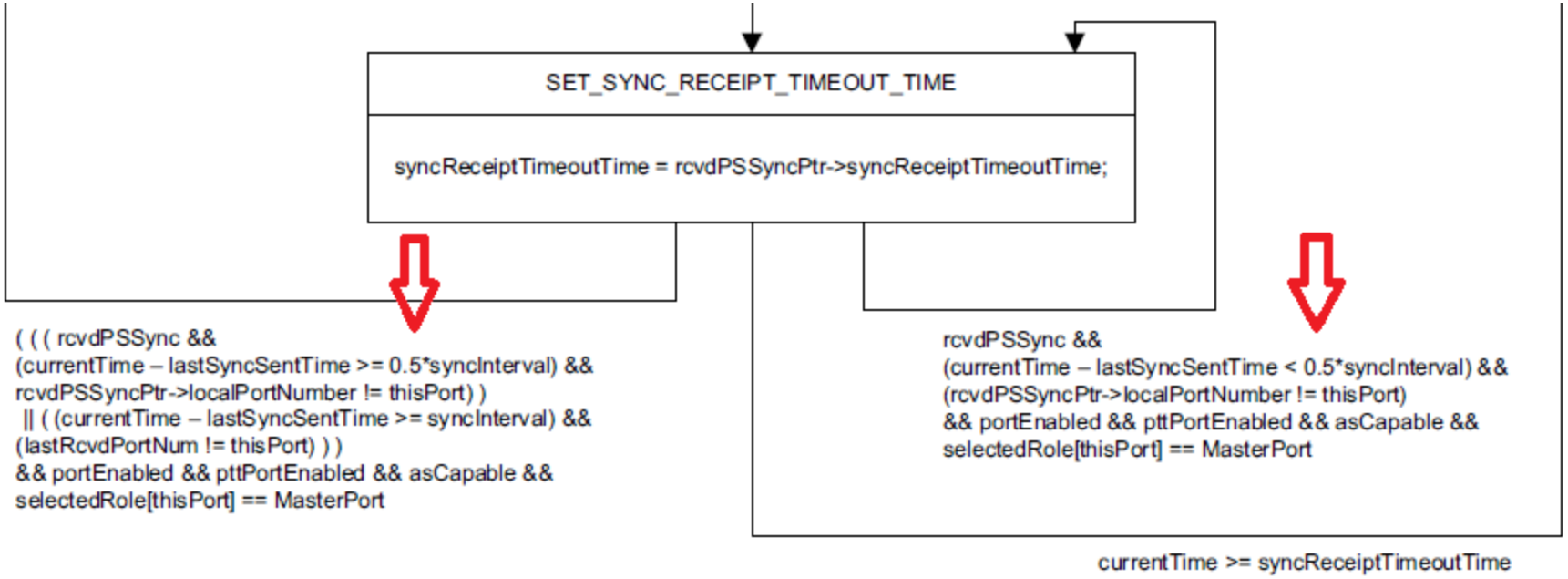
802.1ASbt presentation on Short Sync Interval issue for Clause 10 (Media Independent layer)

Bob Noseworthy (ren@iol.unh.edu)
University of New Hampshire's
InterOperability Laboratory (UNH-IOL)
IEEE 802.1 TSN TG
2014.05.15

Introduction

- The actual Sync Interval may at times be 50% of the desired interval.
  - Due to variances in the timing of Sync messages into the SiteSyncSync SM vs the local device port's own syncInterval, a port may "frequently" send Sync Intervals at 0.5 the intended rate
    - Nominal rate is 125ms, resulting in a 62.5ms interval on occasion (enough to broaden std.dev substantially)
    - Another artifact of the underlying issue is the resulting delay in updated GM time arriving at a slave, as a bridge is likely to delay such updates by more than 125ms (rather than just the residence time delay)

The PortSyncSyncSend SM (Fig 10-8 per 802.1AS-cor)



- Focus is on the left branch:
  - "currentTime- lastSyncSentTime >= 0.5*syncInterval"

vs

  - the "currentTime – lastSyncSentTime >= syncInterval".

The PortSyncSyncSend SM (Fig 10-8 per 802.1AS-cor)

```
( ( ( rcvdPSSync &&
(currentTime – lastSyncSentTime >= 0.5*syncInterval) &&
rcvdPSSyncPtr->localPortNumber != thisPort) )
 || ( (currentTime – lastSyncSentTime >= syncInterval) &&
(lastRcvdPortNum != thisPort) ) )
&& portEnabled && pttPortEnabled && asCapable &&
selectedRole[thisPort] == MasterPort
```

- In the first expression (0.5*syncInterval) would eval to TRUE if a rcvdPSSync came in before the local stations syncInterval timer expired.
  - This is true either:
    - if the device is a bridge that is not GM, thus receiving Sync's from upstream at a different rate than the local syncInterval timer
    - If the device is generating sync messages internally but via a process that is not synchronized with the PortSyncSync SM's syncInterval timer

Issue impact 1 – unintended interval rate increase

- This behavior leads to Sync messages that are sent from a device that often have single gaps of 62.5ms among a majority of 125ms gaps.
  - This increases the std.dev of observed intervals considerably
  - In a pathological case, for a non-GM bridge, the timing of internal syncInterval timers vs upstream syncIntervals may result in slaves connected to the bridge to receive Sync messages at a far faster rate than the slaves are designed to accommodate.

Issue impact 2 – stale sync timestamps

- For a non-GM Bridge, at the core of the issue is the linkage between the local syncInterval timer duration and the upstream device's syncinterval.
  - As written today, there is no allowance for syncInterval variance (presumed to be 125ms nominally, but +/-?)
  - Consider a bridge that has its syncInterval expire moments before receiving a sync message (rcvdPSSync) from an upstream source
    - This bridge will send its own Sync message with the origin timestamp from the Sync received 125ms earlier, plus an updated correctionField
      - Aside: moments later (after SEND_MD_SYNC) a rcvdPSSync from upstream will cause the next Sync to be sent at 0.5 interval)
    - This will delay tracking of GM clock variances (though admittedly, no significant variance is expected in such a short duration (single sync interval), unless continuously delayed)

Possible solution (slide 1)

- A change where the state machine would favor waiting for incoming Sync messages (rcvdPSSync) before generating its own Sync message could be favored.
  - This approach necessitates a bounding of the allowable range for the syncInterval (if nominally 125ms, it will vary by +/- ??)
  - Currently AS does not define an allowed range. IEEE 1588v2 specifies a +/-30% range on such an interval.
    - This may be considered too broad, or perhaps allow messages to be sent 'too fast'.

Possible solution (slide 2)

- If +/-30% were used, then the expression could be modified to:

- (((rcvdPSSync &&
  (currentTime – lastSyncSentTime >= 0.7*syncInterval) &&
  rcvdPSSyncPtr->localPortNumber != thisPort))
  || ((currentTime – lastSyncSentTime >= 1.3*syncInterval) &&
  (lastRcvdPortNum != thisPort)))
  && portEnabled && pttPortEnabled && asCapable &&
  selectedRole[thisPort] == MasterPort

- With a similar change on the looping transition into the SET_SYNC_RECEIPT_TIMEOUT_TIME state (change the 0.5* there to 0.7*)

- This is the Alternative Remedy proposed in related .1 maintenance

Summary

- Nominal Sync intervals may be violated per standard's conformant state machine behavior
  - This results in violation of the anticipated mean and assumed std.dev (assumed given pt. below)
- 802.1AS currently does not give guidance on interval tolerances
  - IEEE 1588v2 specifies +/- 30%
- A possible remedy changing state machine behavior in the PortSyncSyncSend SM was presented, but a range must be agreed on.