Connection Management With AES70

Summary

OCA Alliance
oca-alliance.org
January 2016

The point of this paper is to propose using a subset of AES70, the Open Control Architecture (OCA), as an industry-wide solution to the problem of interoperable media network connection management. This subset is named "OCA-CM".

1. Introduction

It is now feasible to build IP-network-based professional media systems and studios. Such networks offer great flexibility, but pose an operational challenge: How do we control and manage the enhanced routing capability, especially in multivendor systems?

Today's media networks need **standardized connection management** services to help manage devices and their media interfaces and connections. .

Today, there are various media transport protocols available. Some of them include connection management schemes, some don't. The schemes may incorporate standard elements, but none is fully standardized. They do not interoperate.

The need for a standard connection management solution remains unmet.

2. The AES70 Opportunity

OCA, the Open Control Architecture, is a new professional media system control architecture. Recently standardized by the Audio Engineering Society as **AES70**, OCA is not itself a media transport scheme, but is designed to operate with all modern media transport methods.

OCA includes a fully engineered connection management scheme named **OCA-CM** that is suitable for managing connections for all audio and video products and systems, regardless of what media transport protocols they use. Because OCA is modular, products may use OCA-CM by itself, without adopting the whole OCA system control scheme. Thus, OCA-CM offers a universal capability that can be integrated into diverse products.

OCA-CM is an ideal connection management standard for pro media networks.

3. Connection Management Basics

Mechanisms. Connection management is the combination of two mechanisms: **Connection Setup**, the means by which system controllers direct media devices to make and break connections, and **Directory**, a data repository that remembers media network device names, and provides the necessary connection setup information for their use. **Discovery** is a common name for Directory services with automatic registration features.

Implementation. Inside a media network, a connection management implementation has three kinds of components: a set of **connection management agents**, one in each device; a single

common **directory (**aka **discovery**) service; and one or more **system controllers** that system operators use to manage connections and related information. These controllers may be dedicated workstations with user interfaces, or they may be software functions integrated into media devices.

4. OCA-CM Functions

OCA-CM provides a rich repertoire of functions to implement the mechanisms listed above. These functions fall into the following categories:

1. **Connection Setup**, for making and breaking media stream connections
2. **Directory** / **Discovery**, for using and managing the Directory
3. **Multiple Network Control**, to support devices that belong to multiple networks
4. **Device Supervision**, to detect device failures quickly
5. **Clock Management**, to enumerate and monitor device media clocking features
6. **Protocol Options**, to run OCA-CM over TCP, Secure TCP, or UDP networks.

5. Technical Highlights

Technical highlights of OCA-CM include:

1. **Stream-based or Channel-based Connection**. OCA-CM supports making connections either on a stream-by-stream basis (a stream is a "bundle" of signal channels) or an individual channel-by-channel basis, as required by each specific media transport protocol.
2. **Encoding Flexibility**. OCA-CM does not constrain signal encoding (i.e. sample size, sample rate, compression type) choices. Standard or proprietary encodings may be used.
3. **Compatibility With Native Connection Management**. OCA-CM co-operates with native connection management schemes that media transport protocols may have, in a way that allows freely mixing OCA-CM and non-OCA-CM (i.e. legacy) devices.

6. Future

Over time, it is hoped that an OCA-CM ecosystem will grow to embrace the range of audio and video transport protocols in use, and that many compliant products will be made. The OCA Alliance will facilitate this evolution by collecting specifications of OCA-CM **Adaptations**, the specific designs that define how OCA-CM is used with each media transport protocol.

As well, it is expected that **OCA-CM reference implementations** will be produced, either as open-source or licensable products.

Finally, it is envisioned that diverse **OCA-CM-compatible system controllers** will become available from device manufacturers and third parties, for managing network connections in diverse media network applications.

Connection Management With AES70

OCA Alliance
oca-alliance.org
January 2016

7. Introduction

With recent advances in digital networking hardware and software, it is now feasible to build IP-network-based professional media systems and studios. In such applications, the IP network not only saves cabling cost, but also provides huge improvements in flexible, software-defined signal routing.

Although this flexibility is valuable, it poses an operational challenge: How do we effectively control and manage all the new routing capability? How do we discover what devices, streams, and connection points are available, and what their characteristics are? How do we command devices to make and break connections? How do we tell when devices join and leave networks? And how do we do all this in a unified way when our networks include devices from multiple manufacturers?

Today's media networks need **Connection Management** services to help manage devices and their media interfaces and connections. These services must be **standardized**, so networked equipment from various manufacturers can interoperate.

Today, there are various protocols available for transporting audio and video over IP networks. Some of these protocols include connection management schemes, some don't. The schemes have some standard elements, but none is fully standardized. They do not interoperate.

The need for a standard connection management solution remains unmet.

8. The AES70 Opportunity

OCA, the Open Control Architecture, is a new professional media system control architecture. Recently standardized by the Audio Engineering Society as **AES70**, OCA is not itself a media transport scheme, but is designed to operate with all modern media transport methods.

OCA includes a fully engineered connection management scheme named **OCA-CM**. OCA-CM may readily be used to manage connections for all audio and video products and systems, regardless of what media transport protocols they use. Furthermore, because OCA is modular, products may use OCA-CM by itself, without adopting the whole OCA system control scheme.

Thus, OCA-CM offers a universal capability that can be integrated into diverse products. Audio and video networks assembled from such products will enjoy a unified management approach for all devices and connections, even though multiple media transport schemes may be used.

Finally, as a part of the AES70 standard, OCA-CM is a public standard, license-free solution.

AES70 OCA-CM is an ideal connection management standard for pro media networks.

9. Connection Management At A Glance

Connection management is the combination of two mechanisms:

- **Connection Setup**. Connection setup is the mechanism by which system controllers direct media transport services to make and break connections with media streams on the network. Connection setup is a **control** mechanism.

- **Directory**. A **directory** is a data repository that stores media network devices and capabilities. It provides the necessary information to create an OCA-CM connection to a device.

  A **discovery service** is a kind of directory service that allows automatic registering and de-registering of devices in the directory, and provides additional services such as directory browsing and automatic notification of directory contents changes.

Inside a media network, a connection management implementation has three kinds of components:

1. A set of software **connection management agents**, one in each device.
2. A single common **directory service**. This service may run as a central service, or the function may be distributed into the connection management agents. It may include a discovery service.
3. One or more **system controllers** that system operators use to manage connections and related information. These controllers may be dedicated workstations with user interfaces, or they may be software functions integrated into media devices.

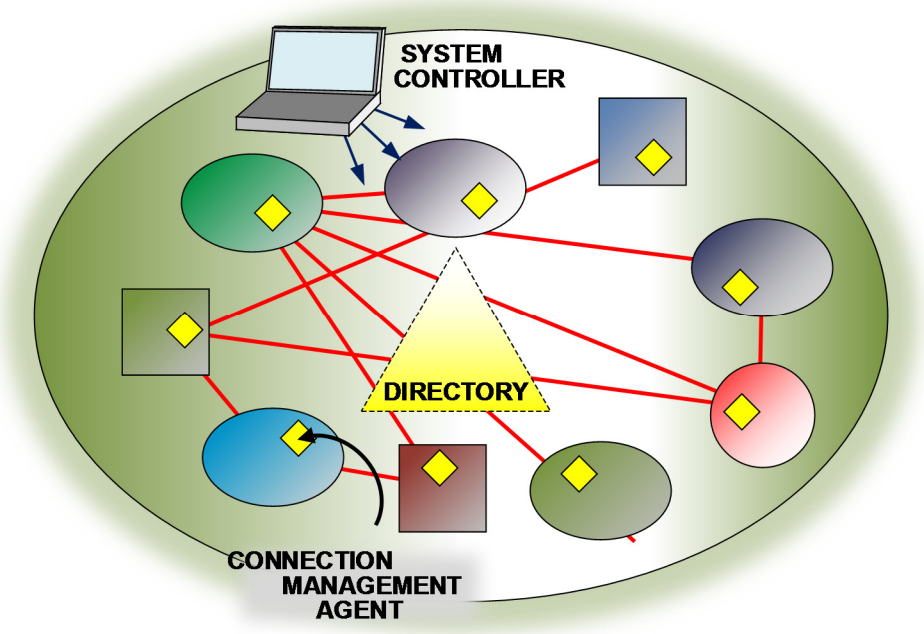

Figure 1. Connection management components

10. OCA-CM Functions

Here is a concise list of OCA-CM functions. Explanations and details are provided in Section 11 below. Not all implementations will support the entire repertoire given here. Requirements for a minimum compliant implementation are to be developed.

Connection Setup

1. Make and break unicast or multicast connections.
2. Create and delete multicasts.
3. Operate on stream-by-stream basis or channel-by-channel basis.
4. Allow fan-out (multiple output streams per output endpoint) connections.
5. Support media-transport-specific stream encoding formats.
6. Enable / disable stream and channel input/output.
7. Monitor connection status.
8. Enumerate available connection endpoints within devices.
9. Enumerate endpoints' available stream encoding options.
10. For dynamically-configurable devices, add/change/delete endpoints.

Directory / Discovery (IP networks)

11. Automatically register & deregister devices as services in the media device directory.
12. Browse the media device directory.
13. Subscribe to media device directory changes, e.g. new registration, device failure.
14. For large networks allow server based unicast discovery.

Multiple Network Control

15. Allow devices to belong to multiple media networks of the same or different types.
16. Enumerate networks to which device belongs.
17. Retrieve/modify network connection parameters - host names, IP parameters, etc.
18. Start/stop/pause input/output operations on network.
19. Monitor network status.

Device Supervision

20. Operate keepalive mechanism to detect device failures within seconds.

Clock Management

21. Enumerate device media clocks and their options.
22. Enquire / track media clock status .

Protocol options

23. Run over TCP or (soon) UDP.
24. Allow secure connections.

11. Technical Highlights

11.1. Connection Modes

OCA-CM's connection control options depend on whether the device's connection mode is **stream-based** or **channel-based.** Which mode is used depends on the particular media transport protocol the device implements.

1. Stream-based connections connect devices to **streams**. A stream is a collection of signal channels. In stream-based routing, the entire stream is connected or disconnected at once.

   OCA-CM includes functions for connecting and disconnecting streams, for managing a device's stream connection endpoints, and for routing stream signal channels to the right places within the device.

2. Channel-based connections connect individual signal **channels** in the network to individual signals in the device. When channels are transmitted over the network, they may or may not be aggregated into streams (they often will be, for reasons of transmission efficiency), but the media transport protocol implementation hides that detail from the application.

   OCA-CM defines functions for connecting and disconnecting channels and for enumerating and managing a device's channel connection endpoints.

OCA-CM supports both stream-based and channel-based connection modes.

11.2. Encoding

Signal encoding details (sample width, sample rate, compression type, etc.) vary among the various media transport protocols. OCA-CM itself does not make any assumptions or impose any restrictions on media format or encoding. Different media transport schemes may define whatever specific encodings they need, and OCA-CM will use those definitions.

OCA-CM does not restrict signal encoding.

11.3. Compatibility

As noted above, some current media transport standards and implementations include native connection management schemes. For upward compatibility, new connection management schemes such as OCA-CM must cooperate with these native schemes.

OCA-CM is designed to cooperate seamlessly with native connection management methods. Adding OCA-CM to a product will allow users to use OCA-CM going forward, while retaining compatibility with parts of their systems using legacy connection management methods.

This compatibility is possible because OCA-CM logically resides *alongside* of media transport software, not *inside* it. Thus, it need not displace legacy connection management elements. Figure 2 illustrates these relationships.

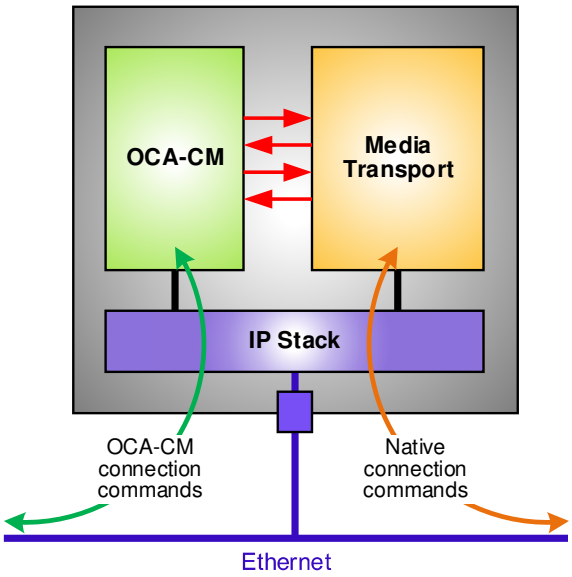

Figure 2. OCA-CM and native connection schemes

In this side-by-side architecture, OCA-CM and the native elements are mutually synchronized, so that both are fully aware of each other's actions and states. This means that a device may operate in a mixed mode, in which some of its connections are made by OCA-CM, and some are made via native functions. When this happens, both OCA-CM and the native functions will be aware of *all* the connections, even ones they did not make.

The key benefit of this approach is that OCA-CM devices can interoperate with legacy devices without making connection management compromises.

OCA-CM is compatible with native connection management.

12. The OCA-CM Ecosystem

Over time, it is hoped that an OCA-CM ecosystem will grow to embrace the range of audio and video transport protocols in use, and that many products will be produced that implement OCA-CM connection management.

When OCA-CM is first combined with a specific transport protocol, a design called an **Adaptation** is prepared that defines encoding descriptors and other specific OCA-CM options for that transport protocol.

As OCA-CM evolves, its implementations will be guided by a growing collection of Adaptation specifications produced by various implementers and distributed by the OCA Alliance.

Adaptations will not become part of the AES70 (OCA) standard, but instead will become engineering reference documents. The OCA Alliance will facilitate the development of Adaptations, and will maintain an up-to-date online library of downloadable public Adaptation specifications.

In addition to Adaptation specifications, **OCA-CM reference implementations** may be produced, either as open-source or licensable products.

Furthermore, it is envisioned that diverse **OCA-CM-compatible system controllers** will become available for managing network connections in various media network applications. These controllers may be integrated into product offerings or sold as standalone software products from media device manufacturers or from application software companies.

Furthermore, as the AES70 standard evolves, OCA-CM will gain the ability to embrace new kinds of networks, and will generally grow in an upwards-compatible, low-risk manner.

Technical Details

Appendix A. Overview

Here is a technical overview of OCA-CM.

Note on OCA-CM specifications: OCA-CM functions are defined in object-oriented terms. In these terms, a template of related control functions is called a **control class**. The parameters and procedures of each control class map directly into control protocol commands, responses, and notifications.

Names of OCA-CM control classes begin with "**Oca**", e.g. **OcaStreamNetwork**.

To implement the functions of a particular control class, a device is said to *instantiate* the class. Each instance of a control class is called a *control object.* An instance of control class XX is referred to as an *XX object.* For instance, an instance of class **OcaStreamNetwork** would be called an *OcaStreamNetwork object*.

Where necessary, a control class may be instantiated multiply within a device to control a replicated function. OCA-CM has unique object identifiers, so there is no confusion between instances.

Specific classes, objects, and processing rules may be collected into larger aggregations called **control models.**

Although OCA-CM is described in an object-oriented manner, its implementation need not employ object-oriented programming methods. As long as a device sends and receives proper OCA commands, responses, and notifications, the internal details of its implementation are unconstrained by OCA-CM.

A.1. Elements

The elements of OCA-CM are:

1. A **connection control model** for describing connections, streams, channels, connection endpoints, and actions on those elements.

   The connection control model allows for the making, monitoring, and breaking of connections either on a stream-by-stream basis or on a channel-by-channel basis, and supports the legacy compatibility described above.

   The connection control model is the same regardless of the type of network or media transport protocol being used. Specific model features for each media transport protocol are given in the Adaptation for that protocol.

   In accordance with OCA practice, the connection control model is defined in object-oriented terms. Thus, it is an object model that is defined according to the object-oriented design conventions described above.

2. **For IP networks**:
    1. A **control protocol definition** for operating the control model over the network. Named "OCP.1", this protocol has a compact binary format designed for rapid processing. It runs over TCP data transport, and will soon run over UDP as well. TCP implementations may be encrypted using the standard TLS suite; encryption parameters are defined by OCA-CM.
    2. A **directory scheme** based on the DNS-SD and mDNS protocols. This combination is informally known as "Bonjour", an Apple name. Directory schemes are more commonly known as **discovery schemes**; "Directory" is the more general of the two terms, and is used here.
3. For each **future network type** (e.g. Bluetooth) that may be defined:
    1. A control protocol definition
    2. A directory scheme

Details of current OCA-CM elements follow.

A.2. Connection Setup Services

OCA-CM supports two connection setup modes - one for stream-based connections, one for channel-based connections. A summary of these modes follows. For full details, please see the AES70 standards documents.

A.2.1. Stream-Based Mode

The central element of OCA-CM stream-based connection mode is a control class named **OcaStreamConnector**. **OcaStreamConnector** objects represent a device's connection endpoints for streams on the network. Each object contains the endpoint name, encoding descriptions, and the mapping between signal channels in the stream and signal channels inside the device.

OCA-CM does not restrict the number of **OcaStreamConnector** objects a device may have.

OcaStreamConnector objects may be configured for input or output (but not mixed) operation. Output **OcaStreamConnector** objects may connect to multiple streams, but input **OcaStreamConnector** objects may connect to no more than one stream.

Figure 3 illustrates stream-based input; Figure 4 illustrates stream-based output.

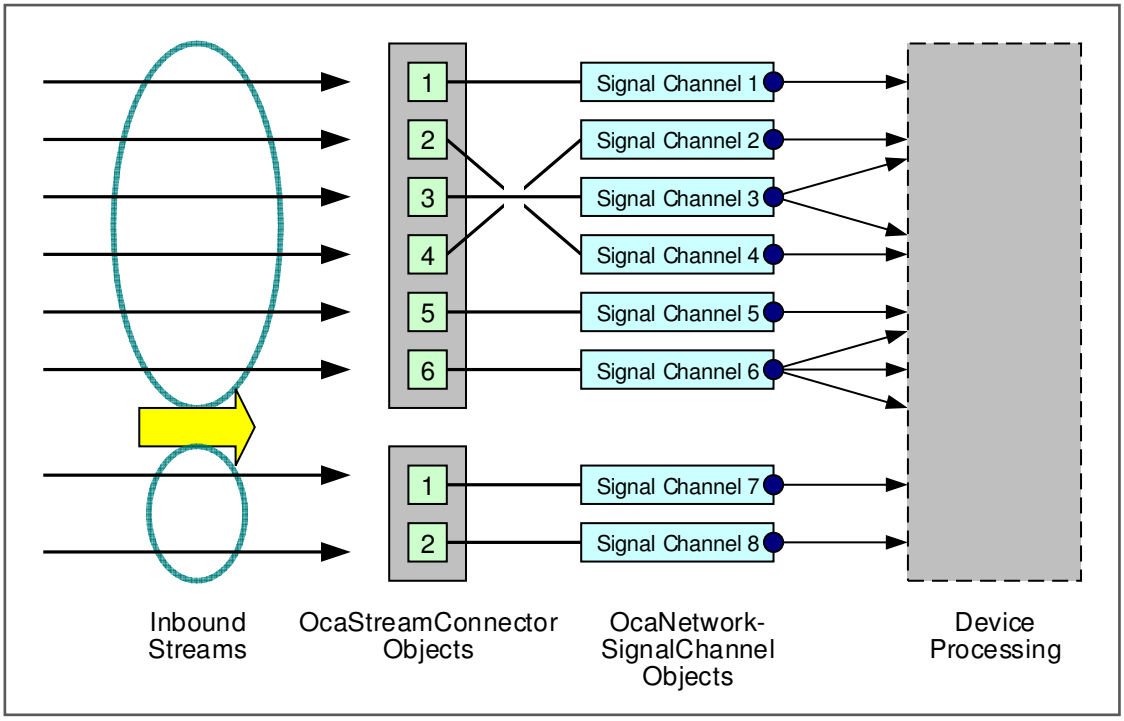

Figure 3. Stream-based input

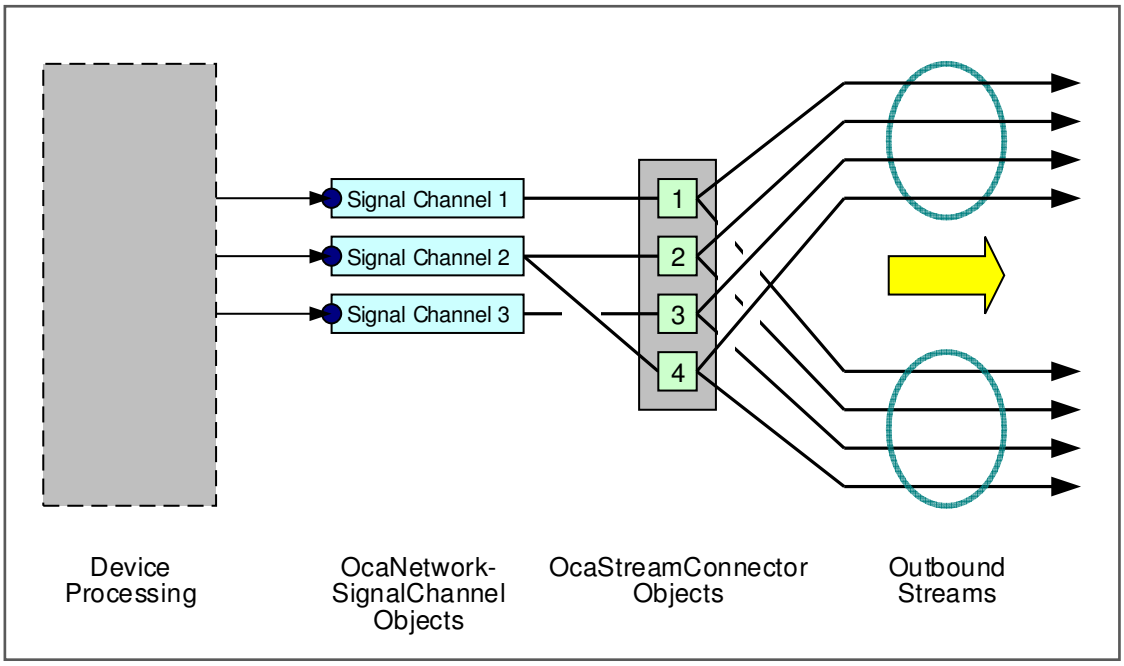

Figure 4. Stream-based output

A.2.2. Channel-Based Mode

The central element of OCA-CM channel-based connection mode is a class named **OcaSignalChannel**. **OcaSignalChannel** objects represent a device's connection endpoints

for signal channels on the network. Each object contains the endpoint name and encoding descriptions.

Each **OcaSignalChannel** object represents one device signal channel. OCA-CM does not restrict the number of **OcaSignalChannel** objects a device may have.

Figure 5 illustrates channel-based input; Figure 6 illustrates channel-based output.

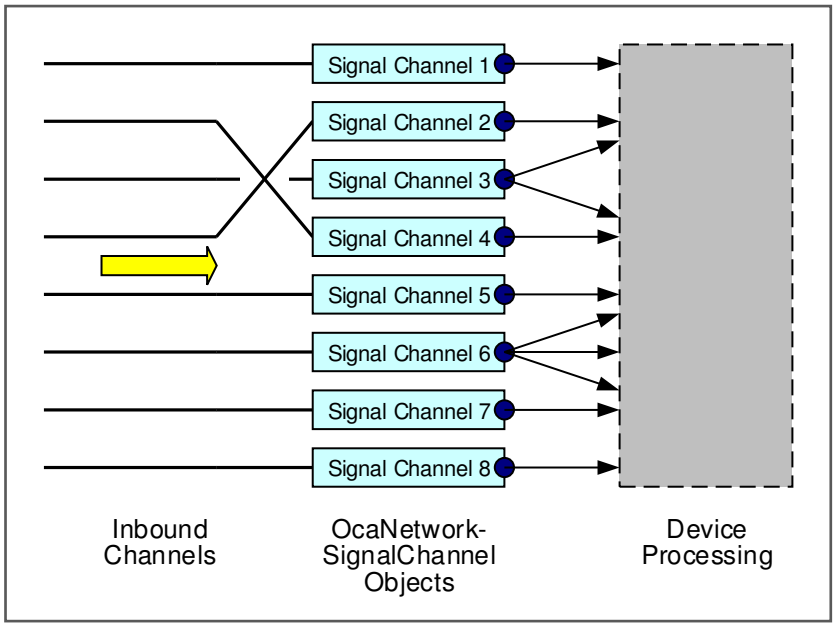

Figure 5. Channel-based input

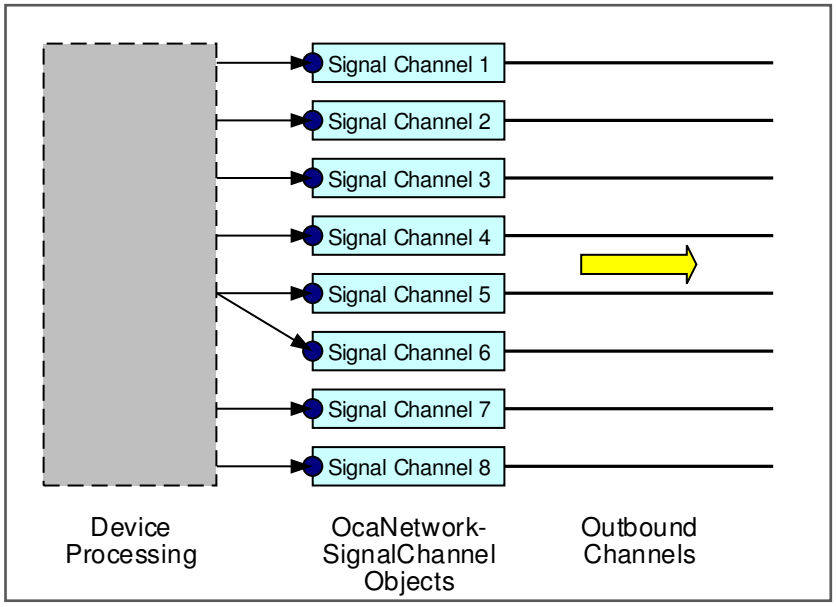

Figure 6. Channel-based output

A.3. Directory/Discovery Services

This appendix describes how OCA-CM directory/discovery services work for IP networks. Full details are in the OCP.1 protocol specification, standardized in AES70 Part 3.

For future network types, OCA-CM will have different directory/discovery architectures. However, the underlying control model will be essentially the same as for IP networks.

A.3.1. Terminology

For discussing directories and discovery, OCA-CM users the following terms:

- A **media network directory** is a database that holds the mappings between media network service names (e.g. "Equalizer.Left") and network host names (e.g. "EQ.Subnet1.Venue").
- A **network host directory** is a database that holds the mappings between host names ("EQ.Subnet1.Venue") and host addresses ("192.168.4.1").
- A **directory service** is a network service that maintains a directory, and allows querying and updating it.
- A **discovery service** is a kind of directory service that provides for automatic registering and de-registering of devices in the directory, and provides additional services such as directory browsing and automatic notification of directory contents changes.

Together, the media network directory and the network host directory can translate the name of a media network service to a host address and port number. This allows connecting to a given media network service by giving its service name.

The term "discovery" is sometimes also used to mean discovery of internal device capabilities. This kind of discovery is outside the scope of the OCA-CM discovery mechanism. In OCA terms, the word "enumeration" is used to describe the discovering of internal device capabilities. Full OCA includes complete enumeration mechanisms, but these are not part of OCA-CM.

A.3.2. Service Discovery Architecture

From AES70 Part 3:

> *" The AES70 device discovery process shall have a service discovery architecture, in which AES70-3 devices shall register themselves in a directory of network services which may subsequently be queried by network entities needing to know device IP addresses.*
>
> *"Service discovery shall be implemented using DNS-based Service Discovery (see RFC 6763)."*

DNS-based Service Discovery, abbreviated as DNS-SD, provides a combined media network directory and network host directory.

OCA-CM devices register themselves in DNS-SD directories as follows:

- If a device uses insecure OCA-CM, it registers itself as a service of type **"_oca._tcp**". If using secure OCA-CM, it registers itself as a service of type **"_ocasec._tcp**".
- In both cases, the service name is given by the **NameAdvertised** property of the **OcaStreamNetwork** object that the connection is using. If this name is changed during operation, the device de-registers itself and re-registers under the new name.

Once a device is registered in the DNS-SD directory, OCA-CM controllers may find its IP address by browsing the directory. Also, controllers may subscribe to the directory to receive change notifications.

A.3.3. Server / Serverless Operation

In single-subnet applications, OCA-CM registration uses the multicast DNS (mDNS) protocol (see RFC 6762). mDNS implements a distributed DNS service that does not require a central server.

mDNS is effective for networks of up to about 100 devices. For networks larger than this, mDNS multicast traffic becomes excessive at startup times. For such networks, a DNS server must be provided. Upon discovering a DNS server, mDNS automatically switches to server-based mode, and its multicast traffic all but disappears.

Appendix B. Implementation

When OCA-CM is implemented in a device, the device acquires a connection management network control interface, accessible over the network by the control protocol defined for the network type being used.

In IP-based devices, the OCA-CM protocol OCP.1 will normally share a network connection with the media transport service, but will use its own TCP or UDP software port. OCP.1 does not have a fixed port assignment; instead, developers may choose any value from the standard dynamic port range (49152-65535). The value chosen is registered in the device's directory entry, so other controllers and devices may learn it.

Where a device supports multiple media transport types, only one OCA-CM implementation need be used. The OCA-CM design is capable of handling multiple transport services.

The relationship between OCA-CM and the media transport service(s) is implemented via a media transport interface on the OCA-CM side that calls the media transport connection setup API(s). This is shown in Figure 7.

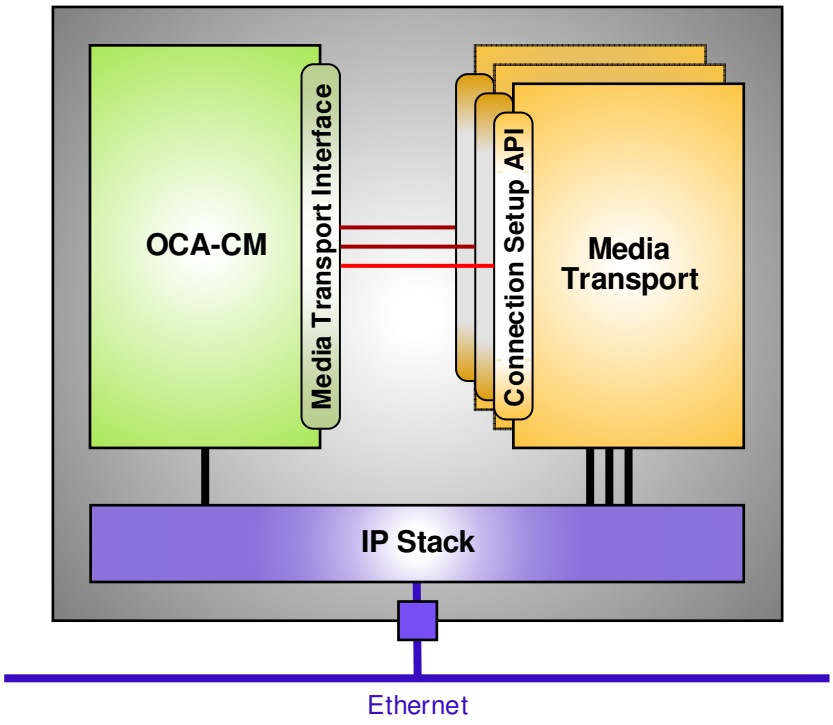

Figure 7. OCA-CM implementation, showing OCA API and multiple media transport protocols

B.1.1. Implementation Steps

Here are typical steps to implementing OCA-CM with a particular media transport protocol. In what follows, we'll call the transport protocol **"TPx".**

1. Decide which OCA-CM functions will be implemented for TPx. The OCA-CM specification will itemize the mandatory and optional functions.
2. Create the Adaptation specification. The OCA Alliance can help do this. Normally this task will require collaborating with the TPx user community. Typical steps include:
   - Choose stream-based or channel-based operation, or both.
   - Identify the OCA-CM classes that apply to the chosen connection mode.
   - Define the correspondence between the OCA-CM and TPx connection parameters.
   - Define any customized classes that may be required (OCA-CM supports the seamless use of customized classes).
   - Define any specific data formats required - for instance, the encoding descriptor.
   - Choose OCA-CM data transport - TCP/Secure, TCP/Insecure, or UDP/Insecure.
   - Secure consensus on all the above in the appropriate constituency.
   - Write the Adaptation specification and, if you want to make it public, file it with the OCA Alliance.
3. Proceed with implementation design. Typical steps include:
   - Identify target software and hardware environments - hardware platform(s), operating system(s), language(s), software libraries, etc.
   - If feasible, identify a usable OCA-CM reference implementation. Or maybe create one by subsetting a full OCA reference implementation.
   - Design the control interface between OCA-CM and the transport protocol software. This interface has two sides: the OCA-CM implementation has a **Media Transport Interface**, and each media transport implementation (if there is more than one) has a **Connection Setup API**).
   - Code the OCA-CM driver, or adapt it from a reference implementation if using one.
   - Adapt the media transport driver to work with the OCA-CM control interface.

When this process has completed, there will be:

- A public or proprietary Adaptation specification.
- An OCA-CM implementation for the chosen media transport protocol and the chosen hardware/software environment(s). This could be a proprietary or an open-source implementation.
- A version of the media transport protocol driver that works with OCA-CM.

Appendix C. References

OCA is fully described by three AES70 standards documents, as follows:

- AES70-1-2015: AES standard for audio applications of networks - Open Control Architecture - Part 1: Framework

  This document describes the models and mechanisms of AES70.

- AES70-2-2015: AES standard for audio applications of networks - Open Control Architecture - Part 2: Class structure

  This document specifies the control class structure for AES70 that defines AES70's control and monitoring functional capabilities.

  Part 2's detailed class definitions are contained in an external Universal Modeling Language (UML) annex. This document may be downloaded from the AES website in two formats, as follows:

  - Official XMI format: www.aes.org/standards/models/AES70-2-AnnexA-151112-class-structure-1.xmi
  - Enterprise Architect format: www.aes.org/standards/models/AES70-2-AnnexA-151112-class-structure-1.eap

- AES70-3-2015: AES standard for audio applications of networks - Open Control Architecture - Part 3: Protocol for TCP/IP Networks

  This document defines the OCP.1 communications protocol of AES70. This protocol supports AES70-compliant remote control and monitoring of media devices over TCP/IP networks.

  A UML description of OCP.1 protocol data unit formats is in an annex, here:

  - Official XMI format www.aes.org/standards/models/AES70-3-AnnexB-151112-tcpip-protocol-1.xmi
  - Enterprise Architect format: www.aes.org/standards/models/AES70-3-AnnexB-151112-tcpip-protocol-1.eap

For the UML files listed above, using the Enterprise Architect format is recommended. Enterprise Architect is available from Sparx Systems, who offers a free viewer, downloadable from the following page:

- http://www.sparxsystems.com/bin/EALite.exe.