Time Sensitive Control Streams in IEEE P1722A v1.3

info: IEEE P1722A-tscs
Revision: 1.3
Date: 2011-07-11
Author: Jeff Koftinoff <jeff.koftinoff@ieee.org>

Table of Contents

1 Overview

While IEEE Std. 1722-2011 provides for time sensitive media streams, time sensitive control streams are necessary for some use cases of control networks.

Compelling reasons to use AVB time sensitive streams for control protocols include:

1. The control streams are given guaranteed bandwidth
2. The control messages can be multicast to multiple end stations
3. Multiple control streams may be time division multiplexed in one stream, evenly sharing an SRP bandwidth reservation
4. The control messages can have AVB style presentation times to allow synchronized control updates in distributed systems which share common logical controls.

IEEE Std. 1722-2011 allows for iec61883-6 AM824 format time sensitive streams which has provision for transport of time division multiplexed MIDI messages which may contain control messages and MIDI System Exclusive messages. While this combination of packet encodings can be used for the transport of time sensitive control streams, a more efficient and flexible time sensitive control stream format is proposed here.

These time sensitive control streams may be used for different packet oriented or message oriented protocols, such as:

- Open Sound Control
- CAN-bus
- IEEE P1722.1
- HTTP
- Industrial control protocols

A TSCS may be used as a response path for the above protocols when necessary.

2 Requirements

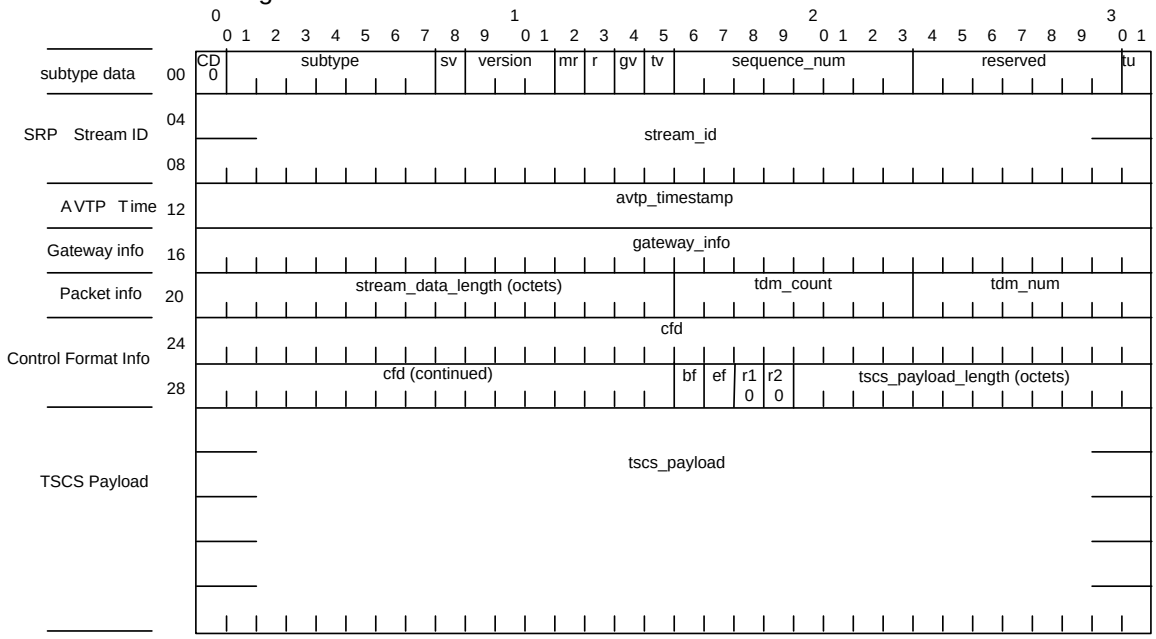
The packet format for time sensitive control streams must be designed to allow the streams to contain control packets or messages which:

- Are limited in their maximum bandwidth usage
- Can span stream packets
- Can be much larger than a standard MTU
- Can be framed to allow receiver resynchronization when a device joins mid-stream or a stream packet is lost
- Can be associated with a 802.1AS-2011 gPTP presentation time
- Allow for Time Division Multiplexing (TDM) of multiple separate control packets and messages over the same time sensitive stream
- Can be used to tunnel existing TCP/IP stream or UDP packet based protocols (in one direction per stream)
- Can be used to tunnel CAN-bus protocols
- Can be used to tunnel IEEE Std. 1722-2011 control protocols such as IEEE P1722.1 verbatim.
- Can be used to transport vendor specific stream or packetized protocols.

- Can be sent more than once per SRP's measurement interval (125µs for class A or 250µs for class B) as the SRP's bandwidth allocation allows.

3 Packet Format

A new subtype would be allocated in IEEE P1722A for time sensitive control streams (TSCS). A TSCS's packet header has the following form:



–

The AVTPDU-TSCS packet header contains the following AVTP common stream data fields (from IEEE Std. 1722-2011 Section 5.4):

- **cd** (control/data) indicator: 1 bit
- **subtype**: 7 bits
- **sv** (StreamID valid) indicator: 1 bit
- **version** (AVTP version): 3 bits
- **mr** (media clock restart): 1 bit
- **r** (reserved): 1 bit
- **tv** (avtp_timestamp_valid): 1 bit
- **sequence_num** (sequence number): 8 bits
- **reserved**: 7 bits
- **tu** (timestamp_uncertain): 1 bit
- **avtp_timestamp**: 32 bits
- **gateway_info**: 32 bits
- **stream_data_length**: 16 bits

The TSCS packet adds the following additional fields:

- **tdm_count** (control stream TDM slots count): 8 bits
- **tdm_num** (control stream TDM slot number): 8 bits
- **cfd** (control format descriptor): 48 bits EUI-48

- **bf** (begin frame flag): 1 bit
- **ef** (end frame flag): 1 bit
- **r1** (reserved1) : 1 bit
- **r2** (reserved2) : 1 bit
- **tscs_payload_length** (time sensitive control stream payload length in octets): 12 bits
- **tscs_payload** (time sensitive control stream payload): data length is determined by **tsc_payload_length**, frame length is determined by **stream_data_length**.

3.1 tv field

When the **tv** field is 1, this means that the **avtp_timestamp** field is valid and represents the presentation time of the first data byte in the **tscs_payload** field.

3.2 cfd field

The **cfd** field is a 48 bit field containing an EUI-48 value, the Control Format Descriptor (CFD).

If the three most significant octets of the **cfd** field is the IEEE Std. 1722-2011 assigned OUI, `90-e0-f0` or `91-e0-f0`, then the **cfd** field specifies a standard CFD, otherwise the **cfd** field specifies a vendor specific control protocol and the CFD is prefixed by the vendor's OUI24 or OUI36.

The least significant bit of the first octet of the **cfd** field is normally reserved in MAC-48 address for the unicast/multicast bit. In the CFD, however it is used as the **packetized** field, regardless if the CFD is a standard or vendor specific CFD.

3.2.1 packetized CFD field

The **packetized** field in the CFD represents the transport style of the time sensitive control data. The **packetized** field may be:

- 0 (zero): control stream
- 1 (one): control packets

3.2.1.1 CFD for control stream

When the **packetized** bit of the CFD is 0, the **tscs_payload** is an undelimited stream of data, analogous to a TCP/IP stream (without acknowledgement or automatic resends) or a unidirectional serial data port. The **bf** and **ef** bits are to be unused and set to 0.

3.2.1.2 CFD for control packets

When the **packetized** bit is 1, the **tscs_payload** is a delimited stream of data, analogous to a UDP packet transport mechanism. The **bf** and **ef** bits are used for delimiting packets. The delimited packets may span multiple TSCS packets and may be larger than one MTU.

3.2.2 Standard cfd's

When the **cfd** field is in the form `90-e0-f0-XX-YY-ZZ` or `91-e0-f0-XX-YY-ZZ`, then the value of `XX` can be one of:

XX code	Meaning
00_{16}	YY represents IEEE Std. 1722-2011 **subtype** field. ZZ represents protocol revision.
01_{16}	YY-ZZ doublet represents protocol defined by IANA "DCCP Well Known Ports" or "DCCP Registered Ports"

02_{16}	CANbus protocol, YY-ZZ represents CANbus protocol style/revision <<TBD>>
$03 - FF_{16}$	Reserved

3.3 tdm_count field

The 8 bit **tdm_count** field specifies the number of TDM channels in this stream.

A **tdm_count** value of 0 means 256 protocol channels are time division multiplexed within this stream.

3.4 tdm field

The 8 bit **tdm** field specifies the control packet TDM channel that this packet's **tscs_payload** is associated with. The **tdm** field shall always be less than the value of the **tdm_count** field.

The AVTP talker may transmit TSCS packets with the **tdm** field changing in any pattern required for the shared bandwidth requirements of the various embedded control channels.

For instance, it may evenly share the stream's bandwidth amonst each embedded TDM channel by incrementing the **tdm** field every packet (and setting it back to 0 when it hits the **tdm_count** value).

Or it may perform any arbitrary pattern that could allow some TDM channels more bandwidth.

3.5 bf field

When the 1 bit **bf** is set it means that the first octet in this payload represents the first octet of the control stream packet.

The **bf** field is only used when the **cfd** field represents a **packetized** cfd form.

3.6 ef field

When the 1 bit **ef** is set it means that the last octet in this payload represents the last octet of the control stream packet.

The **ef** field is only used when the **cfd** field represents a **packetized** cfd form.

3.7 tscs_payload_length

The **tscs_payload_length** field is a 12 bit value and specifies the length of the **tscs_payload** area of the stream packet. The **tscs_payload_length** field may be 0, indicating no data content in the **tscs_payload** area. The **tscs_payload** field may end before the end of the stream packet - this allows for fixed sized packets and bandwidth usage regardless of control stream content.

3.8 tscs_payload

The **tscs_payload** area may be 0 to 2048 octets in length but the entire frame is limited to the network's MTU.

4 Open Items for Discussion

4.1 Examples of 1722.1 in TSCS

We need some examples of how IEEE P1722.1 packets could be transported via a TSCS.

4.2 Examples of TCP and UDP transport

<<TBD>>

4.3 Examples of TDM capabilities

<<TBD>>

4.4 CAN-bus payload structure

As discussed on http://www.can-wiki.info/CanInterfaceAPI it may be relevant to define a specific CAN-bus payload structure containing the following items:

```
/**
 * The CAN message structure.
 * Used for all data transfers between the application and the driver
 * using read() or write().
 */
 typedef struct {
    /** flags, indicating or controlling special message properties */
    int             flags;       /**< Bit 0: Standard (11) or extended (29) length message */
                                 /**  Bit 1: Standard or RTR message */
    int             cob;         /**< Bit 0..3 CAN object number, used in Full CAN  */
                                 /**  Bit 4..7 CAN controller number, used if multiple buses present */
                                 /**    0 could mean 'default channel' in both cases */
    unsigned   long id;          /**< CAN message ID, 4 bytes  */
    struct timeval  timestamp;   /**< time stamp for received messages */
    short      int  length;      /**< number of bytes in the CAN message */
    unsigned   char data[8];     /**< 0 to 8 bytes of message data */
 } canmsg_t;
```

Without the timestamp since avtp_timestamp is already in the AVTPDU

Input is needed from the CAN-bus experts on the best way to tunnel CAN-bus via TSCS.