Overview of PTP for High-Precision Time Synchronization and IIJ's Problem Solving—RPTP

2.1 Introduction

You may have heard of the time synchronization protocol called Precision Time Protocol (PTP). PTP has attracted attention in recent years as a high-precision time synchronization technology. This article first outlines PTP, then looks at related challenges on real-world networks, and finally introduces RPTP, an approach to solving those challenges.

The most widely used protocol for time synchronization on IP networks is NTP (Network Time Protocol). NTP is a hierarchical client/server model designed for practical use even on networks with delay variation and packet loss. These days, NTP is often enabled in operating systems by default, so users benefit from it without even being aware of it. Without time synchronization, the ordinary crystal oscillator built into a PC can drift by several tens of seconds to several minutes over the course of a month. For everyday operation of PCs and servers, NTP-based time synchronization is almost never a problem.

2.2 Why PTP Now?

The number of systems built on the assumption that a common time reference can be shared across systems with high precision has risen in recent years. Examples include mobile communications systems such as cellular networks, smart grids in the area of electric power systems, and financial systems used for high-frequency trading. These systems require strict synchronization both internally and with external systems. Insufficient time or synchronization accuracy can result in problems such as failure of communications control, control signal malfunctions, and data inconsistencies.

Table 1 shows examples of required time synchronization accuracy in different industries.

These systems require accuracy on the order of microseconds to nanoseconds. That's more than three orders of magnitude tighter than the accuracy typically provided by NTP (which, on the Internet in general, is in the millisecond range). It was against this backdrop of demand for high-precision time synchronization that the IEEE published the PTP (Precision Time Protocol, IEEE 1588) standard.

|Sidebar 1|

How do smartphones keep time?

Smartphones synchronize their clocks using radio signals from cellular base stations. This mechanism was standardized back in the 3G era. Base stations obtain time from GNSS (discussed below) or from network-based synchronization protocols such as PTP. Smartphone OSes also make use of NTP.

Table 2: Units of Time

1 second (s)	1 second	10^0
1 millisecond (ms)	One thousandth of a second	10^-3
1 microsecond (μs)	One millionth of a second	10^-6
1 nanosecond (ns)	One billionth of a second	10^-9
1 picosecond (ps)	One trillionth of a second	10^-12

Table 1: Examples of Time-Sync Accuracy Requirements by Industry

Industry	Example applications	Required accuracy
Electric power systems	Smart grids etc.	Microseconds to several tens of microseconds
Telecommunications	Mobile networks (LTE, 5G)	Nanoseconds to microseconds (100 ns level for 5G fronthaul)
Databases	DB synchronization, auditing	Microseconds to milliseconds
Finance	High-frequency trading, auditing	Microseconds to milliseconds (transaction timestamps are at the microsecond level)
Factory automation	Control, measurement	Microseconds to milliseconds (for high-speed control and measurement, microsecond-level accuracy is required)
Broadcast media	Media over IP	Microseconds to milliseconds (for video frame sync, several to several hundred microseconds; for OFDM modulation, several hundred nanoseconds)
Science and technology	VLBI, particle accelerators, etc.	Picoseconds to nanoseconds (VLBI and accelerators require sub-nanosecond precision)

2.3 What is PTP?

PTP was standardized by the IEEE for the purpose of synchronizing real-time clocks over a network. It can be used over IPv4, IPv6, and IEEE 802.3 Ethernet. It was first standardized in 2002 as IEEE 1588-2002 (PTPv1), and PTPv2 was later defined in IEEE 1588-2008. PTPv2 is not compatible with PTPv1. IEEE 1588-2008 was revised again in 2019 as IEEE 1588-2019, which is sometimes informally referred to as PTPv2.1.

PTP supports synchronization accuracy at the sub-microsecond level (i.e., better than one microsecond). Some extended profiles (e.g., White Rabbit) can achieve sub-nanosecond precision.

With NTP, time is distributed in a hierarchical structure made up of strata. PTP, by contrast, works by having each node (PTP instance) select the best clock using an algorithm called BMCA. Time is then synchronized from the master, which has a higher-precision clock, to the slave, whose clock requires correction. PTP instances are designed to autonomously form a time synchronization system among themselves.

In this article, I follow the “master” and “slave” terminology used in the standard, though recently “leader” and “follower” have become more common.

2.4 PTP Profiles and Variations

PTP uses IEEE 1588 as its base specification, and defines profiles optimized for different applications. The profiles specify details such as message type, communication method, and required accuracy. It is important to note that the profile parameters differ. Table 3 and Table 13 show key examples.

2.5 Basic Structure of PTP

I now describe the structure of PTP and several of its characteristic algorithms. The discussion from this section onward is based on IEEE 1588-2019.

2.5.1 Communication Method

To make messages reach the whole network, PTP uses IP multicast or a dedicated multicast MAC address. IANA has thus assigned 224.0.1.129 (IPv4) and ff0x::181 (IPv6, where *x* indicates scope), along with port 319 (PTP event messages) and port 320 (PTP general messages). For profiles that communicate directly over Ethernet, IEEE Registration has also assigned the MAC address 01-1B-19-00-00-00. Some profiles use other MAC addresses as well.

2.5.2 Domain

IPTP has the concept of a domain. A domain is identified by a number, and the system administrator chooses one in the range from 0 to 255 (recommended values differ by

Table 3: Key PTP Profiles

Standards body / industry	Profile name	Main applications	Features
IEEE	Default Profile	General-purpose PTP	Default IEEE 1588 profile; operates over UDP/IP or Ethernet; uses the Delay_Req/Resp mechanism.
IEEE	IEEE 802.1AS (gPTP)	TSN, AVB	Time-synchronized Ethernet control; simplified BMCA; peer-to-peer delay measurement; Layer 2 operation.
ITU-T	G.8275.1	Telecommunications (fully PTP-compatible networks)	For mobile networks; all nodes PTP-aware; TCs mandatory; GNSS-referenced GM assumed.
ITU-T	G.8275.2	Telecommunications (partially PTP-compatible networks)	Can operate even with some non-PTP-aware nodes; extensive use of BCs.
ITU-T	G.8275.5	5G fronthaul	Optimized for phase/time synchronization; stringent accuracy requirements (around ±100 ns).
SMPTE	ST 2059-2	Broadcasting and video	For Media over IP (ST 2110). Video frame synchronization, black burst replacement.
AES	AES67	Audio IP transmission	Audio synchronization for broadcast and professional audio; interoperable with SMPTE ST 2059; related to ST 2110-30.
IEC	Power Profile (IEC 61850-9-2)	Power	For substations and protective relays; reliable sub-microsecond synchronization required.
IEEE	C37.238 (Power Profile)	Power	PTP for power systems; emphasis on UTC traceability.
IEEE	High Accuracy Profile	Science and technology	Derived from White Rabbit; sub-nanosecond accuracy; PTP + SyncE + phase correction.
Avnu Alliance	Automotive Profile	Automotive	Time and trigger synchronization over automotive Ethernet.
ODVA	CIP Sync	Factory automation	Industrial control synchronization over EtherNet/IP.

profile, and some applications configure them automatically). Multiple domains can coexist on a single network. In such cases, synchronization does not occur automatically across domains. When PTP operates over IP multicast, there is no need to separate multicast groups or port numbers according to the configured domain number. Messages for multiple domains flow together on the same multicast group.

2.5.3 PTP Instance Types

PTP instances are classified into the four types in Table 4.

There is also a PTP Management Node, defined separately for management purposes.

Each PTP instance maintains a state for each of its ports. Figure 1 shows a typical relationship between PTP instances and PTP ports.

2.6 PTP's Distinctive Best Master Clock Algorithm (BMCA)

BMCA is a distinctive mechanism of PTP. A PTP instance monitors the state of its own PTP ports, and based on the contents of the Announce messages it receives, it performs state transitions on those ports and selects the best Master Clock. In PTP, there is no need to manually configure who will be master and who will be slave. BMCA is not designed for centralized control over the whole network but as a distributed algorithm in which each PTP port performs state transitions locally.

An Announce message contains information about the sending PTP instance, and the receiver uses that information to choose the best clock from among multiple candidates. Table 5 gives the selection algorithm (criteria) in order.

Table 4: PTP Instance Types

Grandmaster	PTP GM	The reference clock within the domain
Boundary Clock	PTP BC	Has multiple ports and operates as both master and slave
Transparent Clock	PTP TC	A relay node that performs delay correction
Ordinary Clock	PTP OC	An endpoint clock with a single port

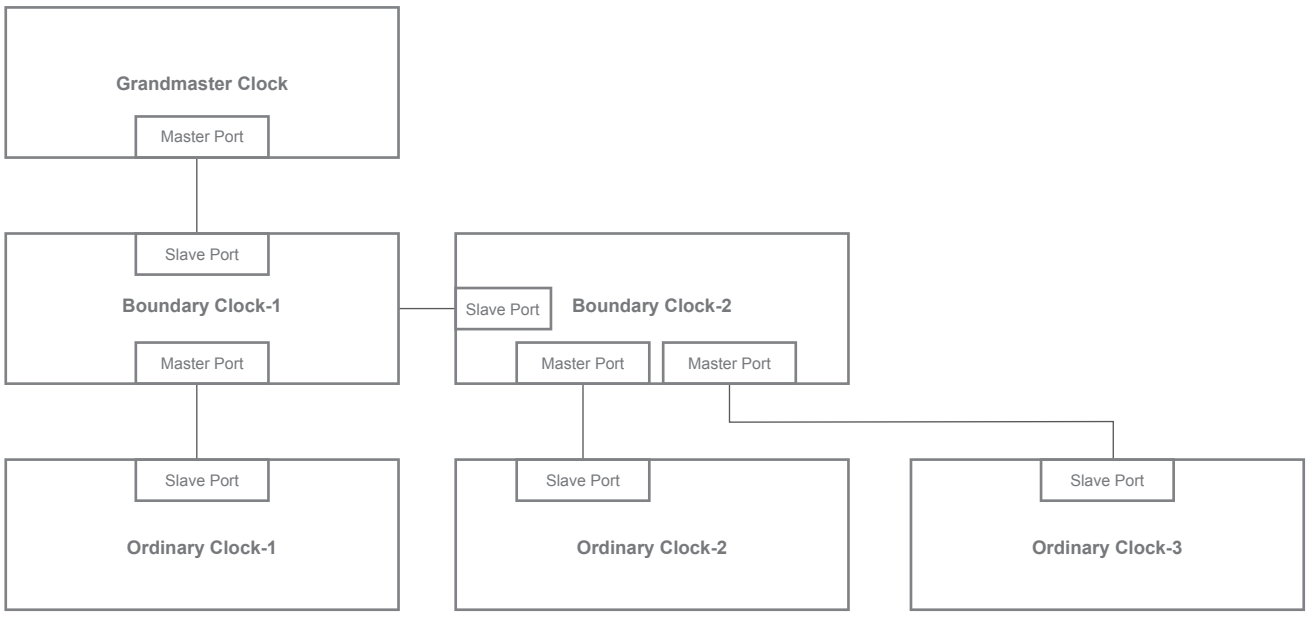

Figure 1: Typical Configuration of PTP Instances and PTP Ports

A common way to generate clockIdentity is from an EUI-48 MAC address.

As Table 6 shows, this is typically done by inserting FF-FE between the front half of the MAC address, the OUI (Organizationally Unique Identifier), and the back half, the EI (Extension Identifier).

Announce messages are sent only by PTP instances with PTP ports in the master state (by a PTP GM or PTP BC). If the design uses PTP GM redundancy, each PTP GM determines for itself which GM is best according to the algorithm above and changes its own state accordingly. Unselected PTP GMs stop sending Announce messages. If Announce messages from the originally selected PTP GM stop flowing due to failure or a link disconnection, the unselected PTP GMs go through the BMCA process and one of them starts sending Announce messages. This is how switchover is achieved in PTP.

A PTP BC normally synchronizes with the PTP GM and sends Announce messages to other PTP instances. If it loses synchronization with the PTP GM, however, it updates various parameters in the Announce message and notifies downstream nodes that it is now providing synchronization based on its own clock. For example, the grandmasterIdentity, which had previously been set to the upstream PTP GM's value, is rewritten to its own clockIdentity, and clockClass is changed to a value indicating the Holdover state. In this state (Holdover), the device continues to provide time synchronization using its own clock based on the information it had just before synchronization was lost. PTP devices often use high-accuracy clocks to maintain holdover accuracy.

Table 6: Example of clockIdentity Generation

Source MAC address	00-00-5E-00-53-00
clockIdentity generated from the MAC address	00-00-5E-**FF-FE**-00-53-00

Table 5: Information Evaluated by the BMCA

1	grandmasterPriority1	Priority 1 of the relevant GM. In many implementations, this is manually configured.
2	grandmasterClockQuality.clockClass	Indicates the traceability of the time or frequency distributed by the GM, its synchronization state, and its clock class. Examples include 6 (GNSS-synchronized GM) and 248 (free-running).
3	grandmasterClockQuality.clockAccuracy	Indicates the expected clock accuracy. This is expressed as a category value representing the upper bound of the expected time error. Lower values are better.
4	grandmasterClockQuality.offsetScaledLogVariance	Indicates the estimated frequency stability of the GM's local clock. Lower values are better.
5	grandmasterPriority2	Priority 2 of the relevant GM. In many implementations, this is manually configured.
6	grandmasterIdentity	The clockIdentity of the GM serving as the time source. clockIdentity is used as an identifier.

|Sidebar 2|

Do PTP operators remember MAC addresses?

During operation, it is important to confirm which PTP GM a given device is referencing. In practice, that means operators end up looking at grandmasterIdentity values over and over in command output or a web UI. As a result, PTP operators naturally start memorizing OUIs—"That address looks like Cisco," or "Seiko Solutions was `00-80-15`, right?" and so on.

2.7 PTP Time Synchronization Algorithm

Here I explain the relationship between the Grandmaster Clock and the Local Clock. If the Grandmaster Clock and the Local Clock were perfectly synchronized, there would be no need for correction. But under that assumption, every clock would need to be highly accurate and therefore expensive. And in any case, it is impossible for free-running clocks to remain perfectly synchronized on their own. One option would be to use GNSS as the time source for every clock, but that brings wiring and cost issues. In practice, the usual arrangement is to combine a more accurate (and correspondingly expensive) Grandmaster Clock with lower-cost local PTP clocks that cannot guarantee accuracy on their own, and then generate the slave-side Local Clock from that. In this setup, the Grandmaster Clock needs to have atomic-clock-class accuracy or similar, while the Local PTP Clock constantly needs to be corrected using the Grandmaster Clock as its time source.

Let's now go through the synchronization method itself. There are two variations: one-step (1-step) and two-step

(2-step). Table 7 and Figure 2 explain the 2-step method as an example.

By calculating t2 - t1, we can determine the forward-path time from when the master PTP instance sends the message until the slave PTP instance receives it. Here, t2 is the result of adding network delay (meanPathDelay) and the time difference between the two clocks (offsetFromMaster) to t1. So,

$$t2 = t1 + meanPathDelay + offsetFromMaster$$

which gives

$$(1)\ offsetFromMaster = (t2 - t1) - meanPathDelay$$

And calculating t4 - t3 gives us the reverse-path time from when the slave PTP instance sends the message until the master PTP instance receives it. So,

$$t4 = t3 + meanPathDelay - offsetFromMaster$$

Here, offsetFromMaster is defined as the time difference slave - master, so its sign flips on the return path. This yields

$$(2)\ offsetFromMaster = (t3 - t4) + meanPathDelay$$

Further, from Equations (1) and (2), we obtain

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

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

The slave PTP clock uses these two values to synchronize itself.

As you can see from Equations (1) and (2), a one-way measurement alone mixes together network delay and clock offset. The two can be separated by measuring both directions and plugging the results into the equations.

By continuously calculating these two values (meanPathDelay and offsetFromMaster), the slave PTP instance synchronizes its own clock (Local PTP Clock) with the master PTP instance's clock (Grandmaster Clock). This method makes it possible to synchronize while taking network delay into account. Depending on the profile, this calculation cycle can be configured to run anywhere from 0.5 times to 128 times per second.

Now, why does the master PTP instance go to the trouble of sending Sync and Follow_Up separately? This is a key mechanism for achieving the accuracy PTP requires. The master PTP instance must do two things: send a Sync

Table 7: 2-Step Sequence

1	The master PTP instance sends a Sync message to the slave PTP instance and records the send time t1.
2	The slave PTP instance receives the Sync message and records the reception time t2.
3	The master PTP instance sends a Follow_Up message containing the time t1 to the slave PTP instance.
4	The slave PTP instance receives the above message. From the Sync and Follow_Up messages, it obtains the two timestamps t1 and t2.
5	The slave PTP instance sends a Delay_Req message to the master PTP instance and records the send time t3.
6	The master PTP instance receives the Delay_Req message and records the reception time t4.
7	The master PTP instance sends a Delay_Resp message containing the time t4 to the slave PTP instance.
8	The slave PTP instance receives the Delay_Resp message. From the Delay_Req and Delay_Resp messages, it obtains the two timestamps t3 and t4.

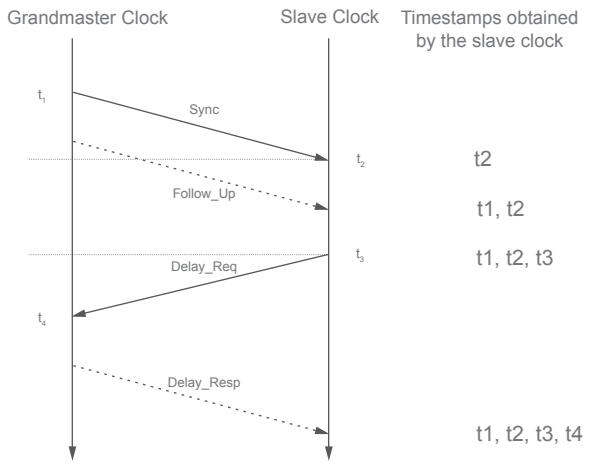

Figure 2: 2-step Sequence

message, and accurately measure and record the moment (t1) when that message actually goes out onto the network (timestamping).

The Sync message is generated in the upper layers, but the exact moment (t1) it will actually be transmitted onto the network is undetermined at that point. If timestamping were performed in the upper layers, the internal processing delay would be added to t1, which would introduce an error into the value t2 - t1. To avoid this, PTP defines a method whereby the actual transmission time is obtained as a timestamp once the Sync message is sent and then reported separately in a Follow_Up message. This is what is known as the 2-step sequence.

If hardware timestamping is supported, timestamping can be performed either immediately before the message is transmitted onto the network or at the instant of transmission. This enables higher synchronization accuracy.

To achieve high-precision time synchronization, implementations that use hardware timestamps at the MAC layer or PHY layer are often adopted in practice. PTP-capable equipment generally uses dedicated hardware or a PTP-capable NIC.

With the 1-step approach, when a Sync message is generated and sent out onto the network, the hardware inserts the precise transmission timestamp directly into the frame immediately before transmission, and this happens the moment the actual transmission time is determined. Because the Sync message alone can then carry the correct transmission time (t1), a Follow_Up message is unnecessary. This is illustrated in Figure 3.

2.8 PTP Networking

PTP provides high-precision time synchronization, but the network also needs to be set up to facilitate that performance. Ideally, every device along the path between PTP instances should itself support PTP. A network built this way is called a PTP-aware network. If a device that does not support PTP exists between PTP instances, internal processing and buffering in that device create packet jitter (in PTP terminology, Packet Delay Variation), which makes it difficult to maintain accurate synchronization between the PTP instances. Such networks are customarily referred to as PTP-unaware networks.

Figure 4 shows a typical PTP-aware network configuration. It is also possible to connect a PTP GM directly to a PTP OC and synchronize them that way. But when multiple PTP OCs are present, the network is usually built using PTP BCs (Boundary Clocks) and PTP TCs (Transparent Clocks). PTP BCs can be arranged in multiple hierarchical stages, and PTP TCs can also be used in multiple stages to successively correct for packet forwarding delay.

Each PTP instance exchanges synchronization information over the network. PTP BCs are placed in the network to regenerate time and distribute it downstream, and PTP TCs are placed in the network to correct forwarding delay. Hardware timestamping as described above is often used on both PTP BCs and PTP TCs to ensure accurate synchronization.

It bears mentioning that PTP packets are treated as special by PTP BCs and PTP TCs, which means they are handled differently from other packets.

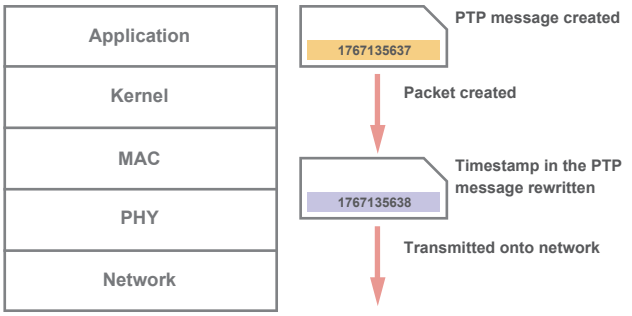

Figure 3: Layer Structure in PTP and Behavior of a PTP-capable NIC (1-step)

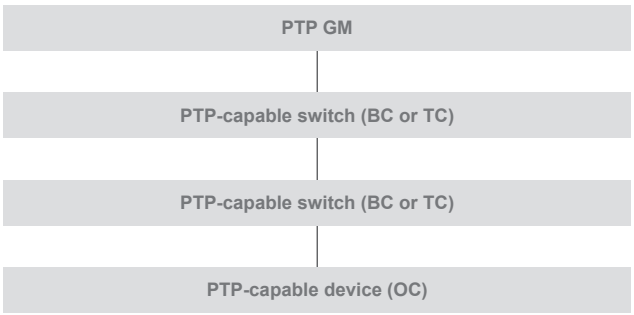

Figure 4: A Typical PTP-Aware Network

In a PTP BC, the local PTP clock is maintained based on the received synchronization information. In other words, the PTP BC regenerates the time internally. Based on that time information, synchronization is then provided to downstream devices. In this case, the PTP packet received on the ingress port and the PTP packet sent from the egress port are not the same.

Meanwhile, PTP TCs, as the name (Transparent) suggests, let PTP packets pass between ports. But when a PTP TC does so, it measures the time spent processing the packet inside the PTP TC device. It then adds that value to the CorrectionField (which records delay time) before forwarding the packet. Receiving PTP instances refer to the CorrectionField value and use it to correct for the delay.

From downstream of a PTP BC or PTP TC, it may look as though PTP packets are simply being sent as is, but in reality, accurate time information is being regenerated or corrected by the PTP BC or PTP TC. This correction process is extremely important for achieving high-precision synchronization on a PTP network.

By contrast, an L2 switch that does not support PTP just forwards packets as is. No timestamp correction is performed. So if you want to achieve the high-precision synchronization performance of PTP, the points in Table 8 must be kept in mind when building the network.

|Sidebar 3|

A caution on the operation and behavior of PTP-capable switches

Many PTP-capable switches let you turn PTP on or off on a per-port basis. So you need to be aware of which ports will be used for PTP from the start. Because PTP packets are handled differently from ordinary traffic, you need to be mindful of PTP-specific implementation details. Also, PTP-capable switches generally can only participate in one PTP domain.

2.9 The Symmetry and Stability Assumption and Associated Challenges

PTP basically assumes the network will be stable. The way meanPathDelay is calculated (add and divide by two) reflects this. In other words, it assumes the outbound and return packet travel times are the same. This is called network symmetry. But under that assumption, it is difficult to achieve the accuracy expected from PTP on an unstable network.

In practice, keeping packet transmission time symmetric can be hard in some cases, such as with shared networks like the Internet, on which the behavior of many nodes is not synchronized, and with video transport traffic, which is characterized by directional asymmetry (more traffic in one direction than the other). Figure 5 shows a typical pattern for a network that combines video transport and PTP transport.

From left to right (forward path), traffic flows from Stream TX to Stream RX and from PTP GM to PTP OC. Traffic flowing from right to left (reverse path) consists only of the PTP OC's messages to the PTP GM.

In general, video transport produces far more traffic than PTP messages (typically on the order of thousands of times more). Hence, traffic volume on the forward path outweighs that on the reverse path, so there is a high load on the forward path. As a result, processing time on intermediate devices is skewed, and transmission delays are asymmetric.

Traffic volume is also in constant flux on public networks like the Internet. There is always jitter. And the frequency and magnitude of the fluctuations take different forms depending on the network. Here are two examples of fluctuations on public networks: jitter measured on the IIJ backbone (Figure 6) and on IIJ's FLET'S connection service between Tokyo and Osaka (Figure 7).

Table 8: Key Points When Configuring a PTP-Aware Network

1	Use the same PTP version across the system to ensure compatibility
2	If you want to synchronize multiple devices (OCs) via PTP, obtain switches that support PTP BC/TC
3	Assume that PTP instances should be connected via PTP BC/TC-capable switches wherever possible
4	If PTP packets and other traffic share the same network, it is advisable to assign PTP packets to a priority queue on the switch

When PTP is run over a network with fluctuations like this, the results produced by the algorithm do not converge stably. As a result of this, PTP OCs will decide that synchronization is not possible (PTP unlock) because the required accuracy cannot be guaranteed.

Yet the standard itself does not define synchronization accuracy criteria. PTP-capable devices often display states such as PTP lock and PTP unlock, but these are merely PTP state definitions internal to each implementation. That is, PTP unlock is displayed when the device has determined

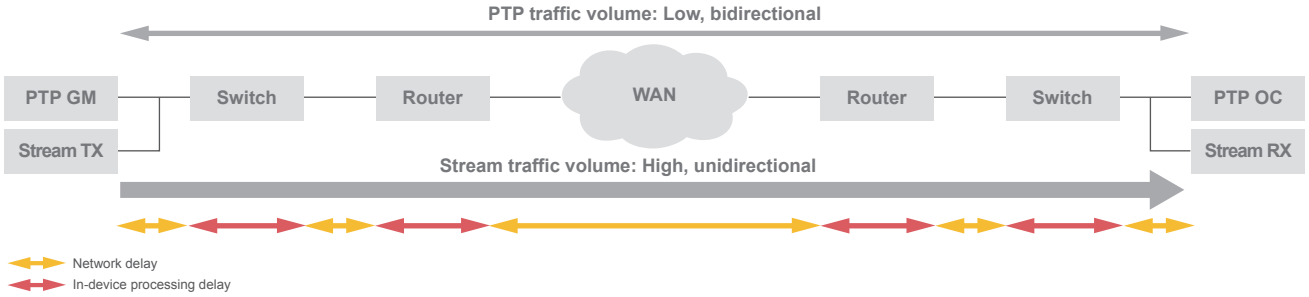

Figure 5: A Network Combining Video Transport and PTP Transport

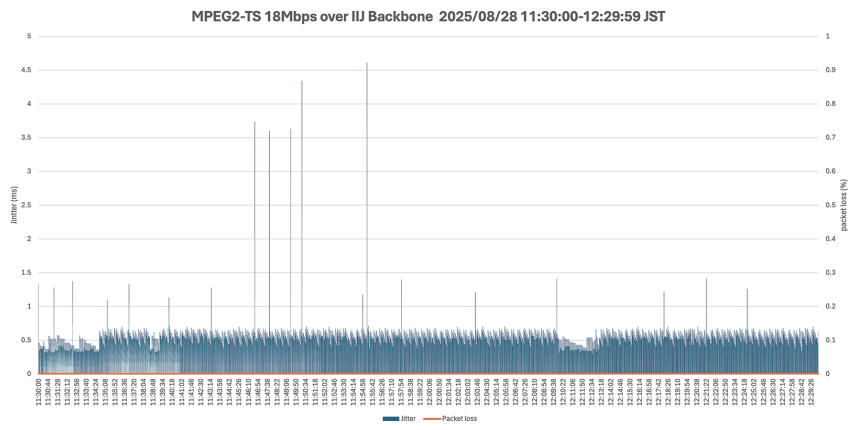

Figure 6: Example of Jitter Between Tokyo and Osaka on the IIJ Backbone
RTP reception jitter observed on the receiving hardware (IBEX Technology HLD-300C)
when transmitting MPEG2-TS 18 Mbps RTP via an L2VPN on the IIJ backbone

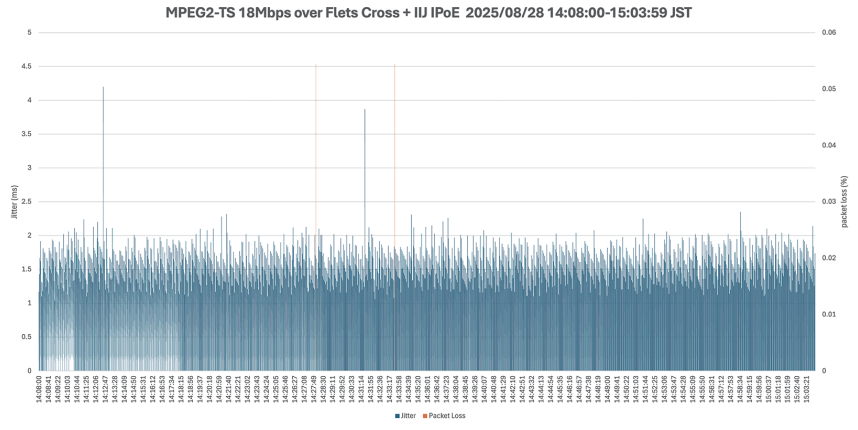

Figure 7: Example of Jitter Between Tokyo and Osaka on IIJ's FLET'S Connection Service
RTP reception jitter observed on the receiving hardware (IBEX Technology HLD-300C)
when transmitting MPEG2-TS 18 Mbps RTP via an L2VPN on IIJ's FLET'S connection service

that it cannot obtain accuracy that is good enough for synchronization. So even on the same network, the criteria for declaring a PTP lock will differ from one device to another.

2.10 RPTP as a Problem-Solving Approach (IIJ's Work)

This section discusses IIJ's approach to time synchronization over public networks, something that has been difficult with conventional PTP.

Conventional PTP was designed on the assumption of a private, high-quality network. The attempt to extend it to public networks is called RPTP. RPTP stands for Resilient PTP, a technology aimed at carrying PTP over unstable public networks. Its key feature is an algorithm that removes the jitter component from packets received from the PTP GM and correctly generates the Local PTP Clock. RPTP equipment applies an algorithmic filter on the slave side, with no changes to the PTP protocol specification at all. So an advantage of RPTP is that it can be used without modifying existing PTP GMs or other existing equipment. RPTP is designed to make PTP time synchronization possible with practical levels of accuracy, even on PTP-unaware networks.

RPTP consists of the two technical elements shown in Table 9.

As discussed above, RPTP uses an improved algorithm for calculating ordinary PTP's offsetFromMaster. Instead of using the received timestamps t2 and t3, RPTP uses virtual timestamps x2 and x3, generated by EVE. x2 and x3 are predicted values (stabilized receive and send times) from which delay variation has been removed using ADAM. Using x2 and x3, RPTP calculates (x2 - t1) and (t4 - x3), and from those measured values together with the values observed when delay is at its minimum (the minimum values), it derives forward and reverse offset estimates as regression lines. The reason it uses the minimum values is that the instant at which delay is smallest is considered to be when it is closest to the true delay.

Table 9: The Two Elements of RPTP

	Abbreviation	Full name	Role
1	EVE	EVEn clock source	Suppresses base time fluctuation on slaves and BCs
2	ADAM	Asymptote Delay Analysis Method	Improves synchronization accuracy over time and selects the best measured values

In EVE, the virtual clock used to calculate the regression line and the counter used to calculate the offset value operate independently. This design using two separate time bases makes stable offset estimation possible even on public networks with large delay variability. ADAM is an algorithm that analyzes these data, generates a regression line, and gradually converges on increasingly accurate predicted values.

RPTP is thus an algorithm and can be regarded as an application technology built on PTP. Because it does not modify the PTP protocol itself, it is not intended to be a standardization effort. We are considering its future development as a means of solving PTP problems in specific domains.

As a member of the RPTP Alliance, IIJ is working to promote wider adoption of the technology. RPTP was originally developed by Network Additions, and with Media Links, Seiko Solutions, and IIJ also now part of the RPTP Alliance, the group continues its work. With its strength as a network operator, IIJ has taken part in many RPTP proof-of-concept tests on real networks.

The RPTP-compatible product (DB3200) is implemented as a PTP BC. Its role is to "rectify," so to speak, the PTP timing that arrives at its upstream port after being disrupted by the PTP-unaware network. In other words, it provides rectified, jitter-free PTP time synchronization to downstream devices.

The DB3200 does more than simply provide PTP time synchronization as a PTP BC. It can also supply frequencies that have long been used in the synchronization world, such as 1PPS, 10 MHz, and 48 kHz. These frequencies are generated in the DB3200 via PTP time synchronization. This makes it possible to deliver accurate time and stable frequency even when the path goes through a public network.

The RPTP Alliance is conducting proof-of-concept tests (PoCs) in a variety of domains. Here, I describe one such PoC that uses IIJ resources (Figure 8). IIJ's Yokohama 1 Data Center and Osaka were connected by a FLET'S Hikari Cross line, and an L2VPN was set up over that line.

PTP was sent from a PTP GM in Osaka, and the packets that arrived over the VPN were received by a DB3200 in Yokohama and corrected using RPTP.

What RPTP aims to achieve in particular is PTP synchronization with remote locations over public networks. Public networks are, naturally, PTP unaware, and they cannot carry the IP multicast that PTP requires. Since PTP communication cannot take place under these conditions, the workaround is to set up an L2VPN over the public network so that PTP packets can be exchanged with remote locations.

There are several ways to measure PTP accuracy, but in this test, we used 1PPS observation.

In the world of timing and synchronization, which includes PTP and GNSS, a signal called 1PPS is commonly used as a reference. This stands for 1 pulse per second, and, as the name suggests, it is a signal characterized by a rising pulse at each second boundary. With this method, clock timing input and output take the form of a 1PPS signal. The accuracy of the rising edge is subject to various rules and conventions, and these are what guarantee precision. It is an approach for measuring timing, but it is also widely used when timing must be delivered accurately between devices. IEEE 1588 cites 1PPS output as an example for monitoring purposes. In Figure 9, the 1PPS outputs from the PTP GM and the DB3200 are both fed into a 1PPS logger and compared. With the PTP GM's 1PPS as a reference, RPTP performance is evaluated by observing how much the DB3200's 1PPS output fluctuates. This comparison is possible because both devices use GNSS as their time source.

PTP is thus versatile, and methods such as RPTP make it possible to broaden the range of use cases.

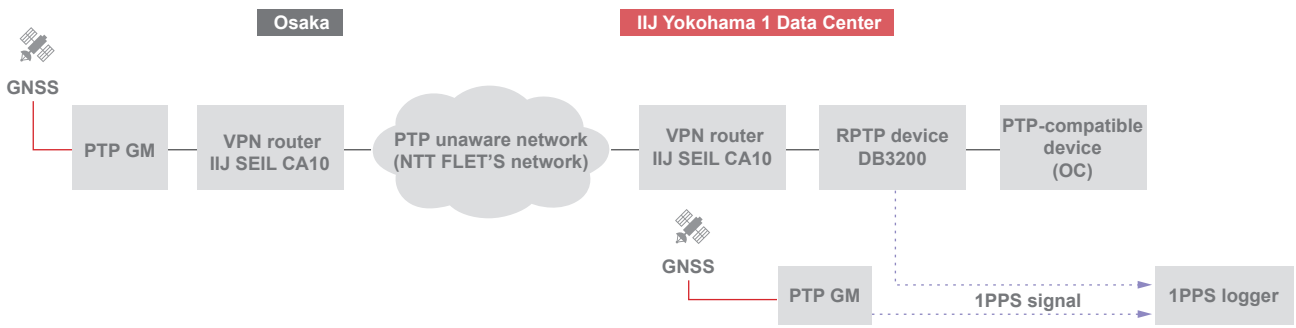

Figure 8: Test on a PTP-Unaware Network using RPTP
An L2VPN is set up between Yokohama and Osaka using SEIL CA10. The CA10 at each end bridges the LAN and VPN at L2. In Yokohama, the 1PPS outputs from both the DB3200 and the PTP GM are simultaneously fed into a logger for comparison.

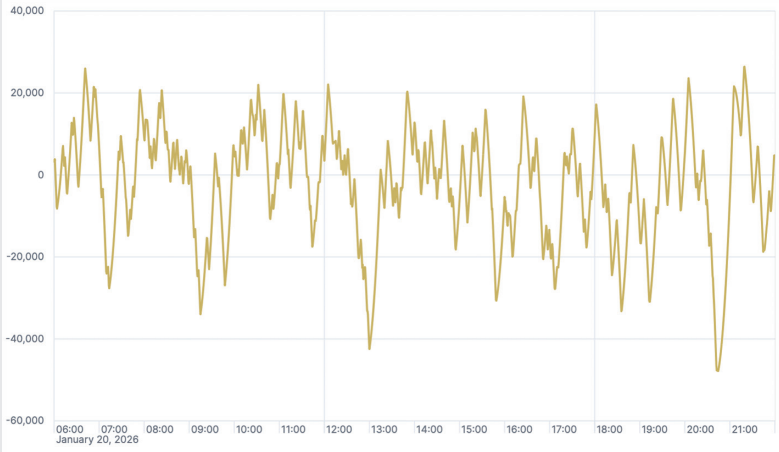

Figure 9: RPTP 1PPS Observed on the 1PPS Logger
Vertical axis is in nanoseconds. Phase difference of the DB3200 remains consistently within around ±20,000 nanoseconds (±20 microseconds). Values on the vertical axis are relative.

2.11 Just What is Time?

Finally, let's talk about the crux of all this: the notion of time.

The time provided by PTP and NTP represents how much time has elapsed since a given epoch (i.e., a reference point in time). They do not provide calendar time in the sense of something like "year, month, day, hour, minute, second." Instead, as Table 10 shows, they use time defined relative to an epoch.

CFAbsoluteTime also goes by names such as Apple Cocoa Core Data timestamp. It is used by Apple OSes. FILETIME is used by Windows OSes (starting with Windows NT 3.1).

Unix time and CFAbsoluteTime can take negative values, meaning they can represent times earlier than their epoch. Epochs, the way leap-seconds are handled, and the internal structures used differ even among commonly used OSes and applications.

The epoch used by PTP and NTP does not necessarily have to be an absolute time (calendar time), but assuming multiple systems exist, it is of course preferable for all of them to be synchronized to the same time source. This is where internationally standardized time scales come in (Table 11).

UTC (Coordinated Universal Time) is used widely around the world as a global reference time.

This does not imply, however, that there is a single physical UTC clock somewhere in the world running in real time. UTC is managed by the BIPM (Bureau International des Poids et Mesures), the organization responsible for realizing the International System of Units (SI). The BIPM collects data from atomic clocks operated by national standards institutes and observatories, and it publishes the offsets of the time scales maintained by those bodies on a monthly basis. The time scale calculated from those data is called TAI (Temps Atomique International).

UTC is based on TAI, but it is adjusted for leap seconds so that the difference from UT1 (the time based on Earth's rotation) does not become too large. Leap seconds are inserted into UTC in one-second units, and so as a time scale, UTC lags behind TAI. Leap seconds were

Table 10: Time Systems Used for Time Synchronization and in OSes

Time systems	Epoch	Leap seconds	Data structure	Signed/unsigned
PTP	1970-01-01 00:00:00 TAI	Not supported; accounted for when converting to UTC	Seconds (48-bit) + nanoseconds (32-bit)	Unsigned
NTP	1900-01-01 00:00:00 UTC	Accounted for	Seconds (32-bit) + fractional part (32-bit)	Unsigned
Unix time	1970-01-01 00:00:00 UTC	Not supported	Seconds (64-bit)	Signed
CFAbsolute Time	2001-01-01 00:00:00 GMT	Not supported	Seconds (64-bit floating point)	Signed
FILETIME	1601-01-01 00:00:00 UTC	Not supported	100-nanosecond intervals (64-bit)	Unsigned

*Unix time and FILETIME use UTC notation but are implemented as a continuous time scale that does not include leap seconds.

Table 11: International Standard Time standards

Time standard	Name	Basis	Maintaining organization
TAI	Temps Atomique International (International Atomic Time)	Based on atomic clocks	BIPM
UTC	Coordinated Universal Time	Based on TAI, with leap-second adjustments so that the difference from UT1 (the time based on Earth's rotation) does not become too large. As of 2025, UTC = TAI - 37 seconds.	BIPM
JST	Japan Standard Time	UTC +9 hours (i.e., TAI - 37 seconds + 9 hours)	NICT

inserted 27 times between 1972 and 2017. When UTC was redefined in 1972, the relationship was UTC = TAI - 10 seconds. As a result of the insertions, as of 2025 the relationship is UTC = TAI - 37 seconds.

The existence of leap seconds means that UTC does not represent a continuous time scale. So UTC and TAI must be treated separately in time synchronization mechanisms.

In Japan, the National Institute of Information and Communications Technology (NICT), the National Astronomical Observatory of Japan, and the National Institute of Advanced Industrial Science and Technology (AIST) maintain time using atomic clocks and also supply data to the BIPM. NICT maintains Japan Standard Time (JST) based on UTC(NICT), and it distributes JST via the longwave standard radio signal JJY, Hikari-Telephone JJY (which uses optical telephone lines), and an Internet-based NTP service (ntp.nict.jp).

GNSS (Global Navigation Satellite System) is widely used worldwide as a time distribution system. GNSS is a collective term that includes the U.S. GPS, Russia's GLONASS, the EU's Galileo, and Japan's Michibiki (QZSS). GNSS is used for positioning and navigation, and it has a third important role in providing high-precision time distribution. The satellites carry atomic clocks, and the use of GNSS-based time synchronization means that highly accurate time can be obtained almost anywhere on Earth. This is why GNSS is widely used as the reference source for NTP and PTP. GPS, Galileo, and Michibiki use continuous time scales without leap seconds (TAI-based time or time scales with a fixed offset from TAI). Galileo and Michibiki are also designed so that their time matches GPS.

That said, external factors can cause GNSS radio reception to become unstable or even impossible, raising the need for countermeasures in recent years (e.g., multipath rejection, anti-jamming, anti-spoofing, and EMI mitigation).

As well as PTP devices, standard time systems and GNSS satellites are also equipped with high-performance clocks to maintain accurate time. High-precision time synchronization equipment requires clocks with excellent frequency stability and accuracy. Table 12 lists some typical clock types.

Table 12: Comparison of Oscillator Types and Clock Generation Methods

Type	Configuration/principle	Key features	Frequency stability (typical)	Example use cases
Crystal oscillator	Quartz crystal resonator	High accuracy, low jitter	Approx. ±10–50 ppm	Clocks, microcontrollers
RC oscillator	Resistor (R) + capacitor (C)	Low cost, low accuracy	Approx. ±1,000–10,000 ppm	Internal clocks
LC oscillator	Inductor (L) + capacitor (C)	Suited for high frequencies	Approx. ±100–1,000 ppm	RF circuits
MEMS oscillator	Silicon resonator	Shock-resistant, compact	Approx. ±10–50 ppm	IoT, automotive
PLL	Uses reference clock	Frequency synthesis	Depends on reference clock	CPUs, telecommunications
TCXO	Temperature-compensated crystal oscillator	Resistant to temperature variations	Approx. ±0.1–1 ppm	Mobile devices, GPS
OCXO	Oven-controlled crystal oscillator	Very high stability	Approx. ±0.001–0.01 ppm (1–10 ppb)	Measurement, PTP GM
VCTCXO	Voltage-controlled TCXO	Fine-tunable	Approx. ±0.1–0.5 ppm	PTP/SyncE equipment
Rubidium atomic clock	Rubidium-87	Excellent long-term stability	Approx. ±0.00001 ppm (10^{-11})	Telecommunications, reference source
Cesium atomic clock	Cesium-133	Defines the second	Approx. ±0.000000001 ppm (10^{-13})	Standard time
Optical lattice clock	Strontium, ytterbium, etc.	Next-generation standard	On the order of 10-18	Research

*PLL is not an oscillator per se but a frequency synthesis method; it is included here because it is widely used for clock generation.

All of these oscillators generate a fixed-frequency signal. Reference ticks for time and frequency, such as 1 second or 10 MHz, are created by dividing or counting the generated frequency. Clocks need to provide frequency stability and frequency accuracy, and when multiple clocks are to be synchronized, it is also important that their phases line up (Figure 10, Figure 11).

Fundamentally, time synchronization entails establishing a state in which the frequency and phase of multiple clocks are aligned, and then sharing an absolute time counter based on a common epoch.

|Sidebar 4|

Standard signal transmitting station honored as IEEE milestone

In 2025, Japan's Standard Time and Frequency Signal Transmitting Station, in operation since 1940, was recognized as an IEEE milestone. The station broadcasts under the call sign JJY, formerly on shortwave and currently on longwave (40 kHz, 60 kHz). It transmits a pulse-coded time, and radio-controlled clocks with JJY reception capability set their time based on this signal.

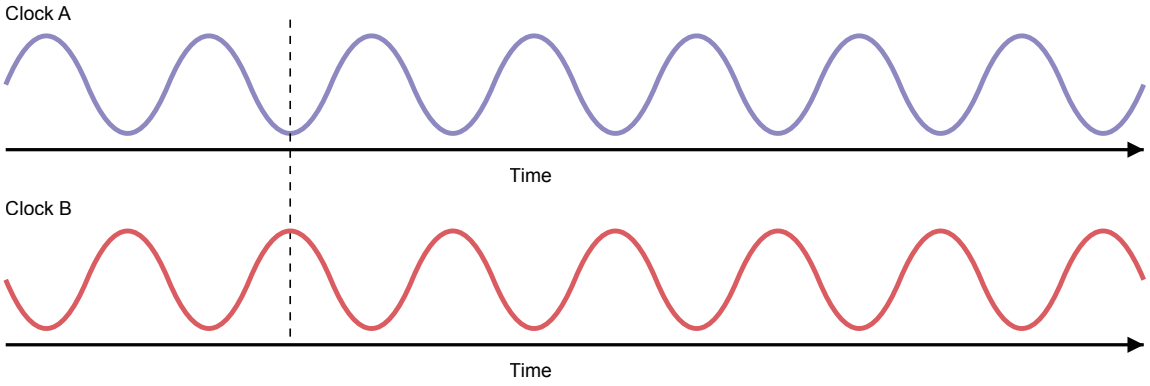

Figure 10: Same Frequency but Out of Phase
The waveforms have the same period, but when compared at the same reference point, the peaks are offset. In this example, the phase difference is 180°, a state known as "antiphase."

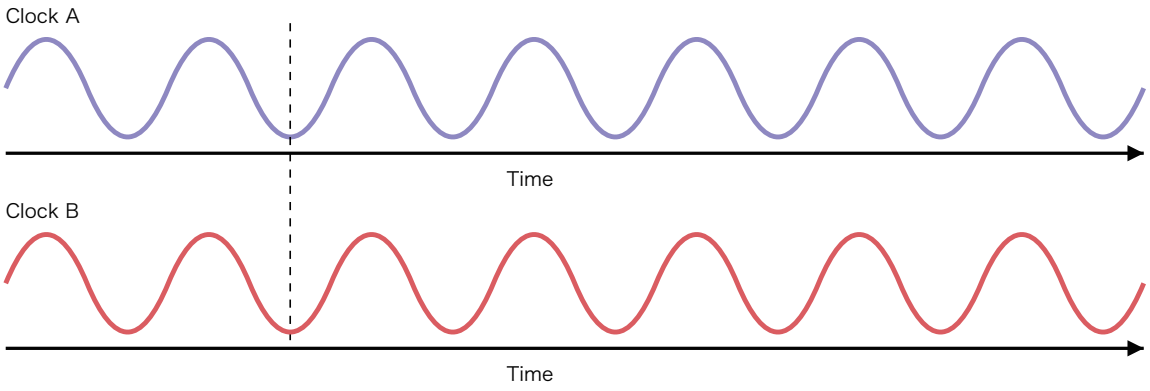

Figure 11: Both Frequency and Phase Are Aligned
Both frequency and phase are matched. This state is called "in phase."

2.12 Conclusion

This article has given an overview of PTP and described IIJ's work in this area. PTP underpins time synchronization, an essential factor in mission-critical systems. Even so, practical limitations (high-precision synchronization has effectively only been achievable on PTP-aware networks) have impeded its broader deployment. Through the RPTP initiative, we hope to further expand the range of PTP use cases.

Acknowledgments: In preparing this article, I benefited greatly from the insights that RPTP Alliance members shared through technical discussions and with respect to proof-of-concept work, for which I am sincerely grateful.

Table 13: Parameters for Typical PTP Profiles

Profile	domainNumber	logSyncInterval	logAnnounceInterval
IEEE 1588-2019 Default	0–255	-7 . . . +1	0 or 1 (profile-dependent)
AES67	0–127 (default 0)	-3	1
SMPTE ST 2059-2	0–127 (default 127)	-3	-2
IEEE 802.1AS (gPTP)	0–255 (fixed at 0 in older specifications)	Profile-defined	Profile-defined
ITU-T G.8275.1	0–255	-4 . . . 0	Profile-defined
ITU-T G.8275.2	0–255	-4 . . . 0	Profile-defined
IEEE C37.238 (Power)	Typically 0	-3	Profile-defined
White Rabbit	Typically 0	Arbitrary (high rate)	Profile-defined
Automotive (802.1AS-Rev)	Configurable	-5 . . . -3	Profile-defined
CIP Sync	Configurable	-4 . . . -3 recommended	Standard-dependent

Bunji Yamamoto

Business Strategy Department, Broadcast Systems Division, Network Services Business Unit, IIJ
Since joining IIJ Media Communications in 1995, Mr. Yamamoto has been involved in efforts to popularize streaming, CDN, and related technologies. His work with Video over IP led to an interest in time synchronization technology, and in 2025 he organized GNSS TimeSync 2025.