Technical Brief

Calnex Solutions Ltd

Implementing IEEE 1588v2 for use in the mobile backhaul

For most, the migration to an all-Ethernet or all-IP network will be a gradual process as network operators endeavour to maximise the lifespan of their existing TDM assets. However, the pace is quickening with the development of the IEEE 1588v2 standard. This standard enables the backhaul network to migrate to all-Ethernet, which provides much-required bandwidth at a far lower cost-per-bit. It is of significant interest to network operators seeking to grow their revenue by capitalising on bandwidth-hungry applications like mobile broadband, music and video downloads. This document provides a brief overview of IEEE 1588v2 and looks at how you can characterise devices that will implement the standard.

Contents

Introduction

Although Ethernet has been the technology of choice for a range on LAN and WAN applications for decades, using it in the mobile backhaul network presents a major challenge. Here, accurate synchronisation of base stations to nanoseconds accuracy is critical to minimise service disruptions and eliminate dropped connections as calls move between adjacent cells. Highly accurate synchronisation also ensures that the radio spectrum is not spread into the adjacent channels. Plus, without stringent phase synchronisation, the multiple signals in LTE's multiple-input/multiple-output (MIMO) architecture can simply cancel one another out. And this is where IEEE1588v2 comes in.

IEEE1588v2 (also known as Precision Time Protocol, PTP) is an industry-standard protocol that enables the precise transfer of frequency and time to synchronise clocks over packet-based Ethernet networks. It synchronises the local slave clock on each network device with a system Grandmaster clock and uses traffic time-stamping, with sub-nanoseconds granularity, to deliver the very high accuracies of synchronisation needed to ensure the stability of base station frequency and handovers. Timestamps between master and slave devices are sent within specific PTP packets and in its basic form the protocol is administration-free.

Of course, the precision and performance of the IEEE 1588v2 protocol is based on the precision of the timestamp. The timestamps of incoming and outgoing packets clearly need to be recorded and assessed to ensure synchronisation of master and slave devices. Differences in time and frequency between clocks and subsequent equipment corrections need to be evaluated, while clocks must be measured to ensure they are within their specified limits. Further, delays and drifts in sync and their effect on the transfer of timing through the network need to be considered too.

Here, we examine the various methods you can use to characterise and measure the precision of timestamp synchronisation, as well as the accuracy of clocks, network devices and topology, before deploying equipment in an operational network.

Maintaining Sync using IEEE1588v2

In a packet transport system, clocks communicate with each other over the communication network using PTP. All clocks, whether master or slave, lead back to – and ultimately derive their time from – the 'Grandmaster' clock. There are 5 types of PTP clock devices:

Ordinary Clock	A single port device that can be a Master or Slave clock.
Boundary Clock	A multi port device that can be a Master or Slave clock. In general deployment, a boundary clock has a built-in Slave clock that recovers a clock. This clock is then used to drive the built-in Master, which supplies the clock to the next node.
End-to-end Transparent Clock	A multi port device that is not a Master or Slave clock but a bridge between the two. Forwards and corrects all PTP Messages. Correction achieved by addition of the bridge residence time into a correction field within the header of the message.
Peer-to-peer Transparent Clock	A multi port device that is not a Master or Slave clock but a bridge between the two. Forwards and corrects Sync and Follow_Up messages only. Correction achieved by addition of the bridge residence time + the peer-to-peer link delay, into a correction field within the header of the message.
Management Node	A device that configures and monitors clocks.

Table 1 – PTP Device Types

Master and slave network devices are kept synchronized by the transmission of timestamps sent within the PTP messages. There are two types of message in the PTP protocol: Event Messages and General Messages. Event messages are timed messages whereby an accurate timestamp is generated both at transmission and receipt of the message. General messages do not require timestamps but may contain timestamps for their associated event message.

Event Messages	General Messages
Sync	Announce
Delay_Req	Follow_Up
Pdelay_Req	Delay_Resp
Pdelay_Resp	Pdelay_Resp_Follow_Up
	Management
	Signaling

Table 2 – PTP Messages

Propagation Delay Measurement Mechanisms

There are two mechanisms used in PTP to measure the propagation delay between PTP ports:

- The Delay Request-Response Mechanism

  This mechanism uses the messages Sync, Delay_Req, Delay_Resp and, if required, Follow_Up.

- The Peer Delay Mechanism

  This mechanism uses the messages Pdelay_Req, Pdelay_Resp and, if required, Pdelay_Resp_Follow_Up.
  It is restricted to topologies where each peer-to-peer port communicates PTP messages with, at most, one other such port.

Ports on Ordinary or Boundary clocks can use either mechanism; ports on end-to-end transparent clocks are independent of these mechanisms, and ports on peer-to-peer transparent clocks use only the peer delay mechanism. It should also be noted that the two mechanisms do not inter-work on the same communication path.

Synchronisation

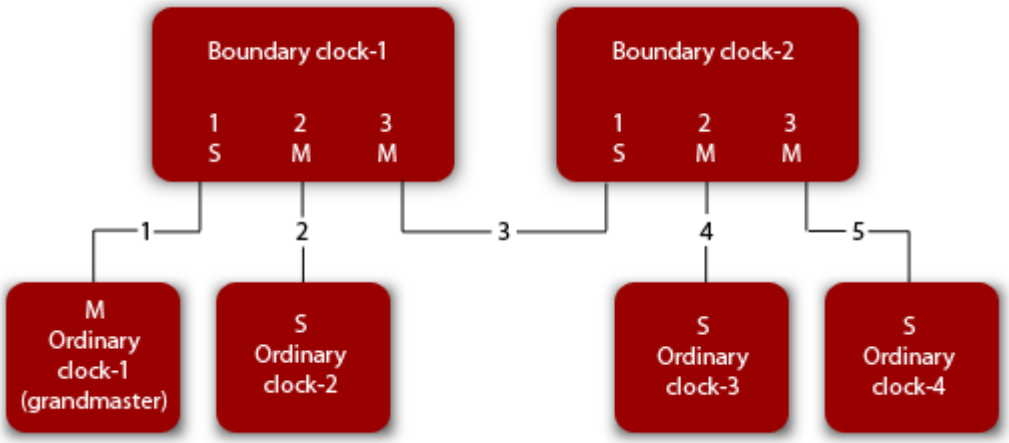
There are two phases in the normal execution of the protocol:

- Phase 1 establishes the Master-Slave hierarchy.
- Phase 2 synchronises the clocks using either of the two mechanisms described above.

Establishing the Master-Slave Hierarchy

In each port of any Ordinary or Boundary clock there is a PTP state machine. These state machines use the 'Best Master Clock Algorithm' (or BMCA) to establish the Master for the path between two ports. The statistics of the remote end of a path are provided to each state machine by the Announce message. Since the local clocks statistics are already known by the state machine, a comparison can be made as to which is the best Master.

A simple Master-Slave hierarchy is shown in the following diagram. Paths 1, 2, 3, 4, and 5 may contain transparent clocks, but these clocks do not participate in the Master-Slave hierarchy.



(M= Master, S = Slave)

Figure 1 – Simple Master-Slave Hierarchy

Synchronising Ordinary and Boundary Clocks (using the delay request-response mechanism)

Method 1.

After the Master-Slave hierarchy has been established the clock synchronisation phase can start. This consists of the exchange of PTP timing messages on the communications path between the two clocks.

There are two parts to this synchronisation method:

(1) Measuring the propagation delay between Master and Slave. Performed using the delay request-response mechanism.

(2) Performing the clock offset correction. Once the propagation delay is known the Master can send Sync and optional Follow_Up messages containing its master timestamp. These are actually sent in part 1 also, but the ratio of propagation delay measurement to Sync message would usually be quite low, that is, the propagation delay will be measured less than the clock offset correction.

(1) Measuring Propagation Delay in Clocks supporting the Delay Request-Response Mechanism

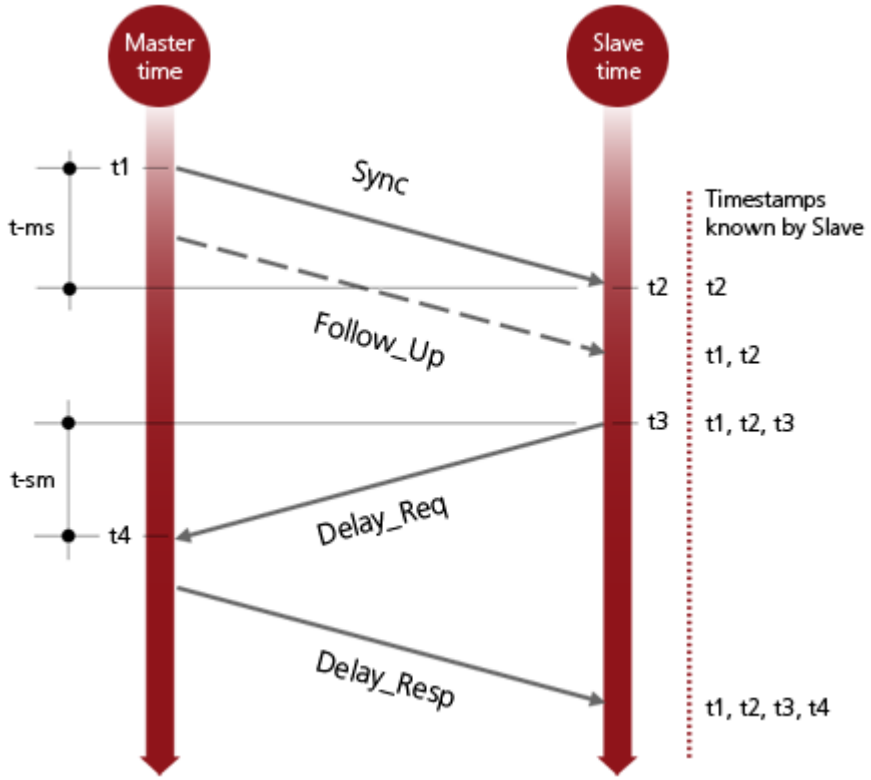


Figure 2 – Propagation Delay Message Exchange

t_1 = Master Time at point of sending Sync Message.
t_2 = Slave Time at point of receiving Sync Message.
t_3 = Slave Time at point of sending Delay_Req Message.
t_4 = Master Time at point of receiving Delay_Req Message.

Once the Slave knows the timing of t1, t2, t3, and t4, it can calculate the mean propagation delay (tmpd) of the messages path. This is calculated from:

$$\frac{(t2 - t1) + (t4 - t3)}{2}$$

The Sync and optional Follow_Up[1] messages give the master to slave message propagation time (t-ms).
The Delay_Req and Delay_Resp messages give the slave to master message propagation time (t-sm).

Any asymmetry between t-ms and t-sm introduces an error into the clock offset correction.

(2) Performing the Clock Offset Correction

Once the Master to Slave propagation delay is known by the Slave, the clock correction can occur in the Slave device.

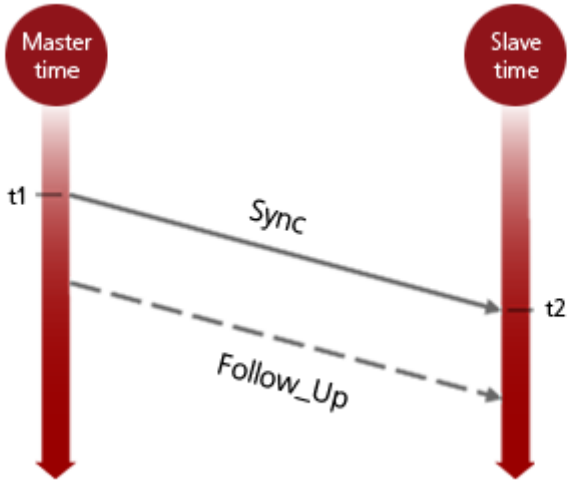


Figure 3 – Basic Synchronisation Message Exchange

The Slave uses the Sync message and the optional Follow_Up message to calculate the clock offset from Master to Slave. This is calculated from

$$t2 - t1 - tmpd$$

The above is a simple model that does not show any end-to-end transparent clocks. End-to-end transparent clocks do not serve as Master or Slave clocks but they do insert/update a correction field into event messages that allows adjustment of timestamps at the Slave device to remove residence times through any transparent clock devices/bridges. The above model would not include any peer-to-peer transparent clocks as these cannot coexist on the same communications path as the delay request-response mechanism.

[1] The Follow_Up message is optional as the t1 timestamp may be sent in the Sync message meaning that the Follow_Up message is not required.

Method 2.

After the Master-Slave hierarchy has been established the clock synchronisation phase can start.

There are two parts to this synchronisation method:

(1) Peer-to-peer ports maintain a measurement of the link propagation to each peer. They do this using the peer delay mechanism.

(2) Performing the clock offset correction. Once the link propagation is known the master can send Sync and optional Follow_Up messages containing its master timestamp.

(1) Measuring Link Propagation Delay in Clocks Supporting Peer-to-Peer Path Correction

The link delay between two ports that implement the peer delay mechanism can be measured using the following exchange of messages.

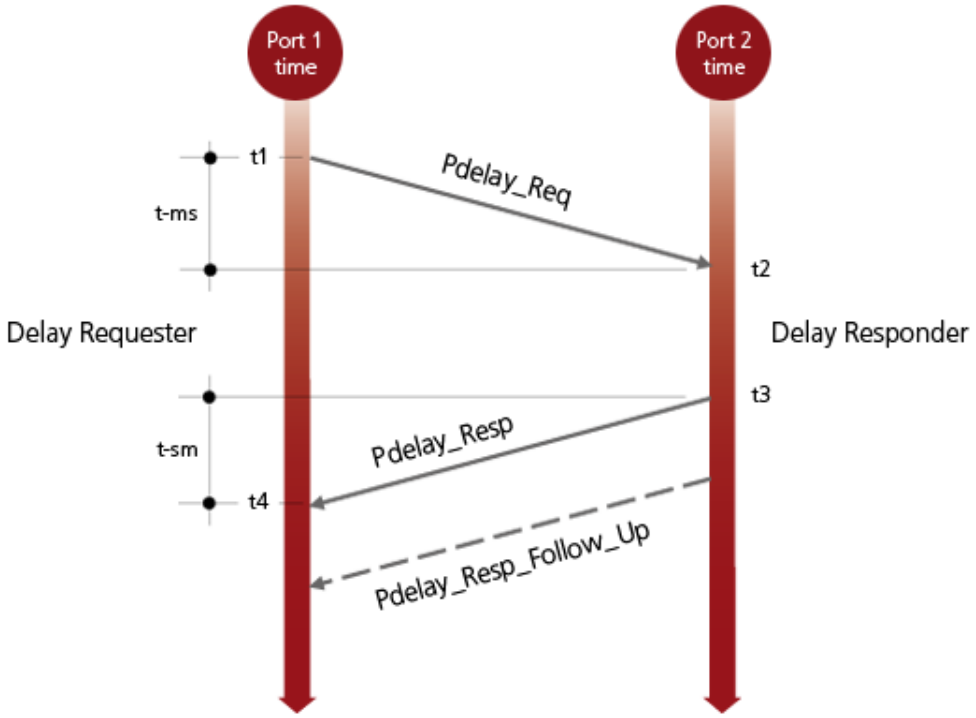


Figure 4 – Link Delay Measurement

t_1 = Port 1 Time at point of sending Pdelay_Req Message.
t_2 = Port 2 Time at point of receiving Pdelay_Req Message.
t_3 = Port 2 Time at point of sending Pdelay_Resp Message.
t_4 = Port 1 Time at point of receiving Pdelay_Resp Message.

Once Port 1 knows the timing of t1, t2, t3, and t4, it can calculate the mean link delay (tmld). This is calculated from:

$$\frac{(t2 - t1) + (t4 - t3)}{2}$$

It then uses this value when calculating the correction field for each Sync or Follow_Up message that passes through the bridge. The outgoing correction field will be the sum of the residence time, the mean link delay and any correction field from upstream ports.

The Pdelay_Req, Pdelay_resp and optional Pdelay_Resp_Follow_Up[2] messages allow the round trip link delay to be calculated (t-ms + t-sm).

Any asymmetry between t-ms and t-sm introduces an error into the clock offset correction.

(2) Performing the Clock Offset Correction

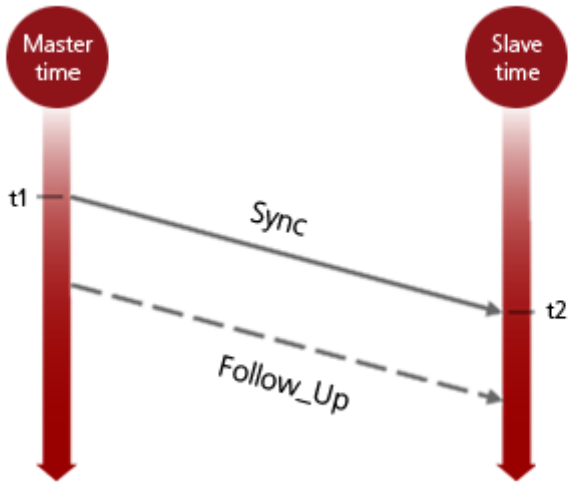


Figure 5 – Basic Synchronisation Message Exchange

The Slave uses the Sync message and the optional Follow_Up message to calculate the clock offset from Master to Slave. This is calculated from

$$t2 - t1 - \text{correctionField}$$

A benefit of peer-to-peer path correction is that the path delay of each individual Sync or Follow_Up message is calculated as it travels along the communication path. It is therefore not affected by a change to the path. When using this mechanism the clock synchronisation does not require the return path to be calculated as it does in the basic exchange, i.e. the Delay_Req, Delay_Resp messages shown in **Figure 1** do not occur. The path delay between the Master and Slave in this mechanism is simply contained within the correction field of each Sync or Follow_Up message.

An added benefit is that the Master has less processing to do as it will not receive any Delay_Req messages. This can be a major benefit in linear topologies, when many slave clocks are connected to a single master.

[2] The Pdelay_Resp_Follow_Up message is optional as the difference between the t2 and t3 timestamps can be returned solely in the Pdelay_Resp message meaning that the Pdelay_Resp_Follow_Up message would not be required.

Topologies

Different applications favour different topologies. The three main topologies – Hierarchal, Linear and Multiply Connected – are shown in the following diagrams (with cyclic paths that should be avoided):

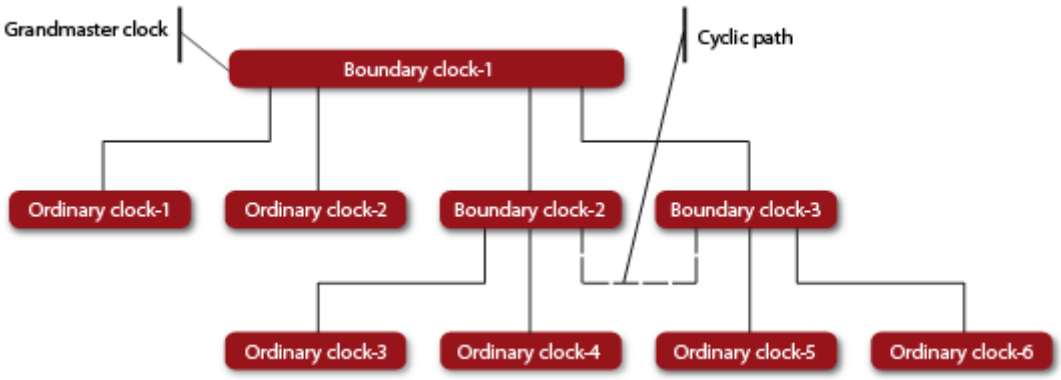


Figure 6 – Hierarchical Topology

Figure 7 – Linear Topology

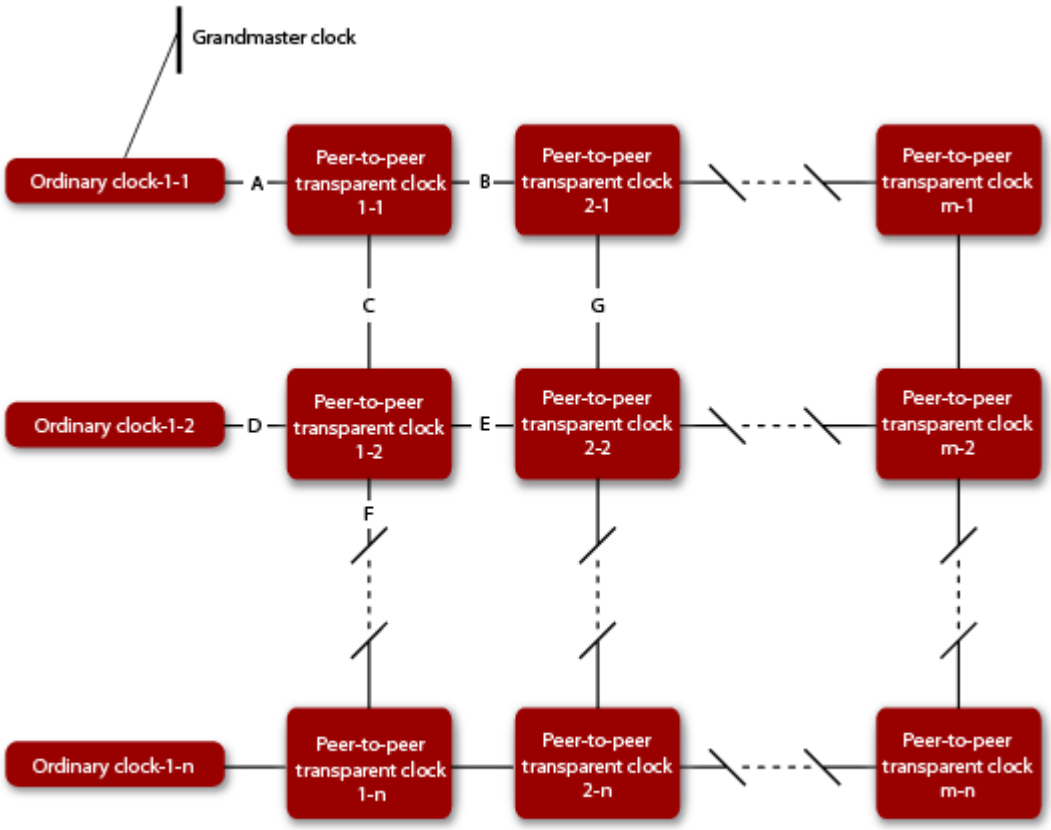


Figure 8 – Multiply Connected Topology

Transport of PTP Messages

PTP messages can be transported over several types of protocol. These are listed below.

Transport Type
PTP over UDP over IPv4
PTP over UDP over IPv6
PTP over IEEE 802.3/ Ethernet
PTP over DeviceNET
PTP over ControlNET
PTP over IEC 61158 Type 10 (Fieldbus)

Table 3 – PTP Transport Protocols

The mapping of each PTP message into the lower layer protocols is shown in the following sections.

PTP over UDP over IPv4 over Ethernet

When carried over UDP the first byte of the PTP message immediately follows the final byte of the UDP header. The UDP port number field identifies the UDP datagram as a PTP message.

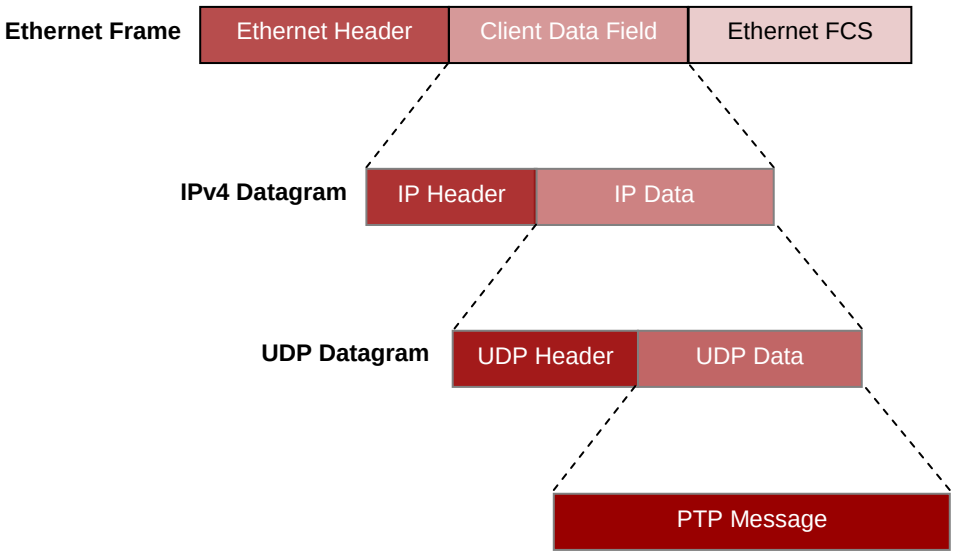


Figure 4 – PTP Message within UDP over IPv4 over Ethernet

PTP over UDP over IPv6 over Ethernet

When carried over UDP the first byte of the PTP message immediately follows the final byte of the UDP header. The UDP port number field identifies the UDP datagram as a PTP message.

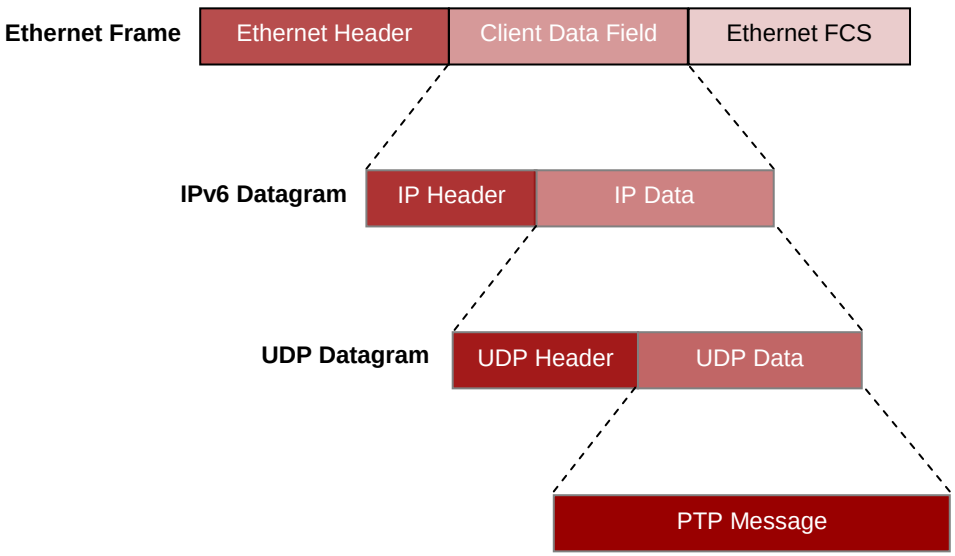


Figure 5 –PTP Message within UDP over IPv6 over Ethernet

PTP over IEEE 802.3/Ethernet

When carried over Ethernet the first byte of the PTP message occupies the first byte of the client data field of the Ethernet frame. The Ethernet type field is set to 0x88F7 and identifies the client data field as a PTP message.

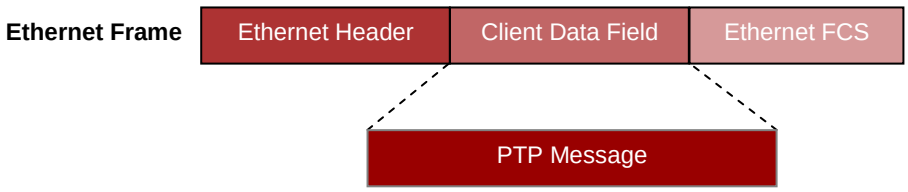


Figure 6 – PTP Message within an Ethernet Frame

PTP Message Formats

All PTP Messages consist of a header, body and optional suffix.

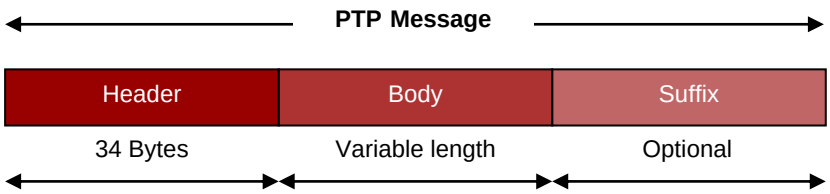


Figure 7 – Basic PTP Message Format

Header

The header is common to all PTP messages. It is 34 bytes long and its format is shown below.

PTP Message Header Format									
Bits								Octets	Offset
7	6	5	4	3	2	1	0		
transportSpecific				messageType				1	0
Reserved				versionPTP				1	1
messageLength								2	2
domainNumber								1	4
Reserved								1	5
Flags								2	6
correctionField								8	8
Reserved								4	16
sourcePortIdentity								10	20
sequenceID								2	30
controlField								1	32
logMessageInterval								1	33

Table 4 – PTP Header Format

The **messageType** field defines which type of message is contained in the body of the message, for example Sync, Delay_Req, Delay_Resp etc.

The **messageLength** field defines the full length of the PTP message, i.e. including the header, body and any suffix (but excluding any padding).

The **domainNumber** field identifies the domain the PTP message belongs to. A domain is a logical grouping of clocks that synchronise to each other using the protocol, but that are not necessarily synchronised to clocks in another domain.

The **flags** field contains various flags to indicate status.

The **correctionField** contains a correction value in nanoseconds for residence time within a transparent clock and will also include the path delay for peer-to-peer transparent clocks.

The **sourcePortIdentity** field identifies the originating port for this message.

The **sequenceID** field contains (with some exceptions) a sequence number for individual message types.

The **controlField** is a historical field whose value depends on the message type; it conforms to version 1 of the standard. The field is similar to the messageType field but with less options.

The **logMessageInterval** field is determined by the type of the message.

Further information on the specific details of each field can be found in standard IEEE P1588.

Body

As previously discussed there are several different types of PTP message. Each is discussed below.

Announce Message

The announce message is used to indicate the capabilities of a clock to the other clocks on the same domain. This allows the Master-Slave hierarchy to be established. (See the IEEE 1588 standard for the specific field details.)

Announce Message Format									
Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (13.3)								34	0
originTimestamp								10	34
currentUtcOffset								2	44
Reserved								1	46
grandmasterPriority1								1	47
grandmasterClockQuality								4	48
grandmasterPriority2								1	52
grandmasterIdentity								8	53
stepsRemoved								2	61
timeSource								1	63

Table 5 – Announce Message Format

Sync Message

The sync request message is sent by a Master clock and contains the Master time when the Sync message was sent. If the Master clock is a two-step clock, the timestamp in the Sync message will be set to zero and the actual sending timestamp will be sent afterwards in the associated Follow_Up message. Sync messages are sent in both types of PTP delay measurement mechanism.

Sync Message Format									
Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (13.3)								34	0
originTimestamp								10	34

Table 6 – Sync Message Format

.

Delay_Req Message

The format of the Delay_Req message is identical to the Sync message. The Delay_Req message is sent by a Slave clock and contains the Slave time when the Delay_Req message was sent. Delay_Req messages are sent only in the delay request-response mechanism.

Delay_Req Message Format									
Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (13.3)								34	0
originTimestamp								10	34

Table 7 – Delay_Req Message Format

Follow_Up Message

The Follow_Up message is optionally sent by a Master clock and contains the Master time when the Sync message was sent. It is used when the Master clock is a two-step clock, i.e. two steps – Sync message and Follow-Up message. Follow_Up messages are sent in both types of PTP delay measurement mechanism.

Follow_Up Message Format									
Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (13.3)								34	0
preciseOriginTimestamp								10	34

Table 8 – Follow_Up Message Format

Delay_Resp Message

The Delay_Resp message is sent by the Master clock and contains the Master time when the Delay_Req message was received. Delay_Resp messages are sent only in the delay request-response mechanism.

Delay_Resp Message Format									
Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (13.3)								34	0
receiveTimestamp								10	34
requestingPortIdentity								10	44

Table 9 – Delay_Resp Message Format

The **requestingPortIdentity** field contains the sourcePortIdentity field (from the header of the associated Delay_Req message) of the Slave that requested this Delay_Resp message.

Pdelay_Req Message

The Pdelay_Req message is sent by a ‘delay requester’ peer-to-peer clock and contains the ‘delay requester’ peer-to-peer clock time when the Pdelay_Req message was sent. Pdelay_Req messages are sent only in the peer delay mechanism.

Pdelay_Req Message Format									
Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (13.3)								34	0
originTimestamp								10	34
reserved								10	44

Table 10 – Pdelay_Req Message Format

The **reserved** field is used to make the message length the same as the Pdelay_resp message as some networks have different transmit times for different bridge message lengths which would introduce asymmetry errors.

Pdelay_Resp Message

The Pdelay_Resp message is sent by a ‘delay responder’ peer-to-peer clock and contains the ‘delay responder’ peer-to-peer clock time when the Pdelay_Req message was received. Pdelay_Resp messages are sent only in the peer delay mechanism.

Pdelay_Resp Message Format									
Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (13.3)								34	0
receiveReceiptTimestamp								10	34
requestingPortIdentity								10	44

Table 11 – Pdelay_Resp Message Format

The **requestingPortIdentity** field contains the sourcePortIdentity field (from the header of the associated Pdelay_Req message) of the ‘delay requester’ peer-to-peer clock that requested this Pdelay_Resp message.

If the ‘delay requester’ peer-to-peer clock is a two-step clock, the timestamp in the Pdelay_Resp message will be set to zero and the actual sending timestamp will be sent afterwards in the associated Pdelay_Resp_Follow_Up message[3].

[3] There is also the option of sending a turnaround time instead of the sending timestamp.

Pdelay_Resp_Follow_Up Message

The Pdelay_Resp_Follow_Up message is optionally sent by a ‘delay responder’ peer-to-peer clock and contains the ‘delay responder’ peer-to-peer clock time when the Pdelay_Resp message was sent. It is used when the ‘delay responder’ is a two-step clock, i.e. two steps – Pdelay_Resp message and Pdelay_Resp_Follow_Up message. This message also has the option of sending a turnaround time instead of the sending timestamp. The turnaround time of receiving the Pdelay_Req to sending back the Pdelay_Resp. Pdelay_Resp_Follow_Up messages are sent only in the peer delay mechanism.

Pdelay_Resp_Follow_Up Message Format									
Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (13.3)								34	0
responseOriginTimestamp								10	34
requestingPortIdentity								10	44

Table 2 – Pdelay_Resp_Follow_Up Message Format

The **requestingPortIdentity** field contains the sourcePortIdentity field (from the header of the associated Pdelay_Req message) of the ‘delay requester’ peer-to-peer clock that requested this Pdelay_Resp message.

Signalling Message

Signalling Message Format									
Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (13.3)								34	0
targetPortIdentity								10	34
One or more TLVs								N	44

Table 33 – Signalling Message Format

The **targetPortIdentity** field contains the address of the target port/ports of this message.

TLV = Type, Length, Value Identifier

Management Message

PTP management messages are used to transmit information from a clock to a node manager and from a node manager to one or more clocks.

Management Message Format									
Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (13.3)								34	0
targetPortIdentity								10	34
startingBoundaryHops								1	44
boundaryHops								1	45
Reserved				actionField				1	46
Reserved								1	47
managementTLV								M	48

Table 4 – Management Message Format

The **targetPortIdentity** field contains the address of the target port/ports of this message.

The **startingBoundaryHops** field contains the number of boundary clocks that this message is allowed to be retransmitted by.

The **boundaryHops** field contains the number of remaining boundary clock retransmissions left for this particular management message request or reply. This field is an identical value to the startingBoundaryHops field for the initial transmission from the issuing clock/node.

The **actionField** contains the type of action that this management message is required to perform. The types are Get, Set, Response, Command and Acknowledge.

The **managementTLV** fields are shown below.

Bits								Octets	TVL Offset
7	6	5	4	3	2	1	0		
tlvType								2	0
lengthField								2	2
managementID								2	4
dataField								N	6

Table 5 – ManagementTLV field Format

The **tlvType** field shall be set to MANAGEMENT (0x0001).

The **lengthField** is the length of the TLV. The format is 2+N where N is an even number.

The **managementID** field defines the type of management message. Examples of which are Initialize, Enable_Port, Disable_Port. See the IEEE 1588 standard for the full list of management IDs.

The **dataField** is managementID dependant. See the IEEE 1588 standard for details.

Testing and Introducing Impairments

While IEEE 1588v2 PTP uses an exchange of specially designed packets to calculate the difference in time and frequency between two clocks, the overall precision of an Ethernet-based mobile backhaul is dependant on many factors. Different topologies, equipment and traffic management introduce different amounts of latency and synchronisation jitter. Servicing delays impact the accuracy of the timestamp which, in turn, reduces the precision of the clock adjustment calculations. Synchronisation between clocks can drift when the frequency offset between the master and the slave is being corrected. Additionally network equipment such as switches, routers, and gateways can all introduce latency and wander errors.

Clearly, there is a need to test network synchronisation before and after the deployment of a network and/or equipment. One way is to use Calnex Solution's Paragon Sync test solution, configured as shown below:

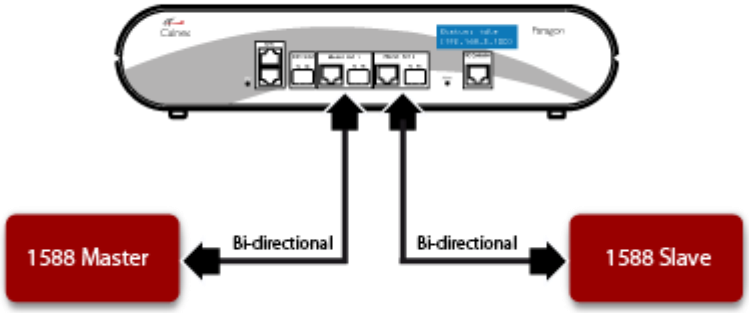
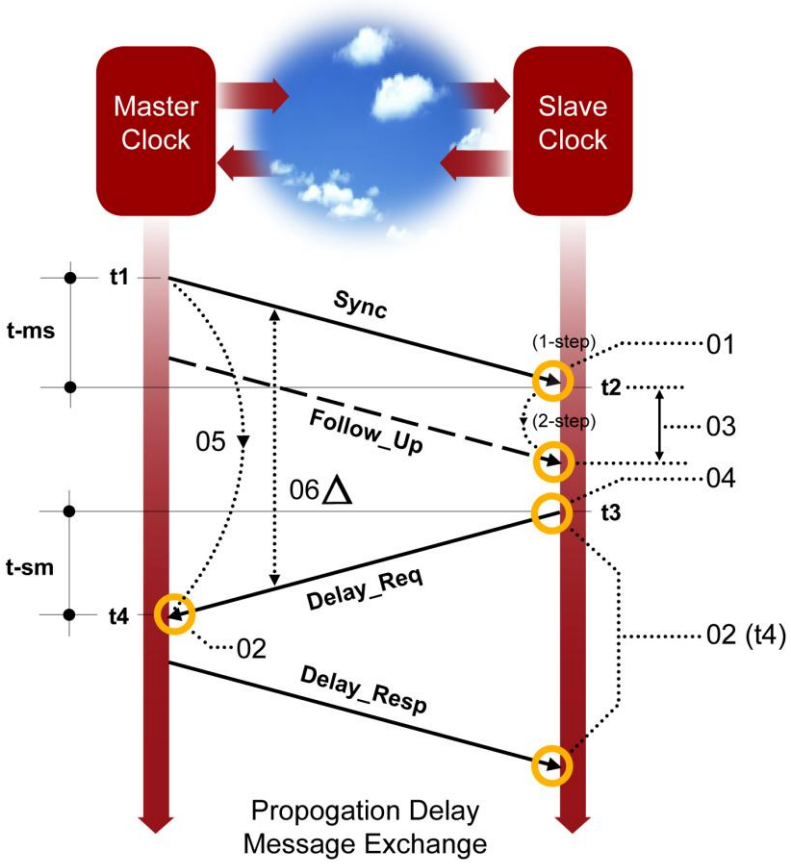


Figure 8 – IEEE 1588v2 Example Test Setup

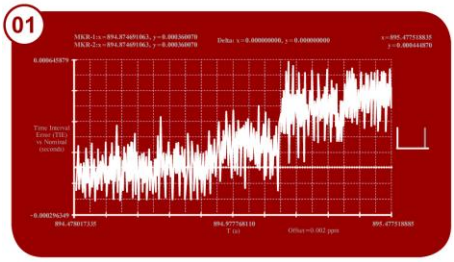
Configured in this way, the Calnex Paragon can:

- Perform analysis of 1588v2 messages and timestamps as shown in the graphs below.
- Run all the ITU-T G.8261 test cases.
- Capture a PDV profile from a real network or trial network over long periods (many days) and replay the same profiles back in the lab during testing.
- Add impairments to PTP messages:
    - Lost PTP message – ability to delete a specific PTP message type.
    - Duplicated PTP message – ability to duplicate a specific PTP message type.
    - Mis-ordered PTP message – ability to mis-order specific PTP message types or mis-order between types, e.g. swap order of a Sync and Follow_Up message.
- PDV insertion onto a specific PTP message type on forward and reverse path. This allows path asymmetry to be tested.
    - Ability to insert an equivalent delay value into the correction field of the PTP message header. This allows the emulation of a transparent clock.

Two-way Timing Protocol, PDVs and Wander

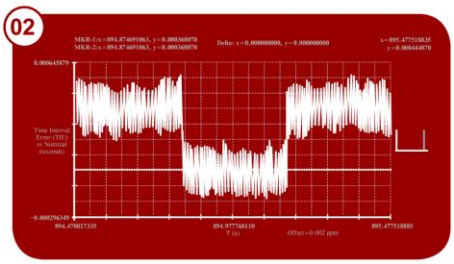


$$\text{Mean Propogation Delay} = \frac{(t2 - t1) + (t4 - t3)}{2}$$



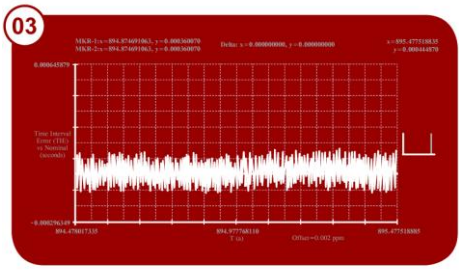
Sync PDV (1-step and 2-step)

The Sync PDV (Packet Delay Variation) graph plots the PDV based on the Sync message arrival time and its embedded timestamp. (When using a 2-step clock, the timestamp in the Follow_Up message is used instead of the timestamp in the Sync.) This determines the PDV from the Master to the Slave. If captured next to the Slave device, this shows the actual PDV experienced by the Sync message as it travels from Master to Slave.



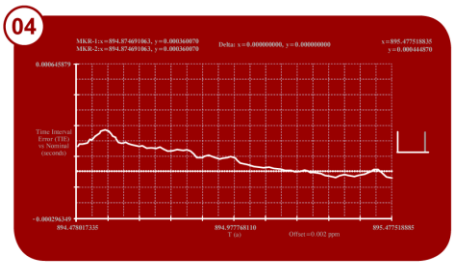
Delay_Request PDV

The Delay_Req PDV is calculated using the arrival time of the Delay_Req messages and the embedded timestamp in the corresponding Delay_Resp messages (t4). This graph shows the PDV from the Slave to the Master, as experienced by Delay_Req messages travelling to the Master clock.



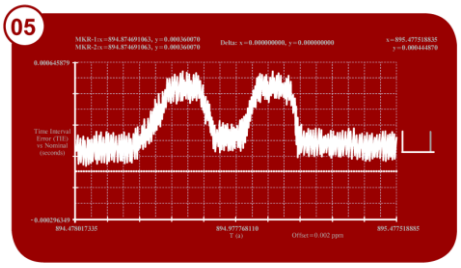
Follow_Up PDV

The Follow_Up PDV graph plots the variation in arrival time between Sync and Follow_Up messages. This can indicate whether or not there is a regular gap between these messages and whether the gap can affect the operation of the Slave clock recovery. An excessive gap may lead to the Slave discarding the corresponding Sync message as it time-outs waiting for its arrival, hence leading to a reduction in the number of timing events received by the Slave clock recovery circuit.



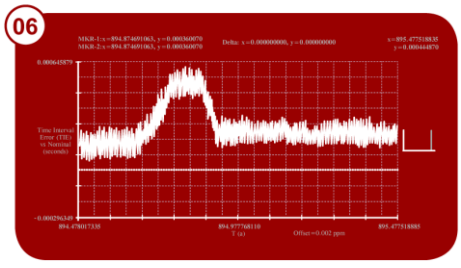
Slave Clock Wander

The Slave Clock Wander is a measurement of the stability of the recovered Slave clock. The embedded timestamp in the Delay_Req message (T3) is plotted while synchronised to the Master. This gives the ability to monitor the Slave clock wander on the Ethernet interface without needing access to the actual recovered clock signal.



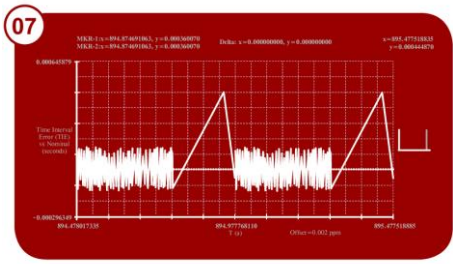
Round Trip Delay Variation

The Round Trip Delay (RTD) variation graph plots the variation in {(t2-t1)+(t4-t3)}/2, which is the RTD calculation performed by the Slave. This indicates the stability of this RTD, and allows the user to assess whether changes in RTD are impacting the Slave clock recovery and stability. Significant variation will stress the Slave's clock recovery circuit to determine the true path delay. Floor delay is an important parameter in some Slave clock implementations. If the floor delay moves up for long periods, this may cause wander in the recovery clock. Other designs are reported to use certain bands of delay e.g. only the message pairs that exhibit a RTD between x% and y% of the total range measured. If the density therefore changes, this may cause wander in the recovered clock.



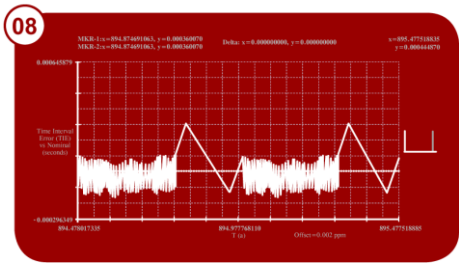
Asymmetry in Path Delay

The Asymmetry graph plots the difference between Master > Slave delay and Slave > Master delay. This variation is effectively a variation in symmetry, which is a basic assumption of IEEE-1588. If a variation is detected, the user can assess its impact on Master and Slave clock operation. As the slave assumes symmetry, changes in symmetry will cause errors in the calculation of time. Significant, sustained changes in path symmetry may lead to wander in the recovered clock.



Sync IPG

The Sync Inter-Packet Gap (IPG) graph shows the variation in arrival time of the Sync messages. There is no requirement for the messages to arrive in a regular spacing as each carries a timestamp. However, excessive variation will lead to periods where the number of Sync messages being received is significantly reduced. If this occurs at the same time as excessive PDV in the network, the pre-selection algorithms in some Slave devices may lead to very few Sync messages being selected to be passed to the clock recovery circuit. This can lead to increased wander on the output.



Delay_Resp Round Trip Delay

The Delay-Resp Round Trip Delay (RTD) graph shows the in time taken by the Master to respond to the Delay_Req message with a Delay_Resp message. There is no specification for this parameter. The Slave must wait for the Delay_Resp message to arrive in order to be informed of the t4 time. Without this information, it can not determine the Round Trip Delay. The Slave will not wait forever for the response to arrive and will have a time-out implemented. Should the Master's response time significantly increase (e.g. when multiple Slaves all send Delay_Req messages concurrently), a time-out may be invoked. This will lead to a significant reduction in the number of RTD calculation events available to the Slave and hence lead to increased wander on the recovered clock.

Port	Packet #	Arrival Time	Inter-Packet Time	Inter-Message Time	messageType	sequenceId
	707	1.943000765	0.004998235	0.008000635	SYNC	50575
	708	1.943398700	0.000397935	0.007810555	DEL-REQ	44685
	709	1.946003160	0.002604460	0.008000630	DEL-RESP	44685
	710	1.951001400	0.004998240	0.008000635	SYNC	50576
	711	1.951037100	0.000035700	0.007638400	DEL-REQ	44686
	712	1.954003795	0.002966695	0.008000635	DEL-RESP	44686
	713	1.958857020	0.004853225	0.007819920	DEL-REQ	44687
	714	1.959002035	0.000145015	0.008000635	SYNC	50577
	715	1.965997550	0.006995515	0.006995515	SYNC	50578
	716	1.974006345	0.008008795	0.008008795	SYNC	50579
	717	1.982005540	0.007999195	0.007999195	SYNC	50580
	718	1.990004250	0.007998710	0.007998710	SYNC	50581
	719	1.997992885	0.007988635	0.007988635	SYNC	50582
	720	2.004998480	0.007005595	0.007005595	SYNC	50583
	721	2.012997675	0.007999195	0.007999195	SYNC	50584
	722	2.020996390	0.007998715	0.007998715	SYNC	50585
	723	2.028995105	0.007998715	0.007998715	SYNC	50586
	724	2.029303420	0.000308315	0.070446400	DEL-REQ	44688
	725	2.031997985	0.002694565	0.077994190	DEL-RESP	44688
	726	2.036951420	0.004953435	0.007648000	DEL-REQ	44689
	727	2.036996220	0.000044800	0.008001115	SYNC	50587
	728	2.040019260	0.003023040	0.008021275	DEL-RESP	44689
	729	2.043993175	0.003973915	0.006996955	SYNC	50588
	730	2.046842460	0.002849285	0.009891040	DEL-REQ	44690
	731	2.048992850	0.002150390	0.008973590	DEL-RESP	44690
	732	2.051993810	0.003000960	0.008000635	SYNC	50589

The 1588v2 Header capture on the Calnex Paragon can also help troubleshoot issues with Master and Slave clocks. The screen above shows a deviation to the normal Sync-Delay_Req-Delay_Resp sequence of messages. In this case, a Delay_Req message (Seq ID 44687) has not had a Delay_Resp response. A batch of Sync messages is also sent by the Master with no response; this can give insight into the operation and interaction between Master and Slave devices.

For more information on the Calnex Paragon Sync, and to take advantage of Calnex's extensive experience in sync and packet testing technologies, please contact Calnex Solutions on +44 (0) 1506 671 416 or email: info@calnexsol.com

References

IEEE Instrumentation and Measurement Society, IEEE Std 1588™-2008 - Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems

Tan, Alexander E. (2007), IEEE 1588 Precision Time Protocol Time Synchronization Performance

Diamond, Pat. (2008), Synchronization for Future Mobile Networks

Calnex Solutions Ltd
Springfield, Linlithgow
West Lothian EH49 7NX
United Kingdom
tel: +44 (0) 1506 671 416
email: info@calnexsol.com

www.calnexsol.com

Calnex Solutions Ltd