IEEE P1722.1

Additional Feature Considerations

IEEE P1722.1 F2F - Detroit - Oct 21, 2014

Jeff Koftinoff
jeff.koftinoff@gmail.com

ACMP Scalability

- ACMP is always Multicast in IEEE 1722.1-2013
- Connecting a stream involves 4 multicast messages: (CONNECT_RX_COMMAND, CONNECT_TX_COMMAND, CONNECT_TX_RESPONSE, CONNECT_RX_RESPONSE)
- Checking connection status of a stream involves at least 6 multicast messages, depending on number of listeners of the stream: GET_RX_STATE_COMMAND, GET_RX_STATE_RESPONSE, GET_TX_STATE_COMMAND, GET_TX_STATE_RESPONSE, and a GET_TX_CONNECTION_COMMAND and GET_TX_CONNECTION_RESPONSE for each connection that that Talker has until the appropriate listener is found.

ACMP Scalability Example

- A controller powering up and needing to enumerate the active connections on a network with 150 talker stream sources connected to 200 listener stream sinks would require approximately:

  200 * 6 = 1,200 multicast messages

- All end stations are would be required to receive and parse each ACMP message even if it is not involved in the transaction. The burst of multicast messages on the network can be a real problem for low cost end stations, especially if the controller is on a gigabit network and the end stations is on Fast Ethernet. Without an FPGA there is no hardware assistance for filtering the undesired ACMP messages

ACMP Scalability Options

1. Allow ACMP commands to be also sent as Unicast messages from the Controller to the Talker and Listener Entities, with the Entities responding via Unicast ACMP responses. This would also allow a “Controller-Driven” connection mechanism which would be useful for connecting IP Talkers with IP Listeners across subnets. This option would be backwards compatible with IEEE 1722.1-2013. Unicast to Unicast - Multicast to Multicast.

2. IEEE P1722.1A can add AECP commands and responses to collect ACMP Connection state data

3. IEEE P1722.1A can add new counters for ENTITY, STREAM_INPUT, and STREAM_OUTPUT descriptors for counts of ACMP connection and disconnection events so that a controller can quickly know if any streams connections have changed since the last time it received the connection list

ACMP FAST_CONNECT

- In IEEE 1722.1-2013 Clause 8.2.2.1.1, ACMP FAST_CONNECT is a mode determined solely by the listener and the mode can not be queried or set or disabled by a Controller.

- Add a FAST_CONNECT capability for ADP's entity_capabilities or listener_capabilities fields to allow a controller to know which entities can perform fast connect

- IEEE P1722.1A can add a FAST_CONNECT_DISABLE flag for ACMP's flags field in the ACMP_CONNECT_RX command to allow a Controller to connect a stream without fast connect

ACMP FAST_RECONNECT

- In IEEE 1722.1-2013 Clause 8.2.2.1.1, ACMP FAST_CONNECT mode is defined for power up behaviour of the Listener
- If the Talker of a stream is rebooted, the stream will be removed and the Talker and the Listener will not re-connect the stream
- A typical user story expects the stream to be reconnected always, without involvement of a Controller
- In order to do this, the Listener needs to be able to trigger ACMP connections
- IEEE P1722.1A can add FAST_RECONNECT as an entity capability and CONNECT_RX_COMMAND flag to allow the Controller to control the Listener's behaviour

Managing SRP Parameters

- SRP's "Leave Timer" and "Leave All" Timer settings are directly related to management CPU overhead and network scalability of number of Talker and Listener attributes possible
- There is no way in the SRP protocol to manually manage these timers
- When timer values are changed for SRP, all devices on the network must have the same timer settings
- Enterprise-grade switches allow their own management of SRP timers and parameters
- IEEE P1722.1A can add AECP AEM control types to represent the timers on a specific AVB_INTERFACE:
    - SRP_LEAVE_TIMER
    - SRP_LEAVE_ALL_TIMER
    - SRP_CLASS_INFO (Traffic Class,VLAN ID,PCP,Observation Interval,Shaper Type,etc.)

AVB_INTERFACE Counters

- A Controller checking status of a Talker or Listener currently needs to use GET_STREAM_INFO and GET_COUNTERS for each talker stream source and listener stream sink to get stream health status

- IEEE P1722.1A can add new counters to AVB_INTERFACE to show counts of:

  - Talker ACMP connection changes
  - Listener ACMP connection changes
  - SRP Talker Advertise transitions to or from Talker Failed
  - SRP Listener Ready transitions to Listener Asking Failed

- Doing this would allow a Controller to quickly check for the normal case where there are no fault conditions

Counter History

- A user connecting to a running system often needs to know not only that some diagnostics counters changed but when they last changed

- IEEE P1722.1A can add a command GET_COUNTERS_HISTORY which would return a response in the same form as GET_COUNTERS response but the values in the payload would contain the number of seconds since the last transition of the associated counter

Lock Status

- While the counters for AVB_INTERFACE and CLOCK_DOMAIN allow for counting lock events separately from unlock events, relying on lock event count being equal to unlock event count plus 1 is not always reliable mechanism to report lock status on all hardware
- IEEE P1722.1A can add a GET_CLOCK_STATUS AECP AEM command to report on actual clock lock status and the number of seconds since the last lock state transition for AVB_INTERFACE, CLOCK_SOURCE, and CLOCK_DOMAIN descriptors

IP Streaming

- IEEE P1722 Draft 10 supports transport of stream and control AVTPDU's via IPv4 and IPv6 UDP
- IEEE P1722.1A can add IP_STREAM_INPUT and IP_STREAM_OUTPUT descriptors to allow configuration and connection of IP talker stream sources and IP listener stream sinks
- These descriptors could be optionally defined or referenced in every place a STREAM_INPUT / STREAM_OUTPUT descriptor is referenced
- These descriptors would be identical to the current STREAM_INPUT/STREAM_OUTPUT descriptors except with the addition of the appropriate IP related information.
- Destination IPv6 address and UDP port number and other dynamic information would be managed via new SET_IP_STREAM_INFO/GET_IP_STREAM_INFO commands.
- There is a standard way of representing an IPv4 address with an IPv6 address. See RFC4291 and RFC6890
- Add a entity_capabilties flag to show that the AVDECC entity can send and receive AVDECC PDU's via UDP

DNS-SD

- IEEE Std 1722.1-2013 Annex C defines the usage of the "_avdecc._tcp" DNS-SD service type for the AVDECC Proxy Protocol on TCP port 17221
- We can add definitions for "_avdecc._udp" to allow for DNS-SD publishing of AVDECC entities available via UDP port 17221.