

Pseudocode update in 802.1CB-2017 format
Fejes Ferenc, Balázs Varga – Ericsson
2025-09-14

CBec amendment refines the sequence generation/recovery functions. This contribution contains an initial pseudocode update in 802.1CB-2017 format. Diffs from current .1CB are highlighted with color. **Green** means new added code. **Red** means deleted code.

Note: this pseudocode does not contain all the discussed functional additions. For example it misses, PostResetTimeGuard related code, definition of new variables, etc.

SEQUENCE GENERATION

7.4.1.3 SequenceGenerationReset

```
void SequenceGenerationReset () {
    GenSeqNum = 0;
    InitGenSeqNum = InitGenSeqStart;
    SendResetFlagNum = 3;          // it may be frerSeqRcvyHistoryLength ??
    If (_UseInitSpaceConfigured_)
        UseInitSeqNum = true
    else UseInitSeqNum = false;
    frerCpsSeqGenResets = frerCpsSeqGenResets + 1;
}
```

7.4.1.4 SequenceGenerationAlgorithm

```
void SequenceGenerationAlgorithm () {
    unsigned int sequence_number = GenSeqNum;
    unsigned int init_flag = 0;
    unsigned int reset_flag = 0;
    if (GenSeqNum >= (GenSeqSpace - 1))
    // Using InitSeqNumSpace
    if (UseInitSeqNum && InitGenSeqNum <= (GenSeqSpace - 1)) {
        sequence_number = InitGenSeqNum;
        init_flag = 1;
        InitGenSeqNum = InitGenSeqNum + 1;
        if (SendResetFlagNum > 0) {
            reset_flag = 1;
            SendResetFlagNum = SendResetFlagNum - 1;
        }
    }
    // Using normal SeqNumSpace
    else if (GenSeqNum >= (GenSeqSpace - 1))
        GenSeqNum = 0;
    else
        GenSeqNum = GenSeqNum + 1;
    SEND_DATA; // Pass packet & sequence_number subparameter down stack
}
```

SEQUENCE RECOVERY

7.4.3.3 SequenceRecoveryReset

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

7.4.3.6 ShiftSequenceHistory

```
void ShiftSequenceHistory (int newZeroValue) {
    int i;
if (0 == SequenceHistory[frerSeqRcvyHistoryLength - 1])
    if (SuppressLostUntilPacket) {
        if (1 == SequenceHistory[frerSeqRcvyHistoryLength - 1])
            SuppressLostUntilPacket = false; // a 1 was found in the history
    }
    else 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;
}
```

X.X.X.X InitSequenceRecoveryReset

```
void InitSequenceRecoveryReset () {
    int i;
    InitRecovSeqNum = InitRecovSeqStart - 1;
    for (i = 0; i < frerSeqRcvyHistoryLength; i = i + 1)
        InitSequenceHistory[i] = 0;
    InitTakeAny = true;
    InitSuppressLostUntilPacket = true;
}
```

X.X.X.X InitShiftSequenceHistory

```
void InitShiftSequenceHistory (int newZeroValue) {
    int i;
    if (InitSuppressLostUntilPacket) {
        if (1 == InitSequenceHistory[frerSeqRcvyHistoryLength - 1])
            InitSuppressLostUntilPacket = false; // a 1 has crossed the window
    }
    else if (0 == InitSequenceHistory[frerSeqRcvyHistoryLength - 1])
        frerCpsSeqRcvyLostPackets = frerCpsSeqRcvyLostPackets + 1;
}
```

```

for (i = frerSeqRcvyHistoryLength - 1; i != 0; i = i - 1)
    InitSequenceHistory[i] = InitSequenceHistory[i - 1];
InitSequenceHistory[0] = newZeroValue;
}

```

X.X.X.X InitVectorRecoveryAlgorithm

```

void InitVectorRecoveryAlgorithm () {
    unsigned int sequence_number;
    // Check that sequence number is present in the packet
    if (sequence_number == frerSeqRcvyInvalidSequenceValue) {
        frerCpsSeqRcvyTaglessPackets = frerCpsSeqRcvyTaglessPackets + 1;
        if (frerSeqRcvyTakeNoSequence) {
            frerCpsSeqRcvyPassedPackets = frerCpsSeqRcvyPassedPackets + 1;
            frerCpSeqRcvyPassedPackets = frerCpSeqRcvyPassedPackets + 1;
            InitRemainingTicks =
                ((frerSeqRcvyResetMSec*TicksPerSecond)+999)/1000;
            PRESENT_DATA;
        } else {
            frerCpsSeqRcvyDiscardedPackets =
                frerCpsSeqRcvyDiscardedPackets + 1;
            frerCpSeqRcvyDiscardPackets = frerCpSeqRcvyDiscardPackets + 1;
        }
        return;
    }
    // After timeout/reset, accept any packet
    if (InitTakeAny) {
        if (sequence_number < InitRecovSeqStart)
        {
            // Packet is out-of-range. Count and discard it.
            frerCpsSeqRcvyRoguePackets = frerCpsSeqRcvyRoguePackets + 1;
            frerCpSeqRcvyDiscardPackets = frerCpSeqRcvyDiscardPackets + 1;
            if (frerSeqRcvyIndividualRecovery)
                InitRemainingTicks =
                    ((frerSeqRcvyResetMSec * TicksPerSecond) + 999) / 1000;
        } else {
            // Packet is accepted due to TakeAny.
            InitTakeAny = false;
            InitSequenceHistory[0] = 1; // Shift, adding a "seen" bit
            InitRecovSeqNum = sequence_number;
            frerCpsSeqRcvyPassedPackets = frerCpsSeqRcvyPassedPackets + 1;
            frerCpSeqRcvyPassedPackets = frerCpSeqRcvyPassedPackets + 1;
            InitRemainingTicks =
                ((frerSeqRcvyResetMSec * TicksPerSecond) + 999) / 1000;
            PRESENT_DATA;
        }
    }
    // Compute signed difference in the linear init space
    int delta = sequence_number - InitRecovSeqNum;
    if (sequence_number < InitRecovSeqStart ||
        (delta > frerSeqRcvyHistoryLength) ||
        (delta <= -frerSeqRcvyHistoryLength) )
    {
        // Packet is out-of-range. Count and discard it.
        frerCpsSeqRcvyRoguePackets = frerCpsSeqRcvyRoguePackets + 1;
        frerCpSeqRcvyDiscardPackets = frerCpSeqRcvyDiscardPackets + 1;
    }
}

```

```

    if (frerSeqRcvyIndividualRecovery)
        InitRemainingTicks =
            ((frerSeqRcvyResetMSec * TicksPerSecond) + 999) / 1000;
}
else if (delta <= 0) {
    if (0 == InitSequenceHistory[-delta]) {
        // Packet has not been seen. Take it.
        InitSequenceHistory[-delta] = 1;
        frerCpsSeqRcvyOutOfOrderPackets =
            frerCpsSeqRcvyOutOfOrderPackets + 1;
        frerCpsSeqRcvyPassedPackets = frerCpsSeqRcvyPassedPackets + 1;
        frerCpSeqRcvyPassedPackets = frerCpSeqRcvyPassedPackets + 1;
        InitRemainingTicks =
            ((frerSeqRcvyResetMSec * TicksPerSecond) + 999) / 1000;
        PRESENT_DATA;
    } else {
        // Packet has been seen. Do not forward. Count the discard.
        frerCpsSeqRcvyDiscardedPackets =
            frerCpsSeqRcvyDiscardedPackets + 1;
        frerCpSeqRcvyDiscardPackets = frerCpSeqRcvyDiscardPackets + 1;
        if (frerSeqRcvyIndividualRecovery)
            InitRemainingTicks =
                ((frerSeqRcvyResetMSec * TicksPerSecond) + 999) / 1000;
    }
}
else {
    // Packet is not too far ahead of the one we want.
    if (delta != 1)
        frerCpsSeqRcvyOutOfOrderPackets =
            frerCpsSeqRcvyOutOfOrderPackets + 1;
    // Shift the history until bit 0 refers to sequence_number.
    while (0 != (delta = delta - 1))
        InitShiftSequenceHistory(0); // Shift, adding a "not seen" bit
    InitShiftSequenceHistory(1); // Shift, adding a "seen" bit
    InitRecovSeqNum = sequence_number;
    frerCpsSeqRcvyPassedPackets = frerCpsSeqRcvyPassedPackets + 1;
    frerCpSeqRcvyPassedPackets = frerCpSeqRcvyPassedPackets + 1;
    InitRemainingTicks =
        ((frerSeqRcvyResetMSec * TicksPerSecond) + 999) / 1000;
    PRESENT_DATA;
}
}
}

```

TIMEOUT RELATED CHANGES

7.4.3.1 Events for sequence recovery

...

c) RECOVERY_TIMEOUT: The event that occurs when the (Init)RemainingTicks (7.4.3.2.4, X.X.X.X) variable reaches 0. It triggers the execution of the ~~SequenceRecoveryReset~~ (Init)SetVariablesDueTimeout routine (7.4.3.3 X.X.X.X).

X.X.X.X SetVariablesDueTimeout

```
void SetVariablesDueTimeout () {
    if (frerSeqRcvyAlgorithm == Vector_Alg) {
        int i;
        for (i = 0; i < frerSeqRcvyHistoryLength; i = i + 1) {
            if (SequenceHistory[i] == 0)
                frerCpsSeqRcvyLostPackets = frerCpsSeqRcvyLostPackets + 1;
            SequenceHistory[i] = 0; // Set all bits 0 (packet not seen)
        }
    }
    TakeAny = true;
    SuppressLostUntilPacket = true;
}
```

X.X.X.X InitSetVariablesDueTimeout

```
void InitSetVariablesDueTimeout () {
    if (frerSeqRcvyAlgorithm == Vector_Alg) {
        int i;
        for (i = 0; i < frerSeqRcvyHistoryLength; i = i + 1) {
            if (InitSequenceHistory[i] == 0)
                frerCpsSeqRcvyLostPackets = frerCpsSeqRcvyLostPackets + 1;
            InitSequenceHistory[i] = 0; // Set all bits 0 (packet not seen)
        }
    }
    InitTakeAny = true;
    InitSuppressLostUntilPacket = true;
}
```