2nd InternationalWorkshop on Advanced Teleservices and High-Speed Communication Architectures
26.-28.09 1994, Heidelberg, Germany

Media Scaling in Distributed Multimedia Object Services

Thomas Käppner and Lars C. Wolf

IBM European Networking Center, Vangerowstr. 18, D-69115 Heidelberg
Mail: {kaeppner, lwolf}@vnet.ibm.com

Abstract: Real-time support for multimedia streams in currently installed workstation environments has been based on resource management systems that provide mechanisms for streams with guaranteed or statistical quality of service (QoS) by admission control and resource reservation. In contrast, media scaling is a technique that dynamically adapts the load of media streams to the current availability of resources. Scaling can keep media streams meaningful to the user which would break during overload situations. Instead of interrupting the service for a stream when an overload situation is encountered, the quality of the stream is gracefully degraded when the resource load situation reaches a critical state. Since media scaling is a technique that dynamically takes actual resource load into account it can easily adapt to changing situations and has the potential to keep the system in a range of optimal load. In this article we show how media scaling can be integrated in a general system support for multimedia in order to simplify the implementation of scalable applications and support their concurrent utilization of scarce resources.

1 Introduction

Due to recent advances in computer technology, high performance workstations with digital audio and video capabilities are becoming available which leads to the integration of multimedia data with computing. This integration allows for scenarios in which computer systems support services such as video conferencing, news distribution, advertisement, and entertainment.

Due to its special nature the processing of multimedia data demands real-time support from the underlying computing platform in order to continuously transmit, synchronize, and present audio and video data streams within a distributed system.

Real-time support for multimedia streams in currently installed workstation environments has been based on resource management systems such as [7] that provide mechanisms for streams with guaranteed or statistical quality of service (QoS). After a strict admission control for the establishment of new streams, provision of guaranteed QoS is based on worst-case assumptions for resource usage, which results in resource underutilization in case of streams with variable bit rates. Statistical streams may experience breaks due to lack of resources.

In contrast, media scaling is a technique that continuously adapts the load of streams to the current availability of resources. In the case of resource overload the graceful degradation of stream quality leads to the situation where resources are shared so to allow the continuous flow of all streams. Media scaling is not depending on the upper bound of the bit rate for a given stream, thus it has the potential to support more streams than traditional resource management that offers only hard guarantees for a stream.

1.1 Related Work

Previous work on media scaling concentrated mostly on communication aspects of multimedia data. Its utilization has mainly been reported to avoid congestion in networks that can not properly be supported by resource management [4]. Jeffay et al. have developed a special queuing mechanism to adapt the bandwidth taken by video sent across packet-switched networks [4]. Fluent Technology has based a product for networked multimedia on a proprietary scaling scheme [6]. Tokuda et al. have implemented a dynamic QOS management for local area networks [1, 5], whereas our work focuses on the end-system. In principle, the scaling operations to adapt to changed system load can be implemented within the application, which forces programmers to construct their own mechanisms and leads to interworking problems between applications. In this article we show how media scaling can be integrated in a general system support for multimedia in order to simplify the implementation of scalable applications. Delgrossi et al. [2] have shown how to integrate media scaling and resource management into a multimedia transport system. In turn, the system we describe can utilize scaling-capable transport systems in order to accelerate scaling operations but does not depend on their availability.

1.2 Our Contributions

We enhance the meaning of media scaling from the network to the end-system level and show its significant value for the user. Media scaling provides two kinds of benefit. First, scaling has the potential to increase the number of streams a system can support simultaneously in comparison to systems using hard guarantees. This is due to its ability to handle and resolve a system's overload situation. Second, scaling keeps media streams meaningful to the user which would break during overload situations. Instead of interrupting the service for a stream when an overload situation is encountered, the quality of the stream is degraded when the resource load situation reaches a critical state.

Since media scaling is a technique that dynamically takes actual resource load into account it can easily adapt to changing situations and can keep the system in a range of optimal load.

Whereas end-system-transparent scaling within transport systems is constrained to scaling on a per-packet basis, scaling that involves changing the encoding of transmitted media at the sender can be based on a much finer granularity. In general changing the quality of transmitted video can relieve the congestion on resources without significantly being perceived by users. We show that distinguishing types of resources for scaling of media can be used in order to optimize the quality/load ratio.

Previous work on scaling of media streams has not considered the behavior of the system in case of several streams being scaled simultaneously, which raises the question of fairness and balancing between streams. We show how to coordinate down- and upscale operations of all scalable streams and guarantee the balanced sharing of resources.

We show how scaling can be integrated in a layer that provides system support for multimedia applications during development and run-time. The Distributed Multimedia Object Services offer an abstract interface to real-time multimedia objects. Multimedia applications create and combine multimedia objects such as audio and video streams, which can form an acyclic graph spanning several machines. Realization of multimedia objects happens in Distributed Multimedia Object Servers, which are responsible for the handling of continuous media data at a single site. Cross-site streams can be realized transparently for the client, automatically establishing real-time connections between sites via a multimedia transport system [3, 8].

This paper describes the principle methods of media scaling on end-systems and its integration into the Distributed Multimedia Object Services.

2 Scaling Mechanisms

Scaling, the ability to adapt to situations in which resource demand is larger than resource availability, requires mechanisms which observe resource usage and detect resource shortages as well as later 'recovered' resource availability – the resources have to be monitored.

2.1 Resource Monitoring

A resource monitor (RM) observes resource usage and resource load. If the RM detects that the resource load state (RLS) changes in such a way that a reaction is necessary, it provides an indication about the RLS and how critical the current situation is. We distinguish the following types of RLS:

1. a resource is in a state of stable and acceptable load,
2. the load of a resource is increasing and reaches a critical state,
3. a resource is overloaded, and
4. the load of a resource is decreasing and left an overloaded state.

This classification not only leads to indications of 'stable' states, but also allows, by the dynamicity of state 2, for a proactive generation of indications which yields earlier and faster reaction to resource shortages and prevents the system from reaching state 3. Similarly, indications for recovered resource availability should not be generated too early to inhibit permanent oscillation between different RLS.

Detecting the dynamicity of states 2 and 4 can be approximated by using a finer granularity for load regions. In addition to the definitely overloaded state 3, several load regions of increasingly high resource load are distinguished by dividing the base load states 1 and 3 into smaller regions $S_0 \ldots S_n$, S_0 being the stable and S_n the overloaded state and state transitions reflecting the dynamicity (see Figure 1). The indications generated by the RMs inform the resource users to reduce the load faster (and stronger) as the load region enters higher load regions.

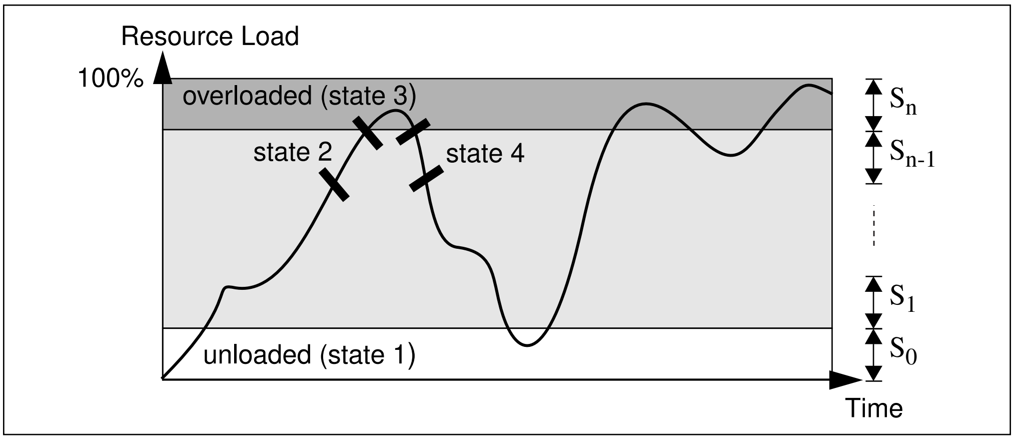

Fig. 1. Resource Load and Resource Load States

In principle resource monitoring is necessary for all resources that are concerned with the processing and transmission of multimedia data. These include resources like CPU, buffer space, and network, but also disks and the system bus. This paper concentrates on the former because they are important for all types of multimedia applications. Other resources, not being considered in this work are I/O related resources, especially file system resources, i.e., disks and their controllers, and the system bus.

2.2 Load Dimensions

In general, we distinguish two dimensions for resource usage:

- throughput and
- processing time requirements.

While they are not completely orthogonal (the time needed to copy data of some amount from one buffer to another depends on the size) they are also not completely correlated (if more time is spent for compression, the resulting space requirements may become smaller). Treating these two dimensions independently produces faster and more exact decisions than using only one measure reflecting the total acceptable load. Additionally, data stream properties can be adapted to available resources along these two dimensions. Two-dimensional scaling matches quality of streams and current load more accurately, yielding higher utilization and better overall quality of streams. Overload with regard to network or buffer resources is resolved by reducing throughput of streams, while CPU problems are tackled by decreasing processing time requirements.

As shown in Figure 2, the resource monitors for CPU, network, and buffer space provide the mechanisms for resource overload detection (in general, for resource state indication) and deliver the respective indications. Based on these indications a policy agent decides which stream, or set of streams, is affected by the resource overload. The policy agent informs the stream about the necessity to adapt to the changed resource availability.

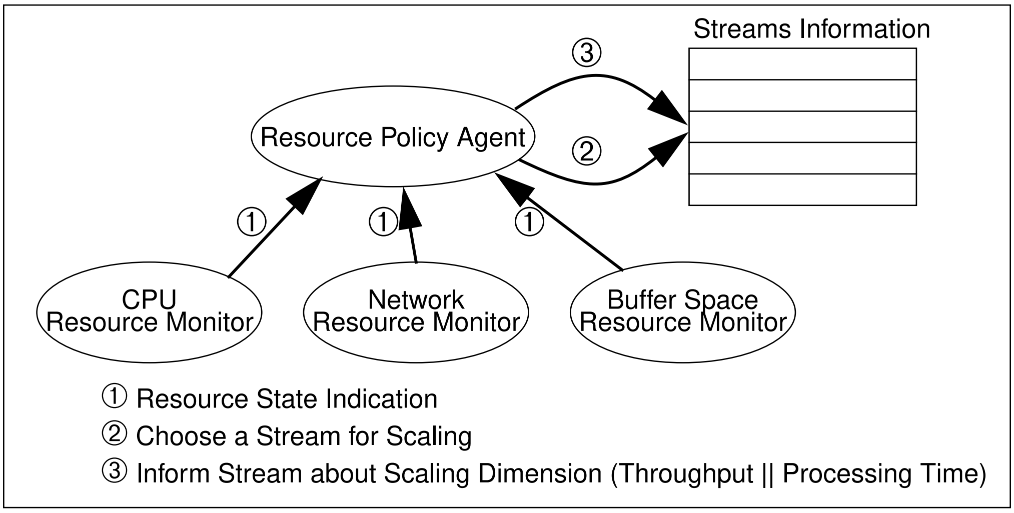

Fig. 2. Resource Monitors and Resource Policy Agents

The information provided by the policy agent indicates whether throughput or processing time usage must be adapted. The reaction of the stream depends on the particular stream handling thread and is determined by its capabilities as well as the data types. For instance, for a motion JPEG stream subsampling and quantization table parameters can be varied in order to adapt the load to either or both dimensions.

This is illustrated in Figure 3 where time and space requirements for the compression of a single JPEG picture are depicted over a range of quantization tables and 4 different settings of subsampling. If the throughput requirements of a stream have to be reduced, the quantization may be changed to a coarser degree or subsampling may be switched to a larger level. As can be seen from the measurements, if a stream operates at high quality with respect to quantization, reducing the quantization decreases frame size considerably. In lower quantization areas (below $\approx$ 80), switching to a different subsampling level yields better buffer space requirement reductions.

2.3 QoS Class for Scaling

The resource policy agent needs the information which streams to take into account when making scaling decisions. We have introduced the service class ‘scalable QoS’ as an extension to our resource management system [7] that already offered guaranteed and statistical QoS.

All streams belonging to this class may be affected by the decisions of the policy agent to ensure fairness to streams and properly balancing of resources. Membership to this class is voluntary, the stream creator decides to request the QoS class ‘scalable’ for a stream, mostly because the charged costs are lower than the costs for using a guaranteed service.

For streams in the ‘scalable’ QoS class, the policy agent and the stream agree on a certain behavior. The resource policy agent is responsible to deliver scaling indications to the streams and a stream has to adapt its load accordingly. Streams that are not members of the QoS class ‘scalable’ do not participate in the adaptation process.

Due to its knowledge about membership to this class, a policy agent can decide which streams should adapt their resource usage and how much. If an agent detects a local problem, e.g., CPU overload, all streams belonging to the scalable QoS class will be informed to reduce their processing time usage. This way, each single stream has to reduce its usage only a bit, to reach the overall savings. If a shortage in network throughput is detected, only the stream for which the problem arises will be asked to reduce its load, since it is likely that other streams are not affected. For instance, if the problem occurs at an overloaded router, other streams that are local or use a different path do not experience any quality degradation.

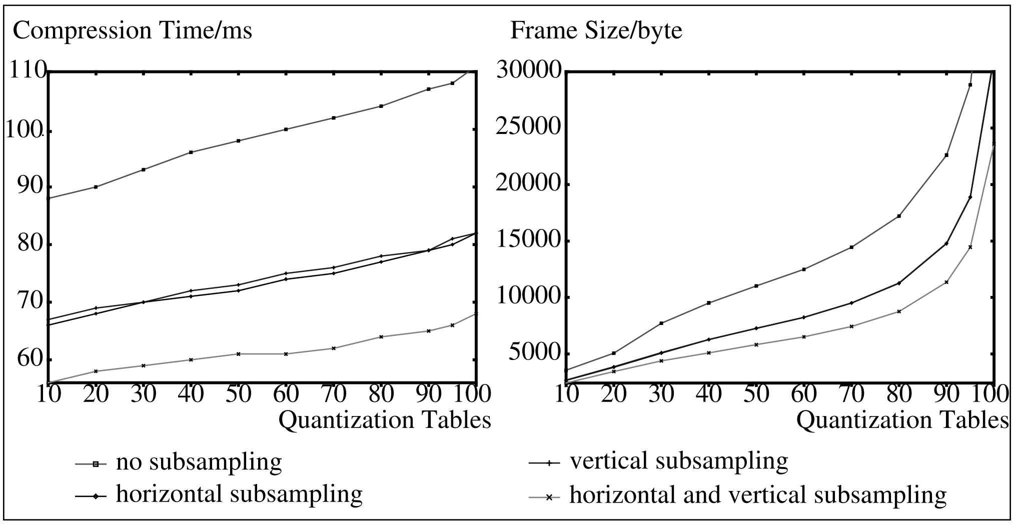

Fig. 3. JPEG Compression Time and Frame Size for different Quantization Tables

3 Scaling in Distributed Multimedia Object Services

Multimedia applications handle several kinds of media including continuous media such as audio and video. Due to its special nature the processing of these data demands real-time support from the underlying computing platform in order to continuously transmit, synchronize, and present audio and video data streams within a distributed system.

Multimedia streams are flowing through the system based on software and special hardware. Offering access to proper abstractions for hardware and data can make exclusive resources sharable between applications and allows easy application development and portability across supporting platforms.

The Distributed Multimedia Object Services (DMOS) support multimedia applications effectively during development and run-time by offering an abstract interface to real-time supported multimedia objects. Applications, as clients of DMOS, can create and combine multimedia objects such as audio and video streams to acyclic graphs potentially spanning several machines. Realization of multimedia objects happens in Distributed Multimedia Object Servers, which are responsible for the handling of continuous-media data at a single site. Cross-site streams can be realized transparently for the client, automatically establishing real-time connections between sites with the support of a multimedia transport system [3, 8].

3.1 Important Classes of Distributed Multimedia Object Services

The *LogicalDevice* class abstracts from hardware and software realizing a specific functionality of input, output, or filtering of multimedia data. Subclasses include Speaker, Camera, and Microphone each of which encapsulates all details of the underlying implementation.

The *Stream* class allows to combine a set of LogicalDevice objects in order to control the flow of data through the devices via a single interface. Control operations include start and stop operations and the acquisition and release of all resources necessary to process the data.

The *QualityOfService* class represents the kind of service the client is requesting. An instance of this class can be associated with a Stream in order to express the service class as guaranteed, statistical, or scalable, and the flow specification in terms of delay, throughput, and loss. The service class scalable is chosen when the type of application allows a temporary service degradation. However, the application itself is not involved in the process of scaling at all but relies on DMO servers. See Figure 4 for the client view of multimedia objects in a remote surveillance application.

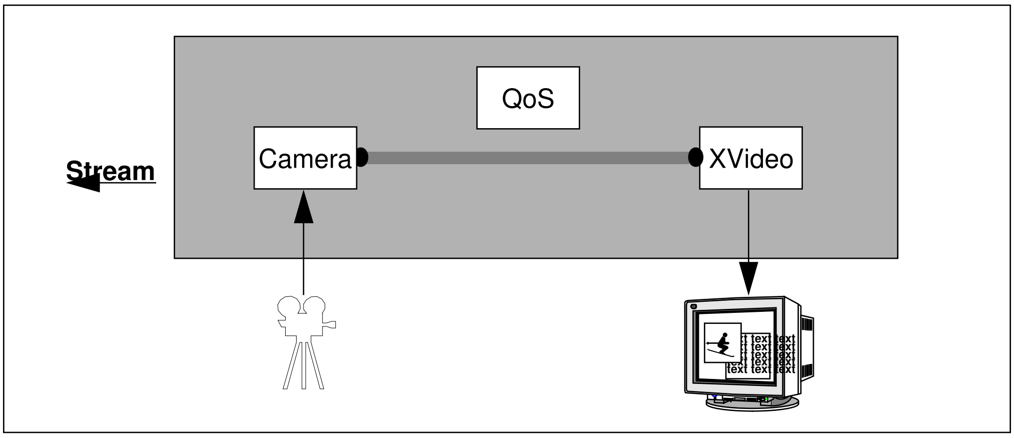

Fig. 4. View of Multimedia Objects for a Remote Surveillance Application

3.2 Scaling in Distributed Multimedia Object Servers

Every object that is created by a multimedia application is realized in a DMO server. However, other objects exist within a server that can not be accessed by clients. In order to provide real-time services the client-visible objects utilize a layer of stream handlers and management threads (see also Figure 5).

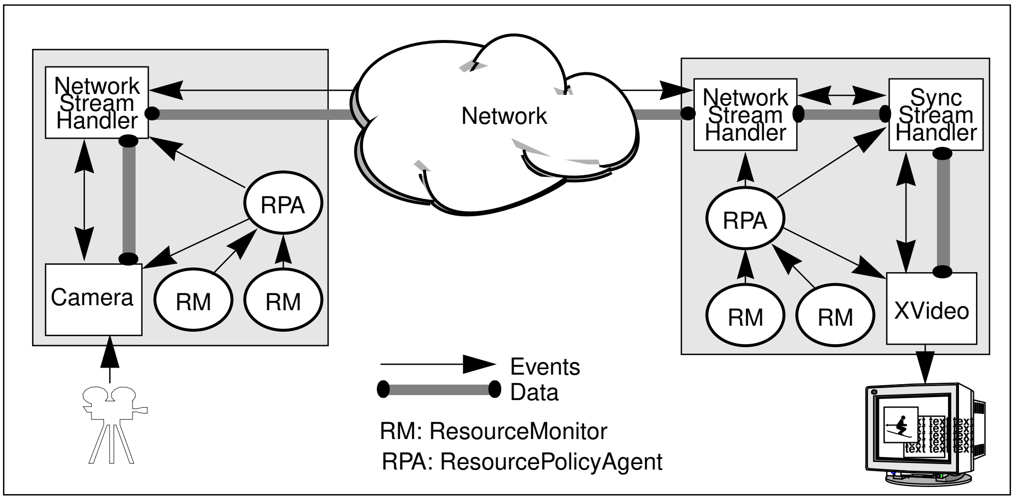

Fig. 5. Implementing Scaling in DMO Servers

Logical devices are mapped to stream handlers, i.e., a stream handler is a real-time thread that performs a specific task with regard to the processing of multimedia data. Stream handlers pass data packets along the stream and can exchange events or pass them to client-visible objects.

A *ResourceMonitor* is responsible for the status of a single resource. If the load reaches a defined critical value the ResourceMonitor notifies a *ResourcePolicyAgent*.

The ResourcePolicyAgent determines which stream handlers affect the resource and which of them belong to the service class ‘scalable’. Scaling of streams is organized along the two dimensions throughput *T* and CPU requirements *C*, i.e., the scaling status of a stream *S* can be seen as a point in the coordinate system of these dimensions that have scales from 0 to 100 percent of the stream’s respective requirements:

$$S = (T, C), \{0 \leq T,C \leq 100\}$$

Depending on the resource that is overloaded the ResourcePolicyAgent may decide to reduce the load with regard to either or both of these dimensions which adds significant flexibility to adapting the quality of streams to available resources. Since scaling states of all streams are known to the ResourcePolicyAgent it can equally balance quality degradation for all streams. Taking current scaling state into account a new scaling status is derived and passed as a message to each affected stream handler.

If its scaling status is changed by the scaling message a stream handler receiving such a message sends an event to the next stream handler up-stream in order to propagate the scaling status to the source of the stream. Note that there are two principle sources for the change of a scaling status: An event received from a stream handler residing down-stream containing the new sub status $S_s = (T_s, C_s)$ or a message from the ResourcePolicyAgent with $S_p = (T_p, C_p)$. A minimum status has to be derived in both cases from the most recently updated sub states as

$$S = (\min(T_s, T_p), \min(C_s, C_p)).$$

This filtering of scaling messages ensures that different perceptions of resource load for stream handlers, potentially being on different machines are properly resolved on their propagation path to the source of the stream.

When the source of a stream changes its scaling status the quality of the stream is adapted according to the new state S such that the stream's new specifications T and C with regard to throughput and CPU are met. Every stream handler that can serve as a source within a scalable stream has a built-in strategy to adapt the stream it generates. For instance the stream handler generating JPEG images from a camera has a considerable flexibility using a combination of subsampling and quantization table parameters (see Figure 3). This gracefully degraded quality is much more acceptable than a sudden drop of the frame rate or even break of the stream in case of packet losses.

4 Conclusion

Media scaling is a technique to adapt the amount of audio and video flowing through a system to available resources. Its usage has been reported to avoid congestion in networks that can not properly be supported by resource management. We have enhanced the meaning of media scaling from the network to the end-system level and have shown its value for the user. Media scaling not only has the potential to increase the number of streams a system can support simultaneously but it also keeps media streams meaningful to the user that would break during overload situations. Scaling of media streams in dependence of actual resource utilization can keep the system in a range of optimal load. The distinction of different load dimensions caused by a stream helps to adapt to available resource capacity more precisely.

Whereas end-system-transparent scaling within transport systems is constrained to scaling on a per packet basis, scaling that involves changing the encoding of transmitted media at the sender can be based on a much finer granularity.

If scaling is not supported by a system layer, applications have to duplicate the effort of implementing scaling mechanisms. Additionally, due to the lack of coordination, scaling mechanisms working concurrently in different applications would rather compete than cooperate. By coordinating down- and upscale operations of all streams on a given system, the balanced sharing of resources can be guaranteed.

We have shown how to integrate scaling in a layer that provides system support for multimedia applications during development and run-time. The Distributed Multimedia Object Services offer an abstract interface to real-time multimedia objects. Scalability of streams can be set by multimedia applications using a QualityOfService object that is associated with a stream. Realizations of streams and actual scaling operations across machines are handled by DMO servers.

References

[1] S.T.-C. Chou, H. Tokuda: *System Support for Dynamic QOS Control of Continuous Media Communication.* Third International Workshop on Network and Operating System Support for Digital Audio and Video, San Diego, 1992.

[2] L. Delgrossi, C. Halstrick, D. Hehmann, R.G. Herrtwich, O. Krone, J. Sandvoss, C. Vogt: *Media Scaling in a Multimedia Communication System*. First ACM Multimedia Conference, Anaheim, 1993.

[3] L. Delgrossi, R.G. Herrtwich, F.O. Hoffmann: *An Implementation of ST-II for the Heidelberg Transport System.* Internetworking – Research and Experience, Vol. 5, Wiley, 1994

[4] D. Hoffman, M. Speer, G. Fernando: *Network Support for Dynamically Scaled Multimedia Data Streams*. Fourth International Workshop on Network and Operating System Support for Digital Audio and Video, Lancaster, 1993.

[4] K. Jeffay, D.L. Stone, T. Talley, F.D. Smith: *Adaptive Best-Effort Delivery of Digital Audio and Video Across Packet-Switched Networks*. Third International Workshop on Network and Operating System Support for Digital Audio and Video, San Diego, 1992.

[5] H. Tokuda, Y. Tobe, S.T.-C. Chou, J.M.F. Moura: *Continuous Media Communication with Dynamic QOS Control Using ARTS with an FDDI Network.* ACM SIGCOMM 92, Baltimore, 1992.

[6]) P. Uppaluru: *Networking Digital Video.* 37th IEEE COMPCON, 1992.

[7] C. Vogt, R.G. Herrtwich, R. Nagarajan: *HeiRAT: The Heidelberg Resource Administration Technique, Design Philosophy and Goals*. Kommunikation in verteilten Systemen, Munich, 1993.

[8] L.C. Wolf, R.G. Herrtwich: *The System Architecture of the Heidelberg Transport System.* ACM Operating Systems Review, Vol. 28, No. 2, April 1994.