# Summary of IEEE Std 1588™ – 2008 Optional Features Related to Redundancy and Potentially Improved Performance

**Geoffrey M. Garner**
**Consultant**

*IEEE 802.1 TSN TG*
2013.05.14

gmgarner@alum.mit.edu

# Acknowledgement

- The author would like acknowledge Rune Haugom [1] for having pointed out this issue and providing the example described in this presentation, and also providing the figure used in slides 6 and 7

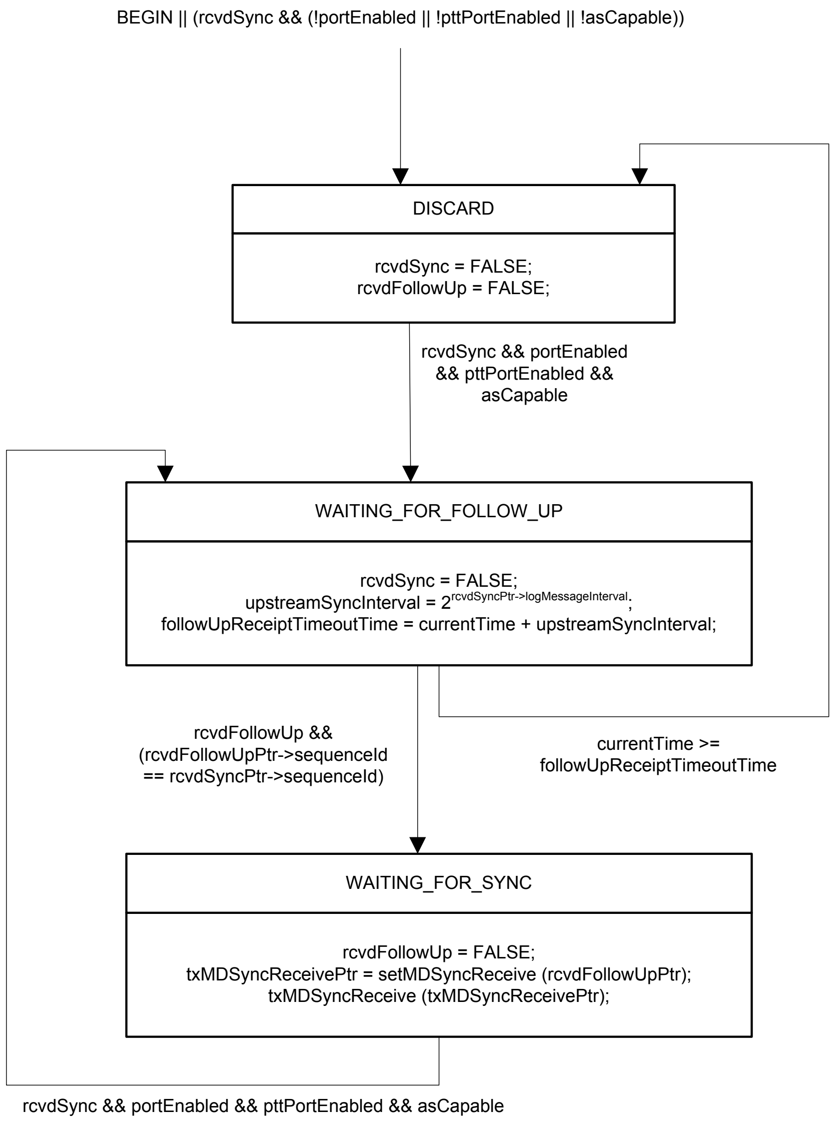
# Introduction - 1

- This presentation describes a scenario (first described in [1]) in which loss of a single Follow_Up message can lead to sync receipt timeout
  - The scenario occurs when a Follow_Up message is lost after a Sync message that has arrived slightly late, the next Sync message is slightly early, and the Sync message after that is slightly late
    - By late and early, we mean relative to the nominal times implied by the specified mean Sync interval
    - The behavior results from the behavior of the MDSyncReceiveSM state machine (Figure 11-6 of 802.1AS)
- It was asked in [1] whether the behavior was intended in 802.1AS
- The scenario was discussed in the June 19, 2013 TSN call

# Introduction - 2

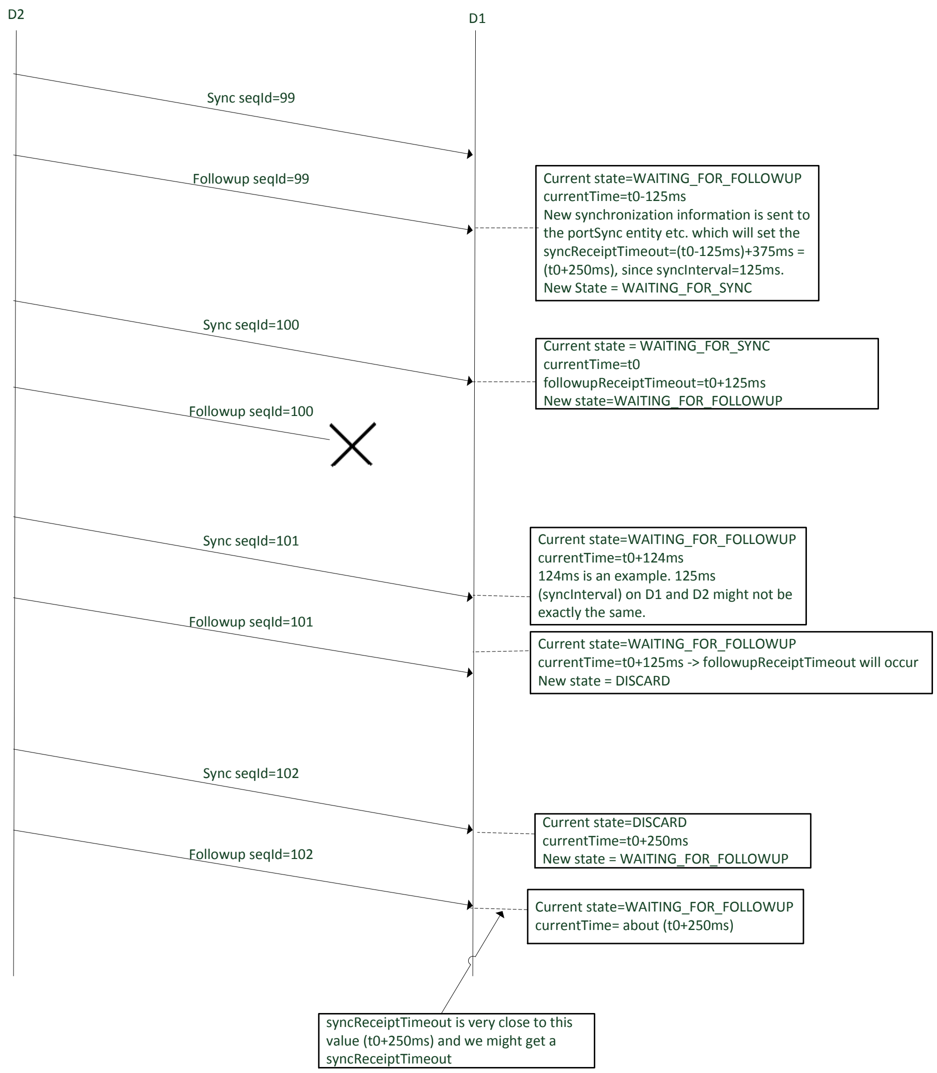
- It was the opinion in the call that the behavior was not intended, and that a fix is needed in 802.1AS to prevent it
  - An initial suggestion was made in the call for a simple fix to the MDSyncReceiveSM state machine
- It was decided in the call that the item should be entered in the 802.1 maintenance database
  - A maintenance request on this was submitted by the editor
- It was also decided in the call that the issue would be further discussed in the July, 2013 TSN meeting
  - The present contribution was prepared for this

# MDSyncReceiveSM State Machine


BEGIN || (rcvdSync && (!portEnabled || !pttPortEnabled || !asCapable))
DISCARD
rcvdSync = FALSE;
rcvdFollowUp = FALSE;
rcvdSync && portEnabled && pttPortEnabled && asCapable
WAITING_FOR_FOLLOW_UP
rcvdSync = FALSE;
upstreamSyncInterval = $2^{rcvdSyncPtr\text{->}logMessageInterval}$;
followUpReceiptTimeoutTime = currentTime + upstreamSyncInterval;
rcvdFollowUp && (rcvdFollowUpPtr->sequenceId == rcvdSyncPtr->sequenceId)
currentTime >= followUpReceiptTimeoutTime
WAITING_FOR_SYNC
rcvdFollowUp = FALSE;
txMDSyncReceivePtr = setMDSyncReceive (rcvdFollowUpPtr);
txMDSyncReceive (txMDSyncReceivePtr);
rcvdSync && portEnabled && pttPortEnabled && asCapable

## Reproduced from Figure 11-6/802.1AS

# Scenario leading to sync receipt timeout - 1



Current state=WAITING_FOR_FOLLOWUP
currentTime=t0-125ms
New synchronization information is sent to the portSync entity etc. which will set the syncReceiptTimeout=(t0-125ms)+375ms = (t0+250ms), since syncInterval=125ms.
New State = WAITING_FOR_SYNC

Current state = WAITING_FOR_SYNC
currentTime=t0
followupReceiptTimeout=t0+125ms
New state=WAITING_FOR_FOLLOWUP

Current state=WAITING_FOR_FOLLOWUP
currentTime=t0+124ms
124ms is an example. 125ms (syncInterval) on D1 and D2 might not be exactly the same.

Current state=WAITING_FOR_FOLLOWUP
currentTime=t0+125ms -> followupReceiptTimeout will occur
New state = DISCARD

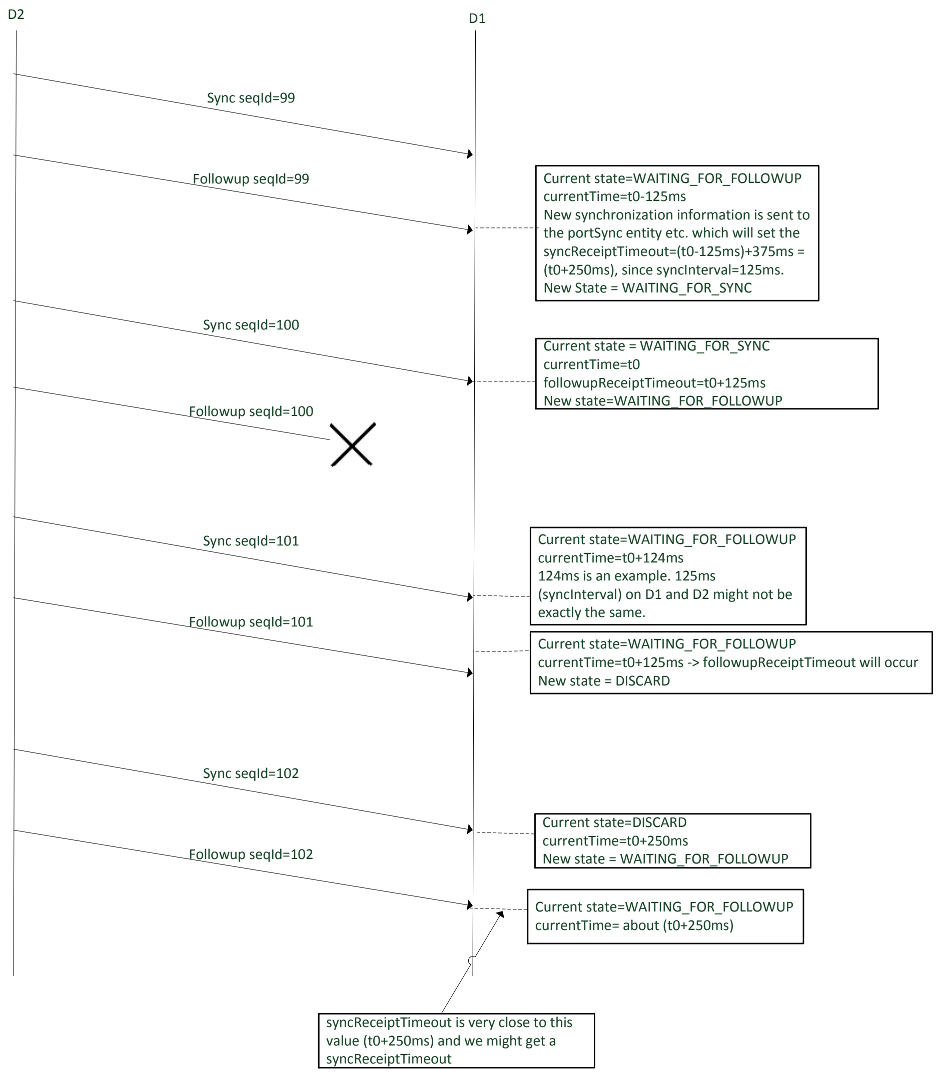
Current state=DISCARD
currentTime=t0+250ms
New state = WAITING_FOR_FOLLOWUP

Current state=WAITING_FOR_FOLLOWUP
currentTime= about (t0+250ms)

syncReceiptTimeout is very close to this value (t0+250ms) and we might get a syncReceiptTimeout

- ❑ Sync interval = 125 ms
- ❑ Sync receipt timeout = 3 sync intervals
- ❑ Sync message 99 arrives; go to state WAITING_FOR_FOLLOWUP
- ❑ Follow_Up message 99 arrives at time t0-125 ms; go to state WAITING_FOR_SYNC
  - Information is sent to PortSync entity, and PortSyncSyncReceiveSM sets syncReceiptTimeoutTime to currentTime+375 ms = t0+250 ms
- ❑ Sync message 100 arrives at time t0 (slightly late since it is more than 125 ms later than previous Sync); go to state WAITING_FOR_FOLLOWUP
  - followUpReceiptTimeoutTime set to t0+125 ms by MDSyncReceiveSM
- ❑ Follow_Up message 100 is lost
- ❑ Sync message 101 arrives at time t0+124 ms, i.e., slightly early
  - It is ignored, because MDSyncReceiveSM is still waiting for Follow_Up

# Scenario leading to sync receipt timeout - 2



D2

D1

Sync seqId=99

Followup seqId=99

Current state=WAITING_FOR_FOLLOWUP
currentTime=t0-125ms
New synchronization information is sent to the portSync entity etc. which will set the syncReceiptTimeout=(t0-125ms)+375ms = (t0+250ms), since syncInterval=125ms.
New State = WAITING_FOR_SYNC

Sync seqId=100

Current state = WAITING_FOR_SYNC
currentTime=t0
followupReceiptTimeout=t0+125ms
New state=WAITING_FOR_FOLLOWUP

Followup seqId=100

Sync seqId=101

Current state=WAITING_FOR_FOLLOWUP
currentTime=t0+124ms
124ms is an example. 125ms (syncInterval) on D1 and D2 might not be exactly the same.

Followup seqId=101

Current state=WAITING_FOR_FOLLOWUP
currentTime=t0+125ms -> followupReceiptTimeout will occur
New state = DISCARD

Sync seqId=102

Current state=DISCARD
currentTime=t0+250ms
New state = WAITING_FOR_FOLLOWUP

Followup seqId=102

Current state=WAITING_FOR_FOLLOWUP
currentTime= about (t0+250ms)

syncReceiptTimeout is very close to this value (t0+250ms) and we might get a syncReceiptTimeout

- ❑ At time t0+125 ms, followUpReceiptTimeout occurs; go to state DISCARD
- ❑ Follow_Up message 101 arrives, and is ignored because Sync message 101 was not processed
- ❑ Sync message 102 arrives at time t0+250ms; go to state WAITING_FOR_FOLLOWUP
  - ▪ followUpReceiptTimeoutTime set to t0+375 ms by MDSyncReceiveSM
- ❑ Follow_Up message 102 will arrive between t0+250 ms and t0+375 ms
  - ▪ However, syncReceiptTimeoutTime is set to t0+250 ms
- ❑ Therefore, sync receipt timeout occurs at time time t0+250 ms
  - ▪ Sync receipt timeout has occurred, even though only one Follow_Up message has been lost

# Scenario leading to sync receipt timeout - 3

❑Sync receipt timeout occurred due to the loss of a single Follow_Up message because

a) After initial Follow_Up message was lost, the MDSyncReceiveSM continued to wait for the Follow_Up, for the remainder of the interval until followUpReceiptTimoutTime

b) Since the Follow_Up receipt timeout interval is equal to the Sync interval, this meant that the state machine waited until the next Sync interval

c) Unfortunately, the next Sync arrived slightly early; it was ignored because the state machine does not process the next Sync until it is finished processing the current Sync (or has declared the current Sync or Follow_Up lost)

- It was decided early in the development of 802.1AS not to require processing of multiple outstanding Follow_Up messages (for multiple Sync messages), to avoid complexity; this behavior is desired

# Scenario leading to sync receipt timeout - 4

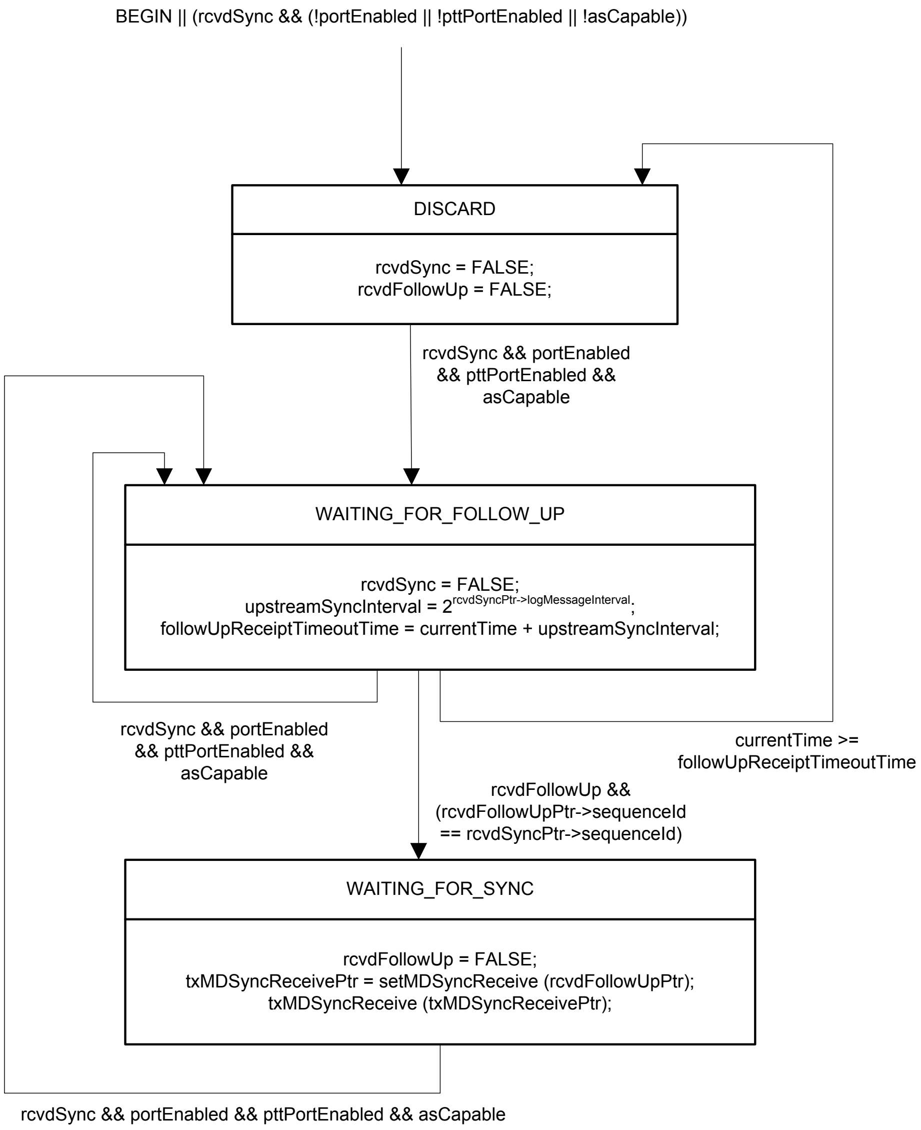
c) Note that the timeout timers are based on the mean Sync interval, which is configured; there is no allowance for variability

- This is as desired; timeouts occur when a timer is exceeded

# Possible Fix - 1

- ❑It was suggested in the June 19, 2013 TSN call that a simple fix would be to declare a Follow_Up message lost if it has not arrived by the time the next Sync message arrives
- ❑This is reasonable, because a sender sends Follow_Up corresponding to the most recent Sync message it has sent before sending the next Sync message
  - ▪This is implied by the MDSyncSendSM state machine (see Figure 11-7/802.1AS-2011)
  - ▪The order of the frames will not change on the link between the sender and receiver, which means that Follow_Up corresponding to a Sync message should arrive before the next Sync message
- ❑This fix can easily be incorporated
  - ▪See next slide for the revised MDSyncReceiveSM State Machine

# Possible Fix - 2


BEGIN || (rcvdSync && (!portEnabled || !pttPortEnabled || !asCapable))
DISCARD
rcvdSync = FALSE;
rcvdFollowUp = FALSE;
rcvdSync && portEnabled && pttPortEnabled && asCapable
WAITING_FOR_FOLLOW_UP
rcvdSync = FALSE;
upstreamSyncInterval = $2^{rcvdSyncPtr->logMessageInterval}$;
followUpReceiptTimeoutTime = currentTime + upstreamSyncInterval;
rcvdSync && portEnabled && pttPortEnabled && asCapable
currentTime >= followUpReceiptTimeoutTime
rcvdFollowUp && (rcvdFollowUpPtr->sequenceId == rcvdSyncPtr->sequenceId)
WAITING_FOR_SYNC
rcvdFollowUp = FALSE;
txMDSyncReceivePtr = setMDSyncReceive (rcvdFollowUpPtr);
txMDSyncReceive (txMDSyncReceivePtr);
rcvdSync && portEnabled && pttPortEnabled && asCapable

- Add a branch out of the WAITING_FOR_FOLLOW_UP state back to itself, with the condition rcvdSync && portEnabled && pttPortEnabled && asCapable
- If a new Sync message is received before either Follow_Up for the current Sync is received or followUpReceiptTimeoutTime is reached, the WAITING_FOR_FOLLOW_UP state is reentered and the followUpReceiptTimeoutTime is reset

# Additional Point

- Aside from the issue discussed in this presentation, the current state machine is in error with respect to its current behavior
  - If the state machine is in the WAITING_FOR_FOLLOW_UP state and a new Sync is received before the FOLLOW_UP corresponding to the current Sync, then
    - rcvdSync will be set to TRUE
    - rcvdSyncPtr will now point to the new Sync that is received
  - If the Follow_Up corresponding to the current Sync is now received, the test rcvdFollowUpPtr->sequenceid == rcvdSyncPtr->sequenceid will fail because rcvdSyncPtr points to the new Sync while rcvdFollowUpPtr points to the Follow_Up corresponding to the old Sync
    - To achieve the desired old behavior (i.e., keep waiting for the Follow_Up, even if a new Sync arrives first), rcvdSyncPtr should have been saved, and the saved value used in the test on sequenceid
  - In any case, the changes on the previous slide make this problem for the current state machine no longer relevant

# Conclusion

- ❑The revised MDSyncReceiveSM State Machine on slide 11 acheives the behavior suggested in the June 19, 2013 TSN call, and fixes the issue described here
- ❑If this is acceptable to the TSN TG, this change can be inserted into the first 802.1ASbt draft

# References

[1] Emails from Rune Haugom of May 31, 2013 and June 5, 2013