. The sink devices next carry out the synchronized replay of the audio given by the source device.

Diagram

Description automatically generated

Figure 5. Basic Idea of the synchronous multi PC setup

When low-latency synchronization is required for audio streaming, PTP clock synchronization is advantageous. PTP allows extremely exact clock synchronization across a network's numerous systems or devices. Accurate and coordinated synchronization between audio samples and the system clock is crucial when streaming audio with low latency. We can obtain better tuning and lower clock drift by utilizing PTP to synchronize the PHC with the system clock. Timing mismatches that may occur due to network latency, different packet delays, or clock drift can be reduced by synchronizing with PTP. By precisely syncing the PHC clock with the system clock, we can ensure that the audio samples are played at the appropriate times and that the other pipeline elements maintain their synchronization.

# 9. LinuxPTP: High-Precision Clock Synchronization for Linux Systems

The PTP protocol is implemented using LinuxPTP, which is part of the Linux kernel. It is intended to offer Linux systems high precision clock synchronization, and it enables timestamping in both hardware and software. There are many applications frequently use LinuxPTP. In this project we focused for audio streaming.

Different cards will result in different crystal level clock rates, but this is not a problem because ptp's Berkeley algortihm will account for it. When sync packages are sent, the idea of hardware timestamping is to send them precisely. At the user space, kernel, and firmware layers, offset and skew are calculated using the same clock.

We must verify support for the following as seen in the [figure 6](#f6).

• The ptp version (it must match with all PCs)

• Support for the Linux ptp library we are using

In order to determine the PTP version of a network interface under Linux, we need to use the ptp4l command line program to do so simply use the next command to determine the PTP version of a certain interface: sudo ptp4l -v -i <interface-name>. The name of the interface on the network you want to verify should be used in place of interface-name=enp2s0>. Information on the PTP version that the interface supports should be included in the command's output.

A screen shot of a computer code

Description automatically generated

Figure 6. PTP version and Linux PTP library

The ldd command lists the shared libraries that are linked to an executable file, allowing you to see which PTP library is being used on a Linux system. For instance, you can use the command below to determine which PTP library the ptp4l command-line tool is utilizing: ldd $(which ptp4l). This following command is utilized to locate the ptp4l executable file on the system. The ldd command then lists the shared libraries that are linked to the ptp4l executable, including any PTP libraries.

The use of the ptp4l tool enables the synchronization of clocks on the two desktop computers to achieve a remarkable level of accuracy, often within a few nanoseconds, assuming that certain requirements are met. The precise degree of synchronization depends on the accuracy and stability of the hardware clocks as well as the network conditions.

# 10. Explanation of code of setup in lab

## 10.1. Installing Linux PTP

To install linuxptp execute the following line; sudo ./src/install\_scripts/linux\_ptp.sh linux\_ptp.sh includes installation command line of linuxptp such as following line; sudo apt install linuxptp

This command will run the installation script called "linux\_ptp.sh" located in the "install\_scripts" directory. The script is specifically designed to facilitate the installation process of linuxptp. The use of the "sudo" command ensures that the installation is carried out with administrative privileges, allowing necessary system-level changes to be made.

## 10.2. PTP Setup

To run PTP, we first need to start the master device using in [[14](#rf14)] github file; sudo ./ptp4l\_master\_launch.sh. The master configuration file for linuxptp, named "ptp4l\_master.cfg" contains specific settings that need to be configured as follows: enabling *gmCapable* with a value of 1 allows the local clock to function as a grandmaster. As a grandmaster, the clock becomes the authoritative source of time for the PTP network, selecting *hardware* with a value of 1 indicates the usage of a hardware based PTP network. Hardware-based PTP typically provides higher accuracy and precision compared to software-based solutions, as it relies on specialized hardware features for timestamping and synchronization, *verbose 1* allows us to print messages to the standard output and this can be helpful for monitoring and troubleshooting purposes, *hybrid\_e2e 1* enables the hybrid delay mechanism which means ports in the slave state send their delay request messages to the unicast address taken from the master's announce message; ports in the master state will reply to unicast delay requests using unicast delay responses, *inhibit\_multicast\_service 1* some unicast mode profiles insist that no multicast message are ever transmitted, *logSyncInterval 1* is the time interval between sync message, *assume\_two\_step 1* treats one-step responses as two-step, *twoStepFlag 1* enables two-step mode for sync messages, in two-step mode, the master clock sends the sync message in two separate steps, allowing more accurate synchronization between the master and slave clocks, *delay\_mechanism 1* selects the default delay mechanism is E2E that leads all clocks on single PTP communication path use the same mechanism, *tx\_timestamp\_timeout* 1 is the number of milliseconds to poll waiting for the tx time stamp from the kernel when a message has recently been sent, *BMCA noop* means that the traditional BMCA algorithm used by 1588 is skipped this means that linuxptp does not perform the automatic selection of the best master clock based on priority and quality metrics, masterOnly and slaveOnly will be used to determine the master or slave role for the device, *masterOnly 1* prevents the port from entering the SLAVE state, *inhibit\_announce 1* disables the timer for announce messages and also the announce message timeout timer this can be useful in certain scenarios where announce messages are not needed or where the timer is not required, *inhibit\_delay\_req 1* does not send any delay requests since we set the *asCapable true* config option to be set to true earlier.

Before running the slaves, first we need to be sure the UDPv4 IP address sets that should match the IP of the master device in following configuration file; /src/ptp/config\_files/ptp4l\_slave.cfg

In this testbed master device IP is 192.168.50.20 therefore in the above file we made the changes accordingly. Next step is to start the slave device using following line; sudo ./ptp4l\_slave\_launch.sh this configuration file has slave configuration file called ptp4l\_slave.cfg has some specifications that has to be setted as following; *unicast\_req\_duration 60* is service time in seconds to be requested during unicast discovery (we request service for sixty seconds), *slaveOnly 1* is enabled for the local clock is a slave only clock, *unicast\_master\_table 1* (set to a positive integer 1) that specifies the table ID to be used for unicast discovery (in our case table has one possible UDPv4 master clock), *logQueryInterval 2* option configures the time to wait between unicast negotiation attempts, lastly UDPv4 transport type and network address of a potential remote master has been indicated. It guarantees the PHC from the NIC is in sync with the GM clock from the gPTP domain. This is achieved by the ptp4l daemon. After running sudo ./ptp4l\_slave\_launch.sh it synchronizes PHC with the GM clock as seen in the [figure 7](#f7).

A picture containing table

Description automatically generated

Figure 7. Ptp Sychronization Output master/slave between 2 PCs

When the port state changes from UNCALIBRATED to SLAVE, the computer has successfully synchronized with a PTP master clock. The master offset value represents the measured offset from the master clock, indicating the time difference between the local clock and the master clock in nanoseconds. When the network is routed through a basic switch which doesn’t support PTP hardware timestamping we achieve in range sub 100ns accuracy for devices with PTP hardware build into their network card as seen in the figure 2.

The clock's frequency adjustment is indicated by the freq value, which is expressed in parts per billion (ppb). It means that the clock's rate has been adjusted to match the reference frequency of the master clock. The path delay value, which is represented in nanoseconds, provides an estimate of the time it takes synchronization messages to move from the master to the slave. This value takes into account the network propagation delay, which has an impact on the precision of overall synchronization.

In the [figure 8](#f8), ptp4l consistently reports the master offset values as sub-100ns, it means that the time offsets between the PHC (PTP Hardware Clock) and the master clock are consistently very low, within a range of sub-100ns. This is an excellent indication that the PHC is closely synchronized with the master clock. A master offset value of less than 100 nanoseconds indicates that the master device is keeping especially accurate time and that the PTP network as a whole uses its clock as the reference time. The slave devices are closely following the master's clock, ensuring that they are all operating in sync with each other.

A blue rectangular graph with white text

Description automatically generatedA graph of a distribution of offset values

Description automatically generatedA blue rectangular graph with white text

Description automatically generatedA blue rectangular graph with white text

Description automatically generatedA blue rectangular graph with white text

Description automatically generatedA blue rectangular bar graph

Description automatically generatedA blue rectangular object with numbers

Description automatically generated

Figure 8 Distribution of Master offset on 8 PC separately

Like ptp4l, phc2sys reports the time offset between PHC and System Clock, which determines if the clocks are synchronized. The phc2sys utility is launched with superuser privileges by using the command "sudo./phc2sys\_launch.sh". When this command is executed, phc2sys extracts the time difference between the PHC and the System Clock and outputs it. The level of synchronization between the two clocks can be determined using the time offset information provided by phc2sys as seen in the [figure 9](#f9).

A screen shot of a computer

Description automatically generated

Figure 9. Time offset between the PHC and the system clock: system clock sync

It's normal to see different timestamps in different computers running phc2sys, as the timestamp corresponds to the local time of each machine when the message was logged. The provided message (phc2sys[3302.137]: CLOCK\_REALTIME phc offset 50 s2 freq 4711 delay 2159) includes a timestamp in the format [3302.137], which represents the number of seconds and fractional seconds since an arbitrary point in time (usually the Unix epoch). If you see a similar message on another machine, the timestamp should be different, as the local time on that machine will be different. However, despite the differences in timestamps, the message content should be similar if the machines are using the same PTP hardware clock and are configured in a similar way.

The phc2sys program is used to synchronize the system clock of both the slave and master devices to the PTP hardware clock (PHC) on the network interface card (NIC). When running phc2sys on both the master and slave devices, the system clocks of these devices are adjusted and tightly synchronized with the PHC.

The system clock of a slave device is synced with the PTP clock supplied by the master using the phc2sys command. Therefore, on each of the 8 slave PCs, we would execute the phc2sys command with the proper configuration file and command-line options.

On each slave desktop, we would use the "-m" option to tell phc2sys to operate in "slave" mode, and we would specify the appropriate network interface ("-s $INTERFACE") and configuration file ("-f config\_files/phc2sys.cfg") as needed. The "-c CLOCK\_REALTIME" option is used to specify that the system clock should be synchronized to the Linux system's CLOCK\_REALTIME clock.

The step\_threshold option is set to 1 in the preceding example. This indicates that the system clock will be changed in a single step to match the PTP clock if the time difference between the two clocks is greater than one second. Let's use the scenario where the system clock is 2 seconds slower than the PTP clock. When step\_threshold is set to 1, the system clock will be advanced by one step, or two seconds, to coincide with the PTP clock. Different transport systems employ various transportSpecific values. For instance, the value 1 for transportSpecific is used by the IEEE 802.3 ethernet transport mechanism.

The command sudo phc2sys -s eth0 -c CLOCK\_REALTIME --step\_threshold=1 --transportSpecific=1 -w -m is used to synchronize the system clock to the PTP (Precision Time Protocol) clock and continuously monitor the synchronization status. If there is a setup with one PTP grandmaster and 8 PTP slaves, we should run this command on each of the 8 PTP slave devices, but not on the PTP grandmaster device. On the grandmaster device, we only need to configure the PTP clock and start the PTP daemon. The phc2sys command is used on the slave devices to synchronize their system clocks to the PTP grandmaster clock.

Make sure to replace eth0 with the actual interface name of the PTP clock on each of the slave devices before running the command. Additionally, the --step\_threshold and --transportSpecific options can be adjusted according to your specific requirements.

In the [figure 9](#f9) at first we will see the 2 clients [[15]](#rf15) PHC offset variation across 1000 sample. It's noteworthy to note that the PHC synchronization has persisted throughout the 1000 timestamps, and the PHC offset has remained constant at 200 ns or less. This suggests that the timing alignment of the synchronization process has been generally steady and consistent over time. This shows a partially consistent offset behavior within this particular range.

A graph of a sound wave

Description automatically generated

Figure 10. PHC Offset Variation Across Samples

The [figure 10](#f10). is conducted with 8 clients [[16]](#rf16) synchronization of clock PHC offset and as seen in the figure there are fluctuations in the master offset values that could be due to factor affecting the time synchronization process. The potential cause for the master offset's quick decrease from a high value to a low value is the master offset may be significant when time synchronization first begins since the slave and master clocks are not yet in phase. The offset gradually decreases as synchronization advances until the clocks are properly synchronized. Network latency and delay variations can impact the accuracy of time synchronization as well. If there are fluctuations in network conditions or delays in transmitting synchronization messages, it can affect the offset values reported by phc2sys. Based on the [figure 10](#f10) we could say that PHC offset clock synchronization is around 250ns with 8 client PCs.

A colorful line graph with numbers

Description automatically generated

Figure 11. PHC Offset Variation Across Samples with 8 client PCs

The statistical spread or variance of the offset values generated from the PTP synchronization procedure of various clients with a master clock is represented by the distribution of PHC offset values in the [figure 11](#f11).

A graph of distribution of phc offset values

Description automatically generated

Figure 12. Distribution of PHC offset

Multiple clients synchronize their clocks with a master clock in a PTP synchronization system to achieve time alignment. Each client measures the variation between its own clock and the master clock throughout this synchronization procedure. The offset is the name given to this difference. The offset values can change as a result of a number of variables, including synchronization issues, clock drift, and network latency. The spread or distribution of these discrepancies among the clients is revealed by the distribution of PHC offset values as we discussed earlier.

We can observe how frequently particular offset ranges occur by plotting a histogram of the offset values from [figure 12](#f12). The histogram depicts the occurrence frequency (y-axis) for various offset ranges (x-axis). This enables you to figure out the offset distribution. Analyzing the distribution of PHC offset values can reveal information about the quality of synchronization between clients. Better synchronization precision and consistency are indicated by a tighter distribution with values grouped around zero. A larger distribution with values spread out, on the other hand, indicates increased variability or potential synchronization concerns. We can find differences in synchronization performance amongst clients by comparing their distributions. Differences in distribution spread, peaks, or skewness might show variances in synchronization quality or potential concerns particular to specific clients. To sum up, evaluating the distribution of PHC offset values aids in understanding the behavior and performance of the PTP synchronization system, revealing important information about the precision and consistency of time synchronization among clients. The [figure 13](#f13) shows for each client individually PHC offset distribution to with fixed interval. The time offset between the PTP Hardware Clock (PHC) and the System Clock for each individual client PC is represented by the values shown from the phc2sys log files.

A graph of a number of blue bars

Description automatically generatedA graph of a distribution of values

Description automatically generatedA graph of a number of blue bars

Description automatically generatedA graph of a distribution of values

Description automatically generatedA graph with numbers and a number of blue bars

Description automatically generatedA graph of a distribution of value

Description automatically generatedA graph of a number of blue bars

Description automatically generatedA graph of a distribution of values

Description automatically generated

Figure Distribution of PHC offset on 8 PC separately

# Bibliography

[1] <https://www.usenix.org/conference/atc21/presentation/chen>

[2] <https://www.mobatime.com/article/ntp-and-ptp-informations/>

[3]<https://docs.fedoraproject.org/en-US/fedora/latest/system-administrators-guide/servers/Configuring_PTP_Using_ptp4l/>

[4]<https://events.static.linuxfound.org/sites/events/files/slides/lcjp14_ichikawa_0.pdf?fbclid=IwAR0MTZA8fqWRyHzM9Su4TP5fmzN8lu4jAbO2LBEQUhILLRuuZNDlPGamCTI>

[5] <https://semiengineering.com/tsn-ptp-a-real-time-network-clock-synchronizing-protocol/>

[6] <https://timemachinescorp.com/2020/04/09/3-differences-between-ptp-hardware-and-software/>

[7] <https://www.perle.com/products/switches/ids-509-industrial-managed-ethernet-switch.shtml>

[8] <https://endruntechnologies.com/pdf/PTP-1588.pdf>

[9] <https://www.makeuseof.com/install-desktop-environment-gui-ubuntu-server/>

[10] <https://www.guru99.com/must-know-linux-commands.html>

[11] <https://www.asus.com/networking-iot-servers/wifi-routers/asus-gaming-routers/rt-ax88u/>

[12] <https://media.dustin.eu/media/d200001001284115/intel-i210-t1-product-data-sheet-brochure.pdf>

[13]<https://linuxptp.sourceforge.net/>

[14]<https://github.com/mfatihkoc/AAU/tree/main/AAU_Audio_Sync/src/ptp>

[15] <https://github.com/mfatihkoc/AAU/tree/main/ptp_outputs>

[16] <https://github.com/mfatihkoc/AAU/tree/main/ptp_outputs/test1>