Proceedings of the
4th Open Workshop on High-Speed Networks, Brest (France), September 7-9, 1994

Development and Control of Distributed Multimedia Applications

Tobias Helbig
University of Stuttgart / IPVR, Breitwiesenstr. 20-22, D-70565 Stuttgart, Germany
Phone: +49-711-7816-275, Fax: +49-711-7816-424
E-Mail: helbig@informatik.uni-stuttgart.de

ABSTRACT: Distributed multimedia applications make use of high-speed networks to transmit data units. Processing of multimedia data is done at end-systems. Appropriate support is required to provide for integrated processing and communicating of multimedia data streams and for efficient development of distributed multimedia applications. It is given by a middleware layer which uses services of operating systems and networks and enriches them by protocols and mechanisms to offer multimedia system services including multimedia communication, resource reservation, synchronization and configuration management. By that, the gap between system services of general purpose operating systems and the requirements of developing distributed multimedia applications is closed.
The *CINEMA* (Configurable INtEgrated Multimedia Architecture) system is such a middleware layer which offers abstractions to develop arbitrarily structured multimedia applications and provides protocols and mechanisms to control them. Its multimedia system services are accessible via easy-to-use interfaces.

1 INTRODUCTION

High-speed networks provide the means to transfer high-bandwidth, time-sensitive data streams among end-systems. They are a prerequisite for developing distributed multimedia applications. However, multimedia communication and processing goes beyond networks. Most of the data processing (production, transformation and consumption of data units) as well as its timing is handled in end-systems. To provide seamless quality of service support and data stream control in a truly end-to-end fashion, the need for the integration of network, transport and operating system services arises. In such an integrated multimedia environment, guaranteed quality of service is delivered to clients of a distributed computing system. Furthermore, clients are supported in developing distributed multimedia applications and in controlling multimedia data streams encompassing more than a single transport system hop by appropriate system services and abstractions.

Integration of multimedia processing into conventional computer systems as well as support for development of distributed multimedia applications are addressed in several projects. In SUMO [CBRS93], the Chorus micro-kernel is extended to support continuous media and quality of service control in an operating system. The focus, however, is on operating system issues, not so much on high-level abstractions for developing and configuring distributed multimedia applications. The problem of configuring distributed applications by using software components interconnected by linked ports is addressed by Conic [KrMa85] and REX [MKSD90], however without focussing on real-time features of multimedia processing. Specific abstractions for controlling multimedia data streams have been proposed as well. Some of them apply to non-distributed environments only (e.g. QuickTime [Appl91] or IBM's Multimedia Presentation Manager [IBM92]), while others are tailored to specific configurations (e.g. ACME [AGH90] and Tactus [DNNR92]). Mainly, they are extensions of network window systems to support streams of digital audio and video data. General requirements that should be met by architectures supporting distributed multimedia applications are specified in the Request for Technology [IMA92] of the Interactive Multimedia Association. A response to this request made by companies [Hewl93] proposes abstractions to structure and control distributed multimedia environments while using multi-vendor processing equipment. The proposal assumes generic multimedia processing elements producing and consuming multimedia data via ports that are associated with formats. However, the nesting of processing elements is not supported and, although grouping is used to handle resource acquisition, stream control and the specification of end-to-end quality of service, no means to specify synchronization relationships between data streams are provided.

The *CINEMA* (Configurable INtEgrated Multimedia Architecture) system, currently under development at IPVR, is a development platform that provides various multimedia system services, including services for communication, synchronization and resource management. *CINEMA* is built on top of operating and

transport systems and provides powerful programming abstractions, which simplify the development of distributed multimedia applications. It aims at closing the gap between the functionality provided by networks and operating systems and the requirements when creating distributed multimedia applications. Applications are composed of multimedia processing elements, so-called *components*, that produce, transform or consume multimedia data. Components are linked to arbitrary complex *flow graphs*. In order to convey multimedia data streams, *sessions* with a certain quality of service are set up. Establishing a session results in the reservation of appropriate system resources (CPU, memory, bandwidth, ...). This is a prerequisite for transmission of multimedia data which is controlled by *media clocks*. Media clocks enable a client to specify temporal properties of continuous data streams and to control the playout of data units. By specifying relationships among media clocks, synchronization requirements may be expressed.

All aspects of configuration management, resource management, and where and how synchronization is done are hidden from the *CINEMA* client. Inside the *CINEMA* system, the required resources are computed and reserved by a distributed reservation protocol. The synchronization service is dynamically configured depending on the structure of applications and the synchronization relationships between data streams. Furthermore, both the establishment of communication channels and placing of components at appropriate nodes are performed transparently. This allows the *CINEMA* system to insert additional filter or compressing components to adapt the load to limitations in resource availability and thus to reduce quality of service in a well-defined manner.

The remainder of the paper is structured as follows. In Section 2, the abstractions and concepts of the multimedia system services interface of *CINEMA* are introduced. Section 3 briefly discusses how multimedia system services may be realized. The paper is closed with a summary and an outlook on future work.

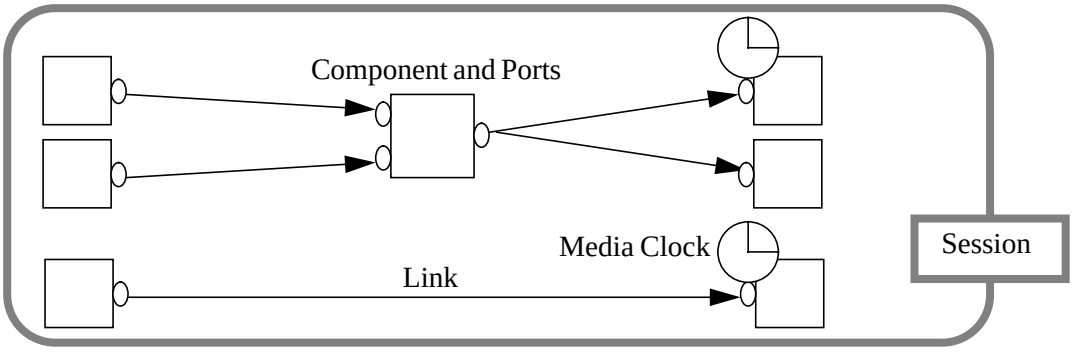

Figure 1 *Example Application*

2 MULTIMEDIA SYSTEM SERVICES INTERFACE

In the following, we describe the main abstractions and concepts of the multimedia system services interface of *CINEMA*. For more details see [RBH94].

2.1 Application Configuration and Resource Reservation

A **continuous media stream** is a sequence of data units like video frames or blocks of audio samples, each of which is associated with a media timestamp. **Components** are active entities that process continuous media streams in various ways. It is distinguished between source components, which produce media streams, sink components, which consume media streams, and intermediate components, which act both as producers and consumers. Components produce data units by writing to typed **ports**, respectively consume data units by reading from them. Ports constitute configuration independent data access points of components and separate processing and transmission functions in multimedia systems. Components are not aware of the kind of transmission channels used - either local communication on the same node or remote communication via high-speed networks.

Applications are configured by defining **links** between input and output ports of components. This allows the establishment of arbitrarily structured data flow graphs. An example configuration consisting of an intermediate as well as three source and sink components is shown in Figure 1.

While link objects are applied to define the topology of applications, **sessions** are the abstraction for resource allocation. A session may comprise multiple sink and source components and any number of intermediate components. Sessions are associated with a set of quality of service parameters. By creating a session, a client causes the *CINEMA* system to reserve the resources that are needed to guarantee the specified quality of service requirements. This is done in an all-or-nothing fashion. After a session has been established, transmission and processing of multimedia data may be started.

2.2 Timing and Synchronization of Data Streams

A media time system associated with a stream is the temporal framework to determine the media time of the stream's data units. For controlling (i.e. starting, pausing, or scaling) the flow of media streams, the concept of **media clocks** is provided. Media clocks span media time systems by that defining the temporal dimension of continuous media streams. A clock *C* is defined as *C ::= (R, M, T, S)* where *R* determines the ratio between media and real-time, *M* the start value of the clock in media time, i.e. the value of the clock at the first clock tick and *T* the start time of the clock in real-time. *S* determines the speed of the clock. Media time progresses in normal speed if *S* equals 1. A speed greater than 1 causes the clock to move faster, a speed less than 1 causes it to progress slower, and a negative speed causes it to move backwards. In the simple scenario shown in Figure 2 clock *C* controls the presentation of a video stream. The play out is started with frame 10. The play out rate is doubled when the presentation reaches frame 7000, and the presentation is halted when reaching frame 10000.

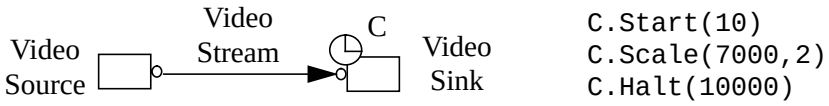

Figure 2 *Controlling a Video Stream*

A **clock hierarchy** is the basic abstraction for grouping media streams, controlling groups of streams, and stream synchronization. A number of streams can be grouped by linking their controlling clocks in a hierarchical fashion to a common clock, which then controls the entire group. Stream groups can be grouped again to groups at a higher level. In the example given in Figure 3, clock C_4 controls streams S_2 and S_3. C_5 controls the subgroup represented by C_4 as well as stream S_1, and thus all streams in the given scenario can be started, halted or scaled collectively by means of this clock.

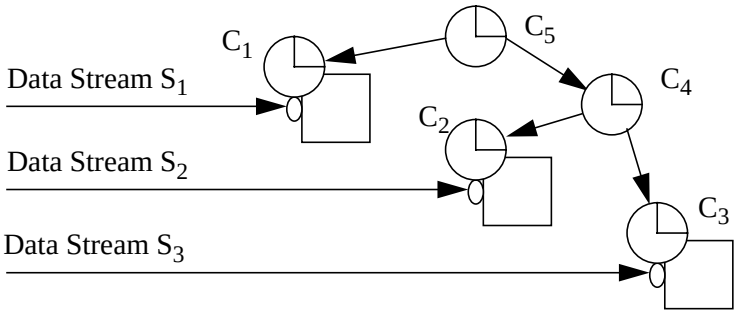

Figure 3 *Grouping of Media Clocks and Streams*

A clock operation issued at a clock not only affects this clock but its entire subhierarchy. An operation called at a clock is **propagated** in a root-to-leaf direction through the clock's subhierarchy, where it is performed at every clock in this hierarchy. In general, clock operations can be issued at every level of

the clock hierarchy. Additionally, clock hierarchies may dynamically grow and shrink even when clocks are ticking. This feature together with the capability of halting and starting individual subhierarchies is important in interactive applications, especially in those where multiple clients with their individual needs participate in the same application.

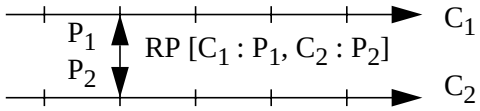

Figure 4 *Mapping Time Systems by Using a Reference Point*

Clocks provide individual media time systems which may relate to each other in various ways. Clock synchronization and propagation of clock operations is done on the basis of so-called **reference points**. A reference point defines the temporal relationship of two media time systems. More precisely, reference point $[C_1 : P_1, C_2 : P_2]$ defines that media time P_1 in C_1's time system corresponds to media time P_2 in C_2's time system. Depending on further attributes of edges in clock hierarchies, additional properties may be expressed. Operations may be delayed while being propagated, which results in delayed execution in subhierarchies. By locking clocks for certain operations, the propagation of operations is blocked for subhierarchies, e.g. enabling the shielding of client domains in multi-user applications. Furthermore, it is distinguished between **control** and **synchronization** relationships. A control relationship between two clocks enables the propagation of clock operations without synchronizing them. Typically, control relationships are defined in settings where groups of streams are to be controlled collectively and a rather loose temporal coupling of the grouped streams is sufficient. Although control hierarchies include reference points, these are considered only when clock operations are propagated to transform the operation's arguments. However, after a hierarchy has been started, its clocks may drift out of synchronization and may be manipulated arbitrarily. For example, two different subhierarchies of the same hierarchy may be scaled in different ways, or clocks in the hierarchy may be halted and continued at any later time with arbitrary start values. When defining a synchronization relationship, the synchronous progress of clocks is ensured in addition to propagation. By synchronizing their controlling clocks, two streams are synchronized implicitly. Thus, synchronization hierarchies are a general and powerful concept to specify arbitrary synchronization requirements between media streams. The structure of the synchronization hierarchy specifies which streams have to be synchronized, while the reference points in the hierarchy define how the temporal dimensions of the streams relate to each other. The system guarantees that all streams controlled by the clocks of the hierarchy are processed synchronously. When a subhierarchy is halted and started again at a later point in time, this is performed in conformance with the temporal constraints.

3 IMPLEMENTATION OF MULTIMEDIA SYSTEM SERVICES

The abstractions and services *CINEMA* offers to develop distributed multimedia applications are provided by a specialized software layer. Such a middleware layer is based on general-purpose operating and transport systems. Its implementation consists of protocols that configure applications on appropriate nodes, negotiate resource reservations, and synchronize data streams. In the following, the latter implementations are sketched and some requirements regarding operating and transport systems are discussed.

In *CINEMA*, **resource reservation** and quality of service negotiation is handled on two levels of abstraction. Application-specific quality of service specifications provided at the *CINEMA* interface, represent the presentation quality a client wants to achieve, e.g. picture size and picture rate if processing video streams. They are mapped to low-level parameters such as packet size, packet rate, or CPU utilization, which are based on parameters of individual resources. The resource reservation architecture consists of two layers. The global resource management is responsible for negotiating quality of service parameters at all the nodes participating in a session and mapping high-level onto low-level parameters by using a distributed resource reservation protocol. The local resource management reserves the resources as

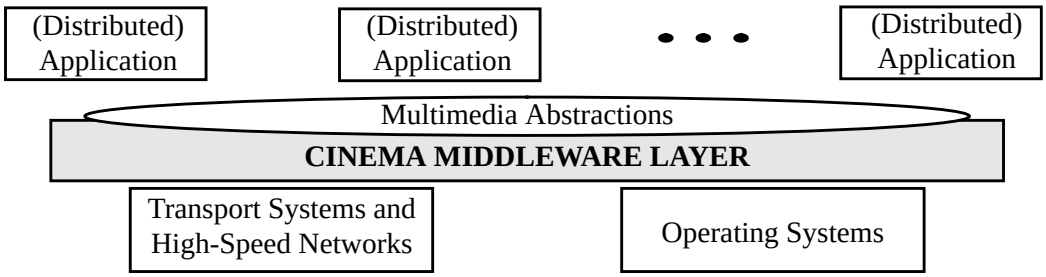

Figure 5 *Middleware Layer Architecture*

required by components or links. It therefore uses resource managers that handle individual resources at each node (e.g. memory, CPU utilization, network bandwidth).

The **general synchronization service** accessible via clock hierarchies is provided by multiple specialized flow control and synchronization modules (Figure 6). Each of them is tailored for a class of topologies, stream types (e.g. stored or live periodic streams) and system requirements, like the provision of globally synchronized system clocks. The design was chosen due to the lack of an universal synchronization protocol. It allows to make use of existing as well as self-implemented synchronization mechanisms and to adapt to different quality of service requirements by choosing appropriate mechanisms. Synchronization modules are instantiated dynamically depending on application scenarios. The specific interfaces of synchronization modules are hidden behind a stream-oriented generic module interface (GMI), which has a flat structure. Operations issued in the application-oriented, hierarchically structured clock hierarchies are analyzed, parameters are computed and then mapped on the GMI of the selected module. Each module individually performs stream control operations in distributed environments.

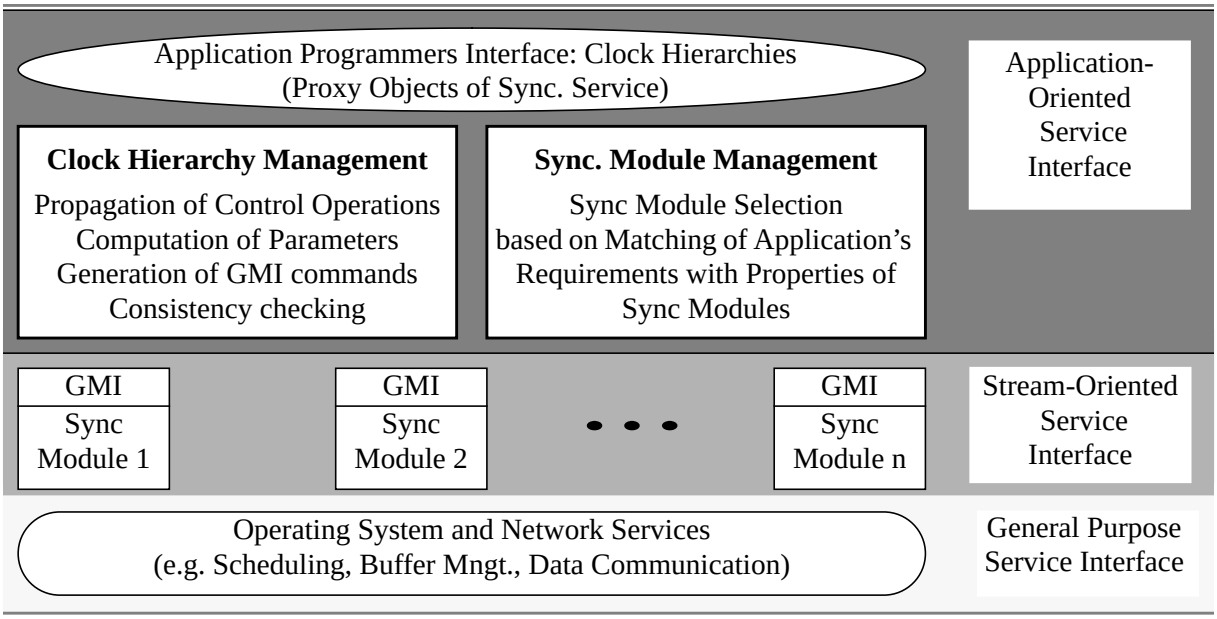

Figure 6 *Synchronization Service Architecture*

Operating systems, which *Cinema* is layered on, have to meet several requirements. E.g. the ability to schedule multiple independent multimedia processing tasks by observing real-time deadlines requires the support of real-time scheduling algorithms and preemptive threads, which the processing functions are mapped to. Current operating systems only partially fulfil this requirement. Furthermore, multimedia data is transmitted among components on the same and on different nodes. The latter involves making

use of transport protocols. When designing the *CINEMA* prototype that is based on IBM AIX and DCE as well as on the SUN Solaris operating system, it was decided to encapsulate multimedia transmission functionality into link objects. That offers two major advantages. It allows to use identical interfaces for data transmission between local and remote components which simplifies the configuration of applications significantly. Moreover, it decouples the *CINEMA* implementation from the transmission mechanism that is actually used. Due to the lack of a real-time transport system, in our current prototype link objects are based on UDP which will be replaced by a real-time protocol.

4 SUMMARY AND FUTURE WORK

The paper introduced the multimedia system services interface of *CINEMA*. It was described how components may be linked to arbitrary flow graphs and sessions are used to reserve system resources before starting data transmission and processing. With media clocks and clock hierarchies, abstractions were proposed to control individual data streams and groups of streams.

The implementation of the *CINEMA* prototype is still in progress. The first version is working. It supports a restricted set of the functionality described in this paper. For example, it is possible to establish applications in a distributed environment and to control and synchronize the flow of data units in limited configurations. Our future work is directed at extending the prototype by developing and refining protocols for resource reservation, stream control and synchronization as well as configuration management for arbitrary complex scenarios.

5 REFERENCES

[AGH90] David P. Anderson, Ramesh Govindan, and George Homsy. Abstractions for Continuous Media in a Network Window System. *Report No. UCB/CSD 90/596, Computer Science Division (EECS), University of California, Berkeley, CA*, 11 1990.

[Appl91] Apple Computer Inc., Cupertino, CA, USA. *QuickTime Developer's Guide*, 1991.

[CBRS93] Geoff Coulson, Gordon S. Blair, Philippe Robin, and Doug Shepherd. Extending the Chorus Micro-Kernel to Support Continuous Media Applications. In *Proceedings of the 4th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 49–60, 11 1993.

[DNNR92] Roger B. Dannenberg, Tom Neuendorffer, Joseph M. Newcomer, and Dean Rubine. Tactus: Toolkit-Level Support for Synchronized Interactive Multimedia. In *3nd International Workshop on Network and Operating System Support for Digital Audio and Video*, 11 1992.

[Hewl93] Hewlett-Packard Company and International Business Machines Corporation and SunSoft Inc. *Multimedia System Services, Version 1.0, available via ftp from ibminet.awdpa.ibm.com*, 7 1993.

[IBM92] IBM Corporation. *Multimedia Presentation Manager Programming Reference and Programming Guide 1.0, IBM Form: S41G-2919-00 and S41G-2920-00*, 3 1992.

[IMA92] Interactive Multimedia Association, Compatibility Project, Annapolis, MD, USA. *Request for Technology: Multimedia System Services, Version 2.0, available via ftp from ibminet.awdpa.ibm.com*, 11 1992.

[KrMa85] Jeff Kramer and Jeff Magee. Dynamic Configuration for Distributed Systems. *IEEE Transaction on Software Engineering*, SE-11(4):424–436, 4 1985.

[MKSD90] Jeff Magee, Jeff Kramer, Morris Sloman, and Naranker Dulay. An Overview of the REX Software Architecture. In *2nd IEEE Computer Society Workshop on Future Trends of Distributed Computing Systems*, 10 1990.

[RBH94] Kurt Rothermel, Ingo Barth, and Tobias Helbig. *CINEMA* - An Architecture for Configurable Distributed Multimedia Applications. *Technical Report 3/94, University of Stuttgart*, 4 1994.