5. Architecture overview

5.1 Application scenarios

5.1.1 Garage jam session

As an illustrative example, consider AVB usage for a garage jam session, as illustrated in Figure 5.1. The audio inputs (microphone and guitar) are converted, passed through a guitar effects processor, two bridges, mixed within an audio console, return through two bridges, and return to the ear through headphones.

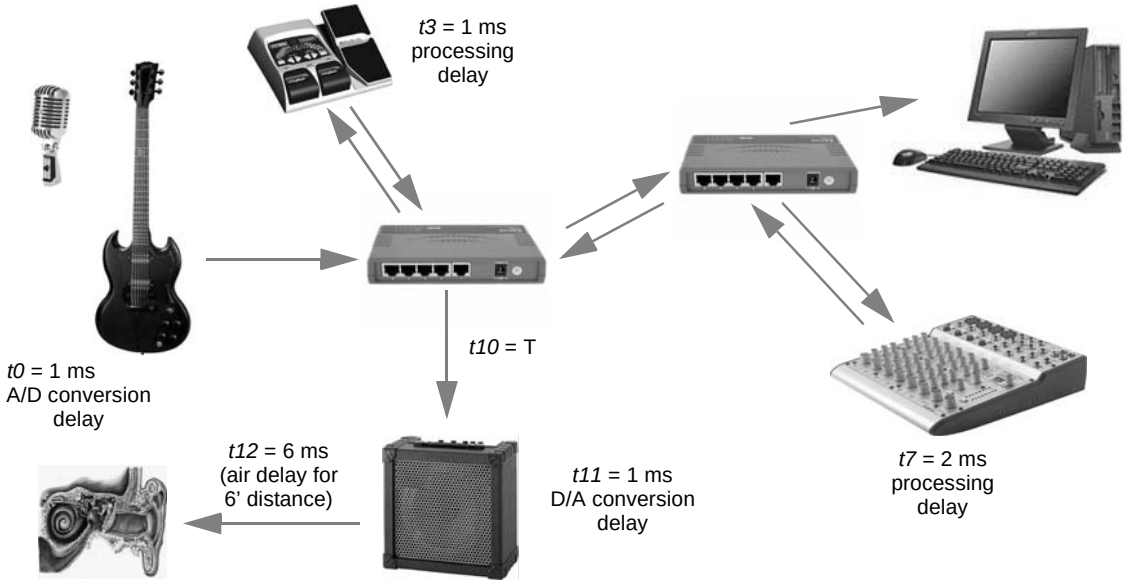

Figure 5.1—Garage jam session

Using Ethernet within such systems has multiple challenges: low-latency and tight time-synchronization. Tight time synchronization is necessary to avoid cycle slips when passing through multiple processing components and (ultimately) to avoid under-run/over-run at the final D/A converter's FIFO. The challenge of low-latency transfers is being addressed in other forums and is outside the scope of this draft.

5.1.2 Looping topologies

Bridged Ethernet networks currently have no loops, but bridging extensions are contemplating looping topologies. To ensure longevity of this standard, the time-synchronization protocols are tolerant of looping topologies that could occur (for example) if the dotted-line link were to be connected in Figure 5.2.

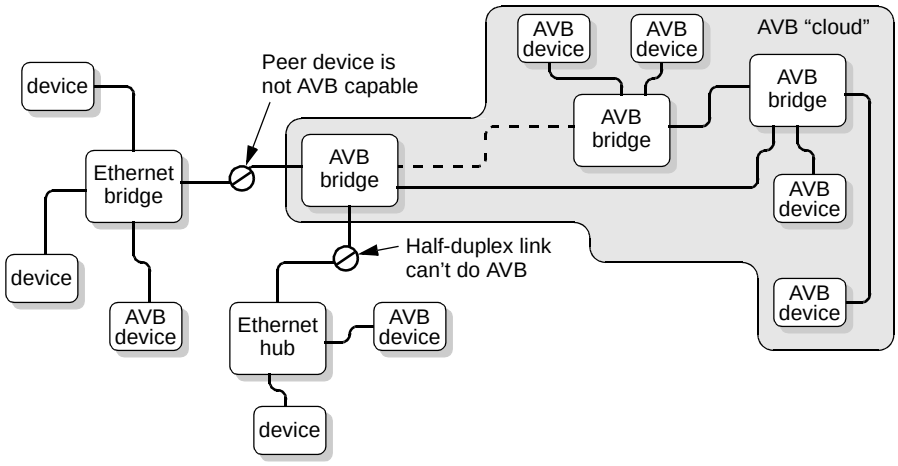

Figure 5.2—Possible looping topology

Separation of AVB devices is driven by the requirements of AVB bridges to support subscription (bandwidth allocation) and pacing of time-sensitive transmissions, as well as time-of-day clock-synchronization.

5.2 Design methodology

5.2.1 Assumptions

This working paper specifies a protocol to synchronize independent timers running on separate stations of a distributed networked system, based on concepts specified within IEEE Std 1588-2002. Although a high degree of accuracy and precision is specified, the technology is applicable to low-cost consumer devices. The protocols are based on the following design assumptions:

a) Each end station and intermediate bridges provide independent clocks.

b) All clocks are accurate, typically to within ±100PPM.

c) Details of the best time-synchronization protocols are physical-layer dependent.

5.2.2 Objectives

With these assumptions in mind, the time synchronization objectives include the following:

a) Precise. Multiple timers can be synchronized to within 10's of nanoseconds.

b) Inexpensive. For consumer AVB devices, the costs of synchronized timers are minimal. (GPS, atomic clocks, or 1PPM clock accuracies would be inconsistent with this criteria.)

c) Scalable. The protocol is independent of the networking technology. In particular:

   1) Cyclical physical topologies are supported.
   2) Long distance links (up to 2 kM) are allowed.

d) Plug-and-play. The system topology is self-configuring; no system administrator is required.

5.2.3 Strategies

Strategies used to meet these objectives include the following:

a) Precision is achieved by calibrating and adjusting *grandTime* clocks.
   1) Offsets. Offset value adjustments eliminate immediate clock-value errors.
   2) Rates. Rate value adjustments reduce long-term clock-drift errors.

b) Simplicity is achieved by the following:
   1) Concurrence. Most configuration and adjustment operations are performed concurrently.
   2) Feed-forward. PLLs are unnecessary within bridges, but possible within applications.
   3) Frequent. Frequent (nominally 100 Hz) interchanges reduces needs for overly precise clocks.

5.3 Grand-master selection

5.3.1 Grand-master overview

Clock synchronization involves streaming of timing information from a grand-master timer to one or more slave timers. Although primarily intended for non-cyclical physical topologies (see Figure 5.3a), the synchronization protocols also function correctly on cyclical physical topologies (see Figure 5.3b), by activating only a non-cyclical subset of the physical topology.

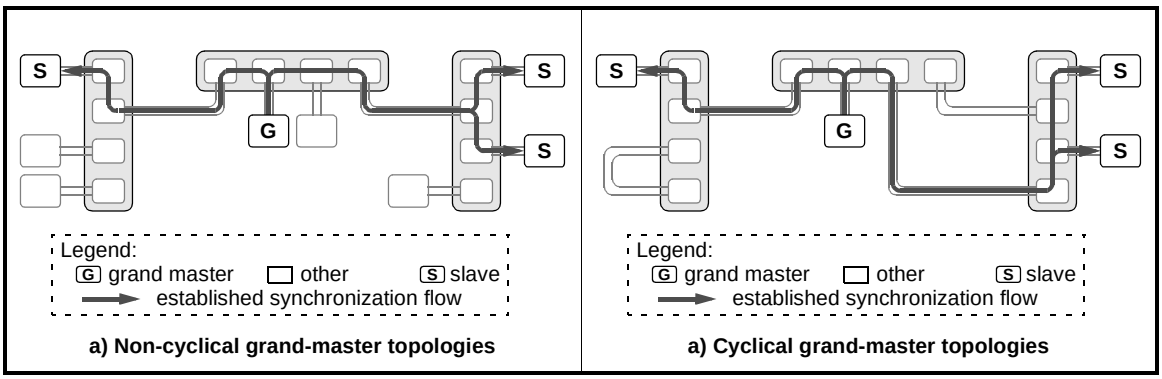

Figure 5.3—Timing information flows

In concept, the clock-synchronization protocol starts with the selection of the reference-timer station, called a grand-master station (oftentimes abbreviated as grand-master). Every AVB-capable station is grand-master capable, but only one is selected to become the grand-master station within each network. To assist in the grand-master selection, each station is associated with a distinct preference value; the grand-master is the station with the “best” preference values. Thus, time-synchronization services involve two subservices, as listed below and described in the following subclauses.

a) Selection. Looping topologies are isolated (from a time-synchronization perspective) into a spanning tree. The root of the tree, which provides the time reference to others, is the grand master.
b) Distribution. Synchronized time is distributed through the grand-master’s spanning tree.

5.3.2 Grand-master selection

As part of the grand-master selection process, stations forward the best of their observed preference values to neighbor stations, allowing the overall best-preference value to be ultimately selected and known by all.

The station whose preference value matches the overall best-preference value ultimately becomes the grand-master.

The grand-master station observes that its precedence is better than values received from its neighbors, as illustrated in Figure 5.4a. A slave stations observes its precedence to be worse than one of its neighbors and forwards the best-neighbor precedence value to adjacent stations, as illustrated in Figure 5.4b. To avoid cyclical behaviors, a *hopCount* value is associated with preference values and is incremented before the best-precedence value is communicated to others.

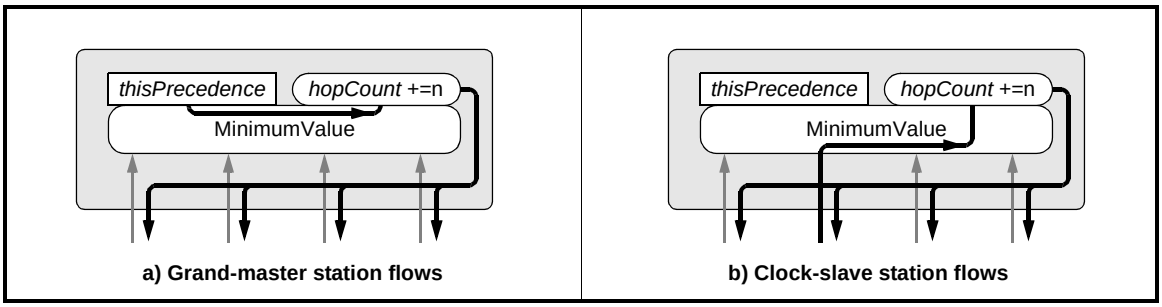

Figure 5.4—Grand-master precedence flows

When stabilized, the value of *n* equals one and the *hopCount* value reflects the distance between this station and its grand master, in units of hops-between-bridges. Other values are used to quickly stabilize systems with rogue frames, as summarized in Equation 5.1.

```
#define HOPS 255                                                    (5.1)
n = (frame.hopCount > hopCount) ? (HOPS - frame.hopCount) / 2 : 1;
```

NOTE—A rogue frame circulates at a high precedence, in a looping manner, where the source stations is no longer present (or no longer active) and therefore cannot remove the circulating frame. The super-linear increase in *n* is intended to quickly scrub rogue frames, when the circulation loop consists of less than HOPS stations.

5.3.3 Grand-master preference

Grand-master preference is based on the concatenation of multiple fields, as illustrated in Figure 5.5. The *port* value is used within bridges, but is not transmitted between stations.

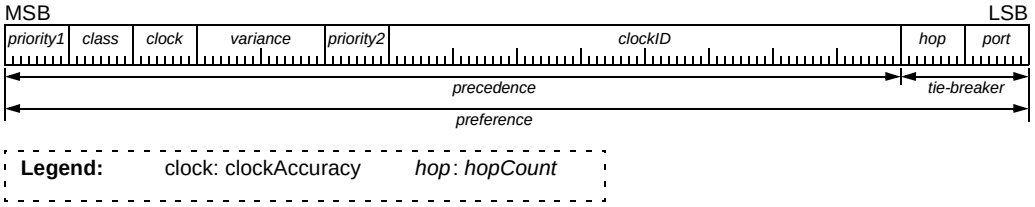

Figure 5.5—Grand-master preference

This format is similar to the format of the spanning-tree precedence value, but a wider *clockID* is provided for compatibility with interconnects based on 64-bit station identifiers.

5.4 Synchronized-time distribution

5.4.1 Hierarchical grand masters

Clock-synchronization information conceptually flows from a grand-master station to clock-slave stations, as illustrated in Figure 5.6a. A more detailed illustration shows pairs of synchronized clock-master and clock-slave components, as illustrated in Figure 5.6b. The active clock agents are illustrated as black-and-white components; the passive clock agents are illustrated as grey-and-white components.

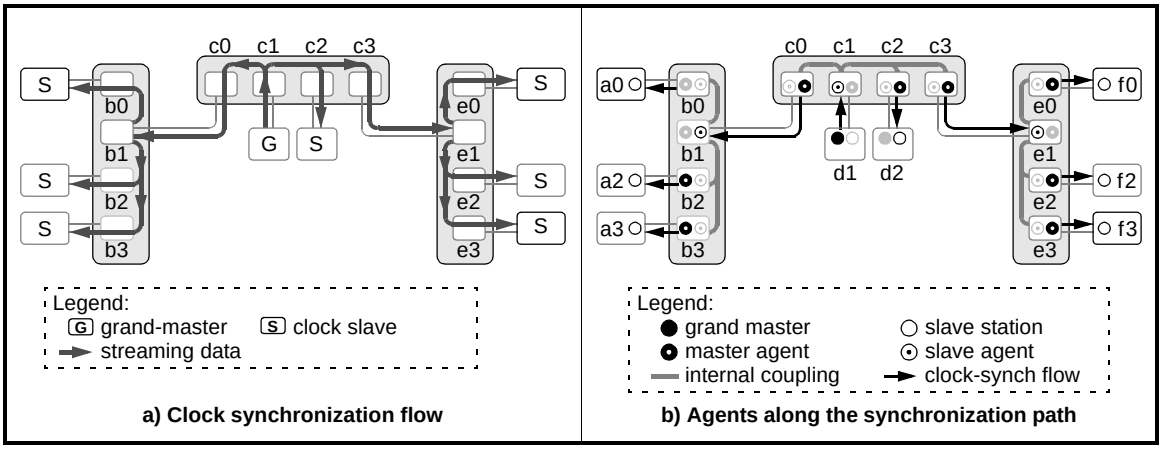

Figure 5.6—Hierarchical flows

Internal communications distribute synchronized time from clock-slave agents b1, c1, and e1 to the other clock-master agents on bridgeB, bridgeC, and bridgeE respectively. Within a clock-slave, precise time synchronization involves adjustments of timer value and rate-of-change values.

Time synchronization yields distributed but closely-matched *grandTime* values within stations and bridges. No attempt is made to eliminate intermediate jitter with bridge-resident jitter-reducing phase-lock loops (PLLs,) but application-level phase locked loops (not illustrated) are expected to filter high-frequency jitter from the supplied *grandTime* values.

5.4.2 Time-synchronization flows

Time-reference information is created at a ClockSource entity, flows through multiple intermediate entities, and is consumed at one or more ClockSink entities, as illustrated in Figure 5.7. Within this illustration, the clock-master station (containing the ClockSource entity) and the clock-slave station (containing the ClockSink entity) are illustrated as multipurpose bridges. Either of the ClockMaster and ClockSlave stations could also be end stations (not illustrated), wherein no MAC-relay functionality is required.

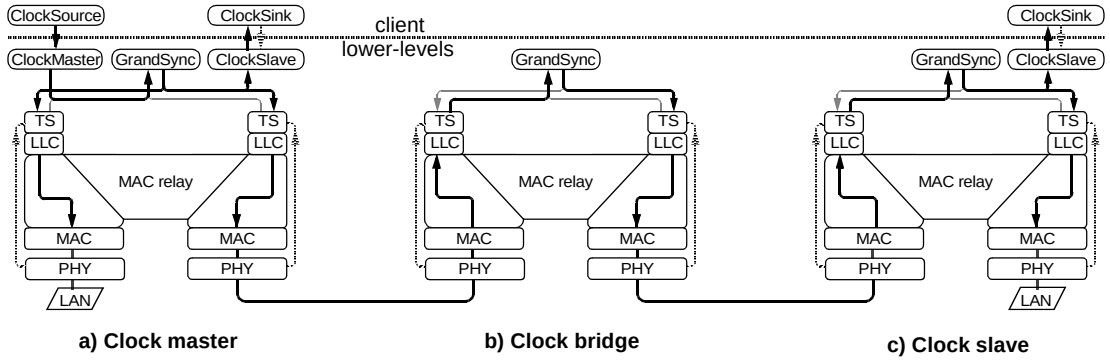

Figure 5.7—Time-synchronization flows

5.4.2.1 Clock-master flows

Referring now to the clock-master (Figure 5.7-a) station. This clock-master station comprises client-level ClockSink as well as ClockSource entities. The ClockSink entity is provided so that the client-clock can be synchronized to the network clock, whenever another station is selected to become that grand-master. (The ClockSource entity on the grand-master station provides the network-synchronized time reference.)

The ClockSource time-reference interfaces indirectly to the GrandSync entity via a ClockMaster entity. The ClockMaster entity supplements the clock-synchronization provided by the ClockSource entity with additional information (such as the grand-master precedence) that is needed by the GrandSync entity.

The GrandSync entity is responsible for selecting the preferred time-reference port from among the possible direct-attached ClockSource and bus-bridge-port entities. The selection is based on user-preference, clock-property, topology, and unique-clock-identifier information.

The GrandSync entity echoes the time-synchronization information from (what it determines to be) the preferred port. Information from lower-preference ports is continuously monitored to detect preference changes (typically due to attach or detach of clock-master capable stations). In the absence of such changes, time-reference information in messages from lower-preference ports is ignored.

The GrandSync entity's echoed time-reference information is observed by the directly-attached ClockSlave and bridge-port entities. The information forms the basis for the time-synchronization information forwarded to other indirectly-attached ClockSlave entities through this station's bus-bridge ports

5.4.2.2 Bus-bridge flows

Referring now to the bus-bridge (Figure 5.7-b) station. This bus-bridge station comprises port and GrandSync entities. Both ports are responsible for forwarding their received time-reference information to the GransSync entity.

The bus bridge's GrandSync entity is responsible for selecting the preferred time-reference port. The selection is based on user-preference, clock-property, topology, and unique-clock-identifier information provided indirectly by remote ClockSource entities.

The GrandSync entity echoes the time-synchronization information from (what it determines to be) the preferred port. Information from lower-preference ports is continuously monitored to detect preference changes (typically due to attach or detach of clock-master capable stations). In the absence of such changes, time-reference information in messages from lower-preference ports is ignored.

The GrandSync entity's echoed time-reference information is observed by all bridge-port entities (including the source port). The information forms the basis for the time-synchronization information forwarded to other indirectly-attached ClockSlave entities through this station's bus-bridge ports

5.4.2.3 Clock-slave flows

Referring now to the clock-slave (Figure 5.7-c) station. This clock-slave station comprises port, GrandSync, and ClockSlave entities, as well as a client-level ClockSink entity. All ports are responsible for forwarding their received time-reference information to the GransSync entity.

As always, the GrandSync entity is responsible for selecting the preferred time-reference port from among the possible direct-attached ClockSource and bus-bridge-port entities. The GrandSync entity echoes the time-synchronization information from (what it determines to be) the preferred port.

The GrandSync entity's echoed time-reference information is observed by the station-local ClockSlave entity. The ClockSlave entity removes the extraneous grand-master preference information and re-times its transmissions to match the client's time-request rate. The time-reference information is then passed to the ClockSink client.

5.4.2.4 Time-stamp flows

Referring now to the hashed PHY-to-TS lines within Figure 5.7 stations. Maintaining an accurate time reference relies on the presence of accurate time-stamp hardware capabilities in or near the media-dependent PHY. A bypass path is thus required at the receiver, so that the time-stamp can be affiliated with the arriving timeSync information, before the SDU or service-interface parameters are processed by the time-synchronization (TS) entity above the MAC.

A similar bypass path is also required at the transmitter, so that the time-stamp of a transmitted frame can become known to the time-synchronization (TS) entity above the MAC. For simplicity and convenience, this time-stamp information is not placed into the transmitted frame, but (via processing by the time-synchronization entity) can be placed within later transmissions.

5.5 Sampling offset/rate conversion

Each clock-master port is responsible for using its received {*grandTime*, *stationTime*} and converting them into the distinct {*grandTimed*, *stationTimed*} affiliations that are transmitted to its neighbor. Since the values of *stationTime* and *stationTimed* are (by convention) coupled to the receive and transmit times, this update involves computation of the next *grandTimed* value.

5.5.1 Forward extrapolation (background)

A typical design approach (and that used by IEEE Std 1588) views the received {*grandTime*, *stationTime*} affiliations as points on a curve, sampled at received-snapshot times *rx*[*n*]. The objective is to generate the distinct set of {*grandTimed*, *tx*[*m*]} affiliations by extrapolating from a distinct set of receive-snapshot times *rx*[*n*], as illustrated in Figure 5.8.

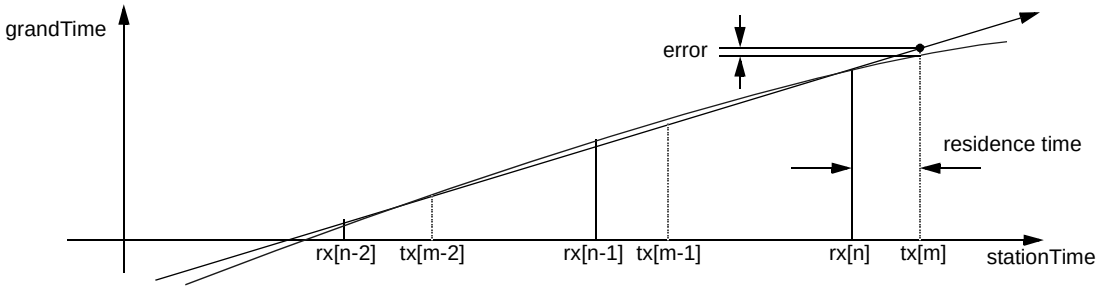

Figure 5.8—Forward extrapolation

$$grandTimed[m] = grandTime[n] + rxSlope * (tx[m] - rx[n]); \quad (5.2)$$

Where:

grandTimed[*m*] is the value for the to-be-transmitted {*grandTimed*[*m*], *tx*[*m*]} affiliation.
grandTime[*n*] is the value from the previously received {*grandTime*[*n*], *rx*[*n*]} affiliation.
rxSlope is the value of slope of previously sampled values, specified by Equation 5.3.

$$rxSlope = (grandTime[n\text{-N}] - grandTime[n]) / (rx[n] - rx[n\text{-N}]) \quad (5.3)$$

Where:

grandTime[*n*] is the value from the previously received {*grandTime*[*n*], *rx*[*n*]} affiliation.

Within such systems, the difference between *rx*[*n*] and *tx*[*m*] times is based on the residence time, delays associated with processing of timeSync information. To obtain more-accurate results, the protocols/designs attempt to reduce this residence time to values that are significantly less than the sample-to-sample interval. This technique has several limitations:

a) Much of the residence-time value is based on the execution of slow-rate (but flexible) firmware:
   1) The firmware-execution times are longer than latencies normally associated with hardware.
   2) The firmware-execution engine is shared and the interrupt latencies can be significant.
   3) Standard lower-priority queues can delay transmissions in favor of latency-sensitive traffic.

b) These algorithms assume identical/synchronized average *rx*[*n*] and *tx*[*m*] sampling rates:
   1) The optimal *rx*[*n*] and *tx*[*m*] sampling rates are not necessarily identical.
   2) Clock-slave designs are complicated by such pseudo-synchronous operation requirements.

c) Cascaded systems suffer a transient phenomenon called whiplash or gain-peaking, depending on how the phenomenon is observed.

A whiplash effect is visible as ringing after a injected spike and/or a step change in frequency. The gain-peaking effect is visible as a frequency gain, that becomes increasingly larger through cascaded PLLs, for selected frequencies. For basic cascaded PLLs, this phenomenon is unavoidable, although its effects can be reduced through careful design or manual tuning of peaking frequencies.

The whiplash/gain-peaking characteristics are caused by extrapolating samples {*rx*[*n*], …*rx*[*n*-N]} to future times *tx*[*m*], wherein an *error* (difference between actual and predicted values) is introduced. This error is correlated to the signal and (for certain error-signal frequencies), the amplitude of the transmit error exceeds the amplitude of the receive error. To avoid the preceding list of concerns, a distinct backward-interpolation design approach is used.

5.5.2 Backward interpolation

A more-scalable backward-interpolation approach also views the received {*grandTime*, *stationTime*} affiliations as points on a curve, sampled at received-snapshot times *rx*[*n*]. However, the objective is to generate the distinct set of {*grandTimed*, *tx*[*m*]} affiliations by interpolating within a distinct set of receive-snapshot times *rx*[*n*], as illustrated in Figure 5.9.

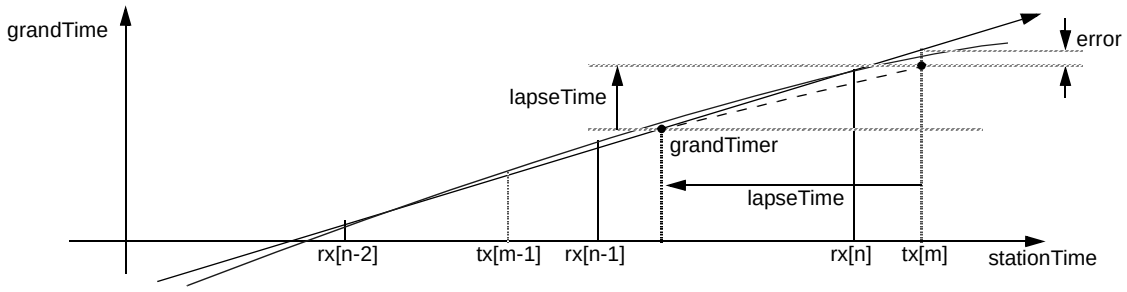

Figure 5.9—Backward interpolation

grandTimed[*m*] = *grandTimer*[*m*] + *lapseTime*; (5.4)

Where:

lapseTime is a constant (average sample rates dependent) *lapseTime* value.

grandTimer[*m*] is corresponds to an interpolated {*grandTime*[*m*], *tx*[*m*]–*lapseTime*) affiliation.

error[*m*] = (*rxSlope* - ONE) * *lapseTime*; (5.5)

The advantage of this technique is the separation of *grandTimed*[*m*] and *error*[*m*] components. The interpolation process eliminates gain-peaking for the *grandTime*[*m*] value, thus reducing error effects when passing through multiple bridges. The sideband *error* signal remains significant, and is therefore carried through bridges, so that the cumulative *grandTimed*[*m*]+*error*[*m*] value can be passed to the end-point application.

From an intuitive perspective, the whiplash-free nature of the back-in-time interpolation is attributed to the use of interpolation (as opposed to extrapolation) protocols. Interpolation between input values never produces a larger output value, as would be implied by a gain-peaking (larger-than-unity gain) algorithm. A disadvantage of back-in-time interpolation is the requirement for a side-band *errorTime* communication channel, over which the difference between nominal and rate-normalized *lapseTime* values can be transmitted.

5.6 Distinctions from IEEE Std 1588

Advantageous properties of this protocol that distinguish it from other protocols (including portions of IEEE Std 1588) include the following:

a) Synchronization between grand-master and local clocks occurs at each station:
   1) All bridges have a lightly filtered synchronized image of the grand-master time.
   2) End-point stations have a heavily filtered synchronized image of the grand-master time.

b) Time is uniformly represented as scaled integers, wherein 40-bits represent fractions-of-a-second.
   1) Grand-master time specifies seconds within a more-significant 40-bit field.
   2) Local time specifies seconds within a more-significant 8-bit field.

c) Locally media-dependent synchronized networks don't require extra time-snapshot hardware.

d) Error magnitudes are linear with hop distances; PLL-whiplash and $O(n^2)$ errors are avoided.

e) Multicast (one-to-many) services are not required; only nearest-neighbor addressing is assumed.

f) A relatively frequent 100 Hz (as compared to 1 Hz) update frequency is assumed:
   1) This rate can be readily implemented (in today's technology) for minimal cost.
   2) The more-frequent rate improves accuracy and reduces transient-recovery delays.
   3) The more-frequent rate reduces transient-recovery delays.

g) Only one frame type simplifies the protocols and reduces transient-recovery times. Specifically:
   1) Cable delay is computed at a fast rate, allowing clock-slave errors to be better averaged.
   2) Rogue frames are quickly scrubbed (2.6 seconds maximum, for 256 stations).
   3) Drift-induced errors are greatly reduced.

6. GrandSync operation

6.1 Overview

6.1.1 GrandSync behavior

This clause specifies the state machines that specify GrandSync-entity processing. The operations are described in an abstract way and do not imply any particular implementations or any exposed interfaces. There is not necessarily a one-to-one correspondence between the primitives and formal procedures and the interfaces in any particular implementation.

The GrandSync entity is responsible for observing time-sync related MS_UNITDATA.indication service primitives, selectively echoing these service-primitive parameters in associated MS_UNITDATA.request parameters, as follows:

a) When a preferred time-sync related MS_UNITDATA.indication message arrives:
   1) The grand-master preference and port-timeout parameters are saved.
   2) MS_UNITDATA.indication parameters are echoed in MS_UNITDATA.request parameters.
   3) The arrival time is recorded, for the purpose of monitoring port timeouts.

b) Arriving non-preferred MS_UNITDATA.indication messages are discarded.
   The intent is to echo only messages from the currently selected grand-master port.

c) If the preferred-port timeout is exceeded, the preferred-port parameters are reset.
   The intent is to restart grand-master selection based on the remaining candidate ports.

6.1.2 GrandSync interface model

The time-synchronization service model assumes the presence of one or more time-synchronized AVB ports communicating with a MAC relay, as illustrated in Figure 6.1. All components are assumed to have access to a common free-running (not adjustable) *localTime* value.

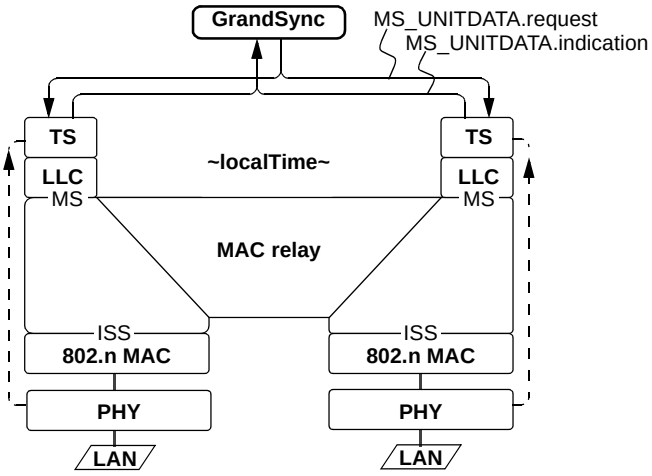

Figure 6.1—GrandSync interface model

A received MAC frame is associated with link-dependent timing information, processed within the TimeSync (TS) state machine, and passed to the GrandSync protocol entity. The GrandSync state machine (illustrated with a darker boundary) is responsible for saving time parameters from observed MS_UNITDATA.indication parameters and generating MS_UNITDATA.request parameters for delivery to other ports.

The preference of the time-sync messages determines whether the message content is ignored by the GrandSync protocol entity or modified and redistributed to the attached TS state machines. The sequencing of this state machine is specified by Table 6.1; details of the computations are specified by the C-code of Annex G.

Information exchanged with the GrandSync entity includes a source-*port* identifier, *hops*&*precedence* information for grand-master selection, a globally synchronized *grandTime*, a station-local *snapTime*, and a cumulative *errorTime*, as illustrated in Figure 6.2. A clock-slave end-point can filter the sum of *grandTime* and *errorTime* values, thereby yielding its image of the globally synchronized *grandTime* value.

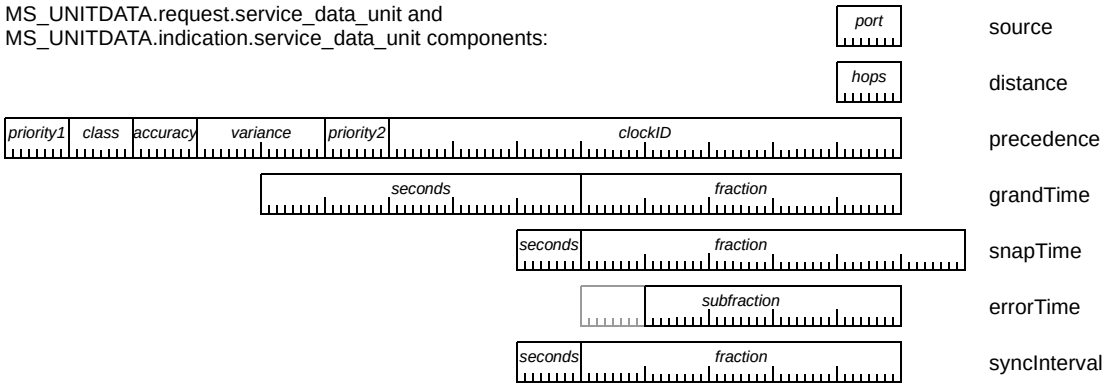

Figure 6.2—GrandSync service-interface components

NOTE—The *syncInterval* value is relative static and could (if desired) be communicated by access to port-specific resources. If this alternative configuration mechanism is preferred, this content will be removed from the service interface contents.

NOTE—The *snapTime* value has additional precision, when compared to the similar externally visible localTime value, to minimize the effects of numerical rounding when transferring values between computational entities within the bridge.

6.2 Service interface primitives

6.2.1 MS_UNITDATA.indication

6.2.1.1 Function

Provides the GrandSync protocol entity with clock-synchronization parameters derived from activities on the attached media-dependent ports. The information is sufficient to identify a single clock-slave port (typically the closest-to-grand-master port) and to disseminate grand-master supplied clock-synchronization information to other ports.

6.2.1.2 Semantics of the service primitive

The semantics of the primitives are as follows:

```
MS_UNITDATA.indication {
    destination_address,      // Destination address
    source_address,           // Optional
    priority,                 // Forwarding priority
    service_data_unit,        // Delivered content
    {                         // Contents of the service_data_unit
        protocolType,         // Distinguishes AVB frames from others
        function,             // Distinguishes between timeSync and other AVB frames
        version,              // Distinguishes between timeSync frame versions
        precedence,           // Precedence for grand-master selection
        grandTime,            // Global-time snapshot (1-cycle delayed)
        errorTime,            // Accumulated grandTime error
        sourcePort,           // Identifies the source port
        hopCount,             // Distance from the grand-master station
        snapTime,             // Local-time snapshot (1-cycle delayed)
        syncInterval          // Nominal timeSync transmission interval
    }
}
```

NOTE—The *grandTime* field has a range of approximately 36,000 years, far exceeding expected equipment life-spans. The *localTime* and *linkTime* fields have a range of 256 seconds, far exceeding the expected timeSync frame transmission interval. These fields have a 1 pico-second resolution, more precise than the expected hardware snapshot capabilities. Future time-field extensions are therefore unlikely to be necessary in the future.

The parameters of the MA_DATA.indication are described as follows:

6.2.1.2.1 destination_address: A 48-bit field that allows the frame to be conveniently stripped by its downstream neighbor. The destination_address field contains an otherwise-reserved group 48-bit MAC address (TBD).

6.2.1.2.2 source_address: A 48-bit field that specifies the local station sending the frame. The source_address field contains an individual 48-bit MAC address (see 3.10), as specified in 9.2 of IEEE Std 802-2001.

6.2.1.2.3 priority: Specifies the priority associated with content delivery.

6.2.1.2.4 service_data_unit: A multi-byte field that provides information content.

For GrandSync-entity time-sync interchanges, the service_data_unit consists of the following subfields:

6.2.1.2.5 *protocolType*: A 16-bit field contained within the payload that identifies the format and function of the following fields.

6.2.1.2.6 *function*: An 8-bit field that distinguishes the timeSync frame from other AVB frame type.

6.2.1.2.7 *version*: An 8-bit field that identifies the version number associated with of the following fields. TBD—A more exact definition of version is needed.

6.2.1.2.8 *precedence*: A 14-byte field that specifies grand-master selection precedence (see 6.2.1.4).

6.2.1.2.9 *grandTime*: An 80-bit field that specifies a grand-master synchronized time (see 6.2.1.6).

6.2.1.2.10 *errorTime*: A 32-bit field that specifies the cumulative grand-master synchronized-time error. (Propagating *errorTime* and *grandTime* separately eliminates whiplash associated with cascaded PLLs.)

6.2.1.2.11 *sourcePort*: An 8-bit field that identifies the port that sourced the encapsulating content.

6.2.1.2.12 *hopCount*: An 8-bit field that identifies the maximum number of hops between the talker and associated listeners.

6.2.1.2.13 *snapTime*: A 56-bit field that specifies the local free-running time within this station, when the previous timeSync frame was received (see 6.2.1.8).

6.2.1.2.14 *syncInterval*: A 48-bit field that specifies the nominal period between timeSync frame transmissions.

NOTE—The *syncInterval* value is a port-specific constant value which (for apparent simplicity) has been illustrated as a relayed frame parameter. Other abstract communication techniques (such as access to shared design constants) might be selected to communicate this information, if requested by reviewers for consistency with existing specification methodologies.

6.2.1.3 Version format

For compatibility with existing 1588 time-snapshot, a single bit within the version field is constrained to be zero, as illustrated in Figure 6.3. The remaining *versionHi* and *versionLo* fields shall have the values of 0 and 1 respectively.

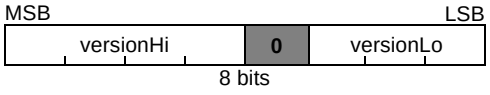

Figure 6.3—Global-time subfield format

6.2.1.4 *precedence* subfields

The precedence field includes the concatenation of multiple fields that are used to establish precedence between grand-master candidates, as illustrated in Figure 6.4.

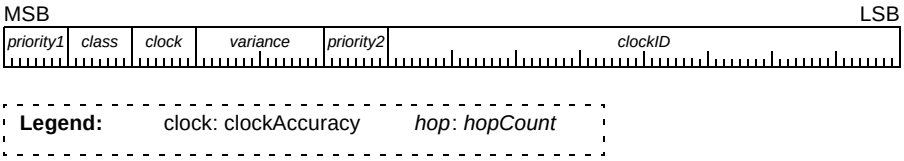

Figure 6.4—*precedence* subfields

6.2.1.4.1 *priority1*: An 8-bit field that can be configured by the user and overrides the remaining *precedence*-resident precedence fields.

6.2.1.4.2 *class*: An 8-bit precedence-selection field defined by the like-named IEEE-1588 field.

6.2.1.4.3 *clockAccuracy*: An 8-bit precedence-selection field defined by the like-named IEEE-1588 field.

6.2.1.4.4 *variance*: A 16-bit precedence-selection field defined by the like-named IEEE-1588 field.

6.2.1.4.5 *priority2*: A 8-bit field that can be configured by the user and overrides the remaining *precedence*-resident *clockID* field.

6.2.1.4.6 *clockID*: A 64-bit globally-unique field that ensures a unique precedence value for each potential grand master, when {*priority1*, *class*, *clockAccuracy*, *variance*, *priority2*} fields happen to have the same value (see 6.2.1.5).

6.2.1.5 *clockID* subfields

The 64-bit *clockID* field is a unique identifier. For stations that have a uniquely assigned 48-bit *macAddress*, the 64-bit *clockID* field is derived from the 48-bit MAC address, as illustrated in Figure 6.5.

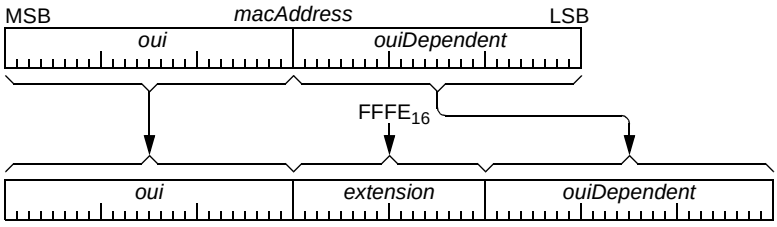

Figure 6.5—*clockID* format

6.2.1.5.1 *oui*: A 24-bit field assigned by the IEEE/RAC (see 3.10.1).

6.2.1.5.2 *extension*: A 16-bit field assigned to encapsulated EUI-48 values.

6.2.1.5.3 *ouiDependent*: A 24-bit field assigned by the owner of the *oui* field (see 3.10.2).

6.2.1.6 Global-time subfield formats

Time-of-day values within a frame are based on seconds and fractions-of-second values, consistent with IETF specified NTP[B7] and SNTP[B8] protocols, as illustrated in Figure 6.6.

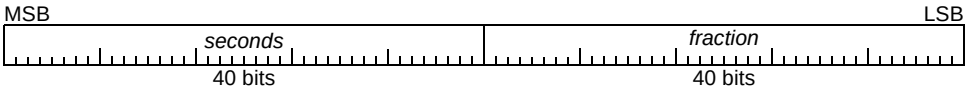

Figure 6.6—Global-time subfield format

6.2.1.6.1 *seconds*: A 40-bit signed field that specifies time in seconds.

6.2.1.6.2 *fraction*: A 40-bit unsigned field that specifies a time offset within each *second*, in units of 2^{-40} second.

The concatenation of these fields specifies a 96-bit *grandTime* value, as specified by Equation 6.1.

$$grandTime = seconds + (fraction / 2^{40}) \quad (6.1)$$

6.2.1.7 *errorTime*

The error-time values within a frame are based on a selected portion of a fractions-of-second value, as illustrated in Figure 6.7. The 40-bit signed *fraction* field specifies the time offset within a *second*, in units of 2^{-40} second.

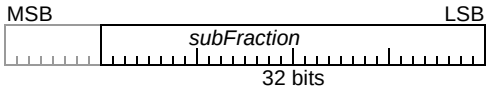

Figure 6.7—errorTime format

6.2.1.8 *snapTime* formats

The *snapTime* value within a frame is based on seconds and fractions-of-second field values, as illustrated in Figure 6.8. The 48-bit *fraction* field specifies the time offset within the *second*, in units of 2^{-48} second.

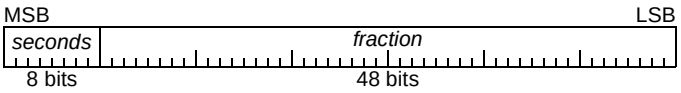

Figure 6.8—*snapTime* format

6.2.1.9 When generated

The time-sync related MS_UNITDATA.indication service primitive is generated when new time-sync information is available. Such information could change the selection of the grand-master or could provide a more-recent {*grandTime*, *stationTime*} time affiliation necessary for maintaining accurate grand-master synchronized time references.

6.2.1.10 Effect of receipt

Receipt of the service primitive by the GrandSync entity triggers an update of the grand-master selection information. If the grand-master selection determines the source-port to be the preferred port, its provided

{*grandTime, stationTime*} time affiliation is also echoed to the attached entities, via invocation of the MS_UNITDATA.request service primitive.

6.2.2 MS_UNITDATA.request

6.2.2.1 Function

Communicates GrandSync protocol-entity supplied information to attached media-dependent ports. The information is sufficient for attached ports to update/propagate grand-master clock-synchronization parameters.

6.2.2.2 Semantics of the service primitive

The semantics of the primitives are as follows:

```
MA_UNITDATA.request
{
    destination_address,    // Destination address
    source_address,         // Optional
    priority,               // Forwarding priority
    service_data_unit,      // Delivered content
    {                       // Contents of the service_data_unit
        protocolType,       // Distinguishes AVB frames from others
        function,           // Distinguishes between timeSync and other frames
        version,            // Distinguishes between timeSync frame versions
        precedence,         // Precedence for grand-master selection
        grandTime,          // Global-time snapshot (1-cycle delayed)
        errorTime,          // Accumulated grandTime error
        sourcePort,         // Identifies the source port
        hopCount,           // Distance from the grand-master station
        snapTime,           // Local-time snapshot (1-cycle delayed)
        syncInterval        // Nominal timeSync transmission interval
    }
}
```

The parameters of the MA_UNITDATA.request are described in 6.2.1.2.

6.2.2.3 When generated

Generated by the GrandSync entity upon receipt of a time-sync related MS_UNITDATA.indication from a preferred (by grand-master selection protocol) source port.

6.2.2.4 Effect of receipt

Receipt of the service primitive by a ClockSlave or TS entity updates entity storage. This storage update allows the destination-port to provide accurate {*grandTime, stationTime*} affiliations during later time-sync information transmissions.

6.3 GrandSync state machine

6.3.1 Function

The GrandSync state machine is responsible for observing MS_UNITDATA.indication parameters, selecting messages with preferred time-sync content, and echoing this content in following MS_UNITDATA.request parameters.

6.3.2 State machine definitions

AVB identifiers
Assigned constants used to specify AVB frame parameters.
AVB_FUNCTION—The function code that corresponds to a time-sync frame.
value—TBD.
AVB_MCAST—The multicast destination address corresponding to the adjacent neighbor.
value—TBD.
AVB_TYPE—The *protocolType* corresponding that uniquely identifies time-sync SDUs.
value—TBD.
AVB_VERSION—The number that uniquely identifies this version of time-sync SDUs.
value—TBD.
LAST_HOP
A constant that specifies the largest possible *hopCount* value.
value—255
NULL
A constant indicating the absence of a value that (by design) cannot be confused with a valid value.
ONES
A large constant wherein all binary bits of the numerical representation are set to one.
queue values
Enumerated values used to specify shared FIFO queue structures.
Q_MS_IND—Queue identifier for MS_UNITDATA.indication transfers.
Q_MS_REQ—Queue identifier for MS_UNITDATA.request transfers.

6.3.3 State machine variables

ePtr
A pointer to entity-dependent storage, where that storage comprises the following:
lastTime—Time of the last best-preference update, used for timeout purposes.
rxSaved—A copy of the best-preference GrandSync message parameters.
new, old
Local variables consisting of concatenated *preference*, *hopCount*, and *port* parameters.
rsPtr
A pointer to the service-data-unit portion of *rxInfo* storage.
rxInfo
Parameters associated with an MS_UNITDATA.indication (see 6.2.1.2), comprising the following:
destination_address, *source_address*, *service_data_unit*
Where *service_data_unit* comprises:
errorTime, *function*, *grandTime*, *hopCount*, *precedence*,
protocolType, *snapTime*, *syncInterval*, *version*
rxPtr
A pointer to the *rxInfo* storage.

stationTime
A shared value representing current time within each station.
Within the state machines of this standard, this is assumed to have two components, as follows:
seconds—An 8-bit unsigned value representing seconds.
fraction—An 40-bit unsigned value representing portions of a second, in units of 2^{-40} second.
ssPtr
A pointer to the service-data-unit portion of *ePtr->rxSaved* storage.
sxPtr
A pointer to the *ePtr->rxSaved* storage.
tsPtr
A pointer to the service-data-unit portion of *txInfo* storage.
txInfo
Parameters associated with an MS_UNITDATA.request (see 6.2.1.2), comprising the following:
destination_address, *source_address*, *service_data_unit*
Where *service_data_unit* comprises:
errorTime, *function*, *grandTime*, *hopCount*, *precedence*,
protocolType, *snapTime*, *syncInterval*, *version*
txPtr
A pointer to the *txInfo* storage.

6.3.4 State machine routines

Dequeue(*queue*)
Returns the next available frame from the specified queue.
info—The next available parameters.
NULL—No parameters available.
Enqueue(*queue*, *info*)
Places the *info* parameters at the tail of the specified queue on all ports.
FormPreference(*precedence*, *hops*, *port*)
Forms a 16-byte *preference* by concatenating the following fields:
precedence (14 bytes)
hops (1 byte)
port (1 byte)
StationTime(*ePtr*)
Returns the value of the station's shared local timer, encoded as follows:
seconds—A 16-bit unsigned value representing seconds.
fraction—A 48-bit unsigned value representing portions of a second, in units of 2^{-40} second.
TimeSyncSdu(*info*)
Checks the frame contents to identify MS_DATAUNIT.indication frames.
TRUE—The frame is a timeSync frame.
FALSE—Otherwise.

6.3.5 GrandSync state table

The GrandSync state machine includes a media-dependent timeout, which effectively restarts the grand-master selection process in the absence of received timeSync frames, as specified by Table 6.1.

Table 6.1—GrandSync state table

Current		Row	Next	
state	**condition**		**action**	**state**
START	(rxInfo = Dequeue(Q_MS_IND)) != NULL	1	—	TEST
	(stationTime – ePtr->timer) > 4 * ePtr->syncInterval	2	ePtr->lastTime = stationTime; ssPtr->hopCount = ssPtr->sourePort = ssPtr->precedence = ONES;	START
	—	3	stationTime = StationTime();	
TEST	TimeSyncSdu(rsPtr) && rsPtr->hopCount != LAST_HOP	4	test = FormPreference(rsPtr->precedence, rsPtr->hopCount, rsPtr->port); best = FormPreference(ssPtr->precedence, ssPtr->hopCount, ePtr->sourcePort);	SERVE
	—	5	—	START
SERVE	rsPtr->port == ePtr->rxSourcePort	6	ePtr->lastTime = stationTime; *ssPtr = *tsPtr = *rsPtr;	HOPS
	test <= best	7		
	—	8	—	START
HOPS	rsPtr->hopCount > ssPtr->hopcount	9	tsPtr->hopCount = Min(LAST_HOP, 1 + (LAST_HOP + rsPtr->hopCount) / 2);	LAST
	—	10	tsPtr->hopCount = rsPtr->hopCount + 1;	

LAST	—	11	txPtr->destination_address = AVB_MCAST;
			txPtr->source_address = MacAddress(ePtr);
			tsPtr->protocolType = AVB_TYPE;
			tsPtr->function = AVB_FUNCTION;
			tsPtr->version = AVB_VERSION;
			Enqueue(Q_MS_REQ, txPtr);
				START

Row 6.1-1: Available indication parameters are processed.
Row 6.1-2: The absence of indications forces a timeout, after a entity-dependent delay
Row 6.1-3: Wait for changes of conditions.

Row 6.1-4: Still-active time-sync messages are processed further, based on grand-master preferences. The *new* and *old* preference values consist of *precedence*, *hopCount*, and *port* components.
Row 6.1-5: Other messages and over-aged indications are discarded.

Row 6.1-6: Same-port indications always have preference.
Row 6.1-7: Preferred preference-level indications are accepted.
Row 6.1-8: Other indications are discarded.

Row 6.1-9: Increasing *hopCount* values are indicative of a rogue frame and are therefore quickly quashed.
Row 6.1-10: Non-increasing *hopCount* values are incremented and are thus aged slowly.

Row 6.1-11: Reset the timeout timer; broadcast saved parameters to all ports (including the source).

7. ClockMaster/ClockSlave state machines

7.1 Overview

7.1.1 ClockMaster/ClockSlave behaviors

This clause specifies the state machines that specify ClockMaster and ClockSlave entity processing. The operations are described in an abstract way and do not imply any particular implementations or any exposed interfaces. There is not necessarily a one-to-one correspondence between the primitives and formal procedures and the interfaces in any particular implementation.

The ClockMaster entity is responsible for forwarding the grand-master time supplied by the ClockSource via the masterSync service primitive, as follows:

a) A count value (this is normally incremented in sequential masterSync messages) is checked.

b) Grand-master time from masterSync[*n*+1] is associated with the masterSync[*n*] invocation time.

c) The masterSync parameters are supplemented and passed to the GrandSync entity.

The ClockSlave entity is responsible for extracting the grand-master time delivered by the GrandSync entity and supplying the current value to the ClockSink entity through the slavePoke service interface, as follows:

a) Grand-master time samples are extracted from GrandSync-supplied MS_UNITDATA-requests, and saved as inputs for computing grand-master times in following slaveSync messages.

b) When triggered by a slavePoke indication, a slaveSync message is delivered to the ClockSink. That message supplies the grand-master time associated with the slavePoke invocation time.

7.1.2 ClockMaster/ClockSlave interface model

The time-synchronization service model assumes the presence of one or more grand-master capable entities communicating with a MAC relay, as illustrated on the left side of Figure 7.1. A grand-master capable port is also expected to provide clock-slave functionality, so that any non-selected grand-master-capable station can synchronize to the selected grand-master station.

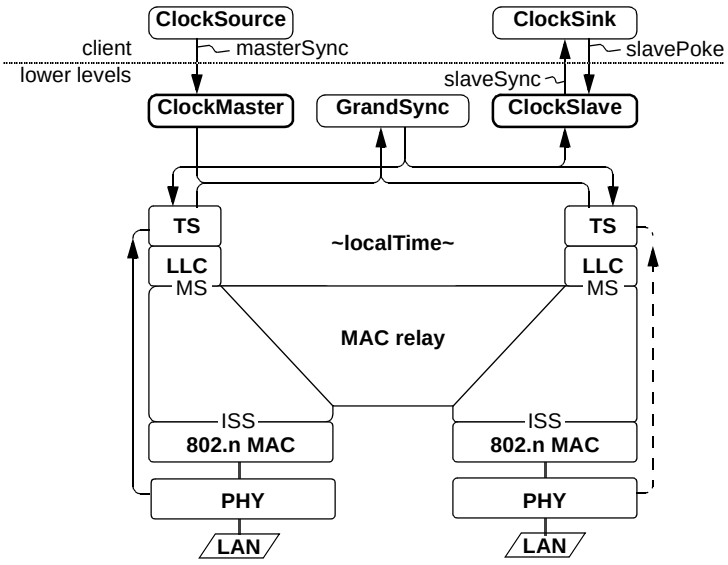

Figure 7.1—ClockMaster interface model

The clock-master ClockMaster state machine (illustrated with an italics name and darker boundary) is responsible for monitoring its port's masterSync requests and sending MAC-relay frames. The sequencing of this state machine is specified by Table 7.1; details of the computations are specified by the C-code of Annex G.

The time-synchronization service model assumes the presence of one or more clock-slave capable time-sync entities communicating with a GrandSync protocol entity, as illustrated on the top-side of Figure 7.1. A non-talker clock-slave capable entity is not required to be grand-master capable.

The ClockSlave state machine (illustrated with an italics name and darker boundary) is responsible for saving time parameters from relayed timedSync frames and servicing time-sync requests from the attached clock-slave interface. The sequencing of this state machine is specified by Table 7.2; details of the computations are specified by the C-code of Annex G.

7.2 ClockMaster service interfaces

7.2.1 Shared service interfaces

The ClockMaster entity is coupled to the bridge ports TS entities via the defined time-sync related MS_UNITDATA.indication service interface (see 6.2.1).

7.2.2 masterSync service interface

7.2.2.1 Function

Provides the ClockMaster entity with clock-synchronization parameters derived from the reference clock. The information is sufficient to provide the ClockMaster with accurate {*grandTime*, *localTime*} associations. The ClockSource entity supplies the reference time for service-interface invocation *n* within the parameters of the next service-interface invocation *n*+1.

7.2.2.2 Semantics of the service primitive

The semantics of the primitives are as follows:

```
masterSync {
    frameCount,        // An integrity-check that is incremented each invocation
    grandTime,         // Global-time snapshot (1-cycle delayed)
}
```

The parameters of the masterSync service-interface primitive are described as follows:

7.2.2.2.1 *frameCount*: An 8-bit field that is incremented on each service-interface invocation.

7.2.2.2.2 *grandTime*: An 80-bit field that specifies the grand-master synchronized time within the source station, when the previous timeSync frame was transmitted (see 6.2.1.6).

7.2.2.3 When generated

The masterSync service primitive is invoked by a client-resident ClockSource entity. The intent is to provide the ClockMaster with continuous/accurate updates from a ClockSource-resident clock reference.

7.2.2.4 Effect of receipt

Upon receipt by the ClockMaster entity, the encapsulated *grandTime* value is affiliated with the *stationTime* snapshot from the previous invocation; the resulting {*grandTime*, *stationTime*} affiliation is passed to the GrandSync entity for redistribution to other ClockSlave and TS entities.

7.3 ClockMaster state machine

7.3.1 State machine definitions

AVB identifiers
: Assigned constants used to specify AVB frame parameters (see 6.3.2).
  AVB_FUNCTION, AVB_MCAST, AVB_TYPE, AVB_VERSION

NULL
: A constant indicating the absence of a value that (by design) cannot be confused with a valid value.

COUNT
: A numerical constant equal to the range of the *info.frameCount* field value.

queue values
: Enumerated values used to specify shared FIFO queue structures.
  Q_CM_SET—The queue identifier associated with received clock-master sync frames.
  Q_MS_IND—A GrandSync queue identifier (see 6.3.2).

7.3.2 State machine variables

count
: A transient value representing the expected value of the next *rxInfo.frameCount* field value.

ePtr
: A pointer to an entity data structure with information comprising the following:
  precedence—A 14-byte field that specifies the grand-master selection precedence.
  rxSaved—Saved parameters from a received masterSync primitive.
  snapShot0—The *info.snapShot* field value from the last receive-port poke indication.
  snapShot1—The value of the *ePtr->snapShot0* field saved from the last poke indication.
  syncInterval—The expected rate of clockMaster service-interface invocations.

rxInfo
: A contents of a higher-level supplied time-synchronization request, including the following:
  frameCount—A value that increments on each masterSync frame transmission.
  grandTime—The value of grand-master time, when the previous masterSync frame was sent.

rxPtr
: A pointer to *rxInfo* storage.

stationTime
: See 6.3.3.

sxPtr
: A pointer to the *ePtr->rxSaved* storage.

tsPtr
: A pointer to the service-data-unit portion of *txInfo* storage.

txInfo
: Storage for to-be-transmitted MS_UNITDATA.request parameters (see 6.2.2.2), comprising:
  destination_address, source_address, service_data_unit
  Where *service_data_unit* comprises:
  errorTime, function, grandTime, hopCount, precedence, protocolType, snapTime, sourcePort, syncInterval, version

txPtr
A pointer to *txInfo* storage.

7.3.3 State machine routines

Dequeue(*queue*)
Enqueue(*queue*, *info*)
SourcePort(*entity*)
StationTime(*entity*)
TimeSyncSdu(*info*)
See 6.3.4.

7.3.4 ClockMaster state table

The ClockMaster state table encapsulates clock-provided sync information into a MAC-relay frame, as illustrated in Table 7.1.

Table 7.1—ClockMaster state machine table

Current		Row	Next	
state	**condition**		**action**	**state**

START	(rxInfo = Dequeue(Q_CM_SET)) **!**= NULL	1	ePtr->snapShot1 = ePtr->snapShot0;
			ePtr->snapShot0 = stationTime;
			count = (sxPtr->frameCount + 1) % COUNT;
			grandTime = rxPtr->grandTime;
			*sxPtr = rxInfo;
				SEND

	—	2	stationTime = StationTime(ePtr);	START

SEND	count == sxPtr->frameCount	3	txPtr->destination_address = AVB_MCAST;
			txPtr->source_address = MacAddress(ePtr);
			tsPtr->prototolType = AVB_TYPE;
			tsPtr->function = AVB_FUNCTION;
			tsPtr->version = AVB_VERSION;
			tsPtr->precedence = ePtr->precedence;
			tsPtr->hopCount = 0;
			tsPtr->sourcePort = SourcePort(ePtr);
			tsPtr->grandTime = grandTime;
			tsPtr->errorTime = 0;
			tsPtr->snapTime = ePtr->snapShot1;
			tsPtr->syncInterval = ePtr->syncInterval;
			Enqueue(Q_MS_IND, txInfo);
				START

	—	4	—	

Row 7.1-1: Update snapshot values on masterSync request arrival.
Row 7.1-2: Wait for the next change of state.

Row 7.1-3: Sequential requests are forwarded as a MA_UNITDATA.request to the GrandSync entity.
Row 7.1-4: Nonsequential requests are discarded.

7.4 ClockSlave service interfaces

7.4.1 Shared service interfaces

The ClockSlave entity is coupled to the GrandSync entity, via the defined MS_SYNC.request service interface (see 6.2.2).

7.4.2 slavePoke service interface

7.4.2.1 Function

Triggers the ClockSlave entity to provide a {*grandTime*, *localTime*} association that is synchronized with the grand-master clock.

7.4.2.2 Semantics of the service primitive

The semantics of the primitives are as follows:

```
slavePoke {
    frameCount          // An integrity-check that is incremented each invocation
}
```

The parameters of the masterSync service-interface primitive are described as follows:

7.4.2.2.1 *frameCount*: An 8-bit field that is incremented on each service-interface invocation.

7.4.2.3 When generated

The slavePoke service primitive is invoked by a client-resident ClockSink entity. The intent is to trigger the ClockSlave's invocation of a following slaveSync primitive, thus providing the ClockSink entity with a recent {*grandTime*,*stationTime*} affiliation.

7.4.2.4 Effect of receipt

Upon receipt by a ClockSlave entity, a copy of the current *stationTime* value is saved and an invocation of a following slaveSync primitive is triggered.

7.4.3 slaveSync service interface

7.4.3.1 Function

Provides the ClockSync entity with clock-synchronization parameters derived from the reference clock. The information comprises {*frameCount*, *grandTime*} associations: *frameCount* is supplied by the previous slavePoke invocation; *grandTime* represents the invocation time of that preceding slavePoke service primitive.

7.4.3.2 Semantics of the service primitive

The semantics of the primitives are as follows:

```
slaveSync {
    frameCount,         // Identifies the previous slavePoke invocation
    grandTime,          // Grand-master synchronized snapshot.
}
```

The parameters of the slaveSync service-interface primitive are described as follows:

7.4.3.2.1 *frameCount*: An 8-bit field that copied from the like-named field of the previous slavePoke service-interface invocation.

7.4.3.2.2 *grandTime*: An 80-bit field that specifies the grand-master synchronized time within the ClockSlave entity, when the previous slavePoke service-interface was invoked.

7.4.3.3 When generated

The invocation of the slaveSync service primitive is invoked by the receipt of a ClockSink supplied slavePoke message. The intent is to provide the ClockSink entity with a recent {*grandTime*,*stationTime*} affiliation.

7.4.3.4 Effect of receipt

Upon receipt by a ClockSink entity, the {*grandTime*,*stationTime*} affiliation is expected to be saved and (along with previously saved copies) used to adjust the rate of the grand-master synchronized ClockSink-resident clock.

7.5 ClockSlave state machine

7.5.1 Function

7.5.2 State machine definitions

NULL
    A constant indicating the absence of a value that (by design) cannot be confused with a valid value.
queue values
    Enumerated values used to specify shared FIFO queue structures.
        Q_MS_REQ—A GrandSync queue identifier (see 6.3.2).
        Q_CS_REQ—The queue identifier associated with slavePoke requests.
        Q_CS_IND—The queue identifier associated with slaveSync indications.

7.5.3 State machine variables

ePtr
A pointer to entity-dependent information, including the following:
rxSaved—A copy of the GrandSync supplied MA_DATAUNIT.request value.
syncInterval—The expected service rate of slavePoke services.
timed[3]—Recently saved time events, each consisting of the following:
grandTime—A previously sampled grand-master synchronized time.
errorTime—The residual error associated with the sampled *grandTime* value.
snapTime—The *stationTime* affiliated with the sampled *grandTime* value.

cxInfo
A contents of a higher-level supplied time-synchronization request, including the following:
frameCount—A value that increments on each masterSync message transfer.

nextTime
Storage representing *grandTime* and *errorTime* values returned from call to *NextTimed*( ).

rsPtr
A pointer to the service-data-unit portion of *rxInfo*.

rxInfo
A contents of a GrandSync supplied MA_UNITDATA.request (see 6.2.2), including the following:
destination_address, source_address, service_data_unit
Where *service_data_unit* comprises:
errorTime, function, grandTime, protocolType, snapTime, version

rxPtr
A pointer to *rxInfo*.

rxSyncInterval
The synchronization interval of this station's GrandSync-selected clock-slave port.

stationTime
See 6.3.3.

ssPtr
A pointer to the service-data-unit portion of the *ePtr->rxSaved* storage

sxPtr
A pointer to the *ePtr->rxSaved* storage

timePtr
A pointer to the *ePtr->timed*[ ] array storage

txInfo
A contents of a ClockSlave supplied slaveSync (see 6.2.2), comprising the following:
frameCount—The saved value of the like named field from the previous slavePoke message.
grandTime—The grand-master synchronized time sampled during the slavePoke transfer.

txPtr
A pointer to *txInfo* storage.

txSyncInterval
The synchronization interval of this clock-master port.

7.5.4 State machine routines

Dequeue(*queue*)

Enqueue(*queue, info*)
See 6.3.4.

NextTimed(*stationTime, rxSyncInterval, syncInterval, timed*)
Returns *grandTime* and *errorTime* values associated with a snapshot taken at *stationTime*, received as well as internal *syncInterval* values, and previous *time* information saved in the timed array.

StationTime(*entity*)
See 6.3.4.

TimeSyncSdu(*info*)
See 7.3.3.

7.5.5 ClockSlave state table

The ClockSlave state machine includes a media-dependent timeout, which effectively disconnects a clock-slave port in the absence of received timeSync frames, as illustrated in Table 7.2.

Table 7.2—ClockSlave state table

Current		Row	Next	
state	**condition**		**action**	**state**
START	(rxInfo = Dequeue(Q_MS_REQ)) != NULL	1	—	TEST

	((cxInfo = Dequeue(Q_CS_REQ)) != NULL	2	rxSyncInterval = ssPtr->syncInterval;
			txSyncInterval = ePtr->syncInterval;
			nextTimes = NextTimed(stationTime, rxSyncInterval, txSyncInterval, ePtr->timed);
			txPtr->count = cxInfo.count;
			txPtr->grandTime = nextTimes.grandTime + nextTimes.errorTime;
			Enqueue(Q_CS_IND, txInfo);
				START

	—	3	stationTime = StationTime(ePtr);	

TEST	TimeSyncSdu(rsPtr)	4	*sxPtr = *rxPtr;
			timePtr->grandTime = rsPtr->grandTime;
			timePtr->errorTime = rsPtr->errorTime;
			timePtr->snapTime = rsPtr->snapTime;
				START

	—	5	—	

Row 7.2-1: The received MS_UNITDATA.request parameters are dequeued for checking.
Row 7.2-2: A clock-slave request generates an affiliated information-providing indication.
The affiliated indication has the sequence-count information provided by the request.
The delivered end-point *grandTime* value is the sum of delivered *grandTime* and *errorTime* values.
The requested content is queued for delivery to the higher-level client.
Row 7.2-3: Wait for the next change-of-conditions.

Row 7.2-4: Validated GrandSync entity requests are accepted; its time parameters are saved.
The back-interpolation time is estimated from the *syncInterval* times of the source and clock slave.
(This back-interpolation time is used by *NextTimed*( ), which provides transmission-time estimates.)
Row 7.2-5: Wait for the next change-of-conditions.

8. Duplex-link state machines

8.1 Overview

This clause specifies the state machines that support duplex-link 802.3-based bridges. The operations are described in an abstract way and do not imply any particular implementations or any exposed interfaces. There is not necessarily a one-to-one correspondence between the formal specification and the interfaces in any particular implementation.

8.1.1 Duplex-link indications

The duplex-link TimeSyncRxDuplex state machines are provided with snapshots of timeSync-frame reception and transmission times, as illustrated by the ports within Figure 8.1. These link-dependent indications can be different for bridge ports attached to alternative media.

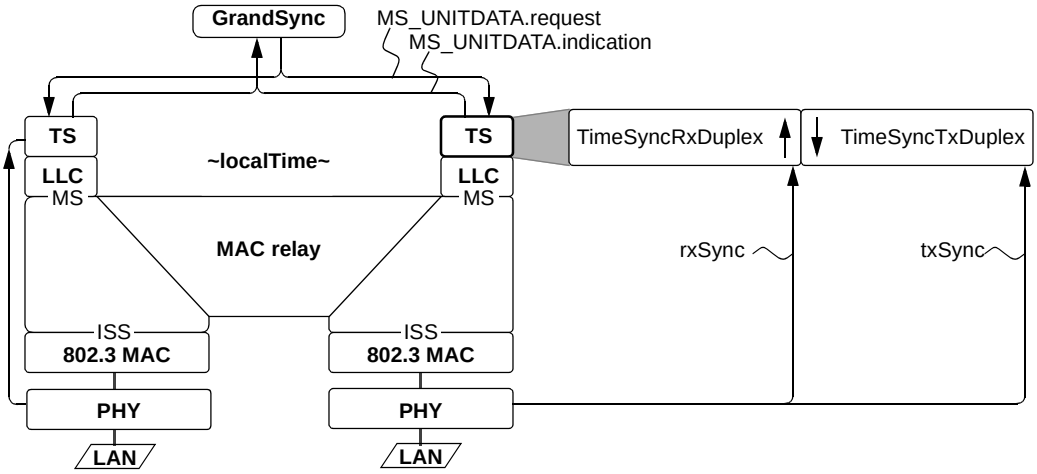

Figure 8.1—Duplex-link interface model

The rxSync and txSync indications provide a tag (to reliably associate them with MAC-supplied timeSync frames) and a *localTime* stamp indicating when the associated timeSync frame was received, as illustrated within Figure 8.2.

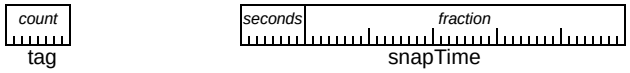

Figure 8.2—Contents of rxSync/txSync indications

8.1.2 Link-delay compensation

Synchronization accuracies are affected by the transmission delays associated with transmissions over links between bridges. To compensate for these transmission delays, the receive port is responsible for compensating {*grandTime*, *stationTime*} affiliations by the (assumed to be constant) frame-transmission delay.

The clock-slave entity uses the computed cable-delay measurement and is therefore (in concept) responsible for initiating such measurements. Cable-delay measurements begin with the transmission of frame F1 between the clock-slave and clock-master stations and conclude with the a clock-master response, a transmission of frame F2 between the clock-master to clock-slave stations, as illustrated in Figure 8.3.

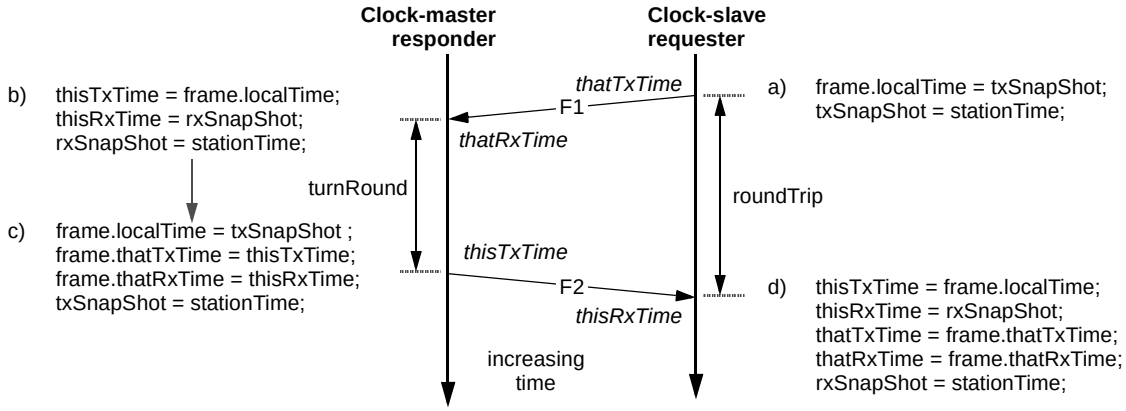

Figure 8.3—Link-delay compensation

The cable-delay computations are performed in multiple steps, as follows:

a) The F1-frame transmission involves multiple steps:
   1) The *txSnapShot* value (time of the last F1 transmission) is copied to *frame*.*localTime* storage.
   2) Remaining fields are copied into *frame* storage; the *frame*-storage content is transmitted.
   3) The *txSnapShot* value is set to the frame-F1 transmission time, for next step (a) usage.

b) The F1-frame reception involves multiple steps:
   1) The *frame*.*localTime* value is copied to a port-local *thisTxTime* field, for next step (c) usage.
   2) The *rxSnapShot* value (time of the last F1 reception) is copied to a port-local *thisRxTime* field.
   3) The *rxSnapShot* value is set to the F1-frame reception time, for next step (b) usage.

c) The F2-frame transmission involves multiple steps:
   1) The *txSnapShot* value (time of the last F1 transmission) is copied to *frame*.*localTime* storage.
   2) The receive-port *thisTxTime* value is copied to *frame*.*thatTxTime* storage.
   3) The receive-port *thisRxTime* value is copied to *frame*.*thatRxTime* storage.
   4) Remaining fields are copied into *frame* storage; the *frame*-storage content is transmitted.
   5) The *txSnapShot* value is set to the frame-F2 transmission time, for next step (c) usage.

d) The F2-frame reception involves multiple steps:
   1) The *frame*.*localTime* value is copied to a port-local *thisTxTime* field.
   2) The *rxSnapShot* value (time of the last F2 reception) is copied to a port-local *thisRxTime* field.
   3) The *frame*.*thatTxTime* value is copied to a port-local *thatTxTime* field.
   4) The *frame*.*thatRxTime* value is copied to a port-local *thatRxTime* field.
   5) The *rxSnapShot* value is set to the F2-frame reception time, for next step (d) usage.

At the conclusion of these steps, the values returned to the clock-slave requester include the values below. (Within Figure 8.3, these values are also illustrated in the center, at their source, using a distinct italic font.)

— *thatTxTime*. The clock-slave transmit time.
— *thatRxTime*. The clock-master receipt time.
— *thisTxTime*. The clock-master transmit time.
— *thisRxTime*. The clock-slave receipt time.

Based on the preceding listed values, Equation 8.1 defines the computations for computing *linkDelay.* Although not explicitly stated, the best accuracy can be achieved by performing these computation every cycle.

linkDelay = (*roundTrip* – *turnRound*) / 2; (8.1)
roundTrip = *thisRxTime* – *thatTxTime*;
turnRound = (*thisTxTime* – *thatRxTime*) * *ratesRatio*;
Where:
ratesRatio ≅ (*deltaRxTime* / *deltaTxTime*);

The value of *ratesRatio* is necessary to maintain tight accuracies in the presence of significant (±200 PPM) differences in clock-master/clock-slave timing references and significant (multiple milliseconds) *turnRound* delays. This value is also readily computed from the preceding listed values, as specified by Equation 8.2.

ratesRatio[*n*] = (*thisRxTime*[*n*] – *thisRxTime*[*n*-N]) / (*thisTxTime*[*n*] – *thisTxTime*[*n*-N]); (8.2)

NOTE—For 802.3 and other inexpensive interconnects, the processing of slow-rate messages is oftentimes performed by firmware and (due to interrupt and processing delays) the turn-around delays can be much larger than the packet-transmission times.

The cable-delay computations assume the transmission delays associated with frame F1 and frame F2 are equal and constant. If the duplex links within a span have different propagation delays, these *linkDelay* calculations do not correspond to the different propagation delays, but represent the average of the two link delays. Implementers have the option of manually specifying the link-delay differences via MIB-accessible parameters, within tightly-synchronized systems where this inaccuracy might be undesirable.

This cable-delay calculation does not rely on the particular timings of F1 and F2 frame transmissions. These transmissions can be triggered independently (as opposed to one triggered by the other) and could occur at different rates (although the accuracies are limited by the slower rate). As a direct benefit of these independence properties, distinct interlocks or timeouts for expected-but-corrupted-and-not-delivered transmissions are unnecessary.

Furthermore, there is no need to transport F1 and F2 content in distinct frames. The contents of clock-slave affiliated F1 and clock-master affiliated F2 frames can be merged and transported within the same frame. Thus, distinct frame types and/or transmission timings are unnecessary; the link-delay calibration protocols do nothing to prevent the same frame from communicating master-to-slave and slave-to-master link delays, in addition to the baseline grand-master timing and selection parameters.

8.2 timeSyncDuplex frame format

8.2.1 timeSyncDuplex fields

Duplex-link time-synchronization (timeSyncDuplex) frames facilitate the synchronization of neighboring clock-master and clock-slave stations. The frame, which is normally sent at 10ms intervals, includes time-snapshot information and the identity of the network's clock master, as illustrated in Figure 8.4. The gray boxes represent physical layer encapsulation fields that are common across Ethernet frames.

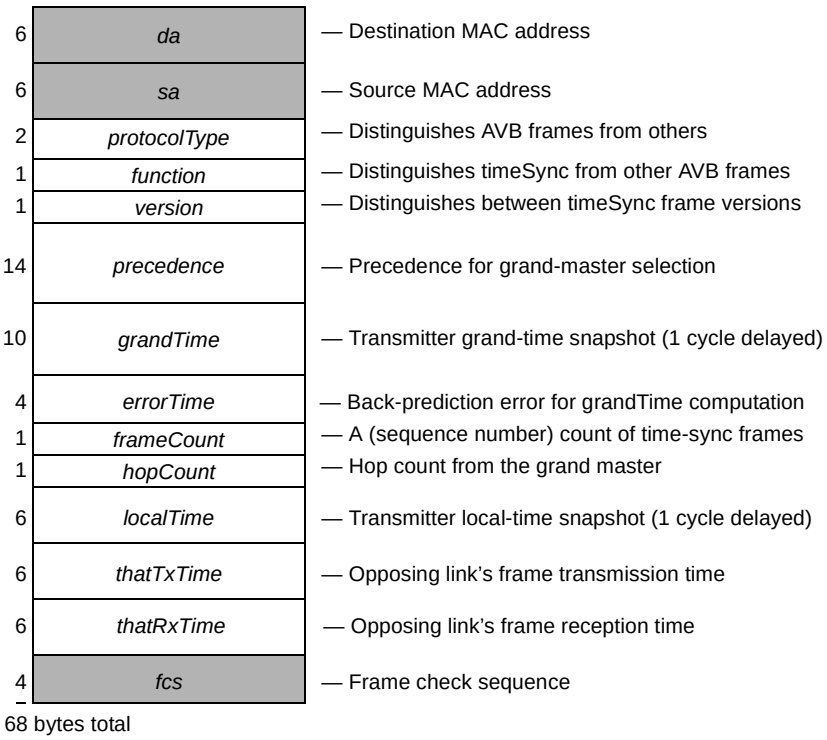

Figure 8.4—timeSyncDuplex frame format

NOTE— Existing 1588 time-snapshot hardware captures the values between byte-offset 34 and 45 (inclusive). The location of the *frameCount* field (byte-offset 44) has been adjusted to ensure this field can be similarly captured for the purpose of unambiguously associating timeSync-packet snapshots (that bypass the MAC) and timeSync-packet contents (that pass through the MAC).

The 48-bit ***da*** (destination address), 48-bit ***sa*** (source address) field, 16-bit ***protocolType,*** 8-bit ***function***, 8-bit ***version***, 14-byte ***precedence***, 80-bit ***grandTime***, 32-bit ***errorTime***, 8-bit ***hopCount***, and 6-byte ***localTime*** field are specified in 6.2.1.2.

8.2.1.1 *frameCount*: An 8-bit field that is incremented by one between successive timeSync frame transmission.

8.2.1.2 *thatTxTime*: A 48-bit field that specifies the local free-running time within the source station, when the previous timeSync frame was transmitted on the opposing link (see 6.2.1.8).

8.2.1.3 *thatRxTime*: A 48-bit field that specifies the local free-running time within the target station, when the previous timeSync frame was received on the opposing link (see 6.2.1.8).

8.2.1.4 *fcs*: A 32-bit (frame check sequence) field that is a cyclic redundancy check (CRC) of the frame.

8.2.2 Clock-synchronization intervals

Clock synchronization involves synchronizing the clock-slave clocks to the reference provided by the grand clock master. Tight accuracy is possible with matched-length duplex links, since bidirectional messages can cancel the cable-delay effects.

Clock synchronization involves the processing of periodic events. Multiple time periods are involved, as listed in Table 8.1. The clock-period events trigger the update of free-running timer values; the period affects the timer-synchronization accuracy and is therefore constrained to be small.

Table 8.1—Clock-synchronization intervals

Name	Time	Description
clock-period	< 20 ns	Resolution of timer-register value updates
send-period	10 ms	Time between sending of periodic timeSync frames between adjacent stations
slow-period	100 ms	Time between computation of clock-master/clock-slave rate differences

The send-period events trigger the interchange of timeSync frames between adjacent stations. While a smaller period (1 ms or 100 μs) could improve accuracies, the larger value is intended to reduce costs by allowing computations to be executed by inexpensive (but possibly slow) bridge-resident firmware.

The slow-period events trigger the computation of timer-rate differences. The timer-rate differences are computed over two slow-period intervals, but recomputed every slow-period interval. The larger 100 ms (as opposed to 10 ms) computation interval is intended to reduce errors associated with sampling of clock-period-quantized slow-period-sized time intervals.

8.3 TimeSyncRxDuplex state machine

8.3.1 Function

The TimeSyncRxDuplex state machine is responsible for monitoring its port's rxSync indications, receiving MAC-supplied frames, and sending MAC-relay frames. The sequencing of this state machine is specified by Table 8.2; details of the computations are specified by the C-code of Annex G.

8.3.2 State machine definitions

LAST_HOP
: A constant representing the largest-possible frame.hopCount value.
  value—255.

NULL
: A constant indicating the absence of a value that (by design) cannot be confused with a valid value.

queue values
: Enumerated values used to specify shared FIFO queue structures.
  Q_MS_IND—The queue identifier associated with MAC frames sent into GrandSync.
  Q_ES_IND—The queue identifier associated with the received MAC frames.
  Q_RX_SYNC—The queue identifier associated with rxSync, sent from the lower levels.

8.3.3 State machine variables

cableDelay
: Values (possibly scaled integers) representing cable-delay times.

count
: A transient value representing the expected value of the next *rxInfo.frameCount* value.

cxInfo
: A contents of a lower-level supplied time-synchronization poke indication, including the following:
  frameCount—The value of the like-named field within the last timeSync packet arrival.
  snapTime—The value of *stationTime* associated with the last timeSync packet arrival.

cxPtr
: A pointer to *cxInfo* storage.

ePtr
: A pointer to a data structure that contains port-specific information comprising the following:
  frameCount—The value of *frameCount* within the last received frame.
  rated—The ratio of the local-station and remote-station local-timer rates.
  snapCount—The value of *frameCount* saved from the last snapshot indication.
  snapShot0—The *info.snapShot* field value from the last receive-port snapshot indication.
  snapShot1—The value of the *ePtr->snapShot0* field at the snapshot indication.
  times[*N*]—An array of time groups, where each array elements consists of:
    thisTime—The local receive time associated with received time-sync frames.
    thatTime—The remote transmit time associated with received time-sync frames.

ratesRatio
: A variable representing the ratio of this station's timer to this port's neighbor timer.

roundTrip
: A variable representing the time between transmit-to-neighbor and receive-from-neighbor events.

rsPtr
: A pointer to the service-data-unit portion of *rxInfo* storage.

rxInfo
Storage for received time-sync messages, comprising:
destination_address, *source_address*, *service_data_unit*
Where *service_data_unit* comprises:
errorTime, *frameCount*, *function*, *grandTime*, *hopCount*, *localTime*, *protocolType*, *precedence*, *thatTxTime*, *thatRxTime*, *version*

rxPtr
A pointer to the *rxInfo* storage.

stationTime
See 6.3.3.

tsPtr
A pointer to service-data-unit portion of *txInfo* storage.

turnRound
A variable representing the time between receive-at-neighbor and transmit-from-neighbor events.

txInfo
Storage for information sent to the GrandSync entity, comprising:
destination_address, *source_address*, *service_data_unit*
Where *service_data_unit* comprises:
errorTime, *sourcePort*, *function*, *grandTime*, *hopCount*, *localTime*, *protocolType*, *precedence*, *syncInterval*, *version*

txPtr
A pointer to *txInfo* storage.

thisDelay, *thatDelay*, *thatDelay*, *thisDelta*, *thisTime*, *thatTime*, *tockTime*
Values (possibly scaled integers) representing intermediate local-time values.

8.3.4 State machine routines

Dequeue(*queue*)

Enqueue(*queue*, *info*)
See 6.3.4.

Min(*x*, *y*)
Returns the minimum of *x* and *y* values.

RemoteRate(*times*)
The ratio of local-to-remote *localTime* rates is computed from samples within the of *times* array.
Each times-array element contains two times:
thisTime - the receive time of the frame.
thatTime - the transmit time of the frame.

SourcePort(*entity*)
See 7.3.3.

StationTime(*entity*)

TimeSyncSdu(*info*)
See 6.3.4.

8.3.5 TimeSyncRxDuplex state machine table

The TimeSyncRxDuplex state machine associates PHY-provided sync information with arriving timeSync frames and forwards adjusted frames to the MAC-relay function, as illustrated in Table 8.2.

Table 8.2—TimeSyncRxDuplex state machine table

Current		Row	Next	
state	**condition**		**action**	**state**

START	(cxInfo = Dequeue(Q_RX_SYNC)) != NULL	1	ePtr->snapShot1 = ePtr->snapShot0;
			ePtr->snapShot0 = cxPtr->localTime;
			ePtr->snapCount = cxPtr->frameCount;
			match= (rxPtr->frameCount==ePtr->snapCount);
				START

	(rxInfo=Dequeue(Q_ES_IND)) != NULL	2	count = (ePtr->rxFrameCount + 1) % COUNT; ePtr->rxFrameCount = rxPtr->frameCount;	TEST
	match	3	—	PAIR
	—	4	stationTime = StationTime(ePtr);	START
TEST	!TimeSyncSdu(rsPtr)	5	Enqueue(Q_CS_IND, rxPtr);	START
	rxPtr->hopCount == LAST_HOP	6	—	
	count != rxPtr->frameCount	7	—	
	—	8	match= (rxPtr->frameCount==ePtr->snapCount);	

PAIR		9	ePtr->times[0].thisTime = ePtr->snapShot1;
			ePtr->times[1].thatTime = rsPtr->localTime;
			ratesRatio = RemoteRate(ePtr->times);
			roundTrip = localTime – ePtr->thatTxTime;
			turnRound =
			rsPtr->localTime – rsPtr->thatRxTime;
			cableDelay =
			Min(0, roundTrip – (turnRound * ratesRatio));
			txPtr->destination_address =
			rxPtr->destination_address;
			txPtr->source_address = rxPtr->source_address;
			tsPtr->protocolType = rsPtr->protocolType;
			tsPtr->function = rsPtr->function;
			tsPtr->version = rsPtr->version;
			tsPtr->grandTime = rsPtr->grandTime;
			tsPtr->errorTime = rsPtr->errorTime;
			tsPtr->snapTime = ePtr->snapShot1 – cableDelay;
			tsPtr->sourcePort = SourcePort(ePtr);
			tsPtr->hopCount = rsPtr->hopCount;
			tsPtr->syncInterval = ePtr->syncInterval;
			Enqueue(Q_MR_HOP, relayFrame);
				START

Row 8.2-1: Update snapshot values on timeSync frame arrival.
Row 8.2-2: Initiate inspection of frames received from the lower-level MAC.
Row 8.2-3: Generate a GrandSync message using matching snapshot and frame information.
Row 8.2-4: Wait for the next change-of-state.

Row 8.2-5: The non-timeSync frames are passed through.
Row 8.2-6: Over-aged timeSync frames are discarded.
Row 8.2-7: Non-sequential timeSync frames are ignored.
Row 8.2-8: Associated snapshot and frame information trigger a GrandSync indication generation.

Row 8.2-9: Generate a time-sync GrandSync indication from saved snapshot and frame information.

8.4 TimeSyncTxDuplex state machine

8.4.1 Function

The TimeSyncTxDuplex state machine is responsible for saving time parameters from relayed timedSync frames and forming timeSync frames for transmission over the attached link.

8.4.2 State machine definitions

NULL
A constant indicating the absence of a value that (by design) cannot be confused with a valid value.
queue values
Enumerated values used to specify shared FIFO queue structures.
Q_MR_HOP—The queue identifier associated with frames sent from the relay.
Q_ES_REQ—The queue identifier associated with frames sent to the MAC.
Q_TX_SYNC—The queue identifier associated with txSync, sent from the lower levels.
T10ms
A constant the represents a 10 ms value.

8.4.3 State machine variables

cxInfo
A contents of a lower-level supplied time-synchronization poke indication, including the following:
snapCount—The value of the like-named field within the last timeSync packet arrival.
snapTime—The value of *stationTime* associated with the last timeSync packet arrival.
dPtr
A pointer this port's associated TimeSyncRxDuplex-entity storage.
ePtr
A pointer to a data structure that contains port-specific information comprising the following:
frameCount—A consistency-check identifier that is incremented on each transmission.
lastTime—The last transmit time, saved for timeout purposes.
rxSaved—A copy of the last received GrandSync parameters.
syncInterval—The expected interval between successive time-sync transmissions.
timed—Timing information saved from previous GrandSink requests.
txSnapCount—The *frameCount* value associated with the last transmission.
txSnapTime—The *stationTime* value associated with the last transmission.
rsPtr
A pointer to service-data-unit portion of *rxInfo* storage.

rxInfo
Storage for received time-sync messages from the GrandSync entity, comprising:
destination_address, source_address, service_data_unit
Where *service_data_unit* comprises:
errorTime, function, grandTime, hopCount,
precedence, protocolType, snapTime, syncInterval, version
rxPtr
A pointer to *rxInfo* storage.
stationTime
See 6.3.3.
ssPtr
A pointer to the service-data-unit portion of the *ePtr->rxSaved* storage
sxPtr
A pointer to the *ePtr->rxSaved* storage.
tsPtr
A pointer to service-data-unit portion of *txInfo* storage.
txInfo
Storage for to-be-transmitted time-sync messages, comprising:
destination_address, source_address, service_data_unit
Where *service_data_unit* comprises:
errorTime, function, frameCount, grandTime, hopCount, localTime,
precedence, protocolType, thatRxTime, thatTxTime, version
txPtr
A pointer to *txInfo* storage.

8.4.4 State machine routines

Dequeue(*queue*)
Enqueue(*queue, info*)
See 6.3.4.
NextTimed(*stationTime, rxSyncInterval, syncInterval, timed*)
See 7.5.4.
StationTime(*entity*)
See 7.3.3.
TimeSyncSdu(*info*)
See 6.3.4.

8.4.5 TimeSyncTxDuplex state machine table

The TimeSyncTxDuplex state machine includes a media-dependent timeout, which effectively disconnects a clock-slave port in the absence of received timeSync frames, as illustrated in Table 8.3.

Table 8.3—TimeSyncTxDuplex state machine table

Current		Row	Next	
state	**condition**		**action**	**state**
START	(rxInfo = Dequeue(Q_MS_REQ)) != NULL	1	—	TEST
	(stationTime – ePtr->lastTime) > T10ms	2	ePtr->lastTime = stationTime;	SEND
	(cxInfo = Dequeue(Q_TX_SYNC)) != NULL	3	ePtr->txSnapTime = cxPtr->snapTime; ePtr->txSnapCount = cxPtr->frameCount;	START
	—	4	stationTime = StationTime(ePtr);	
TEST	TimeSyncSdu(rsPtr)	5	ePtr->rxSaved = rxInfo;	START
	—	6	Enqueue(Q_ES_REQ, rxPtr);	

SEND	—	7	dPtr = PortPair(ePtr);
			nextTimes =
			NextTimed(stationTime, rsPtr->syncInterval,
			ePtr->syncInterval, ePtr->timed);
			ePtr->txFrameCount =
			(ePtr->txSnapCount + 1) % COUNT;
			txPtr->destination_address =
			sxPtr->destination_address;
			txPtr->source_address = sxPtr->source_address;
			tsPtr->protocolID = ssPtr->protocolID;
			tsPtr->function = ssPtr->function;
			tsPtr->version = ssPtr->version;
			tsPtr->hopCount = ssPtr->hopCount;
			tsPtr->frameCount = ssPtr->frameCount;
			tsPtr->grandTime = nextTimes.grandTime;
			tsPtr->errorTime = nextTimes.errorTime;
			tsPtr->localTime = ePtr->txSnapTime;
			tsPtr->thatTxTime = dPtr->thisTxTime;
			tsPtr->thatRxTime = dPtr->thisRxTime;
			Enqueue(Q_ES_REQ, txPtr);
				START

Row 8.3-1: Relayed frames are further checked before being processed.
Row 8.3-2: Transmit periodic timeSync frames.
Row 8.3-3: Update snapshot values on timeSync frame departure.
Row 8.3-4: Wait for the next change-of-state.

Row 8.3-5: The timeSync messages are checked further.
Row 8.3-6: The non-timeSync messages are passed through.

Row 8.3-7: Active timeSync frames are cable-delay compensated and passed through.

9. Wireless state machines

EDITOR DVJ NOTE—This clause is based on indirect knowledge of the 802.11v specifications, as interpreted by the author, and have not been reviewed by the 802.1 or 802.11v WGs. The intent was to provide a forum for evaluation of the media-independent MAC-relay interface, while also triggering discussion of 802.11v design details. As such, this clause is highly preliminary and subject to change.
Specifically, we have not resolved the grouping of information that is transferred through the service interfaces (currently written as all) and the information that would be transferred through standard MAC frames (currently written as none).

9.1 Overview

This clause specifies the state machines that support wireless 802.11v-based bridges. The operations are described in an abstract way and do not imply any particular implementations or any exposed interfaces. There is not necessarily a one-to-one correspondence between the formal specification and the interfaces in any particular implementation.

9.1.1 Link-dependent indications

The wireless 802.11v TimeSyncRadio state machines are provided with MAC service-interface parameters, as illustrated within Figure 9.1. These link-dependent indications can be different for bridge ports attached to alternative media.

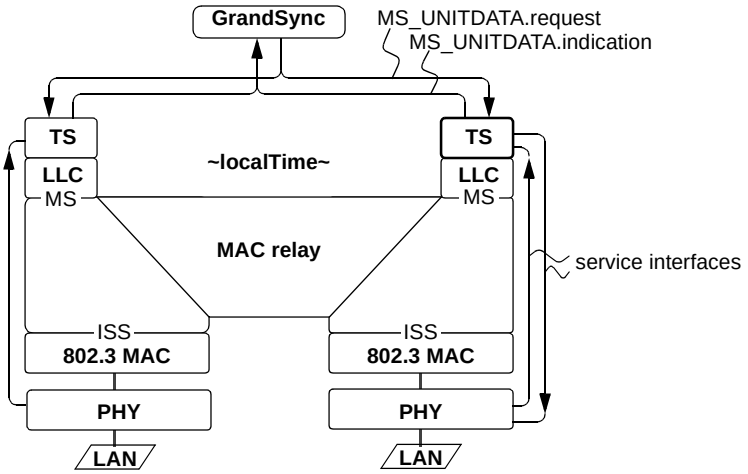

Figure 9.1—Radio interface model

The rxSync and txSync indications are localized communications between the MAC-and-PHY and are not directly visible to the a TimeSync state machines. Client-level interface parameters include the timing information, based on the formats illustrated within Figure 9.2.

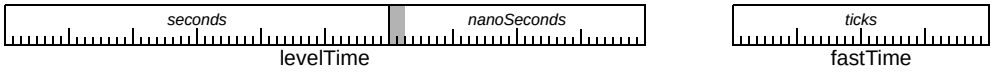

Figure 9.2—Formats of wireless-dependent times

9.1.2 Service interface overview

A sequence of 802.11v TimeSync service interface actions is illustrated in Figure 9.3. A periodic trigger is assumed to initiate the initial MLME_PRESENCE_REQUEST.request action. Processing of the returned MLME_PRESENCE_REQUEST.confirm triggers the following MLME_PRESENCE_RESPONSE.request action. The sequence completes with the final MLME_PRESENCE_RESPONSE.confirm action.

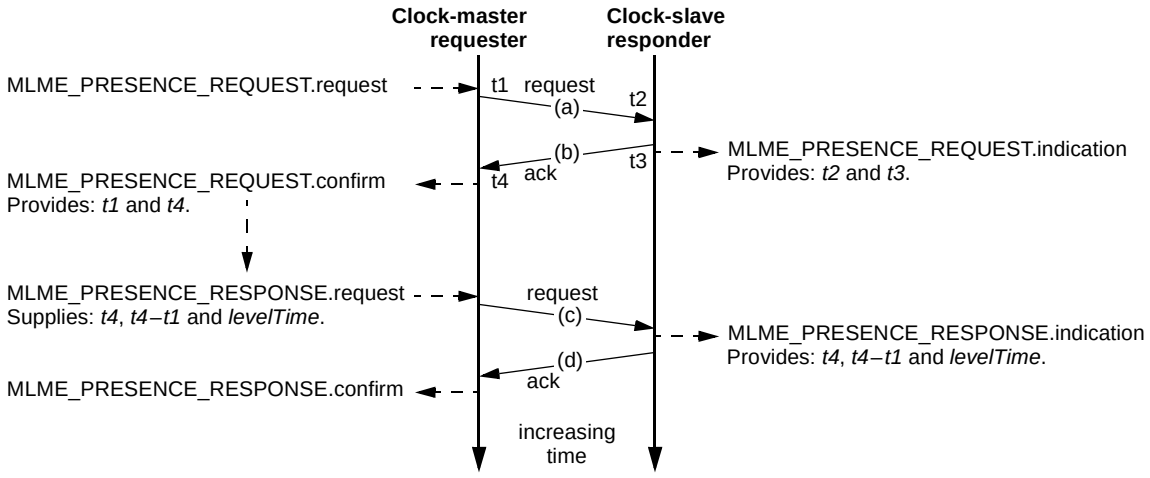

Figure 9.3—802.11v time-synchronization interfaces

The properties of these service interfaces are summarized below:

MLME_PRESENCE_REQUEST.request
- Generated periodically by the clock-master entity.
- Triggers a (Figure 9.3-a) request to update clock-slave resident timing parameters.

MLME_PRESENCE_REQUEST.indication
- Generated in response to receiving a (Figure 9.3-a) request.
- Provides *t2* and *t3* timing information to the clock-slave entity.

MLME_PRESENCE_REQUEST.confirm
- Generated after the (Figure 9.3-b) ack is returned.
- Provides *time1* and *time4* timing information to the clock-master entity.

MLME_PRESENCE_RESPONSE.request
- Generated shortly after processing a returned (Figure 9.3-b) ack
- Triggers a (Figure 9.3-c) request to update clock-slave resident timing parameters.

MLME_PRESENCE_RESPONSE.indication
- Generated in response to receiving a (Figure 9.3-c) request.
- Provides *time4*, *time4–time1*, and *levelTime* information to the clock-slave entity.

MLME_PRESENCE_RESPONSE.confirm
- Generated after the (Figure 9.3-d) ack is returned.
- Confirms completion of the time-synchronization exchange.

9.2 Service interface definitions

9.2.1 MLME_PRESENCE_REQUEST.request

9.2.1.1 Function

The service interface triggers the sending of a (Figure 9.3-a) request from the clock-master requester to the clock-slave responder. A snapshot of the transmit time is also saved for deferred transmission/processing.

9.2.1.2 Semantics of the service primitive

The semantics of the primitives are as follows:

```
MLME_PRESENCE_REQUEST.request {
    other_arguments      // Arguments for other purposes
}
```

9.2.1.3 When generated

Generated periodically by a transmit TS port as the first phase of a time-sync information transfer.

9.2.1.4 Effect of receipt

Upon receipt by a receive TS port, an MLME_PRESENCE_REQUEST.indication is invoked; times of the arriving (Figure 9.3-a) request and departing (Figure 9.3-b) ack are both passed within this indication.

9.2.2 MLME_PRESENCE_REQUEST.indication

9.2.2.1 Function

The receipt of a (Figure 9.3-a) request from the clock-master requester triggers the return of an (Figure 9.3-b) ack from the clock-slave port. The transfer of an MLME_PRESENCE_REQUEST.indication to the clock-slave provides snapshots of the (Figure 9.3-a) request receive time as well as the following (Figure 9.3-b) ack transmission time.

9.2.2.2 Semantics of the service primitive

The semantics of the primitives are as follows:

```
MLME_PRESENCE_REQUEST.indication {
    other_arguments,     // Arguments for other purposes
    time_t2,             // Arrival time of indication
    time_t3              // Departure time of confirm
}
```

9.2.2.3 When generated

Generated by the receipt of a (Figure 9.3-a) request during the first phase of a time-sync transfer.

9.2.2.4 Effect of receipt

Upon receipt, the times of the arriving (Figure 9.3-a) request and (Figure 9.3-b) ack are both saved for deferred processing.

9.2.3 MLME_PRESENCE_REQUEST.confirm

9.2.3.1 Function

The receipt of an (Figure 9.3-b) ack at the clock-master requester triggers the transfer of an MLME_PRESENCE_REQUEST.confirm message. The transmit time of the original (Figure 9.3-a) request and the receive time of the recent (Figure 9.3-b) ack are both returned for inclusion in following service primitives.

9.2.3.2 Semantics of the service primitive

The semantics of the primitives are as follows:

```
MLME_PRESENCE_REQUEST.indication {
    other_arguments,    // Arguments for other purposes
    timeT1,             // Departure time of request
    timeT4              // Arrival time of confirm
}
```

9.2.3.3 When generated

Generated by the receipt of an (Figure 9.3-b) ack during the initial phases of a time-sync transfer.

9.2.3.4 Effect of receipt

Upon receipt, the transmit time of the previous (Figure 9.3-a) request and receive time of the recent (Figure 9.3-b) ack are both saved for deferred processing.

9.2.4 MLME_PRESENCE_RESPONSE.request

9.2.4.1 Function

After the initial phases, a clock-master requester triggers the transfer of an MLME_PRESENCE_RESPONSE.request. The transmit time of the original (Figure 9.3-a) request, the receive time of the recent (Figure 9.3-b) ack, and the current time-sync related information are all included in the service primitives.

9.2.4.2 Semantics of the service primitive

The semantics of the primitives are as follows:

```
MLME_PRESENCE_REQUEST.request {
    other_arguments,    // Arguments for other purposes
    timeT4              // Arrival time of REQUEST.confirm
    timeT4T1,           // Round-trip time
    level_time,         // Current media-dependent time
}
```

9.2.4.3 When generated

Triggered at the clock-master by the servicing of an MLME_PRESENCE_RESPONSE.confirm message, indicating the completion of the initial time-sync phases.

9.2.4.4 Effect of receipt

Upon receipt, an MLME_PRESENCE_RESPONSE.indication is invoked, to provide the clock-slave with sufficient information to send a GrandSync message.

9.2.5 MLME_PRESENCE_RESPONSE.indication

9.2.5.1 Function

Additional information is provided to a clock-slave port. Along with previous information (saved earlier for deferred processing), the clock-slave has sufficient information to send a GrandSync message.

9.2.5.2 Semantics of the service primitive

The semantics of the primitives are as follows:

```
MLME_PRESENCE_RESPONSE.indication {
    other_arguments,    // Arguments for other purposes
    timeT4              // Arrival time of REQUEST.confirm
    timeT4T1,           // Round-trip time
    level_time,         // Current media-dependent time
}
```

9.2.5.3 When generated

Triggered at the clock-slave by the receipt of a (Figure 9.3-c) request, nearing the completion of the final time-sync phases.

9.2.5.4 Effect of receipt

Upon receipt, an MLME_PRESENCE_RESPONSE.indication is invoked, thus providing the clock-slave with sufficient information to send a GrandSync message.

9.2.6 MLME_PRESENCE_RESPONSE.confirm

9.2.6.1 Function

Confirmation is provided to the clock-master, confirming clock-slave has sufficient information to send a GrandSync message.

9.2.6.2 Semantics of the service primitive

The semantics of the primitives are as follows:

```
MLME_PRESENCE_RESPONSE.confirm {
    other_arguments,    // Arguments for other purposes
}
```

9.2.6.3 When generated

Triggered at the clock-master by the receipt of a (Figure 9.3-d) ack, at the completion of the final time-sync phases.

9.2.6.4 Effect of receipt

Upon receipt, the clock-master is provided with a time-sync success status.

9.3 TimeSyncRxRadio state machine

9.3.1 Function

The TimeSyncRxRadio state machine consumes primitives provided by the MAC service interface and (in response) generates MAC-relay frames.

9.3.2 State machine definitions

NULL
A constant indicating the absence of a value that (by design) cannot be confused with a valid value.
queue values
Enumerated values used to specify shared FIFO queue structures.
Q_MR_HOP—Queue identifier associated with MAC frames sent into the relay.
Q_S1_IND—Queue identifier for MLME_PRESENCE_REQUEST.indication parameters.
Q_S2_IND—Queue identifier for MLME_PRESENCE_RESPONSE.indication parameters.

9.3.3 State machine variables

args1
A set of values returned within the MLME_PRESENCE_REQUEST.indication service primitive:
fastTime2—A local-timer snapshot at the time of (Figure 9.3-a) request reception.
fastTime3—A local-timer snapshot at the time of (Figure 9.3-b) ack transmission.
args2
A set of values provided to the MLME_PRESENCE_RESPONSE.indication service primitive:
fastTime4—A neighbor-timer snapshot at the time of (Figure 9.3-b) ack reception.
fastTimed—A neighbor-timer snapshot corresponding to a time difference:
(Figure 9.3-c request transmission) – (Figure 9.3-a transmission)
levelTime—Grand-master synchronized time at the *fastTime4* neighbor-time snapshot.
stationTime
See 6.3.3.
ePtr
Points to entity-specific storage, comprising the following:
rxFastTime2—Saved *args1.fastTime2* value.
rxFastTime3—Saved *args1.fastTime3* value.
rxFastTime4—Saved *args2.fastTime4* value.
rxFastTimed—Saved *args2.fastTimed* value.
rxLevelTime—Saved *args2.levelTime* value.
lapseTime
An variable representing the time-sync frame transmission delay.
radioDelay
An variable representing the round-trip transmission delay.
tsPtr
A pointer to service-data-unit portion of *txInfo* storage.
turnRound
An variable representing the difference between local time-sync transmit and receive times.

txInfo
Storage for information sent to the GrandSync entity, comprising:
destination_address, *source_address*, *service_data_unit*
Where *service_data_unit* comprises:
errorTime, *sourcePort*, *function*, *grandTime*, *hopCount*,
localTime, *protocolType*, *precedence*, *syncInterval*, *version*
txPtr
A pointer to *txInfo* storage.

9.3.4 State machine routines

Dequeue(*queue*)
Enqueue(*queue, info*)
See 6.3.4.
SourcePort(*entity*)
See 7.3.3.
StationTime(*entity*)
TimeSyncSdu(*info*)
See 6.3.4.

9.3.5 TimeSyncRxRadio state table

The TimeSyncRxRadio state machine consumes MAC-provided service-primitive information and forwards adjusted frames to the MAC-relay function, as illustrated in Table 9.1.

Table 9.1—TimeSyncRxRadio state machine table

Current		Row	Next	
state	**condition**		**action**	**state**
START	(req1 = Dequeue(Q_S1_IND)) != NULL	1	ePtr->rxTurnRound = req1.timeT3 – req1.timeT2;	WAIT
	—	2	stationTime = StationTime(ePtr); radioTime = RadioTime(ePtr);	START

WAIT	(rxInfo = Dequeue(Q_S2_IND)) != NULL	3	turnAround = rxPtr->timeT4T1;
			radioDelay= (turnRound– ePtr->rxTurnRound)/2;
			lapseTime = RadioToStation(radioDelay);
			txPtr->destination_address = AVB_MCAST;
			txPtr->source_address = TBD;
			tsPtr->protocolType = AVB_TYPE;
			tsPtr->function = AVB_FUNCTION;
			tsPtr->version = AVB_VERSION;
			tsPtr->precedence = rsPtr->precedence;
			tsPtr->sourcePort = SourcePort(ePtr);
			tsPtr->hopCount = rsPtr->hopCount;
			tsPtr->grandTime = rsPtr->grandTime;
			tsPtr->errorTime = rsPtr->errorTime;
			tsPtr->snapTime = stationTime – lapseTime;
			Enqueue(Q_MS_IND, txPtr);
				START

	—	4	—	—

Row 9.1-1: Wait until indication parameters become available.
Row 9.1-2: Update snapshot values based on MLME_PRESENCE_REQUEST.indication parameters.

Row 9.1-3: Wait until indication parameters become available.
Row 9.1-4: Update snapshot values based on MLME_PRESENCE_RESPONSE.indication parameters. Based on those parameters, generate a timeSync frame for MAC-relay transmission.

9.4 TimeSyncTxRadio state machine

9.4.1 Function

The TimeSyncTxRadio state machine consumes MAC-relay frames and (in response) generates calls to the time-synchronization related MAC service interface.

9.4.2 State machine definitions

NULL
A constant indicating the absence of a value that (by design) cannot be confused with a valid value.
queue values
Enumerated values used to specify shared FIFO queue structures.
Q_MR_HOP—The queue identifier associated with MAC frames sent into the relay.
Q_RX_MAC—The queue identifier associated with the received MAC frames.
Q_RX_SYNC—The queue identifier associated with rxSync, sent from the lower levels.

9.4.3 State machine variables

args1
A set of values returned within the MLME_PRESENCE_REQUEST.confirm service primitive:
time1—A local-timer snapshot at the time of (Figure 9.3-a) request transmission.
time2—A local-timer snapshot corresponding to the time of (Figure 9.3-c) request reception.
args2
A set of values provided to the MLME_PRESENCE_REQUEST.request service primitive:
time1—The value of the previously returned *args1.time1* value.
timed—The difference of previously returned values: *args1.time4 – args1.time1*.
levelTime—The value of *grandTime* associated with the returned *args1.time1* timer.
curentTime
A shared value representing current time. There is one instance of this variable for each station. Within the state machines of this standard, this is assumed to have two components, as follows:
seconds—An 8-bit unsigned value representing seconds.
fraction—An 40-bit unsigned value representing portions of a second, in units of 2^{-40} second.
ePtr
A pointer to the entity-specific storage containing the following:
lastTime—The last transmit time, saved for pacing transmissions.
rxSaved—A copy of the last received GransSync parameters.
syncInterval—The expected interval between successive time-sync transmissions.
timed—Timing information saved from previous GrandSync requests.
txSnapTime1—… .
txRoundTrip—… .
reqArgs
MLME_PRESENCE_REQUEST.request parameters unrelated to time-synchronization services.
resArgs
MLME_PRESENCE_RESPONSE.request parameters unrelated to time-synchronization services.

rsPtr
A pointer to service-data-unit portion of *rxInfo* storage.
rxInfo
Storage for received time-sync messages from the GrandSync entity, comprising:
destination_address, source_address, service_data_unit
Where *service_data_unit* comprises:
errorTime, function, grandTime, hopCount,
precedence, protocolType, snapTime, syncInterval, version
rxPtr
A pointer to *rxInfo* storage.
ssPtr
A pointer to the service-data-unit portion of the *ePtr->rxSaved* storage
sxPtr
A pointer to the *ePtr->rxSaved* storage.

9.4.4 State machine routines

Dequeue(*queue*)
Enqueue(*queue, info*)
See 6.3.4.
NextTimed(*stationTime, rxSyncInterval, syncInterval, timed*)
See 7.5.4.
StationTime(*entity*)
See 7.3.3.
TimeSyncSdu(*info*)
See 6.3.4.

9.4.5 TimeSyncTxRadio state table

NOTE—This state machine is preliminary; sequence timeouts have not been considered.

The TimeSyncTxRadio state machine includes a media-dependent timeout, which effectively disconnects a clock-slave port in the absence of received timedSync frames, as illustrated in Table 9.2.

Table 9.2—TimeSyncTxRadio state table

Current		Row	Next	
state	**condition**		**action**	**state**
START	(rxInfo = Dequeue(Q_MS_REQ)) != NULL	1	—	SINK
	(stationTime – ePtr->lastTime) > T10ms	2	ePtr->lastTime = stationTime;	SEND

	(con1 = Dequeue(Q_S1_CON)) != NULL	3	ePtr->txSnapTime1 = con1.ticksTime1;
			ePtr->txRoundTrip =
			con1.ticksTime4 – con1.ticksTime1;
			phase2 = TRUE;
				WAIT

	—	4	stationTime = StationTime(ePtr); radioTime = RadioTime(ePtr);	START

Table 9.2—TimeSyncTxRadio state table

Current		Row	Next	
state	**condition**		**action**	**state**

SINK	TimeSyncSdu(rsPtr)	5	ePtr->rxSaved = rxInfo;
			timePtr->grandTime = rsPtr->grandTime;
			timePtr->errorTime = rsPtr->errorTime;
			timePtr->stationTime = stationTime;
				SERVE

	—	6	EnqueueService(Q_ES_REQ, rxPtr);	START

WAIT1	phase2 == TRUE	7	lapseTime = radioTime – ePtr->txSnapTime4;
			sendTime =
			stationTime – (lapseTime * RADIO_TIME);
			nextTimes = NextTimed(stationTime,
			ePtr->rxSyncInterval, ePtr->syncInterval,
			ePtr->rxTimes);
			req2.ticksTime4 = ePtr->txSnapTime4;
			req2.roundTrip = ePtr->txRoundTrip;
			req2.levelTime =
			GrandToRadio(nextTimes.grandTime);
			req2.errorTime = nextTimes.errorTime;
			// Should more be transferred?
			req2.precedence = ssPtr->precedence;
			req2.hopCount = ssPtr->hopCount;
			EnqueueService(Q_S2_REQ, req2);
				WAIT2

	—	8	—	—
WAIT2	(args2 = Dequeue(Q_S2_CON)) == NULL	9	—	WAIT1
	—	10	—	START

Row 9.2-1: Relayed frames are further checked before being processed.
Row 9.2-2: Initiate periodic service-interface primitive calls.
Row 9.2-3: Save parameters from a service-interface primitive call.
Row 9.2-4: Wait for the next change-of-state.

Row 9.2-5: Parameters from timeSync messages are saved.
Row 9.2-6: The contents of non-timeSync messages are passed through.

Row 9.2-7: Wait for parameters arriving through the MLME_PRESENCE_REQUEST.confirm interface.
Row 9.2-8: Transmit parameters through the MLME_PRESENCE_RESPONSE.request interface.

Row 9.2-9: Wait for parameters arriving through the MLME_PRESENCE_RESPONSE.confirm interface.
Row 9.2-10: Confirm completion and continue.

10. Ethernet-PON state machines

NOTE—This clause is based on indirect knowledge of the Ethernet-PON specifications, as interpreted by the author, and have not been reviewed by the 802.1 or 802.3 WGs. The intent was to provide a forum for evaluation of the GrandSync interfaces, while also triggering discussion of 802.3-PON design details. As such, the contents are highly preliminary and subject to change.

10.1 Overview

This clause specifies the state machines that support Ethernet-PON based bridges. The operations are described in an abstract way and do not imply any particular implementations or any exposed interfaces. There is not necessarily a one-to-one correspondence between the formal specification and the interfaces in any particular implementation.

10.1.1 Link-dependent indications

The TimeSyncPon state machines have knowledge of network-local synchronized *ticksTime* timers. With this knowledge, the TimeSyncPon state machines can operated on frames received from the LLC, as illustrated in Figure 10.1. Link-dependent indications could be required for bridge ports attached to alternative media.

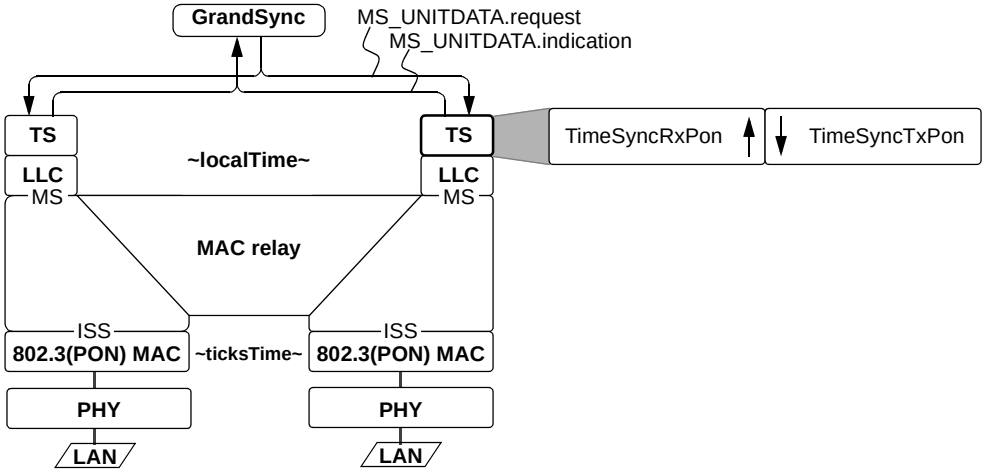

Figure 10.1—PON interface model

The *localTime* values are represented as timers that are incremented once every 16 ns interval, as illustrated on the left side of Figure 10.2. Each synchronized local timer is roughly equivalent to a 6-bit *sec* (seconds) field and a 26-bit *fraction* (fractions of second) field timer, as illustrated on the right side of Figure 10.2.

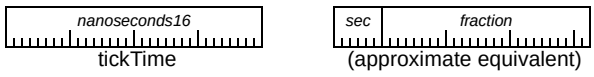

Figure 10.2—Format of PON-dependent times

The Ethernet-PON MAC is supplied with frame transmit/receive snapshots, but these are transparent-to and not-used-by the TimeSync state machine. Instead, these are used to synchronize the *ticksTime* values in associated MACs and the TimeSyncPon state machines have access to these synchronized *ticksTime* values.

10.1.2 Link-delay compensation

The synchronized-clock accuracies are influenced by the transmission delays between ports. To compensate for these transmission delays, the receive port is normally responsible for compensating {*grandTime, ticksTime*} affiliations by the (assumed to be constant) frame transmission delay.

The Ethernet-PON MAC provides access to a subnet-synchronized media-dependent *ticksTime* timer. Thus, the {*grandTime, ticksTime*} affiliation specified the transmitter remains valid within the receiver and transmission-delay compensation (in this sense) is unnecessary.

However, each time-sync related GrandSync message includes an {*grandTime, stationTime*} affiliation, wherein *stationTime* represents a recent snapshot of a shared station-local clock. To provide such an affiliation, the transmission delay (measured as a *ticksTime* difference) is scaled and subtracted from the *stationTime* that is sampled when the conversion is performed. Thus, no additional receiver snapshot hardware is required.

10.2 timeSyncPon frame format

The timeSyncPon frames facilitate the synchronization of neighboring clock-master and clock-slave stations. The frame, which is normally sent at 10 ms intervals, includes time-snapshot information and the identity of the network's clock master, as illustrated in Figure 10.3. The gray boxes represent physical layer encapsulation fields that are common across Ethernet frames.

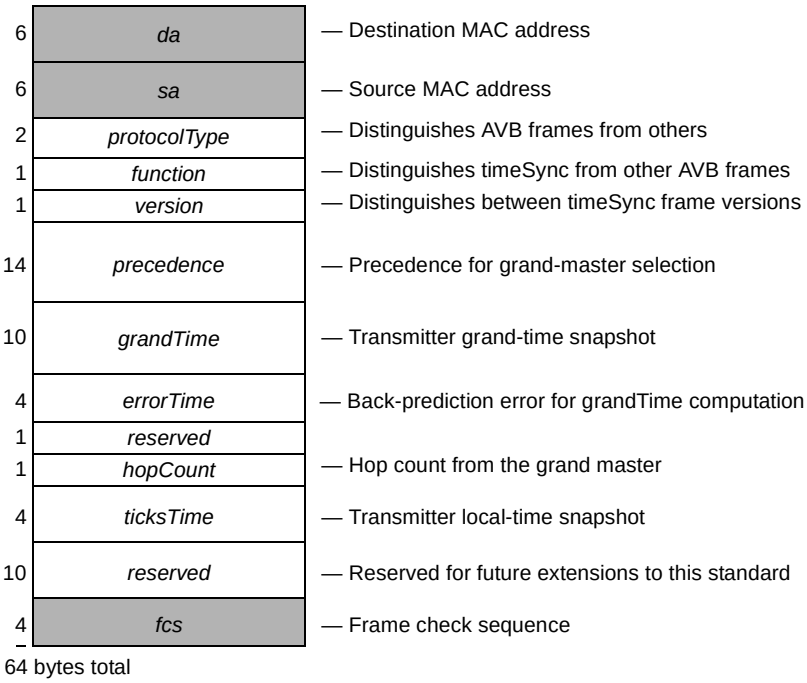

Figure 10.3—timeSyncPon frame format

The 48-bit ***da*** (destination address), 48-bit ***sa*** (source address) field, 16-bit ***protocolType,*** 8-bit ***function***, 8-bit ***version***, 14-byte ***precedence***, 80-bit ***grandTime***, 32-bit ***errorTime***, and 8-bit ***hopCount*** fields are specified in 6.2.1.2.

10.2.1 *ticksTime*: A value representing local time in units of a 16 ns timer ticks, as illustrated in Figure 10.4.

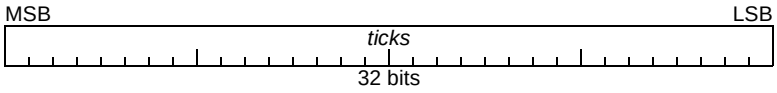

Figure 10.4—tickTime format

10.3 TimeSyncRxPon service interface primitives

10.3.1 ES_UNITDATA.indication

10.3.1.1 Function

Provides the TimeSyncRxPon entity with clock-synchronization parameters derived from arriving time-sync frames.

10.3.1.2 Semantics of the service primitive

The semantics of the primitives are as follows:

```
ES_UNITDATA.indication {
    destination_address,      // Destination address
    source_address,           // Optional
    priority,                 // Forwarding priority
    service_data_unit,        // Delivered content
    {                         // Contents of the service_data_unit
        protocolType,         // Distinguishes AVB frames from others
        function,             // Distinguishes between timeSync and other AVB frames
        version,              // Distinguishes between timeSync frame versions
        precedence,           // Precedence for grand-master selection
        grandTime,            // Global-time snapshot (1-cycle delayed)
        errorTime,            // Accumulated grandTime error
        hopCount,             // Distance from the grand-master station
        ticksTime             // Local-time snapshot (1-cycle delayed)
    }
}
```

The parameters of the MA_DATA.indication are described as follows:

The 48-bit **destination_address**, 48-bit **source_address**, and 8-bit **priority** field are specified in 6.2.1.2.

The service_data_unit consists of subfields; for content exchanged with the GrandTime protocol entity, these fields include the following.

The 16-bit ***protocolType***, 8-bit ***function***, 8-bit ***version***, 14-byte ***precedence***, 80-bit ***grandTime***, 32-bit ***errorTime***, and 8-bit ***hopCount*** fields are specified in 6.2.1.2.

10.3.1.2.1 *frameCount*: An 8-bit consistency-check field that increments on successive frames.

10.3.1.2.2 *ticksTime*: A 32-bit field that specifies the local free-running time within this subnet, when the previous timeSync frame was received (see 10.2.1).

10.3.1.3 When generated

The service primitive is generated upon the receipt of a time-sync related frame delivered from the MAC. The intent is to facilitate reformatting and snapshot-time adjustment before the content of that frame is delivered to the ClockMaster and TS entities.

10.3.1.4 Effect of receipt

The service primitive invokes processing of time-sync related content and forwarding of unrelated content. For time-sync related content, the processing included reformatting and compensation for receive-link transmission delays.

10.4 TimeSyncRxPon state machine

10.4.1 Function

The TimeSyncRxPon state machine is responsible for receiving MAC-supplied frames, converting their media-dependent parameters, and sending normalized MAC-relay frames. The sequencing of this state machine is specified by Table 10.1; details of the computations are specified by the C-code of Annex G.

10.4.2 State machine definitions

NULL
A constant indicating the absence of a value that (by design) cannot be confused with a valid value.
queue values
Enumerated values used to specify shared FIFO queue structures.
Q_MS_IND—Associated with the GrandSync entity (see 6.3.2).
Q_ES_IND—The queue identifier associated with the received MAC frames.

10.4.3 State machine variables

ePtr
A pointer to a entity-specific data structure comprising the following:
syncInterval—The expected interval between time-sync frame transmissions.
lapseTime
A value representing the time lapse between transmission of reception of the timeSync frame.
rsPtr
A pointer to the service-data-unit portion of the *rxInfo* storage.
rxInfo
A storage location for received service-interface parameters, comprising:
destination_address, *source_address*, *service_data_unit*
Where *service_data_unit* comprises:
errorTime, *function*, *grandTime*, *hopCount*,
precedence, *protocolType*, *ticksTime*, *version*
rxPtr
A pointer to the *rxInfo* storage location.
tsPtr
A pointer to the service-data-unit portion of the *txInfo* storage.

txInfo

A storage location for to-be-transmitted MS_UNITDATA.indication parameters, comprising:

destination_address, *source_address*, *service_data_unit*

Where *service_data_unit* comprises:

errorTime, *function*, *grandTime*, *hopCount*, *precedence*, *protocolType*, *snapTime*, *ticksTime*, *version*

txPtr

A pointer to the *txInfo* storage location.

10.4.4 State machine routines

Dequeue(*queue*)

Enqueue(*queue*, *info*)

See 6.3.4.

SourcePort(*entity*)

See 7.3.3.

TicksTime(*entity*)

Returns the value of the station's shared media-dependent subnet-synchronized timer.

This 32-bit timer is incremented once at the end of each 16 ns interval.

TicksToTime(*ticks*)

Returns the time duration of *stationTime* that corresponds to the time duration specified in *ticks*.

TimeSyncSdu(*info*)

See 6.3.4.

10.4.5 TimeSyncRxPon state machine table

The TimeSyncRxPon state machine associates PHY-provided sync information with arriving timeSync frames and forwards adjusted frames to the MAC-relay function, as illustrated in Table 8.2.

Table 10.1—TimeSyncRxPon state machine table

Current		Row	Next	
state	**condition**		**action**	**state**
START	(rxInfo = Dequeue(Q_RX_MAC)) != NULL	1	—	TEST
	—	2	stationTime = StationTime(ePtr); ticksTime = PonTime(ePtr);	START
TEST	TimeSyncSdu(rsPtr)	3	*rxPtr = rxInfo;	SYNC
	—	4	Enqueue(Q_MS_IND, txInfo);	START

SYNC	rsPtr->hopCount != LAST_HOP	5	lapseTime = ticksTime – rsPtr->ticksTime);
			compTime =
			stationTime –TicksToTime(lapseTime);
			txPtr->destination_address =
			rxPtr->destination_address;
			txPtr->source_address = rxPtr->source_address;
			tsPtr->protocolType = rsPtr->protocolType;
			tsPtr->function = rsPtr->function;
			tsPtr->version = rsPtr->version;
			tsPtr->precedence = rsPtr->precedence;
			tsPtr->grandTime = rsPtr->grandTime;
			tsPtr->errorTime = rsPtr->errorTime;
			tsPtr->snapTime = compTime;
			tsPtr->sourcePort = SourcePort(ePtr);
			tsPtr->hopCount = frame.hopCount;
			tsPtr->syncInterval = ePtr->syncInterval;
			Enqueue(Q_MS_IND, txInfo);
				START

	—	6	—	

Row 10.1-1: Initiate inspection of frames received from the lower-level MAC.
Row 10.1-2: Wait for the next frame to arrive.

Row 10.1-3: The timeSync frames are checked further.
Row 10.1-4: The non-timeSync frames are passed through.

Row 10.1-5: Active timeSync frames are adjusted for transfer delays and passed through.
Row 10.1-6: Overly-aged timeSync frames are discarded.

10.5 TimeSyncTxPon service interface primitives

10.5.1 ES_UNITDATA.request

10.5.1.1 Function

Provides the Ethernet-PON entity with clock-synchronization parameters for constructing departing time-sync frames.

10.5.1.2 Semantics of the service primitive

The semantics of the primitives are as follows:

```
ES_UNITDATA.request
{
    destination_address,      // Destination address
    source_address,           // Optional
    priority,                 // Forwarding priority
    service_data_unit,        // Delivered content
    {                         // Contents of the service_data_unit
        protocolType,         // Distinguishes AVB frames from others
        function,             // Distinguishes between timeSync and other frames
        version,              // Distinguishes between timeSync frame versions
        precedence,           // Precedence for grand-master selection
        grandTime,            // Global-time snapshot (1-cycle delayed)
        errorTime,            // Accumulated grandTime error
        hopCount,             // Distance from the grand-master station
        ticksTime             // Local-time snapshot (1-cycle delayed)
    }
}
```

The parameters of the MA_UNITDATA.request are described in 10.3.1.2.

10.5.1.3 When generated

The service primitive is generated at a periodic rate, for the purposes of synchronizing the *grandTime* values resident in other stations.

10.5.1.4 Effect of receipt

The service primitive triggers the transmission of a timeSync frame on the affiliated port.

10.6 TimeSyncTxPon state machine

10.6.1 Function

The TimeSyncTxPon state machine is responsible for modifying time-sync MS_UNITDATA.request parameters to form timeSync frames for transmission over the attached link.

10.6.2 State machine definitions

NULL
: A constant indicating the absence of a value that (by design) cannot be confused with a valid value.

queue values

Enumerated values used to specify shared FIFO queue structures.

Q_MS_REQ—Associated with the GrandSync entity (see 6.3.2).

Q_ES_REQ—The queue identifier associated with frames sent to the MAC.

T10ms

A constant the represents a 10 ms value.

10.6.3 State machine variables

ePtr

A pointer to a entity-specific data structure comprising the following:

lastTime—The last message-transmit time; used to space periodic transmissions.

rxSaved—A copy of the last received GrandSync parameters.

syncInterval—The expected interval between time-sync frame transmissions.

timed[3]—Recently saved time events, each consisting of the following:

grandTime—A previously sampled grand-master synchronized time.

errorTime—The residual error associated with the sampled *grandTime* value.

snapTime—The *stationTime* affiliated with the sampled *grandTime* value.

rsPtr

A pointer to the service-data-unit portion of *rxInfo* storage.

rxInfo

Storage for the contents of GrandSync messages, comprising:

destination_address, *source_address*, *service_data_unit*

Where *service_data_unit* comprises:

errorTime, *function*, *grandTime*, *hopCount*, *precedence*, *protocolType*, *snapTime*, *syncInterval*, *version*

rxPtr

A pointer to the *rxInfo* storage.

stationTime

See 6.3.3.

ssPtr

A pointer to the service-data-unit portion of the *ePtr->rxSaved* storage

sxPtr

A pointer to the *ePtr->rxSaved* storage.

tsPtr

A pointer to the service-data-unit portion of *txInfo* storage.

txInfo

Storage for a to-be-transmitted MAC frame, comprising:

destination_address, *source_address*, *service_data_unit*

Where *service_data_unit* comprises:

errorTime, *function*, *grandTime*, *hopCount*, *protocolType*, *precedence*, *ticksTime*, *version*

txPtr

A pointer to the *txInfo* storage.

ticksTime

A 32-bit shared value representing Ethernet-PON media-dependent time; incremented every 16 ns.

10.6.4 State machine routines

Dequeue(*queue*)

Enqueue(*queue, info*)

See 6.3.4.

NextTimed(*stationTime*, *rxSyncInterval*, *syncInterval*, *timed*)

See 7.5.4.

SourcePort(*entity*)
See 7.3.3.
StationTime(*entity*)
See 6.3.4.
TicksTime(*entity*)
See 10.4.4.
TimeSyncSdu(*info*)
See 6.3.4.

10.6.5 TimeSyncTxPon state machine table

The TimeSyncTxPon state machine includes a media-dependent timeout, which effectively disconnects a clock-slave port in the absence of received timeSyncPon frames, as illustrated in Table 10.2.

Table 10.2—TimeSyncTxPon state machine table

Current		Row	Next	
state	**condition**		**action**	**state**
START	(rxInfo = Dequeue(Q_MS_REQ)) != NULL	1	—	SINK
	(stationTime – ePtr->lastTime) > T10ms	2	ePtr->lastTime = stationTime;	SEND
	—	3	stationTime = StationTime(ePtr); ticksTime = TicksTime(ePtr);	START
SINK	TimeSyncSdu(rsPtr)	4	ePtr->rxSaved = rxInfo;	START
	—	5	Enqueue(Q_ES_REQ, rxPtr);	

SEND	—	6	nextTimes = NextTimed(stationTime, ssPtr->syncInterval, ePtr->syncInterval, ePtr->timed);
			txPtr->destination_address = sxPtr->destination_address;
			txPtr->source_address = sxPtr->source_address;
			tsPtr->protocolType = ssPtr->protocolType;
			tsPtr->function = ssPtr->function;
			tsPtr->version = ssPtr->version;
			tsPtr->precedence = ssPtr->precedence;
			tsPtr->hopCount = ssPtr->hopCount;
			tsPtr->grandTime = nextTimes.grandTime;
			tsPtr->errorTime = nextTimes.errorTime;
			tsPtr->ticksTime = ticksTime;
			Enqueue(Q_ES_REQ, txPtr);
				START

Row 10.2-1: Relayed frames are further checked before being processed.
Row 10.2-2: Transmit periodic timeSync frames.
Row 10.2-3: Wait for the next change-of-state.

Row 10.2-4: The timeSync message is saved and processed further.
Row 10.2-5: Non-timeSync messages are retransmitted in the standard fashion.

Row 10.2-6: Format and transmit the media-specific timeSync frame.