# Precision Time Protocol

**Calculating network latency**

By Paul Krzyzanowski
*October 6, 2017*

## Precision Time Protocol

The Precision Time Protocol, PTP, is designed to synchronize clocks on a local area network (LAN) to sub-microsecond precision. The underlying assumption is that the accurate time source is on the same LAN as the systems that are synchronizing their clocks. That differs from NTP, which assumes that the time server is remote. An advantage of synchronizing via the LAN is that latency becomes far more predictable. There are no queuing delays due to routers (recall that routers have to store an entire message before forwarding it to the next router) and physical distances tend to be far smaller. Ethernet switches do create some latency, of course, and there is a chance that a packet may be queued if a destination is receiving other messages. Switch latency, however, tend to be on the order of a few microseconds and queuing can be nonexistent if there isn't much traffic.

To further minimize delay and jitter, high-precision PTP implementations will try to generate timestamps at the very lowest levels of the network stack such as at the MAC layer (Ethernet transceiver) right before sending the packet out.

### Best master clock

In a network of computers running PTP, one system has to be elected as the **master clock**. This system is deemed to have the most accurate clock and other systems are **slaves** that synchronize from the master.

A **best master clock** selection process is used to determine which system has the best clock to use for synchronization. This is done via an election where systems present information about their clocks and the best clock is selected from the following ranked attributes:

1. Priority 1 (an admin-defined hint, allowing the administrator to force the use of a certain system as the master)
2. Clock class (type of clock)
3. Clock accuracy
4. Clock variance: an estimate of stability based on past syncs
5. Priority 2 (another admin-defined hint, allowing the preference of one system over another if all other values are equivalent)
6. Unique ID (serves as a tie-breaker on the chance that all other values are equivalent among systems)

### PTP messages

PTP synchronizes a clock by exchanging three messages:

1. The master initiates the sync by sending a **sync** message.
2. The slave sends a **delay request** message back to the master.
3. The master responds with a **delay response** message.

A *delay request* message isn't quite what it sounds: it is not a query asking the master what the network delay is. The master has no clue. Rather, it is an additional message that allows the slave to get an accurate estimate of the network delay.
Let's
look at **PTP synchronization**
an example. We'll use hours and minutes instead of microseconds to make the numbers more tangible and feel like time values.

In our example, the clocks on the two systems are not synchronized. In the figure, we note the times for each event with each system's local clock.

### *Sync message*

The master initiates synchronization at $T_1$, which is 10:00 at the master, by sending a **sync** message to the slave. That time happens to be 8:00 on the slave's clock. The slave receives the message at time $T_2$, which is 8:15 on the slave's clock. It happens to be 10:15 on the master's clock but the slave doesn't know that.

The slave now has $T_1$ from the master (10:00) and knows its value of $T_2$ (8:15). The difference between them (ms_difference) is the clock offset plus the delay of sending the message from the master to the slave. We can express this as:

ms_difference = $T_2$ - $T_1$ = offset + ms_delay

In our example, we know that the clock offset is –2:00 (the client is behind by two hours) and the propagation time, ms_delay, is 0:15. The value of offset + ms_delay is –2:00 + 0:15, or –1:45. The slave does not know the delay or offset yet. All it can compute is

ms_difference = $T_2$ - $T_1$ = 8:15 - 10:00 = -1:45

### *Delay request and response messages*

Now the slave sends a **delay request** message back to the master. This message simply asks the master to tell the slave what time it received this message. Suppose the slave sends the message at $T_3$ (8:45) and the master receives it at $T_4$ (10:55). The master sends back the value of $T_4$ (10:55).

The math is similar to that of the sync message. The slave has $T_4$ from the master (10:55) and knows its value of $T_3$ (8:45). The difference between them (sm_difference) is the *negative* clock offset plus the delay of sending the message from the master to the slave. The offset is negative because the message went in the opposite direction. We can express this as:

sm_difference = $T_4$ - $T_3$ = -offset + sm_delay

Again, let's look at the numbers. The clock offset is still –2:00. The message propagation time here was now 0:10. The value of -offset + sm_delay is 2:00 + 0:10 = 2:10. Again, the slave does not know the delay or offset but it can compute:

sm_difference = $T_4$ - $T_3$ = 10:55 - 8:45 = 2:10

### *Two equations, three unknowns*

With these messages, the client has two equations and three unknowns to work with:

ms_difference: $T_2 - T_1 = \text{offset} + \text{ms\_delay}$
sm_difference: $T_4 - T_3 = -\text{offset} + \text{sm\_delay}$

That isn't solvable, of course. However, LANs have symmetric bandwidth and we can assume that the latency of sending a message from the master to the slave is the same as that of sending a message from the slave to the master. There will be variations due to process and packet scheduling as well as possible queuing delays in the Ethernet switch, but these will usually be small - a few microseconds.

Hence, if we assume that ms_delay and sm_delay are equivalent, we have:

ms_difference: $T_2 - T_1 = \text{offset} + \text{delay}$
sm_difference: $T_4 - T_3 = -\text{offset} + \text{delay}$

Now, with two equations and two unknowns, we can solve for the offset:

$\text{offset} = (\text{ms\_difference} - \text{sm\_difference}) \div 2 = ((T_2 - T_1) - (T_4 - T_3)) \div 2$
$\text{offset} = (T_2 - T_1 - T_4 + T_3) \div 2$

In our example, with slightly asymmetric delays, we get:

offset = 8:15 - 10:00 - 10:55 + 10:45 = -1:45 - 10:55 + 10:45 = -12:40 + 10:45 = -1:55

The way the PTP equations were set up, the offset is subtracted from the current time, so the slave will have to decrease its clock by –1:55, or advance it by 1:55. With the asymmetric delay, PTP gave averaged out the message latencies, resulting in an offset of –1:55 instead of the correct value of –2:00.

We can also compute the one-way message delay:

$\text{delay} = (\text{ms\_difference} + \text{sm\_difference}) \div 2 = ((T_2 - T_1) + (T_4 - T_3)) \div 2$
$\text{offset} = (T_2 - T_1 + T_4 - T_3) \div 2$

## References

- Texas Instruments, AN–1728 IEEE 1588 Precision Time Protocol Time Synchronization Performance, Application Report, April 2013
- National Instruments, Special Focus: Understanding the IEEE 1588 Precision Time Protocol, Nov 06, 2015
- RTA IEEE 1588 Unplugged – An introduction to IEEE 1588
- Precision Time Protocol, Wikipedia
- Ken Ichikawa (Fujitsu Limited), Precision Time Protocol on Linux, LinuxCon, Japan 2014.