An Application Framework for Multimedia Communication

Stefan Frey and Daniel.P.Ingold
ETH Zürich

Abstract

Multimedia workstations will not find acceptance unless presented with attractive and easy-to-use applications which maintain a consistent user-interface. Since creating 'nice' applications is quite demanding and their lifetime in the rapidly evolving area of communications pretty short, a framework on which to base might be of some use:

This paper reports on the architecture and implementation of an object oriented framework with special emphasis on multimedia communication.
We introduce Device and Channel objects to control media-streams and to provide their signalling. Additionally we define a Service object as a base for multimedia communication between applications and we extend the existing view concept by new standard MediaViews to present and control audio and video streams.

The development of this framework is part of the ETHMICS project at the Computer Engineering and Networks Laboratory, ETH Zurich/Switzerland [1].

Introduction

The ETHMICS project will provide a workstation platform for integrated multipoint multimedia communication. Special hardware is under development to support broadband communication and integrated video views [2]. The network offers isochronous media-streams of guaranteed bandwidth and it delivers separate channels for each media, one of the channels supporting standard computer communication functions, which are used to control all other media.
Parallel to this hardware development, we work on an object oriented framework, featuring multimedia objects able to present any combination of audio, video and computer data via communication channels which can easily be set-up.
This Framework deals with problems discussed in [3] and [5]. While [4] uses communication objects on operating system level, we try to use them on a more general level within our framework.

Our work is based on the Macintosh™ operating system and MacApp™, a framework covering conventional computer media in an event driven paradigm.
In the following sections we introduce a hierarchy of dedicated objects which manage different kinds of data streams on different system levels. We also describe extensions to MacApp™ objects for displaying and manipulating full motion picture, audio and mixing of this data with conventional computer data.

To keep the system responsive, the workstation platform processes and forwards the media-streams (e.g. video) in hardware. The framework provides also objects that control these hardware or software devices and that direct the data streams within the system.

Framework Architecture

The proposed architecture addresses the following problems:
- Control Abstraction for Multimedia Devices
- Signalling among distributed Devices
- Multimedia Services Objects
- Media Presentation and Control Views.

We introduce a hierarchy of objects which controls the specific Multimedia Devices and which carries out the Signalling among them. These objects have a common origin, which we call 'Stage'. We manage each media stream by a tree of 'Stages' with the stream's source at its root and the data flowing out to the leaves. One tree controls the Multicast of a single media [Figure 1]. For bidirectional communications with multiple media, divers trees will be overlapping crosswise. This should provide enough flexibility to map most physical configurations on their controlling objects. Physical configurations may include custom hardware as well as QuickTime devices.

The 'Stage' defines means for hierarchical notifications and requests within the tree; e.g. they will allow for event delivery and delay estimation. Based on a 'media-type' field in each 'Stage', an application can decide which resources may be tied together to form a chain of stages, a so-called path.
Every path (from root to leaf) is composed of an alternating sequence of Devices and Channels (both descendants of the Stage).

A set of Devices objects may be assembled to form a multimedia 'Service' which is the building block for communication in distributed multimedia applications.
Presentation and Control of media, like full motion video, is implemented by descendants of the 'MediaView' object which, in turn, stems from the 'View' object (the basic visible entity of the underlying framework). Like any basic view of the framework these views can easily be incorporated in an application. Among other MediaViews we put a video-image with its user-handling (scalable, layered, grabbing) and an audio control (volume, mixing) at the programmer's disposal.

Devices and Channels

The Device object specifies some common properties of physical resources such as input, output or intermediary conversion units. A device basically connects an incoming channel to an outgoing one and inherits the stages' ability to forward events between the two.

It defines hooks for
- bandwidth demands/credits
- (restricted) blocking
- jitter measuring/compensation

Device objects are tied together by Channels to form the tree of Stages.
The Channel object encapsulates a 1-to-n propagation of a single media. It sets up the link, requests its bandwidth and forwards information concerning its synchronisation. Fine grain synchronisation (simultaneity) will be dealt with in the context of Services. Coarse grain synchronisation (of temporal relations) exploits the Stages' notification mechanism as a vehicle for annotations. The succession of events transmitted and received by the application may be recorded as a storyboard for presentations.

In the implementation, each Channel object, connecting multiple sites, is in fact distributed (instantiated on all sites involved) to implement the inherited Event propagation mechanism. To carry out the protocol used to open a specific physical channel is the task of a specialised descendant overwriting the methods involved.
We use persistent Stages for local resource management and configuration: Parts of the communication paths are predetermined by the resources which are physically available. These sequences of stages are allocated as persistent objects and their free end-points (devices) are hooked into the set of available local resources.

Distributed Devices

For us, an object is a state machine with fields holding state information and with methods performing state transitions. The object oriented approach provides a powerful tool for specifying extensible modules, but how can these be used to build a distributed application?
We are using events for connecting the objects among different applications and remote machines, since there is no other commonly used way of interaction. The event propagation is carried out by standard process-to-process communication.

We differentiate the following two basic kinds of event handling: We use lower level system events for communication set-up, and highlevel events to address the user (video annotations, conference management, pointing). Highlevel events are dispatched through the applications event queue and event handler, system events, however, asynchronously trigger a method (like a thread) in order to be efficient [Figure 2].
In an event driven system, the objects are extended to change their state in reaction to particular events.

Multimedia Service Object

Service objects are network visible access points for multimedia information exchange between applications. Depending on what media it is composed of, a Service administrates a set of different devices. Service objects are logic entities to build distributed multimedia applications. They are

accessible in the network through their workstation and application names and contain directory methods which allow remote services to find out about functions which this service can provide. If common functions between services exists, these services can work together.
Channels attached to common source and destination Services (working in parallel) attempt to assure sufficient simultaneity of the various media belonging together (time continuous composition).
There are numerous ways to support complex interactions between applications - like majority voting, distributed queues or group scheduling. We propose to hide these interactions, usually called 'Groupware Support', within specialised services. Access to their devices would trigger the common channel's voting or queueing algorithm.

User Interface: MediaViews

The Device in the Service that receives a video stream will be a network socket at the head of a chain of Stages: Network socket -> Channel -> image decompressor -> Channel -> video scaling/ overlay unit.
The last device in such a chain is controlled and presented by a specialised MediaView, in this case by a 'VideoView':
The VideoView defines the destination area for the overlay unit and allows the user to scale and move the motion image the way he is used to work with any other window on his screen.
The Video-View itself handles the local Mouse events used to drag it around, used to focus on a part of the image and to scale back to a full sight as well as pointing on an image location so that all other devices connected to the same source device get a proper indication.
A VideoView can print out a selected part of the image or 'copy'-'paste' it to any other application.
The image size chosen (and the resolution required) is propagated back to the source. On its way, this request may be tuned down to the available bandwidth, such that the source is aware of what bandwidth to deliver.

An audio output device usually does not need a visual presentation of its data. The MediaView attached to it, the so-called 'AudioControl' object, allows for all usual audio manipulations (e.g. volume slider) as well as for mixing multiple channels in conference situations, thereby indicating the active speakers name. If the AudioControl view is attached side by side to a Video-View of the same Service, panorama control of the loudspeakers will reflect the position of the views on the screen.
If a service offers multiple sound channels, e.g. for a film with a separate commentary, both receiving audio Devices may be attached to the same (mixing) AudioControl MediaView or handled separately.

Compiling a Sample Application

A sample application like a picture phone instantiates a service object supervising video and audio devices, possibly also providing a shared pointing device and a blackboard device for synchronous exchange of textual and/or graphical information.

To do this, the application must search the set of available devices for the device-types needed and attach them to its service object.
Based on the user's needs, the application will then present the user with a standard 'Browser'. He may then select among remote applications with services that can exchange information with its own service object. That means that the service describer (list of device-types) of the local Service intersects with the service describer of the remote service.
For each device of the service chosen, the local service assigns a channel object to the corresponding local device. It is then the channel object's task to get the media stream delivered to the sink, i.e. to request the physical channel [Figure 3].

The interface of such a sample application would present the user with video-views and audio-controls. These MediaViews might be placed in a common window, which a user can manipulate as usual.

Summary

In the previous sections we introduced an object hierarchy for multimedia communication between applications on multimedia workstations. We extend existing visible entities by methods of presenting and controlling video- and audio-streams. The framework proposed provides the application developer with re-usable code for service access, with a consistent graphical interface and an exchangeable 'black box' for link set-up.

References

[1] A. Kündig, "Multimediakommunikation" Eidgenössiche Technische Hochschule Zürich, Jahresbericht 90

[2] U. Röthlisberger, D. Ingold "An Architecture for an Advanced Multimedia- Workstation" Proceedings Multimedia '90

[3] Thomas D. C. Little, Arif Ghafoor "Network Considerations for Distributed Multimedia Object Composition and Communication" IEEE Network Magazine, Nov 90, Vol. 4, No. 6

[4] W. H. Leung et al. "A Set of Operating System Mechanisms to Support Multi-Media Applications"
1988 International Zurich Seminar on Digital Communication

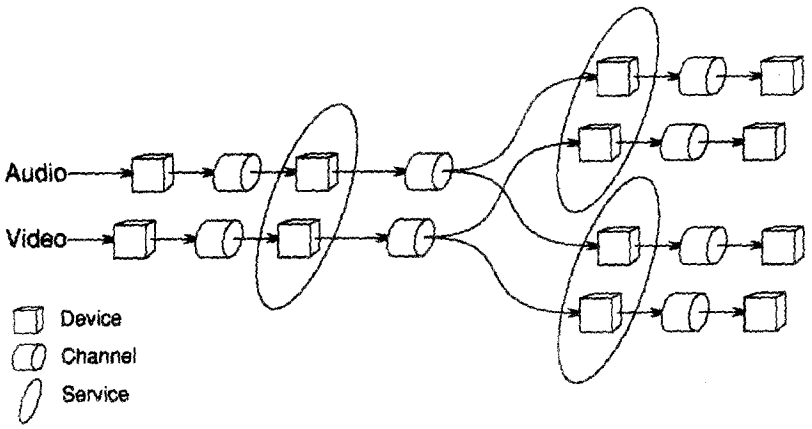

Figure 1: Multicast

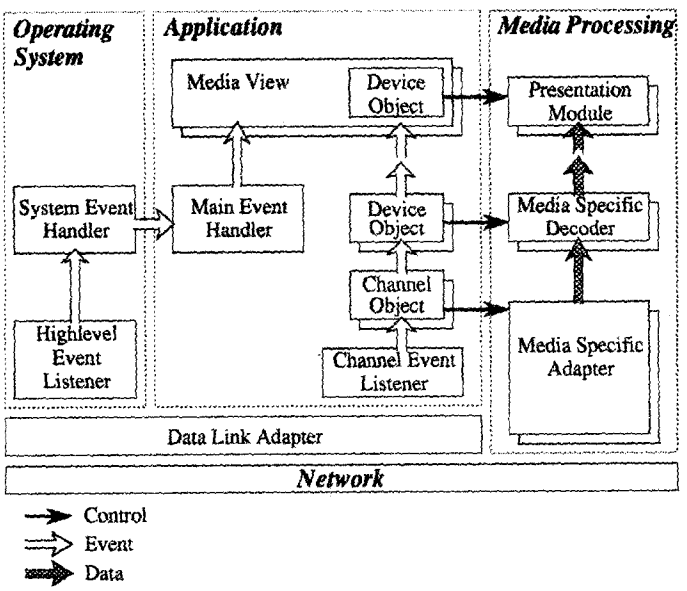

Figure 2: Event Delivery

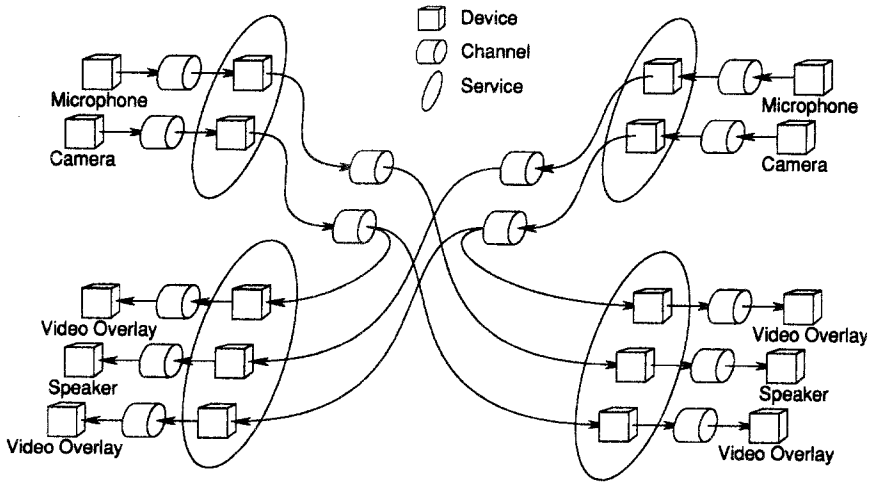

Figure 3: Picture Phone Application