Description of Use of IEEE 1588 Followup Peer-to-Peer Transparent Clock in A/V Bridging Networks

Revision 0.0
2006.03.27

Geoffrey M. Garner
Samsung (Consultant)
gmgarner@comcast.net

1. Introduction

The Audio/Video Bridging (AVB) Task Group (TG) within IEEE 802.1 is considering the use of a subset of IEEE 1588 Version 2 Precise Time Protocol (PTP) (currently under development) to provide timing and synchronization to the various AVB network nodes. Specifically, the Draft PAR for AVB Timing/Synchronization (IEEE 802.1as) indicates that IEEE 802.1as "specifies the use of IEEE 1588 specifications where applicable in the context of IEEE Stds 802.1D and 802.1Q" and that it will "leverage the work of the IEEE 1588 WG to develop the additional specifications needed to address these requirements" [1]. A joint IEEE 802.1 AVB/IEEE 1588 Design Meeting was held February 21, 2006 [2], in which it was suggested that the Follow-up Peer-to-Peer (P2P) Transparent Clock (TC) being developed as part if IEEE 1588 Version 2 could be used advantageously to provide synchronization to AVB networks [2]. A possible way of using the Follow-up P2P TC, consistent with the current Draft Working Technical Description [3] was discussed verbally in the meeting and then documented in [4]. Reference [4], augmented by [3] plus other IEEE 1588 Version 2 Working Documents (e.g., relevant documents include, but are not necessarily limited to [5] and portions of [6] referenced in [5]) and the published IEEE 1588 Version 1 [7] provide a description of how IEEE 1588 can be used to provide timing and synchronization to AVB networks.[1] However, an initial presentation of [4] to the AVB TG indicated it would be desirable to provide a more self-contained description of the use of IEEE 1588 in AVB networks.

The main purpose of this document is to provide a detailed description of the use of IEEE 1588 for timing/synchronization in AVB networks, in a manner that is easily accessible and understandable to the user. The intent is to provide a detailed description of the protocols; nonetheless, this document is not a substitute for the eventual standards documents that are published (i.e., IEEE 1588 Version 2, IEEE 802.1as).

The document is organized as follows. Section 2 is an overview of the subset of the PTP clock synchronization model that will be used by AVB networks and additional assumptions and requirements for AVB that are not part of the PTP specification but are part of a PTP profile for AVB. Section 3 describes the message and frame formats. This description is taken from Reference [6]. Section 4 describes how each message is processed as it originates at the ingress node, arrives at the egress node, arrives at a node that is not the egress node, and is transmitted by a node that is not the ingress node.

[1] Additional information, referred to as a *1588 profile*, is also needed. This information will be specified in an IEEE 802.1 document. Profile information is typically specific to a respective application, and is not specified directly in IEEE 1588 because IEEE 1588 is used in a wide variety of applications that have different requirements.

2. Subset of PTP Clock Synchronization Model used in Audio/Video Bridging Networks

Some of the material and text in this section is either taken from or uses text in [9] as a starting point.

2.1 Overview of PTP Systems Used in A/V Bridging Networks

General PTP systems (i.e., not necessarily limited to AVB network applications) are distributed, networked systems consisting of some combination of PTP and non-PTP devices. PTP devices include ordinary clocks, boundary clocks, transparent clocks, and administrative nodes. Transparent clocks may be further subdivided into two types: (a) peer-to-peer (P2P), and (b) end-to-end (E2E). Non-PTP devices include ordinary network switches (i.e., bridges), routers, and/or other infrastructure devices, and possibly end application devices such as computers, printers, displays, video or audio players, etc.

The PTP protocol is a distributed protocol that specifies how the real-time PTP clocks in the system synchronize with each other. The ordinary and boundary clocks are organized into a master-slave synchronization hierarchy with the clock at the top of the hierarchy, i.e., the Grandmaster (GM) clock, determining the reference time for the entire system. The synchronization is achieved by exchanging PTP timing messages with the slaves using the timing information to adjust their clocks to the time of their master in the hierarchy. The transparent clocks are not part of the master slave hierarchy; however, each transparent clock may syntonize (i.e., synchronize in frequency but not time) to a master boundary or ordinary clock. In addition, an ordinary or boundary clock may be collocated with a transparent clock, in which case a timing signal synchronized to a master is available at the transparent clock. Every AVB network node will contain a collocated ordinary and peer-to-peer transparent clock. One of the nodes will be the master, and all the other nodes will be slaves to this master (and therefore the master will also be the grandmaster). An AVB network will not contain any non-AVB devices (i.e., any nodes that do not have PTP clocks).[2]

Devices in a PTP system communicate with each other via a communication network. In general, the network may include bridges between segments implementing different network communication protocols; in the case of AVB, the network is Ethernet. An ordinary clock (OC) has a single physical or logical connection to the network, i.e., a single port. A boundary clock (BC) may have multiple physical or logical connections to the network, i.e., multiple ports. An E2E or P2P transparent clock (TC) may have multiple physical connections to the network, i.e., multiple ports. In an AVB network, there is an implied single connection between an OC and the collocated P2P TC. This is illustrated in Figure 1.

[2] In a general PTP system, it is possible for multiple TCs to be connected to a non-PTP device in a star configuration, with the non-PTP device as the hub and the TCs as spokes. If more than 2 of the TCs are P2P TCs, the situation is referred to as a 1:N configuration. Version 2 of IEEE 1588 will not allow 1:N configurations of P2P TCs. In an AVB network, all the nodes must be AVB-enabled and therefore will contain P2P TCs; therefore, 1:N configurations will not arise in AVB networks (at least, for the present AVB, using wired Ethernet links).

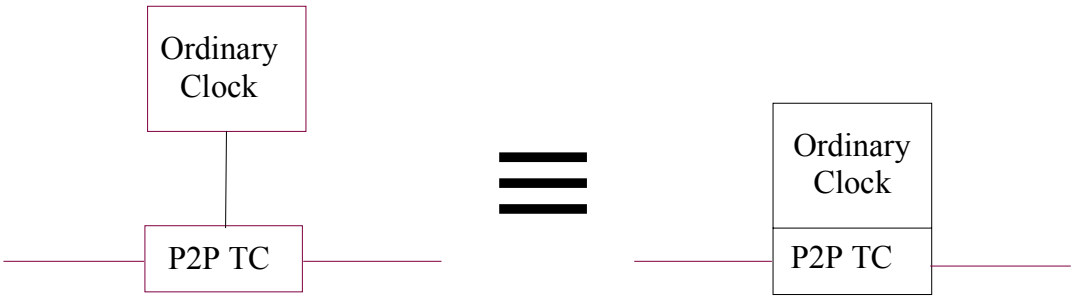


Figure 1. Illustration of implied single connection between an OC and the collocated P2P TC

The PTP protocol executes within a logical scope called a subdomain. All PTP messages, data sets, state machines and any other PTP artifacts are always associated with a particular subdomain. In general, a given physical network and individual devices connected to the network can be associated with multiple subdomains. The time established within a subdomain by the PTP protocol is independent of the time in other subdomains. In the case of an AVB network, the entire network will consist of a single subdomain.

2.2 AVB Synchronization Overview

There are two phases in the normal execution of the PTP protocol in an AVB network:

a) Establishing the master-slave hierarchy, and
b) Synchronizing the clocks.

2.2.1 Establishing the Master-Slave Hierarchy

In an AVB network, each ordinary clock examines the contents of all Announce messages received. Using the best master clock (BMC) and state decision algorithms these contents and the contents of the data sets associated with the OC are analyzed to determine the state of the single implied OC port and of the OC. This process establishes one OC as the GM and the other OCs as slaves, in the single subdomain of the AVB network, as illustrated in Figure . Note that the P2P TCs are not part of the master-slave hierarchy.

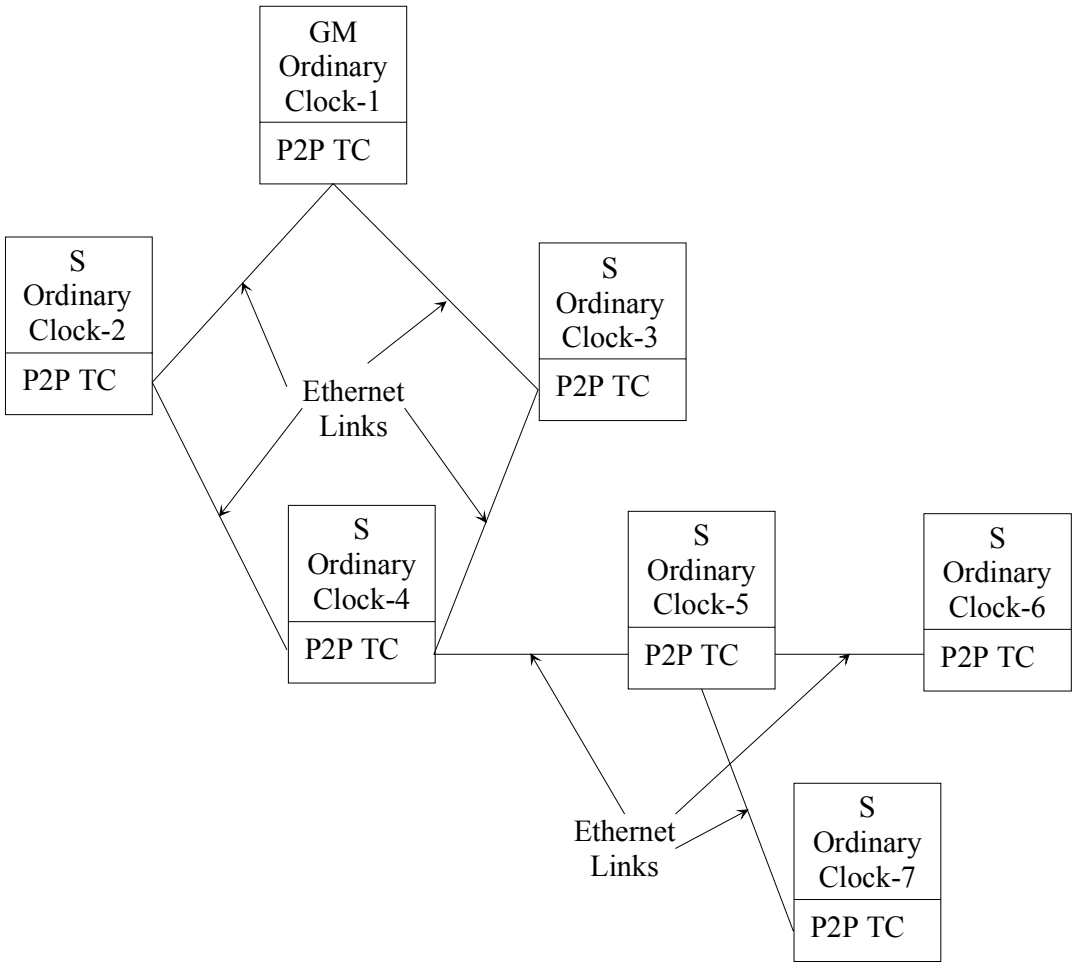


Figure 2. Illustration of AVB clock hierarchy

In Figure 2, the GM and slaves communicate via the Ethernet links. Each node is assumed to have a forwarding database, as would be present in conventional Ethernet, that enables unicast and multicast messages to reach their intended destinations without cycling endlessly. The manner in which the forwarding databases are constructed is not specified in IEEE 1588, Version 2. One way in which the forwarding databases can be constructed is through the use of spanning tree algorithm, but IEEE 1588, Version 2 does not require this.[3]

[3] In IEEE 1588, Version 1, TCs are not specified; a node is either a BC or an OC. In addition, BCs do not forward PTP messages. Therefore, the issue of constructing a forwarding database to ensure that PTP messages reach their intended destinations does not arise in a 1588 Version 1 network that has only BCs and OCs (i.e., no TCs and no non-PTP devices). This is because all communication is point-to-point in such a case. Also in such a case, the application of the BMC algorithm is more complicated than in Figure 2 above because now the master-slave hierarchy has multiple levels; nonetheless, the BMC algorithm establishes the hierarchy. In Figure 2, the master-slave hierarchy and application of the BMC algorithm is much simpler than in general IEEE 1588, Version 1 networks that have multiple BCs, but the network of Figure 2 does need to ensure that the PTP messages are forwarded properly. Finally, note that Version 1 networks that have non-PTP devices also must make sure that the PTP messages are forwarded properly (and the specification of this is beyond the scope of IEEE 1588).

The network in Figure 2 is equivalent to a single PTP communication path. A PTP communication path supports the direct communication among the ports of ordinary and boundary clocks. Boundary clocks do not forward Sync, Follow_Up Delay_Req, Delay_Resp, or Announce messages, and different ports of a boundary clock are on different PTP communication paths. However, AVB networks will not contain boundary clocks (and no BCs are present in Figure 2).

2.2.1.2 Description of Best Master Clock algorithm

To be supplied.

2.2.2 Synchronizing the clocks

In a PTP system, a master and slave clock synchronize by exchanging timing messages. For example, in 2 the GM (Ordinary Clock-1) synchronizes a slave, e.g., Ordinary Clock-2, by exchanging messages with the slave. We first describe how a master and slave are synchronized in Version 1 of IEEE 1588, i.e., where the master and slave are directly connected and there are no TCs in between. We then consider the case of an AVB network, where P2P TCs might be present (we do not consider E2E TCs as these are not used in AVB networks). As part of the latter, we consider (1) synchronization in general PTP systems that use P2P TCs, (2) measurement of link propagation times using the Peer_Delay mechanism, (3) processing Sync and Follow_Up messages at P2P TCs in AVB, (4) syntonizing P2P TCs to the GM, (5) architectural considerations for AVB node synchronization, and (6) application filter requirements. Note that some of the items described here, e.g., aspects of (3), (4), (5), and (6) are not specified for general PTP systems and are properly part of a 1588 profile for AVB networks.

2.2.2.1 Synchronizing a master and slave that are directly connected

Consider a master and slave that are directly connected, i.e., have no TCs and no non-PTP devices between them. The basic pattern of synchronization message exchange is illustrated in 3.

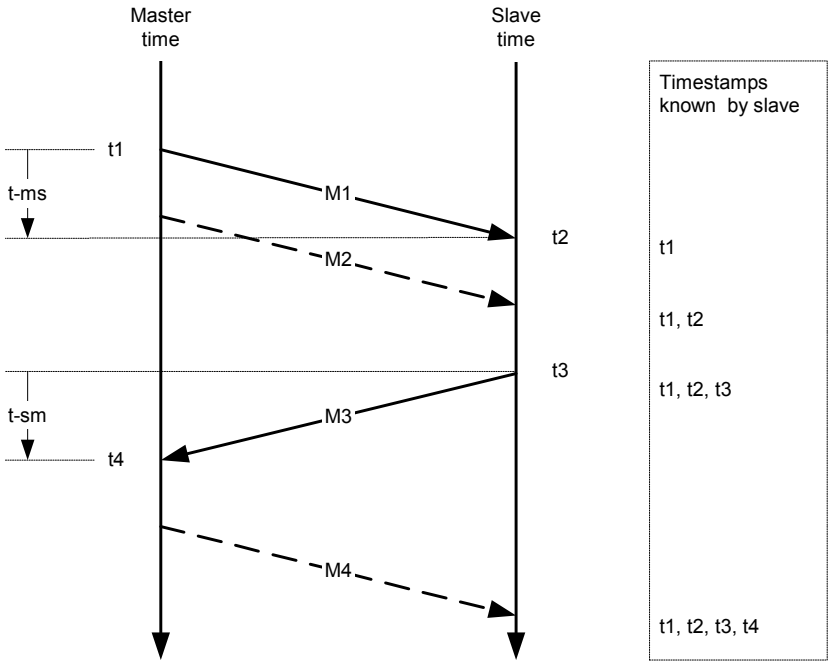


Figure 3. Basic synchronization message exchange (taken from[9])

The message exchange pattern is (items (a) – (f) below are taken from [9]):

a) The master sends a message M1, referred to as Sync, to the slave and notes the time, t1, at which it was sent.

b) The slave receives the message M1 and notes the time of reception, t2.

c) The master conveys to the slave the timestamp t1 by:
   1) Embedding the timestamp t1 in message M1. This requires some sort of hardware processing for highest accuracy, or
   2) Embedding the timestamp t1 in a second message M2, referred to as Follow_Up. This can be done in software since the timing is not critical.

d) The slave sends a message M3, referred to as Delay_Req, to the master and notes the time, t3, at which it was sent.

e) The master receives the message M3 and notes the time of reception, t4.

f) The master conveys to the slave the timestamp t4 by embedding it in a message M4, referred to as Delay_Resp.

At the conclusion of this exchange of messages, the slave possesses all four timestamps. These timestamps may be used to compute the offset of the slave's clock with respect to the master and the mean propagation time of messages between the two clocks, that is the mean of t-ms and t-sm in 3. The computations are

$$
\begin{aligned}
t_{ms} &= t_2 - t_1 \\
t_{sm} &= t_4 - t_3 \\
mean\_propagation\_time &= \frac{t_{ms} + t_{sm}}{2} \\
slave\_offset &= t_2 - t_1 - mean\_propagation\_time
\end{aligned}
\qquad (2\text{-}1)
$$

The expression for slave offset assumes that the master-to-slave and slave-to-master propagation times are equal. Any asymmetry in propagation time will introduce an error in the computed value of slave offset.

The accuracy of the slave offset computation also depends on how accurately the times t_1, t_2, t_3, and t_4 are measured, i.e., the computations in Eqs. (2-1) depend on these times reflecting when the Sync and Delay_Req messages actually are sent and received. This means that the measurement must be made below the Ethernet mac, i.e., at the MII or GMII. In addition, if Follow_Up is not used, i.e., if an accurate measurement of t_1 is placed in the Sync message, the hardware will be more expensive. For this reason, AVB will use the Follow_Up message.

This basic exchange of messages is used in two ways in the PTP protocol:

g) To synchronize between an ordinary clock and a boundary clock, and

h) To measure link propagation time.

The propagation times usually vary very slowly, if at all. Therefore, it is often possible to perform the Delay_Req/Delay_Resp message exchange much less frequently than the sending of Sync and, if implemented, Follow_Up. In this case, multiple Sync and Follow_Up messages are sent between successive Delay_Req/Delay_Resp exchanges. The mean propagation time measured by a Delay_Req/Delay_Resp exchange is used by the slave in all subsequent clock offset calculations until the next Delay_Req/Delay_Resp exchange.5

The time between successive Sync messages is referred to as the synch interval.

2.2.2.2 Synchronizing a master and slave that communicate through one or more P2P TCs

If there are one or more P2P TCs between the master and slave, the master-to-slave and slave-to-master propagation times will, in general, not be equal and will vary appreciably with time. The reason for this is that the Ethernet links in Figure 2 will also carry application traffic; in fact, this will be the majority of the traffic in the network. Even if the PTP messages have highest priority,[4] the priority will be non-preemptive; a large Ethernet frame in service when it is desired to send a Sync message from a TC node can result in addition delay whose value will be between zero and the transmission delay for this frame. For a maximum sized Ethernet frame (1500 bytes of payload), the maximum value for this delay is nearly 125 μs for 100 Mbit/s Ethernet. This is much larger than the delays due to propagation through the PHY and wire.

The TC solves the above problem by measuring the time the Sync message arrives, t_a, and the time the Sync message departs, t_d, and computing the difference, $t_r = t_a - t_d$. This difference is referred to as the residence time. The residence time is accumulated in a field of the Sync or Follow_Up message referred to as the correction field. Specifically, the correction fields of the Sync message

[4] AVB will use the priority mechanism of IEEE 802.1D and 802.1Q.

and Follow_Up message, respectively, are initialized to zero when the GM creates those messages. When a TC measures residence time, it has two choices. If it is able to make a sufficiently accurate measurement of t_r and add the value of this measurement to the correction field of the Sync message, i.e., if it is an on-the-fly TC, it does this. In this case, it does not alter the correction field of the Follow_Up message when that message is received and sent by the TC. However, if it is a follow-up TC, i.e., if it cannot make an accurate residence time measurement sufficiently quickly to add it to the value of the Sync message correction field as the Sync message is transmitted, it instead adds the residence time to the correction field of the Follow_Up message. As it is more expensive to make on-th-fly measurements of residence time, AVB P2P TCs will be follow-up TCs.[5]

The P2P TCs also measure propagation times on the Ethernet links that connect them using the three Peer_Delay messages, referred to as Peer_Delay_Req, Peer_Delay_Resp, and Peer_Delay_Resp_FollowUp. The details of this measurement will be described shortly; however, the result is that each P2P TC at the end of a link knows the propagation time on that link. A P2P TC will also accumulate in the correction field of the Sync or Follow_Up message the propagation time for the link on which the Sync message arrived. Specifically, an on-the-fly P2P TC will accumulate the propagation time in the Sync correction field, and a follow-up P2P TC will accumulate the propagation time in the Follow_Up correction field. When the Sync and Follow_Up messages reach their final destination slave clocks, the sum of the correction fields in these messages is a measure of the total residence time in all the intervening TCs plus the propagation times on all the links except the final ingress link at the destination. This is illustrated in the example in Figure 4.

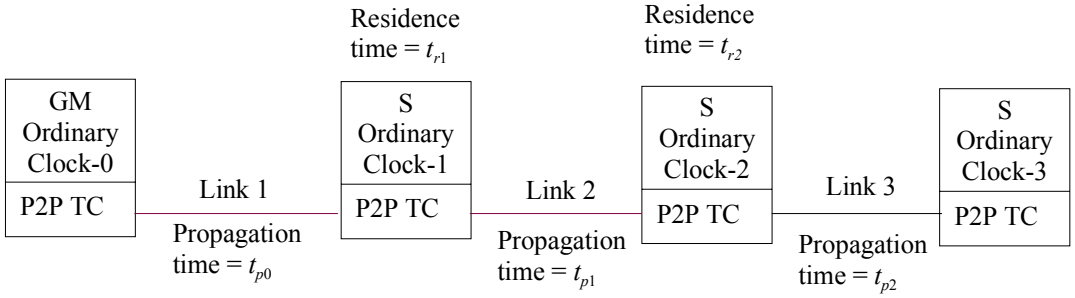


Figure 4. Illustration of accumulation of residence times and propagation times by P2P TCs

Figure 4 consists of a GM (ordinary clock 0) and three slave clocks (ordinary clocks 1 –3). The link connecting clocks i and i+1 is labeled link i+1. The residence time in clock i is denoted t_{ri}, and the propagation time on link j is denoted t_{pj}. If clock N is the destination clock (i.e., there are a total of N+1 ordinary clocks, with N = 3 in the example here), then residence times are measured in clocks 1 through N-1, and link propagation times are measured by the Peer_Delay mechanism on links 1 through N. When the Sync and Follow_Up messages arrive at clock N, the sum of their correction fields is given by

$$Sync\_correction\_field + Follow\_Up\_correction\_field = \sum_{i=1}^{N-1}\left(t_{ri} + t_{pi}\right). \qquad (2\text{-}2)$$

The reason the propagation time on the final link is not included in Eq. (2-2) is that the propagation time is added to the correction field when the Sync or Follow_Up message is transmitted from the node, but at the final destination the Sync and Follow_Up messages are not transmitted to any

[5] Actually, on-the-fly measurements will be allowed, but will not be required.

additional nodes. However, the propagation time on this final link, t_{pN}, is known at the destination node, because the P2P TC in that node (node N) participates in a Peer_Delay measurement with the TC at the other end of the link (node N-1). Therefore, the propagation time may be easily added to the sum in Eq. (2-2) to give

$$total\_propagation\_plus\_residence\_time = \sum_{i=1}^{N-1} t_{ri} + \sum_{i=1}^{N} t_{pi} \,. \tag{2-3}$$

The slave offset is now computed as

$$slave\_offset = t_2 - t_1 - total\_propagation\_plus\_residence\_time \,, \tag{2-4}$$

where we have reverted to the notation of Figure 3 for the times the Sync message is transmitted by the GM and received at the final destination, i.e., t_1 is the time the Sync message is transmitted from the GM (ordinary clock 0), t_2 is the time the Sync message is received by the destination slave clock (ordinary clock N, with $N = 3$ in Figure 4), and *total_propagation_plus_residence_time* is given by Eq. (2-3).

Note that even though a P2P TC is present at the GM, it does not measure residence time. This is because the time t_1 that the Sync message departs the GM network element occurs after the message traverses the P2P TC (because the GM and P2P TC functions are located in the same NE).

With this mechanism, there is no need for the Delay_Req and Delay_Resp messages shown in Figure 3. Instead, the propagation times are computed using the Peer_Delay messages, which are described shortly.

In an actual AVB network, all the slave clock nodes must be synchronized; in addition, the topology will, in general, be a mesh topology (Figure 2) rather than a linear topology (Figure 4). PTP messages are, by default, multicast. This means that the GM sends a single Sync and a single Follow_Up message to all the nodes; it does not have to generate a separate Sync and separate Follow_Up message for each node. The forwarding database ensures that each slave clock receives each Sync and each Follow_Up message that the GM transmits, and also that Sync and Follow_Up messages do not cycle endlessly. The multicast mechanism means that a Sync message received by a P2P TC on one port may be transmitted on multiple ports. If this is done, the transmission time is measured separately for each port that the message is transmitted on, and a separate residence time is computed for each transmitted port. If the P2P TC is on-the-fly, the correction field of the transmitted Sync message is updated separately on each port using the respective residence time for that port and propagation time for the link attached to the port the message arrived on. When the corresponding Follow_Up message arrives, it is transmitted on the same ports that the Sync message was transmitted on. If the P2P TC is a follow-up TC, the correction field of the transmitted Follow_Up message is updated separately on each port using the respective residence time for that port and propagation time for the link attached to the port the message arrived on.

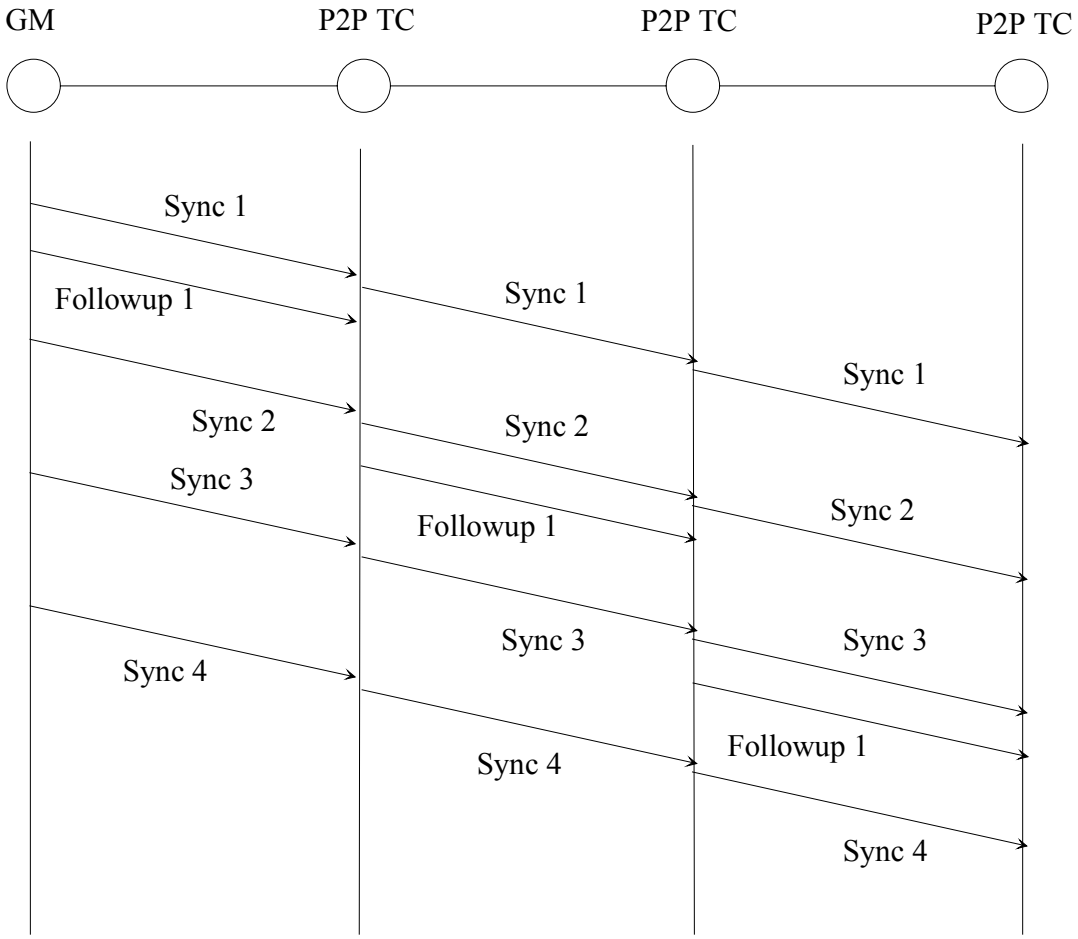
2.2.2.3 Measurement of link propagation times using Peer_Delay mechanism

This section will be filled in. The Short Frames group is considering a 2 timestamp format for the Peer_Delay messages. Note: will the mechanism now be symmetric?

2.2.2.4 Processing of Sync and Follow_Up messages at P2P TCs in AVB

AVB network nodes will be assumed to have standard Ethernet oscillators, with nominal rates of 25 MHz for 100 Mbit/s Ethernet and 125 MHz for 1 Gbit/s Ethernet. This means that the phase measurement granularity in the TC and OC can be as much as 40 ns. Additional phase error will result from the variable component of latency in the Ethernet PHY (the fixed component can be specified by the manufacturer in the design). Some AVB applications have stringent jitter and wander requirements. For example, uncompressed digital video has jitter requirements of less than 1 ns peak-to-peak measured through a 10 Hz low-pass jitter measurement filter, and maximum frequency offset and drift of 0.23 ppm and 0.023 ppm/s, respectively [10]. Consumer grade digital audio has a jitter requirement of 10 ns peak-to-peak measured through a 10 Hz low-pass jitter measurement filter, and maximum frequency offset of 50 ppm [10]. Additional details on these requirements are given in [10]. It is expected that a Sync interval as short as 10 ms or less will be needed in AVB networks to meet the jitter and wander requirements.

AVB network nodes will also use an inexpensive processor, e.g., the 8051. While Sync messages in theory require minimal processing by a Follow-up P2P TC (i.e., the TC needs only to measure the arrival and departure times of the Sync message), Follow_Up messages require more processing. It is expected that the 8051 processor may take up to 10 ms to process a Follow_Up message. This means that for a path through N PTP TCs (excluding the GM and final slave node), it will require at least on the order of $10N$ ms for the Followup message to travel from the master to the slave. Since the Sync messages require little or no processing, they will travel from the master to slave in much less than $10N$ ms (the time for the Sync to travel from the master to the slave will likely be on the order of the AVB latency requirement; values in the range of 2 – 6 ms have been discussed). Figure 5 illustrates this scenario for the case of 2 TCs between the GM and final slave node, i.e., $N = 2$.



Note: The Followup messages corresponding to Sync2 and Sync3 are not shown to keep the diagram from being too cluttered.

Figure 5. Illustration of the accumulation of multiple Sync messages at a node before the Followup corresponding to the first Sync arrives. Followup processing time and sync interval are of the same order. Sync processing time is much less than Followup processing time.

In Figure 5, when Follow_Up 1 (corresponding to Sync 1) arrives at the final node, three Sync messages have arrived (including Sync 1. It is seen that with each successive hop, the Followup message processing delay causes one additional Sync message to get ahead of the Followup message. If the master sends Sync every 10 ms, this means that each P2P TC would have to maintain state information on the residence times for multiple Sync messages at any given time (the final TC in the chain would have to save information for as many as N+1 Sync messages). Note that the Follow_Up message processing time occurs at each successive P2P TC but not at the GM at the beginning of the chain; it is assumed that the GM can send Follow_Up almost immediately after sending Sync (i.e., in a time short compared to the sync interval). If the Follow_Up processing time were also incurred at the GM, then one additional Sync message would have arrived when the

Follow_Up message arrives at each TC, i.e., the final TC in the chain would have to save state information for as many as *N*+2 Sync messages.

The need to save state information on multiple Sync messages at each P2P TC may be avoided by having each Sync message held at a TC until the corresponding Follow_Up message arrives. When the Follow_Up message arrives, its correction field is added to the correction field of the Sync, and the Sync is sent. The time the Sync is sent is noted; using this and the time the Sync arrived, a residence time for the Sync in the current TC node is computed. A new Follow_Up message is generated, and the sum of the residence time just measured and the delay on the upstream link on which the Sync message arrived is placed in the correction field of the Follow_Up message. This approach is illustrated in Figure 6.

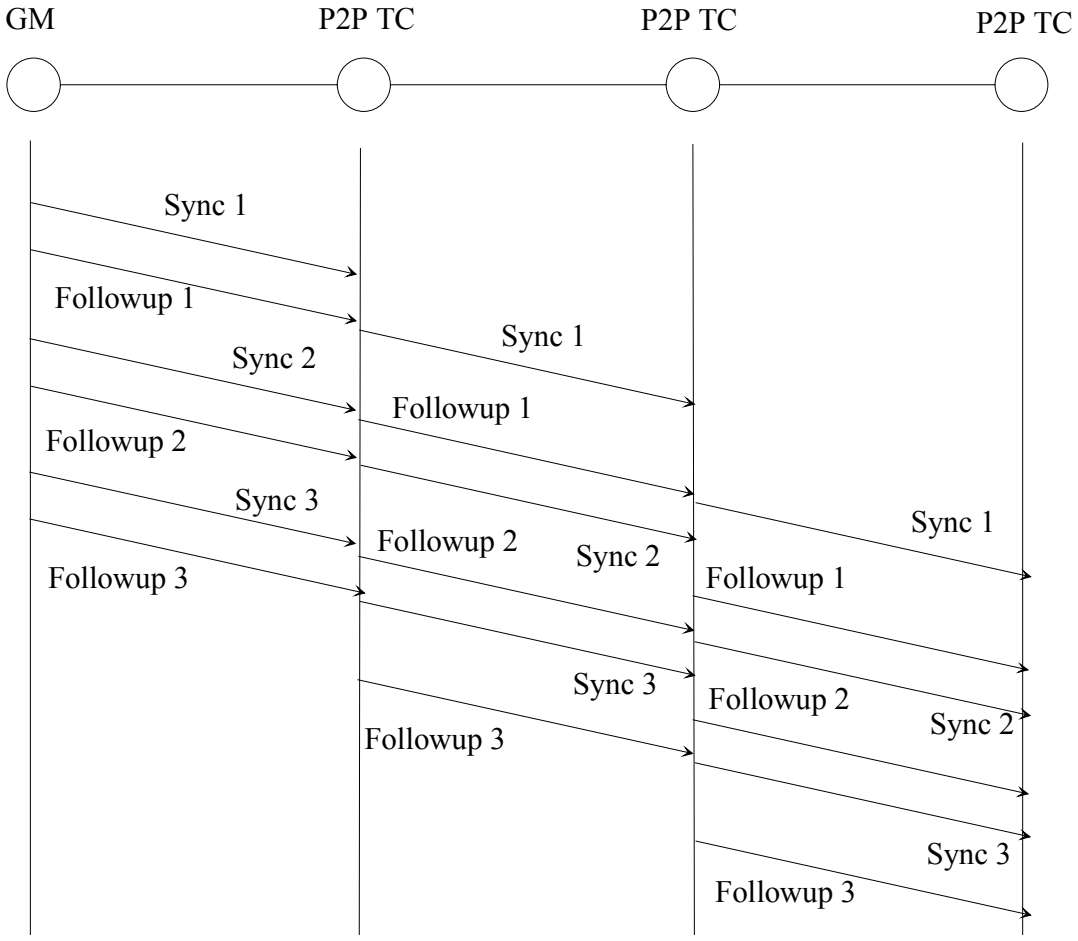


Figure 6. Sync and Followup sent with Sync held at P2P TC until corresponding Followup arrives. Followup processing time and sync interval are of the same order. Sync processing time is much less than Followup processing time.

In Figure 6, successive Sync messages do not get ahead of Follow_Up messages associated with previous Sync messages. Note that in both cases the Follow_Up message processing time occurs at each successive P2P TC but not at the BC at the beginning of the chain; it is assumed that the BC can send Follow_Up almost immediately after sending Sync (i.e., in a time short compared to the

sync interval). If the Follow_Up processing time were also incurred at the BC, the Synch messages would be held longer at the first P2P TC in Figure 6.

With the approach of Figure 6, a Sync message will not get ahead of a Followup message for a previous Sync as long as the time between Sync messages is not shorter than the time required for processing a Followup message at a node (i.e., 10 ms). When a Sync message is sent, at least 10 ms will have elapsed since the previous Sync message, which means that the Followup message will have had enough time to be processed at the next downstream node. It is a general requirement for all PTP systems that use P2P TCs that process Follow_Up messages that the time to process a Follow_Up message not exceed the Sync interval. If this requirement is not met, then an increasing backlog of Follow_Up messages will accumulate over time at each node that is traversed by Sync and corresponding Follow_Up messages.

The approach of Figure 6 is fully consistent with the semantics of how a P2P TC handles Sync and Follow_Up, i.e., Eqs. (2-2) – (2-4). Both the approach of Figure 6 and the conventional approach (Figure 5) result in the sum of the Sync and Follow_Up correction fields being the same on arrival at any P2P TC. Therefore, these equations may be used to compute the slave offset in either approach.

2.2.2.5 Syntonizing the P2P TC to the Grandmaster

A P2P TC contains a free-running oscillator with frequency accuracy no worse than ± 100 ppm. If residence time is measured using this oscillator, there will be an error on the order of the residence time multiplied by the actual frequency offset. With the approach of Figure 6 described in the previous subsection, the residence time may be on the order of a synch interval, e.g., as much as 10 ms, due to the holding of a Sync message at each P2P TC until the corresponding Follow_Up message arrives. This can result in an error in the residence time measurement on the order of (100000 ns/s)(0.01 s) = 1000 ns. To reduce this error, IEEE 1588 Version 2 allows the P2P TC to be syntonized, i.e., synchronized in frequency, to the Grandmaster. While syntonization of the P2P TC to the GM is not mandatory in IEEE 1588, it will be mandatory for AVB networks.

A P2P TC will syntonize to the GM by comparing a time interval measured by the GM with the same time interval measured by the local, free-running oscillator of the P2P TC. The time interval is equal to M Sync intervals. At present, $M = 10$, though this may change based on the results of jitter and wander performance simulations. The measurement is done as follows. The P2P TC already must measure when each Sync message arrives in order to compute the residence time. An estimate of the GM time when the Sync message arrives is given by

$$t_{GM} = t_1 + total\_propagation\_plus\_residence\_time, \qquad (2\text{-}5)$$

where t_1 is the time the GM sends the Sync message as defined in Figure 3 and *total_propagation_plus_residence_time* is given by Eq. (2-3) and is computed as the sum of the correction fields in the Sync and corresponding Follow_Up message plus the propagation time on the link that the Sync message arrived on (this computation may be seen to be equivalent to Eq. (2-3) by comparing Eq. (2-3) with Eq. (2-2)). Note that in order to compute t_{GM}, the P2P TC must wait until the Follow_Up message corresponding to the Sync message arrives. If t_2 is the time the time the Sync message arrives, and if i indexes the synch interval at which the measurement is made, than a measurement of the frequency offset of the P2P TC free-running oscillator relative to the GM is given by

$$y_{TC,i+M} = \frac{t_{2,i+M} - t_{2,i}}{t_{GM,i+M} - t_{GM,i}}. \qquad (2\text{-}6)$$

Note that each P2P TC needs only its own local free-running oscillator phase information and the information conveyed by the GM in the Sync and Follow_Up messages. It is not necessary for any P2P TC to convey its local free-running phase information to any other P2P TC or to the GM.

Each P2P TC will use the measured frequency offset relative to the GM to synthesize a frequency signal that is syntonized with the GM. This synthesis may be done via hardware, firmware, or software.

2.2.2.6 AVB Node synchronization architecture

The above subsection describes how the P2P TC contains a free-running oscillator and syntonizes this to the GM frequency by measuring its frequency offset relative to the GM. Similarly, subsections 2.2.2.1, 2.2.2.2, and 2.2.2.4 describe how a slave clock synchronizes to the GM by measuring its phase offset relative to the GM; the specific computation is given by Eqs. (2-2) – (2-4). Eqn. (2-4) indicates that the slave clock measures the time the Sync message arrives, but is not specific on whether the slave clock uses the local free-running oscillator embedded in the P2P TC, the signal synthesized by the P2P TC that is syntonized to the GM, a separate free-running local oscillator in the slave, or the synchronized timing signal produced in the slave using the computed phase offsets. This subsection discusses the relative merits of each choice and concludes that the best approach for AVB is to use the syntonized timing signal synthesized by the P2P TC.

Since one of the requirements for AVB is low cost, it is desirable to have a single oscillator in an AVB NE for both the P2P TC and slave clock functions. If the P2P TC were not syntonizing to the GM, the slave could still synchronize using Eqs. (2-2) – (2-4). Even if there were no P2P TCs between the slave and GM, the fact that the slave and GM frequencies were different would result in a computed *slave_offset* (Eqs. (2-1) – (2-4)) on the order of the frequency offset between the free-running slave/TC oscillator and the GM multiplied by the synch interval. This could be as large as (100000 ns/s)(0.01 s) = 1000 ns. However, since the frequency offset between the GM and slave/PC oscillator is already being measured and a syntonized frequency is being created, the use of this frequency for the slave offset computation will greatly reduce the magnitude of the computed slave offset phase step. The phase step magnitude will now be on the order of the syntonized frequency measurement accuracy multiplied by the synch interval. For example, if the phase measurement granularity is 40 ns and the P2P TC oscillator offset is measured over 10 synch intervals, i.e., 100 ms, the error in measured frequency offset is 40×10^{-9} s/0.1 s = 400×10^{-9} = 0.4 ppm. The slave offset now is (400 ns/s)(0.01 s) = 4 ns, i.e., is reduced from the 1000 ns computed when the free-running frequency is used for the measurement by a factor of 250. In practice, the reduction will not be this large because other effects are present, e.g., oscillator phase noise and drifts due to temperature effects, phase measurement error due to the variable portion of the PHY latency, and frequency measurement granularity. A better estimate of the synchronization performance will be determined via simulation.

The syntonized signal has the same average frequency as the GM (except for measurement errors described above), but not necessarily the same phase. There may be a large phase difference between the time signal at the P2P TC syntonized to the GM and the GM signal itself (e.g., due to an initial phase offset). However, the only impact of this large initial phase offset is that it is added to the slave time measurement t_2, which is in error by the opposite amount. The conclusion is that the synchronization performance will be very similar if one uses the syntonized signal to make the slave offset measurement versus a synchronized signal, i.e., a signal that has past computed slave offsets added back in.

Note that in the above paragraph we are referring to an *unfiltered* synchronized signal. After adding the slave offset to the signal used to make the slave offset measurement, the signal may be filtered to reduce any jitter and wander due to the offset addition. If the filtered signal were used to measure the message arrival times, performance could be improved. However, in AVB networks filtering will be application dependent, i.e., applications with more/less stringent jitter and wander requirements can require filters with tighter/looser bandwidth and gain peaking requirements. The reason for this is so that the cost of any expensive filtering will be borne by applications that need the filtering. AVB networks will either not require any level of filtering to be present by default (i.e., regardless of what applications are being demapped at the node) or any required filtering will be minimal. If filtering is not assumed to be present, there is likely no major performance difference between the cases where the syntonized and synchronized signals are used for the slave phase offset measurement.

2.2.2.7 Filtering the synchronized timing signal

The synchronized timing signal obtained by adding the measured slave phase offset to the syntonized (to the GM) signal used to measure the times of arrival and departure at/from the P2P TC will have jitter and wander due to the phase steps caused by the offsets. These signals may be filtered to reduce the jitter and wander. An y filter requirements will be application dependent, so that the cost of any expensive filtering will be associated with the application that requires it. Regardless of the level of filtering, any filter requirements may be expressed generically using a transfer characteristic. An example is shown in Figure 7.

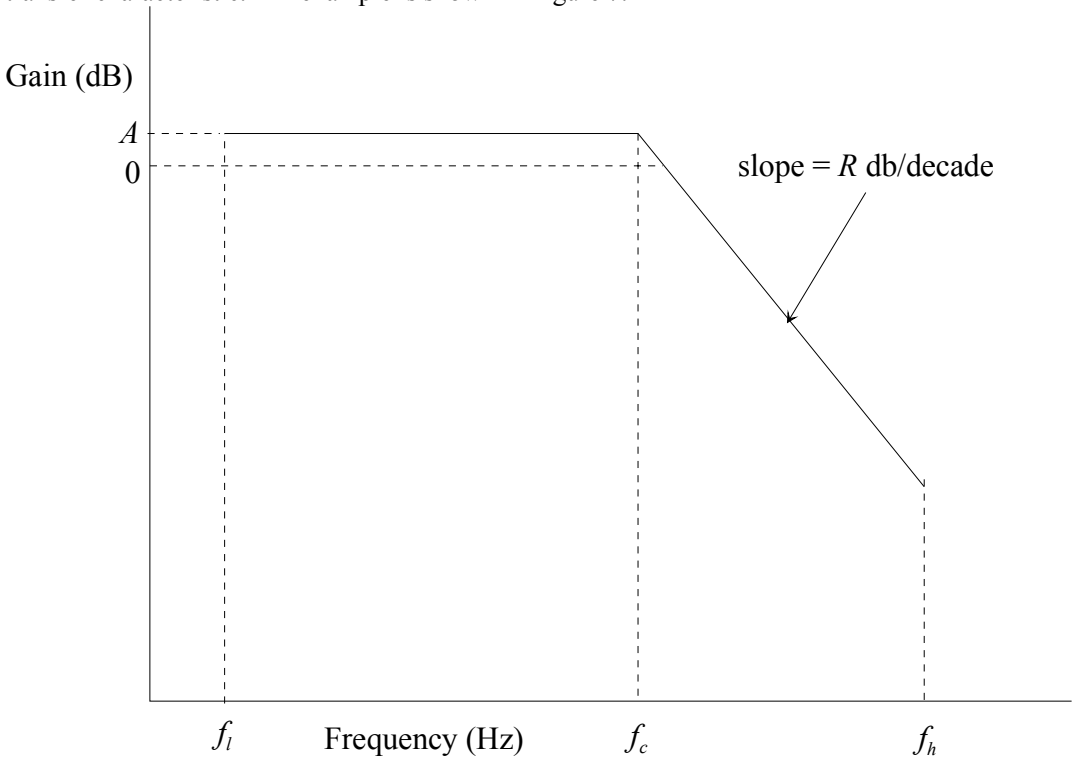


Figure 7. Example transfer requirement for filter, specific to application

A filter used for the application in question would be required to have a transfer characteristic that is below the mask in Figure 7. The filter would be tested by applying input signals of specified

amplitude and frequencies ranging from f_l to f_h, measuring the amplitude of the output, and plotting the gain (in dB) as a function of frequency. The result would have to be below the mask. The quantity A is the gain peaking, and the quantity f_c is the maximum bandwidth. Note that any implementation is allowed for the filter (digital versus analog, second order versus higher order, etc); the only requirement is that the filter transfer function meet the specified mask.

3. Message and Frame Formats

AVB networks will use the following six IEEE 1588 messages:

a) Sync
b) Follow_Up
c) Peer_Delay_Req (formerly called ADelay_Req)
d) Peer_Delay_Resp (formerly called ADelay_Resp)
e) Peer_Delay_Resp_FollowUp (formerly called ADelay_Resp_FollowUp)
f) Announce

Each message will have a standard Ethernet header (with or without 802.1Q tags) that precedes the PTP (i.e., IEEE 1588) payload. Note that AVB will not use Delay_Req and Delay_Resp messages; all AVB nodes will be required to process the Peer_Delay messages.

The message formats are shown in the following subsections. The material is taken mostly from [5] and [6]. One change from [5] and [6] is the fact that the Ethertype is indicated as m_0m_1 m_2m_3; this is intended to denote whatever Ethertype is assigned to frames that must be timestamped [8]. The Ethernet header and message payloads are shown separately for conciseness (i.e., to avoid repeating the header fields for each message). Note that the 4-octet frame check sequence (FCS) follows the PTP payload of each message (the FCS is not shown).

3.1 Ethernet Header (without 802.1Q tags)

N	Octet N	Octet N+1	Octet N+2	Octet N+3	Type (informative)	Field name
0	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Octet[6]	destination MAC address
4	h_8h_9	$h_{10}h_{11}$	k_0k_1	k_2k_3	Octet[6] (cont) | Octet[6]	destination MAC address (cont) | source MAC address
8	k_4k_5	k_6k_7	k_8k_9	$k_{10}k_{11}$	Octet[6] (cont)	source MAC address (cont)
12	m_0m_1	m_2m_3	N/A	N/A	UInteger16	type (this will be whatever Ethertype is assigned to frames that must be time stamped)

3.2 Ethernet Header (with 802.1Q tags)

N	Octet N	Octet N+1	Octet N+2	Octet N+3	Type (informative)	Field name
0	H_0h_1	h_2h_3	h_4h_5	h_6h_7	Octet[6]	destination MAC address
4	H_8h_9	$h_{10}h_{11}$	k_0k_1	k_2k_3	Octet[6] (cont) | Octet[6]	destination MAC address (cont) | source MAC address
8	K_4k_5	k_6k_7	k_8k_9	$k_{10}k_{11}$	Octet[6] (cont)	source MAC address (cont)
12	0x81	00	j_0j_1	j_2j_3	UInteger16 | UInteger16	type (0x8100 = tagged MAC frame) | tag control information
16	m_0m_1	m_2m_3	N/A	N/A	UInteger16	type (this will be whatever Ethertype is assigned to frames that must be time stamped)

3.3 Sync Payload

X = 14 for Ethernet without 802.1Q tags
X = 18 for Ethernet with 802.1Q tags

SOF	N	Octet N	Octet N+1	Octet N+2	Octet N+3	Type (informative)	Field name
X+0	0	h_0h_1	h_2h_3	k_0k_1	k_2k_3	UInteger4 | UInteger4 | UInteger8 | UInteger16	transportSpecific | messageID |versionPTP | reserved
X+4	4	h_0h_1	h_2h_3	k_0k_1	k_2k_3	UInteger16 | UInteger16	totalMessageLength | subdomain
X+8	8	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Octet[4]	flags
X+12	12	k_0k_1	k_2k_3	k_4k_5	k_6k_7	Integer64	correctionField
X+16	16	k_8k_9	$k_{10}k_{11}$	$k_{12}k_{13}$	$k_{14}k_{15}$	Integer64(cont)	correctionField(cont)
X+20	20	k_0k_1	j_0j_1	h_0h_1	h_2h_3	UInteger8 | UInteger8 |Octet[6]	reserved | sourceCommunicationTechnology | sourceUuid
X+24	24	h_4h_5	h_6h_7	h_8h_9	$h_{10}h_{11}$	Octet[6](cont)	sourceUuid(cont)
X+28	28	h_0h_1	h_2h_3	k_0k_1	k_2k_3	UInteger16 | UInteger16	sourcePortId | sequenceId
X+32	32	j_0j_1	00	h_0h_1	h_2h_3	UInteger8 | Octet | UInteger16	control | reserved | epochNumber
X+36	36	h_0h_1	h_2h_3	h_4h_5	h_6h_7	UInteger32	originTimestamp (seconds)
X+40	40	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Integer32	originTimestamp (nanoseconds)
X+44	44	k_0k_1	k_2k_3	N/A	N/A	Integer16	currentUTCOffset

3.4 Follow_Up Payload

X = 14 for Ethernet without 802.1Q tags
X = 18 for Ethernet with 802.1Q tags

Note that the Follow_Up payload differs from the Sync payload in that:

a) currentUTCOffset is not repeated in Follow_Up (it is only in Sync) [***Author's Note:*** *epochNumber must be added to the table below. It also must be decided whether to add currentUTCOffset to Followup (for the same reason it was added to Sync, namely that in theory it could change between an initial, less precise timestamp measurement when Sync is sent and the more precise later measurement.*]
b) Follow_Up has the associatedSequenceId of the corresponding Sync
c) Follow_Up has a preciseOriginTimestamp instead of an originTimestamp.

SOF	N	Octet N	Octet N+1	Octet N+2	Octet N+3	Type (informative)	Field name
X+0	0	h_0h_1	h_2h_3	k_0k_1	k_2k_3	UInteger4 | UInteger4 | UInteger8 | UInteger16	transportSpecific | messageID|versionPTP | reserved
X+4	4	h_0h_1	h_2h_3	k_0k_1	k_2k_3	UInteger16 | UInteger16	totalMessageLength | subdomain
X+8	8	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Octet[4]	flags
X+12	12	k_0k_1	k_2k_3	k_4k_5	k_6k_7	Integer64	correctionField
X+16	16	k_8k_9	$k_{10}k_{11}$	$k_{12}k_{13}$	$k_{14}k_{15}$	Integer64(cont)	correctionField(cont)

SOF	N	Octet N	Octet N+1	Octet N+2	Octet N+3	Type (informative)	Field name
X+20	20	k_0k_1	j_0j_1	h_0h_1	h_2h_3	UInteger8 | UInteger8 |Octet[6]	reserved | sourceCommunicationTechnology | sourceUuid
X+24	24	h_4h_5	h_6h_7	h_8h_9	$h_{10}h_{11}$	Octet[6](cont)	sourceUuid(cont)
X+28	28	h_0h_1	h_2h_3	k_0k_1	k_2k_3	UInteger16 | UInteger16	sourcePortId | sequenceId
X+32	32	j_0j_1	00	h_0h_1	h_2h_3	UInteger8 | Octet | UInteger16	control | reserved | associatedSequenceId
X+36	36	h_0h_1	h_2h_3	h_4h_5	h_6h_7	UInteger32	preciseOriginTimestamp (seconds)
X+40	40	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Integer32	preciseOriginTimestamp (nanoseconds)

3.5 Peer_Delay_Req Payload

To be supplied.

3.6 Peer_Delay_Resp Payload

To be supplied.

3.7 Peer_Delay_Resp_FollowUp Payload

To be supplied.

3.8 Announce Payload

To be supplied.

3.9 Definitions of selected Payload Fields

The definitions below are taken from [3], [5], and [7] and augmented by discussions in the February 22 – 24 IEEE 1588 face-to-face meeting.

Author's Note: *The definitions below are for Sync and Follow_Up message fields; modified versions may be necessary for the Peer_Delay messages when they are supplied.*

a) transportSpecific – not used by AVB networks (likely will be set to all zeros in AVB)
b) messageID – four-bit subtype that indicates the PTP message as indicated in Table 1 below. Note that not all the message types are used in AVB networks. Note also that all the messages that must be timestamped (i.e., the event messages) have the first bit of messageID set to zero (and therefore this bit can be used as an indicator to timestamping hardware of which messages must be timestamped [8].

Table 1. messageID for each PTP message

Category of message	Message	messageID value
Event	Sync	0
Event	Delay_Req	1
Event	Peer_Delay_Req	2
Event	Peer_Delay_Resp	3
Event	reserved	4-7
General	Follow_Up	8
General	Delay_Resp	9
General	Peer_Delay_Resp_FollowUp	10
General	Announce	11
General	PTP Management Message	12
General	To be completed	13-15

c) VersionPTP – version of the PTP standard implemented
d) TotalMessageLength – length of the PTP message payload
e) subdomain – the specific value for AVB is to be defined; note that an AVB network will consist of a single PTP subdomain.
f) flags – To be supplied (an incomplete set of flags is contained in Table 3 of [5])
g) correctionField – the field that the TCs use to accumulate the residence time
h) sourceCommunicationTechnology – indicates the communication medium and technology for the port that issues the PTP message. Initially AVB will focus on Ethernet, for which the value of this field is 1 (see Table 2 of [7] for a list of the various communications technologies recognized in IEEE 1588 Version 1)
i) sourceUuid – for AVB, Ethernet MAC address of the source of the message
j) sourcePortId – the ID of the port that is the source of the message. The ports on a network element with *N* ports are numbered from 1 to *N*.
k) sequenceId – an ID assigned to Sync and Follow_Up messages as they are transmitted on a BC or OC port. IDs are assigned sequentially on each transmitting port for event messages (i.e., Sync), and separately (but also sequentially) for general messages (i.e., Follow_Up and Announce). [***Author's Note:*** *Peer_Delay messages will be covered by this definition when their formats are supplied.*]
l) control – not used in AVB (retained for backward compatibility with IEEE 1588 Version 1; indicates the Version 1 message type (see Table 28 of [7] for allowable values).
m) epochNumber – when the epoch is the PTP epoch, the epochNumber is the total number of times the 32-bit seconds counter has rolled over since the PTP epoch.[6] More generally, the epochNumber may be treated as the most significant part of the total number of seconds since the epoch (the least significant part is the 32-bit integer seconds portion of the PTP timestamp). See Section 6.2.5.7 and Appendix B of [7] for more detail.
n) originTimestamp (seconds) – the seconds portion of the timestamp that carries any timestamp measurement made on-the-fly by the master BC or OC that issues the Sync message. AVB is not required to make on-the-fly measurements and, if it does make them, there is no requirement that they be precise (i.e., AVB may still use Follow_Up messages in this case). However, AVB is allowed to make precise timestamp measurements on-the-fly and not use Follow_Up.
o) originTimestamp (nanoseconds) - the nanoseconds portion of the timestamp that carries any timestamp measurement made on-the-fly by the master BC or OC that issues the Sync message. AVB is not required to make on-the-fly measurements and, if it does make them,

[6] In IEEE 1588, the term *epoch* is defined as the reference time that defines the origin of a timescale. For PTP, the epoch is 0:00:00 on 1 January 1970 (see Appendix B, Table B.2 of [7]).

there is no requirement that they be precise (i.e., AVB may still use Follow_Up messages in this case). However, AVB is allowed to make precise timestamp measurements on-the-fly and not use Follow_Up.

p) CurrentUTCOffset – the offset between the UTC and TAI timescales at the master BC or OC that issues the Sync or Followup message.
q) AssociatedSequenceId – the sequenceId of the Sync message that corresponds to this Follow_Up message.
r) preciseOriginTimestamp (seconds) – the seconds portion of the more precise timestamp measurement carried in a Follow_Up message.
s) preciseOriginTimestamp (nanoseconds) - the nanoseconds portion of the more precise timestamp measurement carried in a Follow_Up message.

4. Processing of PTP Messages

4.1 Sync and Follow_Up Messages

The rules below were obtained by modifying the rules in [3] and [4] to apply to the case of a P2P TC colocated with a GM or slave OC. We could have alternatively left the rules in [4] as is by defining a logical link, internal to a node, between the OC and PTP TC functions in the node. The rules below capture the behavior external to the node without the need to define an internal, logical link.

1) As a result of executing the BMC algorithm, exactly one OC in the network will be the GM. All the other OCs will be slaves. All the ports of the GM are in the master state with respect to Sync and Follow_Up (and Announce) messages. All the ports of the slaves are in the slave state with respect to Sync and Follow_Up (and Announce) messages.[7]
2) Sync can originate only at a master port (i.e., only a GM port), and can egress the network only at a slave port.
3) If Follow_Up is transmitted on a port without a corresponding Sync and Follow_Up having been received on another port, then the port Follow_Up is transmitted on must be a master port.
4) The GM sends Sync on each respective port (i.e., on all ports indicated by the forwarding data base such that the multicast Sync reaches all slaves) with the follow-up flag (i.e., the PTP_ASSIST flag) set and the correction field initialized to zero, and measures the time of departure of Sync on each port
5) The GM sends Follow_Up on each respective port with the correction field initialized to zero and the preciseOrigin timestamp equal to the measured time of departure of the Sync message on that port
6) Each port that receives a Sync message measures its arrival time. This arrival time is used for both (a) computation of the residence time for the case where the node is not the destination, and (b) computation of the slave offset for the case where the node is the destination.
7) When a slave (i.e., port in the slave state) receives the Followup message corresponding to a Sync message for which this slave is its destination, it adds the correction fields in the Sync

[7] The master and slave states of ports are meaningful to Sync, Follow_Up, and Announce messages. This is because an AVB node can be decomposed functionally into an OC function and a P2P TC function, as in the left-hand illustration of Figure 1. In that illustration, the link connecting the OC and P2P TC is in either the master or slave state. The master and slave states are not meaningful to the Peer_Delay messages, because P2P TC ports are stateless. In the left-hand illustration of Figure 1, the Peer_Delay messages are transmitted or received on the ports that emanate horizontally from the P2P TC, but not on the link to the OC.

and Followup messages and the propagation time on the link on which the Sync arrived to the preciseOrigin Timestamp in the Followup message. The result is subtracted from the arrival time of the Sync message to obtain the slave offset.

8) When a Sync message arrives at a node that is not its destination, it is held until the corresponding Follow_Up message arrives.
9) On arrival of the Followup message corresponding to the Sync message in (8), the correction field of the Followup Message is added to the correction field of the Sync message
10) After performing the computation in (9), the Sync message is sent on the respective ports indicated by the forwarding data base for multicast PTP messages, and its departure time on each port is measured.
11) The residence time for the Sync message is computed on each port, and is placed in the correction field of a respective new Follow_Up message generated for each port.
12) The propagation time for the link on which the Sync message arrived is added to the correction field of the Follow_Up message.
13) Each new Followup message is sent on the respective port for which it was generated

4.2 Peer_Delay Messages

To be supplied.

4.3 Announce Message

To be supplied

5. References

[1] IEEE Standard for Local and Metropolitan Area Networks – Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks, Draft 802.1as PAR forwarded by IEEE 802 EC to NesCom on March 10, 2006.

[2] IEEE 1588/802.1 AVB Design Meeting, February 21, 2006 (see meeting minutes).

[3] *Transparent Clock – Working Technical Description*, Revision 13, IEEE 1588 TC Subcommittee, prepared during February 22 – 24, 2006 IEEE 1588 Face-to-Face Meeting.

[4] Geoffrey M. Garner, *Initial Description of Possible Use of Sync and Followup Messages with Peer-to-Peer Transparent Clocks in AVB*, Samsung Contribution to IEEE 1588 and IEEE 802.1 AVB TG, Revision 1.0, March 1, 2006 (plus accompanying VG presentation dated March 6, 2006 and presented at March, 2006 802.1 meeting).

[5] Dave Tonks, *Variable Length Unified Frames, Formats and Contents*, Version 1.3, IEEE 1588 Short Frames Subcommittee, February 22, 2006.

[6] Ron Cohen and Silvana Rodrigues, *Unifying Short and Long Messages*, Contribution to Precise Networked Clock Synchronization Working Group – IEEE 1588 revision, November 30, 2005.

[7] IEEE 1588, *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and control Systems*, September 12, 2002.

[8] IEEE 1588 Timestamp Subcommittee calls through March 21, 2006.

[9] John Eidson and Bruce Hamilton, *PTP Clock Synchronization Model*, Draft material provided to IEEE 1588 Rewrite Subcommittee for Clause 6 of IEEE 1588, Version 2, March, 2006.

[10] Geoffrey M. Garner, *End-to-End Jitter and Wander Requirements for ResE Applications*, Samsung Presentation for IEEE 802.3 ResE SG, May 16, 2005.