An event-based AVB synchronization architecture

802.1AS Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks

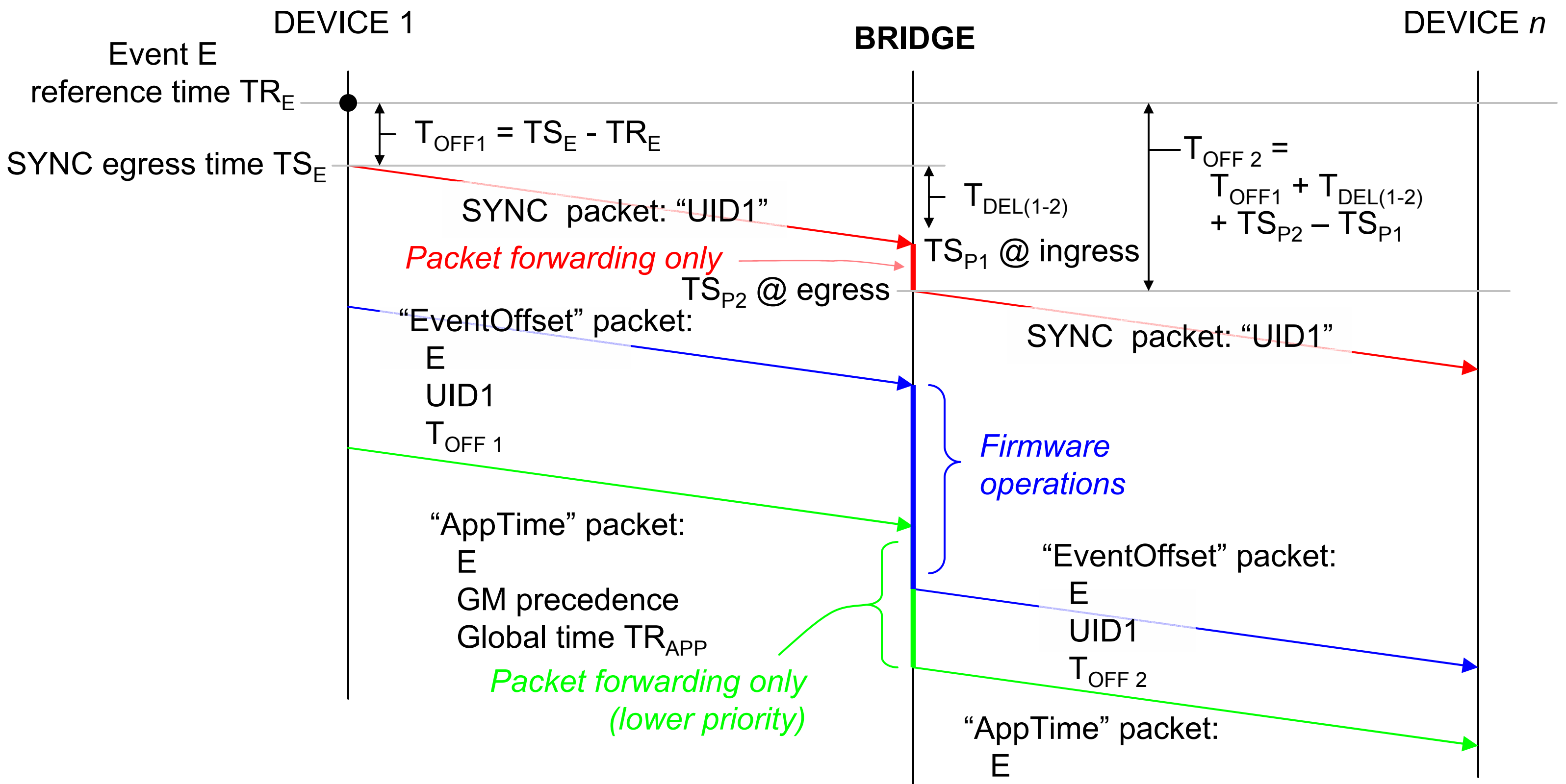
Chuck Harrison

8 February 2007

Overview

- The proposed architecture separates a low-level, epoch-independent, **event-based synchronization service** as a sublayer below TAI time distribution
- The event-based synchronization service does *not* involve a Grandmaster clock but *does* rely on
  1. devices' ability to measure short durations of time
  2. the election of a "sync master" to issue periodic (but not particularly accurate) broadcast SYNC packets
- The proposal uses media-independent 802.1 interfaces to accommodate 802.3 and 802.11 PHYs without requiring BCs
- This proposal includes a P2P frequency and delay correction mechanism at the event sublayer, but the AVB/802.1AS performance goals can usually be met without using them
- A 1588-equivalent Ordinary Clock service is optionally co-located at any bridge and is synchronized by upper-layer messages in combination with the event-based service
- This architecture permits continued operation (albeit with degraded performance) in a mixed network of both AVB and non-AVB bridges

Master Clock distribution over event-based synchronization service

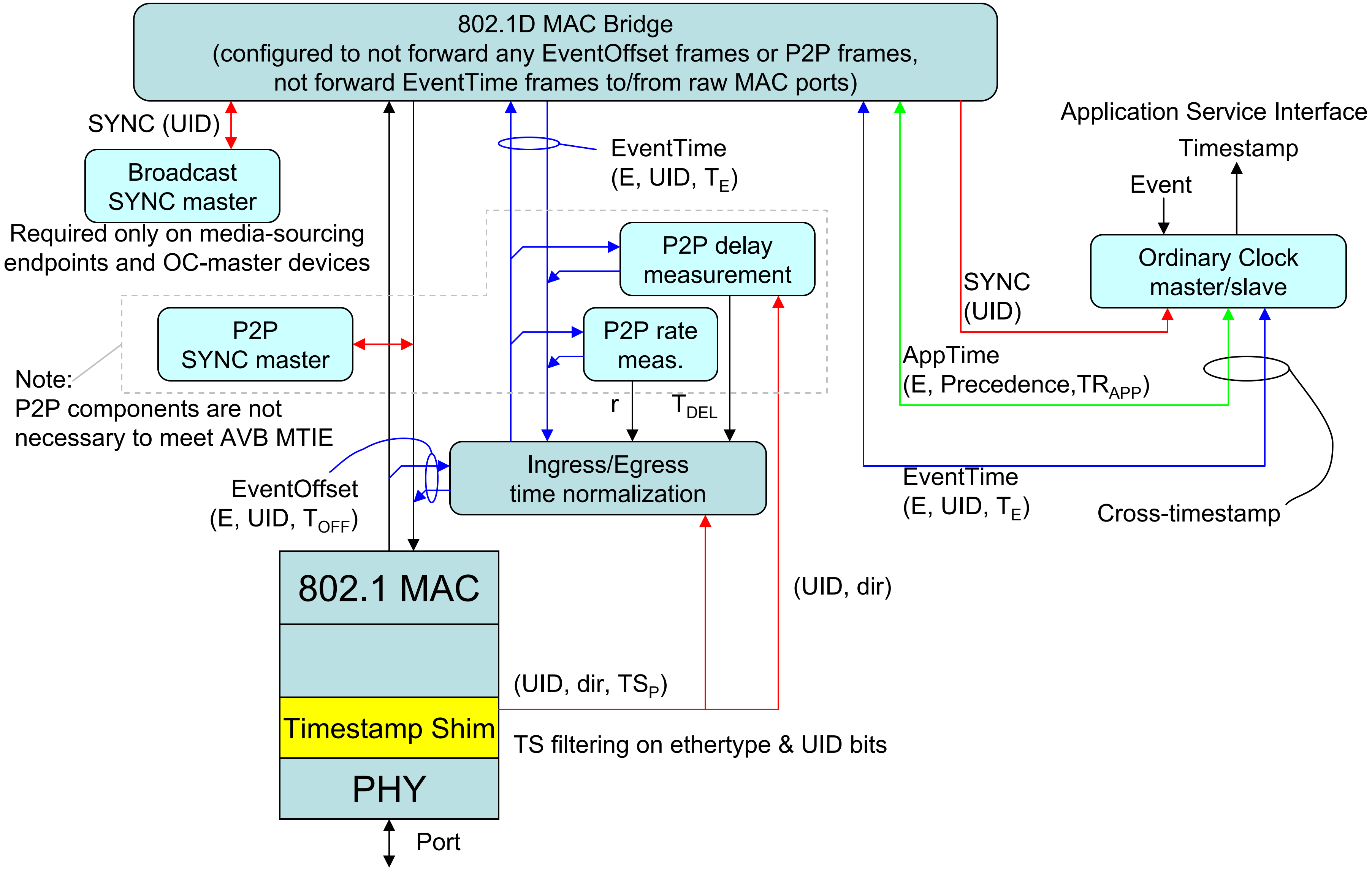


1588 Grandmaster election similar to as-dvj-perspective

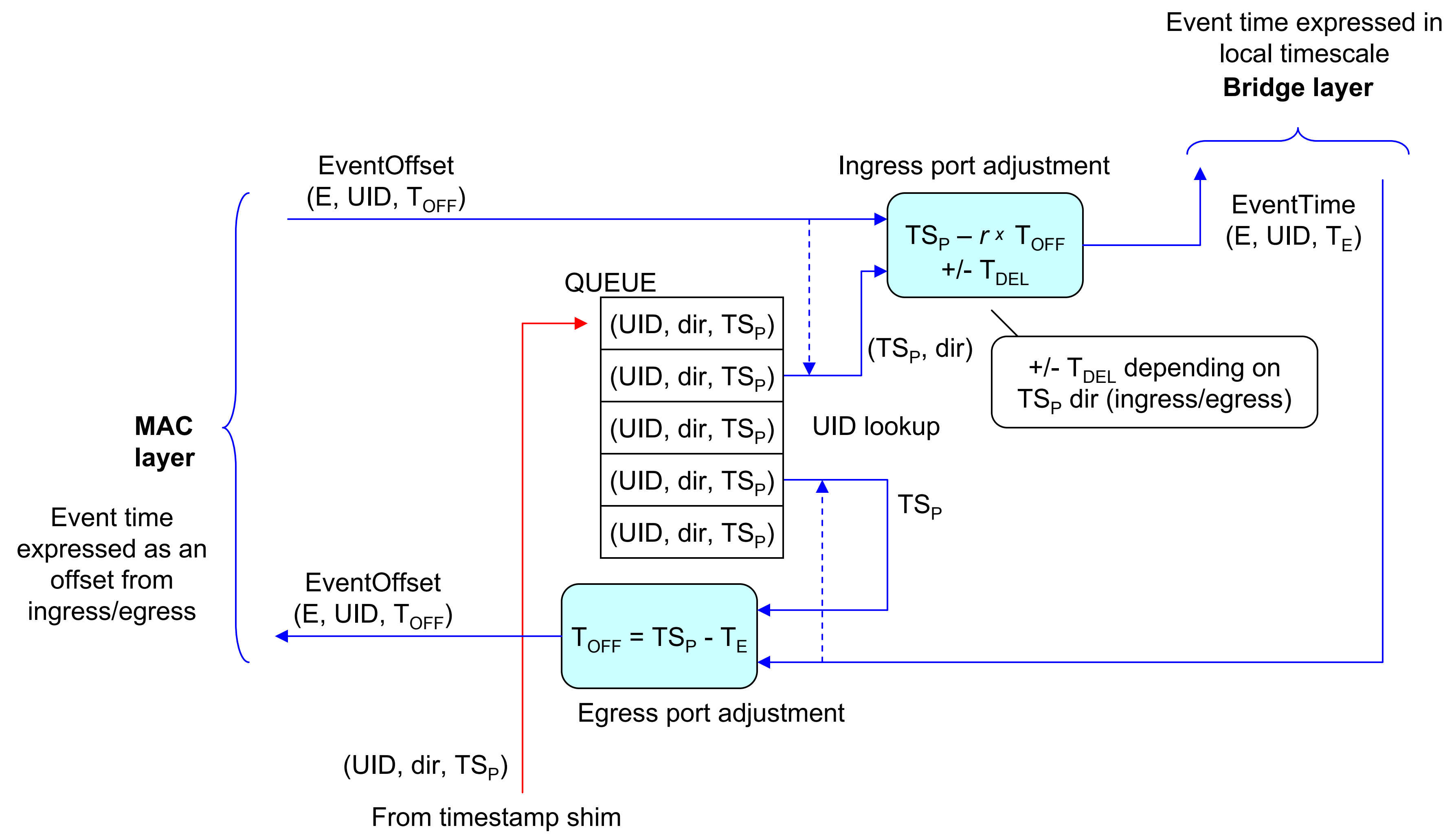
SYNC packet is *not* generated from Grandmaster timescale

Cross-timestamp delivered to OC:
Local time T_E
Global time TR_{APP}

802.1AS Device stack


802.1D MAC Bridge
(configured to not forward any EventOffset frames or P2P frames,
not forward EventTime frames to/from raw MAC ports)
SYNC (UID)
Broadcast SYNC master
Required only on media-sourcing endpoints and OC-master devices
EventTime
(E, UID, T_E)
Application Service Interface
Timestamp
Event
Ordinary Clock master/slave
SYNC (UID)
P2P delay measurement
P2P rate meas.
P2P SYNC master
Note:
P2P components are not necessary to meet AVB MTIE
AppTime
(E, Precedence,TR_{APP})
r
T_{DEL}
Ingress/Egress time normalization
EventOffset
(E, UID, T_{OFF})
EventTime
(E, UID, T_E)
Cross-timestamp
802.1 MAC
(UID, dir)
(UID, dir, TS_P)
Timestamp Shim
TS filtering on ethertype & UID bits
PHY
Port

Ingress/Egress time normalization


Event time expressed in local timescale
Bridge layer
EventOffset (E, UID, T_{OFF})
Ingress port adjustment
$TS_P - r \times T_{OFF}$ +/- T_{DEL}
EventTime (E, UID, T_E)
QUEUE
(UID, dir, TS_P)
(UID, dir, TS_P)
(UID, dir, TS_P)
(UID, dir, TS_P)
(UID, dir, TS_P)
(TS_P, dir)
+/- T_{DEL} depending on TS_P dir (ingress/egress)
UID lookup
MAC layer
Event time expressed as an offset from ingress/egress
TS_P
EventOffset (E, UID, T_{OFF})
$T_{OFF} = TS_P - T_E$
Egress port adjustment
(UID, dir, TS_P)
From timestamp shim

Ordinary Clocks

- Ordinary Clocks are maintained in a 1588-inspired manner.
  1. The Grandmaster issues a broadcast EventTime message, with each AVB bridge acting much like 1588 Transparent Clock.
  2. The Grandmaster also issues corresponding broadcast AppTime messages referencing the event in the EventTime message.
  3. This results in a “cross-timestamp” delivered to each slave OC correlating the device’s local timescale to the Grandmaster’s timescale. (The cross-TS point may be in the recent past.)
  4. Based on this cross-timestamp the OC is aligned to the Grandmaster (mechanism is implementation-specific)
- An OC “talks” *only* when it is Grandmaster
- Grandmaster election is approximately as described in as-dvj-perspective-070124.pdf

Sync Master behavior

- Basic algorithm:
  - If you hear another SYNC master, be quiet
  - If you hear nothing, wait a random interval and start transmitting SYNC
- Broadcast SYNC master (~10ms)

  sends broadcast SYNC packets to all ports of bridge
- P2P SYNC master (~100ms)

  sends link-local SYNC messages to attached port

P2P rate & delay protocol

1. **Rate correction** (roughly equivalent to syntonization)

- Each partner issues a link-local SYNC packet, along with a link-local EventOffset message (E_{P2P}, UID, T_{OFF}) at a periodic rate (~100msec). The E_{P2P} event is defined as the most recent 1-msec-granularity “tick” of the sender’s timescale. Receiver examines interval between adjacent E_{P2P} events, determines number of 1-msec ‘ticks’ (might not be exactly 100), and computes frequency ratio r between link partners’ timescales.
- The computed frequency ratio is filtered and passed to the ingress time normalization function as a correction factor for the incoming T_{OFF} field in EventOffset messages arriving on this port.

2. **Delay correction**

- Each partner sends a link-local EventOffset (E_{P2P}, UID, T_{OFF}) message, referencing the UID of each incoming link-local SYNC packet. The E_{P2P} event is defined as above. Each partner now has two ways of computing the phase offset between 1-ms ‘ticks’ of his own clock and his partner’s: one based on the outgoing SYNC packet, as part of the rate correction scheme, and the other based on the incoming SYNC packet just described. While the phase offset itself is arbitrary, the difference between these two measurements of it is the residual uncorrected round-trip delay on the link.
- The residual round-trip delay is used to correct a filtered estimate of the actual delay. The estimated delay is divided by two and passed to the ingress time normalization function as a T_{DEL} correction to EventOffset messages arriving on this port.

A lot like 1588 with TCs, but…

- SYNC packets contain *only* an opaque UID and need *no* upper-layer processing or MACsec encryption
  - Capture timestamp *and* UID at MAC shim
  - Forward packet as standard 802.1D bridge
- SYNC packet is *not* queued waiting for the followup message; therefore standard 802.1D bridge logic is used and residence times are so short that syntonization is usually unnecessary in bridges
- SYNC master (loose ~10msec period) operates within the event-based sublayer and can be elected with a simple mechanism (no precedence issues)
- Grandmaster must wait until a SYNC issued by the sublayer comes by, then timestamp *that*.
- SYNC packets may pass either the same or opposite direction to other synchronization-related messages on a link
- Identical messages used for time propagation and for link delay measurement

What about 802.11?

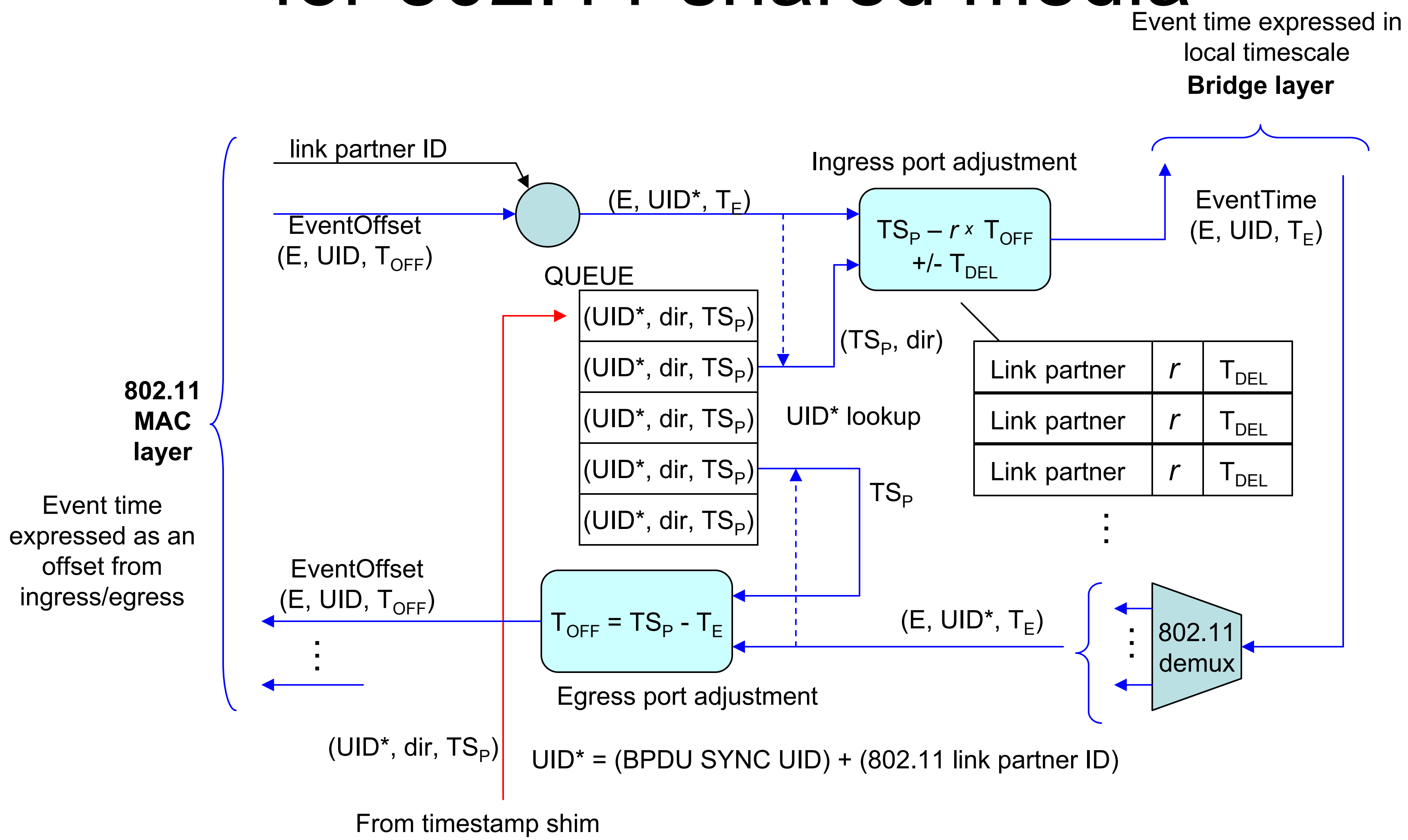
- Where SYNC propagates across a link, this protocol relies on tight correlation between egress timestamp at sender and ingress timestamp at receiver
- If 802.11 broadcast service is used, many STAs will miss SYNC packets, and will lose Events referencing those packets. This is not a disaster: the AppTime protocol for OCs is robust to moderate packet loss.
- Alternatively, if 802.11 P2P service is used (with ACK / retransmission), different STAs will experience different “residence times” at the same bridge port for the same SYNC UID.
- This can be accommodated by translating each broadcast SYNC BridgePDU and each EventTime BPDU into multiple unicast MacPDUs, modifying the lower layer so that SYNC UIDs are effectively extended by link-partner-ID bits (which are ultimately stripped at the receiving STA before forming BPDUs). SYNC UID timestamps propagate to ingress/egress normalization queue only after packet ACK.

Additional material

Incompletely developed ideas

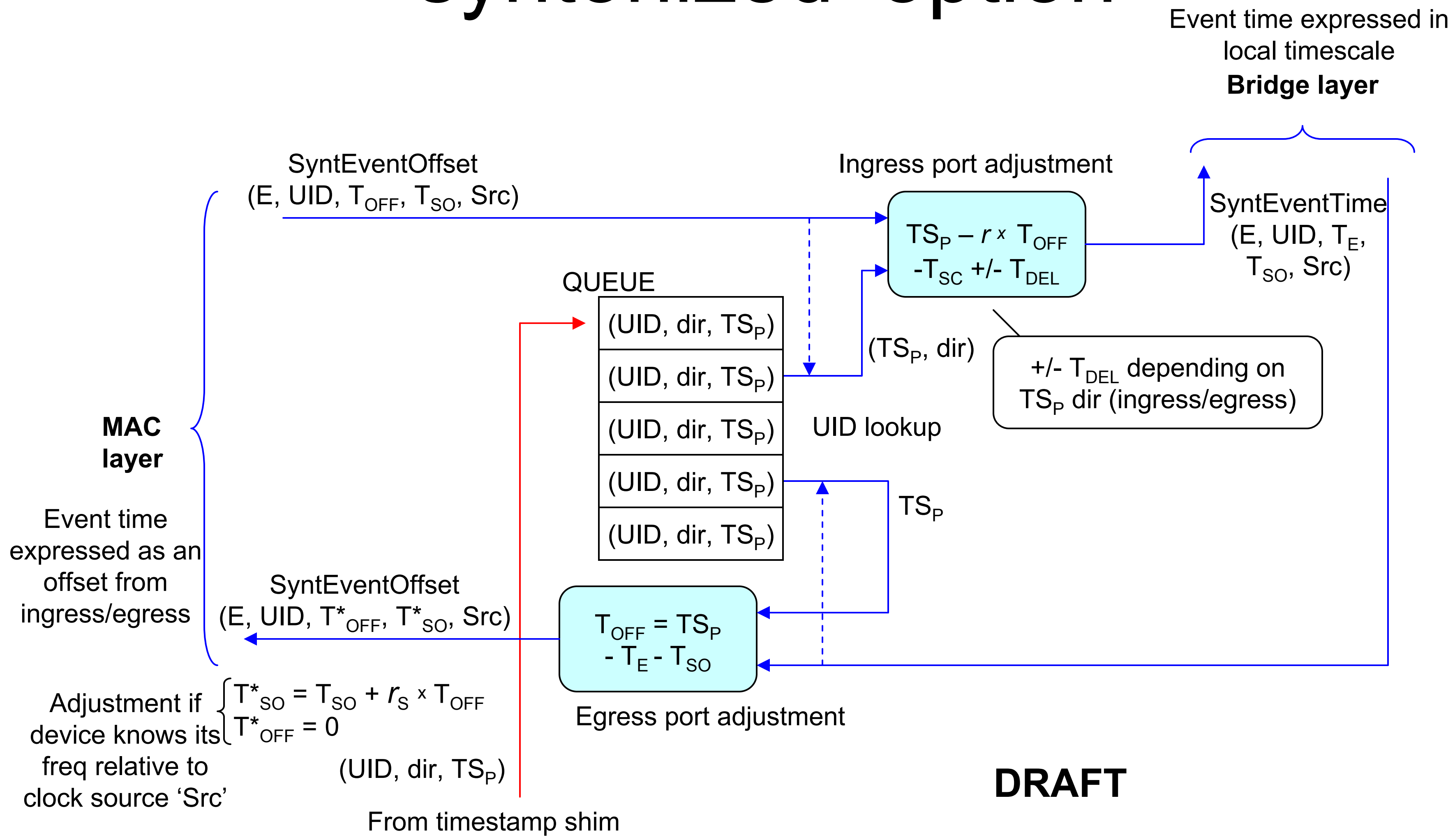
- Potential 802.11 implementation, see slide
- Syntonized offset: assume some part of T_{OFF} is known "precisely" (ref to a Grandmaster, say) and the rest is accumulated through non-syntonized bridges. By identifying the "syntonized" component of EventOffset, residence time error accumulation can be minimized.
- Independent MAC timebases: MACs for different media (or on different blades) may operate from different crystals. If the "timestamp shim" sublayer accepts a hardware event input, these independent MAC timescales may be correlated and made invisible to higher sublayers.
- Link Aggregation: this protocol should work with 802.3 clause 43 link aggregation. The timestamp shim report should include a link identifier so that distinct T_{DEL} values can be maintained.
- Broadcast, multicast, unicast, link-local, see slide
- MACsec: encryption/authentication on everything except SYNC packets is transparent. In principle this works on SYNC packets, too, but it requires two portholes into the stack (below and above MACsec), so it may in practice be more complex. With common MACsec authentication suites, underlying packet is unmodified so we could identify SYNC ethertype and UID inside the MACsec wrapper at a low-layer porthole.

Ingress/Egress time normalization for 802.11 shared media



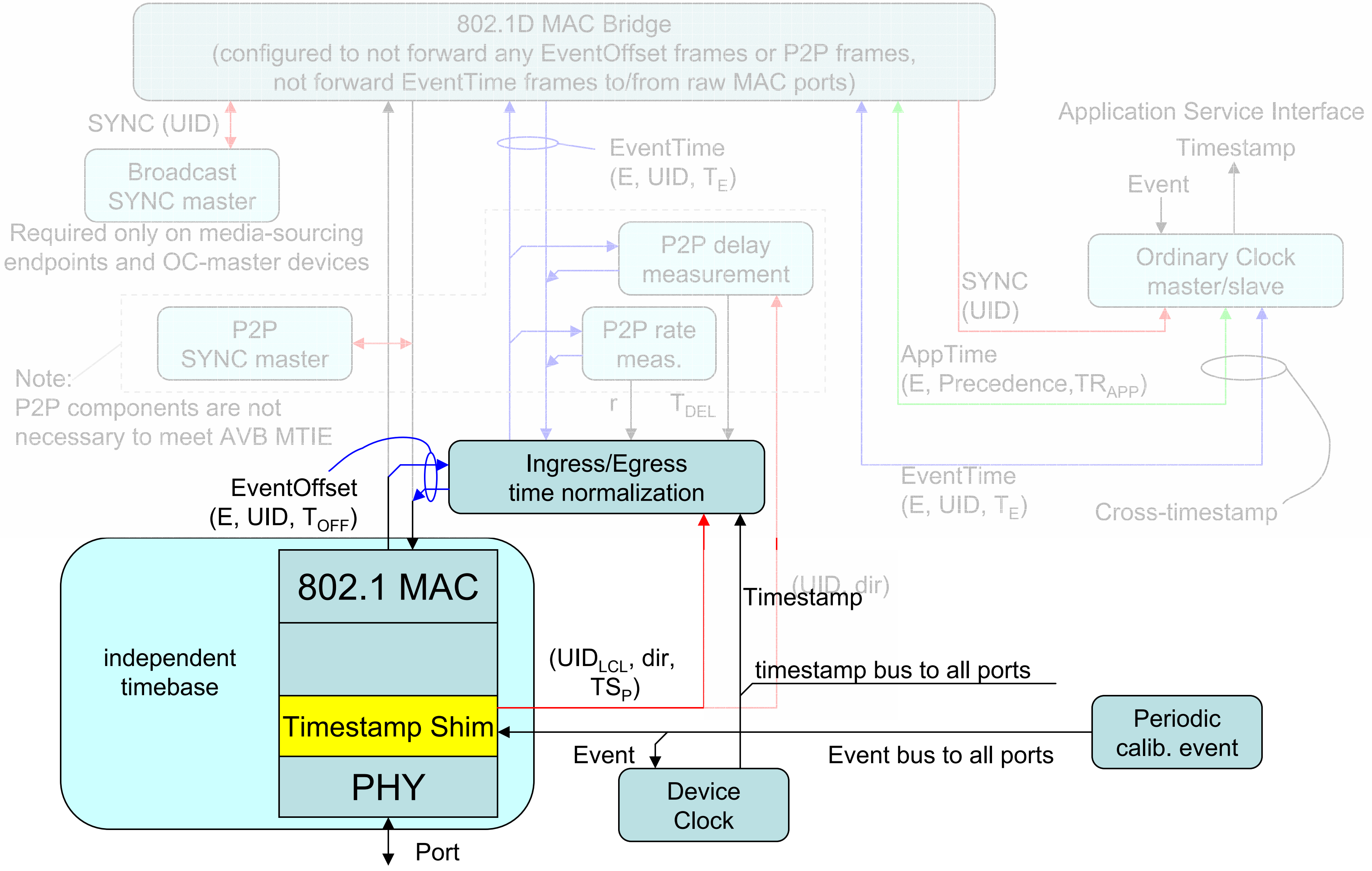
UID* = (BPDU SYNC UID) + (802.11 link partner ID)

Ingress/Egress time normalization 'syntonized' option



DRAFT

Independent MAC timebases


802.1D MAC Bridge
(configured to not forward any EventOffset frames or P2P frames,
not forward EventTime frames to/from raw MAC ports)
SYNC (UID)
Broadcast SYNC master
Required only on media-sourcing endpoints and OC-master devices
EventTime (E, UID, T_E)
Application Service Interface
Timestamp
Event
P2P delay measurement
P2P SYNC master
P2P rate meas.
SYNC (UID)
Ordinary Clock master/slave
Note: P2P components are not necessary to meet AVB MTIE
AppTime (E, Precedence, TR_APP)
r
T_DEL
Ingress/Egress time normalization
EventOffset (E, UID, T_OFF)
EventTime (E, UID, T_E)
Cross-timestamp
802.1 MAC
independent timebase
Timestamp
(UID, dir)
(UID_LCL, dir, TS_P)
timestamp bus to all ports
Timestamp Shim
Periodic calib. event
Event
Event bus to all ports
PHY
Device Clock
Port

Broadcast, multicast, unicast, etc

- This architecture is based primarily on a 802.1D MAC bridge model, so many 802.1 addressing options should work (in addition to broadcast) for Event and AppTime messages. Group multicast and VLAN should work.
- The basic protocol (without P2P rate & delay measurement) will propagate through non-AVB bridges with reduced accuracy. If P2P packets are sent on a link-local MAC address, as proposed, the protocol cannot traverse non-AVB bridges. Alternatively, P2P SYNC and EventOffset may be sent with ordinary MAC addresses, in which case they will traverse non-AVB bridges and the intervening bridges will be modeled as link delay. This provides graceful degradation in mixed networks.