YANG module
ieee802-dot1dj-tsn-config-uni

Rodrigo Ferreira Coelho [Siemens AG]

Contributors:
Günter Steindl [Siemens AG]
Motivation

Multivendor interoperability, i.e. behavior of UNI and its exchanged data, is defined by the YANG module `ieee802-dot1dj-tsn-config-uni`.

This contribution details multiple ballot comments and their respective proposed changes regarding the YANG module `ieee802-dot1dj-tsn-config-uni`.
Agenda

• Ensuring assignment of unique StreamIDs and unique stream dst-MAC address
• ieee802-dot1dj-tns-config-uni YANG module
  • Access of CUC to containers via RPCs
    • Account for unique StreamID assignment
    • Account for unique dst-MAC address assignment
  • Multiple streams per request
  • AddStreams with vs. without scheduling semantics
    • AddStreams, ScheduleStreams, AddAndScheduleStreams
  • JoinStream (↔ AddListeners)
  • RemoveStreams, RemoveListeners
Ensuring assignment of unique StreamIDs in a TSN Domain
According to IEC/IEEE 60802, three operation modes possible:

1. **Offline engineered**
   - StreamIDs assigned offline, no further change allowed

2. **Dynamic topology**
   - StreamIDs assigned online
   - Requests sent per CUCs (CUCs independent of each other)

3. **Offline engineered & dynamic topology**
   - Some StreamIDs assigned offline, some requested by CUCs (CUCs independent of each other)
Current Qcc/ Qdj approach to assign StreamIDs

Qcc/ Qdj approach
- StreamIDs assigned by CUCs when requesting a stream

- **MacAddress**
- **Talker** MAC address
- Pool of MAC addresses managed by CUC
- **UniqueId**
- To distinguish streams within end station (i.e. same MAC address)
Issues with current Qcc/ Qdj approach for IEC/IEEE 60802 operation modes

Use case a)

In operation mode 3: **Offline & dynamic topology**

1. How to ensure MACAddr used by engineering tool will not be reused by other CUCs (example CUC1)?

   1. Vendor independent

Table 46-1—StreamID elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>MacAddress</td>
<td>mac-address-type</td>
<td>46.2.3.1.1</td>
</tr>
<tr>
<td>UniqueID</td>
<td>uint16</td>
<td>46.2.3.1.2</td>
</tr>
</tbody>
</table>
Issues with current Qcc/ Qdj approach for IEC/IEEE 60802 operation modes

Use case b)
In operation mode 2, 3: (Offline &) Dynamic topology

Example:
- Talker MAC address used in StreamID
- 1 stream requested by CUC1. Talker := Device101

<table>
<thead>
<tr>
<th>StreamID</th>
<th>Talker(Dev)</th>
<th>Talker(app)</th>
<th>CUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>MACDev101:01</td>
<td>Dev101</td>
<td>Tapp101_1</td>
<td>CUC1</td>
</tr>
</tbody>
</table>

Table 46-1—StreamID elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>MacAddress</td>
<td>mac-address-type</td>
<td>46.2.3.1.1</td>
</tr>
<tr>
<td>UniqueID</td>
<td>uint16</td>
<td>46.2.3.1.2</td>
</tr>
</tbody>
</table>
Issues with current Qcc/ Qdj approach for IEC/IEEE 60802 operation modes

Use case b)
In operation mode 2, 3: (Offline &) Dynamic topology
Example:
- Talker MAC address used in StreamID
- 1 stream requested by CUC1. Talker := Device101
- 1 stream requested by CUC2. Talker := Device101

<table>
<thead>
<tr>
<th>StreamID</th>
<th>Talker(Dev)</th>
<th>Talker(app)</th>
<th>CUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>MACDev101:01</td>
<td>Dev101</td>
<td>Tapp101_1</td>
<td>CUC1</td>
</tr>
<tr>
<td>MACDev101:01</td>
<td>Dev101</td>
<td>Tapp101_2</td>
<td>CUC2</td>
</tr>
</tbody>
</table>

Duplicated StreamID assignment
Issues with current Qcc/ Qdj approach for IEC/IEEE 60802 operation modes

Use case c)
In operation mode 2, 3: (Offline &) Dynamic topology
Example:
- CUC pool MAC address used in StreamID
  1. Stream 1 requested by CUC1. TalkerApp := Tapp101_1

<table>
<thead>
<tr>
<th>StreamID</th>
<th>CUC(Dev)</th>
<th>CUC(app)</th>
<th>Talker(app)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MACDev1:01</td>
<td>Dev1</td>
<td>CUC1</td>
<td>Tapp101_1</td>
</tr>
</tbody>
</table>

Table 46-1—StreamID elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>MacAddress</td>
<td>mac-address-type</td>
<td>46.2.3.1.1</td>
</tr>
<tr>
<td>UniqueID</td>
<td>uint16</td>
<td>46.2.3.1.2</td>
</tr>
</tbody>
</table>
Issues with current Qcc/ Qdj approach for IEC/IEEE 60802 operation modes

Use case c)
In operation mode 2, 3: *(Offline &)* Dynamic topology

Example:

- **CUC pool MAC address** used in StreamID
  1. Stream 1 requested by CUC1. TalkerApp := Tapp101_1
  2. Device 2 replaces Device 1, i.e. Dev2 hosts CUC1

<table>
<thead>
<tr>
<th>StreamID</th>
<th>CUC(Dev)</th>
<th>CUC(app)</th>
<th>Talker(app)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MACDev1:01</td>
<td>Dev1</td>
<td>CUC1</td>
<td>Tapp101_1</td>
</tr>
</tbody>
</table>

Table 46-1—StreamID elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>MacAddress</td>
<td>mac-address-type</td>
<td>46.2.3.1.1</td>
</tr>
<tr>
<td>UniqueID</td>
<td>uint16</td>
<td>46.2.3.1.2</td>
</tr>
</tbody>
</table>
Issues with current Qcc/ Qdj approach for IEC/IEEE 60802 operation modes

Use case c)
In operation mode 2, 3:(Offline &) Dynamic topology
Example:

- **CUC pool MAC address** used in StreamID
  1. Stream 1 requested by CUC1. TalkerApp := Tapp101_1
  2. Device 2 replaces Device 1, i.e. Dev2 hosts CUC1
  3. Stream 2 requested by CUC1. TalkerApp := Tapp101_1

<table>
<thead>
<tr>
<th>StreamID</th>
<th>CUC(Dev)</th>
<th>CUC(app)</th>
<th>Talker(app)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MACDev1:01</td>
<td>Dev1</td>
<td>CUC1</td>
<td>Tapp101_1</td>
</tr>
<tr>
<td>MACDev2:02</td>
<td>Dev2</td>
<td>CUC1</td>
<td>Tapp101_1</td>
</tr>
</tbody>
</table>
Issues with current Qcc/Qdj approach for IEC/IEEE 60802 operation modes

Use case c)

In operation mode 2, 3: (Offline &) Dynamic topology

Example:

- **CUC pool MAC address** used in StreamID
- **1.** Stream 1 requested by CUC1. TalkerApp := Tapp101_1
- **2.** Device 2 replaces Device 1, i.e. Dev2 hosts CUC1
- **3.** Stream 2 requested by CUC1. TalkerApp := Tapp101_1
- **4.** Device 1 comes back to host CUC2

<table>
<thead>
<tr>
<th>StreamID</th>
<th>CUC(Dev)</th>
<th>CUC(app)</th>
<th>Talker(app)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MACDev1:01</td>
<td>Dev1</td>
<td>CUC1</td>
<td>Tapp101_1</td>
</tr>
<tr>
<td>MACDev2:02</td>
<td>Dev2</td>
<td>CUC1</td>
<td>Tapp101_1</td>
</tr>
</tbody>
</table>

Table 46-1—StreamID elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>MacAddress</td>
<td>mac-address-type</td>
<td>46.2.3.1.1</td>
</tr>
<tr>
<td>UniqueID</td>
<td>uint16</td>
<td>46.2.3.1.2</td>
</tr>
</tbody>
</table>
Issues with current Qcc/ Qdj approach for IEC/IEEE 60802 operation modes

Use case c)
In operation mode 2, 3: (Offline &) Dynamic topology
Example:

- CUC pool MAC address used in StreamID
1. Stream 1 requested by CUC1. TalkerApp := Tapp101_1
2. Device 2 replaces Device 1, i.e. Dev2 hosts CUC1
3. Stream 2 requested by CUC1. TalkerApp := Tapp101_1
4. Device 1 comes back to host CUC2
5. Stream 2 requested by CUC2. TalkerApp := Tapp101_2

<table>
<thead>
<tr>
<th>StreamID</th>
<th>CUC(Dev)</th>
<th>CUC(app)</th>
<th>Talker(app)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MACDev1:01</td>
<td>Dev1</td>
<td>CUC1</td>
<td>Tapp101_1</td>
</tr>
<tr>
<td>MACDev2:02</td>
<td>Dev2</td>
<td>CUC1</td>
<td>Tapp101_1</td>
</tr>
<tr>
<td>MACDev1:01</td>
<td>Dev1</td>
<td>CUC2</td>
<td>Tapp101_2</td>
</tr>
</tbody>
</table>
Issues with current Qcc/Qdj approach for IEC/IEEE 60802 operation modes

Use case c)
In operation mode 2, 3: (Offline &) Dynamic topology
Example:
- CUC pool MAC address used in StreamID
  1. Stream 1 requested by CUC1. TalkerApp := Tapp101_1
  2. Device 2 replaces Device 1, i.e. Dev2 hosts CUC1
  3. Stream 2 requested by CUC1. TalkerApp := Tapp101_1
  4. Device 1 comes back to host CUC2
  5. Stream 2 requested by CUC2. TalkerApp := Tapp101_2

There are possibly more use cases where this problem occurs

<table>
<thead>
<tr>
<th>StreamID</th>
<th>CUC(Dev)</th>
<th>CUC(app)</th>
<th>Talker(app)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MACDev1:01</td>
<td>Dev1</td>
<td>CUC1</td>
<td>Tapp101_1</td>
</tr>
<tr>
<td>MACDev2:02</td>
<td>Dev2</td>
<td>CUC1</td>
<td>Tapp101_1</td>
</tr>
<tr>
<td>MACDev1:01</td>
<td>Dev1</td>
<td>CUC2</td>
<td>Tapp101_2</td>
</tr>
</tbody>
</table>
How to ensure unique StreamID assignment in a TSN Domain?

StreamIDs must be centrally managed, i.e. assigned by the CNC
Ensuring assignment of unique stream dst-MAC address to streams in a TSN Domain
Motivation: Operation modes in IEC/IEEE 60802

Similar to StreamID assignment, local management of stream dst-MAC, i.e. by CUCs, cannot ensure uniqueness.

Stream dst-MAC addresses must be centrally managed, i.e. assigned by the CNC.
Requesting a stream using ieee802-dot1dj-tns-config-uni without pre-defining a StreamID? (i.e. CNC assigns StreamIDs)
IEEE802-Dot1D-Tsn-Config-Uni

- YANG Containers
- YANG Remote Procedure Calls (RPCs)
- YANG Notifications
- Not addressed in this contribution
YANG Containers & RPCs

A hierarchical data model would fit better for the required access control.
- Domain[1..n]
- Client[1..k] -> NACM (write access limited to “this” client)
- Stream [1...l]
- Root access to the stream list only via RPC

Access to stream list only via YANG RPCs
- Facilitates Network Configuration Access Control Model (NACM)
- Allows for decoupling stream request parameters from datastore structure whenever needed

Extracted from 60802-Stamenic-NETCONF-Simultaneous-Requests-12-21-v01.pdf

ieee802-dot1dj-tns-config-uni: graphical representation of root container
Stream requests: RPC AddStreams

- **Description**
  - Adds multiple streams to CNC datastore
  - Assigns StreamIDs to streams
  - Input (list of (StreamID, request container)) where StreamID := unknown (optional?)
  - Output (list of (StreamID)) where StreamID assigned by CNC

- CUC calls RPC
- RPC notifies CNC of new AddStream request
- CNC assigns StreamIDs to these streams
- RPC stores stream with StreamIDs in YANG container
- RPC responses with list of assigned StreamIDs sent to CUC
Stream requests: RPC ScheduleStreams

- **Description**
  - Schedules/plans multiple streams
- **Input** (list of (StreamID))
- **Output** (stream list)
  - Modified configuration containers after scheduling/planning requested streams
  - **Stream dst-MAC addr** assigned by CNC (configuration container)
  - Configuration information (relevant to CUC and talker) of streams impacted by requested streams

**Scheduling multiple streams in one request is an IA requirement (bidirectional communication)**
Stream requests: RPC ScheduleStreams

- ... 
- **Stream dst-MAC addr** assigned by CNC (configuration container) 
- ... 

This restriction must be removed from the standard

---

15 **46.2.5.3.1 IEEE802-MacAddresses**

The IEEE802-MacAddresses group provides the source and destination MAC addresses that apply to the 17 network side of the user/network boundary.

**NOTE 1**—On the user side, the MAC addresses are in DataFrameSpecification.IEEE802-MacAddresses.

**NOTE 2**—The source MAC address of the network is typically the same as the user. The destination MAC address can 19 be different. For example, the user can use an individual address, but the network can use a group (multicast) address.

20 This group uses the specifications from DataFrameSpecification.IEEE802-MacAddresses (46.2.3.4.1).

21 This configuration value is not provided unless IEEE Std 802.1CB is supported and a value for Active 22 Destination MAC and VLAN Stream identification is provided in CB-StreamIdenTypeList of 23 InterfaceCapabilities.
Stream requests: RPC AddAndScheduleStreams

- **Description**
  - Adds multiple streams to CNC datastore
  - Assigns StreamIDs to streams
  - Schedules multiple streams
  - Input (list of (StreamID, request container)) where StreamID := unknown (optional?)
  - Output (stream list)
    - Modified configuration containers after scheduling/planning requested streams
    - StreamID assigned by CNC
    - Stream dst-MAC addr assigned by CNC
    - Configuration information (relevant to CUC and talker) of streams impacted by requested streams

Scheduling multiple streams in one request is an IA requirement (bidirectional communication)
Further `ieee802-dot1dj-tns-config-uni`

RPCs for stream management
Add listeners to stream: RPC JoinStream ↔ AddAndScheduleListeners

- Description
  - Adds listeners to a stream into CNC datastore
  - Schedules listeners of stream
  - Input (list of (Listener))
  - Output (stream list)
  - Modified configuration containers after scheduling/planning requested streams
  - Configuration information (relevant to CUC and talker) of streams impacted by requested streams
Remove Streams: RPC RemoveStreams

• Description
  • Removes streams from CNC datastore
• Input (list of (StreamID))
• Output (stream list)
  • Modified configuration containers after removing requested streams
  • Configuration information (relevant to CUC and Talker) of streams impacted by requested streams
Remove listeners of a stream

- **Description**
  - Removes listeners of a stream
- **Input** (StreamID, list of (Listener))
- **Output** (stream list)
  - Modified configuration containers after removing listeners of requested streams
  - Configuration information (relevant to CUC, Talker, and Listeners) of streams impacted by requested streams
Summary

- **StreamIDs** and **Stream dst-MAC addresses** managed by the CNC

- Stream requests via YANG RPCs
  - Requesting **multiple streams in one request** is an IA requirement
  - Add & Schedule
    - Separated
    - In a single request
  - AddListeners, RemoveListeners, RemoveStream

Next steps
- **YANG module refinement and textual contribution**
Further questions?
Contact

Dr. Rodrigo Ferreira Coelho
System Architect
DI FA CTR ICO ARC
Siemenspromenade 1
91058 Erlangen
Deutschland

Phone: +49 9131 17-45546
E-mail: rodrigo.ferreira_coelho@siemens.com