The side effects of "one-size-fits-all"

Balázs Varga, János Farkas – Ericsson

2025-07-09

Analysis of reset functions defined in 802.1CB-2017

There is a single "reset" method defined in 802.1CB-2017, which is used for different events. However, the characteristics of these events and the expected reactions to them differ significantly. Tailor made reactions to these events could eliminate the side effects of a single one-size-fits-all reset functionality.

1, BEGIN event related reset

This is a global event that resets all functions, including the sequence recovery function. This event triggers the execution of the SequenceRecoveryReset routine (7.4.3.3).

This event happens, e.g., during the boot of a system (e.g., after power-on or a re-start). A system can be a hardware node or a virtualized one. In case of hardware node, the boot is usually longer (10s of sec) than the e2e delay of a network (sub-sec). In case of virtualized node, the boot can be very fast, even shorter than the e2e delay of the network.

Expectation: no memory about former state of the node (as “power is gone”). All variables are initialized, counters are set to their initial value (e.g., error counters = 0). Forwarding is done with the assumption that no packets were forwarded in the past, i.e., before the BEGIN event.

Verdict: this is a traditional complete reset use-case, with a very likely outage period, which is expected and is not a surprise.

2, Management driven reset (frerSeqRcvyReset=True)

Setting the value of frerSeqRcvyReset to True triggers a reset via its corresponding SequenceRecoveryReset function (7.4.3.3).

This event is typical during troubleshooting. For instance, if some issues/anomalies/problems/etc. are detected by the operational personnel and they intend to re-initialize the system by reseting selected functions. These functions are considered to cause partly or entirely the unwanted network behavior.

The reset of the SeqRcvy is a very fast action as it affects only some variables. The standard does not define all the details, only the execution of the SequenceRecoveryReset function. It is up to an implementation to define additional actions, potentially adapted to typical operation & maintenance procedures. Such add-ons can be re-initialization of further error-specific counters of the system. Such a related function is the Latent Error Detection Function (7.4.4). The standard does not define the relation between a management driven reset and the latent error detection (note: only BEGIN event specifics are defined).

The management driven reset of the SeqRcvy function might be combined with reset of other functions, like the SeqGen function.

Expectation: starting a "new life" of the function is a usual expectation, where former state of the function is forgotten during the reset. Operational personnel usually also expect some transient disturbance of the corresponding traffic flows (e.g., typically short-term drops), but no further trouble (e.g., duplicates impacting network resources and other streams).

Verdict: this is a widely used reset use-case, where short time disturbance is taken into account, it is not a surprise.

3, RECOVERY_TIMEOUT related reset

The event occurs when the RemainingTicks (7.4.3.2.4) variable reaches 0. It triggers the execution of the SequenceRecoveryReset routine (7.4.3.3).

This event is an important part of the SeqRcvy functionality, to ensure that it can return to its normal operation in scenarios where a base recovery function somehow gets out of step with its corresponding sequence generation function. Such out of step situations results in packet drops as out-of-window packets are treated as "rouge" and get discarded. So, this is not really a "reset" situation but more of a "state mismatch" between the SeqGen and SeqRcvy functions.

How can the out-of-step situation arise?
1, SeqGen function was reset (e.g., reboot, management action)
2, Network failures on ALL paths towards the SeqRcvy function
3, Other !?

Expectation:
Situation "1" needs immediate reaction at the recovery point to avoid drop of packets and possible emergency-stop of the application belonging to the stream. Situation "1" can be easily detected via signaling, where the SeqGen function informs the SeqRcvy function about the presence of SeqGen reset, like discussed in former contributions. Any outage of the Stream can be avoided.

Situation "2" is a multi-failure scenario, where a stream outage is already happening. The main challenge here is the detection of situation "2", as it cannot be distinguished from the situation, where, e.g., the Talker temporarily stops sending any packets to the Listener(s). A temporarily stopped stream does not need any action from the network as the SeqGen and SeqRcvy are in-step (i.e., NOT out-of-step).

Anyway, as in situation "2" the stream is already failed in the system, a somewhat longer outage (e.g., 36.1 sec instead of 36 sec) are less relevant. The primary expectation is to ensure a safe return to normal operation. If ALL paths towards a SeqRcvy function fail, most probably the operator has more severe concerns than the outage of a single stream.

Verdict: this is NOT a usual reset use-case, it is an inherited part of the SeqRcvy concept.

Summary:

- The current single reset function is under-specified and sub-optimal
- Tailor made reset functions are needed for the above situations
- Exclude the reaction to out-of-step situation from reset, and make it as an integral part of SeqRcvy