PTP Background and Overview

by Jeff Laird, July 2012

Introduction

Precision Time Protocol (PTP) allows computers in a local area network to synchronize their clocks to within a microsecond of each other.

The standard was originally defined in 2002. The second and current version of the standard was published in 2008. It is known as "IEEE 1588-2008" (*IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*).

PTP compared to other synchronization methods

PTP is more accurate than NTP but less accurate than GPS.

NTP

NTP (Network Time Protocol) was standardized in 1985 and is the most common clock synchronization method, being used by tens of millions of client computers to synchronize to NTP servers over the Internet. NTP is primarily used in wide-area networks, where it can achieve accuracy to 10 ms. On a local-area network NTP can achieve **accuracy to 200 µs**.

NTP includes SNTP (Simple NTP), which is stateless (no averaging) and therefore suitable for embedded devices, but less accurate.

In NTP terminology, *stratum-0* clocks are atomic clocks, GPS satellites, and radio clocks not directly on a network. They provide the time to *stratum-1* clocks (network time servers) using a PPS (pulse-per-second) signal over an RS-232 serial connection. *Stratum-2* clocks typically synchronize with multiple stratum-1 clocks. Stratum-3 clocks synchronize with one or more stratum-2 clocks, and so on.

An NTP timestamp is 64 bits long—32 bits for seconds (to 136 years) and 32 bits for fraction of a second (to <0.25 ns). The NTP clock synchronization algorithm is intuitive: Tell each other your current time and subtract out the network delay. NTP uses only the UTC time scale.

GPS

GPS-disciplined clocks are the most accurate clocks outside of the atomic clocks used to create the international time scale TAI. They achieve **accuracy to 0.01 µs** (10 ns).

PTP

PTP fills a clock-synchronization need between NTP and GPS-based clock synchronization, with **accuracy to 1 µs**. There are two reasons to use PTP instead of GPS. One is that in a PTP network only one GPS receiver is needed, which is less expensive than giving each network node its own GPS receiver. The other is that in the extreme case of the GPS system becoming inoperable (for example during wartime) a PTP-synchronized network will stay synchronized. This is important for life-critical applications such as the electric power grid and telecommunication services.

	LAN	WAN
atomic clock	0.1-1 ns per day	
GPS (expensive)	10 ns	
PTP (inexpensive)	0.2 µs	10 µs
NTP	0.2 ms	10 ms

approximate accuracies

(Technically, the atomic clock unit of measure is not for *accuracy*, but rather for *skew*.)

IRIG

IRIG (Inter-Range Instrumentation Group) time codes are encoded timestamps used by the telecommunication industry since 1956. IRIG B uses 100-bit time codes transmitted once per second at a rate of 100 b/s. IRIG B, like GPS, is unidirectional. It can synchronize clocks to within 1 µs.

PTP, in addition to filling a niche between NTP and GPS, also fills a clock-synchronization need between IRIG and NTP. While IRIG is more accurate than NTP, it requires its own cables, connectors, and circuit hardware. PTP provides the accuracy advantage of IRIG together with the communication advantage of NTP because it works over existing Ethernet or IP networks.

Another advantage of PTP over both NTP and IRIG is that it has built-in support for redundancy and failover. If a PTP grandmaster node fails, the next-best node will automatically take over as grandmaster.

Synchronous Ethernet

Synchronous Ethernet (SyncE) is a competitor to PTP in telecommunication (i.e., carrier Ethernet), data centers, and audio-video bridging (AVB), but just for frequency synchronization, in contrast to PTP's focus on time-of-day synchronization. SyncE adds a timing signal to the Ethernet physical layer much like that used in telecommunication time-division multiplexing. Whereas PTP is supported by up to 1 GigE, SyncE is supported by up to 10 GigE. SyncE requires hardware replacement throughout the network and is therefore expensive, but it is unaffected by traffic load. PTP requires hardware replacement only at the NICs and is therefore cheaper, but it is degraded by high traffic load because it shares the main data stream.

White Rabbit

Taking clock synchronization to the next level, from microsecond accuracy to nanosecond accuracy, is the White Rabbit project, led by CERN starting in 2006. CERN is developing its next-generation control system with the goal of sub-nanosecond accuracy among thousands of nodes separated by as much as 10 km. The project is run as an open hardware project (ohwr.org). White Rabbit combines PTP with Synchronous Ethernet. Key papers were published in 2009 and 2011 (slide presentation for the latter).

Time scales

The five main time scales you are likely to encounter when dealing with PTP are the following.

TAI (*Temps Atomique International*, or International Atomic Time) – average of ~200 atomic clocks around the world. More stable than any one clock. Fundamental unit is the *second*. Not tied to astronomy. No leap seconds.

UT1 (*Universal Time*) – based on astronomy. Fundamental unit is the average solar *day*. The old definition of a second was 1/86,400th of this day. No leap seconds.

UTC (Coordinated Universal Time, or *temps universel coordonné*) – the standard for civil time around the world. A hybrid of TAI and UT1. Combines the stability of TAI with the astronomical correctness of UT1. Accomplishes this by the use of leap seconds. 35 seconds behind TAI as of 2012-Jul-1.

> **Why UTC?** The earth's rotation is not as stable as an atomic clock. The average solar day is now 2 ms longer than in 1820—the year for which the atomic second was defined. Multiply that 2 ms by 365 days and you find that the earth is three quarters of a second behind after a year's accumulation. This means that UT1 and TAI are diverging by almost a second per year (35 s since 1960). So which time scale do you choose for civil time? UTC is the compromise answer, gaining the benefits of both TAI and UT1 at the expense of added complexity in the form of leap seconds.

GPS (Global Positioning System) – always behind TAI by 19 seconds. No leap seconds.

PTP (Precision Time Protocol) – defined to be the same as TAI

See http://www.leapsecond.com/java/gpsclock.htm to see the relationships in real time.

Incidentally, there are hundreds of other time scales. For example, here are several more, roughly increasing in quality.

- **apparent solar time** – varies ±16 minutes
- **mean solar time**
- **sidereal time** – stellar, 23:56:04 day
- **Greenwich Mean Time (GMT)** – Obsolete, but the term is still used for UT/UT1.
- **Universal Time (UT)** – more accurate version of mean solar time

- **UT0** – location-based UT
- **UT1** – longitude-corrected UT0
- **UT2** – seasonally smoothed UT1, obsolete
- **UT1R** – tidally smoothed UT1
- **UT2R** – seasonally and tidally smoothed UT1
- **Ephemeris Time (ET)** – used 1952-76, now obsolete
- **Terrestrial Dynamical Time (TDT)** – atomic version of ET, now obsolete
- **Barycentric Dynamical Time (TDB)** – TDT with relativistic corrections, barycenter = center of mass of solar system
- **Terrestrial Time (TT)** – redefinition of TDT
- **Geocentric Coordinate Time (TCG)** – TT at center of earth
- **Barycentric Coordinate Time (TCB)** – slightly different from TT

The prosaic **civil times (e.g. standard or daylight)** deviate from some universal time such as UTC by round amounts such as hours.

Interestingly, some astronomical time standards begin their day at noon so that nighttime astronomical observations are not split across two days.

The PTP protocol

What it does

When the PTP protocol is active two activities continually occur on the network:

- Grandmaster-capable nodes negotiate to determine which will be the grandmaster.
- All nodes synchronize to the grandmaster.

The first activity is based on *Announce* messages, in which grandmaster-capable clocks broadcast their characteristics. Each clock on the network compares all of the Announce messages that it receives and, independently of the other clocks, determines the most suitable grandmaster. This determination is done through the *best master clock algorithm*, which by design ensures that all clocks will reach the same conclusion. This best master clock negotiation continues even after a grandmaster is selected so that in case the current grandmaster fails another clock can quickly replace it.

The second activity is based on *Sync* messages, which are broadcast by the grandmaster. These messages contain the time of day according to the grandmaster's internal clock. Since by the time the receiving node receives a Sync message the timestamp is already slightly obsolete, this second activity uses additional messages to compensate for the delay. (See *Network Delay Measurement* below.)

Device types

In PTP a **clock** is an entire network node that participates in the PTP protocol. A single clock can have multiple ports, each participating in PTP activities independently.

The PTP standard specifies the following five device types (in bold).

- **ordinary clock** – has a single PTP port, can be grandmaster-capable, can be slave-only
- **boundary clock** – has multiple PTP ports, synchronizes network segments
- transparent clocks – new to 1588-2008, modify timestamps in PTP messages
    - **end-to-end transparent clock** – residence times accumulated in the correctionField of Sync (one-step), Follow_Up (two-step), Delay_Req (one-step), or Delay_Resp (two-step) messages
    - **peer-to-peer transparent clock** – residence times plus egress-link delays accumulated in the correctionField of Sync (one-step) or Follow_Up (two-step) messages
- **management node**

Note the absence of "grandmaster clock", "master clock", and "slave clock" as device types. "Grandmaster" is a role filled by an ordinary clock. "Master" and "slave" are roles filled by a port on a clock.

Domains

A PTP **domain** is a network (or a portion of a network) within which PTP operates, i.e., a network within which all of the clocks are in sync. A single computer network can have multiple PTP domains operating separately, e.g., one set of clocks synchronized to one time scale and another set of clocks synchronized to another time scale. PTP can run over either bare Ethernet or UDP/IP, so a domain can correspond to a local area network or it can extend across a wide area network.

Data sets

A PTP clock maintains several data sets. Most data sets are per-clock. The exceptions are portDS, foreignMasterDS, and transparentClockPortDS, which are per-port.

Each data set member is either static, *dynamic*, or configurable. Highlighted members appear in both ordinary/boundary clocks and transparent clocks.

Each **ordinary or boundary clock** MUST have the following data sets.

- **defaultDS** (twoStepFlag, clockIdentity, numberPorts, *clockQuality* (comprises *clockClass*, *clockAccuracy*, *offsetScaledLogVariance*), priority1, priority2, domainNumber, slaveOnly)
- **currentDS** (*stepsRemoved*, *offsetFromMaster*, *meanPathDelay*)
- **parentDS** (*parentPortIdentity*, *parentStats*, *observedParentOffsetScaledLogVariance*, *observedParentClockPhaseChangeRate*, *grandmasterIdentity*, *grandmasterClockQuality*, *grandmasterPriority1*, *grandmasterPriority2*)
- **timePropertiesDS** (*currentUtcOffset*, *currentUtcOffsetValid*, *leap59*, *leap61*, *timeTraceable*, *frequencyTraceable*, *ptpTimescale*, *timeSource*)
- **portDS** (one data set per port) (portIdentity, *portState*, *logMinDelayReqInterval*, *peerMeanPathDelay*, logAnnounceInterval, announceReceiptTimeout, logSyncInterval, delayMechanism, logMinPdelayReqInterval, versionNumber)
- **foreignMasterDS[5+]** (one data set (array) per port, implementation-specific) (*foreignMasterPortIdentity*, *foreignMasterAnnounceMessages*)

Each **transparent clock** SHOULD have the following data sets.

- **transparentClockDefaultDS** (clockIdentity, numberPorts, delayMechanism, primaryDomain)
- **transparentClockPortDS** (one data set per port) (portIdentity, *logMinPdelayReqInterval*, *faultyFlag*, *peerMeanPathDelay*)

A defaultDS.clockIdentity is a data structure of type **ClockIdentity** (Notice the capital 'C'.), which is an 8-byte field typically containing the IEEE EUI-64 **e**xtended **u**nique **i**dentifier created from the clock's MAC address by inserting the hex value 0xfffe in the middle. For example, a clock with MAC address `00:11:22:33:44:55` will probably have defaultDS.clockIdentity `0x001122fffe334455`.

A portDS.portIdentity is a data structure of type **PortIdentity** (Notice the capital 'P'.), which is a 10-byte structure comprising a ClockIdentity called *clockIdentity* and a UInteger16 called *portNumber* (e.g. `0x001122fffe3344550001`). The sourcePortIdentity field in every PTP message header is a PortIdentity.

Message types

Event messages are time-critical. **General** messages are not. The event message types are `Sync`, `Delay_Req`, `Pdelay_Req`, and `Pdelay_Resp`.

hex	message type
00	Sync
01	Delay_Req
02	Pdelay_Req
03	Pdelay_Resp
08	Follow_Up
09	Delay_Resp
0a	Pdelay_Resp_Follow_Up
0b	Announce
0c	Signaling
0d	Management

PTP message types

Communication paths

At its simplest, a PTP **communication path** is a single network cable, with a master port at one end and a slave port at the other. However, by introducing switches (transparent clocks) in the middle, a communication path can branch from a single master to multiple slave ports. Transparent clock ports do not have master or slave states, so the terms "master port" and "slave port" apply only to ports at the ends of a communication path.

Although a communication path can have switches in the middle, it cannot have boundary clocks in the middle. A boundary clock terminates a communication path. For example, if a grandmaster clock communicates with a boundary clock and that boundary clock communicates with a slave-only clock then there are two communication paths involved, one on either side of the boundary clock. In this case no PTP messages travel from the grandmaster to the slave-only clock. The slave-only clock sees PTP messages only from the boundary clock in the middle.

Network delay measurement

Synchronizing clocks demands knowledge of the communication delay between the clocks. Such communication delay is also called "path delay" or, in a network environment, "network delay". The PTP standard uses the term **delay mechanism** to mean "network-delay measurement mechanism". This document uses the latter phrase (or just "delay measurement mechanism") for clarity. The network delay of interest is that between two ports on the same communication path (not between a grandmaster clock and a more distant clock on the other side of a boundary clock).

IEEE 1588-2002 defined only one network-delay measurement mechanism: the **end-to-end delay measurement mechanism**. The standard also calls this the "delay request-response mechanism". This delay measurement mechanism uses message types Sync, Follow-up (optional), Delay_Req, and Delay_Resp. It measures the network delay from a master port to a slave port on the same communication path, through all intervening switches (transparent clocks).

If a PTP-enabled switch in the middle of a communication path is an *end-to-end transparent clock* then it treats non-PTP messages and most PTP messages the way normal switches treat them, but it treats PTP timing messages specially, adding the switch residence time to their `correctionField`. End-to-end delay measurement concentrates PTP load on the master port and so is not scalable as the number of switches and slave ports in a communication path increases.

IEEE 1588-2008 introduced a new network-delay measurement mechanism: the **peer-to-peer delay measurement mechanism**. The standard calls this the "peer-to-peer delay mechanism" or just the "peer delay mechanism". This delay measurement mechanism uses message types Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up (optional). It measures the network delay between directly connected ports, with no intervening switch. For example two peers might be a port on an ordinary clock and a port on a transparent clock. Unlike end-to-end delay measurement, peer-to-peer delay measurement does not concentrate load on the master port and so scales well as the number of switches and slave ports in a communication path increases.

Peer-to-peer delay measurement works "locally", from every node to each of its neighbors. For each pair of directly connected ports, each end periodically measures the round-trip network delay by sending a Pdelay_Req message and receiving in response a Pdelay_Resp message. This way all ports learn the delay on their link. If a PTP-enabled switch in the middle of a communication path is a *peer-to-peer transparent clock* then it treats non-PTP and PTP messages the same as the end-to-end transparent clock described above does, except that in addition to adding the switch residence time it also adds the

egress-link delay to the `correctionField`, and it does this to only one message type: Sync (in the case of one-step ports) or Follow_Up (in the case of two-step ports).

> In a communication path containing no transparent clocks and therefore only a single link, peer-to-peer delay measurement is essentially the same as end-to-end delay measurement. In other words, compensating for a chain of link delays individually and compensating for a chain of link delays in aggregate are equivalent when there is just one link in the chain.

An **ordinary clock**, which necessarily has just one port, can be configured for end-to-end or for peer-to-peer delay measurement. A **boundary clock**, which can have a single slave port and multiple master ports, can be configured for end-to-end delay measurement on some ports and peer-to-peer delay measurement on other ports. An **end-to-end transparent clock** is connected only to end-to-end ports and so participates in only end-to-end delay measurement. A **peer-to-peer transparent clock** is connected only to peer-to-peer ports and so participates in only peer-to-peer delay measurement.

While a single domain can have both end-to-end and peer delay mechanisms running simultaneously, each communication path in the domain must run only one or the other. An end-to-end communication path can have more than two ports, but a peer-to-peer communication path, also known as a **link**, involves just two ports.

Profiles

The IEEE 1588 standard provides for custom specification of configuration options in the form of *profiles*. This allows other standards organizations to use IEEE 1588 as a general foundation for more specific clock synchronization standards. In each of the profile sections below, several configuration options are listed.

Default profiles

The IEEE 1588 standard includes two default profiles, one for each of the delay measurement mechanisms.

Delay Request-Response profile (corresponds to end-to-end delay measurement)

domain 0, Announce interval 1 (range 0–4), Sync interval 0 (-1 to 1), Delay_Req interval 0 (0–5), Announce timeout 3 (2–10), Priority1 128, Priority2 128.

path delay measurement mechanism: Default is delay request-response (e2e). Peer delay is also allowed. Limited to a single mechanism per communication path.

Peer-to-Peer profile (corresponds to peer-to-peer delay measurement)

Same as above except that the path delay measurement mechanism default is peer delay (with delay request-response allowed). Additionally, Pdelay_Req interval 0 (0-5).

Power profile (IEEE PC37.238)

For use within and between potentially widely separated power substations.

transport: layer-2, multicast

domain 0, Announce interval 0, Sync interval 0, Pdelay_Req interval 0, Announce timeout 3 (2 for preferred grandmasters), Priority1 128, Priority2 128 (255 for slave-only clocks).

path delay measurement mechanism: only peer delay (p2p)

one-step recommended

Telecommunication profile (ITU G.8265.1)

transport: layer-3, unicast

domain: 4, Announce timeout 2

path delay measurement mechanism: delay request-response (e2e)

gPTP default (IEEE 802.1AS)

This is not actually a 1588 profile, but 802.1AS, being based on IEEE 1588, is comparable to a 1588 profile.

transport: layer-2, multicast (unicast in WLANs)

Pdelay_Req interval 0

path delay measurement mechanism: only peer delay (p2p)

two-step