CORBA-Based Stream Control and Management for IP-Based Production Studio Networks

Terence Song[1], Dritan Kaleshi[2], and Alistair Munro[2]

[1] Wireless and Networks Research Labs, Centre for Communications Research, University of Bristol, Merchant Venturers Building, Woodland Road, Bristol, BS8 1UB, United Kingdom
Terence.Song@bristol.ac.uk
[2] Department of Electrical and Electronic Engineering, University of Bristol, Merchant Venturers Building, Woodland Road, Bristol, BS8 1UB, United Kingdom
{Dritan.Kaleshi, Alistair.Munro}@bristol.ac.uk

Abstract. Traditionally, device relationships in a production studio are established and managed by physically routing the different cables that carry the relevant streams, together with any required synchronization references. IP-based networks offer a common infrastructure for the distribution of broadcast content as well as the deployment of integrated production and management solutions. This paper describes the design and implementation of components based on the OMG Audio/Video Streams Specification to facilitate the rapid integration of devices during the initial set-up phase, and the control and management of media content flow during normal operations for the purpose of program production within an IP broadcast environment.

1 Introduction

The use of heterogeneous IP-based networks as a basis for the distribution of production studio content presents a number of interesting challenges. The operational requirements of a production studio far exceed those of more common Internet-based streaming applications (e.g. video-conferencing, stock quotes, etc.). The production studio is made up of heterogeneous sources, sinks, and intermediate processing devices; such as microphones, speakers, mixers, recorders, players, and storage devices. These devices form complex relationships with each other, usually under the control of a studio manager or producer, to define the flow of media content during the course of program production (i.e. capture, editing, storage, distribution, and presentation). The two main issues related to the distribution of production content are: the control and management of stream relationships, and the distribution of synchronization information. The work described in this paper relates to the first of these two issues. In particular, we describe the design and implementation of IDL-defined components for the control and management of flows within the audio plane of an IP broadcast studio environment for the purpose of program production.

Traditionally, device relationships in a production studio are established and managed by physically routing the different cables that carry the relevant flows and

A. Marshall and N. Agoulmine: MMNS 2003, LNCS 2839, pp. 1–17, 2003.

streams, together with any required synchronization references. IP-based networks, however, offer a common infrastructure for the transport and automatic routing of media data, as well as the deployment of integrated broadcast content production and management solutions. The heterogeneity of IP-based networks allows the stream establishment and management problem to be addressed at the application-level, instead of the physical-level. The control and management of streams within the production studio consists of two main parts—the definition of studio device relationships during the initial set-up phase, and the dynamic re-configuration and modification of those relationships and their properties (e.g. formats, device parameters, etc.) thereafter.

In order to achieve maximum extensibility, maintainability, and interoperability, our objective has been to re-use standard technologies to the greatest extent possible. To this end, the aims have been to evaluate the applicability of the Object Management Group's (OMG) Audio/Video Streams Specification [1] and to develop extensions for the control and management of audio flows within an IP broadcast studio environment. The Common Object Request Broker Architecture (CORBA) [2] distributed processing environment (DPE), which provides a scalable, extensible, and robust environment for distributed objects to co-exist and inter-play, forms the basis for the implementation of the control and management components. The stringent performance requirements for streaming data often preclude the use of distributed object computing middleware as the transport mechanism, as demonstrated in [3]. The focus of the work described in this paper, however, is on middleware support for stream control and management, not stream transport. Instead, the transport of time-based media in our implementation is facilitated by the Real-Time Transport Protocol (RTP) [4] through the Java Media Framework (JMF) [5]. Previous works based on the OMG Audio/Video Streams Specification have focused on media streaming, with the control and management aspect being of lesser importance. Our contribution demonstrates its application to a production studio environment, where the control and management of complex flow and stream topologies is critically important.

The remainder of this paper is organized as follows. Section 2 begins with a definition of what makes up a stream, followed by a brief overview of the OMG Audio/Video Streams Specification. In section 3, we present the production studio's audio plane and describe the definition of flow control and management components to model its layout and contents (i.e. the devices and their relationships). Section 4 provides a description of the production studio's set-up and normal operations using a design patterns-based approach. The relevant issues that influenced our design and implementation of flow control and management objects are highlighted in section 5, followed by a brief description of our prototype implementation in section 6. The conclusions are presented in section 7.

2 Overview of the OMG Audio/Video Streams Specification

The OMG Audio/Video Streams Specification [1] defines a standardized, flexible, and efficient CORBA-based architecture for the control and management of streams. Stream control and management concerns the establishment, control, management, and release of streams (i.e. the specification defines control flow, not data flow). The

architecture is independent of media streaming frameworks and is generic enough to support arbitrarily complex streams.

2.1 Stream Model

In the Internet domain, the term *stream* is more commonly used to refer to the continuous transfer of a particular format of media content from a (synchronization) source to one or more sinks. In this paper, however, the continuous transfer of media content from a synchronization source is explicitly qualified by the term *content stream*, and the terms *flow* and *stream* refer to abstractions (i.e. local representations of distributed processes) used to facilitate the control and management of content stream instances.

Figure 1 depicts an abstract model of the basic elements that make up a stream. The origin and destination of a content stream is referred to as the *producer* and *consumer flow endpoint* respectively. Flow endpoints are created by *flow devices* to support flows. A *flow* abstracts one or more content streams of the same type (i.e. format). More precisely, a flow represents the continuous transfer of media content of a particular format (e.g. audio or video) in a clearly identified direction, from one or more producer endpoints to one or more consumer endpoints. A *stream* is an abstraction for aggregating multiple flows in parallel, which may travel in either direction, between *stream endpoints*. Similarly, a stream endpoint, which represents the termination point of a stream, logically contains one or more consumer and/or producer flow endpoints (of which some or all may be involved in the stream). No directionality is attached to a stream endpoint, since it may contain both producer and consumer flow endpoints simultaneously. However, specialised or typed stream endpoints may be used to assist in matching compatible flow endpoints during the stream establishment process. Stream endpoints are created by *multimedia devices* to support streams. A multimedia device abstracts a collection of one or more flow devices. Flow and multimedia devices are responsible for creating and destroying their respective flow and stream endpoints.

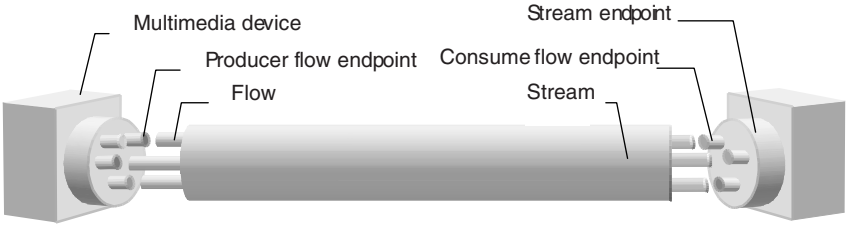

Fig. 1. The basic elements of a stream.

The stream abstraction defines a containment hierarchy, whereby a stream contains one or more flows, and each flow contains one or more content streams. That is, the containment hierarchy defines one-to-many relationships between containers and con-

tainees, and by transitivity, a *stream* contains one or more *content streams*. It is important to note that flows can exist independently of streams. The stream-related elements simply provide the ability to aggregate their respective flow-related counterparts, thereby allowing the latter to be controlled and managed as a composite entity.

2.2 Stream Control and Management

The OMG Audio/Video Streams Specification [1] defines the set of component interfaces that abstract the basic elements that make up a stream. There are two conformance levels, allowing implementations to trade-off between flexibility and efficiency. The light-profile defines components that represent a stream and its associated multimedia devices, virtual devices, and stream endpoints. Flow-related functionality is co-located within and accessed through the encapsulating stream-related component interfaces. The full-profile provides for greater granularity of control by also exposing flow-related interfaces (i.e. flows, flow devices, and flow endpoints). Since flows also support the relevant management interfaces, they can exist independently of streams. The component interfaces can be grouped into two distinct sets—interfaces onto flow and stream *control and management objects* (representing the continuous transfer of media content), and interfaces onto flow and stream *interface control objects* (representing the participants involved in the media transfer). More importantly, the specification defines how control messages are transmitted and received in a CORBA-compliant way between these two sets of interfaces to set-up, control, manage, and release flows and streams.

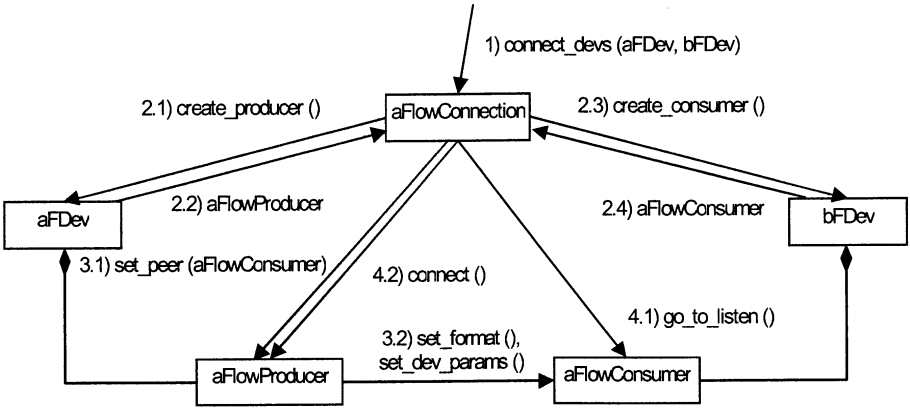

Fig. 2. The point-to-point flow establishment process.

Flow and stream *control and management objects* play a central role in the architecture. A control and management object provides two different but interrelated levels of abstraction. At the highest-level of abstraction, it represents the continuous transfer of media content (i.e. a flow or stream). This abstraction is suitable for mod-

elling potential, as well as established, device relationships. A flow or stream, however, involves multiple distributed processes that facilitate the media transfer (i.e. it is a distributed entity). Therefore, the control and management object is also a local representation of the flow or stream participants. To support these two abstractions, the control and management object supports two sets of operations: operations for binding devices and endpoints, and operations for controlling and managing that binding. Because the binding mechanism is standardized, interoperability between different implementations is possible. The establishment of a point-to-point flow binding is illustrated in Figure 2. The process consists of three main phases: endpoint creation, configuration, and transport set-up. Upon receiving a request to connect two flow devices, the flow control and management object (as a potential binding) requests each device to create an endpoint that will support the flow. The flow control and management object then requests one of the endpoints to ensure that its peer is configured similarly. Once the endpoints have been configured, the flow control and management object completes the binding process by instructing the consumer to begin listening on a particular address, which may be explicitly specified or implicitly determined, and for the producer to connect to that same address. Once established, the flow control and management object (as an established binding) supports operations for starting, stopping, and destroying the flow.

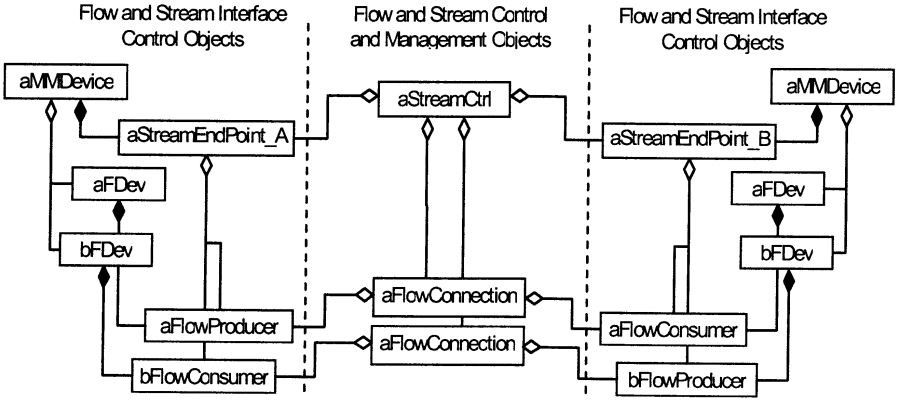

Fig. 3. A stream binding established between full-profile IDL-defined components.

Figure 3 shows the associations between components involved in a full-profile binding between two multimedia devices (c.f. Figure 1). The stream establishment process is similar to that for a flow, except for an additional flow endpoint matching phase; where the compatibility of flow endpoints supported by stream endpoints is determined. In a full-profile implementation, a collection of flows can be established independently and then individually added to a stream via operations supported by the stream interface. The stream-related components simply provide a way of aggregating flow-related components, as illustrated in Figure 3. The component interfaces can be extended to provide custom stream handling appropriate to the application.

Because the Audio/Video Streams Specification provides a programmatic description of the binding process, the fact that an IDL-defined component is a CORBA object is often overlooked. In our modelling efforts, we found that an object service provider/requestor view provides a greater understanding of the mechanics that underlie the architecture. Additionally, in our analysis of the control and management architecture, we discovered that the roles of control and management objects, and indeed the dynamics of the architecture, can be described simply in terms of two design patterns—*Mediator* and *Facade* [6]. It is unclear whether the occurrence of these two design patterns in the architecture is by design, or a consequence of it. In any case, sections 4.1 and 4.2 will further assert the roles of control and management objects within our IP broadcast environment using a design patterns-based approach.

2.3 Property Management

Fundamental to the operation of any control and management application is the ability to represent, access, modify, and exchange management-related information. The Audio/Video Streams Specification makes extensive use of properties to describe devices, streams, flows, and their endpoints. In-line with CORBA concepts, the specification does not build non-typed properties into interfaces. Instead, component interfaces inherit from CORBA's Property Service [7], which is used extensively for managing properties. This inheritance separates the concern of property management from those of stream control and management. Supporting a generic interface for property management allows the management information set to be extended dynamically without requiring changes to stream control and management interfaces. More importantly, however, a standard interface allows for interoperability between the stream components, allowing peers to be queried to establish their status and their compatibility constraints. In addition to a standard interface, a standardised set of parameters is also defined to allow for interoperability between different implementations. These properties are managed internally by stream components and are defined as read-only to clients. Additionally, application-specific parameters may also be associated with stream components; for example device model, serial number, owner, location, etc.

3 Modelling the Production Studio

The production studio is made up of heterogeneous sources, sinks, and intermediate processing devices that form complex relationships with each other dynamically or under the control of a studio manager or producer. The talkback system, which forms an integral part of the production studio, provides participants in the production process with communication and audio monitoring capabilities. An IP-based network forms the underlying infrastructure used for transporting the media streams (live or stored audio and video, and auxiliary data) used as input to produce a program for broadcasting and/or storage. In addition, the infrastructure supports facilities for service discovery and brokering [8]. In this section, we describe the modelling of the production studio's audio plane by full-profile *flow components*. Although stream-

related components have not been considered, the concepts discussed extend naturally to include collective control and management of multiple flows through stream-related interfaces (as described in section 2.1 and 2.2).

3.1 The Studio Audio Plane

A schematic of the production studio's audio plane is shown in Figure 4. The schematic shows the flow relationships established between the various devices as well as the direction of those flows. The main source of audio is a microphone. The most common sink of audio is a speaker. Other relevant audio devices include mixers, recorders, playback devices, effects processors, etc. The creation and processing of media content occurs in both digital and analogue form. The point-of-presence (PoP) of a device is the point where the digital content it produces or consumes can be identified and managed. It is realized that not all devices present in the studio can or will be directly controlled at their PoP. However, in our modelling efforts, we have assumed that gateways can be used to provide suitable converting functions to enable the transfer of control and management functions. The synchronization streams shown in the figure will not be modelled as synchronization of content streams will be derived primarily from RTP/RTCP algorithms [4]. This schematic forms the basis for the definition and implementation of audio flow control and management components.

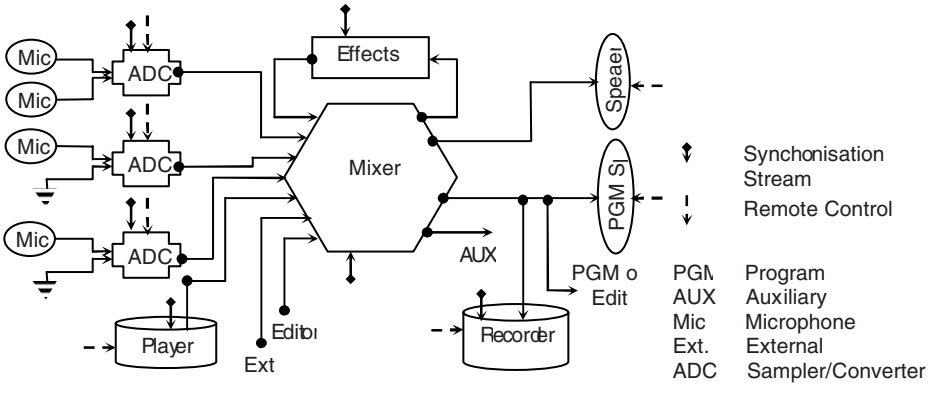

Fig. 4. Studio Schematic: Audio Plane.

3.2 Audio Flow Control and Management Components

Our modelling of the production studio's audio plane is based on full-profile components of the Audio/Video Streams Specification. A full-profile implementation allows for a modular approach to the development of stream components at design-time, and greater flexibility in the composition and distribution of media content at run-time. Indeed, our current production studio environment is modelled and built on audio

flow components, and this can subsequently be extended to include video flow components as well as more complex stream configurations as required. At runtime, flows (e.g. audio, video, talkback, etc.) can be structured into layers or hierarchies, and dynamically added to or removed from stream structures, allowing flows to be controlled and managed individually, or collectively, as dictated by the production process.

Multimedia device objects and their associated resources (i.e. virtual devices, endpoints) are anticipated to execute on the devices they represent, or on gateway or proxy devices capable of providing a suitable execution platform (e.g. analogue-to-digital converter devices), such that their execution is closest to the true source or sink of media content. This means that the lifetime of virtual devices and endpoint objects will likely depend on the multimedia device it executes on and represents. Unlike device and endpoint objects however, a flow control and management object does not have a physical equivalent—it is a local representation of the state of distributed processes. Consequently, its existence is not physically or operationally bound to any one particular device. The control and management object need not be created, co-located, nor managed by the same party that invokes it. Depending on application requirements, control and management objects may be transient or persist beyond its service lifetime as well as that of its devices and endpoints. They can be accessed locally or remotely and can have local library or remote service styles of implementations. Furthermore, control and management objects (i.e. their implementations) can be modified without affecting the rest of the components in the system or how they interact. Solutions are anticipated to be application and policy specific.

The fact that flow control and management objects can be run and managed independently of peer devices is of particular importance for production studio environments. In order to create a composite program, the production process relies on a predictable set of flows and streams with a predetermined layout or topology for routing media content. Because control and management objects abstract flows and streams, the production studio's layout can be determined in advance by specifying the set of control and management objects that model the possible connection points into the production studio (e.g. using third-party establishment under the direction of the editor). A predetermined set of flow and stream control and management objects (possibly involving intermediate interconnected components) can be used to specify a system of fixed paths (similar to a pipeline for transporting media content) for routing content streams within the production studio. Once instantiated and activated, the set of flow control and management objects represent the *potential* bindings within the production studio.

Figure 5 shows an object model of the production studio's audio plane. The physical connections established between the audio devices in the production studio have been replaced with appropriate audio flow control and management objects (c.f. Figure 4). These flow control and management objects form the principal components of the production studio's management application, which is responsible for keeping track of all flow control and management objects (i.e. flows). It should be noted that the associations shown in the figure represent object relationships, not flow relationships. The flow relationships are represented by the flow control and management objects themselves. Also, the model illustrates only one of many possible scenarios in which flow control and management objects may be created and used. Furthermore,

the flow objects highlighted in the figure are elementary point-to-point and point-to-multipoint flow components (i.e. based on unicast and multicast transports respectively). More complex flow topologies (e.g. multipoint-to-multipoint) can be constructed from compositions of these elementary flows, allowing management clients to perform control and management operations on the composite flow or on its constituent flows. This provides for a more controlled and predictable approach to studio planning, set-up, flow selection and handling, as well as the overall management of the production process during normal operations. Solutions are anticipated to be application and policy specific.

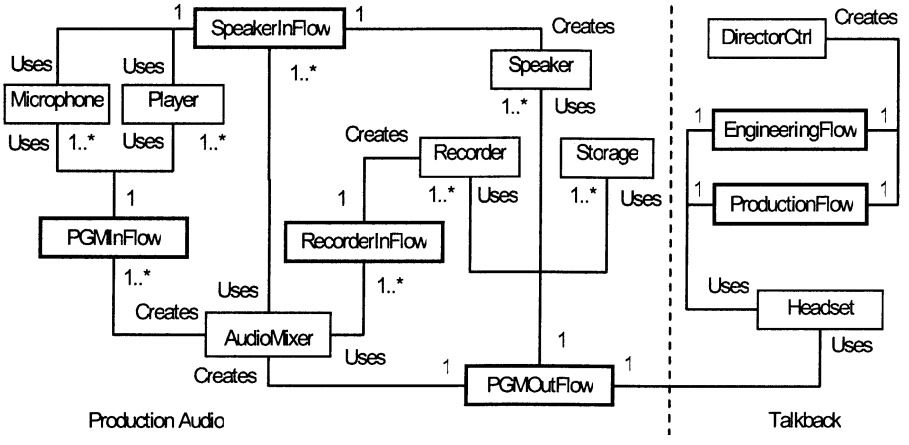

Fig. 5. An integrated object of production studio audio components.

4 Studio Dynamics

There are two main aspects to control and management within the IP broadcast studio environment. The *automatic* object relationship establishment at studio set-up between devices (i.e. studio-on-demand), and the management of the studio functionality during normal operations, expressed in terms of static relationships between the devices in the studio—delivering what streams where (i.e. program production). The flow (and stream) abstraction allows the manager to focus on content management rather than device management. Since the production studio's layout is established by a set of flow control and management objects, and the same set of objects represent both potential as well as established flows (i.e. bindings), the studio set-up and its management thereafter involves the device or studio manager locating the relevant flow control and management object, and invoking one of two sets of services—flow establishment, or flow control and management. One of the fundamental requirements of any distributed processing environment (DPE) is the ability for a component to locate other components with which it can interact. All the necessary information required to invoke a CORBA object is encapsulated in its object reference. Since object

references are typed, clients are able to determine the services provided by a particular component, as well as the mechanisms for using those services, via the IDL typing system [9, 10 and 11]. Because the flow components are structured as CORBA objects, there does not need to be a special way of finding control and management objects representing a particular flow. Finding the components is orthogonal to using the services they provide. In other words, clients invoke operations on a CORBA object the same way regardless of how they obtained its object reference. Therefore, flows can be identified and located using generic CORBA facilities or application-specific directory services. For example, an implementation of the *Factory* design pattern [6] can be used to supply a predictable set of control and management objects with varying execution policies (e.g. lifetime, ownership, activation, etc.) and types. Such location and execution transparency, scalability, and robustness are the established traits of CORBA.

4.1 Studio Set-Up

Each flow control and management object supports operations that perform the necessary negotiation between flow devices and endpoints to set-up the flow, which it then represents. The role of the control and management object in the flow establishment process can best be described by the *Mediator* design pattern [6]. The intent of this pattern is to define an object (i.e. the control and management object) that encapsulates how a set of objects (i.e. the devices and endpoints) interact. A mediator promotes loose coupling by keeping peer objects from referring to each other explicitly, and allows their interaction to vary independently.

During the studio set-up phase, components in the studio try to discover and identify suitable peer components and, if required, determine their compatibility and establish the necessary control or consumer relationship with the service (content or otherwise) provider entities. The target "studio-on-demand" provided by the studio DPE requires mechanisms to rapidly integrate sources such as cameras or microphones into the production environment [8]. Devices and endpoints do not interact or negotiate with their peers directly. That is, they do not invoke each other's operations, except for the initial call on a peer to create a binding using first-party flow or stream establishment, and during the configuration phase. Instead, flow control and management objects (acting as mediators) implement and execute the peer negotiation mechanisms and protocols (i.e. determining peer compatibility and negotiating protocols, formats, QoS, security, etc.). The binding process is initiated by invoking the "connect-dev" operation on the control and management object with the target devices or endpoints as parameters. The control and management object conspires with peer devices and endpoints to create the binding. Interoperability between components is facilitated by the set of rules specified in [1] that implementations of the interfaces must follow. Two approaches to flow establishment are possible—first-party and third-party flow establishment. It is important to realize that first-party and third-party flow establishment does not refer to the co-location of the control and management object, but to the initiation of the flow establishment process. First-party flow establishment is initiated by an autonomous peer device or endpoint, and third-party

flow establishment is initiated by a client that is not one of the participating devices or endpoints (e.g. a management application).

Since the target flow can be identified as a CORBA object, the integration of sources and sinks simply entail the particular source or sink locating the correct flow control and management object and invoking its "connect-dev" operation to be "automatically" integrated into the existing studio set-up. The term automatic used in the establishment process refers to the fact that no influence external to the control and management architecture (e.g. from the studio manager) is required for peers to establish a flow relationship. The rapid integration of sources and sinks is possible because the flow establishment process is standardised and encapsulated by flow control and management objects. This fact is particularly significant in the case of multicast flows. Potential sinks can simply identify the target flow to connect to and not the device or address to listen on. The necessary information is conveyed to sinks by the multicast flow's control and management object. The IDL typing system ensures that flows can only be established between compatible devices and endpoints.

4.2 Studio Normal Operations

Once the binding is established, the control and management object that previously represented the potential flow now represents the established flow and supports operations for starting, stopping, and destroying that flow. The identifiable flows established during studio set-up result in a topological layout of paths that route content streams from sources to sinks for the purpose of program production. These flows are managed or processed under the control of the editor during normal studio operation. The devices and their associated endpoints participating in a flow can be registered centrally or located via their respective control and management object. The role of the control and management object as a flow can be fully described by yet another design pattern—the *Facade* design pattern [6]. The intent of this pattern is to provide a unified interface to a set of interfaces in a subsystem. It defines a single higher-level, simplified, and localized interface (the flow) to the more general facilities of a distributed subsystem (in this case, the flow participants). This helps to reduce the complexity of the subsystem and makes it easier for clients to use (i.e. control and manage).

During normal operations, management clients simply locate the control and management object that represents the target flow. Once located, the control and management object allows the management client to control the transfer process, modify its QoS profile, or destroy the binding (i.e. close the stream and tear down its transport connections) when it is no longer needed. Since the flow has a distributed state, the control and management object maintains references to all of its participating endpoints. This allows it to exercise control over the transfer of content by issuing requests to participating endpoints whenever changes are requested. As with flow establishment, the management application does not interact with devices and endpoints directly, nor manage them individually. Hidden to all its management clients, a control and management object interacts and conspires with its participants to control and manage the media transfer. The control and management object in its role as a facade is therefore the primary client of peer devices and endpoints.

5 Design Considerations

The flow abstraction is semantically closer to its content than the stream abstraction. The stream abstraction is only relevant in the aggregation of multiple flows. It is important to distinguish between a flow relationship and its content relationships. The use of the general term flow refers to the transport of media content of a single format (e.g. audio, video) between producers and consumers. Although a flow is generally considered as a single entity, it may be composed of multiple content streams. Each producer contributing to a flow defines a content relationship between the producer and its consumers in that flow relationship. In other words, a flow may carry content from multiple sources, raising the issue of source synchronization and flow topology. It is the contractual responsibility of a flow control and management object to define and represent the content relationships between the set of producers and consumers. This section discusses the correlation between a flow abstraction and its content streams, and how this affected the design and implementation of our prototype.

5.1 Handling Multiple Content Streams

Each content stream instance within a flow is identified by its synchronization source. The content stream that is transferred from a producer to its consumers consists of media objects or presentation units (e.g. an audio sample, a video frame) that must be presented according to the temporal relationships that existed during the capturing process of the media objects [12]. Because the presentation units of a content stream are temporally-related, its source—or more precisely its synchronization source—is used to distinguish it from other synchronization sources (including those from the same device or endpoint) so that consumers can identify, synchronize, and present the media objects of the content stream in a timely fashion. Additionally, consumers must identify the type of data being received, detect packet loss, and determine the order in which the arriving packets should be presented. In the Internet, synchronization of information streams is derived primarily from RTP/RTCP algorithms [4].

As a flow may involve more than one producer, the contract between the flow control and management object and its participants are such that, consumers bound to that flow will receive content 'injected' into the flow by all producers bound to that flow. Content and flow relationships share all but two characteristics—their type and cardinality[1]. The maximum cardinality of the relationship defined by a content stream stipulate that a content stream always originates from one producer, but more than one content stream may terminate at a consumer. That is, from the producer's viewpoint, a flow relationship is synonymous with a content relationship. In contrast, from the consumer's viewpoint a flow relationship may translate to multiple content relationships.

The handling of multiple content streams is particularly important where multicasting is used. At the protocol level, multicasting minimizes the bandwidth and complex-

[1] For each role in a relationship type, the *minimum* and *maximum cardinality* specifies the minimum and maximum number of relationships respectively, in which a role will participate.

ity required to send information to multiple hosts on a network. A multicast group address represents a flat virtual network without any capability to subset group members logically or by physical location/distance in the underlying network. Multicasting in the Internet relies on establishing a well-known shared network-layer address space (so-called Class D addressing in IPv.4 and distinguished by a specific prefix in IPv.6) in which messages, or streams of information, are routed from one source to multiple receivers that recognize that address [8, 13]. The transport or routing of multicast streams is not as important in the discussions here as what happens to the different stream of packets arriving from different producers at the consumer. Content streams transported on a single multicast address are received by every consumer listening on that same address. There are two approaches to handling multiple content streams. Each content stream received by the consumer may either be handled (i.e. processed and presented) separately, or a consumer may support mixer operations directly to create a new compound content stream. The mixing of multiple content streams however is typically performed by a dedicated mixer device. The *audio mixer* plays a central role in the management of multiple audio content streams (as can be seen in Figure 4). As Figure 6 illustrates, it is an intermediate system that receives data packets from one or more (contributing) sources, possibly changes the data format, combines the packets in some manner and then forwards a new data packet. Since the timing among multiple input sources will not generally be synchronized, the mixer will typically make timing adjustments among the incoming content streams and generate its own timing for the combined content stream. Thus, all data packets originating from a mixer will be identified as having the mixer as their synchronization source.

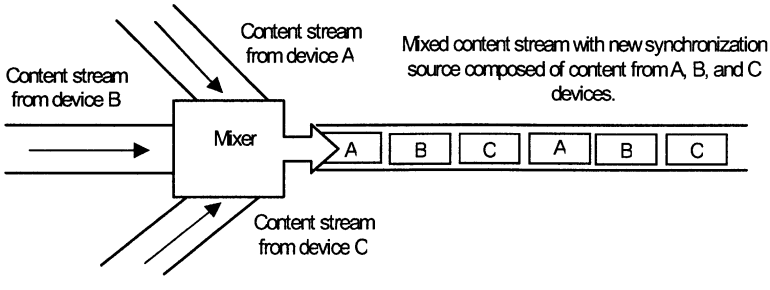

Fig. 6. The mixing of content streams from multiple synchronization sources.

Although no intermediate mixer device is necessary where mixer operations are supported by consumer devices directly, the use of dedicated mixer devices may be desirable for reasons of compatibility, performance, and controllability. Consumers inherently support at least one content stream. However, not all consumers provide capabilities for handling multiple content streams. Removing this responsibility from consumers ensures that irrespective of whether the flow is composed of one content stream or multiple content streams, the consumers remain compatible. Assigning the responsibility for mixing multiple content streams to a dedicated mixer device means less processing load on consumers. The data packets received can be processed and

presented immediately, since all data packets originating from a mixer are identified as having the mixer as their synchronization source. Additionally, the production activities will require selective mixing of multiple content streams. A dedicated mixer device may provide greater flexibility over the mixing and delivery of multiple content streams with respect to the production studio's overall management. The mixer's capability to selectively combine input content streams can be used to provide a more predictable and controllable means of routing multiple content streams.

5.2 Flow Topologies

The flow relationship defines a topological layout that determines what content streams are delivered where. The support for multiple producers by flow control and management objects means that point-to-point, point-to-multipoint, multipoint-to-point, and multipoint-to-multipoint flow configurations are supported directly. The number of producers involved in a flow does not only affect the handling of content streams, but it also influences the flow's resulting topology. The content streams established between producers and consumers describe a maximal bipartite[2] graph where the direction of content flow is from producer nodes to consumer nodes. The graph in effect describes the flow's topology.

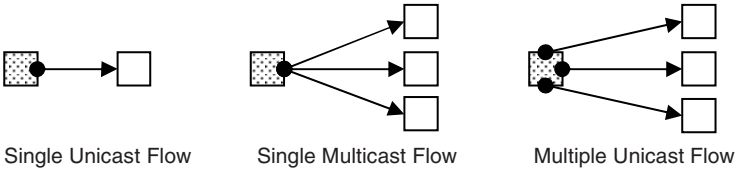

Fig. 7. Point-to-point and point-to-multipoint topologies.

As with the handling of content streams, the implementation of topologies involving only one producer (i.e. point-to-point and point-to-multipoint) is straightforward; since the data received by consumer(s) constitutes only one content stream and as such no mixing of content streams is required. Naturally, point-to-point and point-to-multipoint flows can be based on simple unicast and multicast connections respectively. Notice that a point-to-point topology is a special case of a point-to-multipoint topology where there is only one consumer. As such, the point-to-multipoint flow need not necessarily use a multicast connection. If multicasting is not available, or would be unsuitable (e.g. when the content stream received by each consumer needs to be controlled and managed separately), multiple unicast connections may be used instead. In this case, the sender device creates a producer endpoint for each unicast connection supported, as shown in Figure 7. This implementation detail is hidden from clients.

[2] A bipartite graph is an undirected graph in which the set of nodes can be partitioned into one of two sets, where all edges go between the two sets.

The support of point-to-point and point-to-multipoint topologies by a flow control and management object is implicit. These topologies form the basic components from which more complex topologies can be constructed, which include multipoint-to-point and multipoint-to-multipoint topologies. The support of multipoint-to-point and multipoint-to-multipoint topologies by flow control and management objects is optional. A flow control and management object may refuse to support more than one producer. Figure 8 depicts a multipoint-to-multipoint flow that uses an intermediate mixer device to mix the content streams from multiple producers and multicasts the composite stream to multiple consumers. Producers are connected to the mixer device by unicast connections. Notice also that the multipoint-to-point topology is a specific case of the multipoint-to-multipoint topology where only one consumer receives the mixed stream.

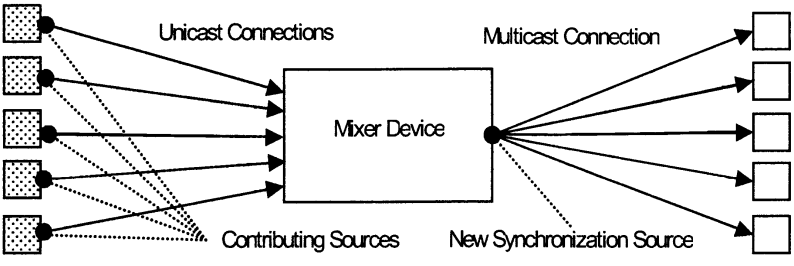

Fig. 8. A mixer device in a multipoint-to-multipoint flow connection.

6 Prototype Implementation

In-line with our objectives in using standard technologies, our prototype implementation of the audio flow control and management components is based on the Java Media Framework (JMF) [5]. JMF provides a unified architecture and messaging protocol for managing the acquisition, processing and delivery of time-based media data. In particular, it provides high-level abstractions of the data sources, sinks, media handlers, processing components, controls, and user interfaces. The framework also defines components for the transmission and reception of audio and video data using RTP [4]. JMF provides facilities for performing the actual processing and transfer of media content, but it does not define interfaces for external (i.e. third-party) control and management of streams. By wrapping JMF components in flow control and management interfaces, the JMF streaming process becomes controllable and manageable. The mixing of multiple content streams is implemented by gathering all the incoming streams under one data source and creating a new outgoing stream with a new synchronization source (SSRC).

Figure 9 illustrates a simple example deployment scenario of flow control and management objects. Device and endpoint objects execute on or closest to the devices. Flow control and management objects exist independently of devices. From the manager's viewpoint, it is irrelevant whether the device is controlled via a gateway or

a management agent (See section 3.1). All that matters is that the interfaces defined by the OMG Audio/Video Streams Specification are supported at its point-of-presence. Every other implementation detail is hidden from the manager. The management station is responsible for managing all flow control and management objects within the production studio, which are the objects used to facilitate the integration of devices, and the management of the streaming process thereafter. For example, to establish a flow between the microphone and audio mixer (the hub where flows are mixed), the manager obtains a reference to the target flow control and management object (a potential flow) and calls its "connect-dev" operation passing the microphone and audio mixer references as parameters. The flow control and management object mediates between the device endpoints to establish the binding (Mediator interactions). Once the binding is established, the flow control and management provides the manager with the ability to control the distributed endpoints through a unified interface (Facade interactions).

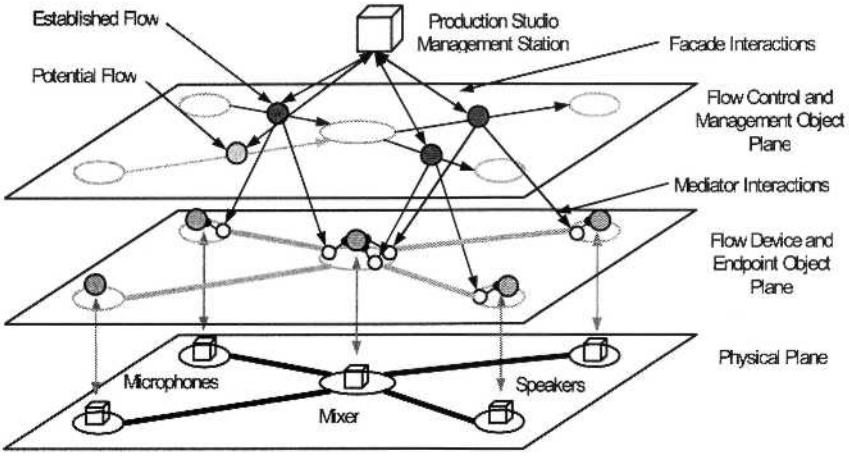

Fig. 9. Conceptual model of a simple example deployment scenario.

7 Conclusions

In this paper, we investigate the applicability of the Audio/Video Streams Specification to a production studio environment. In particular, we describe the design and implementation of components for the control and management of audio flows within an IP broadcast environment for the purpose of program production. The production studio's layout is modelled as a set of flow control and management objects that identify the possible bindings into the studio environment. We emphasize the roles of a control and management object in the OMG Audio/Video Streams Specification's architecture using two design patterns. As a mediator, it provides the necessary services

for smoothly integrating peer devices and endpoints into the production studio during the set-up phase. As a facade, it provides services for controlling and managing participating endpoints during the normal program production process. The complexity of binding, controlling, and managing the distributed endpoints of flows is hidden from the studio manager; thus removing the need for the studio manager to interact with devices and endpoints individually. The studio manager can focus on content management instead of device management. The correlation between a flow control and management object and its content, as well as the need for mixer operations and how this affects the flow topology is also described. Because the current model is based on full-profile components, it can be extended in future work to include additional flow types and subsequently incorporated into the existing arrangement. Our prototype implementation is based on the Java Media Framework, which provides support for the delivery of time-based media via the Real-Time Transport Protocol. Being CORBA-based, the architecture allows for considerable flexibility as regards to the implementation, distribution, and management of control and management objects, particularly in the design of ad-hoc solutions for flow and stream establishment. Because of the highly relational nature of the application, the manager needs a management system that heightens the observability and controllability of the network, in real-time. For this reason, we are developing a virtual world supported by a CORBA-based graph drawing system for the visualization and control of the production studio.

References

1. Object Management Group: Audio/Video Streams Specification v1.0. OMG Domain Specifications, Telecommunications, 00-01-03, January (2000)
2. Object Management Group: The Common Object Request Broker: Architecture and Specification v2.6. 01-12-35, December (2001)
3. Mungee, S., Surendran, N., Schmidt, D. C.: The Design and Performance of a CORBA Audio/Video Streaming Service. HICSS-32 International Conference on System Sciences, minitrack on Multimedia DBMS and the WWW, Hawaii, January (1999)
4. Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V., RTP: A Transport Protocol for Real-Time Applications. Internet Engineering Task Force, RFC1889, January (1996)
5. Java Media Framework API Specification, Version 2.0, FCS. 10 March (2001)
6. Erich, G., et al.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley (1995)
7. Object Management Group: Property Service Specification v1.0. 00-06-22, April (2000)
8. Munro, A. (Editor) et al.: Studio Production Networking and the Internet – State of the Art Report. Technical Report D1, PRONET LINK project, February (2000)
9. Orfali, R., Harkey, D.: Client/Server Programming with Java and CORBA, Second Edition. John Wiley & Sons, Inc. (1998)
10. Hoque, R.: CORBA 3: Developing Industrial-Strength Client/Server and Web Applications. IDG Books Worldwide Inc. (1998)
11. Siegel, J. (Ed.): CORBA 3 Fundamentals and Programming, 2nd Edition. Wiley (2000)
12. Kaleshi, D., Munro, A.: Studio Production Networking and the Internet – Synchronization Considerations. Technical Report D7, PRONET LINK project, September (2000)
13. Ooms, D., Sales, B., Livens, W., Acharya, A., Griffoul, F., Ansari, F.: Overview of IP Multicast in a Multi-Protocol Label Switching (MPLS) Environment. Internet Engineering Task Force, RFC 3353, August (2002)