Technical Guide

Precision Time Protocol and Transparent Clock

Feature Overview and Configuration Guide

Introduction

Precision Time Protocol (PTP) is an Ethernet or IP-based protocol for synchronizing time clocks on a collection of network devices using a timeTransmitter/timeReceiver distribution mechanism.

Both Network Time Protocol (NTP) and PTP protocols do the same thing and carry the same information. However, they do differ in how they operate, and they have different applications, with NTP providing general purpose time distribution, and PTP being used for high precision time distribution.

The **Transparent Clock** feature is an IEEE 1588 capability that is used by bridges or routers that assists other 1588 clocks in measuring and adjusting for packet delay. The Transparent Clock computes the variable delay as the PTP packets pass through the switch or the router.

Transparent Clock is available on selected AlliedWare Plus[1] platforms and is compliant with the IEEE 1588-2008 (version 2) standard, (referred to as IEEE 1588 throughout this document).

This guide describes how AlliedWare Plus uses PTP and the Transparent Clock in an IEEE 1588 network.

For details on AlliedWare Plus limitations, see:

- "AlliedWare Plus limitations" on page 19

1.When using End-to-End delay mechanism

Contents

Products and software version that apply to this guide

This guide applies to all AlliedWare Plus products, that support IEEE 1588 Precision Time Protocol, running version **5.4.7** or later.

PTP in a VCStack environment is supported from version:

- 5.5.3-2.1 onwards on SBx908 GEN2, x950, and x550 Series switches.
- 5.5.4-1.2 onwards on x930 and x530 Series switches
- 5.5.5-2.1 onwards on SBx908 GEN3 Series switches

Note: 100G interfaces are not supported for stacking with PTP. Stack links need to be 40G rather than 100G if PTP is required on an x950 or SBx908 GEN2 stack.

PTP VCStack and Link Aggregation is supported from version 5.5.4-1.2 onwards on:

- x530/x530L Series switches
- x930 Series switches
- x950 Series switches
- SBx908 GEN2 Series switches

PTP VCStack and Link Aggregation is supported from version 5.5.5-1.1 onwards on:

- SBx908 GEN3 Series switches

PTP over UDP (Layer 4) is supported from version 5.5.5-1.1 onwards on:

- x530/x530L Series switches.

For more detail on PTP over UDP, see "PTP over UDP for the x530 Series" on page 26

Latest product information

Feature support may change in later software versions. For the latest information, see the following documents:

- The product's Datasheet
- The AlliedWare Plus Datasheet
- The product's Command Reference

These documents are available from the above links on our website at alliedtelesis.com.

Terminology

This document uses the optional alternative terms defined in IEEE Std 1588g-2022:

- **timeTransmitter** instead of master
- **timeReceiver** instead of slave

These terms are specified in:
Institute of Electrical and Electronics Engineers, "IEEE Std 1588g-2022, IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems – Amendment 2: Master-Slave Optional Alternative Terminology," December 2022, www.ieee.org.

Overview of the PTP feature

PTP is a network-based time synchronization standard, but instead of millisecond-level synchronization, PTP networks aim to achieve nanosecond or even picosecond-level synchronization.

PTP time stamping is so accurate because it uses hardware time stamping instead of software, and PTP equipment is dedicated to one specialized purpose: keeping devices synchronized. For that reason alone, PTP networks have much sharper time resolutions, and unlike NTP, PTP devices will actually time stamp the amount of time that synchronization messages spend in each device, which accounts for device latency.

In this overview we describe the type of applications where precise Ethernet or Ethernet/IP timing is used, the wall clock concept, synchronization and clock types, delay mechanisms, and time stamping modes.

Applications

There are several applications that require precise timing using Ethernet or Ethernet/IP:

- Telecom - applications, such as cellular, where not only frequency, but also phase precision is needed in order to control hand-off of mobile phones from one cell tower to the next.
- Factory Automation - legacy field buses are being replaced with Ethernet-based field buses and timing is needed for drive controllers and distributed I/O. For example, robots working in conjunction with automation conveyor belts need to be precisely timed.
- Power Substations - legacy process buses in power substations are being replaced with Ethernet-based buses. These buses have several components that need precision time for handling power transfers and recording fault measurements.

Understanding time and synchronization

What is time?

IEEE 1588 uses the **Wall Clock** concept, that is it uses the Time of Day (ToD), not only hours, minutes, and seconds, but also the number of nanoseconds within the second. More precisely, PTP carries time that consists of 48 bits worth of seconds, and 32 bits worth of nanoseconds. The 48 bit seconds is actually the number of seconds that have elapsed since January 1, 1970 at midnight, which is known as the PTP epoch.

Synchronization types

Applications can use IEEE 1588 to achieve the following types of synchronization:

Frequency The nodes in the network have the 48 bits and in particular the 32 bits of time change at the same **rate**, without caring necessarily what the 48/32 bit values are. Telecom applications initially used IEEE 1588 to distribute frequency only.

Phase The nodes in the network not only have the 48/32 bits of time changing at the same rate, but have at least the seconds boundary time occurring at the same **time**. That is to say, when the nanoseconds

time rolls over and increments the seconds time, all nodes do this at the same time. These nodes may not necessarily need to know what the year, month, day, and hour are.

Time of Day The nodes in the network are not only frequency and phase synchronized, but they also want to know what the year, month, day, hour, and seconds are, along with nanoseconds.

IEEE 1588 core concepts

Roles

IEEE 1588 defines two main roles for distributing time across a network:

- **timeTransmitter**
  The entity that distributes time to other devices. A timeTransmitter can also act as a Grand Master (GM), which obtains its time from a primary reference source, typically GPS.
- **timeReceiver**
  The device that synchronizes its clock to the remote timeTransmitter.

Clock types

There are different kinds of clocks defined in IEEE 1588:

- **Ordinary Clock** - only has one clock port. For an Ordinary Clock, a clock port is abstract and may not necessarily correspond to a physical port. The following can be an Ordinary Clock:
  - timeTransmitter - this distributes IEEE 1588 time to timeReceivers
  - timeReceiver - this synchronizes IEEE 1588 time with the timeTransmitters.
- **Boundary Clock** (BC) - has multiple clock ports, abstract or not. Unlike the Grand Master, which communicates individually with each timeReceiver, a Boundary Clock helps scale by combining a timeReceiver half and a timeTransmitter half. The timeReceiver half synchronizes using one clock port, while the timeTransmitter half uses other clock port(s) to distribute the clock to its set of timeReceivers
- **Transparent Clock** (TC) - assists in the delay measurement between a timeTransmitter and a timeReceiver by including a Correction Factor (CF) that tells the timeReceiver how much delay the Transparent Clock added. This in general applies to Layer 2 bridges, and a Transparent Clock's clock ports correspond to physical interfaces.

Delay mechanisms

As detailed in the next section, in order for PTP nodes to achieve phase or Time of Day synchronization, timeReceivers have to determine the delay between themselves and the timeTransmitter.

IEEE 1588 defines two different delay mechanisms that can be used to determine the delay between a timeTransmitter and a timeReceiver, they are End-to-End (E2E) and Peer-to-Peer (P2P), but only one can be used at a time.

- **E2E**

This delay mechanism requires the timeReceiver to measure the delay between itself and the timeTransmitter (thus End-to-End). The timeTransmitter and timeReceiver send IEEE 1588 messages called DELAY REQUEST and DELAY RESPONSE back and forth between the two, allowing the delay to be measured. Once the delay is known, the timeReceiver can adjust its time to be phase and Time-of-Day synchronized with the timeTransmitter.

In the ideal case, the delay between the timeTransmitter and the timeReceiver is constant, such as when using a wire. In practice, there are Layer 2 and/or Layer 3 devices in between that make the delay variable. As will be seen later, the role of a Transparent Clock is to add a correction to certain PTP messages which assists the timeReceiver in removing this variable delay.

- **P2P**

This delay mechanism requires each network element to measure the delay between its input port and the device attached on the other end of the wire of this input port (the peer device). As the timeTransmitter sends its view of time (using SYNC messages) towards timeReceiver(s), each network element along the way receives the SYNC message and adds a correction to the SYNC message. The correction includes the measured wire delay of the input port the SYNC message was received on.

For Transparent Clocks, the correction also includes the delay through the bridge. This correction is cumulative as it traverses nodes hop-by-hop. As the SYNC message finally arrives at a timeReceiver, the accumulative correction in the SYNC message will contain the total delay from the timeTransmitter to the timeReceiver. This eliminates the timeReceiver from having to send messages back and forth with the timeTransmitter. Peer-to-Peer is a newer IEEE 1588 technology, and not all devices deployed today support Peer-to-Peer.

Technically, End-to-End delay mechanism can be used at the same time as Peer-to-Peer delay mechanism as long as the two are not used along the same IEEE 1588 messaging path. The different delay mechanisms are mutually exclusive of each other. That is to say, between a timeTransmitter and timeReceiver inclusive, all the IEEE 1588 participating nodes must use either the End-to-End or Peer-to-Peer delay mechanism but can not use a mix.

Time stamping modes

The key to IEEE 1588's accuracy is the ability to Time Stamp specific IEEE 1588 messages as they enter and as they leave a physical interface. Hardware is most often used to accomplish this and is done as close to the edge of the physical port's hardware as possible. There are two types of time stamping modes used, 1-step and 2-step:

- **1-step**

A Time Stamp is captured in real time as the message starts transmitting out the physical port. The message is edited on the fly to carry the captured time stamp.

- **2-step**

A Time Stamp is captured in real time as the message starts transmitting out the physical port, but the implementation is unable to edit the packet on the fly, and thus can not carry the captured time stamp. A secondary message is used instead to carry the captured time stamp. Early implementations of IEEE 1588v2 used the 2-step time stamping mode, as the earlier hardware was unable to edit the message.

Most modern implementations support the 1-Step time stamping mode.

Understanding PTP and Transparent Clocks

The following sections describe how the Transparent Clock is used in an IEEE 1588 network with the End-to-End delay mechanism. First, here is a brief overview of IEEE 1588v2 messaging.

Messaging

IEEE 1588 supports a variety of messages. Not all of these are of interest to Transparent Clocks but they are listed here for completeness:

Event Messages - Those that require time stamping:

- SYNC - used by the timeTransmitter to convey time. It is used in all scenarios.
- DELAY REQUEST - used between the timeTransmitter and timeReceiver when using End-to-End delay Mechanism to measure delay. The timeReceiver sends this message to the timeTransmitter.
- PEER DELAY REQUEST - used between IEEE 1588 devices to measure the delay of an incoming link. Only used when Peer-to-Peer delay mechanism is used.
- PEER_DELAY_RESPONSE - used between IEEE 1588 devices to measure the delay of an incoming link. Only used when Peer-to-Peer delay mechanism is used.

General Messages - Those that do **not** require time stamping:

- FOLLOW UP - used by the timeTransmitter to convey a captured time stamp of a transmitted SYNC message. This is used in 2-Step mode to send the earlier captured time stamp of a SYNC message.
- DELAY RESPONSE - used between the timeTransmitter and timeReceiver when using End-to-End delay Mechanism to measure delay. The timeTransmitter uses this to respond to the timeReceiver.
- PEER DELAY RESPONSE FOLLOW UP - used between IEEE 1588 devices to measure the delay of an incoming link. Only used when Peer-to-Peer delay mechanism is used with 2-Step mode.
- ANNOUNCE - is sent and received by local clock ports with a variety of information. It can be used to determine which one out of several possible timeTransmitters is to be selected as the Best timeTransmitter. It can also be used between timeTransmitter and timeReceiver to negotiate a unicast service.
- MANAGEMENT - used between management devices and clocks.
- SIGNALLING - used by clocks for conveying things such as how often to send messages, supporting unicast services instead of multicasting etc.

A Transparent Clock is primarily interested only in **Event messages**.

End-to-End delay mechanism

Figure 1 below shows a timeTransmitter Ordinary Clock and a timeReceiver Ordinary Clock that are connected together by a yellow Ethernet cable. In this figure, the timeTransmitter can be considered as the Grand Master (GM) where it has acquired Time of Day (ToD) synchronization from a system such as a Global Positioning System (GPS).

The timeTransmitter as well as the timeReceiver have a running time stamp clock that is available at its physical port. The timeReceiver's time clock however is not yet synchronized with the timeTransmitter, and the following describes how synchronization is achieved:

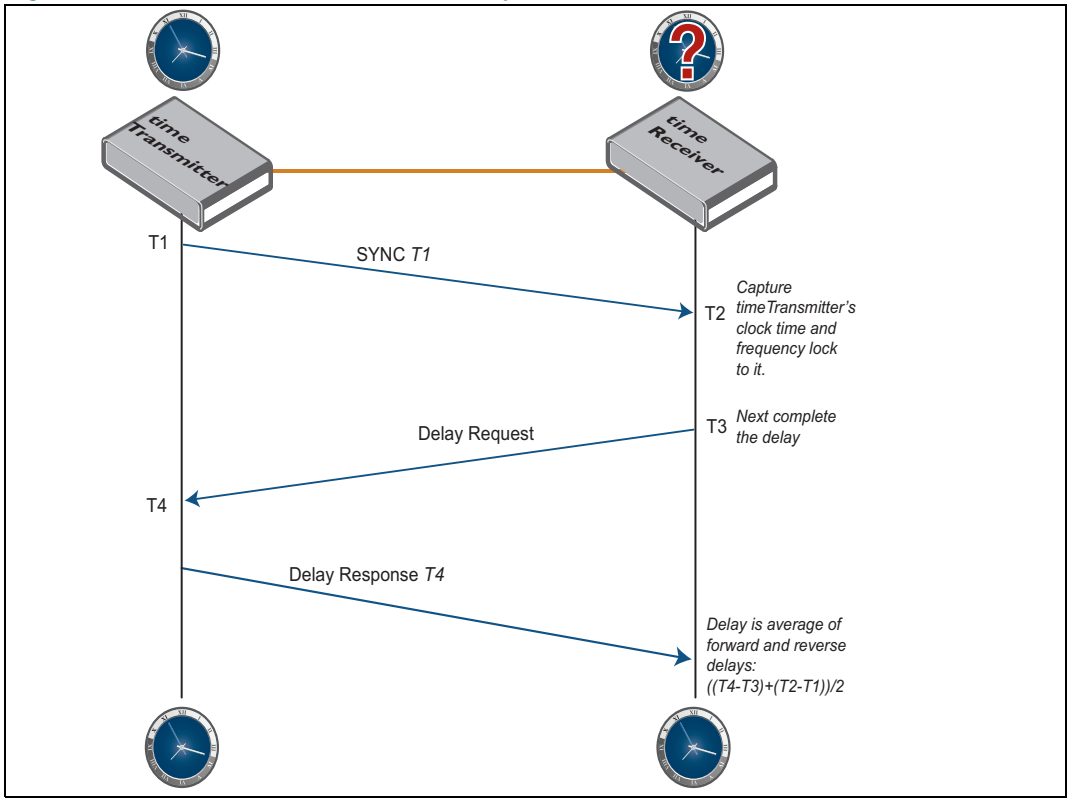
A typical PTP sequence involves a series of four messages between timeTransmitter and timeReceiver:

- The initial sync message from timeTransmitter to timeReceiver
- A delay request message from timeReceiver to timeTransmitter
- A final delay response message from timeTransmitter to timeReceiver

This sequence produces four different time stamps:

- T1 when the timeTransmitter sends the initial sync message
- T2 when the timeReceiver receives the initial sync message
- T3 when the timeReceiver sends the delay request
- T4 when the timeTransmitter receives the delay request

Figure 1: Ideal IEEE 1588 end-to-end delay mechanism



Here is a more detailed look at the PTP sequence shown above:

1. The IEEE 1588 timeTransmitter periodically sends out an IEEE 1588 SYNC message to the IEEE 1588 timeReceiver device. As the SYNC message leaves the timeTransmitter's physical interface, it captures a running time stamp in the timeTransmitter, shown as T1. In the 1-Step mode that is being illustrated here, the timeTransmitter sets the 'Origin Time Stamp' field in the SYNC message to T1 before the message completely exits the interface.

2. The IEEE 1588 timeReceiver receives the SYNC message and its running time stamp clock captures the time (T2) that the SYNC message starts to arrive at its physical port.

   - Although the timeReceiver could set its time stamp clock to that of the timeTransmitter using T2, it would leave the timeReceiver's clock in an inaccurate state due to the propagation delay of the wire. Also, the timeReceiver's time stamp clock will be running slightly faster or slower than the timeTransmitter's. In the beginning stages, although not shown, the next stage of operation is for the timeReceiver to try and frequency lock its running clock with the timeTransmitter's. During this phase, the timeReceiver will only receive SYNC messages until it believes its time stamp clock is changing at the same rate as the timeTransmitter's.

   - After frequency locking, the timeReceiver will next proceed to determine what the delay is between itself and the timeTransmitter.

3. The timeReceiver next calculates what the delay is by sending a DELAY REQUEST message to the timeTransmitter. As the message starts to be transmitted out the timeReceiver's physical interface, the timeReceiver's running time stamp clock is used to capture the time (T3) and the timeReceiver stores this time while it waits for the reply.

4. The timeTransmitter receives the DELAY REQUEST and uses the timeTransmitter's running time stamp clock to capture the time (T4) as the message starts to be received on its physical interface. After retrieving the captured T4 value, the timeTransmitter will shortly thereafter send the timeReceiver a DELAY RESPONSE containing the captured T4 value.

5. The timeReceiver receives the DELAY RESPONSE message and extracts the T4 value in it.

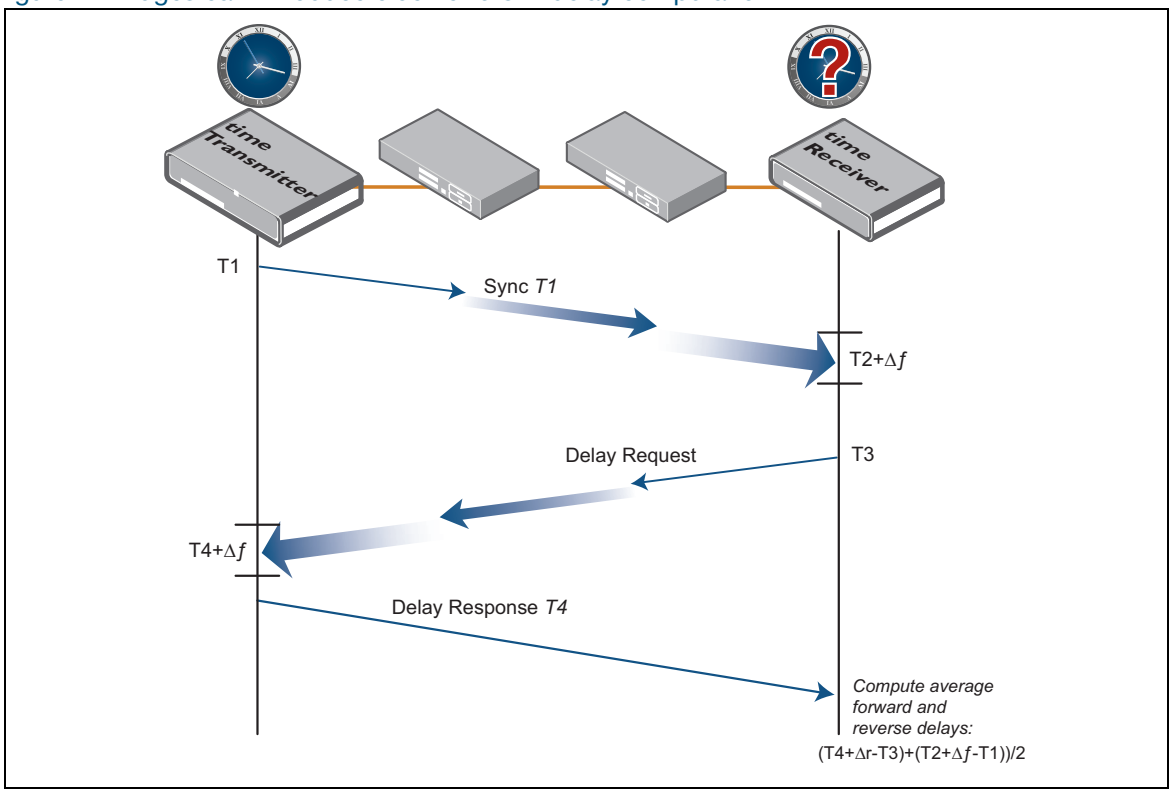
   - The timeReceiver can calculate the reverse delay as (T4-T3). It can then adjust its time stamp clock to account for the wire delay, at least in the beginning stages. After a few iterations of this to make sure the reverse delay measurement is stable, the timeReceiver can now measure the forward delay using captures of (T2-T1).

   - Finally, instead of using just the reverse delay, IEEE 1588 uses both the forward and reverse path delay in steady state to account for the wire delay. This delay is called the **mean path delay**, calculated as {(T4-T3) + (T2-T1)}/2. Once this is computed, the timeReceiver will readjust its clock to align with the timeTransmitter's which now takes into account the wire delay.

How bridges can introduce clock errors

The figure below shows how adding a couple of Ethernet bridges can introduce errors in the delay computation. To start with, the timeReceiver is not synchronized with the timeTransmitter.

Figure 2: Bridges can introduce clock errors in delay computation



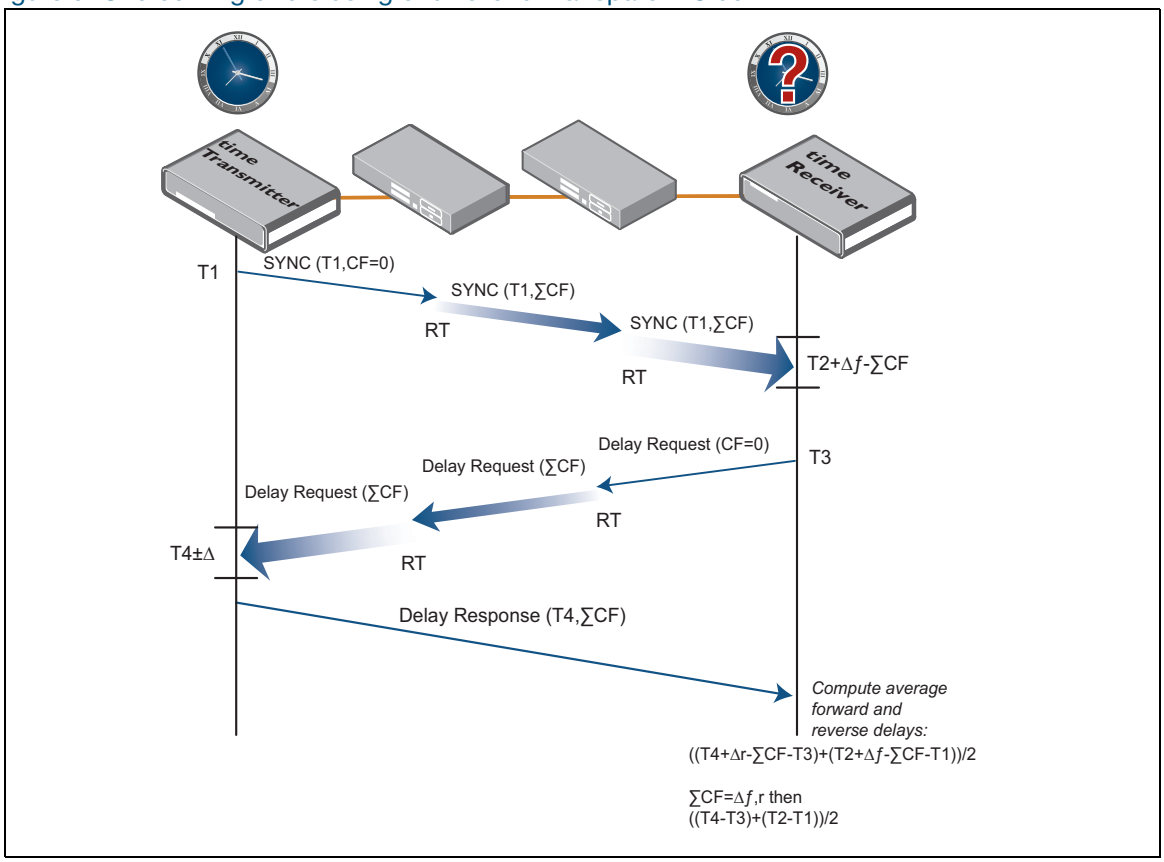
- As the SYNC message leaves the timeTransmitter, it traverses a couple of regular Ethernet bridges which do not participate in IEEE 1588.
- Bridges forward the SYNC message with a delay known as Residence Time (RT), influenced by traffic load and queuing delays. Under no load conditions, the queuing delay can be nearly zero, leaving a fairly constant propagation time through the bridge. Under high load conditions, queues may cause delays in IEEE 1588 messages, resulting in variable delays that worsen with additional bridges in the path.
- The same occurs in the reverse direction for the DELAY REQUEST messages. (The DELAY RESPONSE message is not time critical, so bridge delays have no impact on them).
- The timeReceiver receives several SYNC messages over time, each have a different delay denoted by T2+Delta-Forward. The timeReceiver also receives several DELAY RESPONSE messages containing T4 which will also vary by T4+Delta-Reverse.

As a result, the timeReceiver's calculation of the mean delay will be affected by both Delta-Forward and Delta-Reverse, introducing jitter. This jitter then propagates to the timeReceiver's clock, which continually readjusts itself.

How Transparent Clock overcomes errors

The figure below shows how clock errors can be overcome when Ethernet bridges implement a Transparent Clock by adding an accumulative correction.

Figure 3: Overcoming errors using end-to-end Transparent Clock



- The timeTransmitter sends a SYNC message which in addition to containing a Origin Time Stamp field (which contains the timeTransmitter's Time of Day), also contains a Correction Field (CF). The timeTransmitter sets the CF to 0 prior to sending the SYNC message.
- As the SYNC message arrives at the first bridge, the bridge uses its time stamping clock to capture the time the message starts arriving at its physical port. Similarly as the SYNC message starts to leave the bridge's physical port heading toward the timeReceiver, it uses the same time stamping clock to capture the time the message starts leaving the physical port. With these two time stamp values, the bridge is able to calculate the Residence Time (RT) of the message, which is the delay through bridge, by taking the difference of the two captured time stamps[2]. As a 1-Step device, the bridge will add its residence time to the CF of the SYNC message as it leaves the exiting physical interface heading towards the timeReceiver. In the figure, this is shown as SumCF.
- The SYNC message arrives at the second bridge which also calculates its Residence Time and ‘adds’ it to the CF portion of the SYNC message. The CF is accumulative as makes its way to the

2.The transparent clock's time stamping clock need not be synchronized with a timeTransmitter, as it is computing the difference.

timeReceiver. The timeReceiver then receives the SYNC message, and captures its time stamp (T2) and extracts and stores away both T1 and CF from the message.

- Similarly in the reverse direction, the timeReceiver sends out a DELAY REQUEST message to the timeTransmitter with CF=0 and captures T3, and the bridges add their Residence Time to the CF portion of the DELAY REQUEST message in an additive way. When the DELAY REQUEST message arrives at the timeTransmitter, the timeTransmitter time stamps as usual (T4). The timeTransmitter then takes both the T4 captured time stamp and the CF that arrived and sends a DELAY RESPONSE to the time Receiver.
- The timeReceiver now has in this one sequence of SYNC, DELAY REQUEST, DELAY RESPONSE messages: T1, T2 and accumulative CF in the forward direction, as well as T4, T3 and the accumulative CF in the reverse direction. At this point, the time delay error that was introduced by both bridges (shown as Delta) has been told to the timeReceiver via the accumulative CFs. From the figure, this is shown as SumCF = Delta-Forward or Delta-Reverse. The timeReceiver can effectively subtract out the bridge's residence time (which may be small under low traffic load conditions, or large under high load conditions) during each message sequence. This leaves the timeReceiver with the wire delay and thus is the same as in the first case above when no bridges were used.

Note: Note that IEEE 1588's End-to-End delay mechanism is susceptible to Layer 2 or Layer 3 topology changes. Although End-to-End Transparent Clock bridges eliminate their delay, wire delay remains a factor. Topology changes affecting wire delay may cause the timeReceiver to adjust its clock, introducing some stabilization time to accommodate the new delay.

Peer-to-Peer delay mechanism

From version 5.5.5-2.1 onwards, AlliedWare Plus supports Peer-to-Peer Transparent Clock (TC) on the IE560, IE360, and IE340 platforms. PTP Peer-to-Peer Transparent Clock enables highly accurate time synchronization across distributed systems and supports a wide range of industrial and networking applications.

The PTP Peer-to-Peer Transparent Clock function can be broken up into two logical parts:

1. Correction field updates to SYNC messages.
2. Measuring the peer-delay.

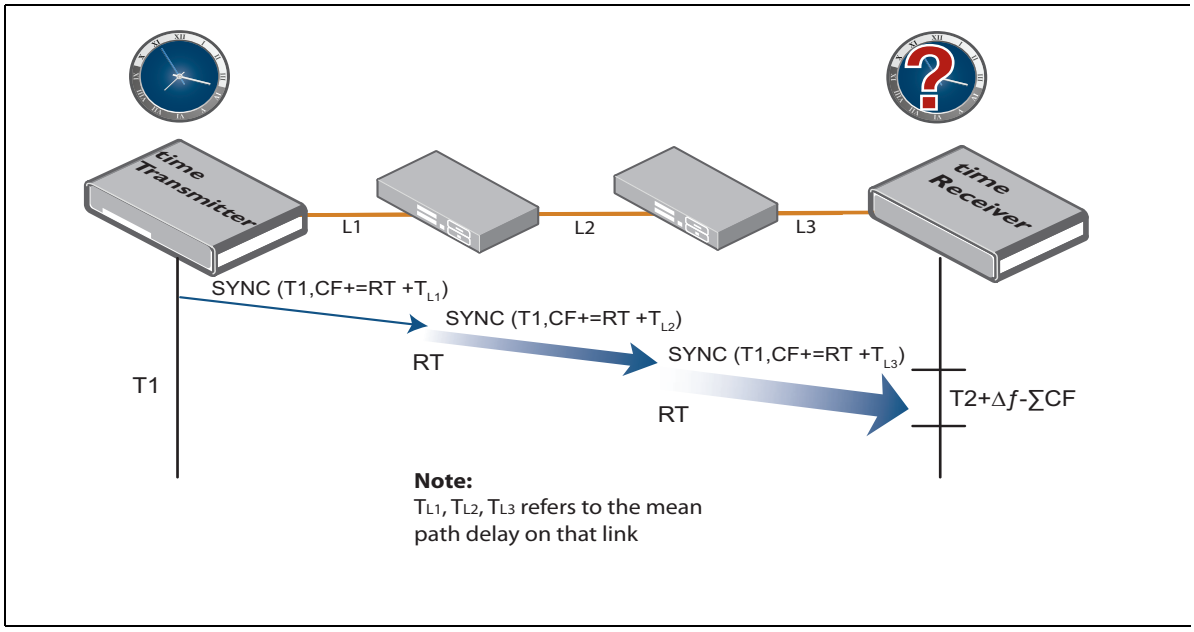
Part 1: Correction field updates to SYNC messages

In Precision Time Protocol (PTP), SYNC messages carry timing information from the timeTransmitter. For two-step transmitters, this information is sent in a SYNC message followed by a Follow_Up message. For simplicity, this section assumes only SYNC messages are used.

As SYNC messages propagate through the network, each PTP peer-to-peer (P2P) transparent clock updates the Correction Field (CF) in the packet. This update includes:

- **Residence time**: The time the packet spends within the device, similar to end-to-end transparent clocks.
- **Mean path delay** (link delay): The estimated delay across the link to the next hop.

By adding both values, the CF accumulates the total delay experienced by the packet across all links and devices. When the SYNC message reaches the time receiver, the CF represents the complete transit time from the timeTransmitter to the time receiver (including all link delays and residence times). The receiver can then combine this correction value with the timestamp in the SYNC message to compute an accurate time of day aligned with the timeTransmitter.



Part 2: Measuring the peer-delay

The peer-delay mechanism measures the mean path delay (peer-delay) between a PTP peer-to-peer (P2P) transparent clock and its neighboring PTP devices. This process runs continuously (every one second by default) and operates independently of SYNC message exchanges.

Measurement Process

1. Peer-Delay Request

   The transparent clock generates a Peer-Delay Request message and records the egress timestamp as T1.

2. Reception at Neighbor

   The neighboring PTP device receives the request and records the ingress timestamp as T2. It then generates a Peer-Delay Response message, inserting T2 and the egress timestamp T3 into the response.

3. Response Reception

   The original transparent clock receives the response and records the ingress timestamp as T4.

Calculation

The mean path delay is calculated using the formula:

Mean Path Delay = ((T4 – T1) – (T3 – T2)) / 2

Where:

- (T4 – T1) = Total round-trip time (link traversal to neighbor and back).
- (T3 – T2) = Residence time within the neighboring device.

Subtracting the residence time from the round-trip time gives the total link traversal time in both directions. Dividing by two provides the average link delay, hence the term ***mean path delay***.

Two-step differences

In the PTP Peer-to-Peer case, the peer delay response is a message type which carries a timestamp of when it left the device. So in the same way as SYNC, two-step clocks will send a peer delay-response follow up packet to convey this information as it can only be obtained after transmission of the peer delay-response packet.

Transparent Clock configuration limitations

Single Active Clock

Although the CLI supports configuring up to five PTP clocks, only one clock can be active at any given time. The active clock is referred to as the “Active Clock.” This means you cannot run PTP Peer-to-Peer (P2P) on one port and PTP End-to-End (E2E) on another simultaneously.

Configuring PTP

To configure PTP Transparent Clock, follow these two main steps:

1. Create, configure, and enable PTP clocks.
2. Attach PTP clocks to ports.

These steps can be performed in any order. For example:

- If an enabled PTP clock is attached to a port, that port will immediately begin performing PTP.
- If a disabled PTP clock is attached, the port will begin performing PTP once the clock is enabled.

Step 1: Create, configure and enable PTP clocks

Create

Although the CLI supports configuring up to five PTP clocks, only one clock can be active at any given time. The active clock is referred to as the Active Clock. This means you cannot run PTP Peer-to-Peer (P2P) on one port and PTP End-to-End (E2E) on another simultaneously.

Configure

To configure a PTP clock, use commands like:

```
awplus(config)# ptp clock test
awplus(config-ptp-clock)# delay-mechanism p2p
awplus(config-ptp-clock)# transport-type ethernet
awplus(config-ptp-clock)# step-type twostep
```

This example creates a P2P, Ethernet, two-step transparent clock named test.

Enable

Once configured, enable the clock:

```
awplus(config-ptp-clock)# clock-enable
```

If the configuration is invalid, this command will fail with an error. Otherwise, the clock becomes active, and all associated ports will have PTP enabled.

To verify, use the command: **show ptp clock**

Step 2: Attach PTP clock to ports

A PTP clock must be attached to ports to be useful. Ports with an active clock will participate in PTP using that clock's configuration

A clock can be attached to a port as follows:

```
awplus# configure terminal
awplus(config)# interface port1.0.1
awplus(config-if)# clock-port test_clock
```

Configuration example

E2E one step transparent clock over Ethernet

1. First create, configure, and enable the clock:

```
awplus(config)# ptp clock test
awplus(config-ptp-clock)# delay-mechanism e2e
awplus(config-ptp-clock)# transport-type ethernet
awplus(config-ptp-clock)# step-type onestep
awplus(config-ptp-clock)# clock-enable
```

2. Configure an untagged VLAN on the PTP ports.

Configure an untagged VLAN on the interface ports used for PTP. If PTP messages are to be switched through the device without VLAN tags, assign an untagged VLAN to those ports.

This example assumes that:

- The ports are already configured in trunk mode.
- The untagged VLAN used for IEEE 1588 PTP frames is configured as the native VLAN.

On AlliedWare Plus devices operating as End-to-End Transparent Clocks, Ethernet-encapsulated PTP packets are Layer 2 switched based on VLAN membership. If PTP frames arrive on one set of interface ports and must egress through other port(s), ensure that:

- The VLAN is a member of all required interfaces.
- VLAN tagging is configured correctly on each port.

```
awplus# configure terminal
awplus(config)# interface port1.0.1-port1.0.2
awplus(config-if)# switchport trunk native vlan 300
awplus(config-if)# clock-port test_clock
```

Note: The **ptp clock** command is now the standard method for configuring PTP in all scenarios. The legacy **ptp-clk** command, previously used for E2E one-step configuration, is still supported but deprecated.

AlliedWare Plus limitations

Current limitations of IEEE 1588 Transparent Clock

The current AlliedWare Plus implementation of the IEEE 1588 Transparent Clock protocol has the following limitations:

- PTP messages (except PEER_DELAY_REQUEST and PEER_DELAY_RESPONSE) are switched through the Layer 2 Transparent Clock like any other tagged or untagged Ethernet traffic.
- You must configure a VLAN to carry these PTP messages on all ports that are part of the same IEEE 1588 domain.
- The Peer-to-Peer (P2P) delay mechanism is supported only when using two-step clock mode.
- Two-step PTP does not support UDP transport.
- Two-step PTP does not support tagged VLANs. This means that only access ports or the native VLAN on a trunk port will work.

Other limitations

- PTP P2P one-step mode is not supported.
- Two-step PTP is not supported on stacks or link aggregators.
- PTP End-to-End (E2E) one-step mode on aggregators is supported but strongly discouraged.
- Two-step PTP cannot be used with UDP transport.

Table 1: AlliedWare Plus PTP Mode Comparison

	end-to-end		**peer-to-peer**	
	L2	**UDP**	**L2**	**UDP**
one-step	Yes	Yes	No	No
two-step	Yes	No	Yes	No

PTP with link aggregation

PTP accuracy can be compromised when packets traverse different paths in the send and receive directions across an aggregator. With this in mind, there are some considerations to consider to help maintain PTP accuracy:

- When PTP packets traverse the switch over an aggregated link, the individual link that the packet is sent on is determined in hardware by the switch chip. For this reason, the characteristics of links used in the aggregated link should be as similar as possible.

  For example, all links in the aggregated link should have the same:

  - link type, e.g. all fibre
  - cable distance

  This helps to ensure that PTP packets traversing the aggregated link share similar characteristics in both the TX and RX direction, and in turn helps to minimise error introduced during the PTP calculation.

Monitoring PTP

Global Use the **show ptp clock** command to show the PTP clocks currently configured. If no clock name is supplied, then all configured clocks are shown.

```
awplus# show ptp clock
PTP Clocks

Abbreviations:
  (A) = Active
  E2E = End to End
  P2P = Peer To Peer

Clock Name                          Parameter          Configuration
-------------------------------------------------------------------------------
default                             Clock Type         Transparent
                                    Delay Mechanism    E2E
                                    Step Type          1-Step
                                    Transport Type     Ethernet, UDP IPv4 & IPv6

PeerClock (A)                       Clock Type         Transparent
                                    Delay Mechanism    P2P
                                    Step Type          2-Step
                                    Transport Type     Ethernet
```

Port/LAG Use the **show ptp port** command to display the configuration and status information of ports that have been associated with a PTP clock.

Example 1

The information below **PTP daemon port information:** is only present if the active clock is E2E two-step, or if the clock is P2P delay mechanism.

```
awplus# show ptp port
PTP Ports

Abbreviations:
  (A) = Active
  E2E = End to End
  P2P = Peer To Peer

                                                Delay
Port Name   State  Clock Name   Clock Type      Mechanism  Step    Transport
-------------------------------------------------------------------------------
port1.0.23  Down   wow(A)       Transparent     P2P        2-Step  Ethernet
port1.0.24  Down   wow(A)       Transparent     P2P        2-Step  Ethernet

PTP daemon port information:

port1.0.23 data
 portState FAULTY
 logMinDelayReqInterval 0
 peerMeanPathDelay 0
 logAnnounceInterval 1
 announceReceiptTimeout 3
 logSyncInterval 0
 master_sync_timeout 0
 delayMechanism 2
 logMinPdelayReqInterval 0
 versionNumber 2
port1.0.24 data
 portState FAULTY
 logMinDelayReqInterval 0
 peerMeanPathDelay 0
 logAnnounceInterval 1
 announceReceiptTimeout 3
 logSyncInterval 0
 master_sync_timeout 0
 delayMechanism 2
 logMinPdelayReqInterval 0
 versionNumber 2
```

Example 2

PTP P2P running with 3 ports as clock ports, one down.

Note for the fields beginning with 'log' the value represents a power of 2. For example, for the field **logMinDelayReqInterval** the value of 0 represents 2^0 = 1 second.

```
awplus# show ptp port
PTP Ports

Abbreviations:
  (A) = Active
  E2E = End to End
  P2P = Peer To Peer

                                                Delay
Port Name   State  Clock Name   Clock Type   Mechanism  Step    Transport
-------------------------------------------------------------------------------
port1.0.1   Up     p2p(A)       Transparent  P2P        2-Step  Ethernet
port1.0.11  Down   p2p(A)       Transparent  P2P        2-Step  Ethernet
port1.0.12  Up     p2p(A)       Transparent  P2P        2-Step  Ethernet

PTP daemon port information:

port1.0.1 data
 portState UNCALIBRATED
 logMinDelayReqInterval 0
 peerMeanPathDelay 629
 logAnnounceInterval 1
 announceReceiptTimeout 3
 logSyncInterval 0
 master_sync_timeout 0
 delayMechanism 2
 logMinPdelayReqInterval 0
 versionNumber 2
port1.0.11 data
 portState FAULTY
 logMinDelayReqInterval 0
 peerMeanPathDelay 0
 logAnnounceInterval 1
 announceReceiptTimeout 3
 logSyncInterval 0
 master_sync_timeout 0
 delayMechanism 2
 logMinPdelayReqInterval 0
 versionNumber 2
port1.0.12 data
 portState MASTER
 logMinDelayReqInterval 0
 peerMeanPathDelay 2364
 logAnnounceInterval 1
 announceReceiptTimeout 3
 logSyncInterval 0
 master_sync_timeout 0
 delayMechanism 2
 logMinPdelayReqInterval 0
 versionNumber 2
```

Timeouts There are counters for how many times various types of timeout have occurred. Timeouts happen when a certain number of messages (expected to be sent at periodic intervals and are not received). These are generally irrelevant for transparent clocks.

However, the two counters included in the output, which are more useful are:

- **sync_mismatch** - How many times the device has been expected a follow up packet but received a SYNC instead.
- **followup_mismatch** - How many times the device has been expecting a SYNC packet but received a follow up instead.

Show ptp statistics

```
awplus# show ptp statistics

port1.0.23 service statistics
 announce_timeout 0
 sync_timeout 0
 delay_timeout 0
 unicast_service_timeout 0
 unicast_request_timeout 0
 master_announce_timeout 0
 master_sync_timeout 0
 qualification_timeout 0
 sync_mismatch 0
 followup_mismatch 0
port1.0.24 service statistics
 announce_timeout 0
 sync_timeout 0
 delay_timeout 0
 unicast_service_timeout 0
 unicast_request_timeout 0
 master_announce_timeout 0
 master_sync_timeout 0
 qualification_timeout 0
 sync_mismatch 0
 followup_mismatch 0
```

show ptp counters

These are specifically the packet counters for various types of packets:

```
awplus# show ptp counters

port1.0.23 packet counters
 rx_Sync 0
 rx_Delay_Req 0
 rx_Pdelay_Req 0
 rx_Pdelay_Resp 0
 rx_Follow_Up 0
 rx_Delay_Resp 0
 rx_Pdelay_Resp_Follow_Up 0
 tx_Sync 0
 tx_Delay_Req 0
 tx_Pdelay_Req 0
 tx_Pdelay_Resp 0
 tx_Follow_Up 0
 tx_Delay_Resp 0
 tx_Pdelay_Resp_Follow_Up 0
port1.0.24 packet counters
 rx_Sync 0
 rx_Delay_Req 0
 rx_Pdelay_Req 0
 rx_Pdelay_Resp 0
 rx_Follow_Up 0
 rx_Delay_Resp 0
 rx_Pdelay_Resp_Follow_Up 0
 tx_Sync 0
 tx_Delay_Req 0
 tx_Pdelay_Req 0
 tx_Pdelay_Resp 0
 tx_Follow_Up 0
 tx_Delay_Resp 0
 tx_Pdelay_Resp_Follow_Up 0
```

PTP over UDP for the x530 Series

From AlliedWare Plus version 5.5.5-1.2 onwards, the x530 Series switches support PTP over UDP transport. In this mode, PTP packets are carried over Layer 4 (UDP) and can use either unicast or multicast transmission. Note that this support applies only to end-to-end (E2E) one-step operation

Overview

PTP over UDP offers several key advantages over traditional Layer 2 (Ethernet-based) transport:

1. Network Flexibility: Enables time synchronization across routed Layer 3 networks, ideal for large or segmented environments.
2. IPv4/IPv6 Compatibility: Supports both protocols, ensuring adaptability and future readiness.
3. Multicast and Unicast Options: Offers flexible deployment models for both broad and targeted synchronization.
4. Infrastructure Integration: Easier to manage with existing network devices, firewalls, and QoS policies.
5. Cloud and Virtualization Friendly: Better suited for modern data centres and hybrid cloud environments.
6. Standards-Based: Compliant with IEEE 1588-2008 (PTPv2), ensuring broad interoperability.

Configuration notes

PTP over UDP works for both IPv4 and IPv6 and the packets can be unicast or multicast.

To enable PTP over UDP with IPv4, use the following command:

```
ptp-clk transparent transport-type udp v4 delay-mechanism e2e step-type onestep
```

For IPv4, the following multicast addresses are supported:

- 224.0.1.129
- 224.0.1.130
- 224.0.1.131
- 224.0.1.132

To enable PTP over UDP with IPv6, use the following command:

```
ptp-clk transparent transport-type udp v6 delay-mechanism e2e step-type onestep
```

For IPv6, the following multicast address is supported:

- FF0x::181

To ensure proper multicast functionality, devices must join the correct multicast group. If not automatically handled, this can be resolved by:

1. Adding a static IGMP group entry to the interface
2. Setting the interface as a static mrouter port
3. Disabling IGMP snooping (globally or per interface)

C613-22098-00 REV F

North America Headquarters | 19800 North Creek Parkway | Suite 100 | Bothell | WA 98011 | USA | T: +1 800 424 4284 | F: +1 425 481 3895
Asia-Pacific Headquarters | 11 Tai Seng Link | Singapore | 534182 | T: +65 6383 3832 | F: +65 6383 3830
EMEA & CSA Operations | Incheonweg 7 | 1437 EK Rozenburg | The Netherlands | T: +31 20 7950020 | F: +31 20 7950021

alliedtelesis.com