P802.1CBec

FRER refinements for vPLC/NFV scenarios

IEEE 802.1 TSN TG
2026-02-10

Balázs Varga, Ferenc Fejes, János Farkas Ericsson Research 2026-02-10

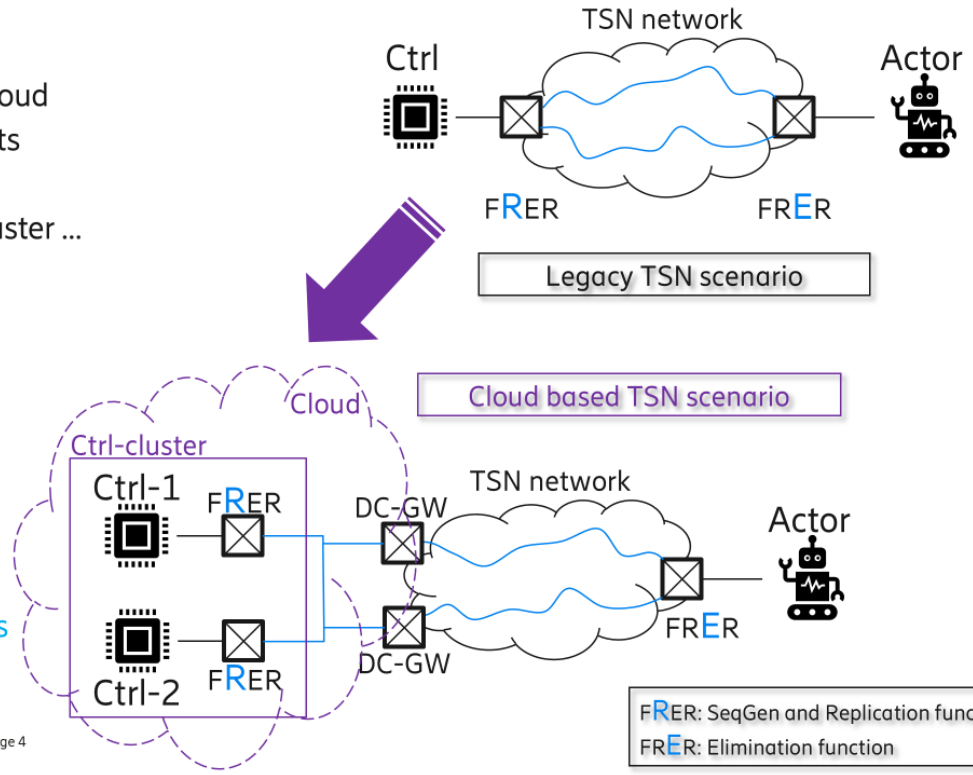
vPLC/NFV scenarios

- Much more frequent state changes of SeqGen & SeqRcvy functions
- Replication / Elimination functions may be virtual functions
- Create/Termination of them involves reset of the functions

Moving towards Virtualized environments
Using FRER in a Cloud based scenario

Scenario: Moving a Talker/Listener to the Cloud
- TSN functions must go with the endpoints
  - FRER must work inside Cloud ...
  - FRER can be an instance in a Ctrl-cluster ...
- Typical Cloud actions
  - Run multiple VMs/instances
  - Create a VM/instance
  - Move a VM/instance
  - Reset a function
  - Remove a VM/instance
  - Etc.

Main motivation for proposed improvements is to allow cloudification, i.e., "cloud native" implementation of FRER functions.



Balázs Varga, János Farkas | 2020-03-16 | TSN - FRER improvements (Seamless Reset) | Page 4

Overview: Root Cause and Solution

Root cause of issues (it is all about SeqNum)

- SeqGen and SeqRcvy have their own internal states to drive their operation
- SeqGen and SeqRcvy functions require to be "in-step" (i.e., have the same view on the expected SeqNum range of a given Stream's packets in flight in the network)
- Currently, there is no state communication between SeqGen and SeqRcvy, hence, "out-of-step" situations are "identified" based on timer-based guesses driven by the observed SeqNum of the received packets
- Timer-based reaction to "SeqNum out-of-step" situations is hard to achieve and has side effects

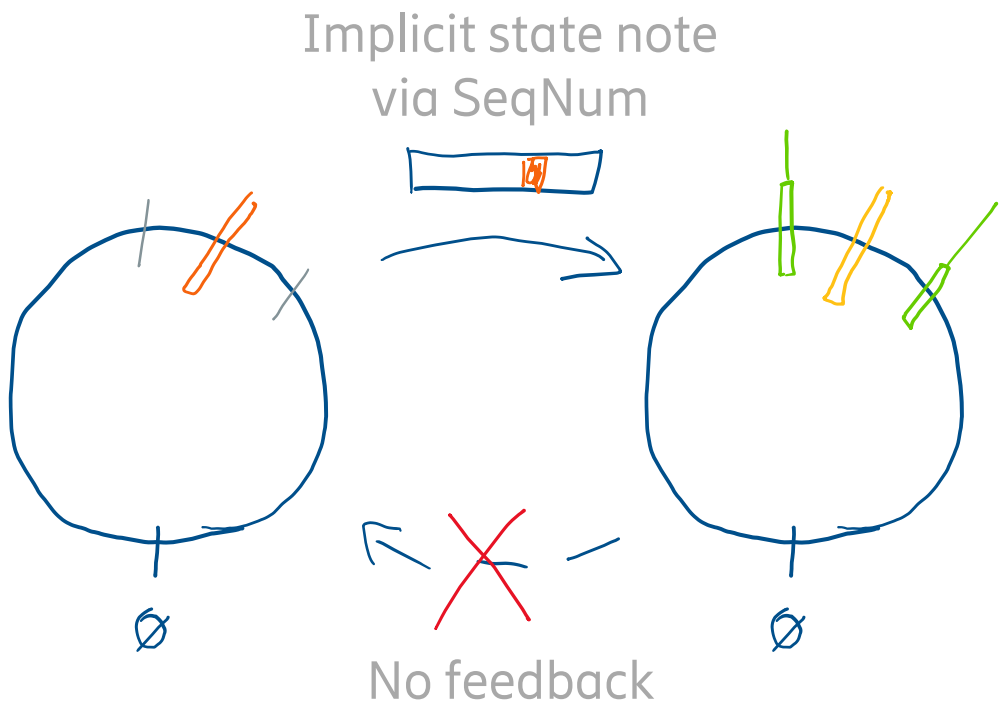
Solution: Explicit Signaling (see details on next slides)

- Add explicit signaling between SeqGen and SeqRcvy on SeqNum specific actions
  - Signaling can be in-band & unidirectional (SeqGen → SeqRcvy)
- Keep timer-based reaction as a last resort for extreme failure scenarios (e.g., too many paths fail)

States: in-step (normal operation)

GenSeqNum

GenSeqSpace



Guess based on SeqNum

RecovSeqNum

SequenceHistory

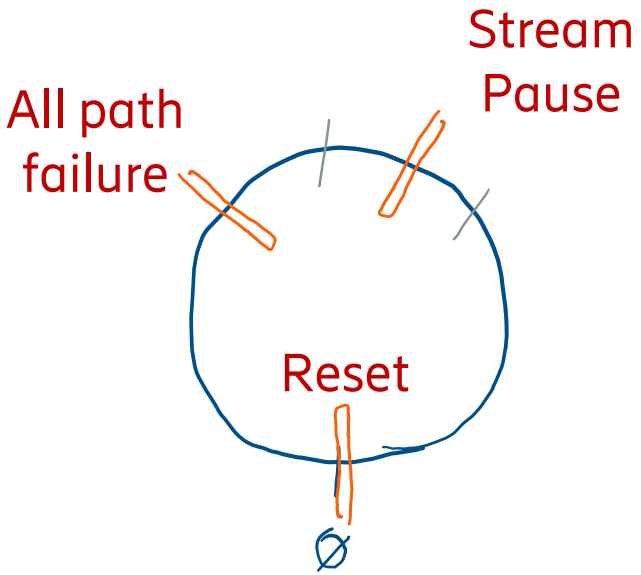
RecovSeqSpace

Assumptions: always in-step operation

— At least one path alive between SeqGen/SeqRcvy

— Continuous stream

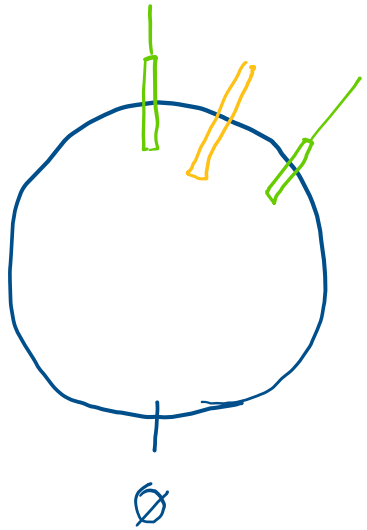
States: out-of-step (detection & abnormal operation)



GenSeqNum

GenSeqSpace

SeqGen



SeqRcvy

RecovSeqNum

SequenceHistory

RecovSeqSpace

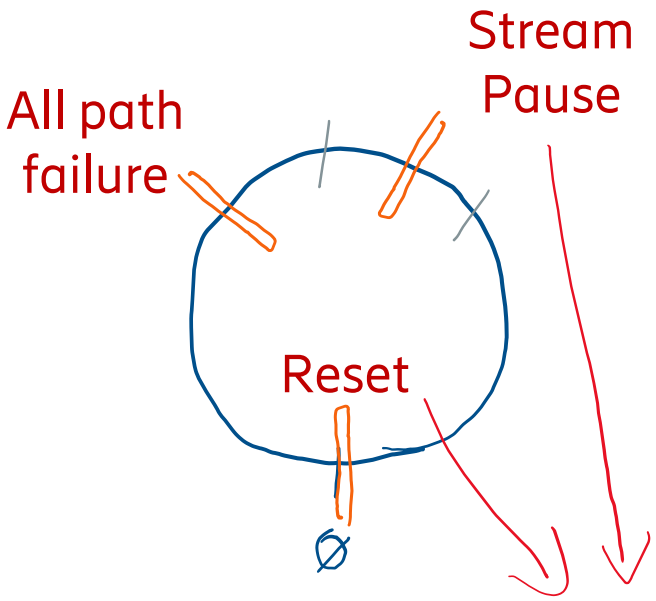
Detection: solely timer based

— Cannot distinguish reason for out-of-step (reset, failure, pause)

— False detection (e.g., sender pause scenario)

States: out-of-step (detection & abnormal operation)

Rare in vPLC/NFV

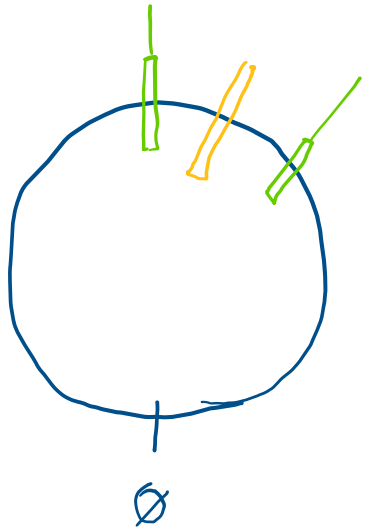


GenSeqNum

GenSeqSpace

SeqGen

Most frequent in vPLC/NFV



SeqRcvy

RecovSeqNum

SequenceHistory

RecovSeqSpace

Detection: solely timer based

- Cannot distinguish reason for out-of-step (reset, failure, pause)
- False detection (e.g., stream pause scenario)

SeqGen/SeqRcvy Improvements
Overview

How: Change Sequence Number Space characteristics

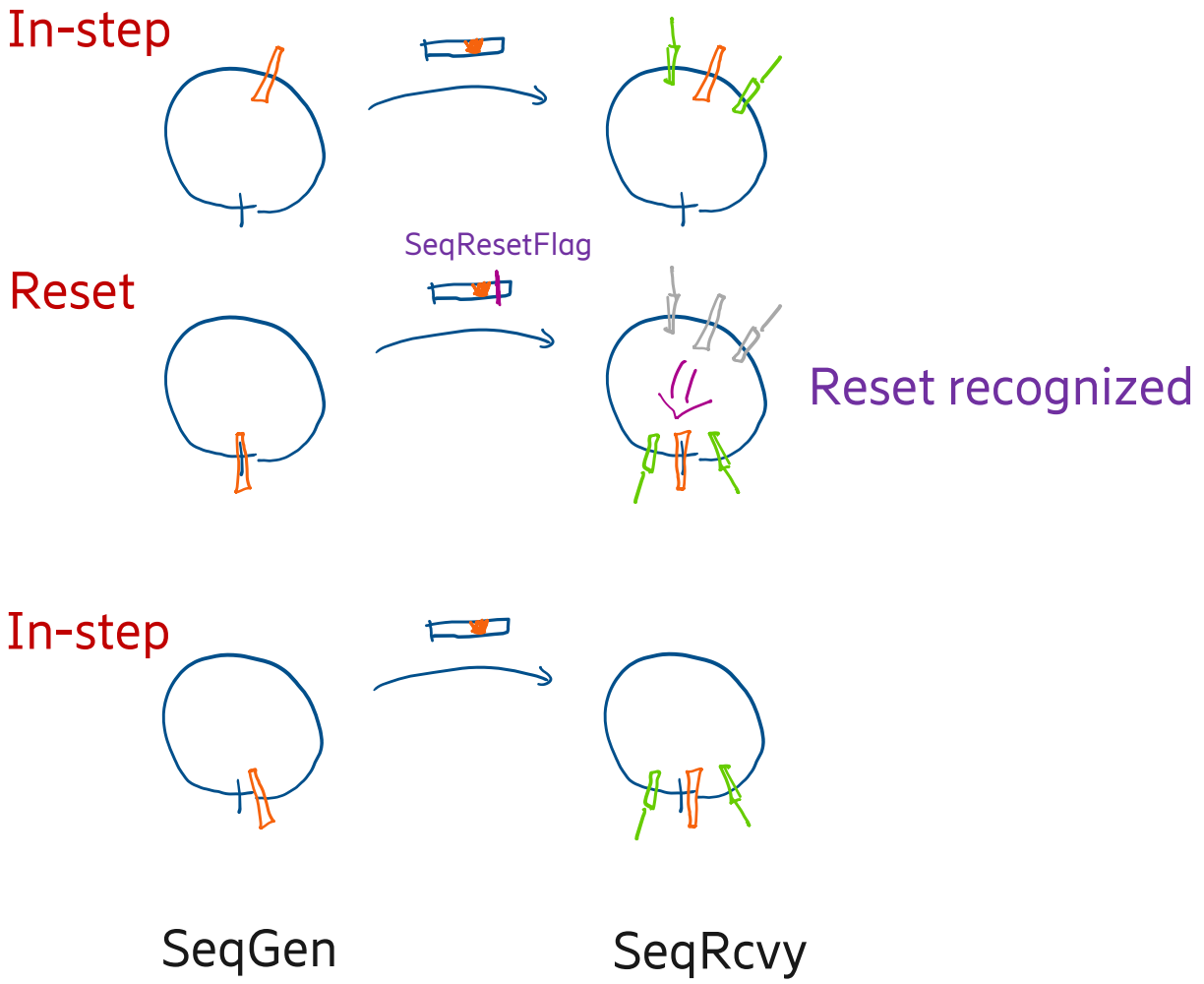
1. explicit notification of the reset event, and
2. extend the sequence number space
   (additional new linear initial sequence number space)

1. Explicit notification of the reset event is based on a flag
   - "SeqResetFlag": signals a recent SeqGenReset event (e.g., first 3 packets sent with the flag set after the SeqGenReset event)
2. Extension of the SeqNum space
   - SeqNum space I.: used during normal operation (circular sequence number space)
   - SeqNum space II.: used after SeqGenReset event (linear sequence number space)
   - SeqNum spaces are distinguished by a flag "InitSeqFlag"

1, NEW Reset Notification
Operation Basics

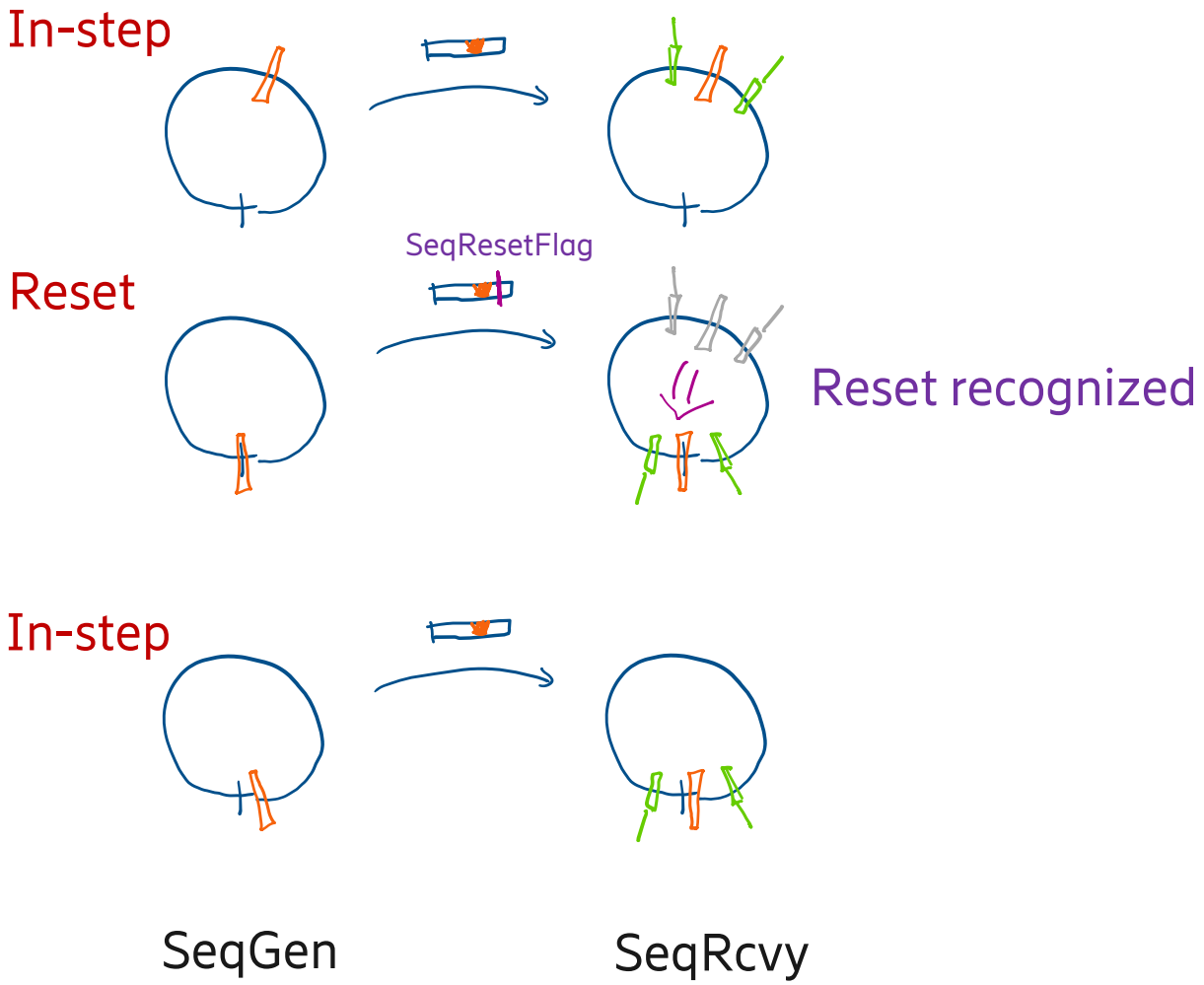
Explicit notification of the reset event, to avoid mess-up and allow immediate adaptation to new state.

- SeqGen function reset
  - After reset, multiple (e.g., 3) packets are marked with a new flag, namely the "SeqResetFlag".
  - New flag means: the SeqGen function was recently reset.
- SeqRcvy function
  - Capable to detect the reset of the SeqGen function and adapt to the situation.
  - Prepare itself for the reception of packets with SeqNum that are out of the History Window with a high probability.



1, NEW Reset Notification
Issue: in-flight packets

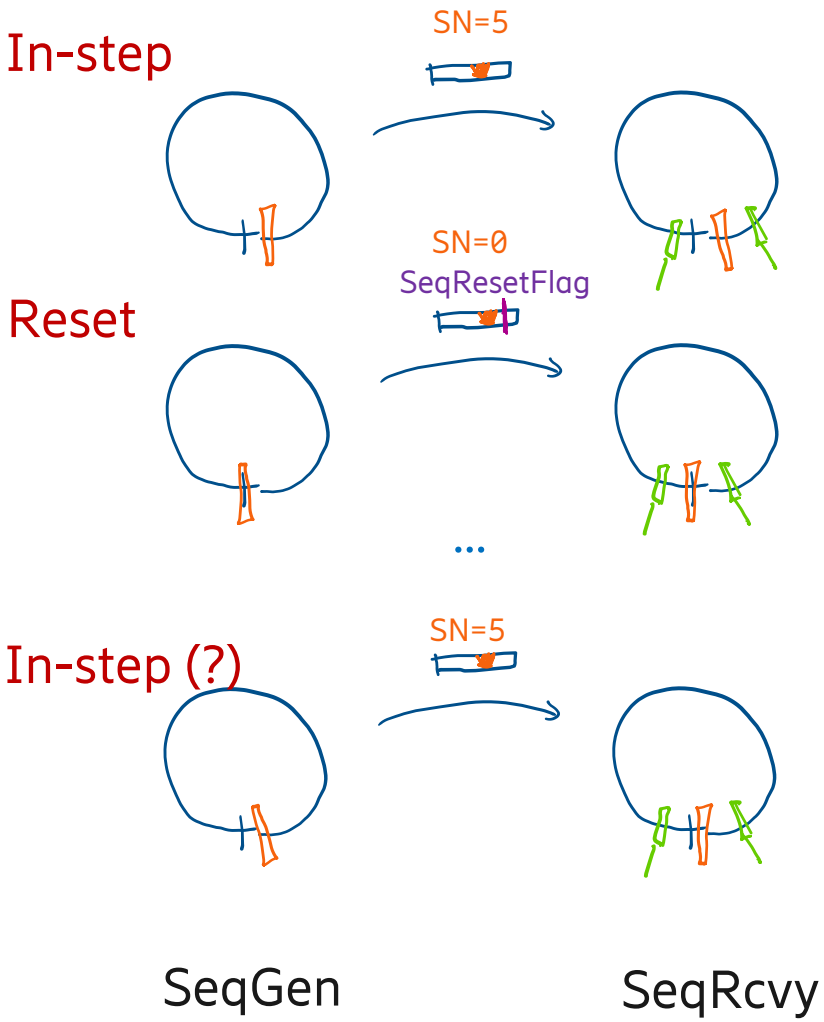
- Question:
  - What about packet(s) that are in-flight over slow-path(s and that have been sent before the reset but are about to arrive after the reset signal recognized?
- Note:
  - SeqResetFlag is not enough to cover all scenes.
  - Would be great to be able to distinguish packets sent before/after reset …



1, NEW Reset Notification
Issue: overlapping History Window

- Issue: there is a non-zero probability that the History Windows overlap before and after the reset
  - I.e., Delta of SeqNum(before reset) and SeqNum(after reset) is within HistoryWindow (5 - 0 = 5 < 12)

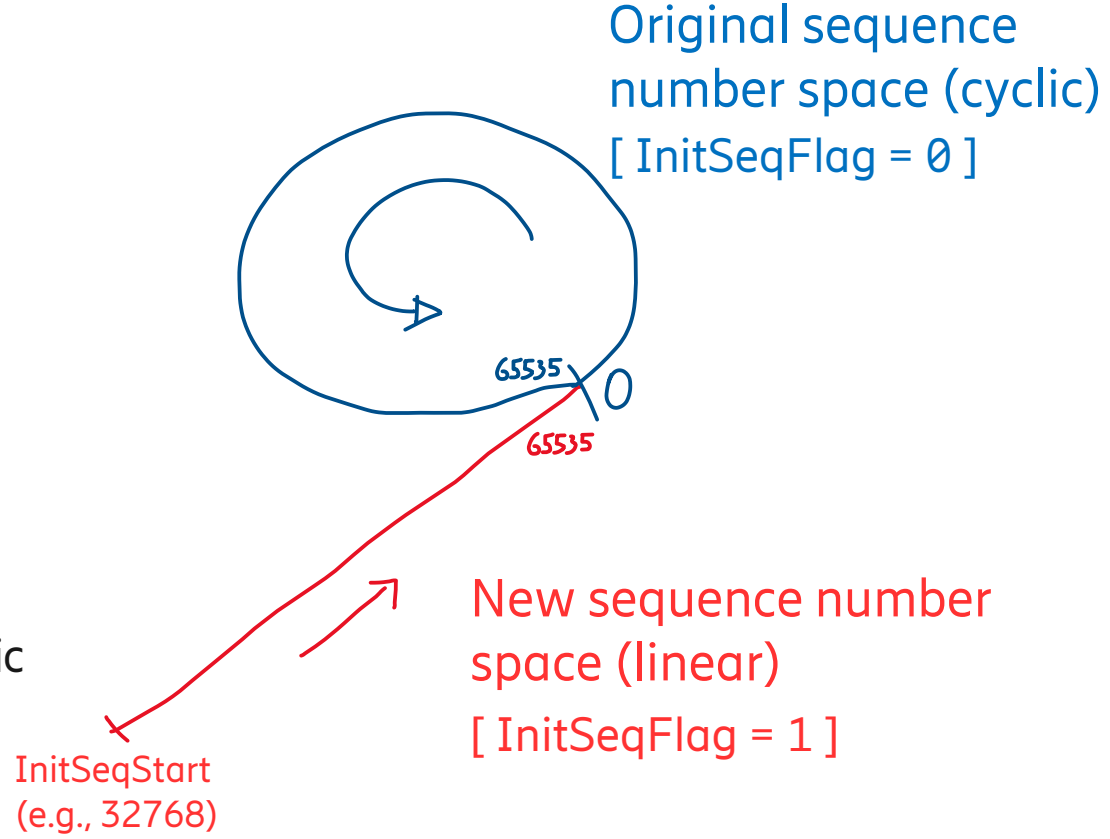
- Questions:
  - Drop or forward packets that are within the History Window after reset? (maybe already seen)
  - What about packet(s) that are in-flight over slow-path(s) and that have been sent before the reset but are about to arrive after the reset signal recognized?



2, NEW Sequence Number Space
Operation Basics

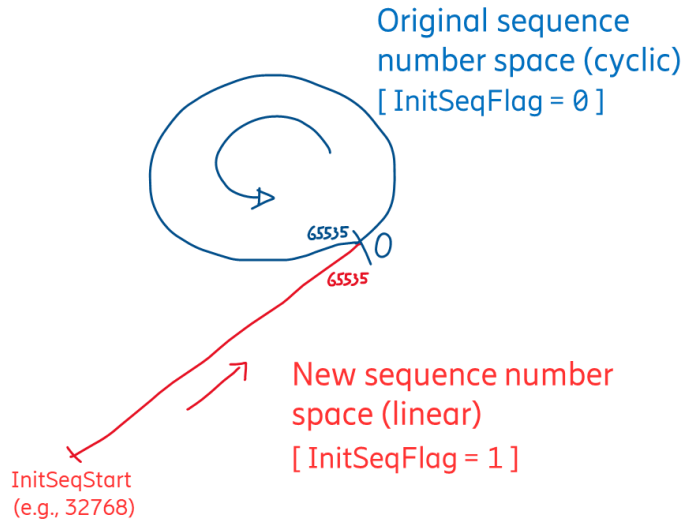
Distinct sequence number spaces for packets before and after reset make sure that sequence recovery does not get messed up.

- Solution: Introduce a new additional sequence number space "*InitSeqNumSpace*", which is used after initialization/reset of the sequence generation function.
  - *InitSeqNumSpace* is illustrated by the red linear sequence number space.
  - The sequence_number of the next packet is stored in a new variable, namely the "*InitGenSeqNum*".
  - When this new sequence number space is exhausted, the sequence generation function starts to use the original sequence number space, which is illustrated by the blue cyclic sequence number space.
- New flags:
  - The use of the new Sequence Number Space is marked by a new flag, namely the "InitSeqFlag".
  - After reset, multiple (e.g., 3) packets are marked with a new flag, namely the "SeqResetFlag".



An implementation may use e.g., two variables to enable/disable the use of the reset flag ("*UsingResetFlag*" = 1/0 (enable/disable)) and the new initial sequence number space ("*UsingInitSpace*" = 1/0 (enable/disable)).

2, NEW Sequence Number Space
Operation Details (InitSeqNumSpace)



"*InitSeqNumSpace*" is used as follows:

- When sequence generation function is initialized/reset, the "*InitSeqNumSpace*" is used to generate the sequence number for the packets.
- "*InitSeqNumSpace*" is a linear sequence number space starting with "*InitSeqStart*" (configurable, e.g., 32768) and ends with "GenSeqSpace-1" (i.e., 65535 as per 802.1CB-2017).
- When this new sequence number space is exhausted the sequence generation function starts to use the original sequence number space

- "*InitSeqNumSpace*" has its own set of variables that are used to process the sequence_number of a packet by the Replication function and Elimination function.
  Note: *InitSeqNumSpace* variable names are derived from the variable names of the original sequence number via appending them with an "*Init*" prefix, e.g., "*InitGenSeqNum*", "*InitRecovSeqNum*", "*InitSequenceHistory*", "*InitTakeAny*", "*InitRemainingTicks*", etc.

Realization: Encoding the Flags and the SeqNum
R-TAG to be updated

- Explicit notification of the reset event is based on the flag carried in the R-TAG, namely the "SeqResetFlag".
- The use of the new linear initial sequence number space ("InitSeqNumSpace") is marked by a new flag carried in the R-TAG, namely the "InitSeqFlag".
- Sequence values of the new sequence number space ("InitSeqNumSpace") are also carried in the R-TAG (in the existing sequence number field).

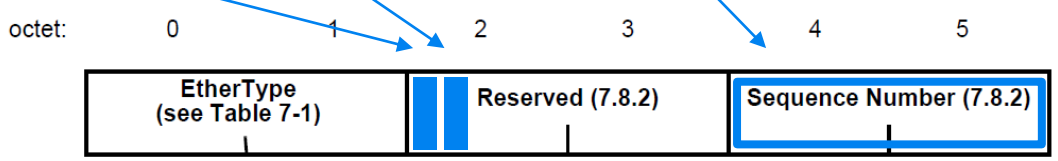


Figure 7-4 in IEEE 802.1CB

Summary of solution components

- ResetFlag (i.e., explicit signaling)
  - Informs SeqRcvy function about the reset of the SeqGen function
  - Expected SeqNum value of the next packets known

- InitSeqNumSpace (i.e., distinct sequence number space)
  - Solves HistoryWindow overlap before and after the reset
    - Delta of SeqNum(before reset) and SeqNum(after reset) is within History Window
  - Solves all possible state combinations of SeqGen/SeqNum during the reset

Impacted parts of 802.1CB

Potential changes

- Reset functionality of
  - Sequence generation
  - Sequence recovery
- State machines of SeqGen and SeqRcvy
- R-Tag (e.g., reset related signaling)
- State variables, counters

Discussion