802.1CB-2017

Maintenance item #378

Venkat Arunarthi
Sunil Raj

March 10 2025

Topics

- Maintenance item #378 description
- Problem illustration with examples
- Proposed changes

Maintenance item #378

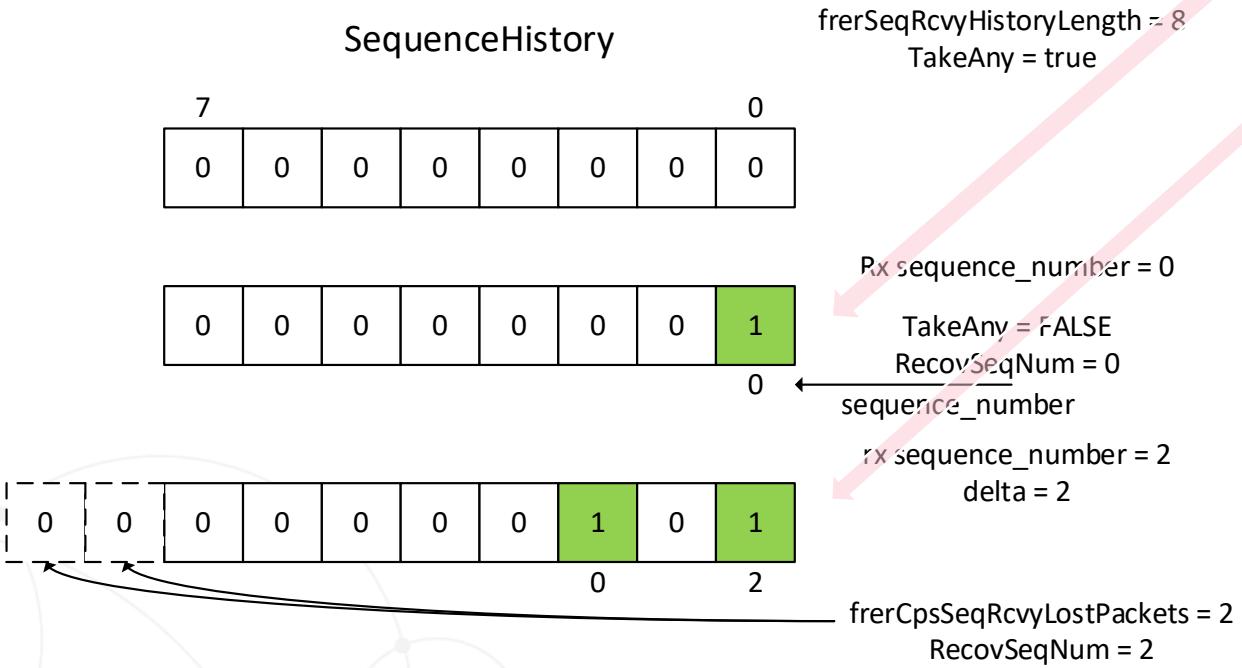
- Packets not yet seen after a sequence history reset are erroneously counted as lost
  - Description
    - The VectorRecoveryAlgorithm calls ShiftSequenceHistory routine when it processes a packet with a sequence number that is in the future (but still falls within the SequenceHistory). The ShiftSequenceHistory routine erroneously increments the counter frerCpsSeqRcvyLostPackets as it can't 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.

7.4.3.3 SequenceRecoveryReset

SequenceRecoveryReset is called whenever the BEGIN event (item a in 7.4.3.1) or the RECOVERY_TIMEOUT event (item c in 7.4.3.1) occurs. It resets the RecovSeqNum (7.4.3.2.3) and SequenceHistory (7.4.3.2.2) variables to their initial states, increments frerCpsSeqRcvyResets (10.8.9), and sets TakeAny (7.4.3.2.6). Note that RecovSeqNum and SequenceHistory are reset only if the VectorRecoveryAlgorithm (7.4.3.4) is configured.

```
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;
}
```

Problem Illustration

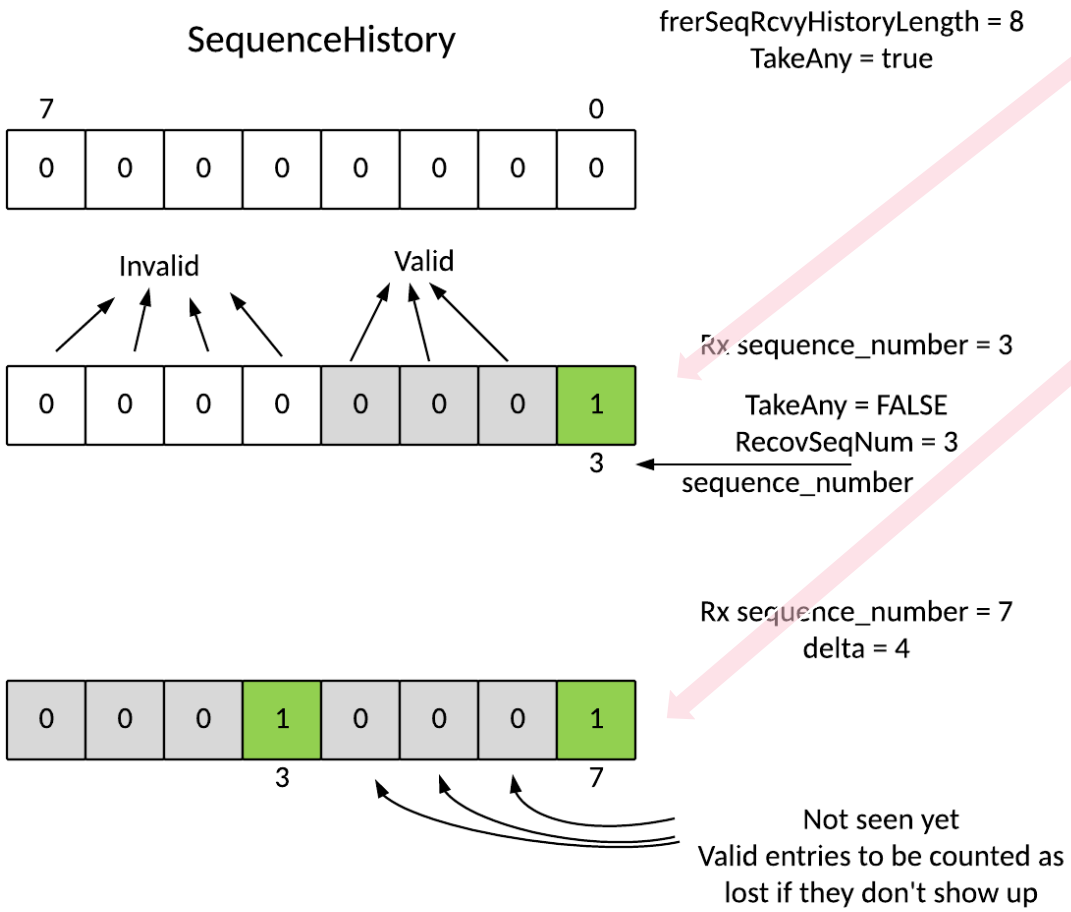


- First frame received with sequence_number 0 in this example
  - Bit 0 of SequenceHistory is set, bit 0 represents sequence_number 0
- Next frame that arrives with sequence_number 2
  - SequenceHistory array gets shifted as shown
- frerCpsSeqRcvyLostPackets is incremented erroneously twice for zero bits that got shifted out though these are not missing
- Potentially up to SequenceHistoryLength – 1 packets could be miscounted as lost
  - How many depends on the arrival order of sequence numbers.
- Algorithm needs to mark the array entries one through seven as invalid when first frame with sequence_number 0 is received

```
// 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;
}
```


BROADCOM

Another case

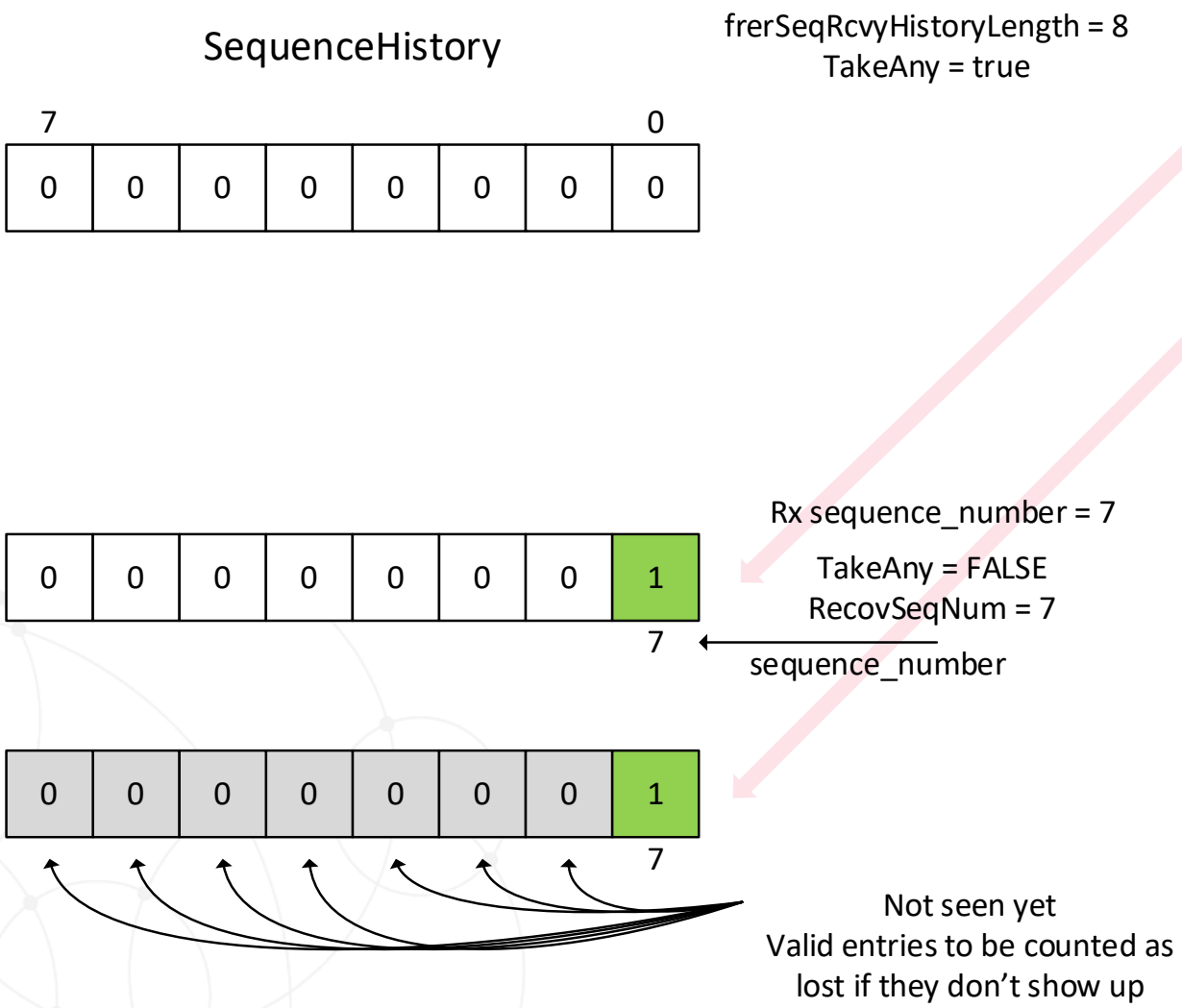


- First frame received with sequence_number 3 in this example
  - Bit 0 of SequenceHistory is set, bit 0 represents sequence_number 3
  - Three cells corresponding to bit positions 1, 2 and 3 are valid to be considered
  - Four cells corresponding to bit positions 4 through 7 are invalid, i.e., can't be evaluated for Lost Packets
- Next frame that arrives with sequence_number 7
  - SequenceHistory array gets shifted as shown
- frerCpsSeqRcvyLostPackets is incremented erroneously four times for four zeros that got shifted out (*if cell positions 4 through 7 were not marked invalid*)
  - It will get miscounted three more times, if for example, next frame received has sequence_number 10
- Cells in grey are valid to be considered but not the ones in clear

```
// 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;
}
```


BROADCOM

One more case – Boundary condition



- First frame received with sequence_number 7 in this example
  - Bit 0 of SequenceHistory is set, bit 0 represents sequence_number 7
  - All the other entries in this case are valid to be evaluated for missing frames
- This applies to any other frame received with sequence_number >= frerSeqRcyHistoryLength -1 first up

BROADCOM

Proposed changes

Changes to clause 7.4.3.3 SequenceRecoveryReset

7.4.3.3 SequenceRecoveryReset

SequenceRecoveryReset is called whenever the BEGIN event (item a in 7.4.3.1) or the RECOVERY_TIMEOUT event (item c in 7.4.3.1) occurs. It resets the RecovSeqNum (7.4.3.2.3) and SequenceHistory (7.4.3.2.2) variables to their initial states, increments frerCpsSeqRcvyResets (10.8.9), and sets TakeAny (7.4.3.2.6). Note that RecovSeqNum and SequenceHistory are reset only if the VectorRecoveryAlgorithm (7.4.3.4) is configured.

```
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;
        InvalidHistoryCount = (frerSeqRcvyHistoryLength – 1);

        TakeAny = true;
}
```

- Expand clause *7.4.3.2 Variables for sequence recovery* to add definition of SequenceHistoryInit and InvalidHistoryCount
  - Propose 7.4.3.2.7 for SequenceHistoryInit and 7.4.3.2.8 for InvalidHistoryCount

Proposed changes

Changes to clause 7.4.3.4 VectorRecoveryAlgorithm

```
Void VectorRecoveryAlgorithm () {

. . .

// 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;

        if (RecovSeqNum >= (frerSeqRcvyHistoryLength – 1))  {
                InvalidHistoryCount = 0;
                SequenceHistoryInit = false;
        }
        else
                DecrementInvalidHistoryCount (RecovSeqNum);

        frerCpsSeqRcvyPassedPackets = frerCpsSeqRcvyPassedPackets + 1;
        frerCpSeqRcvyPassedPackets = frerCpSeqRcvyPassedPackets + 1;
        RemainingTicks =
                ((frerSeqRcvyResetMSec * TicksPerSecond) + 999) / 1000;
        PRESENT_DATA;
} else if (delta >= frerSeqRcvyHistoryLength ||

. . .
```

Proposed changes

Changes to clause 7.4.3.6 ShiftSequenceHistory

7.4.3.6 ShiftSequenceHistory

This routine is called by the VectorRecoveryAlgorithm routine (7.4.3.4) to advance the SequenceHistory bit array (7.4.3.2.2) and to count lost packets (frerCpsSeqRcvyLostPackets, 10.8.7). ShiftSequenceHistory takes one parameter, which is the new value for index 0 in the SequenceHistory bit array.

```
void ShiftSequenceHistory (int newZeroValue) {
        int i;

        if (InvalidHistoryCount == 0)
                SequenceHistoryInit = false;
        else
                DecrementInvalidHistoryCount (1);

        if ((0 == SequenceHistory[frerSeqRcvyHistoryLength - 1]) &&
SequenceHistoryInit == false))
                frerCpsSeqRcvyLostPackets = frerCpsSeqRcvyLostPackets + 1;
        for (i = frerSeqRcvyHistoryLength - 1; i != 0; i = i - 1)
                SequenceHistory[i] = SequenceHistory[i - 1];
        SequenceHistory[0] = newZeroValue;
}
```

Proposed changes

New routine DecrementInvalidHistory

```
void DecrementInvalidHistoryCount (int count) {
        int i;
        for (i = count; i != 0; i = i - 1)
                InvalidHistoryCount = (InvalidHistoryCount - 1);
}
```

- Propose to add sub-clause 7.4.3.7

BROADCOM®

Thank You