1.1 AVB Connection Sequencing

Guy Fedorkow, Adamson Systems, Dec 4, 2009, (with thanks to Kevin G for inspiration)

1.1.1 Use Case

For the pro-audio, live-sound use case, I think there are a couple of goals:

- Assume that the initial desired functionality is a “distributed, peer-to-peer patch panel,” i.e., there’s a list of sources of streaming content, and a list of destinations, and the goal of connection sequencing is to get the right content to the right destinations across named connections.
- It’s important that the system be resilient, and that it recover quickly after failure.

I'm also assuming that a System Controller is desirable for setting up complex systems, but that reliance on a System Controller once configuration is complete is not desirable.

The approach I'm thinking about uses two interlocking phases:

- Part of the procedure discovers and enumerates all the visible AVB components on a network segment, then allows a connection manager to identify and configure talkers and listeners, and to associate them via named connections. A "named connection" might be something like "Right Side Fill", a human-readable name that would be (for example) assigned to an output channel from a mixing desk (the talker) and sent to a group of amplifiers that power side fill speakers on the right side of a stage.

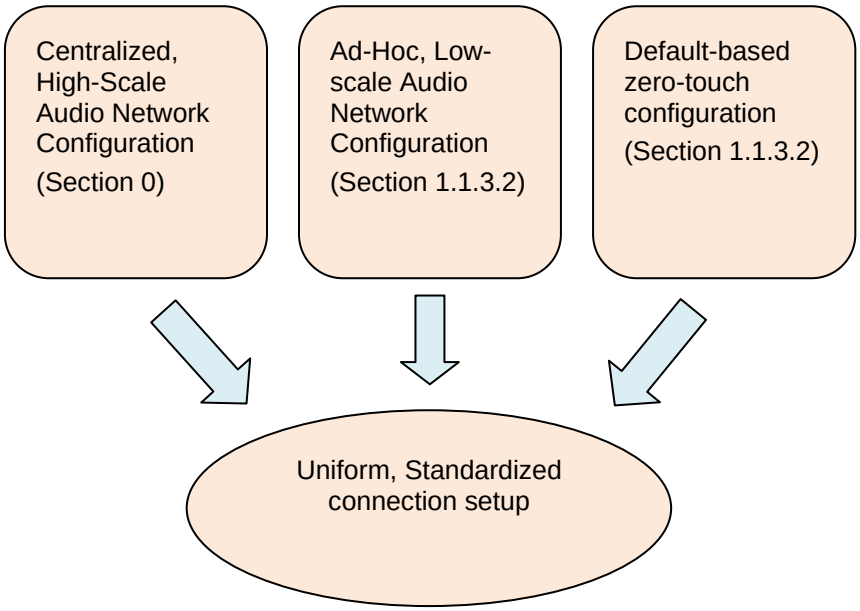
  The result of this first phase would not be actual creation of connections, but rather the configuration of each endpoint with the relevant name(s) of connections it should make.

  It seems possible that this phase could be bypassed entirely in a consumer or automotive use-case, where powered speakers (for example) might be pre-configured to subscribe to named connections Front Left or Front Right based on the setting of a physical switch on the device.
- In the second part of the procedure, talkers advertise the named connections that they are configured to source, and listeners subscribe to the named connections that they've been configured to receive using protocol sequences defined for SRP.

The two phases shouldn't be thought of as distinct and non-overlapping... it's more that the first part identifies which endpoints should participate in what connections, and transmits that information to the endpoints in the form of named connections, while the second part carries out the actual nuts and bolts of making the connection. If the management station changes the goal while a system is operating, the affected endpoints should drop whatever connections they're no longer supposed to have, and start making new ones.

I am assuming a bit more reliance on DNS-SD than at least I'd originally expected. Assuming the protocol can scale adequately, I think each endpoint should advertise not only that it's AVB-capable, but also what named connections it's capable of sourcing, along with the associated StreamID. The AVB-capable notification allows the system controller to discover everything, and then once nodes are configured with connection goals, seeing which Talkers can source what named connections would allow the Listeners to make connections on their own.

1.1.2 Configuration and Connection Phases



1.1.3Configuration Phase Sequencing Steps

This section sketches two ways to configure named connections

- One assuming a large centrally-managed system
- The other assuming a small ad-hoc system

1.1.3.1Large-Scale Pro

Fresh-from-the-factory Initialization

For Centrally-Managed, Large Scale Pro-Audio application

AVB Endpoint(s)	Connection Manager (System Controller?)
Initialize Ethernet link	
[start 802.1AS synchronization]	
Use AutoIP or DHCP to obtain an IP address	
Use mDNS to register an endpoint device name	
	Collect a list of device names
Use DNS-SD to advertise basic AVB capability “I can do AVB” “I’m an Adamson Powered Speaker, Model YXXX”	
	Collect a list of all AVB-capable end points
	Issue AV/C (or SNMP) commands to enumerate capabilities of each AVB-capable endpoint
Respond to unicast AV/C (or SNMP) commands for enumeration information	
	Compile a table of channel counts, bit rates, coding, etc for all devices. Use proprietary techniques (rotary switches, LCD panels, cable-ID, etc) to match device names with physical devices. Read a list of desired connections from local storage / GUI / whatever.
	Issue proprietary commands to set gain, DSP params, delay compensation, etc.
Respond to proprietary configuration commands	
	Issue (what protocol? SNMP?) commands to configure names of desired connections to talkers and listeners
Respond to connection name configuration commands.	
(Transition to Section 1.1.4 below)	

1.1.3.2 Small-Scale Configuration Phase

The point of this configuration sequence is to tell each talker and listener which named connections they should source or connect to. These steps could be sidestepped entirely for very simple systems where there are a couple of pre-configured connections (e.g. Front-Left-48kHz, Front-Right-48kHz).

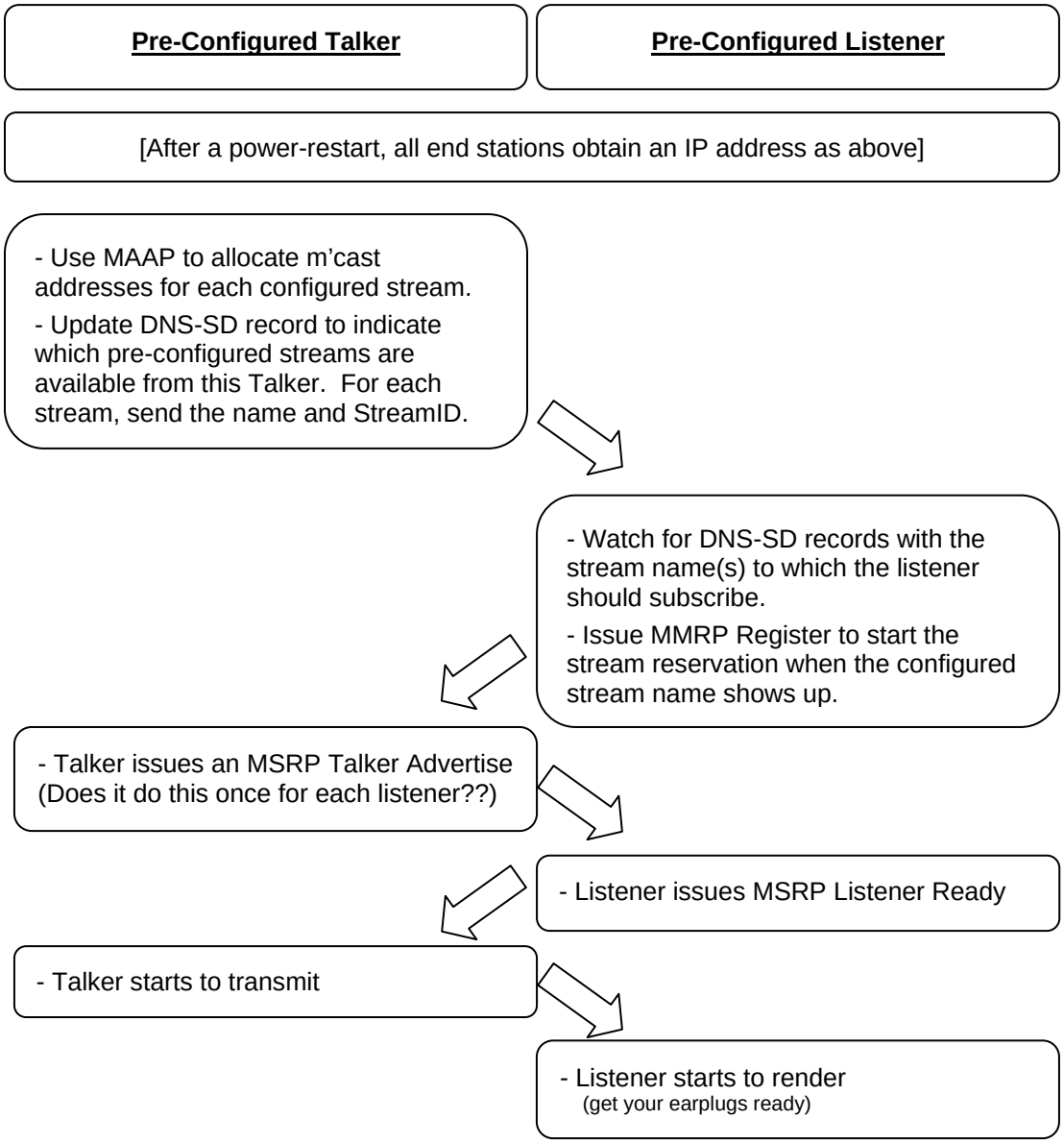
AVB Endpoint(s)	Connection Manager (System Controller)
Initialize Ethernet link	
[start 802.1AS synchronization]	
Use AutoIP or DHCP to obtain an IP address	
Use mDNS to register an endpoint device name	
Use DNS-SD to advertise basic AVB capability "I can do AVB" "I'm an Adamson Powered Speaker, Model PXXX"	
	Use a Bonjour Browser to see all AVB devices
	Log in to each device with a browser Configure each device using its web page Configure connection names for each device (or leave the defaults unchanged)
Respond to HTTP commands for configuration	
(Transition to Section 1.1.4 below)	

Questions

- Simplified configuration would require clearly-specified "profiles" for streaming capability, and one "least common denominator" that all AVB devices can support in case no further clues are available.
  Can we identify how many such profiles might be needed?

1.1.4 Connection Phase

At this point, assume that devices all have their named connections configured, and are now starting to set up connections. This stage could be reached either because the connection manager above finished configuring connection(s) or because a previously-operational system is starting up after a power failure.



For Further Study

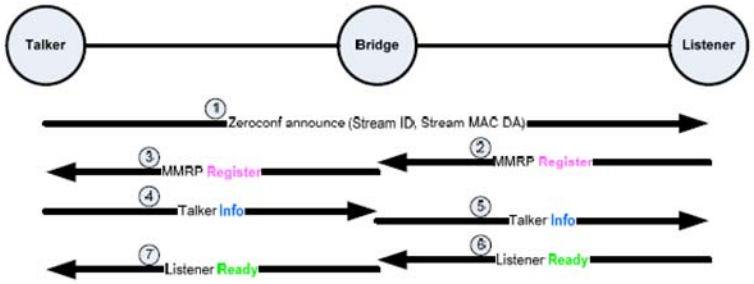
- Clarification on how the second listener joins would be good…
- Usually we want devices to "remember" what streams to join through power failure, but after a physical reconfiguration, it may be good to make sure they forget early in the process. (This general approach seems to make it difficult to promise that when a device gets plugged in, it won't make noise until it's told to.)
- Security considerations; some consideration should be given to who is allowed to tell whom what connections to make.

- We need to consider operation in mixed-vendor networks.
- Something has to configure the number of channels per stream.

Appendix

From Craig Gunther's at-cgunther-3step-msrp-0408-v2.pdf

Successful Stream Join



- Talker advertises stream via higher layer protocol (e.g. Zeroconf)
- Listener issues MMRP Register
- Talker responds with MSRP Talker Info
- Listener requests Stream with MSRP Listener Ready
- After receipt of the Ready the Talker can begin transmitting the audio/video stream at any time

Just change the phrase Talker Info to Talker Advertise and it should all make sense.

- Craig