P1722.1 Connection Sequencing Proposal

Apr 6, 2010 Face to Face Meeting
Guy Fedorkow, Adamson Systems
Guy {at} Adamsonsystems.com

This chart outlines how a connection manager would discover and enumerate AVB devices in a network, then set up connections from Talkers to Listeners.

To recap how we get to this chart…

- The starting point is 1722.1-fedorkow-Connection-Sequencing-0210-v1.pdf on the P1722.1 contributions web page.
- We're beginning with no 1722.1 configuration at all in the devices
- mDNS has assigned addresses, DNS-SD has announced capabilities.
- We now want a management station to identify all the devices and makes connections

Listeners and Talkers	Connection Manager
(As the starting point from Section **Error! Reference source not found.**, Listeners do **not** know the name of the channel and stream that they want. The management station has access to the AVB endpoints and figure out what's out there using a "DNS-SD Browser" methodology, with the ultimate goal of telling Listeners to connect to Talkers.	
All Listeners and Talkers submit DNS-SD Text Records as identified in Section **Error! Reference source not found.**	
	Collect a list of the names and descriptions of all AVB-capable end points that are advertising a media type the Manager wants to see (e.g. Audio and/or Video) This allows the Manager to display a list of AVB devices.
	For each service in each endpoint of interest in the DNS-SD records, ask which coding types the endpoint is capable of supporting (e.g. AM824, 48, 96 and 192 khz) [This could be hard to encode, given all the options. And if it's not hard, we should put it in the TXT record!] [This assumes that the [service in the] endpoint itself is capable of applying whatever coding it's capable of doing to any stream, i.e., coding capabilities would be an [service and] endpoint

	capability, not a stream capability. If this is not true, we'll have to report this capability per-stream somehow. This does not imply that all streams to/from the endpoint must actually use the same coding format!] This coding information might include generic limitations such as the number of slots that can be packed into a single stream [depending on how far we want to go with off-line configuration and self-describing devices]
Respond with a list of coding capabilities for the endpoint.	
	[insert steps for creating streams on endpoints here? If we do this, the steps here would set the coding params and channel lists on each Talker stream]
	For each Talker of interest, ask the endpoint to list all the streams it *can* source, and which ones it *is* sourcing. (Do we actually care whether the Talker is actually talking or not? Other than a debug clue, it might not matter.) For each Listener of interest, ask the endpoint to list what stream sinks it has, and which streams it is actively listening to (if any). This implies that the listener responds with two sets of names, the in-built channels it can receive, and the streams and channels it's already been configured for. This request might be limited in scope by using names with wild-cards in the request (e.g. "tell me all streams named "Left*")
Endpoints respond with a list of names of streams [and sinks]. Each Name might be a just the name of a stream if it carries a single channel, or it might be a stream and channel (eg RightStream/SideFill) Talkers respond with the list of names for streams and channels. Listeners respond with the list of preset names of listening ports and with names of talkers to which they're listening (if any)	
	For each Stream of interest, the Connection Manager can ask the endpoint for the StreamInfo for a stream with the name learned in the previous step. [Alternately, it could ask for the StreamInfo for a list of streams with wildcards, combining this step and the previous one]
The Endpoint responds with the info for the	

requested stream(s): Stream Name Stream ID (unique number used in SRP) Stream MAC-DA (mcast dst addr) Stream Coding Info [this is complicated] Number of Channels List of Channel Names (and AM824 tag per channel) (Listeners respond with both talker and listener names if connected) Clocking Source (for Talkers only; do we even want to display this information?) [I'm assuming that all channels share the same coding characteristics; we need to figure out whether that's not always true – AM824 can code MIDI, Audio, etc all in one stream]	
	Compile a table of channel counts, bit rates, coding, etc for all devices. Read a list of desired connections from local storage / GUI / whatever. Match up desired connections with devices and device capabilities to select "the best" combination of parameters for each connection.
	Issue proprietary commands to set gain, DSP params, delay compensation, etc using Protocol XX extensions
Respond to proprietary configuration commands	
	Issue Protocol XX commands to configure names and coding information for desired connections to talkers and listeners. For a Listener, set the params for each connection it should seek to establish: Stream Sink name Talker Name Stream Name Stream ID (unique number used in SRP for two-step) Stream Coding Info Channel Name (or Names?) (MAC-DA is not configured, as it should be learned as in Chart XX above once config is complete)
Respond to connection name configuration commands.	
At this point, each listener can start the connection SRP process on its own.	