System-Level Resource Management for Network-Based Multimedia Applications*

Louis C. Schreier and Michael B. Davis
SRI International, 333 Ravenswood Avenue, Menlo Park, California 94025

Abstract

SRI International (SRI) has developed a model for system-level resource management in distributed systems, and applied this model in the context of a multimedia conferencing application. The model considers user objectives, resource constraints, and adaptable execution techniques. User objectives are specified by means of benefit functions. We have implemented a prototype of a distributed multimedia display application that demonstrates key aspects of the model, including adaptation to a changing execution environment.

1. Introduction

Multimedia applications, especially distributed ones, require large amounts of processing, communication, and storage resources. Current operating system and resource management technologies attempt to provide the maximum set of system resources requested by each application, whether or not the application actually benefits from the full use of those resources.

SRI International (SRI) has developed a system resource management (SRM) model that allows users to express their preferences for media quality by associating *benefit functions* with performance attributes. In our model, the underlying execution system adjusts the amount of processing, communication, and storage resources provided to an application so as to maximize the set of benefit functions that characterize the application. We have developed a demonstration prototype that uses commercial technologies and implements a meta-level execution system on top of operating-system and communication services. In our prototype, we parameterized the benefits a user derives from different frame rates, display window sizes, and compression quality factors. As the underlying resources become more or less available, or as the user changes the benefit functions, the underlying execution system integrates the benefit functions and adjusts the resource usage pattern to maximize the user's benefit.

2. Benefit Model

The users of a distributed system have requirements and preferences regarding the resources that should be made available for accomplishing various tasks. For example, in a multimedia conferencing application, the recipient of an audio and video stream has preferences regarding how the information will be communicated and presented (e.g., the audio quality, the frame rate, the image quality, and the degree of audio/video synchronization). Each quality-of-service preference, or *objective*, can be quantified

*This research was supported by Rome Laboratory under Contract F30602-91-C-0099.

by being expressed as a *benefit function* that relates the benefit accrued to the level of service obtained. This abstraction is similar to the time-value function used in the Alpha operating system [1], in which the value of a computation is related to the time at which it is completed. We extend and generalize the Alpha abstraction to allow the specification of arbitrary objectives, not just timelines.

Figure 1 shows two sample benefit functions for multimedia streams. In Figure 1(a), the benefit increases as the frame rate increases, up to a plateau after which the user perceives no improvement. In Figure 1(b), the benefit decreases as the time difference between the corresponding audio and video streams increases.

By quantifying the level of service for a given objective and assigning relative benefits to various levels, we can construct a benefit function for any objective we wish to define. A resource manager can use benefit functions to compare the relative benefits of different objectives, or of different levels of a single objective, without needing to understand the semantics of particular objectives.

3. Execution Model

Based on the objectives expressed by users, and on the availability of system resources, tradeoffs must be made and appropriate execution techniques must be selected. For example, using lower frame rates and lower video resolutions may free up CPU and communication resources to provide higher-quality audio. Using additional memory and disk buffers may improve audio and video synchronization, while increasing delays.

Resource management decisions are made at two levels, as shown conceptually in Figure 2. Decisions involving medium- to long-range tradeoffs among activities and their objectives are made by a high-level decision maker. The high-level decision maker considers the user objective functions, resource constraints, characteristics of candidate techniques, and system status information to choose (1) the activities (e.g., multimedia conferences or periodic tasks) to execute; (2) the nominal amount of resources to devote to each activity; and (3) the techniques and parameters to use when executing these activities, such that the objectives can be met to the optimal degree consistent with satisfying the resource constraints.

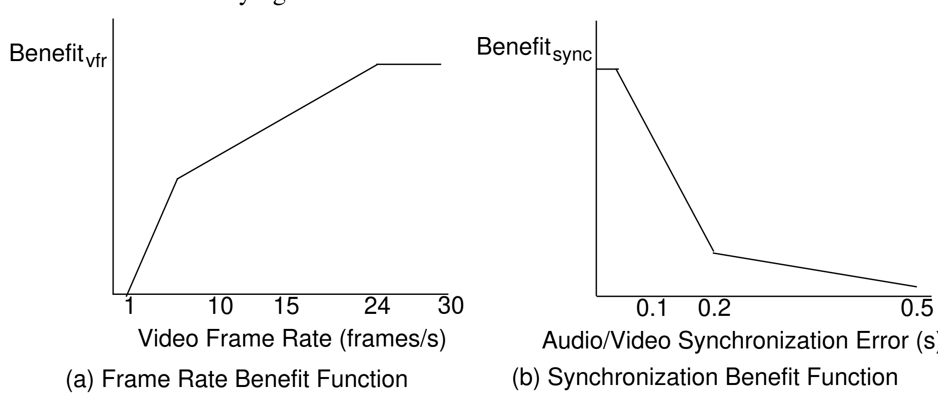

Figure 1. Sample Benefit Functions for Multimedia

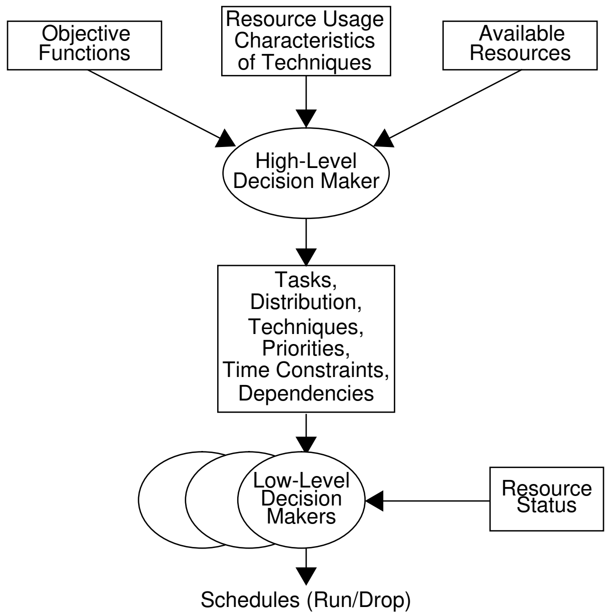

Figure 2. Decision-Making Concept

Real-time scheduling decisions among competing tasks are made by low-level schedulers for each resource. Each low-level scheduler uses status information about its resource, as well as process control abstractions passed down from the high-level decision makers, to make local decisions. While low-level resource scheduling is decentralized, integrated control (through shared resource status information and the consistent interpretation of process control abstractions) is used to ensure that the schedulers complement each other's decisions.

4. Proof-of-Concept Implementation

We implemented a simple remote video presentation application in which video from a video camera or video cassette recorder (VCR) is captured and digitized at a computer workstation, transferred across a local area network to a destination workstation, and displayed on the monitor of the destination workstation [2,3]. The key concept demonstrated is the adaptation of the distributed application to limited communication resources, according to preferences specified by the user.

The logical architecture of the demonstration implementation is shown in Figure 3. The implementation consists logically of three parts: a sending process (executing on the source workstation); a receiving process (executing on the destination workstation); and a controlling process (executing on any workstation in the network, but typically on the destination workstation). The controlling process acts as the high-level decision maker.

A computer workstation with appropriate video hardware captures full-motion video from a video camera or VCR, digitizes the video frames, and compresses them,

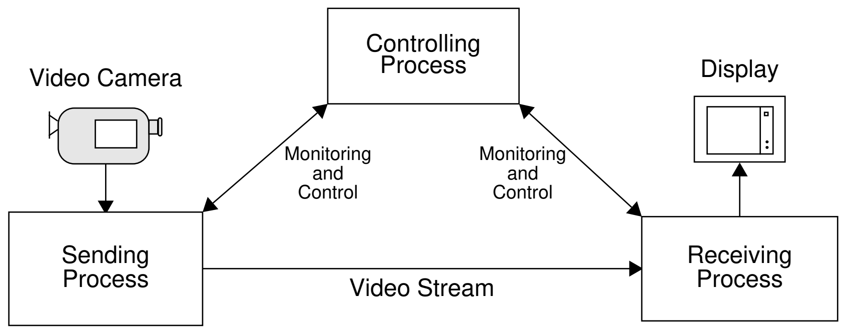

Figure 3. Logical Architecture of SRM Multimedia Demonstration

using the Joint Photographic Experts Group (JPEG) image compression algorithm. The stream of digitized video frames is sent via a local-area network to a destination workstation, using standard communication protocols. The destination workstation reads the digital video stream from the local-area network, uncompresses the frames, and displays them in full motion on its color monitor. By means of a graphical user interface on the destination workstation, the user can connect to the source workstation, start and stop the video playback process, and exit the program. The user can also specify benefit functions that express his or her preferences regarding frame rate, frame size, and Q factor (the latter controls the lossiness of the compression and therefore the quality of the images). The controlling process monitors the amount of video data transferred, and uses a simple heuristic technique to determine the combination of frame rate, frame size, and Q factor that produces the highest total benefit under the current resource constraints. The controlling process controls the video capture, compression, and transfer process accordingly.

The implementation is currently in the prototype stage, but preliminary results show that the system is able to adapt to available resources, according to the preferences expressed by the user. If additional funding becomes available, we plan to further develop and formalize the benefit and execution models, develop distributed algorithms for making distributed control decisions, and extend the demonstration application to support multiple sending and receiving processes.

References

1. J.D. Northcutt and R.K. Clark: *The Alpha Operating System: Programming Model*, Archons Project Technical Report 88021, Department of Computer Science, Carnegie Mellon University, February 1988.
2. M.B. Davis, A. Downing, and T. Lawrence: Adaptable System Resource Management for Soft Real-time Systems, *Symposium on Command and Control Research and Decision Aids*, Monterey, California, June 1994.
3. M.B. Davis: *System Resource Management for Distributed Real-Time Systems*, Final Technical Report, ITAD-2655-FR-95-060, SRI International, Menlo Park, California, 1995.