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.

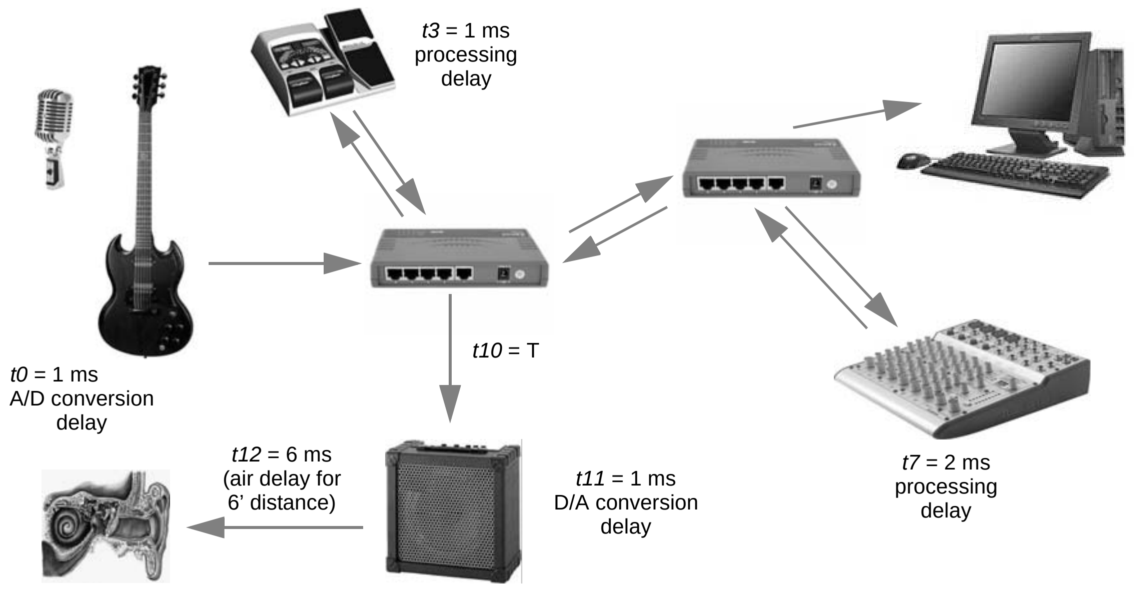

Figure 5.1—Garage jam session

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.

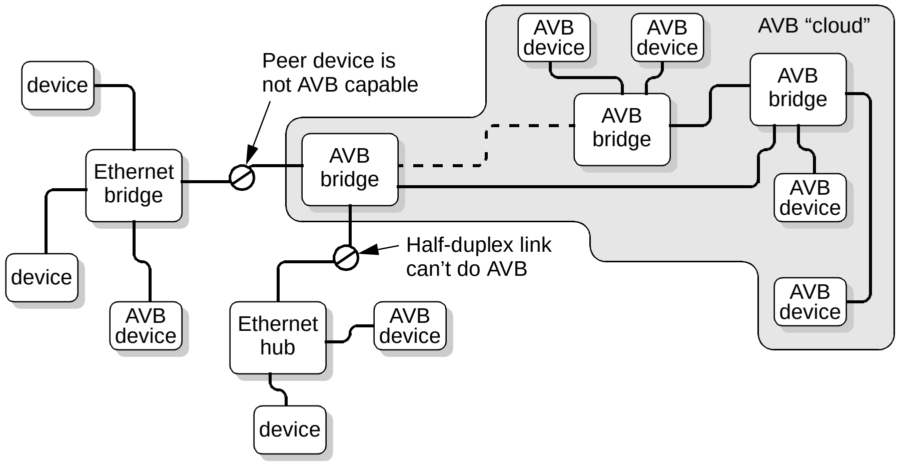

Figure 5.2—Possible looping topology

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) Firmware friendly. Clock offset and rate adjustments can be performed by low-rate firmware; analog PLLs are unnecessary within bridges (although necessary within some applications).
   3) Frequent. Frequent (nominally 100 Hz) interchanges reduces needs for overly precise clocks.

5.3 Grand-master hierarchies

5.3.1 Synchronized-system topologies

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.

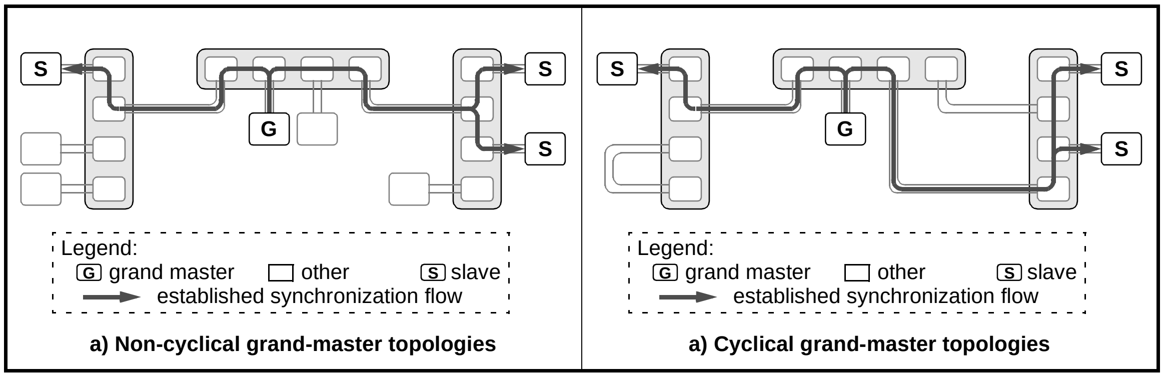
S
G
S
S
Legend:
G grand master
other
S slave
established synchronization flow
a) Non-cyclical grand-master topologies
S
G
S
S
Legend:
G grand master
other
S slave
established synchronization flow
a) Cyclical grand-master topologies

Figure 5.3—Synchronized-system topologies

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. Topologies are pruned into spanning tree with the grand-master at the root of the tree.
b) Distribution. Synchronized time is distributed through the spanning tree from the root.

5.3.2 Clock-content flows

A ClockSource entity provides time-reference information to a ClockSlave entity, wherein the time-reference information is converted to a common format and supplemented with a unique grand-master identity before flowing through the interconnect, as illustrated in Figure 5.4.

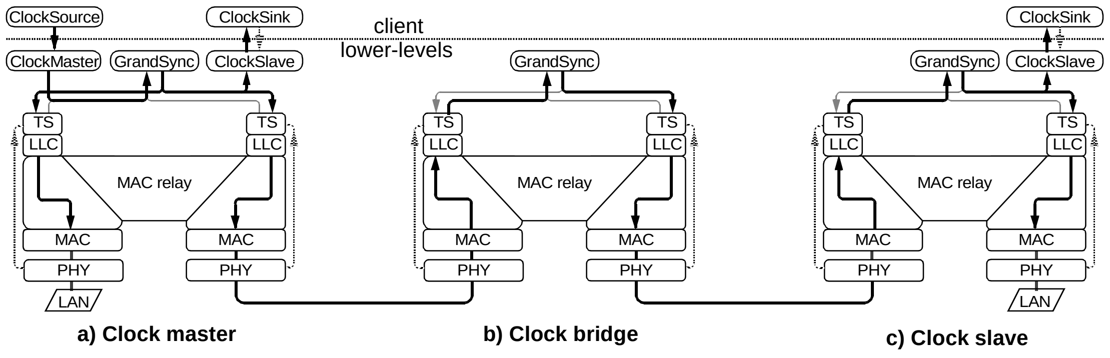

Figure 5.4—Time-synchronization flows

The PDUcontent flows through bridges to attached ClockSlaves, where the common content is converted to time-reference information for ClockSink entities.

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).

5.4 Grand-master selection

5.4.1 Announce-message content

As part of the grand-master selection process, announce messages are generated by clock-master capable stations. The announce messages transport clock-preference values to other stations. Stations forward only the announce message with the best observed clock-preference value to neighbor stations, allowing the overall best-preference value to be ultimately selected and known by all. The station with the best preference value ultimately becomes the grand-master.

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

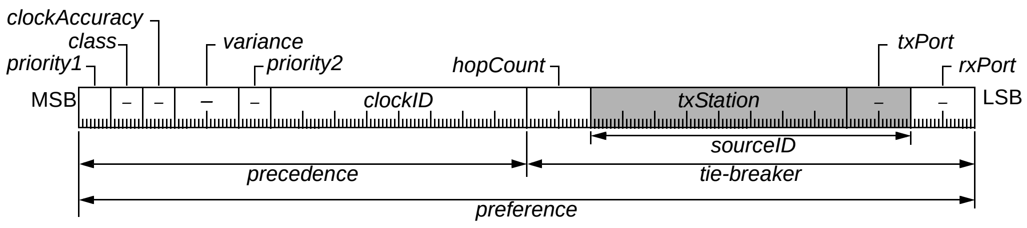

Figure 5.5—Grand-master preference

This format is conceptually similar to the format of the spanning-tree preference value, but larger identifiers and additional values are provided for the sake of generality.

5.4.2 Announce-message processing

Intermediate bridge entities are responsible for providing standard/centralized preference-selection and media-specific/distributed rejection-blocker entities, as illustrated in Figure 5.10. The preference-selection entity is responsible for selecting and echoing only the best announce messages; the rejection-blocker entity is responsible for blocking (unnecessary) reverse-flow announce messages.

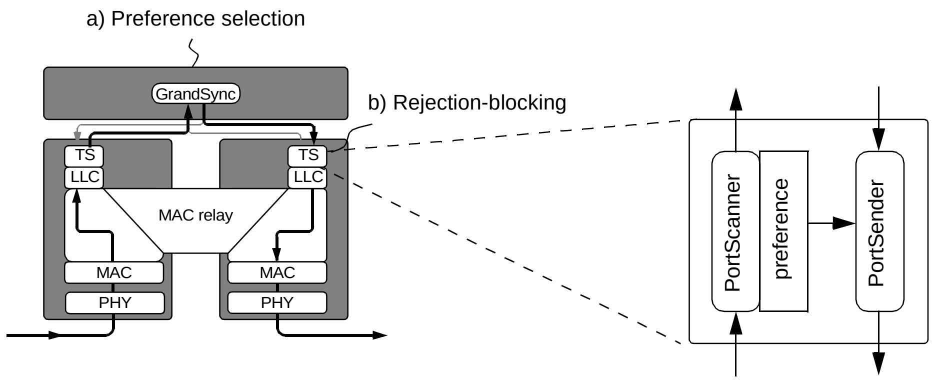

Figure 5.6—Intermediate-bridge responsibilities

```
// Performed by the GrandSync entity                                   (5.1)
while (FOREVER) {
  rxAnnounce = Get();
  if (rxAnnounce NULL) {
    if (rxAnnounce.preference < core.preference) {
      core.preference = rxAnnounce.preference;
      Put(rxAnnounce);
    }
  if (timeCurrent >= nextTimeout)
    grandPreference = ONES;
}
```

```
// Performed by the TS.PortScanner entity                              (5.2)
while (FOREVER) {
  rxAnnounce = Get();
  if (rxAnnounce != NULL) {
    rxPreference = Preference(rxAnnounce);
    if (rxAnnounce.preference < port.preference)
      port.preference = rxAnnounce.preference;
    Put(rxAnnounce);
  }
  if (timeCurrent >= nextTimeout)
    port.preference = ONES;
}
```

```
// Performed by the TS.Sender entity                                   (5.3)
while (FOREVER) {
  rxAnnounce = Get();
  if (rxAnnounce != NULL) {
    rxPreference = Preference(rxAnnounce);
    if (rxPreference < port.preference) {
      rxAnnounce.hopCount += 1;
      Put(rxAnnounce);
    }
  }
}
```

Questions:
a) Should we worry about transparent non-802.1as clocks? (No)
This simplifies the protocols, by:
eliminating txPort from the comparison
allowing isolation of txStation fields to below the TS entity
b) Should the media dependent txStation field be isolated below the TS entity? (Yes)
This field is only required for shared media and is thus media dependent.

A ClockSource entity provides time-reference information to a ClockSlave entity, wherein the time-reference information is converted to a common format and supplemented with a unique grand-master identity before flowing through the interconnect, as illustrated in Figure 5.4.

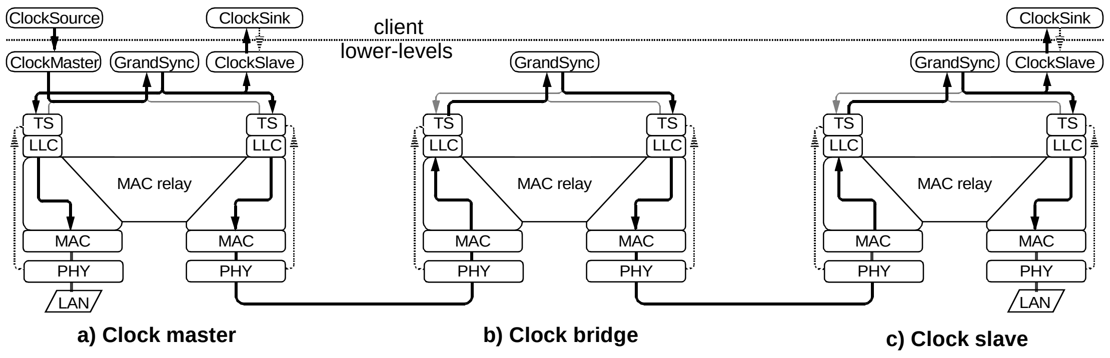

Figure 5.7—Time-synchronization flows

The grand-master station observes that its precedence is better than values received from its neighbors, as illustrated in Figure 5.8a. 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 5.8b. To avoid cyclical behaviors, a *hopCount* value is associated with preference values and is incremented before the best-precedence value is communicated to others.

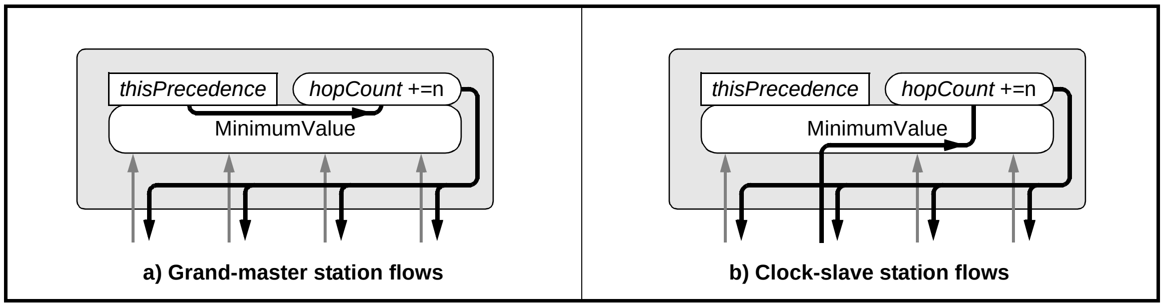

Figure 5.8—Grand-master precedence flows

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 5.4.

```
#define HOPS 255                                                        (5.4)
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.

5.4.3 Clock master/slave agents

Clock-synchronization information conceptually flows from a grand-master station to clock-slave stations, as illustrated in Figure 5.9a. A more detailed illustration shows pairs of synchronized clock-master and clock-slave components, as illustrated in Figure 5.9b. The active clock agents are illustrated as black-and-white components; the passive clock agents are illustrated as grey-and-white components.

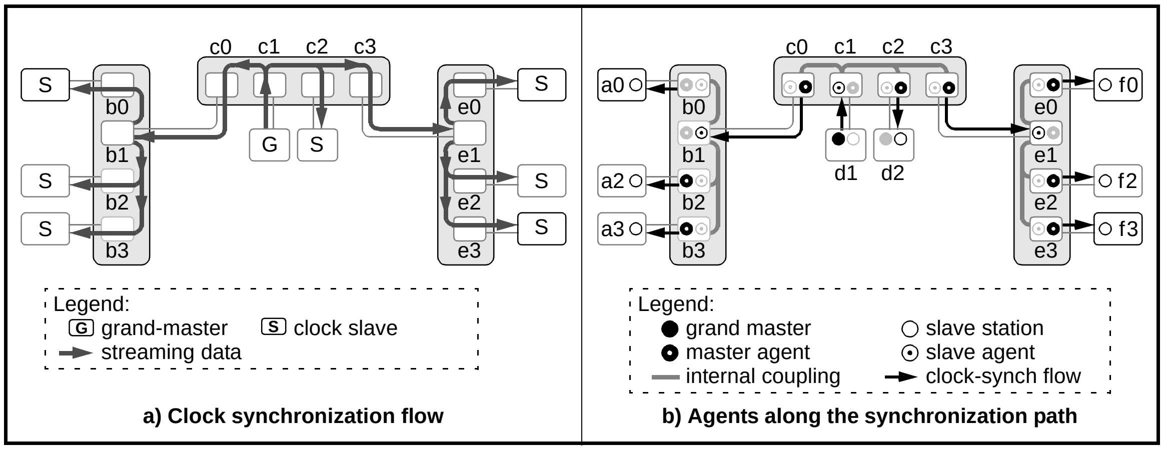

Figure 5.9—Hierarchical flows

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.

5.5 Clock-synchronization flows

Entities within the intermediate bridge (see Figure 5.4b) are responsible for performing three distinct (and largely decoupled) functions, as illustrated in Figure 5.10. A clock-slave port (see Figure 5.10a) is responsible for compensating for time-reference transmission delays between this station and its neighbor. The GrandSync entity (see Figure 5.10b) is responsible for selecting the timeSync PDUs from the grand-master station; only thus selected PDUs are forwarded to transmitter ports.

The clock-master port (see Figure 5.10c) is responsible for revising the GrandSync-supplied timeSync PDUs to supply the appropriate media-dependent service-interface parameters and/or frames. Since the transmission times and rates may differ from those on the clock-slave port, the clock-master port is responsible for interpolating/extrapolating between previously received time samples to generate parameters corresponding to the recently observed transmit-snapshot.

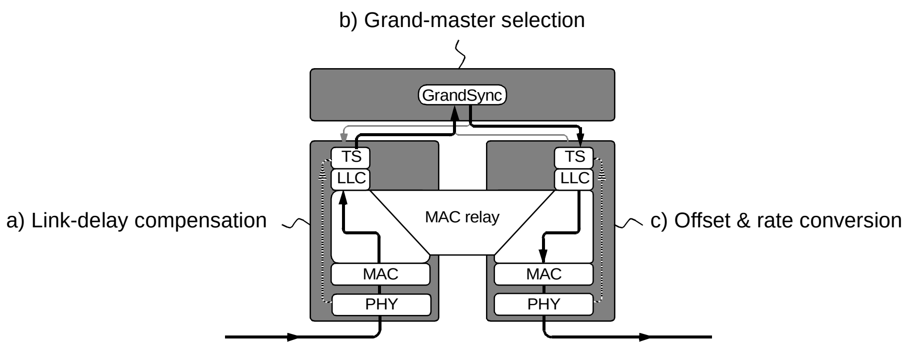
b) Grand-master selection
GrandSync
TS
LLC
TS
LLC
MAC relay
a) Link-delay compensation
c) Offset & rate conversion
MAC
MAC
PHY
PHY

Figure 5.10—Intermediate-bridge responsibilities

5.6 Synchronized-time distribution

5.6.0.1 Clock-master flows

Referring now to the clock-master (Figure 5.4-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 PDUs 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

5.6.0.2 Bus-bridge flows

Referring now to the bus-bridge (Figure 5.4-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 PDUs 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.

5.6.0.3 Clock-slave flows

Referring now to the clock-slave (Figure 5.4-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.

5.6.0.4 Time-stamp flows

Referring now to the hashed PHY-to-TS lines within Figure 5.4 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.

5.7 Cascaded clock topologies

5.7.1 Cascaded clocking limitations

The naive approach towards forwarding time-synchronization information is to quickly propagate time-reference snapshots through successive stations. Unfortunately, relatively small (¼ interval) worst-case residence-time delays in each station can cause significant bunching on relevant topologies, as illustrated in Figure 5.11.

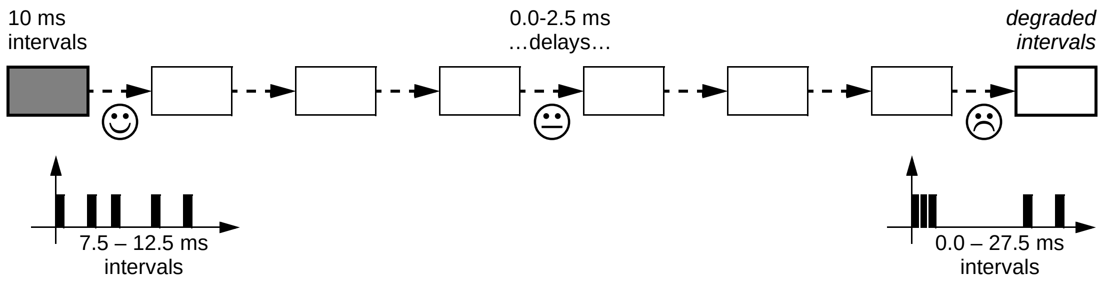

Figure 5.11—Cumulative sync-interval bunching

Techniques for avoiding such bunching are well known and practiced in the form of reclocked synchronous circuits. For example, Ethernet stations accept (baud-rate) information at a closely matched input clock rate, reclock the data with a local clock reference, and then forward the reclocked information without degrading data-jitter performance.

Applying these techniques to clock-sync transmission is straightforward. Rather than quickly forwarding these frames, their information is saved. That saved information is then forwarded in the same periodic fashion, based on local-station timing, as illustrated in Figure 5.12. While such reclocked systems more susceptible to gain-peaking/whiplash effects, their inherent design and verification simplicities favor their use.

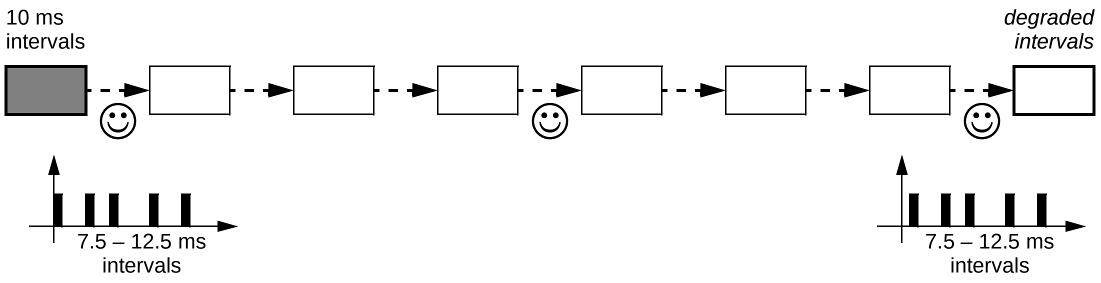

Figure 5.12—Cumulative sync-interval bunching

5.7.2 Mixed sync-interval systems

The reclocked sync-interval strategy is compatible with bridged mixed-media systems. The persistent or transient sync-interval rate of an intermediate (perhaps longer or more power sensitive) link could be less than the rate assumed for the clock-master, as illustrated in the center of Figure 5.13. Similarly, wireless links could base their timing events on triggers initiated by the clock-slave station, as illustrated in the right side of Figure 5.13.

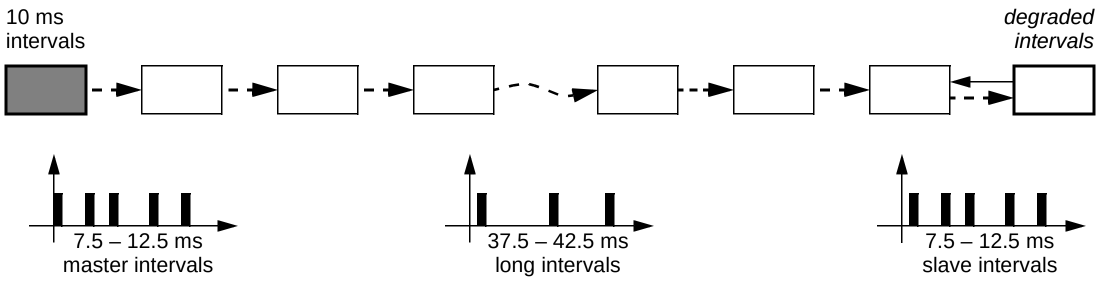

Figure 5.13—Mixed sync-interval systems

Other flow-through clocking designs would require special “boundary clock” architectures to support such mixed systems. With the interval retiming strategy, the additional (specification and implementation) complexities of such boundary-clock architectures are easily avoided.

5.8 Time-affiliation adjustments

5.8.1 Distinct receive/transmit adjustments

Distinct forms of time-delay adjustments occur at the receive (clock-slave) and transmit (clock-master) ports, as illustrated in Figure 5.14. The details of these time-delay adjustments are media-dependent, but the high-level concepts are the same.

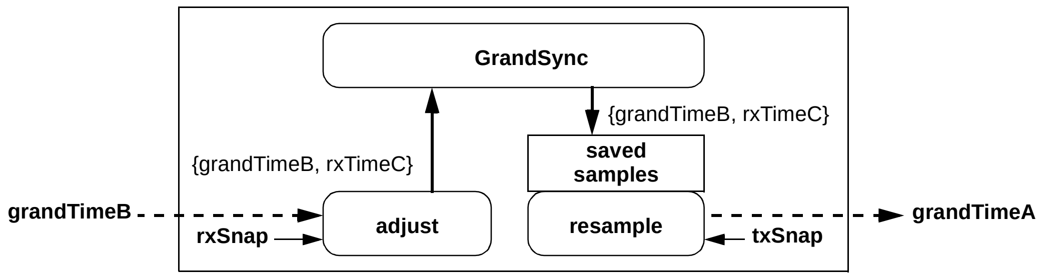

Figure 5.14—Receive/transmit adjustments

When a frame with the clock-master sourced *grandTimeB* is received, a snapshot of the station-local time is taken; that snapshot is called *rxSnap*. A more accurate *rxTimeC* snap-shot value is formed by compensating by the relatively-constant precomputed *linkDelay* value, as follows:

rxTimeC = rxSnap – linkDelay

The delay-compensated {*grandTimeB*, *rxTimeC*} affiliation parameters are passed to the GrandSync entity. That GrandSync entity ignores PDUs from lower-precedence stations, echoing only PDUs from the (perceived to be) grand-master station.

The echoed delay-compensated {*grandTimeB*, *rxTimeC*} affiliation parameters are saved in storage at each of the clock-master ports (this is an architectural model; implementations need not replicate physical storage). The forwarded *grandTimeA* value (which is renamed *grandTimeB* when received at the next station) is derived by interpolating between (or extrapolating from) previously saved samples.

5.9 Sampling offset/rate conversion

Each clock-master port is responsible for using its received {*grandTimeB*, *stationTimeB*} and converting them into the distinct {*grandTimeA*, *stationTimeA*} affiliations that are transmitted to its neighbor. Since the values of *stationTimeB* and *stationTimeA* are (by convention) coupled to the receive and transmit times, this update involves computation of *grandTimeA* values based on observed {*grandTimeA*, *stationTimeB*} values.

5.9.1 Forward interpolation inaccuracies

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 {*grandTime*, *tx*[*m*]} affiliations by extrapolating from a distinct set of receive-snapshot times *rx*[*n*], as illustrated in Figure 5.15.

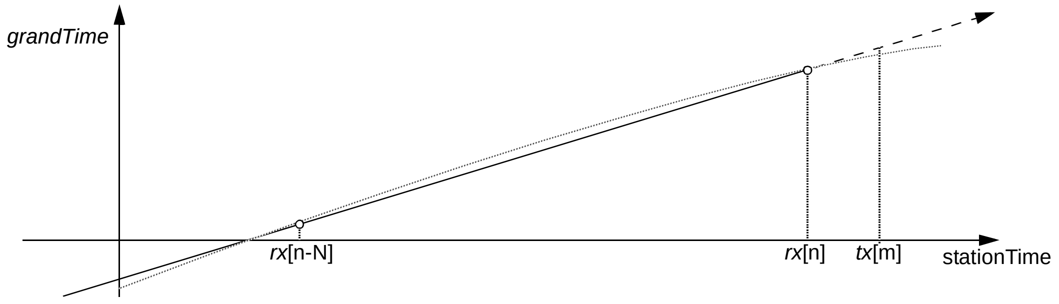

Figure 5.15—Extrapolation for *grandTime*

Extrapolation techniques exhibit gain peaking at frequencies whose wavelength is twice the {*rx*[*n*-N],*rx*[*n*]} slope-averaging interval, because the extrapolated value can exceed what would have been the sampled time value. A cascade of multiple stations emphasizes the gain-peaking inaccuracies, allowing errors to accumulate in an O(N^2) fashion.

5.9.2 Forward interpolation inaccuracies

To reduce gain-peaking effects, the resampling computation can be migrated to a safe-interpolation domain. This involves subtracting a *backTime* constant from *tx*[*m*], yielding a new time *tb*[*m*], for which a less gain-peaking sensitive interpolation is viable, as illustrated in Figure 5.16. In concept, the stale (but {*grandTime*[*m*], *tb*[*m*]} affiliations could be passed to the terminal clock-slave stations, wherein a single extrapolation-to-the-future accumulation could be performed. A preferred technique is to compensate the interpolation result on an per-station basis as the time-reference flows towards the clock-slave station, as discussed in the following subclauses.

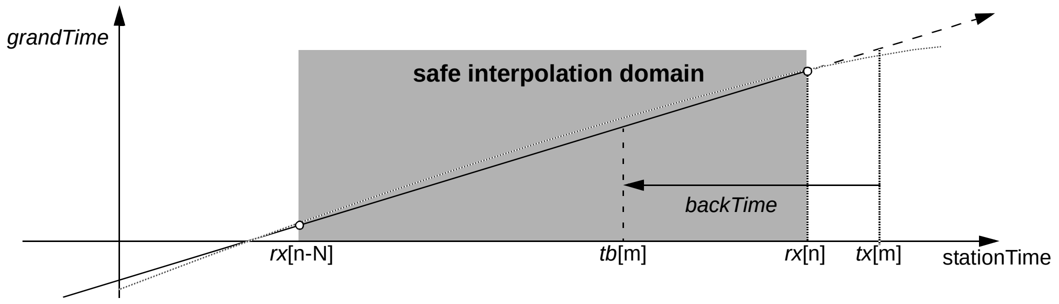

Figure 5.16—Extrapolation for *grandTime*

Extrapolation techniques exhibit gain peaking at frequencies whose wavelength is twice the {*rx*[*n*-N],*rx*[*n*]} slope-averaging interval, because the extrapolated value can exceed what would have been the sampled time value. A cascade of multiple stations emphasizes the gain-peaking inaccuracies, allowing errors to accumulate in an O(N^2) fashion.

5.9.3 Backward interpolation

5.9.3.1 Interpolation of *grandTime*

A more-scalable backward-interpolation approach also views the received {*grandTimeB*, *stationTimeB*} affiliations as points on a curve, sampled at received-snapshot times *rx*[*n*]. However, the objective is to generate the distinct set of {*grandTimeA*, *tx*[*m*]} affiliations by interpolating within a distinct set of receive-snapshot affiliations {*grandTimeB*[*n*], *rx*[*n*]), as illustrated in Figure 5.17.

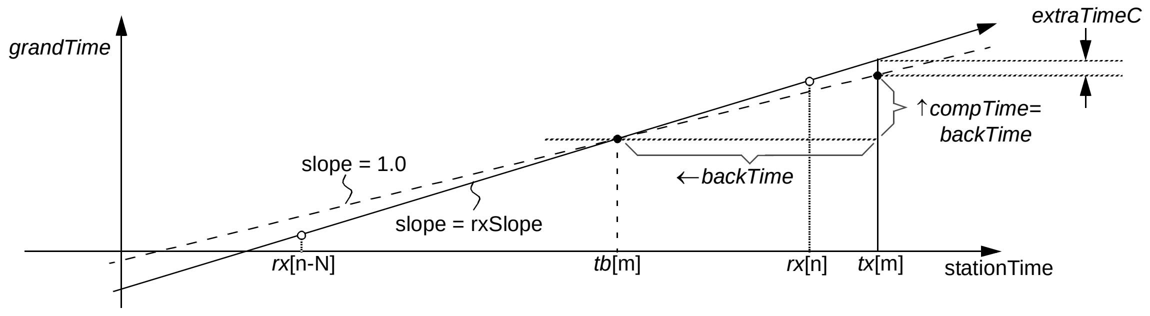

Figure 5.17—Interpolation for *grandTimeA*

grandTimeA[*m*] = *grandTimeB*[*m*] + *rxSlope* * ((*tx*[*m*] – *backTime*) – *rx*[*n*]) + *backTime*; (5.5)
grandTime0[*m*] is the value for the to-be-transmitted {*grandTime0*[*m*], *tx*[*m*]} affiliation.
backTime is a constant (sync-interval dependent) value.
rxSlope is the value of slope of previously sampled values, specified by Equation 5.3.

rxSlope = (*grandTimeB*[*n*] – *grandTimeB*[*n*-N]) / (*rx*[*n*] – *rx*[*n*-N]) (5.6)
Where:
grandTimeB[*n*] is the value from the previously received {*grandTimeB*[*n*], *rx*[*n*]} affiliation.

extraTimeC[*m*] = (*rxSlope* – ONE) * *backTime*; (5.7)

The advantage of this technique is the separation of *grandTime*[*m*] and *extra*[*m*] components. The interpolation process eliminates gain-peaking for the *grandTime*[*m*] value, thus reducing error effects when passing through multiple bridges. The sideband *extraTime* signal remains significant, and is therefore carried through bridges, so that the cumulative *grandTimed*[*m*]+*extraTime*[*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 *extraTime* communication channel, over which the difference between nominal and rate-normalized *backTime* values can be transmitted.

5.9.3.2 Averaging of *extraTime*

An averaging (rather than backward-interpolation) approach is applied to the received {*extraTimeB*, *stationTime*} affiliations as points on a curve, sampled at received-snapshot times *rx*[*n*]. The {*extraTimeD*, *tx*[*m*]} affiliations are produced by averaging recently observed *extraTimeB* values, as illustrated in Figure 5.18.

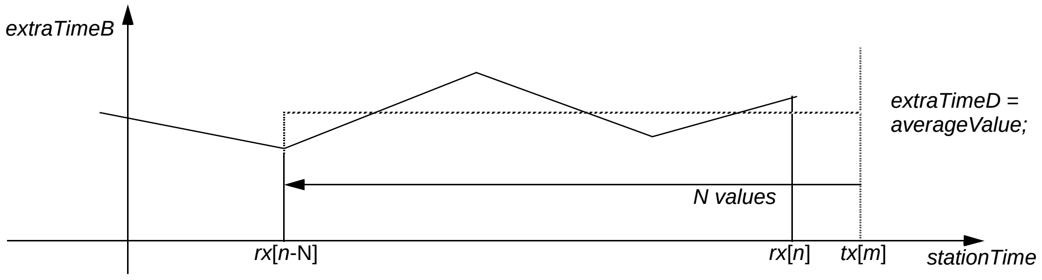

Figure 5.18—Interpolation of *extraTimeD*

$$extraTimeD[m] = (extraTime[n\text{–}N] + \ldots\ extraTime[n]) / N \quad (5.8)$$
$$extraTimeA[m] = extraTimeC[m] + extraTimeD[m]; \quad (5.9)$$

The to-be-transmitted value of *extraTimeA*[*m*] consists of a contribution *errorTimeC* (coming from this station’s *grandTime* interpolation) and a contribution *extraTimeD* (accumulated from previous stations’s *grandTime* interpolations). Note that the averaging of *extraB* values is effectively a low-pass filtering process that removes noise without causing a gain-peaking frequency response.

NOTE—For simplicity and scalability, the computed *extraTimeC* time is based on *n*, a fixed number of samples, where *n* is a convenient power-of-two in size.

5.10 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.