An algorithm for sequence recovery

Norman Finn
Cisco Systems

July 3, 2015

Introduction

- new-tsn-specht-cb-failure-modes-0521-v1 gave examples of failure modes that the sequence recovery algorithm in P802.1CB D1.1 clause 7.3.2 does not handle, well.
- Most of the problems can be characterized by the inability of the D1.1 algorithm to distinguish between:
  - Normal operation, during which duplicate packets are dropped.
  - Legitimate violations of proper sequencing caused by resetting the Sequence Generation function.
  - Restarting a Sequence Recovery function while a Stream is in progress.
  - Error conditions (e.g., repeated transmissions of the same packet) that can confuse the Sequence Recovery function.
- This is an attempt to describe a small change to the D1.1 algorithm that may be more satisfactory.

Proposal: add a timer to the Recovery function

- The timer is reset to the value T_{reset} whenever a packet is accepted and passed on, but not when a packet is discarded.
- If the timer reaches 0, SequenceRecoveryReset() is called.
- SequenceRecoveryReset () resets all history bits to “not seen” and sets the new variable tsnSeqTakeAny to TRUE.
- **For rogue packets: (**sequenced outside the range of “delta” [P802.1CB D1.1 page 38 line 22]) If tsnSeqTakeAny is FALSE, the packet is discarded (SequenceRecoveryReset is not called). If tsnSeqTakeAny is TRUE, then the packet is accepted and tsnSeqTakeAny is reset to FALSE.
- The managed object tsnSeqReqResetOnRogue is no longer needed.

Proposal: add a restriction to the SG function

- A Sequence Generation function is required to never transmit packets with non-consecutive sequence_number values within time T_{reset}.
  - This can be accomplished by ensuring that a reset takes longer than T_{reset}.
  - This can be accomplished by adding a delay of T_{reset} before the first transmission.
  - This can be accomplished by caching the last-transmitted sequence number so that it is retained in non-volatile memory across a reset.
- I would propose no normative algorithm be specified in P802.1CB—just the requirement—and leave it to the implementer to do the right thing. However, we can mention the above choices, and can put something in the PICS requiring an explanation of how the requirement is met, so that buyers can tell whether a piece of equipment is satisfactory.

Proposal for P802.1Qci:

- In addition to these changes, I would suggest that the Input Gates have the ability to:
    1. Take note of, and complain to the network administrator about, streams that are not arriving in order.
    2. Suppress packets whose sequence_number value duplicates the last-received sequence_number value for a given Stream.

With these changes in place

- The examples of a “stuck” transmitter cause the repeated packets to be discarded, and they do not interfere with the good stream, until the packet number wraps around to the stuck value, again, in which case the stuck packet will likely be transmitted in place of the correct packet.
  - The assumption is that the Qci input error detection will either suppress repeated packets with the same sequence number, or at least notify the network administration to fix the problem before the sequence number can wraps around.
- If a Sequence Generation function is reset, then all of the Sequence Recovery functions will time out, and accept the new Stream.
- If the stuck transmitter is still going after a reset, there is still a 50/50 chance that the Sequence Recovery functions will recover, properly.
  - But, Qci input error detection fixes this problem, too.

Why did this author not propose a "Talker reset" handshake protocol?

Well, he tried. His conclusion:

1. The reset is unreliable, because there can be multiple Listeners joining, leaving, and/or being reset at the same time that the Talker is being reset.
2. The reset protocol bits are either:
   - Carried in the protocol, which makes interworking with other protocols (HSR, pseudowires) much more difficult;
   - Separate best-effort packets, which means they may not follow the same path through the network as the Stream packets, so they may be out of order with the Stream, or not even arrive; or
   - Separate in-stream packets, which means they must be accounted for in the Stream bandwidth calculation.

But, perhaps someone else is more clever and can come up with a protocol.

Thank you.

cisco