802.1CB-2017

Maintenance item #378

Christophe Mangin

2025/01/21

MITSUBISHI ELECTRIC R&D CENTRE EUROPE

Content

- Maintenance item #378
- VectorRecoveryAlgorithm behavior upon BEGIN or TIME_OUT events
- Proposed work-around

802.1CB-2017, item #378

- Packets not seen after a sequence history reset are considered as lost
  - Description
    - Following the initialization of the variables used by the VectorRecoveryAlgorithm upon a BEGIN or TIME_OUT event, the VectorRecoveryAlgorithm calls the ShiftSequenceHistory routine when it processes a packet with a sequence number that is in the future (but still within the SequenceHistory). The ShiftSequenceHistory routine erroneously increments the counter frerCpsSeqRcvyLostPackets as it cannot differentiate the packets that are actually lost from the packets that are not yet seen. This is due to the fact that SequencyHistory is initialized to all zeros by the SequenceRecoveryReset routine that initializes the SequenceRecovery instance.

VectorRecoveryAlgorithm behavior upon BEGIN or TIME_OUT

- Functions and variables involved when the *VectorRecoveryAlgorithm* is used
  - `SequenceRecoveryReset`
    - Function called when the BEGIN event or the RECOVERY_TIMEOUT event occurs.

```
void SequenceRecoveryReset (
      if (frerSeqRcvyAlgorithm == Vector_Alg) {
            int i;
            RecovSeqNum = RecovSeqSpace - 1;
            for (i = 0; i < frerSeqRcvyHistoryLength; i = i + 1)
                  SequenceHistory[i] = 0; // Set all bits 0 (packet not seen)
      }
      frerCpsSeqRcvyResets = frerCpsSeqRcvyResets + 1;
      TakeAny = true;
}
```

frerSeqRcvyHistoryLength = 8

SequenceHistory	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
	0	0	0	0	0	0	0	0

RecovSeqNum = 65535
TakeAny = true

  - `ShiftSequenceHistory`
    - Function called by the *VectorRecoveryAlgorithm* routine to advance the *SequenceHistory* bit array and to count lost packets (*frerCpsSeqRcvyLostPackets*). *ShiftSequenceHistory* takes one parameter, which is the new value for index 0 in the *SequenceHistory* bit array.

```
void ShiftSequenceHistory (int newZeroValue) {
      int i;
      if (0 == SequenceHistory[frerSeqRcvyHistoryLength - 1])
            frerCpsSeqRcvyLostPackets = frerCpsSeqRcvyLostPackets + 1;
      for (i = frerSeqRcvyHistoryLength - 1; i != 0; i = i - 1)
            SequenceHistory[i] = SequenceHistory[i - 1];
      SequenceHistory[0] = newZeroValue;
}
```

VectorRecoveryAlgorithm behavior upon BEGIN or TIME_OUT

- *VectorRecoveryAlgorithm*
  - Immediately after *SequenceRecoveryReset* is called, the *VectorRecoveryAlgorithm* accepts the first packet received as valid. After the first packet has been accepted, all subsequent packets that are in the recovery window (i.e., last packet number accepted – *frerSeqRcvyHistoryLength* + 1 to last packet number accepted + *frerSeqRcvyHistoryLength*) are accepted, and and those packets with *sequence_number* values outside that range are discarded. […]

- The following pseudo-code only shows the branches of the *VectorRecoveryAlgorithm* taken when the *SequenceRecoveryReset* function has been invoked, i.e. upon initialization of the SequenceRecovery instance

VectorRecoveryAlgorithm behavior upon BEGIN or TIME_OUT

```
void VectorRecoveryAlgorithm () {
// Check that sequence number is present in the packet
unsigned int sequence_number;
if (sequence_number == frerSeqRcvyInvalidSequenceValue) {

      …
}
// Compute signed difference modulo RecovSeqSpace.
int delta = (sequence_number-RecovSeqNum) & (RecovSeqSpace - 1);
if (0 != (delta & (RecovSeqSpace/2)))
      delta = delta - RecovSeqSpace;
// Here, -(RecovSeqSpace/2)<=delta<=((RecovSeqSpace/2)-1)
// After reset, accept any packet
if (TakeAny) {
      TakeAny = false;
      SequenceHistory[0] = 1; // Shift, adding a "seen" bit
      RecovSeqNum = sequence_number;
      frerCpsSeqRcvyPassedPackets = frerCpsSeqRcvyPassedPackets + 1;
      frerCpSeqRcvyPassedPackets = frerCpSeqRcvyPassedPackets + 1;
      RemainingTicks = ((frerSeqRcvyResetMSec * TicksPerSecond) + 999) / 1000;
      PRESENT_DATA;
} else if (delta > frerSeqRcvyHistoryLength || delta <= -frerSeqRcvyHistoryLength) {
      // Packet is out-of-range. Count and discard it.

      …
      // Reset timer if working on an individual Stream

      …
      } else if (delta <= 0) {
            // Packet is old and in SequenceHistory; have we seen it before?
                  // Packet has not been seen. Take it.
                  // Packet has been seen. Do not forward. Count the discard.
                  // Reset timer if working on an individual Stream
      }
```

First packet received after BEGIN :

Sequence_number = 0

SequenceHistory	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
	0	0	0	0	0	0	0	**1**

RecovSeqNum = 0

TakeAny = false

Next packet received :

Sequence_number = 2

Delta = 2

Next + 1 packet received :

Sequence_number = 5

Delta = 3

VectorRecoveryAlgorithm behavior upon BEGIN or TIME_OUT

```
        } else {
                // Packet is not too far ahead of the one we want.
                // Packet is out-of-order unless it directly follows RecovSeqNum
                if (delta != 1)
                        frerCpsSeqRcvyOutOfOrderPackets = frerCpsSeqRcvyOutOfOrderPackets + 1;
                // Shift the history until bit 0 refers to sequence_number.
                while (0 != (delta = delta - 1))
                        ShiftSequenceHistory(0); // Shift, adding a "not seen" bit
                ShiftSequenceHistory(1); // Shift, adding a "seen" bit
                RecovSeqNum = sequence_number;
                frerCpsSeqRcvyPassedPackets = frerCpsSeqRcvyPassedPackets + 1;
                frerCpSeqRcvyPassedPackets = frerCpSeqRcvyPassedPackets + 1;
                RemainingTicks = ((frerSeqRcvyResetMSec * TicksPerSecond) + 999) / 1000;
                PRESENT_DATA;
        }
}
```

```
void ShiftSequenceHistory (int newZeroValue) {
        int i;
        if (0 == SequenceHistory[frerSeqRcvyHistoryLength - 1])
                frerCpsSeqRcvyLostPackets = frerCpsSeqRcvyLostPackets + 1;
        for (i = frerSeqRcvyHistoryLength - 1; i != 0; i = i - 1)
                SequenceHistory[i] = SequenceHistory[i - 1];
        SequenceHistory[0] = newZeroValue;
}
```

Next packet received :

Sequence_number = 2

Delta = 2

SequenceHistory

[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
0	0	0	0	0	0	**1**	0	**1**

RecovSeqNum = 2

TakeAny = false

frerCpsSeqRcvyLostPackets = 2

Next + 1 packet received :

Sequence_number = 5

Delta = 3

SequenceHistory

[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
0	0	0	0	0	**1**	0	**1**	0	**1**

RecovSeqNum = 5

TakeAny = false

frerCpsSeqRcvyLostPackets = 5

VectorRecoveryAlgorithm behavior upon BEGIN or TIME_OUT

- Observed side effect
  - *frerSeqRcvyHistoryLength* packets are counted as lost when a packet with a sequence number distant of *frerSeqRcvyHistoryLength* from the sequence number of the packet initially received after a BEGIN or TIME_OUT event
    - These packets may have never been sent.
  - Result of the *ShiftSequenceHistory* function shifting out sequence number positions from the sequence number history that are anterior to (smaller than) the first received packet's

- Proposed work-around
  - Detect the initialization phase of the SequenceRecovery instance
    - new Boolean variable set to TRUE by the *SequenceRecoveryReset* function : *SequenceHistoryInit*
    - *SequenceHistoryInit* remains TRUE as long as the sequence number positions anterior to the sequence number of the first packet received after a BEGIN or TIME_OUT event are not flushed out of the sequence history (*SequenceHistory[]*)
    - While *SequenceHistoryInit* is TRUE, *frerCpsSeqRcvyLostPackets* is not incremented in *ShiftSequenceHistory*

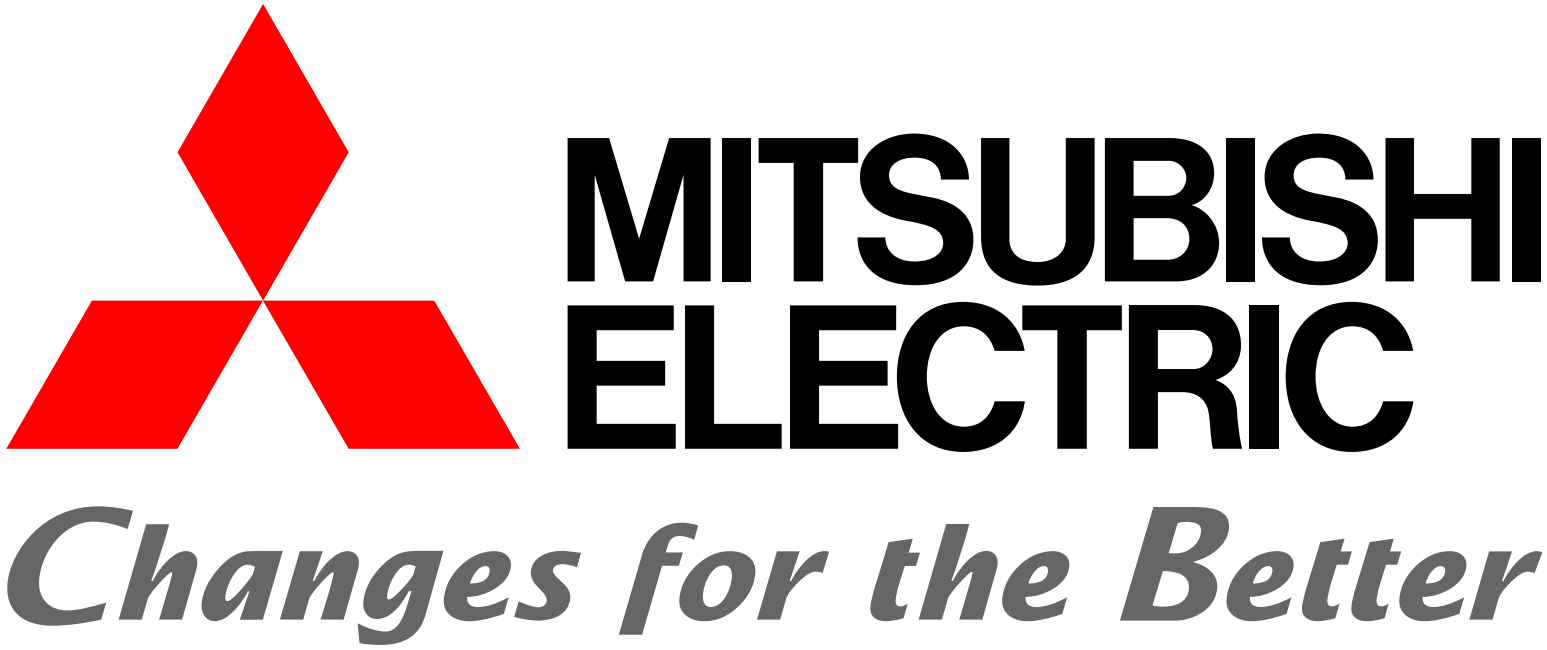
Proposed work-around

- Proposed modified recovery functions
  - *SequenceRecoveryReset*

```
void SequenceRecoveryReset (
      if (frerSeqRcvyAlgorithm == Vector_Alg) {
            int i;
            RecovSeqNum = RecovSeqSpace - 1;
            for (i = 0; i < frerSeqRcvyHistoryLength; i = i + 1)
                  SequenceHistory[i] = 0; // Set all bits 0 (packet not seen)
      }
      frerCpsSeqRcvyResets = frerCpsSeqRcvyResets + 1;
      SequenceHistoryInit = true;
      InitNotSeenPackets = 0;
      TakeAny = true;
}
```

  - *ShiftSequenceHistory*

```
void ShiftSequenceHistory (int newZeroValue) {
      int i;
      if (0 == SequenceHistory[frerSeqRcvyHistoryLength - 1]) {
            If SequenceHistoryInit == true {
                  InitNotSeenPackets = InitNotSeenPackets + 1;
                  If InitNotSeenPackets >= frerSeqRcvyHistoryLength;
                        SequenceHistoryInit = false;
            } else
                  frerCpsSeqRcvyLostPackets = frerCpsSeqRcvyLostPackets + 1;
      }
      for (i = frerSeqRcvyHistoryLength - 1; i != 0; i = i - 1)
            SequenceHistory[i] = SequenceHistory[i - 1];
      SequenceHistory[0] = newZeroValue;
}
```


MITSUBISHI
ELECTRIC
Changes for the Better