IEEE 802.1CB-2017 Maintenance Item #378

FRER Sequence History Initialisation Algorithm

Response to Proposal

David McCall

13th March 2025

References

[1] Venkat Arunarthi & Sunil Raj, "IEEE 802.1CB-2017 Maintenance Item #378", Contribution to 802.1 Maintenance Task Group, 10th March 2025

Content

- Desired Behaviour
  - Normal Sequence
  - A Few Missing Initial Packets
  - Many Missing Initial Packets
  - Missing Packets After Some In-Sequence Packets
- Proposed Changes, Some Issues & A Counterproposal

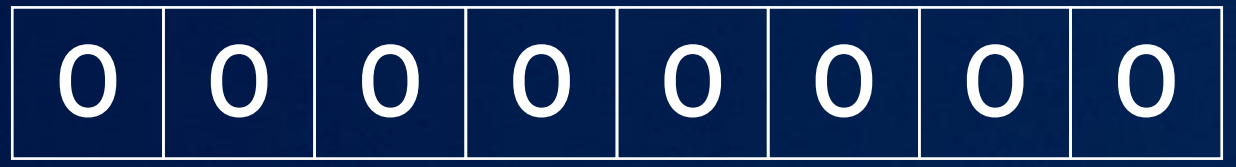
NOTE
If you'd like a copy of this presentation including the animations on slides 5 to 31, please contact David McCall

Variables of Interest

- Existing:
  - sequence_number – sequency number of incoming packet
  - frerSeqRcvyHistoryLength – length of SequenceHistory in bits
  - SequenceHistory – bitmap of recently received packets
  - RecovSeqNum – highest sequence number received
  - frerCpsSeqRcvyLostPackets – count of lost packets (should have been received before falling outside the Sequence History window)
  - TakeAny – TRUE when no SequenceHistory window has been established
- New:
  - SequenceHistoryInit – TRUE if SequenceHistory is fully or partly initialising
  - InvalidHistoryCount – Number of bits in SequenceHistory that are initialising

Desired Behaviour – Normal Sequence

SequenceHistory



frerCpsSeqRcvyLostPackets: 0

RecovSeqNum: 0

TakeAny: TRUE

SequenceHistoryInit: TRUE

InvalidHistoryCount: 7

Desired Behaviour – Normal Sequence

SequenceHistory

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1

frerCpsSeqRcvyLostPackets: 0

RecovSeqNum: 0

TakeAny: FALSE

SequenceHistoryInit: TRUE

InvalidHistoryCount: 7

Desired Behaviour – Normal Sequence

frerCpsSeqRcvyLostPackets: 0

RecovSeqNum: 1

TakeAny: FALSE

SequenceHistoryInit: TRUE

InvalidHistoryCount: 6

Desired Behaviour – Normal Sequence

frerCpsSeqRcvyLostPackets: 0

RecovSeqNum: 2

TakeAny: FALSE

SequenceHistoryInit: TRUE

InvalidHistoryCount: 5

Desired Behaviour – Normal Sequence

SequenceHistory

frerCpsSeqRcvyLostPackets: 0

RecovSeqNum: 3

TakeAny: FALSE

SequenceHistoryInit: TRUE

InvalidHistoryCount: 4

Desired Behaviour – Normal Sequence

SequenceHistory

frerCpsSeqRcvyLostPackets: 0

RecovSeqNum: 4

TakeAny: FALSE

SequenceHistoryInit: TRUE

InvalidHistoryCount: 3

Desired Behaviour – Normal Sequence

frerCpsSeqRcvyLostPackets: 0

RecovSeqNum: 5

TakeAny: FALSE

SequenceHistoryInit: TRUE

InvalidHistoryCount: 2

Desired Behaviour – Normal Sequence

frerCpsSeqRcvyLostPackets: 0

RecovSeqNum: 6

TakeAny: FALSE

SequenceHistoryInit: TRUE

InvalidHistoryCount: 1

Desired Behaviour – Normal Sequence

frerCpsSeqRcvyLostPackets: 0

RecovSeqNum: 7

TakeAny: FALSE

SequenceHistoryInit: FALSE

InvalidHistoryCount: 0

Desired Behaviour – Normal Sequence

frerCpsSeqRcvyLostPackets: 0

RecovSeqNum: 8

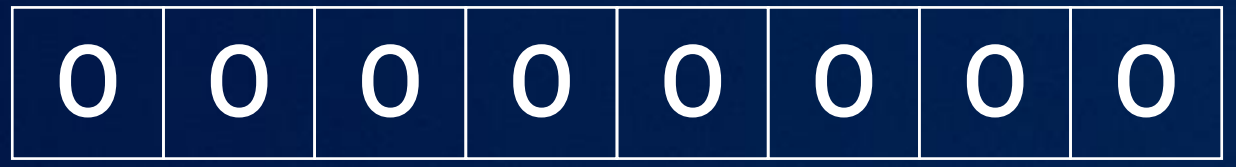
TakeAny: FALSE

SequenceHistoryInit: TRUE

InvalidHistoryCount: 0

Desired Behaviour – A Few Missing Initial Packets

SequenceHistory



frerCpsSeqRcvyLostPackets: 0

RecovSeqNum: 0

TakeAny: TRUE

SequenceHistoryInit: TRUE

InvalidHistoryCount: 7

Desired Behaviour – A Few Missing Initial Packets

frerCpsSeqRcvyLostPackets: 0

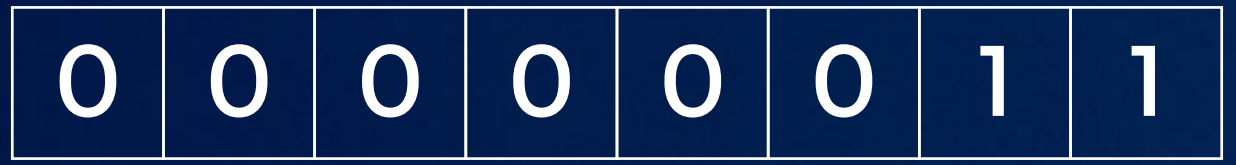
RecovSeqNum: 3

TakeAny: FALSE

SequenceHistoryInit: TRUE

InvalidHistoryCount: 4

Desired Behaviour – A Few Missing Initial Packets



frerCpsSeqRcvyLostPackets: 0

RecovSeqNum: 4

TakeAny: FALSE

SequenceHistoryInit: TRUE

InvalidHistoryCount: 3

Desired Behaviour – A Few Missing Initial Packets

frerCpsSeqRcvyLostPackets: 0

RecovSeqNum: 5

TakeAny: FALSE

SequenceHistoryInit: TRUE

InvalidHistoryCount: 2

Desired Behaviour – A Few Missing Initial Packets

SequenceHistory

0	0	0	0	1	1	1	1
7	6	5	4	3	2	1	0

frerCpsSeqRcvyLostPackets: 0

RecovSeqNum: 6

TakeAny: FALSE

SequenceHistoryInit: TRUE

InvalidHistoryCount: 1

Desired Behaviour – A Few Missing Initial Packets

frerCpsSeqRcvyLostPackets: 0

RecovSeqNum: 7

TakeAny: FALSE

SequenceHistoryInit: FALSE

InvalidHistoryCount: 0

Desired Behaviour – A Few Missing Initial Packets

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

frerCpsSeqRcvyLostPackets: 1

RecovSeqNum: 8

TakeAny: FALSE

SequenceHistoryInit: FALSE

InvalidHistoryCount: 0

Desired Behaviour – Many Missing Initial Packets

SequenceHistory

0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0

frerCpsSeqRcvyLostPackets: 0

RecovSeqNum: 0

TakeAny: TRUE

SequenceHistoryInit: TRUE

InvalidHistoryCount: 7

Desired Behaviour – Many Missing Initial Packets

SequenceHistory

0	0	0	0	0	0	0	1
7	6	5	4	3	2	1	0

frerCpsSeqRcvyLostPackets: 0

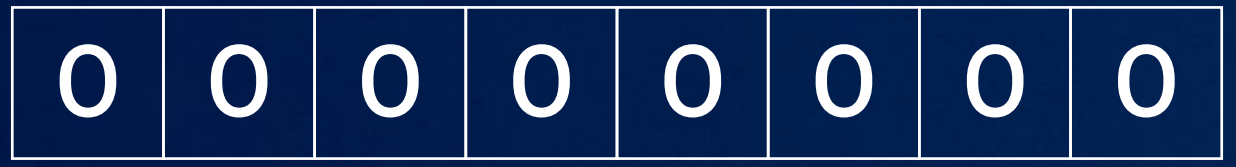
RecovSeqNum: 9

TakeAny: FALSE

SequenceHistoryInit: FALSE

InvalidHistoryCount: 0

Desired Behaviour – Missing Packets After Some In-Sequence Packets



frerCpsSeqRcvyLostPackets: 0

RecovSeqNum: 0

TakeAny: TRUE

SequenceHistoryInit: TRUE

InvalidHistoryCount: 7

Desired Behaviour – Missing Packets After Some In-Sequence Packets

-	-	-	-	-	-	-	0

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

frerCpsSeqRcvyLostPackets: 0

RecovSeqNum: 0

TakeAny: FALSE

SequenceHistoryInit: TRUE

InvalidHistoryCount: 7

Desired Behaviour – Missing Packets After Some In-Sequence Packets

SequenceHistory

0	0	0	0	0	0	1	1
7	6	5	4	3	2	1	0

frerCpsSeqRcvyLostPackets: 0

RecovSeqNum: 1

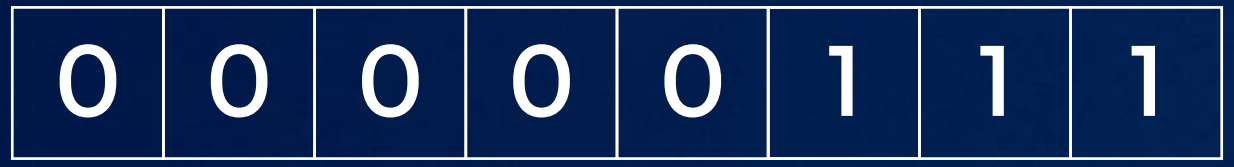
TakeAny: FALSE

SequenceHistoryInit: TRUE

InvalidHistoryCount: 6

Desired Behaviour – Missing Packets After Some In-Sequence Packets

SequenceHistory



frerCpsSeqRcvyLostPackets: 0

RecovSeqNum: 2

TakeAny: FALSE

SequenceHistoryInit: TRUE

InvalidHistoryCount: 5

Desired Behaviour – Missing Packets After Some In-Sequence Packets

frerCpsSeqRcvyLostPackets: 0

RecovSeqNum: 3

TakeAny: FALSE

SequenceHistoryInit: TRUE

InvalidHistoryCount: 4

Desired Behaviour – Missing Packets After Some In-Sequence Packets

SequenceHistory

0	1	1	1	1	0	0	1
7	6	5	4	3	2	1	0

frerCpsSeqRcvyLostPackets: 0

RecovSeqNum: 6

TakeAny: FALSE

SequenceHistoryInit: TRUE

InvalidHistoryCount: 1

Desired Behaviour – Missing Packets After Some In-Sequence Packets

SequenceHistory

1	1	1	1	0	0	1	1
7	6	5	4	3	2	1	0

frerCpsSeqRcvyLostPackets: 0

RecovSeqNum: 7

TakeAny: FALSE

SequenceHistoryInit: FALSE

InvalidHistoryCount: 0

Desired Behaviour – Missing Packets After Some In-Sequence Packets

frerCpsSeqRcvyLostPackets: 0

RecovSeqNum: 8

TakeAny: FALSE

SequenceHistoryInit: FALSE

InvalidHistoryCount: 0

Proposed Change – 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;
        InvalidHistoryCount = (frerSeqRcvyHistoryLength – 1);

        TakeAny = true;
}
```

- Initialises SequencHistoryInit and InvalidHistoryCount correctly.
  - InvalidHistoryCount is sent to frerSeqRcvyHistoryLength-1 to account for arrival of sequence_num of 0
- Looks good!

Proposed Change – 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 ||
```

- Behaviour on arrival of first packet after initialisation
- If sequence_num < length of SequenceHistory, decrement InvalidHistoryCount by the sequence_num
- If sequence_num >= length of SequenceHistory, end the SequenceHistoryInit behaviour
- Looks good!

Also Worth Noting - VectorRecoveryAlgorithm

```
// 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;
```

- If sequence_num isn't the next in order…
  - Shift SequenceHistory, adding UNSEEN bits (0) until it is, then…
- Shift SequenceHistory and add a SEEN bit (1)
  - Set RecovSeqNum to sequence_num

Proposed Change – 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;
}
```

- If InvalidHistoryCount is **already** zero…
  - Set SequenceHistoryInit to FALSE
- Else decrement InvalidHistoryCount by 1
- If the value that's about to be shifted off the end of SequenceHistory is 0 **and** SequenceHistoryInit is FALSE, increment frerCpsSeqRcvyLostPackets by 1
- Shift SequenceHistory by 1 and add the new value to the [0] position
- This…isn't so good

Proposed Change – ShiftSequenceHistory – Issues

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;
}
```

- Uses InvalidHistoryCount as the "gate" for processing SequenceHistoryInit and InvalidHistoryCount
  - Using SequenceHistoryInit would make more sense
- Keeps setting SequenceHistoryInit to FALSE on every call after invalidHistoryCount reaches 0, which will be a vast majority of the time
- Adjusts SequenceHistoryInit or InvalidHistoryCount before processing SequenceHistory, which feels wrong
  - Exits function leaving InvalidHistoryCount at 0 (after decrementing from 1) but SequenceHistoryInit as TRUE, which feels…wronger.

Counterproposal – ShiftSequenceHistory

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

     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;

     if (SequenceHistoryInit)
          DecrementInvalidHistoryCount (1);
          if(InvalidHistoryCount == 0)
               SequenceHistoryInit = false
}
```

- Processes SequenceHistory **then** SequenceHistoryInit and InvalidHistoryCount
- Uses SequenceHistoryInit as the “gate” for processing SequenceHistoryInit and InvalidHistoryCount
- Sets SequenceHistoryInit to FALSE as soon as InvalidHistoryCount reaches 0
- Looks good?

Proposal – DecrementInvalidHistoryCount (New Function)

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

- Decrements InvalidHistoryCount by count, unless count is greater than InvalidHistoryCount, in which case set InvalidHistoryCount to zero
- Which suggests…

Alternative – DecrementInvalidHistoryCount (New Function)

```
void DecrementInvalidHistoryCount (int count) {
    if (count >= InvalidHistoryCount)
        InvalidHistoryCount = 0;
    else
        InvalidHistoryCount = InvalidHistoryCount – count;
}
```

- Same effect, but maybe a bit clearer?
  - Fewer operations, especially if there are big jumps in the sequence_num
- Looks good?


intel