INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Two-level Resource Management for QOS Support in Multimedia Workstations

by

Govindachari Raghunath, B.Sc., M.C.A

A Thesis Submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

Ottawa-Carleton Institute for Computer Science

School of Computer Science

Carleton University

Ottawa, Ontario

January 1998

© Copyright

1998, Govindachari Raghunath

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-26885-3

Canada

The undersigned hereby recommend to
The Faculty of Graduate Studies and Research
acceptance of the thesis,
**Two-level Resource Management for QOS Support in Multimedia
Workstations**

submitted by
Govindachari Raghunath, B.Sc., M.C.A
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

External Examiner

Director, School of Computer Science

Thesis Co-supervisor

Thesis Co-supervisor

Carleton University

16 December 1997

Abstract

Workstation-based multimedia applications are becoming increasingly common with the continuing advances in workstation hardware technology. In these applications, continuous media types such as audio and video have timeliness properties. The performance requirements of both traditional and multimedia applications can be expressed as *Quality of Service* (QOS). Allocation of workstation CPU to meet the QOS requirements of applications is the principal subject of this dissertation. A novel *Application and User-oriented Resource allocation Approach* (AURA) is proposed and investigated in this research. The salient aspects of this approach are: identification of multimedia application behaviors that allow efficient resource allocation (*demand synergy*), a two-level architecture that distinguishes between *competing* and *cooperating* workloads, group scheduling for isolating competing workloads and collaborative mechanism for sharing within a group.

Three group scheduling algorithms, GRP-RR, GRP-EDF and GRP-RC, were implemented and studied for their effectiveness in achieving isolation and meeting the execution rate requirements as well as deadline requirements. Although all the three schedulers achieved these objectives under specific conditions, GRP-EDF scheduler was found to be the most effective under a wide range of conditions. The GRP-RC scheduler had the next best performance but had much lower overheads than the GRP-EDF. A reactive QOS control strategy based on focus state of applications for sharing capacity within a group, when used in combination with group reservation was found to be effective in meeting the QOS requirements of the groups.

Acknowledgements

First of all I would like to thank my supervisors, John Neilson and Murray Woodside, for their guidance. Both of them showed unlimited patience with my floundering and went out of their way to ensure that I would complete this research. Prof. Woodside also provided me generous financial support.

I am grateful to the Canadian Commonwealth Scholarships and Fellowships committee and the current administrators, ICCS for their financial support through out my doctoral program. Thanks are also due to my employers, Bharat Electronics, for granting me special leave for my doctoral studies.

I have had the good fortune of having two mentors and friends in Prof. Anil Maheshwari and Prof. John Oommen at the School of Computer Science. They provided me constant encouragement and guidance throughout my stay at Carleton.

My host and friend Murali Thiagaraja treated me as a member of his family during my stay at his home. I wish to thank him and his family for this.

School of Computer science at Carleton is a unique place in terms of its friendly and cooperative environment, with able support from technical and office staff. It is also the place where I got the opportunity to make many friends from several cultures. I wish to thank them all for this.

My mother and sisters deserve special mention, for nurturing in me this dream and providing all the support and love. I am also indebted to my uncles who have been the sources of inspiration to me. I would also like to thank my cousins who showed me how much fun and multidimensional life can be.

Contents

Abstract	iii
Acknowledgements	iv
List Of Tables	viii
List Of Figures	ix
1 Introduction	1
1.1 Workstation-based Multimedia Applications	1
1.2 Multimedia and Quality of Service	4
1.3 Research Goals, Approach and Scope	6
1.3.1 Research Goals	7
1.3.2 Research Approach	9
1.3.3 Scope of Research	9
1.3.4 Contributions	11
2 Background	13
2.1 Quality of Service Specification and Guarantees	14
2.2 QOS Support in Integrated Services Networks	19
2.3 QOS Support in Workstations	20
2.3.1 Real-time Scheduling for QOS Support	21

2.3.2	Overload Handling and QOS Support	24
2.3.3	Processor Capacity Reservation	25
2.3.4	Best Effort QOS Provisioning	26
2.3.5	Summary	27
2.4	Dynamic Quality of Service Management	28
2.5	Summary	29
3	Application and User-Oriented Resource Allocation	31
3.1	Application and User-oriented Approach	32
3.1.1	Motivations for a New Approach	32
3.2	Application Structure and Behavior	38
3.2.1	Definitions	38
3.2.2	Application Structures	41
3.2.3	Synergistic Behavior of Applications and Sessions	47
3.2.4	Benefits of Demand Synergy	51
3.3	AURA Approach to Resource Allocation	57
3.3.1	QOS Model	59
3.3.2	Discussions	62
3.3.3	Related Work	65
3.4	Summary	67
4	AURA: Architecture and Algorithms	68
4.1	Overview of Resource Allocation	68
4.2	AURA Architecture	71
4.3	Group Reservation for Isolation and Sharing	74
4.3.1	Model of Computation for Scheduling	76
4.4	Group Scheduling Algorithms	78
4.4.1	GRP-RR Scheduler	80
4.4.2	GRP-EDF Scheduler	86

4.4.3	GRP-RC	91
4.5	Collaborative Sharing Within Group	92
4.5.1	Focus-based QOS Control	94
4.6	Summary	100
5	Implementation and Evaluation	102
5.1	Implementation of the Algorithms	102
5.1.1	PARASOL Environment	103
5.1.2	Simulation Model Implementation in PARASOL	104
5.1.3	Implementation Experience	106
5.2	Evaluation of AURA Resource Allocation	107
5.2.1	Performance Objectives and Metrics	108
5.2.2	Overview of the Experiments	112
5.3	Results and Discussions	119
5.3.1	Rate Experiments	121
5.3.2	Delay Experiments	123
5.3.3	Mixed Load Experiments	129
5.3.4	Summary of Group Scheduler Performance	132
5.4	Collaborative Sharing Experiments	133
5.4.1	Workload and QOS Model	134
5.4.2	Observations and Conclusions	137
6	Conclusions	146
6.1	Contributions and Conclusions	146
6.2	Future Directions	149
	Bibliography	151
	References	151

List of Tables

2.1	Categories of QOS parameters	14
2.2	Summary of scheduler support for QOS in workstations	27
3.1	Random and state-dependent demand variations	46
5.1	Summary of experiments	120
5.2	Rate experiment with greedy load with symmetric reserves	121
5.3	Rate experiment with greedy load with asymmetric reserves	122
5.4	Rate experiment with periodic Load with symmetric reserves	123
5.5	Rate experiment with periodic load with asymmetric reserves	123
5.6	Mean rate achieved by group 1 in <i>Delay-Period-Greed</i> experiments. .	124
5.7	Preemption counts in the delay-period-greed experiment	129
5.8	Deadlines missed by multimedia workload under greedy competition .	130
5.9	Mean rate achieved by multimedia workload under greedy competition	131
5.10	Deadlines missed by multimedia group in isolation	131
5.11	Deadlines Missed by multimedia group under bursty competition . . .	133

List of Figures

2.1	Examples of Benefit Functions	15
3.1	Structure of a multimedia application	40
3.2	Dynamic structure of a multimedia session	41
3.3	Temporal hierarchy and resource management	43
3.4	News presentation application	48
3.5	Medical tele-conferencing session	50
3.6	A Simple session with synergy	52
3.7	A session with no Synergy	52
3.8	AURA approach to resource allocation	58
3.9	Mapping of QOS parameter to utility value	61
3.10	Feasible, acceptable and infeasible QOS vectors	61
4.1	AURA two-level resource allocation architecture	72
4.2	AURA internal structure	73
4.3	Virtual processor share in group reservations	75
4.4	Generic model of group scheduling	79
4.5	Pseudo-code for the GRP-RR scheduler	82
4.6	GRP-EDF Scheduling States	89
4.7	Pseudo-code for the GRP-EDF algorithm	90
4.8	Pseudo-code for the GRP-RC algorithm	93
4.9	Reactive QOS control algorithm	95

4.10	Selection policy for upgrading within reserved capacity	96
4.11	Upgrading QOS beyond QOS guaranteed by reserved share	97
4.12	Downgrading policy.	98
4.13	QOS Space for focus-based reactive QOS control	99
5.1	Parasol model of Simulation	105
5.2	Fraction of deadlines missed by Group 1 activities with Group 1 reserved share of 10%.	125
5.3	Fraction of deadlines missed by Group 1 activities with Group 1 reserved share of 20%.	125
5.4	Fraction of deadlines missed by Group 1 activities with Group 1 reserved share of 30%.	126
5.5	Fraction of deadlines missed by Group 1 activities when scheduler resolution was 100ms.	127
5.6	Fraction of deadlines missed by Group 1 activities when scheduler resolution was 300ms.	128
5.7	Fraction of deadlines missed by Group 1 activities when scheduler resolution was 460ms.	128
5.8	Background load intensity, and foreground and background Utilizations in reservation-less collaboration.	140
5.9	Frame rates achieved in reservation-less collaboration	141
5.10	Background load intensity, and foreground and background utilizations in independent reservation collaboration.	143
5.11	Frame rates achieved in independent reservation collaboration	144
5.12	Background load intensity and foreground and background utilizations in Group reservation collaboration.	144
5.13	Frame rates achieved in group reservation collaboration	145

Chapter 1

Introduction

This dissertation is concerned with management of resources, in particular the CPU, in multimedia capable workstations. This chapter will motivate the need for supporting multimedia applications in a workstation environment. It will introduce the intrinsic characteristics of multimedia applications and their formalization into the notion of Quality of Service (QOS). Then, it will proceed to outline the approach taken in this research, establish the goals within the context of this approach and state the scope of the research described in this dissertation.

1.1 Workstation-based Multimedia Applications

Recent advances in communication and computer technology makes possible many interesting and innovative applications. Among them multimedia applications are gaining importance due to the synergistic benefits of integrating multiple media types into a single vehicle of communication. Such an integration opens up new vistas for expression and dissemination of information, suggests new paradigms for cooperative work and communication, and ushers in new possibilities such as telemedicine and virtual reality [1, 18].

The fundamental factor that contributes to these advances is the continuing improvement in the cost/performance of workstation hardware. Besides this, three other factors can be identified. First of all, government initiatives towards national and global information infrastructures have provided a major impetus to stimulating active research, academic and commercial. This was evidenced by a surge in installation of wide-area fiber-optic connectivity, and commissioning of metropolitan area network. Following this, many strategic alliances between telecommunication service providers and local cable service distributors are taking place. This will bring to many homes and businesses the necessary global connectivity. Several information and entertainment content providers have also begun to offer the so-called multimedia infotainment programs. This segment is primarily aimed at non-computer-based home applications such as Video on Demand, using set-top boxes.

The second factor is the growth of the Internet. The explosion in the Internet connectivity can be seen from the increasing number of local "Internet Service Providers". Although the Internet as a technology has existed for almost a couple of decades, two recent developments have catalyzed its potential as a vehicle for global information, and added much value to the Internet-based information access in general and computer-based multimedia applications in particular. The first is the World Wide Web (WWW), an Internet-based application which provides convenient access to information in several formats and using several protocols along with search and indexing capabilities. The second is the availability of MBONE, the multi-cast extensions to Internet protocol [14].

The World Wide Web has made possible many interesting ways of organizing information including text, audio, images, animations and video. The web documents are organized as hypertext-based collection of text, graphics, audio and video. With the full induction of hypermedia language standards such as HYTIME and MHEG, it is expected that the WWW will fully support multimedia information storage, retrieval and presentation. With the availability of multi-cast routing software and MBONE applications such as *nevot*, *nv*, *vat*, *sd* etc., desktop video conferencing and computer supported cooperative work applications have also been developed

and increasingly used. For example, the IETF meetings are audio-cast using the Internet [5].

A third technical development contributing to the growth of workstation-based multimedia applications is advances in video compression technology. Even a short 1 minute clip of video needs a large amount of disk storage and buffer space, as much as 1200MB for a 640x480 image size with 24-bit plane depth and at a rate of 30 frames-per-second, and hence compression is essential. Several video coding standards have been proposed and are continued to be improved. The initial adaptation of JPEG coding scheme as a standard for static photographic images, the development of standards such as H.261 for ISDN communication and the currently popular MPEG-II standard for motion pictures are some evidences for this trend. Based on the experience gained with these, more sophisticated compression techniques such as wavelet-based coding and layered coding schemes are being evaluated and seem to hold promise of further improvement.

Although hardware codecs are expected to ease the computational requirements for the compression and decompression algorithms, software codecs will remain the main component of these systems. Integration of multiple media types (other than traditional data and graphics) into first-class entities in computers provides immense potential for direct manipulation of audio and video. Automatic analysis of multimedia news content and video indexing by content are some of the emerging applications that are handled by computers. Once processed these media objects are stored and retrieved similar to traditional data objects common in workstation environments. Given these increasing use of desktop workstations for multimedia applications, it is necessary that the workstation resources are managed in a manner suitable for these different classes of applications. Although the workstation CPU speed is steadily increasing, new innovative applications requiring more computational power are constantly appearing. Thus the multimedia capable workstations of today and the near future will be operating in a window of scarcity for several years to come [24]. Therefore, mechanisms and policies for efficient management of workstation resources are crucial for the large scale successful realization of futuristic

applications.

1.2 Multimedia and Quality of Service

Multimedia applications include among the media types, apart from data and graphics, audio and video as well. The integration of audio and video into applications along with text and images places heavy demand on the workstation CPU. Audio and video are two of the important media types that carry with them an inherent notion of timeliness and are therefore called *continuous media* [2]. Therefore, in principle, continuous media types are viewed as streams of information bits flowing as a function of time. In practice, they are considered as flow of discrete units of data such as frames of audio or video. The timeliness requirement necessitates that the maximum delay in processing each data unit be bounded. In contrast to traditional interactive applications where the requirement is to control the average or percentile delay, in the case of continuous media types, the time bounds are applicable for a large percentage of data units if not for every single unit. Multimedia applications are also distinct from hard real-time systems such as process control applications. In hard real-time systems, the correctness of the application is directly determined by whether or not the timeliness requirement is met. Worse still, in mission critical systems, the missing of deadlines could be catastrophic. On the other hand, the timeliness of continuous media is less severe in that, it primarily determines the usefulness of processing a data after a certain time. Thus, missing deadlines is not fatal but instead, reduces the value to the user of the information processed. Thus, there is some inherent flexibility in the timeliness requirements of multimedia applications which suggest resource allocation techniques more flexible than those used in hard real time systems.

Audio, while not data intensive, is very sensitive to delays and variations in delays. Video is highly bandwidth intensive. For example, uncompressed video at a rate of 30 frames per second, at a resolution of 640 x 480 pixels and a 24-bit plane will require a bandwidth of more than 200 Mbps. With compression this rate is

reduced by as much as 1:30 but still, the required bandwidth can be high. Video compression standards are still evolving, making it less common to have hardware-based solutions. Even if hardware-based video compression becomes common, there is a necessity for software handling of video data for such tasks as indexing and searching, content analysis. These tasks are application specific and hence need to be realized in software. If such large volumes of data are to be handled in a timely manner, the workstation hardware and software should be optimized for efficient movement and processing of large volumes of data. The amount of data that can be buffered in the workstation memory will however be limited. Buffering is used customarily to reduce delays in accessing data from secondary storage or across the network from a server or remote host. Therefore, resource allocation policies and mechanisms should attempt to reduce the amount of buffering.

In conversational applications, audio and video streams will have to be tightly synchronized. In multimedia presentation type of applications, the presentation of various objects of different streams will have to be synchronized with respect to a presentation time-line. Such synchronization are presently considered part of the application and are built into the application when the multimedia document is being authored. General techniques for specification of timing relationships as well as for deriving transmission and playout schedules also exist. However, it is usually assumed that the resources are available whenever an object is to be transmitted or played out. A notable exception is the Limited a priori (LAP) scheduler [20] which deals explicitly with a reservation-less network by modeling the delay and constructing a quasi-dynamic transmission schedule. The effect of user interactions and non-availability of resources is an important area of active research in media synchronization.

The requirements of timeliness, high throughput, bounds on the loss rate and the need to control skew among different media types make multimedia applications distinct from traditional applications. The term *quality of service* (QOS) has come to represent the entire collection of these requirements and the responsibility of the system to support these requirements is being seen as a contract between the system

and the applications [15,80,81]. The notion of QOS has its origins in telecommunication systems. The CCITT defines quality of service as “collective effect of service performance which determine the degree of satisfaction of a user of the service”. The OSI Reference Model extends this term to situations where the assurances are not pre-designed but instead, are explicitly negotiated between the service provider and the service user. Therefore, the QOS Requirements are explicitly specified by the clients through a set of well-defined parameters and after having been successfully negotiated, are guaranteed by the service provider. Multi-service networks which provide mechanisms and policies for guaranteeing a quality of service level, in order to make such guarantees, require that not only the QOS requirements but also the traffic flow requirements be specified to the service provider [15].

In distributed multimedia systems, QOS is an end-to-end requirement and includes every resource in the path from an application component to its remote peers. These resources on the path will have to be managed so as to meet the end-to-end QOS required [9]. Since open distributed systems are organized into many layers of abstraction, the QOS parameters will have to be expressed at each layer in its own terms [73,74], and negotiated between the peers at that layer. For example, a QOS parameter at a layer above the transport subsystem would specify a frame rate requirement which will be expressed as packet throughput at the transport layer, as cell rate at the network layer (ATM networks) and so on [51].

1.3 Research Goals, Approach and Scope

In this section, the goals for this research are set based on the characteristics of workstation-based multimedia applications described in the previous section. These goals will be stated in terms of resource management objectives and will be justified in the light of the nature of workstation-based multimedia applications. Then, the solution approach towards meeting these goals is presented. Finally, the scope of this research will be established before identifying the contributions made.

1.3.1 Research Goals

1. Fairness

Workstation-based multimedia applications are expected to coexist with traditional applications. Moreover, in the near future the interactions between traditional applications and multimedia applications is expected to increase. For example, a multimedia computer supported cooperative work (CSCW) application may have to interact with a traditional database or a spread-sheet application. Many multimedia systems are designed to support primarily multimedia applications, often at the expense of other applications. Such a biased resource allocation can not only hinder the progress of those non-multimedia applications but also the multimedia applications as well. For instance, in the experimental study by Nieh *et al.* [55], several pathological situations arose due to assigning priority for multimedia applications over window subsystem.

Fairness refers to the requirement that all applications, whether they are multimedia applications or traditional data-oriented applications must be guaranteed progress of execution as defined by the applications. For continuous media, progress can naturally be defined in terms of meeting the real-time requirements. For a data-oriented application, progress may be expressed in terms of average throughput. Therefore, resource allocation mechanisms should not be tuned specifically for real-time response or throughput only. In the approach of this research, resource guarantees are made in terms of a guaranteed execution rate (or utilization) which in turn can be appropriately used by each set of applications to meet their notion of progress.

2. Flexibility

Advance resource reservation is a means for providing a minimum performance guarantee independent of the behavior of other applications in the system. In order to make these guarantees, the resource allocation schemes construct a model of workload and the interactions, parameterize the model into a few simple parameters and expect the applications to specify their resource require-

ments succinctly in terms of these parameters. It is often the case that the real resource requirement of applications significantly deviate from the model assumptions and specified parameters. This deviation leads to inefficiency and ineffectiveness of the reservation-based schemes to meet their performance objectives. On the other hand, reservation-less systems, while achieving high utilization leave the applications open to interference by other workloads.

Flexibility is the requirement that applications be provided sufficient leeway in making resource requirements specifications which will allow them to have variable demands. At the same time they should have some form of protection against interference from competing workloads.

3. Effectiveness and Efficiency

Quality of service is often seen as a contract between the service provider and the user. This requires that the applications specify their performance requirements as well as workload requirements in advance and the service provider makes arrangements for meeting these requirements. The main challenge in this model is to estimate the QOS and resource requirements of the application. It is often derived from the nature of the media types and assumed to be fixed. On the contrary, the QOS requirements is an end-to-end requirement, arising ultimately from the user's perceptual needs, which is dynamic and subjective. If the performance requirements do not capture the end-user requirements at any given time, successfully meeting the performance specifications is of little value.

Effectiveness goal requires that resource allocation decisions be dynamic so that the resources are appropriately focused in accordance with the user's current requirements.

Efficiency goal is related to the above effectiveness goal in that by effectively focusing the resources in tune with the user's requirements, unnecessary wastage in resources allocated with insufficient return in value can be avoided. This is especially important in frequently resource-constrained situations such as

common workstation platforms.

1.3.2 Research Approach

The approach taken in this research is an application and user-oriented approach. The fundamental assumption behind this approach is that multimedia capable workstations of today and in the near future will be operating in a window of scarcity [24] and hence, mechanisms and policies for reducing the impact of overloads on the performance of multimedia applications are required. In particular, it is argued that significant gains in efficiency and effectiveness can be achieved by utilizing information about the application and user states and behavioral patterns when making resource allocation decisions.

The second emphasis of this approach is on the need for balancing between a pro-active reservation based approach and a completely reactive approach to dealing with resource overloads. In accordance with this, a single mechanism for providing isolation among competing workloads while encouraging sharing among cooperating workloads is sought.

A third emphasis of this approach is the need for a composite model of Quality of Service and resource demand of an application as a whole, based on the interrelationship among the application components. This is in contrast to the prevalent practice of ignoring these relationships altogether. This model of application level quality of service drives the policies developed here for adaptation to resource constraints.

1.3.3 Scope of Research

This research deals mainly with the allocation of CPU as a resource for workstation-based distributed multimedia applications. In the rest of this dissertation, the term resource is used to mean CPU unless otherwise stated. The term multimedia workstation brings up the image of a multitasking system dedicated to a primary user who is logged on through the console. Even though remote users and computations may be present, they are treated as non-critical and hence are only provided with

any surplus capacity that is not needed by the console user. Moreover, all the media devices such as audio player and the video display are dedicated to the console user.

The CPU allocation strategies studied in this research deal with a more general type of multiuser workstations. The fundamental premise is that remote computations and users are not necessarily of secondary importance, but require resources as first-class users of the system on equal footing with the console user. For example, the use of network computing to achieve speedup in parallel computations is becoming increasingly common. This is evidenced by the number of parallel applications developed using MPI [50], PVM [19] and similar software on network of workstations. Several scientific computations are being carried out across wide area networks as well. These computations, especially in scientific visualization applications, require performance guarantees as well. The assumption that only a single multimedia application will be running on a workstation is also not necessarily realistic. For example, the use of agents to gather multimedia information, and data mining is an emerging practice. In a corporate environment, the agents run on a server on behalf of several users within the domain and fetch multimedia data across the internet. An example of such a system can be found in the News-on-Demand application being developed by NyNex and Dow Jones [48]. The use of distributed collaborative multimedia applications between several participants in geographically distributed business units of a corporation is also becoming common. In such scenarios, users at a single location participate in the collaboration using their desktops through a conferencing multiplexer system which at each location. In such intermediate systems, several multimedia streams are simultaneously active and require QOS guarantees from the system. Therefore, the scope of the resource allocation approach developed in this research has been defined in this wider sense in contrast to the narrower notion of workstation environments.

Both stored multimedia presentation applications and live conversational distributed multimedia applications are considered. Example classes of applications include desktop video conferencing, computer supported cooperative work (CSCW) and computer-mediated tele-presence applications such as medical teleconferencing

applications. The research focuses on specific characteristics of these applications that have bearing on their resource usage. Particularly, a common set of behavioral patterns are extracted. These patterns are shown to have the beneficial effect of reducing the resource requirements for a given level of performance. The resource reservation and sharing algorithms developed here are aimed at exploiting these patterns.

1.3.4 Contributions

Algorithms for making resource reservations, scheduling, policing (enforcement) and adaptation to reduced resource availability are the main focus of this research. An application-wide resource and QOS requirement model is assumed based on which a two-level allocation architecture is investigated. Architectural framework, scheduling policies and QOS adaptation algorithms are proposed and investigated.

In summary, the contributions of this research are that:

- this research identifies certain structural and behavioral properties of multimedia application sessions which can be exploited in making resource allocation decisions. The notion of *demand synergy* is introduced to represent such behavior patterns and a focus-based QOS model is constructed to exploit them.
- it proposes a two-level model of resource allocation called *group reservation*. Under this model cooperating activities are grouped for allocation of resources and competing groups are isolated from interfering with each other.
- within this framework, it investigates three group scheduling policies for achieving isolation of competition and a QOS control strategy for sharing within a group which exploits demand synergy. The three scheduling policies are shown to achieve isolation equally well but with varying degrees of delay performance.
- the QOS control strategy is shown to meet the QOS requirements in a dynamic and severely competitive environment. Further, the need for effective isolation

in order to successfully meet the QOS requirements in the above scenario is demonstrated. It is also shown that group reservation provides such isolation.

The rest of the dissertation is organized as follows. Chapter 2 provides survey of existing resource allocation practices with a view to defining commonly used terms and concepts as well as identifying the basic issues in resource allocation for QOS support. Chapter 3 critically reviews the resource allocation techniques reported in Chapter 2 with a view to enunciating their deficiencies, in the light of the stated goals of this research. Details of multimedia application structures and behavior are given with a view to understanding and exploiting their impact on resource allocation. A new Application and User-oriented Resource allocation Approach (AURA) is proposed which addresses these aspects. Chapter 4 provides details of the architecture and algorithms used in this new approach. Chapter 5 describes the implementation of the algorithms and their evaluation. Conclusions drawn from this research and further scope for extending this work are described in Chapter 6.

Chapter 2

Background

Early attempts to build multimedia systems used analog technology mainly for capturing, transmission and display of video data [25, 62]. These pioneering projects served the purpose of understanding the basic issues in developing multimedia information systems. With the experience gained from these projects many interesting issues and challenges have come to light. The continuous-time nature of audio and video information, the need for synchronizing audio and video, the advantages of digital implementations of audio and video as well as treating these as fundamental data types in programming are a few of the immediate insights gained. Research on those aspects such as workstation architectures to support multimedia, storage and retrieval architectures and algorithms, hardware and software for image compression, protocol processing for transport and operating system support for efficient handling of data have been reported. While all these issues are important and are still under research, only those ideas and results that are directly of relevance to the problem of CPU allocation in workstations capable of running multimedia applications are presented here.

Category	Example Parameters
Performance-oriented	End-to-end delay, throughput, loss rate
Format-oriented	Video resolution, frame-rate, storage format and compression scheme
Synchronization-oriented	Skew between audio and video sequences
Cost-oriented	Connection and data transmission charges and copyrights
User-oriented	Subjective image and sound quality

Table 2.1: Categories of QOS parameters

2.1 Quality of Service Specification and Provisioning

Vogel *et al.* [80] define Quality of Service as the quantitative and qualitative characteristics required to fulfill the functionality of an application. Examples of qualitative characteristics are color, size and resolution for images and video, stereophonic quality audio or the need for synchronization between audio and video streams. Quantitative requirements are the specification of such performance requirements as throughput and response time and reliability requirements such as error rates. Table 2.1 summarizes the categories of QOS parameters [80].

In distributed multimedia applications where the performance requirements are end-to-end, the quantitative QOS parameters will be specified at each layer of the system in distinct ways and will have to be mapped across the layers. For instance, video frame rate is the throughput requirement at the application layer which gets translated into packet or cell throughput at the communication subsystem. This mapping is not necessarily simple or unique. The nature of the compression scheme might dictate not only the bit rate but also the error rate. For instance, a fixed frame rate video may require a constant bit rate (CBR) transmission under certain circumstance but when coded with MPEG-like compression schemes, will require variable bandwidth. Moreover, the bandwidth availability at the communication layer will in turn affect the compression quality as is the case with certain feed-

Figure 2.1: Examples of Benefit Functions

back-based compression schemes [36].

User-oriented QOS parameters represent a collection of aspects which together give satisfaction to the user and are subjective measures and hence not easily quantifiable. Schrier *et al* [66] proposed the idea of benefit functions to quantify the user satisfaction in achieving a particular level of quality of service. Figure 2.1 illustrates benefit functions for the QOS parameters throughput and delay respectively. Such a function can be used to set a target for the QOS level to be provided.

Schenker [65] describes a set of utility functions to translate service delivered by the network into the application performance. In this framework, given a set of utility functions for the set of applications, the goal of resource management is to maximize the sum of the utilities (efficacy). Efficacy in heterogeneous mix of applications is gained by shifting resources from applications that are not extremely performance-sensitive to those that are. Schenker [65] uses these utility functions and the efficacy to compare different architectural alternatives rather than for actual control. For example, he compares the relative merits of delay-adaptive and rate-adaptive approach in adaptive bandwidth multimedia applications.

In general, meeting the user requirements in each of the QOS categories should result in high user satisfaction. However, this may be expensive in terms of the resources needed to meet the stringent requirements of all the QOS parameters. In many situations, a loss of quality in one of the QOS parameters could be com-

compensated by the meeting or enhancing the quality of other parameters. In other situations, the QOS guarantee on one stream will be more important than that of the other. For example, loss of stereo quality sound is less important under resource constraints when the single audio stream is of excellent quality. Understanding the relative criticality of different QOS parameters on the user-perceived QOS is both important and difficult and several rules of thumb based on empirical evidence have been proposed [32].

QOS parameters are often seen as a contract between the user and the service provider. The service user first will have to specify one or more QOS parameters. These parameters are then mapped on to corresponding parameters at each layer of service. Then, the service provider takes a decision as to its ability to support these requirements. If it decides to accept the requirements then it will have to make internal arrangements to honor these commitments; if rejected, the service user will either renegotiate or abort the application. Several architectures and protocols for negotiation of QOS parameters have been proposed for distributed multimedia systems [52, 54, 81].

Guaranteed Service

The service provider such as the operating system or the network subsystem provides one or more of the following three types of service. In the first category of service called *Guaranteed Service*, the provider gives a guarantee to the client that the negotiated level of QOS will be achieved. This guarantee could be a deterministic guarantee where each data unit is guaranteed a delay bound, or it could be statistical guarantee where a delay bound is assured with a very high probability. Similar guarantees for other QOS parameters such as loss rate, throughput and jitter are also provided. The guarantees may be made to individual connections or to a class of connections. The Tenet suite of protocols assure both deterministic and statistical guarantees on a per-connection basis [15]. In contrast in the MAGNET II system, guarantees are provided on a per-class level [28].

In order to provide guarantees (both deterministic and statistical), the clients

will have to specify their traffic requirements in addition to their QOS requirements. The provider then translates these into a bandwidth requirement and makes reservation for the bandwidth required to meet the QOS requirements. At this time an *admission control* mechanism ensures that admitting the new request will not affect previously committed requests. After this point, a *scheduling mechanism* orders the workload waiting at the resource according to some criterion such as priority, deadline or FCFS which optimizes some performance measure. In addition to admission control and scheduling, a *policing mechanism* is often included to ensure that the clients adhere to the traffic specifications made at the time of reservation. This ensures that no malicious client can disrupt the level of service to other clients. Guaranteed service is suitable for real-time applications that have rigid performance requirements. For example, process control applications and mission critical applications fall in this category.

Predictive Service

The second category of service termed *predictive service* was introduced by Clark *et al* [7] for Integrated Services Packet Networks. The network guarantees a certain quality of service based on currently-observed traffic behavior, with the assumption that the load continues to remain the same in the future. However, applications may attempt to improve their performance by increasing the load offered, provided the system is not overloaded. The system may downgrade the service in response to congestion in the network. Thus, unlike the guaranteed schemes, the QOS agreed is not independent of the behavior of other competing traffic but instead is conditional on their behavior. Hence, the performance bounds are *post facto* bounds as opposed to the *a priori* bounds provided by guarantee schemes.

In this service category, the applications are considered to have a more flexible performance requirement and hence are called adaptive applications. Most multimedia applications, particularly playback type applications fall into this category. In playback applications, a continuous stream of information such as audio or video is sampled, packetized and transmitted by a sender and the receiver depacketizes

and faithfully reproduces the stream's timing. Since the network introduces a jitter, the receiver buffers the received packet to smooth out these variations. The playback point is the time when a packet is to be played back from the buffer. Predictive service is based on the assumption that the applications are flexible enough to move the play-back point to match the observed delay, maintaining a balance between excessive packet-loss due to small play-back delays and excessive delay due to long-time buffering.

Adaptability in applications could be built into the application code or the applications may specify a predefined set of QOS levels and traffic at the time of admission. In the latter case, the provider is given some means to control either the load or the QOS level of the application. Thus for example, a multimedia playback application can either adapt its delay requirement (delay-adaptive) or its rate (rate-adaptive) [65]. Other means of adaptations are also possible. For example, dynamic resolution control is described as a means of congestion control in non-QOS supported networks [78]. This is mainly suitable for video streams where the video's temporal or spatial resolutions are modified in response to congestion.

Best-effort Service

The third category called *best-effort* service offers neither *a priori* nor *a posteriori* guarantees to the applications. It also applies no restrictions on the application's workload. This is the type of service that is typically offered by a time-shared workstation or the Internet. Adaptive multimedia applications such as *nevo*, *nv* and *vat* were originally developed for best effort service of the internet. However, they perform much better with predictive service.

The general principles of QOS provisioning in networks and in end systems are very similar. In fact, some of the service disciplines for allocating link bandwidth such as Earliest Due Date, Static Priority, Hierarchical Round Robin and Packet-Generalized Processor Sharing also called Fair Queuing are adopted from those developed for CPU allocation in end systems. However, there are many differences in the details and objectives. The obvious difference is that scheduling the network

links is inherently in terms of discrete units such as cells or packets and hence, these schedulers are non-preemptive.

The primary difference between networks and workstations is the number of clients as well as jobs in the pending queue for the resource. In networks, particularly backbone networks, both the number of connections and packets are likely to be very high in comparison to the task set of a processor and the number of jobs in the CPU ready to run queue. This difference has many implications on the choice of mechanisms for resource allocation. First of all, the data structures and algorithms have to be very efficient. Secondly, the amount of buffer space will be significantly different. Thus, in network nodes, the emphasis is on simplicity of the service disciplines which will enable them to be implemented in hardware. Switches in wide area networks are usually very simple low cost hardware in contrast to the sophisticated CPU used in workstations. The existence of a large number of independent and identical sources of traffic makes multiplexing gains possible. In contrast, in workstation CPU scheduling, the workloads are heterogeneous and the number of sources are relatively small. The workload sources are also not independent but are dependent on the structure of the applications. Another difference is the fact that the state of the network is only available in a localized manner. This results in local decisions, especially in allocating bandwidth. This is in contrast to the information available for allocating the CPU in a workstation.

2.2 QOS Support in Integrated Services Networks

QOS provisioning in Integrated Services Networks include protocols for resource negotiation and reservation, admission control algorithms, reservation as well policing. Aras *et al.* [4] provide an excellent survey of scheduling and admission control for providing QOS support in integrated services networks. The packet scheduling algorithms for guaranteed service are classified into scheduler-based and rate-based schemes. Scheduler-based schemes analyze the interactions of the packet arrivals in the different streams and determine the service sequence. Included in this cat-

egory are Earliest Due Date (Delay) [16], EDD-Jitter [17] and Shortest Response Time [37]. Rate-based methods translate the QOS requested by the connection into a transmission rate or bandwidth; real-time streams are assigned a rate from a predefined set of allowable rates with corresponding priorities. Some examples of rate-based scheduling methods are Hierarchical Round Robin (HRR) [35], Stop and Go [21], Weighted Fair Queuing (WFQ) [13], Packet Generalized Processor Sharing (PGPS) [57] and Rate controlled static priority (RCSP) [87]. Rate-based schemes, while simpler to implement than scheduler based methods, couple the delay bounds to the transmission rate [4] which is less desirable.

The rationale behind the statistical approach to quality of service provisioning is that it improves utilization of the network due to the multiplexing gain. In computing the performance guarantees, the system needs to have a model for the traffic in statistical terms. The accuracy of the source models are very important for the accuracy of the guarantees made. However, traffic model complexity may be prohibitive. Hence some alternatives have been investigated, which simplify the source model. Yet another alternative approach is to use observed parameters of the traffic to make allocation decisions.

In summary, a variety of scheduling and bandwidth allocation schemes have been investigated in supporting QOS in multi-service networks. The primary goal in designing these schemes is that of simplicity since the network switches, in order to operate at gigabit rates, have to be implemented in hardware. Admission control schemes have to be designed to deal with incomplete information about the network state.

2.3 QOS Support in Workstations

Herrtwich [24] states the following three criteria for successfully meeting the demands of multimedia applications in workstations:

- The capacity of the system should be sufficient to meet the demands. i.e., the raw performance of the system should equal or exceed the requirements

- The resources should be assigned carefully according to the individual requirements of the entities that offer load
- Access to the system resources should be controlled so that competing applications are isolated.

The first requirement is a capacity planning/system sizing requirement and the second is the scheduling of resources according to some criterion of optimality. The third requirement is that of dealing with situations of overload.

2.3.1 Real-time Scheduling for QOS Support

The time-dependent nature of continuous media has prompted the adoption of real-time scheduling theory originally developed for real-time embedded systems for multimedia systems [69]. The rate monotonic scheduling algorithm (RMA) is a fixed priority preemptive scheduler which schedules optimally a given static set of periodic tasks which are independent, do not synchronize or share a common critical resource [41]. The computation model of RMA uses a fixed task-invocation period and a fixed maximum execution time for each task. Given these parameters, the tasks are arranged in decreasing order of their invocation periods. A task set of n such tasks is schedulable using the RMA when

$$\sum_{i=1}^n C_i/P_i \leq n(2^{1/n} - 1), n = 1, 2, \dots \quad (2.1)$$

where, C_i is the deterministic execution time and P_i the period of i th task. The worst-case utilization of a large set of tasks scheduled using RMA is 69% [41]. The average efficiency is about 89% [39].

Earliest Deadline First (EDF) is a dynamic preemptive priority algorithm which arranges the task ready queue in the increasing order of task deadlines. This is also an optimal algorithm and its efficiency is 100%. However, this is more complicated to implement than the RMA. Another drawback of EDF is that under overload, EDF often fails to satisfy the deadlines of any of the tasks. The periodic model although

simpler, is not always realistic. Lehoczky *et al.* extended the RMA to support soft real-time aperiodic tasks, through a Deferrable Server (DS) which handles all aperiodic tasks [71]. This server is assigned a period based on the response-time requirement for the aperiodic tasks. Aperiodic tasks with hard real-time constraints are very difficult to schedule since the scheduling problem has been shown to be computationally intractable [49].

Real-time scheduling algorithms will fail to satisfy the deadline constraints of tasks if the system utilization exceeds the limit imposed by the underlying scheduling algorithm. If these periods of exceeding of required capacity are short and infrequent, these are called *transient overloads*. Such transient overloads may arise due to failure of system components before a standby system starts or due to unexpected prolonging of execution of a task. Under such overloads, some of the computations have to be excluded to process the more important computations. Such a notion of criticality can be expressed in several ways. The period transformation method [68] manipulates the period parameter of tasks to express the notion of criticality when using a rate monotonic scheduler. The imprecise computation model [42] on the other hand fixes the period while splitting the tasks into mandatory and optional subparts and schedules the mandatory parts to meet the deadlines while the remaining capacity is shared among the optional sub-tasks set. There could be many criteria for dividing a task into mandatory and optional parts. This approach has been shown to be useful in monotonic computations which are usually iterative algorithms where the accuracy of the computation (and hence its value) monotonically increases with each iteration. More recently, this idea has been applied to processing video streams compressed using layered coding techniques [27].

Several operating systems have been reported, where the support for multimedia is provided by either extending real-time or time-sharing schedulers [60]. Simple extensions to time sharing systems by providing a Real-Time class with fixed priority scheduler for this class have been attempted [38, 77]. Experimental studies have shown that such schedulers do not satisfy the requirements for multimedia workstations [55]. An important criterion for workstation schedulers is that not only should

they support the timeliness requirements of multimedia applications but should do so without severely limiting the performance of non-multimedia applications as well.

The periodic model of tasks for continuous media streams is prevalent since the media streams derive their timing constraints from the underlying sampling process or frame rates [69]. Graph-based models such as the real time producer/consumer (RT-P/C) data flow model of the YARTOS system have also been proposed [29]. Even though the continuous media stream may be periodic on the average, the stream could have bursty arrivals. Compressed video is known to be bursty and hence the execution time to process a frame of video can vary drastically. Hence, other models such as the Linear Bounded Arrival Process (LBAP) [2, 11] have been proposed. Jeffay [30] suggests a rate-based execution model where a task is specified by three parameters: the number of events in an interval of time, the length of the interval and the deadline of each of the arriving task instances [30].

Continuous Media Schedulers

In the DASH [3] and extended CHORUS [59] systems, the computation model used is the Linear Bounded Arrival Process (LBAP). In this model which was originally proposed to model the packet arrivals in network switches, the workload of an activity is expressed in terms of number of fixed-size messages arriving during an interval. The arrivals are expected to have a nominal arrival rate l and are allowed a finite bounded burst size b . During any interval of time t , the maximum number of messages that are allowed to be in the buffer is bounded from above by $l \cdot t + b$. All messages are assigned a logical arrival time and a deadline. Messages that arrive on time are assigned a logical arrival time same as their actual arrival time. Messages that arrive ahead are assigned a logical arrival time greater than the arrival time, thus artificially smoothing the bursty arrival to appear same as the nominal rate. The deadlines are computed as the sum of logical arrival time and the fixed computation time for the packet. The scheduling model used in DASH has three classes of queues – real-time, non-real-time and work-ahead (WA) and are prioritized in that order. Messages arriving early are assigned to work-ahead queues until

their logical arrival time at which time they are moved to real-time class. Real-time queues are scheduled using EDF policy while the non-real-time class is scheduled in round-robin manner. The policy for work-ahead queues is unspecified. A session reservation protocol (SRP) provides the reserve abstraction and a system call for reducing reserves. Admission control is based on the schedulability test.

2.3.2 Overload Handling and QOS Support

Unlike embedded systems, in workstation environments overloads may be more frequent and persistent. Hence, it is desirable to avoid or minimize the impact of overload on the QOS offered to the applications. Reserving the processor capacity is a technique for avoiding overload. Reservation guarantees sufficient capacity for all the competing computations through an admission control protocol. Depending on how the reservation share is computed, the reservation scheme could be termed pessimistic or optimistic. Pessimistic schemes assume a worst case scenario of workload and make reservations based on that assumption. Therefore, they lead to poor utilization of capacity. Optimistic reservation allows the possibility of overload occurrence but improves utilization. Non-reservation schemes allow maximum utilization but can offer no guarantee *a priori*.

Resource reservation involves the following four aspects [46]:

- specification of resource requirement
- underlying scheduling algorithm
- admission control mechanisms and policies
- enforcement scheme for making applications to adhere to reservations made.

The specification of resources is based on the computational model of the application being served. Such a model specifies not only the amount of resources required but also the manner in which the resource will be utilized. For example, it is common for real-time schedulers to cast the tasks into a periodic model of computation where the specification includes information about the per-instance execution

time of the task and the period. Also included implicitly or explicitly is the task deadline.

Scheduling algorithms usually achieve a sequencing of task instances so as to ensure some performance criteria. A typical algorithm is to arrange the tasks so that all tasks will complete by their deadlines. This sequencing could be done statically (fixed priority schemes) or dynamically. Rate Monotonic and Earliest Deadline First algorithms are the common static and dynamic priority scheduling algorithms respectively.

Admission control is basically a check to ensure that only those workloads are admitted into the system with guarantees, whose resource and deadline requirements could be accommodated without violating those of others. Typical admission control procedures use a schedulability test which is dependent on the underlying scheduling mechanisms used.

It is possible that applications that specified a certain resource requirement may exceed the agreed limit in actual usage. This could be a transient unavoidable violation or could be persistent unplanned or even malicious violation. If the system is to honor its commitments to other tasks independent of the behavior of competing tasks, it is necessary to have an enforcement mechanism that will monitor the resource usage of admitted entities and prevent their violating the agreed resource limits.

2.3.3 Processor Capacity Reservation

The idea of processor capacity reservation of Mercer *et al.* [46] is an extension to end systems of the bandwidth reservation proposals for network QOS support. Each activity specifies a rate of execution R , a period P and a computation time C . Since these three parameters are related by the equation $R = C/P$, specifying two of them is sufficient. If the activity is associated with a periodic stream of information such as a continuous media stream, it is natural to specify its period and (worst-case) computation time. If the activity is aperiodic, the total execution time S and the execution rate R are specified and an average period is computed as $P = S/R$. There

are two classes of ready queues in the system - a real-time class with reservation and a time-sharing class without reservation. The time-sharing class is serviced when there are no jobs in the real-time queue. The scheduling policy used in the real-time class is either the rate-monotonic policy or the Earliest-Deadline policy of priority assignment. Admission control is a simple capacity test for schedulability - by summing up the rates (interpreted as partial-utilizations) of the various activities. Once admitted, the actual computation times of the activities are accounted by a high-resolution timer-based measurement mechanism. The time used by servers and the system kernel while executing on behalf of an activity are charged against the reservation made by that activity. If an activity is found exceeding its resource reservation, its tasks in the ready queue are shifted to the time-sharing class.

2.3.4 Best Effort QOS Provisioning

In contrast to the reservation-based scheduling schemes, Wall *et al.* [83] and Compton *et al.* [8] advocate the use of optimistic reservation-less schemes. In their view, multimedia applications in workstation environments are quite different from hard real time systems in that there is no need for predictability of performance in these applications. Besides, it is assumed that overloads will be transient and infrequent in a well planned and tuned system. Therefore, their approach is to reactively handle overloaded situations. Wall *et al.* propose the use of Time-Driven Scheduler [31]. This scheduler is identical to the earliest deadline first scheduling under normal load. In overloaded conditions, tasks which are assigned a value as a function of time and the task set is scheduled so as to maximize the total value. Compton and Tennenhouse [8] describe single-user environments in which, in resource constrained situations, the user indicates his current focus by giving cues through the user interface such as the window manager. The applications that are not in user focus are degraded by a scheme similar to the slow start congestion control schemes used in TCP/IP protocols. However, they found that when two similar applications are run, their performances tended to diverge [8].

System	Workload Model	Scheduling	Overload Handling	Enforcement
DASH	LBAP	ED + WA	SRP	Traffic Filter
YARTOS	Sporadic	RT-P/C	NA	unspecified
Solaris	Unspecified	FP + RR	User	None
VuSystem	Unspecified	RR	Load Shedding	None
CHORUS	Unspecified	ED + WA	SRP	Traffic filter
RIALTO	Periodic and sporadic server	EDF	Planner	Policing
RT MACH	Periodic sporadic server	RMS+RR	Reservation	Preemption

Table 2.2: Summary of scheduler support for QOS in workstations

2.3.5 Summary

Real time scheduling algorithms are needed in multimedia workstations due to the timeliness requirements of continuous media streams. However, due to the non-critical nature of this timeliness requirement, some amount of deadline misses are acceptable. Since continuous media streams derive their timing from the sampling rate of the stream, periodic models of computations are common; but, the bursty nature of video streams calls for a more flexible and efficient model for video streams. Overload handling is an important requirement for multimedia workstations due to the possibility of frequent overloads. CPU capacity reservation is an abstraction to avoid overloads and to provide guaranteed processing powers to time-sensitive activities. Reservations could be made based on worst-case requirements when absolute guarantees of performance are sought. Reservations could also be optimistically made to limit but not avoid probability of overloads occurring. Overloads could also be allowed to occur and dealt with explicitly by load shedding. Table 2.2 provides a summary of existing scheduling algorithms for workstation CPU in terms of the workload model used, the service discipline, means for overload handling and enforcement.

2.4 Dynamic Quality of Service Management

A common situation in workstation-based multimedia systems is that in response to the user's requirements, the quality of service for an activity might vary during the lifetime of the activity. This aspect is called dynamic QOS and mechanisms for dealing with dynamic QOS have been increasingly recognized as a system requirement. In most schemes the policy decisions for dealing with dynamic resource requirements and non-availability of resources is in the application domain. In the proposal of Mercer *et al.* [46], the interface for specifying the rates of an activity allows for specification of a set or range of rates and for indicating minimum and desirable rates. It is then left to the system to choose the actual allocated reservation. This decision is made either at the time of admitting the activity or during the life-time of the activity.

Mercer and Rajkumar [45] describe a dynamic QOS management scheme based on monitoring. The monitoring tool provides a visual interface to view the allocated and actual resource usage for each activity. Users are allowed to modify the allocation by using this interface. The modifications are realized through a QOS server. The QOS manager uses static user defined policies for dealing with resource constraints in a configuration file. These rules state the relative importance of different applications such as "an audio/video application is more important than a file transfer application" or as "a recently created application is preferred over an older one" [45].

The QOS manager is expected to formulate policies for altering the reservations to different activities based on the static preferences and more elaborate knowledge about the application's use of resources. Two simple QOS managers namely, reactive QOS manager and pro-active QOS manager are also described by them. The reactive QOS manager monitors system load and if the load becomes high enough that new reservation requests are not accommodated, it downgrades the already admitted activities. In the case of the pro-active QOS manager, in addition to desired and minimum reserve levels, the activities specify adjustment priority and adjustment

granularity. Adjustment priority indicates the priority of the activity with respect to downgrading or upgrading its reserved level as the system load changes. Adjustment granularity determines the step-size in the rate for downgrading or upgrading. This has been introduced with the view that rate requirements step sizes of video-like streams are discrete and directly related to the frame rates. The pro-active QOS manager also uses information about actual resource usage by activities and adjusts the reservation levels dynamically, but in a pro-active manner.

In the resource management architecture proposed by Jones [34], the QOS manager is called a resource planner. Activities instead of making reservations directly, make a reservation request to the resource planner. The resource planner in turn evaluates many criteria before deciding on the amount of resource to be allocated to this activity. It then makes the request to the underlying resource provider on behalf of the activity. If successfully reserved, the activity is free to use this reserved capacity until notified otherwise. If unsuccessful, the planner notifies the activity which in turn is expected to revise its requirement and retry. During the life-time of an activity, the QOS and the associated resource requirement may change. This can be indicated either by the activity explicitly or the resource provider may indicate detection of persistent overload. In either case, the resource planner renegotiates for the resources. In doing this the resource planner uses user-specified policies and preferences for apportioning the available resources.

2.5 Summary

The need for real-time scheduling algorithms in workstation environments arises due to the timeliness requirement of continuous media. Unlike the time-critical nature of hard real-time systems, continuous media are only time-sensitive and hence, missing a few deadlines is not catastrophic. Rate Monotonic or Earliest Deadline schedulers have been widely adopted for scheduling Continuous Media. However, scheduling by itself is insufficient when resource overloading is frequent. Processor capacity reservation is a way to assure a certain share of resource for an activity in the face

of competition. Reservations, if made pessimistically, waste CPU bandwidth but give absolute guarantees. Optimistic reservations allow a non-zero probability of overloads occurring and hence can give only statistical guarantees. Reservation-less scheduling schemes can only provide best-effort services.

Workstation environments have dynamic mix of independently authored applications with time-varying QOS requirements. The type of applications and their QOS requirements vary widely. Existing resource allocation and QOS control schemes fall short of the stated goals of this research in such environments as argued in Chapter 3. A new approach will therefore be proposed and evaluated.

Chapter 3

Application and User-Oriented Resource Allocation

In the previous chapter an overview of the issues involved in providing QOS support in workstation environments was provided and a number of scheduling and admission control schemes were described. In this chapter, these schemes are reviewed with respect to the stated goals of this research and several limitations are discovered. A new approach which addressed these limitations is proposed in this chapter.

The need for this Application and User-Oriented Approach (AURA) to resource allocation is motivated by analyzing the structural and temporal relationships of multimedia applications and their impact on the resource requirements and quality of service requirements. It is argued here that a comprehensive model of workload and QOS that encompasses the application structure and user behavior results in better resource allocation decisions. A particular behavior pattern, called *demand synergy* which is believed to be novel and useful, is identified to exist in a wide variety of scenarios. The resource allocation advantages arising out of exploiting this behavior are also discussed. Finally, the AURA model of resource allocation is presented.

3.1 Application and User-oriented Approach

To reiterate, the goals of the present approach to resource allocation as stated in Chapter 1 are, *Fairness*, *Flexibility*, *Effectiveness* and *Efficiency*. An application and user-oriented approach is taken in this research towards meeting these goals. By an application-oriented approach it is meant that resource allocation decisions must ultimately be related to the actual resource requirements of the applications as dictated by the structure of the application and as induced by the dynamic behavior of the user interacting with the application. This requires that all aspects of resource allocation such as the model of workload, the QOS model and the allocation policies and mechanisms will have to reflect as closely as possible the application's true resource requirements. This is in contrast to the current practice of ignoring these properties.

3.1.1 Motivations for a New Approach

The responsibility of providing QOS support in a workstation remains mainly within the domain of the workstation Operating System. This is in accordance with the traditional role of the operating system as primarily a control program for management of resources which ensures the integrity of resource allocation decisions by means of controlled access to resources through a well-defined interface. The survey of schemes presented in Chapter 2 confirms that this paradigm is adopted in most reported schemes for QOS provisioning in multimedia workstations. Accordingly, the operating system constructs a model of the resource requirements and allocates resources according to a set of policies that optimize a certain objective function. For example, the RT-MACH system [46] uses the Rate Monotonic and the Earliest Deadline First (EDF) scheduling algorithms, a periodic arrival and worst-case execution time task model. In this case, the EDF algorithm meets the optimization objective of minimizing the maximum latency of tasks.

Although the above paradigm is well founded, it leaves a considerable semantic gap between the real requirements of the application and the service provided by

the system. These gaps manifest in the following ways:

- Model Restrictions
- Proactive/Reactive Dichotomy
- Inadequate Information and Control
- Application Complexity
- Lack of Portability and Suboptimal Solutions

Model Restrictions

Scheduling and admission control algorithms necessarily impose a model of workload and expect the applications to specify their resource requirements through a set of model parameters. The model not only represents the application requirements but also the assumptions about the behavior of the application as well as the environment in which the application executes. For instance, the Rate Monotonic Scheduling algorithm assumes that the tasks are independent and fully preemptible [41]. The need for analytical tractability dictates that the model be simple and the set of parameters be small.

These factors leave a gap between the resource requirements as exhibited by the applications and the resource requirements as specified to or estimated by the workload model. The implications of such a gap in the modeled workload and the actual requirements are that either the allocation scheme fails to meet the QOS objectives or the resources are not efficiently utilized in meeting these QOS objectives. Consider for example, the periodic task model, commonly adopted in current multimedia systems [69]. In order to provide absolute guarantees on the delay bounds, the QOS provisioning schemes require that the applications specify their worst-case workload requirements, as is the case, for example, in the DASH [3] and the CHORUS [64] systems. Making reservations based on worst-case execution time assumptions, although prevalent, is inefficient when there is a large variability in the execution times. Such large variability in the computation times are likely when the

computation is data-dependent; for example, the compression and decompression and packetization times are dependent on the complexity of the video scene being coded because of the variability (and burstiness) of Variable-Bit-Rate (VBR) coding schemes. The periodic model for task arrivals is too restrictive for the mix of workload in workstation environments. The sporadic server [71] model was developed as an extension to the fixed priority scheduling model for handling non-real-time aperiodic tasks in embedded systems. This idea has also been adopted in RT-MACH system for multimedia workstations [46].

Such extensions are still inadequate for expressing the wide ranging requirements of the application mix in workstations without biasing one class of applications over the other. For example, in the sporadic server model, all aperiodic tasks are serviced by a periodic task. The period of this task is chosen based on the average response time requirements of aperiodic tasks in the system. Still, these are given lower priority in comparison with real-time multimedia tasks. There are two problems with this solution. Unlike embedded systems, a workstation environment has a dynamic mix of independently authored applications. Estimating a common response-time requirement and assigning a period to the sporadic server acceptable to all such applications is not simple. Secondly, biasing towards multimedia applications is not acceptable from the point of view of the fairness goal.

Unlike simple periodic tasks, multimedia application streams have several factors such as application state and user interactions which modify the resource and QOS requirements dynamically. Even a video stream by itself, when compressed produces a long-term correlated stream of large variance bandwidth. Several sophisticated source models such as Markov-modulated deterministic/Poisson processes have been developed to capture the autocorrelation properties of multimedia streams such as video and conversational audio. Models for coded video that describe long range dependencies are also being developed. What is not common however, is the representation of the modulating processes in the resource requirement estimations.

Such a characterization would have to include the application states and the relationship between the application components. For example, in a video on demand

application, the video stream can be in a normal playback mode or in a fast-forward mode, the resource demands for which are significantly different. As another example, consider the scenario where, two or more objects in distinct streams may be presented in unison or in inter-leaved manner in a given application. Such characteristics are admittedly application-specific and also will increase the complexity of the workload models. However, given the resource intensive nature of multimedia applications and the fact that they will have to operate in the so called window of scarcity [24], accurately computing the actual resource requirements and allocating available resources accordingly is very important.

Proactive-Reactive Dichotomy

QOS provisioning schemes in multimedia workstations fall into two categories – reservation-oriented and reservation-less. In the reservation-oriented approach, all tasks in the system are viewed as competing for the CPU. In order to meet their QOS requirements, these tasks will have to make prior reservation request for the workload that they will offer. The admissibility of the request is evaluated using capacity tests which treat the requests as independent from earlier requests. The capacity test is based on current resource allocation status only. In the reservation-less approach, all applications are considered to be cooperating as part of a common problem or under the control of a single user, and it is assumed that resource usage conflicts can be handled by the application in a cooperative manner by load shedding.

In multimedia workstation environments neither of these approaches by itself is adequate. Pro-active admission control is pessimistic in the sense that in trying to isolate activities it increases the blocking probability (or equivalently, the rejection rate) of a resource request. This results in excessive rejection in situations where a more flexible policy would have succeeded. Such a flexible policy will recognize that the new request is a component of a cooperative set, one or more of the members of the set which are already admitted can lower their resource usage to accommodate the new request. Reactive schemes, such as those proposed by Wall *et al* [83], usually result in higher system utilization but at the risk of offering no guaranteed

performance in the presence of overloads. When overloads are frequent, as is the case with multimedia workstations, overload handling through load shedding is both wasteful and inadequate to make any QOS guarantees.

A more flexible approach will therefore need to combine the benefits of both pro-active and reactive approaches. This is possible by distinguishing between those workloads which are *cooperating* and those which are *competing*. Cooperative workloads have logical, structural and temporal relationships that allow them to coordinate their resource demands in response to inadequate resource availability.

Inadequate Information and Control

Both the reservation-oriented and reservation-less systems as such provide little information or controllability to the applications in dealing with overloads. For instance, in reservation-based systems such as those described in CHORUS [10], the admission control policy is FCFS policy where later requests will be rejected when the system is overloaded. This policy ignores the fact that later requests may be more critical to the application than the earlier ones. Even when priorities are associated with resource reservation requests, such as in the RT-MACH system [45], there is the drawback that assigning priorities globally on a system-wide level in a multi-user multi-application dynamic environment is not practical.

Reservation-less systems which allow applications to deal with overloads by load shedding overcome the limitations of FCFS admission policy; but, they make the assumption that overloads are transient, caused by temporary variations in the loads of the media streams which can be dealt with by load shedding and other reactive mechanisms. In multimedia workstations, overloads are frequent due to the launching of several independent applications, often by remote-users and can be persistent. In such situations, the load shedding techniques will be inadequate to sustain the progress of multimedia applications with a useful level of QOS.

Meaningfully dealing with resource overloads require information about the cause of overload as well as the ability to control the distribution of the available resources to activities whose progress are of most value. A new resource allocation mecha-

nism which combines pro-active and reactive strategies, if enhanced with adequate information about the state of the system and the state of the application, can make correct allocation decisions.

Application Complexity

Multimedia applications presently have to deal with not only the content of information and format of its presentation but also with the vagaries of resource availability. The existence of QOS guaranteeing mechanisms, be it absolute, statistical or predicted service, reduces a major burden on the applications' need to deal with resource availability. However, even with absolute guarantee services, the changing requirements of the application together with the dynamic load level common in shared workstation environments require that the applications must be able to have several alternate strategies for resource acquisition, renegotiation and adaptation. This is exacerbated when the underlying system provides only best-effort or less than absolute guarantees. This situation is akin to the early days of multiuser systems where main memory storage was a limited resource and applications had to use overlay techniques to deal with it. The development of demand paged virtual memory techniques have shifted memory management concerns from the applications allowing the application developers to focus primarily on the development of application functionality. Similarly, concerns about CPU resource availability, if taken away from applications, will help developers of multimedia applications to focus on the functionality.

Lack of Portability and Suboptimal Solutions

Most multimedia systems are developed as dedicated systems designed to perform a single set of applications. In doing so, several assumptions are made about the nature of the applications and the environment in which they operate. For example, in the PANDORA system [32], a set of design heuristics to deal with resource non-availability are outlined. However, the basic assumption is that the system is dedicated to a single user and a single application. When such applications are

evolved to multi-user multiple application desk-top platforms, these assumptions become invalid and hence the applications become non-portable.

Application-level handling of resource allocation policies may seem to provide opportunities for customizing decisions to suit the application requirements. However, due to the additional complexity and lack of information about the system state, the applications or the users may often deal with it in sub-optimal manner. Besides, when an application is ported from one environment to another with different resource usage characteristics, the resulting decisions may become suboptimal in the new context.

AURA Approach The AURA approach attempts to overcome these limitations. It does so primarily by distinguishing between competing and cooperating sources of resource demands, and providing mechanisms for isolating competing workloads while encouraging flexible sharing of resources among cooperating workloads. It also identifies a set of cooperative behavior which underlies many apparently distinct application and user behavior patterns. Such a set facilitates dealing with resource related decisions in a common and consistent manner leading to removing complexity, enhancing portability and optimality.

3.2 Application Structure and Behavior

In this section application structures and behaviors are looked at with a view to studying their implications on the resource allocation decisions. Before proceeding with the descriptions, it is necessary to provide definitions of terms frequently used in the rest of this dissertation.

3.2.1 Definitions

The term *resource* refers to hardware resources such as CPU, I/O devices and buffers. However, unless otherwise stated, the term resource will be used interchangeably to

mean CPU. Note that our usage is distinct from the common usage referring to locks.

By the term *application* it is meant a collection of software modules which provide a specific service to the user. For example, a multimedia document presentation software can be considered as an application. The software modules of an application are either developed and linked together into monolithic code or are organized as dynamic linkable libraries such as the X11 and other GUI libraries. Some of the modules may also be organized as servers which are invoked through interprocess communication mechanisms such as Remote Procedure Calls. Examples of modules are software video decompression, protocol processing at one or more layers, device drivers and so on. Fig 3.1 depicts the software modules and their interaction in a desk-top video conferencing application. In this figure, the presentation, recording and conference control modules are application-specific, but the video compression, audio decoding and transport modules are generic and common to many applications. The dashed lines indicate threads of control within a module while the solid lines connecting the modules represent the data and control flow across modules.

An *application session* is the interval of time between the invocation and termination of an application. During an application session, the application creates, activates and destroys several activities dynamically.

A *user-session* is the collection of applications that a user invokes during the period he is logged on to a workstation. During this user-session, the user may launch the applications sequentially, in an overlapped manner or concurrently. For example, in a desk-top video conferencing system the user may have a one or more conferencing applications running concurrently with a local database application.

A *phase* of an application refers to distinct stages in the lifetime of the application. For instance, a conferencing application may go through a set-up phase, a sequence of conferencing phases and a termination phase. The important aspect of the notion of phases is that they are usually sequential and they represent different levels of criticality from the application's point of view with respect to availability of resources. A phase may include several sequential or concurrent activities.

Figure 3.1: Structure of a typical multimedia application. The boxes represent software modules and activities are represented as winding through one or more modules, in each of which they are executed by a thread, shown in dashed lines.

An *activity* is a thread of control and data flow to which resources are allocated and by which resources are consumed. Note that an activity may transcend operating system domains and object boundaries. For instance, when in a client-server interaction, an activity may originate at a client, wind through the server and end back at the client. It is also important to note that when the servers are multi-threaded more than one activity may be executing or ready to execute in the server.

Each activity has one or more *modes* of operation. Modes refer to distinct resource demand states or QOS requirement states of the activity. For example, a video stream activity in a video-on-demand application may exhibit a normal playback mode and fast forward modes. Mode changes in this case are caused by user interactions. The important aspect of modes is that they cause dynamic changes to the resource requirements of an activity. Another example of phase transition in an activity is an image stream where a sequence of image objects are being presented.

Figure 3.2: Dynamic structure of a multimedia session

These image objects could have different resource requirements and when no image object is present the stream is in an “inactive” mode.

During the time an activity is in a given mode, it offers load to the resource at a specific rate. For example, in the normal playback mode of a video stream, the stream is divided into several frames, the frames into packets and packets into cells and so on at different layers of the software. The events at this level are the arrival of tasks (packets, jobs) to the system according to some arrival process and their departure under a service discipline. Fig 3.2 relates these events and intervals in a single application session.

3.2.2 Application Structures

While several structural relationships can be envisioned in a multimedia session, three types of relationships (associations) are of interest to us from resource allocation point of view:

- *Logical Associations*

The term logical association refers to such relationships as producer-consumer and client-server relations that are common in concurrent systems. Producer-consumer relations have bearing on the buffering requirements. In the previous subsection, the structural relationship between modules of an application was illustrated. Such a structure is useful in computing the actual resource requirement of an activity, including the CPU time an activity spends in the application modules as well as in the server processes and operating system modules such as device drivers. This aspect is largely ignored in estimating the resource requirements and hence leads to inaccuracies in reservations. Two notable exceptions to this are the schemes proposed by Mercer *et al* [46] and Jones [33]. The former provides fine grain measurement facility for measuring the time spent by an activity in various modules. Jones [33] suggests that the activity's resource requirement be specified in terms of the time spent in various components and be summed up before making reservations.

- *Temporal Associations*

Temporal association refers to the relationship between two or more events with respect to their start and end times. In the context of synchronization in stored multimedia presentation applications, thirteen relations are possible between two objects. Out of these six relations are distinct when we ignore specific object orderings [76]. We can further simplify these for resource allocation purposes into basically three types – sequential, overlapping and concurrent. Knowledge of this temporal structure is very useful for resource reservation purposes. It was earlier noted that operating system mechanisms model each source of workload as independent and deal with their resource requirements independently as and when the requests arrive. However, the existence of these temporal relationships show that the independence assumptions results in an inaccurate model of the real resource requirements. For instance, when two streams of objects are synchronized, the resource demands for these objects always occur together. On the other hand, when objects in the different

Figure 3.3: Temporal hierarchy and resource management

streams are interleaved, the resource usage will be mutually exclusive. Such cross correlations in the demands are completely overlooked by the models that assume the streams to be independent resulting in either overly optimistic or overly pessimistic quality of service guarantees. In multimedia streams such correlated workloads are highly likely and must be addressed as such.

- *Qualitative Associations*

Qualitative associations refer to the statement of relative importance of concurrent activities or applications in a session. These are useful in dealing with situations of resource overloads.

Apart from the above structural properties, multimedia sessions exhibit a temporal hierarchy of events. Figure 3.3 illustrates this temporal hierarchy. In this figure, the four levels of the temporal hierarchy are shown. On the left hand side,

these levels are labeled according to the entity type of that level. At the lowest level we have tasks, modes at the next level and so on. On the right hand side the levels are labeled with the resource management mechanism(s) that may be employed at that level. At the lowest level there is the scheduling mechanism, at the mode level exists dynamic QOS/Bandwidth adaptation mechanisms; at the activity level, the admission control mechanisms may be present, if needed. At the highest level, which is the session level, a resource planner mechanisms is shown. Some of these mechanisms may fall in the domain of the operating system while the others may fall in the domain of the application or an intermediary agent of the application.

The existence of such a temporal hierarchy has two implications on resource allocation. Primarily, it illustrates the existence of a multiple-level modulation of the underlying arrival/demand processes of the system. Even though at the lowest level the multimedia streams are seen as periodic arrival of workload, the events occurring at higher levels in the hierarchy modulates or even abruptly changes the underlying resource demand process. For example, at the task level, the video stream is offering a periodic load at say a nominal rate of 30 frames-per-second. When a mode change occurs to change the video stream to fast forward mode, the frame rate is scaled up accordingly. Supposing that the scheduling mechanism is not aware of the cause of such a completely new arrival pattern and uses fixed priority scheduling algorithm, the new higher frame rate cannot be supported.

Secondly, events occur at different time scales at the different levels of the temporal hierarchy and at each level a different QOS requirement exists. For example, task periods of activities and activity holding times differ at least by an order of magnitude. At the task level, the QOS parameter of interest may be the meeting of deadlines while at the activity level activity blocking probability may be of interest. The implications are that mechanisms for dealing with QOS errors at each level will be different at the different levels. For example, Delgrossi *et al.* [12] describe several alternatives for QOS adaptation in the presence of overloads, such as media scaling, through scaling temporal or spatial resolution, changing the video compression factor or hierarchical coding and so on. The appropriateness of the mechanism

chosen depends on what causes the overload situation. A transient overload caused by the randomness in the execution time of a frame can be dealt with by simple mechanisms which may be expressed through scheduler parameters such as priority. On the other hand, mode changes cause overloads of intermediate durations and hence may require temporal or spatial scaling mechanisms. On the other hand, an arrival of a new activity brings a much longer duration of overload which require more stable strategies for dealing with the overload.

It is useful to distinguish between two types of variations in the demand of a session, based on the cause, impact and duration of the variations. In the first type of demand variation called hereafter *random demand variation*, the variations are inherent to the nature of the computation, mostly due to data dependencies. For example, the number of iterations in a computation loop may be dependent on the current value of a loop control variable. Another example is the variation in the bit rate of a coded video stream which depends on the scene complexity of the video source. Such variations are difficult to model except as random variables although, in the case of video streams, several sophisticated models are being developed to capture the long-range dependencies. Except for such self-similar sources, the random variations are usually of short durations in comparison with the life time of the activity or sessions.

State-dependent demand variation represent the phenomenon where changes in resource demands occur at the lower level in the temporal hierarchy due to the occurrence of an event at the higher level. This is in contrast to random fluctuations in the resource demand that is inherent to the computation performed at the given level. For example, variations in the task execution time will be seen as random variability whereas change in the arrival rate of a video source due to the transition event from normal playback mode to fast forward mode will be seen as state-dependent demand variation. Table 3.1 summarizes the distinctions between state-dependent and random demand variations.

Capturing state-dependent demand variation is important from two aspects. Firstly, if the event that causes the change in the resource demands is known, a

Random demand variation	State-dependent demand variation
Data/Information-content dependent	Event-dependent
May be modeled as random processes.	Reasonably predicted given some knowledge of event occurrence
Often manifests as bursts of resource demands	Usually manifests as a modulation of the underlying demand process
Operates over short to moderately short intervals relative to lifetime of activities.	Persists over long intervals compared to the time scale of the level's activities

Table 3.1: Random and state-dependent demand variations

more appropriate measure can be used. Thus, dealing with a short term surge in resource demand can be done using simple QOS adaptation policies such as dropping of frames or parts of frames. However, if the overload is persistent due to the state change, this policy is ineffective. Instead a dynamic resolution control such as frame rate control will have to be used.

Capturing state-dependent demand variation is also important in estimating the resource requirement of the session as a whole. This aspect is largely ignored in most workload models. Markov Modulated Deterministic processes have been used to describe the silence-talk spurt behavior of audio streams. Similarly, Markov Modulated Poisson Processes have been used to characterize the correlation structure in video streams. However, these are intrinsic characteristics of a single stream. A notable exception is the layered required bandwidth model proposed by Hui [26]. This model was proposed for ATM networks and is aimed to provide admission control at the cell layer, burst layer and call layer. At each level the blocking probability is to be estimated and the layered bandwidth computation is used in computing this blocking probability. However, the interrelationships among the entities at a layer are ignored in this model. In contrast, the model being suggested by the above structural and temporal hierarchy is different in that, it relates the occurrence of a higher level event to the resource requirements of a lower level event, and it also relates the temporal relationship across activities of an application or applications

of a session.

3.2.3 Synergistic Behavior of Applications and Sessions

In addition to the structural and temporal relationships described so far, a type of behavioral dynamics of multimedia sessions is of interest. Essentially, it shows that in dynamic multimedia sessions, the QOS and resource requirements of individual activities of the session would be time varying, in such manner that the total resource requirement to support the required QOS for each of the activities remains nearly constant. It manifests in two ways:

- Workloads concurrently active in the system offer significantly differing levels of load intensity during their lifetime in phased manner. Transitions from lower to higher load intensity in one of the sources is coupled in time with a corresponding transition from higher to lower load intensity states in one or more workloads so that the total resource demand of the set remains nearly constant.
- Among a large set of concurrently active workload sources, only a small subset of the workload sources are actually active or operating at their peak resource demand levels and the rest of the sources are either inactive or at lower levels of resource requirements. The exact composition of this subset changes with time from among the members of the whole set but the size of the set remains nearly constant.

Note that in the former case, the key aspect of the behavior is the coupling of transitions in the load state while in the latter type it is the dynamicity in the composition of the subset membership. Such behavior is given the name *synergy behavior*, because of the fact that this is a behavior arising out of collective dynamics of the components of a system. When either of the two types of behavior is due to the structural and temporal relationships inherent to the application, then the synergy

Figure 3.4: News presentation application

is termed *inherent synergy*. If this behavior is induced by the user interactions or by the resource manager, then the synergy is called *induced synergy*.

Consider a multimedia presentation application such as a multimedia news presentation. Figure 3.4 depicts the sequence of events in the various streams. This example was first presented by Little [75] in describing a representation model for media synchronization and play out scheduling. This figure depicts two video streams, two audio streams and two image streams. The shaded region depicts the intervals during which an object is present in the stream. It can be seen in this figure that, at the start of the presentation all objects of all the streams are presented for a short time after which objects are mostly interleaved. Particularly, this is the case for the two video streams and the two audio streams. Even though, the presence or absence of an object in a stream causes changes in the bandwidth required during an interval for that stream, such an interleaving of object presentation times across two or more streams leads to a more or less uniform bandwidth for the application as a whole. This is an example of inherent demand synergy.

Yet another example of inherent synergy occurs in video composition applications [86]. In such applications, a single video presentation is composed from multiple video objects. For example, again consider a news presentation application such as the News On Demand application developed at NyNex and Dow Jones [48]. In such applications, the video objects are fetched from several geographically distributed servers over a network connection and are composed into a single presentation at the receiver site. Therefore several video connections are opened between this site and the server sites. However, the video information arriving from the different sites is selectively played out according to a script which is controlling the media mixing. Therefore, even though objects are continuously arriving from the several connections, at any interval only some of the streams are actually part of the presented composite video. This suggests the presence of the second type of synergy behavior – a small subset of active workloads – mentioned before.

Consider a desktop-based medical tele-conferencing system such as described by Little and Ghafoor [76]. Figure 3.5 illustrates the various types of participants in

Figure 3.5: Medical tele-conferencing session

the conference. This application illustrates two important aspects of desk-top video conferencing applications in general. When the specialist is participating in the conference as a passive listener, the requirement is that he or she be able to see the video and any graphical images being presented by the other participants at the highest possible quality. However at some point in the conversation, the specialist is asked to respond to some questions from the other participants for which he needs to consult his research database. Thus, the specialist now changes from a passive to an active role. Firstly, this would mean that any video/audio and image transmissions from this node will have to be presented at the highest quality required. Thus, the resource demands on this node shifts from presenting the received media streams to transmission of the media streams. Such role changes to the participants leading to a shift in the emphasis of workload from one stream to other is a very common pattern in conferencing applications. This type of role shifts leading to change of resource demands from one stream to the other within an application occurs in many forms in a wide variety of applications,.

It is worthy to note a subtle but, important point here. Shift of emphasis occurs not only within activities of an application but also across applications controlled by

a single user. For example, the fact that the specialist needs to consult the database application requires that the database application be very responsive. However, operating system resource allocation schemes which prioritize continuous media data streams over interactive applications will not meet this requirement when the system is near saturation. In this case, the progress of the conference depends more on the query reply from the database rather than whether or not the video is presented at highest resolution. Thus, applications that are logically related should be able to flexibly share the available resources based on the shifting of emphasis.

The usefulness of studying synergy behavior of multimedia sessions arise from the fact that it can be exploited to allocate resources more efficiently by focusing the available resources to a more needy activity and taking it away from a less needy activity at any given time. Given that multimedia workstations are operating in the window of scarcity, this will be very useful.

3.2.4 Benefits of Demand Synergy

A composite model of resource allocation that includes the structure of the application and the QOS requirements as modified by the user behavior is called a *synergy model* because, the resource requirements of the composite whole is quite different from the sum of the resource requirements of the constituent parts.

Definition 3.1 *A group of activities or applications is said to exhibit demand synergy when the combined resource requirement of the group for a given level of QOS is lower than the sum of the bandwidth required independently to provide the same level of QOS.*

Several possible factors could lead to the above synergistic behavior. A common feature of groups with demand synergy is that the resource demand of individual components of the group vary significantly whereas the combined demand has little variability. This facilitates the efficient allocation of resources to the group as a whole as opposed to allocating them independently. This will be a very fundamental principle to guide the AURA resource allocation approach.

Figure 3.6: A Simple session with synergy

Figure 3.7: A session with no Synergy

This can be illustrated with a simple example. Consider a set of three activities of a multimedia application that are concurrent. Each of these activities further have three distinct modes each with different resource demands. Two possible temporal relationships among the set of activities are depicted in Figure 3.6 and Figure 3.7 respectively. In both the figures, the different modes are depicted using different shaded regions. The resource demand is specified as the fractional utilization of the activity in the given mode. Note that there is a significant variability in the partial utilization of each activity during its life time, from as low as 5% to as high as 25%. However, from Figure 3.6 we can see that the combined utilization of all the activities together is actually a constant (45%). This group evidently exhibits demand synergy as per Definition 3.1. In particular, the maximum combined utilization is significantly lower than the sum of the maximum partial utilizations. In contrast, Figure 3.7 shows a case where the variations in the partial utilization of the activities are such that the maximum combined utilization does not diminish due to aggregation.

The above example shows how there is a potential gain in making resource allocations in a collective manner. In particular, if reservations are made statically for each activity and to meet deterministic QOS guarantees, then reservation will have to be made for the peak requirements. Since no sharing is permitted across independently made reservations, the effect is that the total requirement is the sum of the maximum requirements. In contrast, for the same level of guarantee, if reservations were to be made together for the whole set, the required reservation is for the maximum of the sum of the requirements which is always less than or equal to the former.

Let us assume that the model of computation is the classical periodic model where each activity i is associated with a period P_i and computation time C_i . The resource requirement of such an activity is then its partial utilization U_i computed as $U_i = C_i / P_i$.

Since we recognize that during the life-time of an activity there are several modes with differing demand levels, we extend the computational model of a single activ-

ity to that of a sequence $\{U_{ij}\} j = 1, 2, \dots N_i$ of partial utilizations, where j is the index of the phase. $\{T_{ij}\} j = 1, 2, \dots N_i$ is a sequence of time intervals indicating the length of each phase j of each activity i . For the sake of simplicity we assume that all activities are started at the same time in the beginning of the application session and that the number and length of modes of *all* activities are identical; i.e.,

$$N_i = N, \forall i = 1, 2, \dots M \text{ and}$$

$$T_{ij} = T_j, \forall i = 1, 2, \dots M, \forall j = 1, 2, \dots N. \text{ Let}$$

$$UMAX_i = \max_{1 \leq j \leq N} U_{ij} \quad (3.1)$$

be the maximum resource demand over the holding time of the activity. As in the CPU capacity reservation model of Mercer [47], assume that the resource reservations are made at the arrival of an activity and held for the duration of its holding time. Furthermore, assume that absolute guarantees are required for each activity over its entire lifetime. In this case, reservation should be made for $UMAX_i$ for each activity i over the activity's life time.

The following simple theorem assures that when reservations are made for deterministic guarantees of QOS, the bandwidth required by a collection of activities is always lower than the sum of the bandwidth requirements of individual activities, when resources are reserved statically for the entire duration of the session. The proof is trivial and is based on the observation that maximum of the sum of a set of numbers is always lower than the sum of their maximums.

Theorem 3.1 *Let*

$$URES = \sum_{i=1}^M \max_{1 \leq j \leq N} U_{ij} \quad (3.2)$$

be the reservation required when separately made and

$$UGRPRES = \max_{1 \leq j \leq N} \sum_{i=1}^M U_{ij} \quad (3.3)$$

be the reservation required if collectively made for the group. Then,

$$UGRPRES \leq URES \text{ under all conditions.}$$

Proof:

Intuitively, $UGRPRES = URES$ occurs when all the activities simultaneously offer peak load. This is more formally shown below.

Let $jmax$ be the sub-interval for which the maximum occurs in Equation 3.3. i.e.,

$$\max_{1 \leq j \leq N} \sum_{i=1}^M U_{ij} = \sum_{i=1}^M U_{ijmax} \quad (3.4)$$

case 1:

$$U_{ijmax} = UMAX_i \quad \forall i = 1, 2, \dots, M.$$

Then, from Equations 3.1, 3.2, 3.3 and 3.4,

$$UGRPRES = \sum_{i=1}^M U_{ijmax} = \sum_{i=1}^M UMAX_i = \sum_{i=1}^M \max_{1 \leq j \leq N} U_{ij} = URES. \quad (3.5)$$

This case is depicted in Figure 3.7.

case 2:

$$\exists i \text{ s.t., } U_{ijmax} < UMAX_i$$

In this case again, from Equations 3.1, 3.2, 3.3 and 3.4,

$$UGRPRES = \sum_{i=1}^M U_{ijmax} < \sum_{i=1}^M UMAX_i = \sum_{i=1}^M \max_{1 \leq j \leq N} U_{ij} = URES \quad (3.6)$$

□

Comparison with Statistical Multiplexing

The notion of synergy, as defined above, may appear very similar to the well-known principle of statistical multiplexing. Although in terms of the final effect both may be same, they differ entirely in the nature of its cause. Statistical multiplexing [58] is the phenomenon wherein, “given a *large* set of *uncorrelated* traffic flows, the total amount of bandwidth required to satisfy all the flows is bounded, even though the individual flows may substantially vary the amount of traffic they send”. Statistical multiplexing is based on the strong law of large numbers. Thus, for statistical multiplexing the existence of a large set of uncorrelated identical sources is essential. In contrast, the principle of synergy depends on a relatively smaller set of correlated sources of workload.

In wide area backbone networks which carry the traffic from several subnets and hosts the existence of large set of independent and identical sources of traffic is self-evident. However, workloads of a multimedia application in a workstation environment have structural and temporal relationships as shown in previous sections. Further, comparing with the gigabit bandwidth of wide area optical networks, the bandwidth of an end system or local server CPU is very modest and the number of entities sharing this bandwidth is also relatively low in comparison with link bandwidth sharing. Given these, the search for non-statistical multiplexing behavior to improve workstation resource utilization takes added significance.

Statistical multiplexing achieves gains through potential matching of crests and troughs in the random demand functions. In contrast, synergy addresses the state-dependent demand variation phenomenon. state-dependent demand variation causes coupled (correlated) transitions among the states of entities at a given level when a state change occurs at a higher level. For example, a state change to a fast forward mode in the video stream may be correlated with the state change to the audio stream that leads to silence period in the audio stream. State-dependent demand variation is more persistent and hence any gain in resource utilization due to synergy

present can be significant.

3.3 AURA Approach to Resource Allocation

The AURA approach aims to provide a comprehensive model of resource requirement that includes all structural and behavioral properties described in the previous two sections. However, it is important to ensure that any comprehensive workload model that is used is simple enough to be practicable.

The AURA approach can be summarized in the following steps:

- Characterize the resource requirement of a whole group of interrelated activities treating them as a single complex source. In computing this resource demand of the group, all aspects of the session including structural and temporal relationships and synergy are taken into account.
- Provide a session QOS model for guiding the distribution of resources within the group.
- Make reservation for the required resource using the *Group Reservation* mechanism.
- Collaboratively share this reserved capacity using the QOS model.

Key to controlling the complexity in the workload model in the AURA approach is the fact that the collaborating group of applications is characterized as a single complex source of workload. Although complex, it is expressed in the same way as a simple source would be, a required rate of execution and a period over which this required rate is to be consumed. This will be described in detail in Chapter 4.

The combined resource requirement of the application as a whole, taking into considerations all interrelationships and synergy behavior is called *Effective Required Bandwidth* of the group. This effective required bandwidth can either be estimated based on models of the application and session or can be directly measured from the execution of the application at different levels of synergy. It should be noted that

Figure 3.8: AURA approach to resource allocation

in the AURA approach, the *Effective Required Bandwidth* of a group is assumed to be given as input to the AURA resource management system. This implies that the characterization of the application group is part of the development and configuration of the application. This assumption is valid for the following reasons.

Group reservation assures that if reservation is successful, the application is guaranteed to receive the reserved bandwidth independent of the system load fluctuations. This is useful in designing applications which deal with flexible QOS. The applications can be designed to utilize this capacity in an optimal manner, in accordance with the state of the application. Applications, before entering critical stages, can make reservations to ensure successful completion or can abort. Applications can also deal with resource reservations in advance or in lazy manner dynamically, based on the observations of the system load level and its own state. Finally, when the session has demand synergy, then the applications can exploit this to deal with dynamic resource requirements of the activities without having to actually make dynamic reservations at the activity level.

This characterization of the resource requirement is then a part of the development process. Note that this is usually the case in developing embedded systems. The main difference in the case of multimedia systems is that the bandwidth requirement can be computed with allowance for some variability in the demand, which can be dealt with by flexible QOS requirement of multimedia.

Figure 3.8 depicts the overview of the AURA approach. In this research, only the Group reservation and Collaborative sharing aspects are addressed in detail and are presented later in Chapter 4. The QOS model and the term target QOS will be explained here. Some possible ways in which workload characterization can be done will be outlined.

3.3.1 QOS Model

Figure 3.8 shows that from the application characterization, not only is the effective required bandwidth computed but also the QOS model is derived. This QOS model is explained here. The objective of this QOS model is to provide a group level

QOS which is a combined QOS requirement for the application as a whole. This application QOS is stated as a function of the QOS requirements of individual activities as well as the synergy state of the group. The synergy state of the group is expressed in terms of a *focus vector* $\vec{F}$. This focus vector is a vector of focus levels, one component per activity of the group. Each component F_i represents a relative focus on the activity i and takes values in the range $0 \leq F_i \leq 1$ subjected to the condition $\sum_i F_i = 1$. This focus level will be used as a fundamental parameter to describe the synergy state of the system and represents naturally, the user attention focus level as well as any application state that has a distinct QOS level or demand state.

The QOS model is then described by a utility function $U^{\vec{F}}(\vec{Q})$ for the group and maps the QOS vector $\vec{Q}$ to the set of real values R . Figure 3.9 shows an example of such a utility function for a group with two activities. Note that the components Q_i of the QOS vector $\vec{Q}$ take values as specified by the activity's QOS requirements. For example, a video activity could specify a set of frame rates. A group with two such video activities will then have a utility function which characterizes the utility value to the application of achieving each of the combinations of the frame rates. This function is given for each focus state.

Given a certain reserved capacity of the resource, the application group's QOS space can be divided into a feasible and infeasible region, the boundary of which is determined by the capacity value and the sharing policy used. Figure 3.10 depicts the QOS state space for a two-video-activity multimedia session. The QOS is expressed in terms of frame rates. Both activities have a range of frame rates, from $5fps$ to $30fps$. At any given time, a certain combination of frame rates for the two activities is considered to be optimal based on a utility function that maps the achieved QOS to a utility value. The capacity available determines the feasible set of frame rate combinations. The QOS vector $P(20, 10)$ is a feasible target vector whereas $P'(25, 20)$ is infeasible given the bandwidth for $45fps$ only. One of the functions during characterization is to find the feasible region for the given policy for different capacity levels. The goal of resource sharing policy is to select a feasible

Figure 3.9: Utility function that assigns value to QOS vectors achieved.

Figure 3.10: QOS vector space and feasibility. Besides the minimum acceptable QOS values for each activity, the available bandwidth determines the feasible region of QOS vectors. The slanted line corresponds to an available bandwidth to support a total of 35 frames-per-second.

QOS vector which maximizes the utility value at any given time, for the available capacity.

Within this framework, a simple QOS control strategy for resource sharing within a collaborating group will be presented in Chapter 4.

3.3.2 Discussions

In considering the demand synergy principle, the following questions may naturally arise:

1. *Is this phenomenon common in multimedia applications?*

Example scenarios from stored multimedia presentation applications, through video composition to live tele-conferencing systems were presented to point out the presence of demand synergy. The idea of demand synergy is related to the idea of channel groups developed in the Tenet group [23]. Channel group is principally a specification mechanism for the network clients to describe resource sharing relationships among network flows. Szyperski and Ventre [72] describe the models of interactions among the participants of group communication applications. They classify such interactions into *static*, *dynamic* and *controlled* types. In the controlled type of interactions, the fact that a small subset of participants from the group contribute to the traffic flow is noted. This property is then used in determining the admission controls for connection requests from members of the group [72].

It is important to note that many such behaviors will be observed as more and more multimedia applications are widely deployed. Moreover, the bandwidth intensive nature of continuous media will require that such behavioral patterns which lead to lowering the resource allocation to provide the same level of quality of service that is actually needed by the user be actively sought and exploited. For example, Venkatesh *et al.* [79] describe a service aggregation scheme for video on demand systems. In this work, video connections are classified into three levels of service based on whether or not the user requires

interactions with the server. These types are viz., point-to-point interactive, multi-cast non-interactive and multi-cast interactive. The system then observes the interactions that occur in each connection. If no interactions occur for a stream for an interval Δ units, all such connections within a time domain of δ are aggregated together. When interactions occur in a connection, it is promoted to the interactive service category.

Even though the number of commercial grade multimedia systems is steadily growing, the number of applications in wide spread usage is limited. Therefore, no empirical evidence is available for existence or non-existence of synergistic behavior. However, as the example scenarios presented earlier indicate, these patterns of behavior are in fact likely. The goal of this research is to highlight the possibility and benefits of exploiting such behavioral patterns. By demonstrating the benefit of synergistic behavior when present, and by providing mechanisms and framework which can exploit these, it is hoped that applications can be designed to exhibit and benefit from synergistic behavior. Further, system service pricing policies in future networks and operating systems can influence and encourage the users to exhibit demand synergy. Thus, user sessions that exhibit demand synergy can be preferentially treated under a pricing scheme. However, this requires that the synergy exhibited by a session can be measured and translated into terms suitable for pricing.

User perception studies on single media stream with reference to compression and congestion have been prevalent. For example, Delgrossi *et al* [12] conducted experiments to evaluate media scaling strategies to deal with congestion in the HEITS system. At the other end of the spectrum, psychological aspects of multimedia as a communication paradigm have also been the focus of research by both behavioral psychologists and computer systems developers. For example, McKinlay *et al* [44] studied the effect of turn-taking in computer mediated communications. However, models of user behavior patterns in response to resource limitations have been very limited. Similarly, even though adaptive applications such as *nv* and *vic* have been in use with the best effort

Internet service currently available, not many studies on the structural and temporal properties have been reported.

2. *What information is required to detect and exploit synergy?*

In orchestrated multimedia applications, the temporal relationships among the streams are completely specified. In each stream, for each object, the object's start time and duration are either directly available or can be derived. Several models for specifying the inter-stream temporal relations have been proposed, including the Object Composition Petri Net (OCPN) model [76]. This model has been used for determining the transmission schedule for communication over a best effort service network [20] and over ATM networks with QOS guarantees [84]. In the former case, the transmission schedules are computed by estimating current network delay and compensating for this delay at the sender side by transmitting the objects ahead so as to ensure that minimum buffering is needed at the receiving side. In the latter case, the OCPN specification is used to do dynamic virtual channel establishment on ATM networks. Raghavan *et al.* describe a network traffic source model for orchestrated presentations. In this model, starting from the OCPN model of a session, a probabilistic attributed context free grammar (PACFG) is generated. This model then generates event sequences which are associated with probabilities. The probabilities capture the impact of user interactions on the exact path taken in a presentation. From this model, the bandwidth requirements are computed for the network links. Using a similar technique, it is therefore, possible to extract the collective demand of a multimedia document presentation applications.

Initial multimedia conferencing systems were monolithic applications that handled media specific functions as well as conference control functions such as floor control. However, the current trend in conferencing applications such as *vic* indicates that the functions of transport, conference management and local control and marshalling of resources will be handled independently [67].

For example, McCanne and van Jacobson describe the architecture of the *vic* system to allow composition of applications and tools such as *sd* for conference control, *vat* as visual audio tool and *wb* the white board for shared editing [43]. These applications are composed using a scripting language such as Tcl. In this architecture the “conference bus” mechanism which is a local multi-cast mechanism acts as a control mechanism for event notices to be broadcast among the component applications. Similarly Schulzrinne described a local multi-cast mechanism that serves the same purpose. The important implications from this trend is that focus change and role change events will be readily available in a standard format using a standard mechanism for not only those applications but also to any resource allocation mechanism. For instance, one of the reported features of *vic* is the voice-switched-windows which uses cues from the *vat* tool to focus the current speaker in the video window and the focus messages are broadcast over the conference bus [43].

3.3.3 Related Work

The AURA approach to resource allocation can be summarized as a two-level allocation approach. At the system level, each group is allocated a certain amount of capacity independent of the system load, once reservation is successfully made for this capacity. Within a group, this capacity is distributed using a collaborative sharing mechanism. The idea of multilevel resource allocation commonly appears in several contexts in the literature. In the parallel and multiprocessor allocation studies, allocation of jobs to processing elements is separated from the allocation of the CPU to the task or threads of these jobs [40]. The former is some times called space sharing and the latter is called time sharing. In the AURA approach, making reservations for a certain capacity for the virtual processor can be thought of as space sharing. Again, in multiprocessor scheduling, the idea of coordinated allocation of processors to concurrent communicating threads has been used, as in the case of gang scheduling [56]. Collaborative sharing within a group in AURA uses coordinated scheduling but is based on user and application state. Govindan

and Anderson [22] proposed a two-level scheduler for scheduling multimedia applications. In this approach, the objective is to minimize the overheads due to moving video data between kernel and user domains.

The use of economy-based paradigms for achieving fair allocation of resources has been addressed in multiprocessor allocation and workstation CPU allocation studies. For example, Stoica *et al.* propose a microeconomic scheduler for parallel computers [70]. In this approach, each user is assigned a savings account into which money is deposited at a constant rate. This rate determines the rate at which the system resources are allocated. To run a job, the user creates an expense account for the job to which money is transferred from the savings account. The rate of money transfer controls the job's performance. The scheduler divides time into time-slices. At the beginning of each time-slice, jobs in the ready queue bid prices and the CPU is allocated to the highest bidder. Under the assumption that the running times are bounded from above, the transfer rates are bounded from below and a finite number of users, it can be proved that starvation does not occur [70]. This approach does not explicitly have mechanisms for guaranteeing timeliness unless timeliness can be converted in terms of price. Moreover, guarantees are made for meeting the mean response time requirements rather than deadlines or delay bounds. Another novel mechanism based on economic paradigm is the lottery scheduling algorithm [82]. In this, resource rights are represented by lottery tickets and allocation is determined by holding a lottery and allocating it to the winning ticket. This scheduler provides probabilistic fairness guarantees such that the expected allocation of resources to clients is proportional to the number of tickets they hold [82]. Throughputs are proportional to ticket allocation and response times are inversely proportional to the tickets allocated. The main advantage of this scheduler is its simplicity since it does not require tracking of current usage and updating of priorities as is the case with normal fair share schedulers. However, besides the fact that the guarantees are only probabilistic and will be proportional to the number of tickets held only in the long term, the main disadvantage of this scheduler is that due to the randomness in the scheduler, excessive preemptions and instability may occur.

3.4 Summary

Existing models of workload in resource allocation schemes for multimedia workstations do not sufficiently take into account the interactions among the activities in the system. They also do not provide sufficient information or controllability to the applications in dealing with resource overloads. Mechanisms for dealing with overloads are either too rigid for the mix of applications or are too customized to provide portability and reduce complexity. The AURA approach attempts to overcome these limitations by considering the resource requirements of a group of interacting activities. It distinguishes between transient and persistent overloads and deals with them differently. By identifying and exploiting reduced resource demands of a group, AURA aims to improve the efficiency of the system. Towards this goal, AURA uses a QOS model that assigns a utility value for the application in meeting a specified QOS level. This utility value mapping which is a function of the focus level of the system guides the manner in which resources are to be allocated to the activities of the group.

Chapter 4

AURA: Architecture and Algorithms

So far, the need for an application-oriented approach was presented and the AURA approach was proposed towards meeting this need. In this chapter, the overview of the resource and QOS management solution provided in AURA is first presented. Then, an architectural frame work for realizing the same is provided. Finally, the details of the models and algorithms for resource allocation are described.

4.1 Overview of Resource Allocation

The main goals that drive the AURA approach to resource allocation for QOS support in workstation environments are fairness, flexibility, effectiveness and efficiency. AURA approach provides a two-level resource allocation architecture. The set of all resource consuming activities in the system are partitioned into *cooperating groups*. Resource allocation mechanisms at the operating system level, deal with isolating the groups from interfering with each other and guarantees a certain share of the resource to each group. This guarantee ensures fairness for all types of applications irrespective of whether or not they are real-time, interactive or batch workloads. Resource allocation mechanisms within each group deal with how the guaranteed share of the group is utilized to meet the performance objectives of the group. This being specific to the nature of the application, it is at the user (application) domain. This two level approach therefore achieves the necessary flexibility. The guaranteed

bandwidth together with application-specific information improves the effectiveness of the QOS control actions taken in response to dynamics of the application session. Finally, by making use of resource consumption behavior profile of the group and making resource reservations accordingly better efficiency can be achieved.

The main features of the AURA approach are:

1. It distinguishes between cooperating and competing workloads in the system. Cooperating workloads are usually activities of a single application or applications of a single multimedia session under a single user and they have logical, temporal and qualitative associations among them.
2. Resource requirements for a group of cooperating activities are computed or estimated taking into consideration all the relationships and demand synergy behavior that are present. This composite resource requirement is specified to the resource reservation mechanisms and reservations are made on a per-group basis.
3. A group reservation mechanism performs necessary admission control tests to ensure that the system resources are not over-booked.
4. Group scheduling mechanisms ensure that competing groups are isolated from each other and each is guaranteed a share of CPU that it made reservations for.
5. In addition to the collective resource specification, each group specifies a collective QOS model which will be used to guide the manner in which the group's guaranteed share of the CPU are distributed among the members of the group. This QOS model takes into consideration the application and user states and the relative importance of the activities in determining a target QOS for each member. The group QOS is defined as a function of the QOS of individual members as well as their focus states and is expressed as a utility value to the group when each of the members achieved a particular level of QOS when at a specific focus level.

6. Both pro-active QOS control and reactive QOS adaptation mechanisms are provided as desired for ensuring that the above QOS requirements are met. Either of these mechanisms can be used to ensure that the maximum value for the reserved capacity can be achieved irrespective of the presence and behavior of competing workloads.
7. If the value achieved thus is not sufficient for the group's progress, then the group's reserved capacity can be renegotiated or the group's current session terminates unsuccessfully.

The above approach has several advantages. First of all, reservations can be used by all types of applications in the workstation and not merely by multimedia applications. This together with the fact that the group scheduler guarantees the reserved capacity allows all applications to make progress irrespective of whether or not they are real-time, multimedia, timesharing or batch applications. Secondly, since reservation is made for the entire application as a whole, comprehensive strategies for dealing with resource limitations can be designed into the applications. Thus, applications can have well-defined "checkpoints" or stages where they make reservations. If at such points, sufficient capacity is not available, they can enter into well-defined alternative modes of operation or terminate the application. Contrast this with a reservation-less system or even with a reservation system that makes reservations on a per-activity basis. Activity arrival and holding times are relatively more frequent compared to that of mode changes of the session. So, when an activity fails to acquire a certain minimum reserved capacity, the whole application will stall at that point resulting in undesirable consequences. Reserving at a group level avoids this pitfall and instead provides guaranteed performance independent of system load fluctuation. Fourthly, this approach allows each application to have its own policies for distributing the guaranteed resource among its members. This can be done both pro-actively and reactively. Such a flexibility is required in workstation environments.

4.2 AURA Architecture

AURA architectural frame work facilitates the realization of the AURA resource allocation objectives. In particular, it provides a set of well-defined components that deal with various aspects of resource allocation as well as the interfaces between the components. The AURA architecture is essentially a two level resource allocation architecture. Resources are allocated to groups of applications and their activities by the AURA resource manager which resides in the operating system domain. This resource is in turn distributed among the members of the group by the AURA agent which resides at the application domain. This high-level view of the AURA architecture is shown in Figure 4.1. This figure shows a set of application groups, one AURA agent per group and a single AURA resource manager inside the operating system.

The main functions of the AURA resource manager are to provide isolation between the application groups and guarantee a reserved share of resource to each group. The AURA agent handles all the resource-related functions on behalf of the application group that it represents. The dashed line connecting the AURA agent with the AURA resource manager represents the resource allocation interface provided at the operating system layer. The dotted line between the AURA agent and the application group represents the resource and QOS control interface provided by the AURA agent. The solid line in both cases represents the actual workload in terms of jobs offered by the application group. The AURA agent in this case acts as a traffic filter that controls the actual workload injected into the system ready queue by the application group.

Figure 4.2 shows the internal details of the AURA agent and the AURA resource manager and the interfaces. The AURA resource manager consists of a group scheduler and a group admission control mechanism and a set of buffers for queuing the ready jobs from each group. The AURA agent consists of a QOS controller, a reservation manager and a traffic control mechanism. The reservation manager takes as input the resource requirement profile for the application group. Based on this

Figure 4.1: AURA two-level resource allocation architecture

it makes an initial reservation for the group by invoking the *Grp-reserve* interface system call. This reservation request is handled by the AURA resource manager which applies a group admission control test to ensure that the system is not over-booked. If the *Grp-reserve* request is successful, the AURA agent proceeds with setting the QOS requirements for the member activities of the group. Note that this is in contrast to the reservation systems described in the literature where the operating system reservation mechanism requires that each activity specify its traffic and QOS parameters at the time of reservation.

The reservation manager performs two additional functions as well. It keeps track of the system load and the unreserved capacity of the system. It also monitors the QOS utility value achieved by the group and based on these parameters it makes decisions regarding renegotiation of the reservation by the group. Secondly, it also keeps track of the how the focus distribution of the session is evolving. This information along with the bandwidth allocated and the QOS value achieved are

Figure 4.2: AURA internal structure

updated on to the profile of the application group for future control purposes.

The AURA agent QOS manager takes as input the application group's state information. This state information essentially is the current focus distribution among the component activities of the group. Based on this and the current available bandwidth the QOS manager sets up target QOS values for each activity. The AURA agent receives as feedback from the operating system, the current resource utilization by the group. The focus state information is readily available to the AURA QOS manager because the AURA agent resides in the application domain. In addition, it provides an interface through which the application or the user can indicate whether or not the current QOS level achieved is satisfactory. In Figure 4.2 these are shown as dashed arrows from the AURA Resource Manager and the application session to the AURA agent. Based on these two inputs and the QOS model, the QOS manager applies sharing algorithms to determine how the resources are to be distributed. It does this by controlling the traffic parameters of the activities. The traffic controller determines which jobs of which activities arriving for the group are allowed to enter the group's ready queue maintained by the operating system. Note that this traffic control mechanism will be specific to the nature of the activity. Both pro-active and reactive control mechanisms for traffic control can be supported, depending on the nature of the application.

4.3 Group Reservation for Isolation and Sharing

The concept of group reservation is similar to that of the processor capacity reservation [47] scheme. In group reservation, the basic unit of reservation is a group of activities. Thus, built into this is a provision for sharing among the group members. A group, by making reservations, essentially gets a virtual processor for itself to share among its members. Figure 4.3 illustrates this idea of virtual processor. This figure shows a pie-chart of the share of CPU reserved for each of the groups. The shaded sector is the unreserved system capacity and is available for new group reservation requests or for increasing the reservations of existing groups if needed.

Figure 4.3: Virtual processor share in group reservations

The sub-area within the sector group $G2$ demarcated by dotted lines and labeled $G2'$ indicates a reserved, but currently unused capacity of this group. This capacity will be available to other groups but, group $G2$ can claim it any time it needs it.

Group reservation differs from schemes such as proposed by Ramakrishnan *et al* which partitions the system capacity into pre-defined classes [61] for a video on demand server. Similar partitions based on service classes have been commonly used in network quality of service provisioning schemes [7, 28]. The main difference between these and group reservation is that in the former type of partitioning, the capacity for each class is determined by the network designer or operational authority, whereas in group reservation, the groups can dynamically request any share of the system capacity at any time. Further, instead of statistical multiplexing or simple sharing, the sharing of the reserved capacity among the group members will be according to a model of synergy for the group as a whole.

Group reservation scheme consists of the following aspects:

- a model of computation

- admission control
- scheduling algorithms
- sharing algorithms

4.3.1 Model of Computation for Scheduling

The model of computation here refers to the specification of the execution pattern of each group as a whole from scheduling viewpoint. It is very fundamental in that it is used as the basis for specification of reserved share of the resource, application of admission control, guaranteeing through scheduler. Each group G_i specifies its resource requirement through the pair of parameters $\langle r_i, t_i \rangle$ where $0 < r_i \leq 1$ refers to the rate of execution and $t_i > 0$ is the interval of time (measured in us or ms) over which this rate is achieved. From the application group's point of view, the rate r_i of the group is the execution rate of the virtual processor allocated to it. From a scheduling and policing point of view it is the permitted utilization by the group over each interval of length t_i . Thus, if the rate is specified to be 0.33, and the period 10 ms, the virtual processor has one third the rate of the real processor and the group expects to receive exclusive use of the CPU for 3.3ms in an interval of length 10ms, 6.6ms over an interval of length 20ms and so on. This can be formally defined as:

Definition 4.1 *The rate specification $\langle r, t \rangle$ such that $0 < r \leq 1$ and $t > 0$ for a group means that the group demands $k.r.t$ CPU cycles over any interval of time $k.t, \forall k = 1, 2, \dots$, from the start of service for the group.*

Although a group of activities have no natural period as would be the case for a periodic task, the group can be assigned a period over which the execution rate of the group is to be guaranteed. This model can express the rate requirement of a periodic task set as well as any arbitrary interactive application. For example, assume that the group consists of a set of M periodic tasks with periods P_j and computation times C_j respectively. Then, the partial utilization of each task will be

$U_i = \frac{C_i}{P_i}$ and the required capacity share for the group will be the sum of the partial utilizations. In this case, the rate specification for the group will be $r = \sum_{i=1}^M U_i$. The choice of t is left to the user. Smaller values indicate a stricter rate requirement while larger values indicate a more relaxed rate requirement. It should be noted however, that the group scheduling and enforcement mechanisms have a resolution parameter $R < t$ which will be the granularity for enforcement of rate guarantees rather than the user specified parameter t .

The rate specification in this manner is also suitable for non-periodic applications such as a simple numerical computation or compilation or an interactive query to a database and similar ones that can be found in a normal workstation environment. Even for continuous media streams such as variable bit rate video, such a rate specification may be sufficient. Rate-based specifications have been proposed previously by Jeffay and Bennet [30] and Yau and Lam [85]. In the former, a model with three parameters $\langle x, y, d \rangle$ where x is the number of events to be processed in an interval of time y and d is the response time, i.e., the maximum elapsed time between arrival of an event and the completion of the event processing is used. In the model used by Yau and Lam [85], no response time parameter is specified since the underlying scheduler guarantees execution rate. Our computation model is different from both these models in that it is a model of execution for a group and not a single activity. Hence, no response time parameter will be specified. Instead, the response time (deadline) or any other QOS requirements for the members of the group will be specified to the sharing mechanism within the group rather than to the scheduling mechanism that guarantees the rate for each group.

Note that the rate specification while specifying the average rate of execution does not include any parameter for burstiness specification or any other form of variability of rate. This is in contrast to the Linear Bounded Arrival Process (LBAP) [2] or the flow specifications for ATM networks. The principal justification for the absence of any variability parameter is that the groups are meant in our case to be synergistic in their workload and hence by definition have low variability in their rate requirements. Any variability in the rate requirements are to be handled by

the QOS adaptation mechanism within the group. Any large deviations in the rate requirements will be due to significant changes to the phase of the application group and hence will have to be dealt with by re-negotiating the rate requirements.

Given the above rate specification for the groups, the admission control criterion for a new group G_k requesting a rate r_k is :

$$\sum_{i=1}^M r_i \leq 1 \quad (4.1)$$

where, M is the number of groups in the system including the newly arrived group G_k . If this inequality is not satisfied, the new request is rejected.

Note that the admission control done here differs from call admission control algorithms in that only a capacity test is performed and no schedulability or delay test is performed. This is because, at the group level what is guaranteed is only a specified rate of execution. However, the exact manner in which this rate is delivered to the system will influence the delay bounds or deadline considerations for the group members. The mechanism that guarantees the delivery of the required rate is the scheduling algorithm which will first be described. The QOS delivered to the individual members of the group will be the responsibility of the sharing algorithms which will be described subsequently.

4.4 Group Scheduling Algorithms

It was mentioned that group reservation provides both isolation and sharing. In the AURA approach, the provision of isolation across competing groups is well separated from the mechanisms for sharing within a group. This separation allows us to select appropriate mechanisms independently thereby giving flexibility, which is one of the stated goals of AURA approach. This section describes scheduling algorithms for isolation of groups from each other. These schedulers are referred to as *Group Schedulers* to emphasize the fact that they determine the order in which the groups are serviced.

Figure 4.4: Generic model of group scheduling

A generic model of this approach to scheduling is shown in Figure 4.4. The figure shows that the system has a fixed number of independent queues. Whenever a group makes a reservation, it is assigned to a queue of its own. The goal of the group scheduler is to service these queues in such a way as to guarantee the required rate of execution. This is indicated by the labels at the head of the queues. In this model, each group is assigned to a distinct queue and each queue acts as a buffer for backlogged jobs of a single group. Jobs are introduced into each queue in accordance with the sharing policy for the group. Note that in principle each queue can have its own policy for determining how jobs are queued in it. This is emphasized in the figure by showing at the tail of each queue a different scheduling policy. The sharing policy has complete control over the injection of jobs into the group's queue, the relative ordering of the jobs in the queue as well as which jobs may be dropped upon congestion.

The primary goal of the group schedulers is to guarantee that each group receives a minimum rate of execution r_i specified by it. In addition, the group scheduler in providing this execution rate, should interfere minimally in the QOS guarantees for the individual members of the group. Finally, the group scheduler should be simple to implement and have as little overhead as possible.

Three different scheduling algorithms are considered for providing isolation. These

three schedulers, viz., **GRP-RR**, **GRP-EDF** and **GRP-RC** differ in the manner in which they enforce rate of execution. The GRP-RR scheduler is an adaptation of the classical time sharing schedulers used in workstation environments. The GRP-EDF scheduler is an extension of the EDF schedulers used in real-time systems. The GRP-RC scheduler is derived from the fair share schedulers such as virtual clock and its CPU version, Rate Computation scheduler. These choices present themselves as obvious because of the requirement that a mixture of workloads be allowed to co-exist in a workstation environment and at the same time achieve fairness. Other possible scheduling algorithms can also be considered. For example, instead of EDF, a Minimal Laxity (ML) algorithm can be chosen. Even though the ML algorithm is also optimal, it has several disadvantages over the EDF. For instance, it requires that the priorities be continuously updated as computation progresses as opposed to just when scheduling events occur. This results in severe overheads. Further, it results in instability due to excessive preemptions.

The details of the GRP-RR, GRP-EDF and GRP-RC algorithms are described below and their relative merits are also discussed subsequently. In the next chapter, their performance under different workload conditions will be evaluated and compared.

4.4.1 GRP-RR Scheduler

The goal of the group schedulers is to partition the system capacity according to the specified set of rates for the groups. Ideally, a processor sharing discipline would be able to achieve such a virtual processor partitioning. In an ideal partitioning scheme, each group receives exactly the rate associated with its partition, independent of the workloads in other groups. However, processor sharing is only a theoretical artifice. Round Robin scheduler is an approximation for processor sharing and hence is a natural option for group scheduling as well, to guarantee rates.

In classical round robin scheduling, each queue is selected in turn for servicing in a fixed order. The jobs in the selected queues are serviced for a fixed length of time called a *quantum*. At the end of the quantum or when there are no more jobs

pending in the queue, the next queue will be selected. In *weighted round robin*, every queue is assigned an integer w_i which determines the number or proportion of quantum received by a group in each round. In adapting this to group scheduling, the following refinements are added.

The time line is divided into fixed length allocation cycles. Each allocation cycle of length R is divided into sub-intervals, $Q_i, i = 1, \dots, M$ such that for each queue $Q_i = r_i * R$ where r_i is the rate specified for the group associated with the queue. At the beginning of each allocation cycle, the quanta are replenished for all the queues and any unused quantum is not accumulated. When a backlogged queue is selected in its turn for service, it is given the exclusive use of the CPU for a *maximum* duration of Q_i provided it has sufficient backlog. At the end of this time or when there is no more work, the next queue is selected for service.

The GRP-RR algorithm is presented in figure 4.5. Note that a queue currently in service can be either having positive residual quantum or zero quantum. The latter case arises when there is spare capacity due to lower utilization by some groups. The queue in service is non-preemptible when it has non-zero residual quantum. If it has zero residual quantum, it is fully preemptible by those with non-zero residual quantum.

Since round robin is an approximation of the ideal processor sharing, it is interesting to consider how closely it can approximate processor sharing. In particular, the GRP-RR scheduler approximates the ideal processor sharing by dividing the time line into cycles of length R which is a tunable parameter. Therefore, it is important to understand the impact of the choice of R on the rate guaranteed and the responsiveness of the scheduler to deadlines. In the case of GRP-RR, the value of R determines the length of the interval over which a group is not preemptible. This makes the GRP-RR scheduler sensitive to the choice of R since, longer periods of non-preemptibility implies possible violations of rate guarantees and deadlines. The following simple theorems state the conditions under which the GRP-RR scheduler will provide the rate and deadline guarantees in a worst-case scenario.

It is easy to show that GRP-RR meets the rate specifications of all groups.

Algorithm GRP-RR

Input : scheduler events: *Arrival*, *Departure*, *QuantumExpiration* and *Quantum-Replenishment*

Begin

```

switch (EventType)
case Arrival(group, job):
    If (IdleHost) Then Dispatch(group, job);
    Else
        QueueJob(group, job);
        If (running-group = group) Then
            DoIntraGrpSchedule(running-group);
        Else
            If (QuantumExpired(running-group) Then
                DoIntraGrpSchedule(NextGroupInOrder);
            Else
                ContinueRunningJob();
            EndIf
        EndIf
    EndIf
EndIf

case Departure(running-group, running-job):
    If (BackLogged(running-group)) Then
        DoIntraGrpSchedule(running-group);
    Else
        group := NextGroupInOrder();
        DoIntraGrpSchedule(group);
    EndIf
EndIf

case QuantumExpiration:
    group := NextGroupInOrder();
    DoIntraGrpSchedule(group);

case QuantumReplenishment:
    ReseplenishQuanta();
    ContinueRunningJob();
EndSwitch

```

END Algorithm GRP-RR

Figure 4.5: Pseudo-code for the GRP-RR scheduler

Note that with the GRP-RR algorithm, the second parameter of the group's rate specification, t_i , is not directly used in providing the service. However, this is not a problem since R is usually smaller than the t_i s. If R divides each t_i exactly, then, for each t_i , $\exists k$ such that $t_i = k \cdot R$. Since, in each round the group receives a minimum of $r_i \cdot R$ units of CPU time, provided there is sufficient backlog, it would receive a total of $k \cdot r_i \cdot R$ units of CPU time over an interval $t_i = k \cdot R$, and hence, satisfy the rate requirement according to Definition 4.1.

Theorem 4.1 *For an arbitrary rate specification $\langle r_i, t_i \rangle$, and a scheduler resolution $R < t_i$, the GRP-RR algorithm services the group i at a rate r_i every t_i units of time within a maximum error $\frac{r_i \cdot R}{t_i}$, given that all the groups are sufficiently backlogged.*

Proof: Without loss of generality, let $t_i = k \cdot R + \delta$, $0 \leq \delta < R$. Under an ideal processor sharing the group receives $t_i \cdot r_i$ units of CPU time every t_i units of time. Therefore, $consumption = r_i \cdot (k \cdot R + \delta) = k \cdot r_i \cdot R + r_i \cdot \delta$.

Since it is given that all groups have backlogs, GRP-RR has enforcement mechanism that ensure that a *QuantumExpiration* event occurs at most after q_i units in every cycle of R . In the worst case, during the residual time of length δ the group may not receive any service at all. Therefore, over k rounds, the group would have $consumption = k \cdot q_i$. Therefore, the difference in the consumption is at most $r_i \cdot \delta$ and hence the difference in the rate will be at most $\frac{r_i \cdot \delta}{t_i}$. But since, $0 \leq \delta < R$, the difference is less than $\frac{r_i \cdot R}{t_i}$. $\square$

Note that as $R \rightarrow 0$ or $t_i \rightarrow \infty$, or when $R \ll t_i$, the difference will tend to zero.

From the above theorem it is clear that in each scheduling round of length R , GRP-RR guarantees to each group G_i with reserved rate r_i , a minimum of $Q_i = r_i \cdot R$, units of CPU time. However, this Q_i units is made available as a single slot to the group at a time within this allocation cycle as determined by the GRP-RR algorithm's round robin service order. In the worst case, a particular group may receive service after every other group in each allocation cycle. For such a group, if

its tasks arrive at the beginning of the cycle, then these tasks will face a maximum delay in each round and hence there is a concern as to whether they can meet their deadlines or not. Clearly, R can not be arbitrarily large, since, if the delay is larger than the slack available to this task, then the task will miss its deadline. The following theorem arrives at a *sufficient* condition for a task set to be feasible on a shared processor with GRP-RR scheduler if it is given that the task set is feasible under an isolated processor with rate r_i , for a given scheduler round length R .

Theorem 4.2 *Let there be in the system, M groups G_i , $i = 1, \dots, M$, each with a rate of execution of r_i such that $\sum_{i=1}^M r_i = 1$. Consider a particular group G_i with N_i periodic tasks whose computation times are deterministic and deadlines coincide with their periods.*

If the group G_i is schedulable on a processor with rate r_i and its tasks are ordered in EDF order, then a sufficient condition for the group to be schedulable on a processor partitioned with GRP-RR scheduler with scheduler round length of R and tasks within the group ordered using EDF order is given by

$$r_k \geq \frac{C_k}{P_k - R} \quad (4.2)$$

for an arbitrary task $t_k \in G_i$ with period P_k , computation time T_k , and allocated rate r_k , assuming that all other groups are also fully utilizing their reserved shares.

Proof:

Since the group is schedulable under EDF scheduler on a processor with rate r_i consider the worst case scenario for the EDF scheduler. This occurs when the total utilization by the group $\sum_{k=1}^{N_i} \frac{C_k}{P_k} = r_i$ and when all tasks arrive at the same time. In the GRP-RR partitioning the worst case arises for the group G_i when all its tasks arrive at the start of a scheduler round and this group is the last to be serviced in each round according to the service order. For any task $T_k \in G_i$ with period P_k , computation time C_k and rate r_k , the time to completion F_k after arrival will be given by

$$F_k = \left\lceil \frac{C_k}{r_k \cdot R} \right\rceil \cdot R \quad (4.3)$$

For this task not to miss its deadline, $F_k \leq P_k$ and therefore,

$$\left\lceil \frac{C_k}{r_k \cdot R} \right\rceil \cdot R \leq P_k$$

Or,

$$\left\lceil \frac{C_k}{r_k \cdot R} \right\rceil \leq \frac{P_k}{R} \quad (4.4)$$

But, it is always true that

$$\frac{x}{y} \leq \left\lceil \frac{x}{y} \right\rceil < \frac{x}{y} + 1,$$

Therefore,

$$\left\lceil \frac{C_k}{r_k \cdot R} \right\rceil \leq \frac{P_k}{R}$$

whenever,

$$\frac{C_k}{r_k \cdot R} + 1 \leq \frac{P_k}{R} \quad (4.5)$$

Equation 4.2 follows readily from Equation 4.5 by rearranging the terms. $\square$

From Equation 4.2 it is easy to see that as $R \rightarrow P_k$, the rate required for that task $r_k \rightarrow \infty$. Therefore, a natural question that arises is by how much more should the rate r_k allocated to task T_k be than its partial utilization in order that it will meet its deadline in the worst-case, under GRP-RR partitioning with scheduler round length R . A sufficient condition for the same can be derived from Equation 4.2 as follows:

Let $U_k = \frac{C_k}{P_k}$ be the partial utilization of the task T_k . Therefore, $C_k = U_k \cdot P_k$. Substituting this in Equation 4.2 and letting $r_k = U_k \cdot \delta$ we get,

$$\delta \geq \frac{P_k}{P_k - R} \quad (4.6)$$

From the above equation, we can see that if R is 75% of P_k , then the required rate will be $r_k \geq 4 \cdot U_k$.

GRP-RR algorithm has the advantage that it is very simple to implement. Apart from the buffers for each group, a service order list is required. Queue IDs are added to the rear of the service list and removed from the front of the list. For each queue a bit flag or count is maintained to indicate the backlog status of the queue. Replenishing the quantum requires updating the residual quantum value for all the queues. Jobs in each queue are ordered based on the nature of the intra-group scheduling required by the group. The default order is EDF order for the jobs within the queue. The major drawback of GRP-RR algorithm is that it requires that R be very small if no deadlines are to be missed. But, smaller R values while lowering the maximum delay, may cause increased context switch overheads.

4.4.2 GRP-EDF Scheduler

The Earliest Deadline First (EDF) algorithm is known to be optimal for scheduling a set of periodic task set. Therefore, adapting this for group scheduling is attractive. The system has a set of M queues, and each is associated with a group. Each group specifies a rate r_i and an interval t_i over which the rate is spread. As before, the time line is divided into fixed length cycles of length R . In each cycle, a given group Q_i is permitted a maximum of $q_i = r_i \cdot R$ units of CPU time. The difference between this and GRP-RR is that this q_i units of CPU is allocated in arbitrary fragments in GRP-EDF whereas it is allocated as a single non-preemptible chunk in GRP-RR.

All activities of the group introduce task instance into this queue. The tasks are arranged in each queue locally in EDF order. The group's deadline is set to be the deadline of the task at the head of its queue. The scheduler then selects the task at the head of that queue whose deadline is the earliest.

In GRP-EDF, each queue is associated with the execution rate of the group to which the queue is assigned. GRP-EDF monitors the CPU time consumed by each group over its interval t_i . At any given time, the set of groups are partitioned into two sets: those groups which have not exhausted their rate limit and those which have exhausted it. GRP-EDF then prioritizes the former set over the latter. Only when there are no tasks in the system whose group's have not exhausted their consumption for the current interval, will the tasks from the second set be considered for scheduling. Moreover, a task from the second set, if executing, may be preempted by the arrival of a task from the first set, *independent* of which one has the earlier deadline. Within each set, the tasks are scheduled in EDF order. Thus, so long as all groups have non-zero residual quantum, GRP-EDF behaves exactly identical to Global-EDF. When some of the queues have no backlogs, the extra cycles will be consumed by the group with backlog and the earliest deadline task in the system. The GRP-EDF algorithm is formally presented with the help of the state machine for a single group shown in Figure 4.6.

A group may be in any one of the following states:

INIT-QT: No job of this group is executing and its ready-to-run queue is empty;
The group has non-zero residual quantum.

INIT-OT: Same as INIT-QT except that the group has exhausted its quantum.

RTR-QT: The group has at least one job in the ready to run queue but none of its jobs is executing.

RTR-OT: Same as RTR-QT but, with zero residual quantum.

RUN-QT: The group has its earliest deadline job currently executing. The group run queue may or may not be empty. The group has non-zero residual quantum.

RUN-OT: Same as RUN-QT except for zero residual quantum. When in this state, the group is vulnerable to lose the CPU to any group with non-zero residual quantum irrespective of the new job's deadline.

The events that occur when the group is in any of the above states are described below:

Arr_idle: A job from this group arrives and finds the CPU idling.

Arr_gain: Arrival of a job in this group causing it to gain the CPU either from another job of the same group or from a different group.

Arr: All other job arrivals

Dep_empty Currently running job completes execution and the queue for the group is empty.

Dep_lose Currently running job completes execution causing the group to lose the CPU to another group. This happens because the job currently at the head of the queue for the group does not have the earliest deadline.

Dep All other departures.

Lose A job arrives at a different group which takes away the CPU from this group. This happens either when this group is running with zero residual quantum or has later deadline than the new job.

Gain A job departs from a different group yielding the CPU to this group (because the running group has no more backlogs or has jobs with later deadlines).

Exp_keep Running group exhausted its quantum but there is no other group with non-zero residual quantum and non-empty ready-to-run queue and earlier deadline jobs. This group retains the CPU.

Exp_lose Running group exhausted its quantum and lost the CPU to another group with non-zero residual quantum and non-empty ready-to-run queue.

Repl The system replenished the residual quantum of all groups to their corresponding reserved shares for the new round. No carry overs of unused shares is allowed.

Figure 4.6: Scheduling states and transitions for a single group under GRP-EDF Scheduler

The GRP-EDF scheduling algorithm is presented in Figure 4.7.

The main advantage of GRP-EDF algorithm is that it can guarantee the meeting of deadlines for individual activities of a group, provided the group itself reserved adequate resources. In other words, GRP-EDF is identical to standard EDF when all groups make appropriate reservation for execution rate and conform to the reserved usage. In addition, it provides isolation among the groups by means of the enforcement, which allows preemption of tasks of an over-using group by conforming tasks. Finally, it focuses any unused capacity of non-back-logged queues to the earliest deadline task in the system. Since standard EDF is proven to be optimal for scheduling a set of periodic tasks with deadlines coinciding with the periods, the same is applicable to GRP-EDF as well.

The main drawback of the GRP-EDF algorithm follows from the complexity of the EDF algorithm. EDF algorithms are difficult to implement because of the need for keeping track of task deadlines. In addition, due to the dynamic priority of EDF scheduling, depending on the task arrivals, the number of preemptions may be very high. This results in significant context switch overhead. GRP-EDF allows

Algorithm GRP-EDF

Input : *Arrival, Departure, QuantumExpiration* and *QuantumReplenishment*

Begin

```

    switch (EventType)
    case Arrival(group, job):
        If (IdleHost) Then Dispatch(group, job);
        ElseIf ((running-group = group) or
                (QuantumExpired(running-group) and QuantumExpired(group)) or
                ( $\neg$ QuantumExpired(running-group) and  $\neg$ QuantumExpired(group))) Then
            If (Deadline(running-job)  $\leq$  Deadline(job)) Then
                Enqueue(group, job);
            Else
                Preempt(running-group, running-job, group, job);
            EndIf
        ElseIf (QuantumExpired(running-group)) Then
            Preempt(running-group, running-job, group, job);
        Else
            Enqueue(group, job);
        EndIf
    case Departure(running-group, running-job):
        If ( $\exists grp, | \neg$ QuantumExpired(grp)) Then
            (newgroup, newjob) := EarliestDeadline( $\forall grp | \neg$ QuantumExpired(grp));
        Else
            (newgroup, newjob) := EarliestDeadline( $\forall grp$ );
        EndIf
        Dispatch(newgroup, newjob);
    case QuantumExpiration(running-group, running-job):
        If ( $\exists grp, | \neg$ QuantumExpired(grp)) Then
            (newgroup, newjob) = EarliestDeadline( $\forall grp | \neg$ QuantumExpired(grp));
        Else
            (newgroup, newjob) = EarliestDeadline( $\forall grp$ );
        EndIf
        If (newgroup  $\neq$  running-group) Then
            Preempt(running-group, running-job, newgroup, newjob);
        Else
            Dispatch(newgroup, newjob);
        EndIf
    case QuantumReplenishment:
         $\forall grp$  ResidualQuantum(grp) := QuantumShare(grp);
    EndSwitch

```

END Algorithm GRP-EDF

a group in service to be preempted by a new arrival in another queue if the newly arriving task has the earliest deadline in the system. This preemptibility makes it highly responsive and hence to have a performance comparable to that of EDF schedulers. Similarly, the preemptibility together with the quantum enforcement, makes GRP-EDF achieve rate guarantees accurately. However, the preemptibility results in large number of context switches.

4.4.3 GRP-RC

Given our computation model, it will be desirable to use a scheduler that directly allocates priority based on the desired rate of execution and the actual rate of consumption. Such a priority will necessarily be dynamic. The GRP-RC algorithm is one such algorithm. It is an adaptation of the Rate Control scheduler described in Yau and Lam [85] which in turn, is an adaptation of the Virtual Clock algorithm proposed by Zhang [88]. As before, each group specifies a rate r_i and a period t_i . The system maintains a variable called $Finish_i$ which is initially zero for all groups. This finish time is updated upon every scheduling point and in addition periodically by a high resolution clock. The system also maintains a priority value $RCVal$ which is derived from the $Finish_i$ value as follows:

$$RCval_i = k \cdot t_i \text{ such that, } (k - 1) \cdot t_i \leq Finish_i < k \cdot t_i$$

Whenever a job arrives to a queue i , $Finish_i$ is set to the maximum of current time and $Finish_i$. This is a crucial difference between virtual clock and GRP-RC. This ensures that a group that has been idle for long time does not get higher priority in trying to catch up.

Whenever a job from queue i finishes, $Finish_i$ is updated as :

$$Finish_i = Finish_i + \frac{\Delta_i}{r_i}$$

where, Δ_i is the incremental CPU cycles consumed by queue i since last time its *RCVal* was computed for queue i . *RCVal* is computed as before. Note that the ratio $\frac{\Delta_i}{r_i}$ effectively paces the running group. If it has consumed at a higher rate and thus the numerator is larger, then the *RC val* is larger too. If it lags behind, then it receives a lower *RCVal* and hence a higher priority. This, together with the fact that the *RCVal* computation is performed at every scheduling point as well as at a high resolution clock, ensures that the rate control is accurate.

When the clock event occurs and if the currently running job belongs to group q , then its *Finish* value and *RCVal* are updated similarly.

For all these scheduler events, the scheduler selects the queue with the lowest *RCVal* for intra-group scheduling. The pseudo-code for the GRP-RC algorithm is shown in Figure 4.8.

The GRP-RC algorithm does not deal with deadlines at all but, instead, allocates priorities dynamically based on execution rates required and currently achieved. GRP-RC algorithm is relatively simple to implement. The *RCVal* computation is only done for the group for which an arrival or departure event occurs or the group which is running when the clock event occurs. The dynamic priority will require repositioning this group in the priority order however.

Having described the mechanisms for isolating groups from each other and guaranteeing a minimum execution rate, it is necessary now to investigate the mechanisms for sharing this reserved rate among members of a group.

4.5 Collaborative Sharing Within Group

A general QOS model was described in Chapter 3 which helps in guiding the distribution of available bandwidth to the activities of a collaborating group so that the utility value of the group is maximized. In the following a simple reactive QOS control strategy based on a simplified QOS model is described.

Algorithm GRP-RC

Input : *Arrival, Departure and ClockTick* event

Begin

```

    switch (EventType)
    case Arrival(groupi, jobj):
         $Finish_i = \text{Max}(Finish_i, \text{CurrentTime});$ 
         $RCval_i = k \cdot t_i \text{ such that, } (k - 1) \cdot t_i \leq Finish_i < k \cdot t_i$ 
        If (IdleHost) Then Dispatch(group, job);
        Else
            Enqueue(group, job);
            Reschedule();
        EndIf
    case Departure(running-group, running-job): ,
    case ClockTick(running-group, running-job):
         $Finish_i = Finish_i + \frac{\Delta_i}{r_i}$ 
         $RCval_i = k \cdot t_i \text{ such that, } (k - 1) \cdot t_i \leq Finish_i < k \cdot t_i$ 
        Reschedule();
    EndSwitch

```

END Algorithm GRP-RC

Figure 4.8: Pseudo-code for the GRP-RC algorithm

4.5.1 Focus-based QOS Control

The QOS model, the session focus model and a group reservation provide the necessary framework for achieving the sharing of the reserved resources in a manner that optimizes the QOS value of the application as a whole. QOS control refers to the function of selecting a QOS level to be supported for a given activity. Given a reserved capacity for a group and given the utility function, a feasible QOS vector which maximizes the utility function (which will be referred to as target QOS) can be determined. The resource allocation for the activities then is controlled so that this QOS vector will be achieved. This process is repeated whenever the focus state of the session changes. This approach is called the **pro-active QOS control** approach. The advantages of this approach is its simplicity, provided the determination of the target QOS vector is not computationally inefficient. The drawback of this approach is that, apart from the need for a simple to compute utility function, it provides only a fixed QOS and does not take advantage of any surplus capacity available in the system. On the other hand, a reactive QOS control strategy can also be conceived which adapts to the available resource and attains the target QOS provided the capacity permits. The advantages of such a QOS control strategy are two fold: it can adapt to variability in the group demand and secondly it can be actually used for on-line characterization of the session's resource requirement. If guided by a value function, it can identify the points in the feasible region for available bandwidths which maximize the utility value. Such a reactive QOS control policy is described here.

Reactive QOS Control Strategy

The application sets up a target QOS vector for the session based on its current focus level vector. Given the target QOS vector, this scheme also assumes that the reserved bandwidth is sufficient to meet the target QOS requirements of the activities as determined by their focus level distribution. The main difference is that instead of directly setting the traffic parameters for the activities, it gives feedback to the application to upgrade or downgrade the traffic parameters (and hence the current QOS level). The QOS control scheme receives feedback from the

Algorithm Reactive-QosCtl

Input : Current System Load, session utilization and Achieved Frame Rates of activities of the session

Begin

If (utilization(*session*) < reserve(*session*)) **Then**

 UpgradeWithinReserve(*session*);

ElsIf ($\exists activity \in session | \neg \text{AchievedTgtQOS}(activity)$) **Then**

 Downgrade(*session*);

ElsIf (underutilized(*system*)) **Then**

 Upgrade(*session*);

ElsIf (overloaded(*system*) **and**

 (utilization(*session*) > reservation(*session*))) **Then**

 Downgrade(*session*);

EndIf

END Algorithm Reactive-QosCtl

Figure 4.9: Reactive QOS control algorithm

application regarding whether or not an activity achieved its target QOS level or not. In addition it receives feedback from the system regarding the resource utilization by the group and the system load level. Based on these information, it decides to do upgradation or downgradation of an activity of the group. The activity to be upgraded or downgraded is selected by using the activity's focus level as priority. Thus, when downgrading, the activity with the lowest focus level will be selected and while upgrading the activity with highest focus level will be selected.

The pseudo-code for the reactive QOS control algorithm is formally presented in Figure 4.9.

Note that the above QOS control algorithm essentially partitions the group into those which achieved their target QOS and those which did not. It aims to apply control until the latter set is empty. This will be achieved whenever the available bandwidth is sufficient to meet the target QOS level of all activities. If the available bandwidth exceeds this required bandwidth, then the excess capacity is used to

```

Procedure UpgradeWithinReserve (session)

Input : Current System Load, session utilization and Achieved Frame Rates of activities of the session

Begin
  If ( $\exists activity \in session | \neg AchievedTgtQOS(activity)$ ) Then
    select := HighestFocus( $\forall activity \in session | \neg AchievedTgtQOS(activity)$ );
  Else
    select := HighestFocus( $\forall activity \in session |$ 
                          AchievedTgtQOS(activity) and
                          CurrentQOS(activity) < MaxQOS(activity));
  EndIf
  CurrentQOS(select) := NextQOS(select);

END Procedure UpgradeWithinReserve

```

Figure 4.10: Selection policy for upgrading within reserved capacity

upgrade the QOS level of the highest focus level activity beyond its target QOS level. This will be the case when the system is under utilized. If the available capacity is insufficient to meet the target QOS levels of all the activities, then the QOS controller attempts to achieve the target QOS level of the highest focus level activities first.

It should be noted here that the upgrading of QOS level beyond what can be supported by the reserved capacity can cause hysteresis. This happens when the group upgrades the QOS of one of its activities when it sees the system to be under-utilized; but, this upgrading can cause the system to be saturated and hence, it will result in immediate downgrading back of the activity. In contrast, when upgrading within the group's reserved capacity, such oscillations are not possible because, at the time the reservation is made, the group admission control ensures that the reservation can be guaranteed without overloading the system. Several well known techniques used in dynamic control systems can be applied to deal with

```

Procedure Upgrade (session)

Input : Current System Load, session utilization and Achieved Frame Rates of activities of the session

Begin
  If ( $\exists activity \in session | \neg \text{AchievedTgtQOS}(activity)$ ) Then
    select := HighestFocus( $\forall activity \in session | \neg \text{AchievedTgtQOS}(activity)$ );
  Else
    select := HighestFocus( $\forall activity \in session |$ 
       $\text{AchievedTgtQOS}(activity) \text{ and}$ 
       $\text{CurrentQOS}(activity) < \text{MaxQOS}(activity) \text{ and}$ 
       $\text{AdditionalBandwidth}(activity) \leq \text{SurplusCapacity}(system)$ );
  EndIf
  CurrentQOS(select) := NextQOS(select);

END Procedure Upgrade

```

Figure 4.11: Upgrading QOS beyond QOS guaranteed by reserved share

```

Procedure Downgrade (session)

Input : Current System Load, session utilization and Achieved Frame Rates of activities of the session

Begin
  If ( $\exists activity \in session \mid$  AchievedTgtQOS(activity) and
    CurrentQOS(activity) > TgtQOS(activity)) Then
    select := LowestFocus( $\forall activity \mid$  AchievedTgtQOS(activity) and
      CurrentQOS(activity) > TgtQOS(activity));
  ElsIf ( $\exists activity \mid \neg$ AchievedTgtQOS(activity)) Then
    select := LowestFocus( $\forall activity \mid \neg$ AchievedTgtQOS(activity));
  Else
    select := LowestFocus( $\forall activity \mid$  AchievedtgtQOS(activity) and
      CurrentQOS(activity) = TgtQOS(activity));
  EndIf
END Procedure Downgrade

```

Figure 4.12: Downgrading policy.

Figure 4.13: Feasibility of target QOS vectors and achieved QOS vector in the focus-based reactive collaborative sharing scheme. Besides the minimum acceptable QOS values for each activity, the available bandwidth determines the feasible region of QOS vectors. The slanted line corresponds to an available bandwidth to support a total of 35 frames-per-second.

hysteresis. A simple strategy is to use a delay mechanism. In this strategy, when an activity was immediately downgraded within a small time window from being upgraded, further upgrades are delayed for a fixed period of time even when the system may be underutilized during this time. Another possible strategy is to use a linear estimator at each control point to estimate the system load based on current load and the past estimate.

Figure 4.13 depicts the QOS state space for a two-video-activity multimedia session. The QOS is expressed in terms of frame rates. Both activities have a range of frame rates, from 5fps to 30fps. At any given time, a certain combination of frame rates for the two activities is considered to be optimal based on a utility function that maps the achieved QOS to a utility value. The capacity available determines the feasible set of frame rate combinations. In the figure, a maximum of 35fps can be supported with the available resource. As shown in the figure, activity B

is at a higher focus level than activity A. Now, if the target frame rates for this focus level are feasible, then the collaborative sharing scheme first achieves this. Then, any surplus capacity is used to enhance the QOS for the higher focus level activity. Thus, in the figure, for the target QOS vector of (20, 10), the reactive control scheme actually achieves (25, 10). When the target QOS vector is infeasible, then, the reactive sharing scheme first tries to achieve the target QOS of the higher focus activity (activity B) and then uses the remaining capacity to either achieve whatever feasible QOS level for the other activity or to enhance that of the higher focus activity. Thus, in the figure, two possible achieved QOS vectors are shown for the infeasible target QOS of (25, 20). Either of these two options can be justified. For example, when the utility value achieved beyond target QOS level is not significantly higher but value achieved below the target QOS level steeply falls, the former policy is favorable. On the other hand if the utility value below target QOS level is insignificant in comparison with the additional value from achieving beyond target QOS level, the greedy policy of allocating all available capacity to the highest focus activity may be preferable.

The above reactive QOS control mechanism is very useful in many ways. Apart from meeting the target QOS values its adaptability enhances the QOS achieved whenever the system load will permit. A more important benefit of this algorithm is that it can be used to characterize the application session in terms of the bandwidth utilized and QOS levels achieved as a function of the focus level. This characterization will form the profile for the session and is useful in determining the bandwidth requirements of the application group as well as in providing the basis for pro-active QOS control in the future sessions. The performance of the reactive QOS control mechanism for a multimedia session is evaluated and presented in the next chapter.

4.6 Summary

This Chapter built on the Group Workload and QOS models developed in Chapter 3. An architectural framework for realizing the isolation and sharing under this

workload and QOS model was detailed. Further, a group reservation model together with group scheduling mechanisms for isolation and focus-based QOS control algorithms for sharing were described. Three group scheduling algorithms were presented and their relative merits were compared. A pro-active and a reactive QOS control mechanism that used focus information were presented. In the next chapter the implementations of these algorithms and their performance evaluation results will be presented.

Chapter 5

Implementation and Evaluation

In the previous chapter, three different algorithms for group scheduling and proactive and reactive QOS control algorithms for collaborative sharing were described and their relative merits were discussed. This chapter describes the implementation and evaluation of these algorithms. The basis for choosing PARASOL as the implementation platform is explained along with the main features of PARASOL. Then, the experiments performed to evaluate the group scheduling algorithms are described followed by a discussion of the observed results. Finally, the reactive collaborative sharing strategy is evaluated in a highly competitive environment. The effectiveness of collaborative sharing under a reservation-less, independent per-activity reservation and group reservation are compared.

5.1 Implementation of the Algorithms

The group scheduling and the collaborative sharing algorithms were implemented using PARASOL [53], an execution-based simulation tool that helps in performance prediction of parallel and distributed systems. By execution-based, it is meant that while PARASOL provides the basic framework for generating and organizing events in the system through a set of system calls, the functions of the system being simulated are themselves written by the user and actually executed by PARASOL. Therefore, PARASOL can be viewed both as a simulation environment and as a

prototyping platform.

The choice of PARASOL as an implementation platform instead of a real workstation environment was dictated primarily by the fact that in evaluating scheduling algorithms a controlled environment is required. Modification of schedulers in commercial workstations requires much internal implementation information which is not easily available. PARASOL provides all the necessary underpinnings of a multi-threading package without having to deal with the details of components such as file and device managers which are not central to evaluating scheduling algorithms. Using the execution-based model of PARASOL, arbitrarily complex workload models which include user and application behavior dynamics can be generated and characterized at various levels of abstraction. It is also easy to measure the performance of the scheduler as well as its overheads without having to instrument the internals of a commercial operating system. Finally, PARASOL is highly portable on a multitude of platforms. Several independent experiments were conducted in parallel on a heterogeneous network of inexpensive workstations and PCs running SOLARIS, and LINUX operating systems for quickly arriving at the confidence interval widths for a specified confidence level.

5.1.1 PARASOL Environment

PARASOL as a simulation tool supports both process-oriented and event-oriented discrete simulations. It manages the future events list as well as providing a library of routines for generating pseudo random numbers and for gathering statistics. PARASOL as a prototyping tool, essentially provides all the features of a distributed operating system such as multitasking, inter-task communication and synchronization across nodes.

Although PARASOL provides built-in scheduling mechanisms such as FIFO and preemptive resume schedulers, it supports user-written scheduling policies as well. User schedulers are written similar to application tasks which wait on scheduling events such as the transition of a task from blocked to ready state and the transition

from running to blocked state. Both built-in and user written schedulers can manage allocation of multiple CPUs.

PARASOL supports parallel execution of multiple independent experiments through its Distributed Simulation Manager (DPSM). It distributes PARASOL experiment replications using PVM [19] to achieve execution speedup. Replications of experiments are performed to achieve specified confidence interval for performance statistics. Given a list of statistical variables for which confidence intervals are sought and given the width of the intervals, DPSM monitors the gathered results and determines the number of additional replications required. These are launched as parallel runs on networked machines

5.1.2 Simulation Model Implementation in PARASOL

Most of the evaluations of both group scheduling algorithms and collaborative sharing algorithms were performed through simulation experiments. Besides implementing the scheduling algorithms themselves, the evaluations involved implementing workload and QOS models and statistics gathering.

The PARASOL model executes on two PARASOL nodes. Node 0 executes the initial PARASOL task, called Genesis as well as the various tasks representing the *workload sources*. Node 1 represents the *system model* which includes the CPU, the buffers for jobs queued to receive CPU cycles and the service discipline that allocates the CPU, which in the case of AURA includes the group scheduler and the intra-group scheduling algorithms. The workload model consists of a fixed number of user sessions and activities of each session. The sessions and activities were modeled as PARASOL tasks that execute on node 0 and together act as the workload generator.

The behavior of sessions and activities was modeled as finite state machines. The state of the session state machine captured in essence the joint state of the activities of the session. The session model parameters included the state residence time distribution and transition probabilities. State transitions occurring at the session state machine resulted in change of state for at least one of the activities of the session. The activity states captured the resource and QOS requirements of

Figure 5.1: Parasol model of Simulation

the activity in a given state. Thus, each activity state was associated with parameters describing the arrival process, the service demand process as well as the QOS parameters. Each activity task when in a given state generates workload according to the arrival and service process. The actual workload generated by these tasks, namely the job instances are again PARASOL tasks, but are executed on PARASOL node 1. These PARASOL tasks are referred to as *job tasks* and were created from a dynamic pool of PARASOL tasks. They bring actual load to the CPU on node 1 and after completing service or after being aborted due to buffer overflow or deadline expiration, these tasks are returned to this free pool. Figure 5.1 illustrates the PARASOL implementation of the simulation model.

Only tasks running on node 1 contribute to the consumption of resources in the system under study and hence, only these are monitored for performance evaluation purposes. In particular, the arrival time and completion times of job tasks as well as their CPU consumption are monitored at node 1. Descriptor tables that map a session to its activities and each activity to its job instances allow the computation of performance statistics for activities and sessions from the job level statistics. The session and activity workload model parameters are read from configuration files at the start of simulation. The configuration files also specify the statistics variables of interest and the level of details. Thus, for example, statistics can be captured for an activity on a per-state basis or overall for the activity as a whole.

5.1.3 Implementation Experience

The PARASOL environment was found to be adequate for implementing and evaluating the group scheduling and sharing algorithms. Implementing the scheduler was relatively straight forward with the user-scheduling interface defined by PARASOL. Although the built-in schedulers support multiple CPUs per node as well as distributed environment using links and buses with queuing delays, the implementations used only a single CPU per node. Extending the group schedulers to shared multiprocessor nodes is relatively straightforward. One of the modifications that was required was the capturing of node utilization statistics.

Although PARASOL provides both utilization statistics for the whole node as well as for each PARASOL task, this was inadequate for our purposes. In order to reduce the task creation overhead, PARASOL tasks that carried out the computation for jobs generated by the workload were recycled rather than created afresh, as depicted in Figure 5.1. This dynamic mapping of jobs to parasol tasks made it hard to use the built-in utilization statistics. Further, utilization by jobs had to be mapped back to the activity to which the job belongs and ultimately to the session in which the activities are members. For these reasons, utilization statistics gathering were added as part of the scheduler implementations. Although PARASOL provides mean and variance as well as confidence intervals statistics, it lacked computing histograms and tail distributions which are useful in evaluating QOS guarantees. Therefore, these functions were also added in the above implementations.

A peculiar problem was encountered in implementing the group schedulers. Since PARASOL's simulation clock was advanced only by jobs executing a computation or waiting, several scheduler events which are actually sequential were treated as simultaneous in reference to the simulated clock, and were scheduled in an arbitrary order. For instance, when a job belonging to a group completed its execution exactly on its deadline or exactly on expiration of the group's quantum, these scheduler events were arbitrarily ordered within the future events list because they all have identical simulated clock values. Such non-determinism required additional checks to ensure correctness of the scheduler implementations. It should be noted that in a real implementation, the scheduler operations also consume CPU cycles and hence such race conditions do not arise.

5.2 Evaluation of AURA Resource Allocation

The fundamental aspect of the AURA approach to resource allocation is the separation of mechanisms for achieving isolation of competing workloads and for achieving sharing among collaborating activities. Towards this end, three group scheduling algorithms for isolation and a reactive sharing algorithm were described in the previous

chapter. This section is concerned with the evaluation of the isolation mechanism.

5.2.1 Performance Objectives and Metrics

The primary role of the group scheduling algorithms is to partition the CPU bandwidth into virtual processors such that for each group with a specified rate of execution, a virtual processor of corresponding rate is exclusively available. Three different group scheduling policies, GRP-RR, GRP-EDF and GRP-RC were described previously. In discussing their relative merits, some important differences in their workings were mentioned. For instance, all the three group schedulers divided the time line into scheduling cycles (rounds); GRP-RR and GRP-EDF determined a maximum CPU usage limit for each group which is replenished at the end of each scheduling cycle. This limit is enforced in each scheduling cycle by using a timer. GRP-RC algorithm instead recomputed the *RC-value* of each group at scheduling points including at the end of each scheduling cycle. This *RC-value* computation was based on the actual consumption in this cycle and the reserved rate of execution.

The main difference between the GRP-RR and GRP-EDF was that GRP-RR used a fixed service order whereas GRP-EDF used a dynamic order. Further, under GRP-RR a currently running group was less preemptible than under GRP-EDF. In the latter, if an earlier deadline job arrives at another group with unused reserved bandwidth, the current group is preempted. The length of the non-preemptible interval is critical in the performance of the scheduling algorithms, both for guaranteeing reserved rate of execution as well as for meeting the delay bound requirements. This will be clearly demonstrated when we look at the results of the evaluation experiments later. GRP-RC, gives the CPU for the entire duration of the scheduling cycle for the group with lowest *RC-value*. This *RC-value* is computed not only based on the reserved rate and consumed rate as seen during the current cycle, but also based on the reservation period, as can be seen from the pseudo-code of the GRP-RC algorithm. Two groups with identical *finish-time* value may have different *RC-values*, if their reservation periods differ.

The three schedulers differ in the way they distribute any spare capacity in the system at any given scheduling instant. The GRP-RR scheduler will allocate the spare capacity in the current scheduling cycle to the next group in the round robin service order. GRP-EDF will give it to the group with the earliest deadline at the moment. GRP-RC will distribute it to the group that is most lagging behind in achieving its rate. Whether or not one policy is preferable over the other is debatable and depends on the nature of the applications. For example, if all the spare capacity is distributed to the group with real-time tasks, it may help in reducing the deadlines missed by the tasks. On the other hand, proportional allocation is desirable in a general workstation workload mix.

In an ideal partitioning, the competing workloads are so perfectly isolated that they have no impact on the performance of each other. The QOS performance of a given group under perfect isolation will therefore be purely a function of the workload of the group, its QOS requirement, its capacity share and the sharing algorithm. Therefore, the goal of any group scheduler that provides isolation is to not only achieve the rate specification as closely to the ideal processor partitioning, but also to achieve this in such a manner that the QOS performance are also comparable to that under ideal partitioning.

Finally, the goal of any sharing algorithm is to maximize the QOS achieved by the group for the given bandwidth share or to minimize the bandwidth required to be reserved in order to achieve a fixed QOS. In doing so, it may take advantage of availability of any information about the workload, application state or system state. Using such information, it may make control decisions to change the CPU cycles distributed to the members of the group. But, the effectiveness of such control will depend on the ability of the group to preserve within the group, a certain minimum bandwidth independent of the system load fluctuation. In the absence of such protection, the control decisions will be ineffective as will be demonstrated later in the collaboration experiments.

The objectives of the following performance evaluation experiments can be summarized as to be able to address the following questions:

1. How perfect is the isolation in terms of rate of execution achieved by each competing group?
2. How fairly is any spare capacity, due to insufficient workload in one or more groups or unreserved capacity in the system, distributed among the competing groups?
3. How does the choice of system parameters affect the effectiveness of isolation? How does it affect the QOS performance?
4. How does the manner in which capacity partitioning is achieved under a given group scheduler affect the QOS performance of any group?
5. How well does the reactive collaborative QOS control algorithm perform under heavy competition in a dynamic session? How important is the isolation mechanism in aiding the effectiveness of the sharing algorithm?

The most immediate measure for the isolation effectiveness of the group schedulers is the actual rate of execution achieved by a group i in comparison with the reserved rate r_i that it specified. This achieved rate will have to be measured over the reservation period t_i . During a session, as the system load varies, the actual rate at which the group is serviced may also vary. The lower the variation, the more accurate is the isolation effectiveness with respect to rate of execution.

- *Mean Rate*

The number of CPU cycles actually consumed by the group over the rate period t when divided by t gives the average rate of execution over the given period t . The mean rate is the sample mean of this average consumption per reservation period, with the mean taken over the entire session duration.

It should be noted that fairness of allocation of unreserved spare capacity will also be reflected in the achieved mean rate being higher than the reserved rate.

- *Deadline Miss Ratio*

When jobs miss their deadlines, they are dropped. This is considered a deadline miss error. The fraction of jobs that missed their deadlines is an indicator of the deadline miss probability. It is typical for QOS measures such as delay bounds or tails of delay distributions to be used in multimedia applications. Even though they are suitable from application point of view, they are not convenient measures for evaluating scheduler performance in an application independent manner. This is so because, the delay distributions are specific to each activity and in a heterogeneous mix group of tasks, it is desirable to have a single measure that indicates the QOS performance achieved by the group as a whole. The deadline miss ratio is a convenient alternative. Although the name suggests that it is only applicable for hard real time tasks, this can be used easily for soft real time tasks as well. Later on, when evaluating the QOS of a collaborating group under a sharing algorithm, more direct QOS measure will be used. A counter is incremented whenever a delay bound (or deadline) is violated. This counter can be cumulated for the whole session, for all activities of the session. The ratio of this count over the number jobs arriving for the session will be the session's deadline miss ratio.

- *Preemption Count*

Scheduling overhead primarily depends on the complexity of the scheduling algorithm and its operations for maintaining the system state, which in turn depends on the choice and implementation of the data structures used. GRP-EDF and GRP-RC schedulers maintain dynamic priority lists and hence incur the costs of insertion and deletion into these lists. GRP-RR is relatively simpler to implement. In the following evaluations, instead of measuring the overheads directly only the number of preemptions incurred under each scheduling policy is used. This enables us to compare the policies somewhat independent of the choice of the implementation. If we assume that the cost of context switch overheads outweigh the other operations of the scheduler and if this overhead is common for all the policies, then the number of preemptions is a good

measure of the overhead incurred in each case. It should be noted that in our experiments, preemptions within the group as well as across the groups are counted although, it may be sufficient to count the latter if we are merely interested in the comparison of group schedulers.

5.2.2 Overview of the Experiments

In the following, four categories of experiments will be described each of which address the different performance objectives mentioned previously. In accordance with the objective of the experiments, the type of workloads will be chosen. For the sake of ease of discussion the workloads as well as experiments will be given mnemonics which will be used through out the rest of this chapter.

In the following experiments for evaluating the relative performance of the group schedulers, synthetic workloads were used instead of actual workloads. The main reason for this is that by using synthetic workloads, the specific behavior differences between the three schedulers can be precisely exercised. This is important not only for the purpose of validating the implementation correctness but also for the sake of understanding how a specific behavior of a scheduling algorithm influences its rate allocation and deadline performance. If a real workload such as a video trace or a conference session is used, the performance observed can not be always explained in terms of the behavior of the group scheduling algorithms. This is especially true when a resource other than the CPU, such as disk, buffer, network interface or a software resource becomes the bottleneck.

Types of Workload

Greedy: An activity is said to be greedy if it is capable of using as much CPU capacity as available. Such workloads have no specific period and per-instance execution time. An example of a greedy workload is any compute-bound application such as compilation. Note that strictly speaking, even these jobs have a total finite execution time and eventually complete. However, for the

sake of the experiments described here, these workloads are considered to arrive at the beginning of the session and are present for the entire duration of the experiment. They bring an infinite amount of demand so that they are always able to utilize any CPU cycles, either their own reserved or unreserved or unused capacity.

The role of such workloads is two-fold: primarily they are useful for stress-testing the isolation performance of the schedulers. Secondly, they provide a well-understood form of competition which minimizes the number of workload parameters to be varied in studying certain aspects of the schedulers. For example, when trying to understand the impact of the scheduler resolution parameter on the rate achieved, one need not have to study the system for various workloads with different task periods.

Periodic: These are individual activities with deterministic inter arrival times. Therefore, these workloads are assumed to have hard deadlines. Unless otherwise specified, the deadline of each job instance of the periodic activity is set to be the arrival time of the next job instance. Many hard real time activities fall into this category.

Bursty: These are workloads with a large deviation in demand. Further, they have autocorrelation in the load intensity brought in by them. In the experiments where they are used, they are modeled as Markov Modulated processes. The exact models will be described in the context of the experiments.

It should be noted that burstiness is known to be a basic characteristic of video sources that are compressed using variable-bit-rate (VBR) compression algorithms such as MPEG-II. It is also true that any state-dependent or data-dependent processing could cause burstiness of workload. Developing models for characterizing video traffic is an active area of research. Several sophisticated models that capture the long-range dependencies have been reported. However, for our purposes, a relatively simple characterization based on a Markov-modulated process is sufficient.

Multimedia: This is nothing but a combination of periodic and bursty workloads.

Types of Experiments

Three distinct sets of experiments are described in the following.

- Rate Experiments
 - Greedy workload
 - Periodic workload
- Delay Experiments
 - Periodic and Greedy workload
- Mixedload Experiments
 - Multimedia and Greedy workload
 - Multimedia and Bursty workload

Rate Experiments: The main objective of these experiments is to evaluate the mean rate obtained by the three group schedulers under different competitive environments. The rationale behind these experiments is that a policy like GRP-EDF is primarily deadline oriented and a policy like GRP-RC is primarily rate-oriented. Hence, it would be necessary to see how the relative periodicity and relative reserved capacity affect the allocation.

Rate-Greed-Sym: The system consists of three groups of a single *Greedy* workload each. All the groups had identical reserved shares of 0.25 and the system had an unreserved capacity of 0.25. The reservation periods for each group was 100ms. The scheduler resolution was varied from 1.0ms, 10.0ms, ..., 100ms, 120ms, ..., 200ms.

It is particularly interesting to see how the excess 25% capacity is distributed. Since the workload is completely symmetric, it is reasonable to expect the excess capacity also to be distributed equally. However, the GRP-RR uses a

fixed service order and GRP-EDF uses dynamic deadline based priority. Given that all the workloads are greedy, GRP-EDF will be identical to GRP-RR, since no deadline is earlier than the other.

Rate-Greed-Asym: The workload is identical to *Rate-Greed-Sym* except that the reserved capacities are different. The reserved shares for the three groups are: 0.15, 0.35, 0.25 respectively. The objective here is to see the effect of the relative reservation levels on the rate allocation, particularly the excess 25%.

Rate-Period-Sym: Three groups with single *Periodic* task compete when provided with identical reserved shares of 25% each, leaving an unreserved capacity of 25%. The task periods are deterministic values of 70ms, 50ms and 60ms respectively. The task execution times are also deterministic with values 42ms, 30ms and 36ms respectively. Note that the partial utilizations of these workloads are 60%, which is chosen so that each of these tasks has sufficient demand per job to fully utilize the unreserved capacity of 25%. The goal here is to see how this setup performs in comparison with *Rate-Greed-Sym*, and particularly whether or not the relative periodicity has any effect on the distribution of the slack capacity.

Rate-Period-Asym: This is similar to *Rate-Period-Sym* except that the reservations are asymmetric, 15%, 35% and 25% respectively.

Delay Experiments: In this set of experiments, the objective is to evaluate the delay performance of the groups under the different group schedulers.

Delay-Period-Greed: The system consists of two competing groups. Group 1 consists of a set of three **Periodic** tasks. Group 2 consists of a single **Greedy** task. The reserved shares for Group 1 is varied from 0.1 to 0.9. Correspondingly, the Group 2 reserved share is varied from 0.9 down to 0.1. The scheduler resolution was varied from 1.0 ms, 20.0 ms, ..., 460 ms. The experiment was repeated with a different periodic task sets for Group 1. In each experiment, the task periods were selected from uniformly distributed bins. The first task's periods were picked randomly from a bin of range [75ms, 100ms],

the second from a bin of range $[150ms, 200ms]$ and the third from a bin with range $[300ms, 400ms]$. In each experiment the Group 1 tasks were given service times so that the group will fully utilize its reserved capacity.

It is expected that GRP-EDF, being the most responsive due to being most preemptible will perform very well in terms of both rate and deadline misses. GRP-RC is also expected to perform reasonably well with respect to rate but, with respect to deadline, it is not clear if it will be better than GRP-EDF. Both GRP-EDF and GRP-RC should perform much better than GRP-RR.

Mixedload Experiments: The experiments so far have separately dealt with how the schedulers differed in providing rate guarantees and deadline miss performance guarantees. In doing so, specific workload sets which bring out the key differences between the schedulers were used. In the current set of experiments, a more typical mix of workload is chosen. The idea here is to evaluate the performance of a simple multimedia workload in the presence of competing workloads in a typical workstation environment. The performance being evaluated are not only the QOS performance of the multimedia workload but also the rate of execution of the non real-time workload. Such an evaluation is important because one of the stated goals of the AURA approach is fairness which was defined to be the ability to guarantee progress to all applications in the system instead of preferring one type of application over the other. Thus, a multimedia application should not be allowed to dominate all other non-real-time applications in the system. Similarly, a compute intensive application should not affect the delay performance of a multimedia applications.

MM-Greed: The system consists of two groups of workloads. Group 1 is a *Multimedia* workload consisting of two activities. Activity 1 is a video stream and Activity 2 is an audio stream. The audio stream is modeled by a simple *Periodic* task model and the video stream is modeled as a *Bursty* task. The audio stream is a constant bit rate (CBR) activity with a deterministic inter arrival time $20ms$ between jobs. Each job takes a fixed service time of $4ms$.

Thus the partial utilization by the CBR audio stream is 0.2. The video stream is assumed to be a variable bit rate (VBR) stream such as produced by MPEG compression algorithm.

Although, several sophisticated models for characterizing the video stream bit rate exist, a relatively simpler but representative model was chosen. In this model, the video data arrives at a fixed frame rate of 16 frames-per-second or equivalently, an inter-frame gap of $62.5ms$. Each frame's bit-rate is a random duration ON-OFF process. The ON or active period is uniformly distributed in the range $[10ms, 40ms]$. During the ON period, a constant bit rate of 10 Mbits/s of data is produced. Assuming that packetizing and protocol processing is directly proportional to the number of bits of data in the stream and assuming a $10Mbit/s$ processing rate for packetizing, the service demand per frame of video will be uniformly distributed in the range $[10ms, 40ms]$. Note that this model is identical to the one used for characterizing VBR video streams used in the performance evaluation of the MAGNET-II Real-time Switch described by Hyman *et al* [28]. Note that the peak-to-average ratio of this video stream is 1.6 which is also close to the values of 1.7 reported in [63]. They report a peak-to-average ratio of the bit rates ranging from 1.7, 3, 4, 5 to 7 for MPEG coded movie clips [63] depending on a variety of movies varying from simple to action-packed clips.

Both the audio and video frames are dropped when they missed their deadlines. The deadlines missed by the multimedia workload is evaluated under several competitive conditions in the following sets of experiments.

MM-Greed-Compete is a set of experiments where the multimedia workload is competing against a simple *Greedy* workload. The mean utilization required by the multimedia workload is 0.6 and its reserved capacity is varied from 0.3 to 0.7 in steps of 0.05. The Greedy competition's reserved share is set so that the total reserved capacity is always 0.9. The deadlines missed by the multimedia workload is noted as well as the achieved rate of the greedy

workload under the three schedulers.

In the second set of experiments *MM-Greed-Isol*, the multimedia workload is run in perfect isolation in a rate reduced processor, with rate equivalent to each of the reservation levels in the previous experiments. The idea here is to compare the deadlines missed ratio results of the previous experiments with the perfect isolation situation. Note that since the total reserved capacity was only 0.9 in the previous case, there are two possible assignments of the reduced rate for the isolated processor. An immediate assignment is to have the rates varied from 0.3, 0.35, ..., 0.7. However, the unreserved capacity available to the previous case would not be available now. An alternative assignment is to proportionately distribute the slack of 0.1. Thus the rate assignment will be 0.33, 0.39, ..., 0.78. This is indeed how a GRP-RC scheduler would allocate the slack. Both the possibilities are considered and the performance achieved are compared with the groups in competition. These variations are labeled as *MM-Greed-Isol-1* and *MM-Greed-Isol-2*.

MM-Bursty: As a final variation of the multimedia workload experiments, the multimedia workload is made to compete with a *Bursty* workload instead of the Greedy workload. The objective here is to compare the performance of a multimedia session in the presence of fluctuating system load. In particular, the performance of the session with no reservation will be compared with the performance under a group reservation scheme. For these experiments, only the GRP-EDF algorithm will be used.

The workload for the multimedia session is the same as in the case of the *MM-Greed* experiments. The model for the competing workload is as follows: The competing workload is represented by a single activity which may reside in one of two states. In the low utilization state, it offers a periodic workload with an inter-arrival time of 200ms and a computation time of 40ms. In the high utilization state, the inter arrival time is 40ms and the computational time is 28ms. Thus the partial utilizations in the two states are 0.2 and 0.7. The

state residence times are exponentially distributed with a mean of 2000ms. The state transition probabilities are set so that the steady state probabilities are 0.8 and 0.2 respectively for the two states, to give a mean utilization of 0.3. Thus the system utilization is again 0.9 as before.

In the non-reservation version of this experiment *MM-Bursty-Nores*, the three activities were run together as a single group with common reservation of 100%. In the group reservation version of the same, *MM-Bursty-GrpRes*, the two sets have separate reservations. The reservation for the MM workload are varied from 0.3 to 0.7 as before.

It is expected that the group reservation results in lower deadline miss ratio for the audio and video activities, than in the non reservation case.

Table 5.1 summarizes the experiments and their setup. The description gives the system model, reservation specification, type of workload and performance measures of interest.

5.3 Results and Discussions

In this section, the results obtained are summarized and the observed results are discussed. It is important to note that some of the experiments involved deterministic workloads. For example, the *Rate-Greed-Sym* experiment had Greedy jobs that arrived at the start of the simulation and stayed till the end. Hence, for such experiments, the results show no confidence intervals, as there is no randomness in the system. On the other hand, in the Delay-Period-Greed experiment, even though the task periods and computation times are deterministic for each experiment, since the task periods are drawn from a randomly distributed range of values, the results for these experiments are presented with confidence intervals.

Experiment	Description
<i>Rate-Greed-Sym</i>	<i>Workload:</i> Three Greedy tasks competing <i>Reserved Rates:</i> 25% per 100ms each, <i>Measure:</i> Mean rate
<i>Rate-Greed-Asym</i>	<i>Workload:</i> Three Greedy tasks competing <i>Reserved Rates:</i> 15%, 35%, 25% per 100ms, <i>Measure:</i> Mean rate
<i>Rate-Period-Sym</i>	<i>Workload:</i> Three Periodic tasks competing <i>Reserved Rates:</i> 25% per 100ms each, <i>Measure of Interest:</i> mean rate
<i>Rate-Period-Asym</i>	<i>Workload:</i> Three Periodic tasks competing <i>Reserved Rates:</i> 15%, 35%, 25% per 100ms, <i>Measure:</i> Mean rate
<i>Delay-Period-Greed</i>	<i>Workload:</i> Grp 1 has 3 periodic tasks, Grp 2 is Greedy <i>Reserved Rates:</i> Varied from 0.1 to 0.9, <i>Measures:</i> Mean rate, Deadline miss ratio, Preemption count
<i>MM-Greed-Compete</i>	<i>workload:</i> Two Groups, Group 1 with Periodic CBR and periodic bursty tasks, Grp 2 is greedy <i>Reserved Rates:</i> varied from 0.3 to 0.7, <i>Measures:</i> Mean Rate, Deadline miss ratio
<i>MM-Greed-Isol-1</i>	<i>Workload:</i> Single group in isolation, with periodic CBR, periodic bursty tasks <i>rate:</i> varied from 0.3 to 0.7, <i>Measure:</i> Deadline miss ratio
<i>MM-Greed-Isol-2</i>	<i>Workload:</i> Single group in isolation, with periodic CBR, periodic bursty tasks <i>rate:</i> varied from $\frac{0.3}{0.9}$ to $\frac{0.7}{0.9}$, <i>Measure:</i> Deadline miss ratio
<i>MM-Bursty-Nores</i>	<i>Workload:</i> Single group, with periodic-CBR and Periodic-bursty and MMDP tasks <i>Rate:</i> Varied from 0.3 to 0.7, <i>Measure:</i> Deadline miss ratio
<i>MM-Bursty-Grpres</i>	<i>workload:</i> Two groups, Group 1 periodic-CBR and Periodic-bursty, Group 2 MMDP <i>Reserved Rates:</i> Varied from 0.3 to 0.7, <i>Measure:</i> Deadline miss ratio

Table 5.1: Summary of experiments

Scheduler	Mean Rate(Grp1)	Mean Rate(Grp2)	Mean Rate(Grp3)
GRP-EDF	0.500	0.250	0.250
GRP-RR	0.500	0.250	0.250
GRP-RC	0.333	0.333	0.333

Table 5.2: Mean rate achieved by three greedy groups in *Rate-Greed-Sym* experiments. The reserved rates for the three groups were 0.25 each. The results had a confidence interval width of less than 0.001 for a 95% confidence level.

5.3.1 Rate Experiments

The first result that is the basis for understanding the workings of the schedulers is for *Rate-Greed-Sym* and *Rate-Greed-Asym* experiments. The mean rates achieved by the three competing Greedy workloads are summarized in Table 5.2. Note that the three Greedy workloads were competing with equal reservation, but, GRP-RR and GRP-EDF both allocated 50% of the total capacity to Group 1 as against its reserved capacity of 25%. In contrast, GRP-RC allocated equally 33% to each group. The reason for this is that GRP-RR allocates the spare capacity in each scheduling round to the groups in the service order and GRP-EDF gives this to the group with the earliest deadline job. However, since the Greedy jobs arrived at the start and all had infinite deadlines, the GRP-RR and GRP-EDF schedulers are identical in the schedules generated. GRP-RC on the other hand computes priorities based on actual consumption in each cycle and hence rotates the allocation of excess capacity evenly.

The main conclusion of this experiment is that all the three schedulers allocate the minimum rate equal to the reserved capacity. They differ in the way the excess capacity is distributed.

Now consider the *Rate-Greed-Asym* experiment where the reserved shares for the three groups were respectively 0.15, 0.35, 0.25 over a period of 100ms. The mean rates achieved by the three groups under the three schedulers are shown in Table 5.3. It can be seen that again, GRP-RR and GRP-EDF had identical results in terms of mean rate achieved with all the additional 25% of the capacity going to the

Scheduler	Mean Rate(Grp1)	Mean Rate(Grp2)	Mean Rate(Grp3)
GRP-EDF	0.400	0.350	0.250
GRP-RR	0.400	0.350	0.250
GRP-RC	0.200	0.467	0.333

Table 5.3: Mean rates achieved by three greedy groups in *Rate-Greed-Asym* experiments. The reserved rates were respectively, 0.15, 0.35 and 0.25. The results had a confidence interval width of less than 0.001 for a 95% confidence level.

first group. GRP-RC on the other hand distributes the same in proportion to the reservations made.

The results for the *Rate-Period-Sym* are worth noting. Although the experiment was similar to *Rate-Greed-Sym* experiment in that the workloads and reservations are symmetric, the results obtained are different. Mainly, the GRP-RR and GRP-EDF algorithms now distribute the surplus capacity equally, and each group achieved a mean rate of 0.33. The mean rates achieved are shown in Table 5.4.

For the *Rate-Period-Asym* experiments, where the reservations were 0.15, 0.35, 0.25 respectively and the tasks were periodic, GRP-RR and GRP-EDF allocated the rates of 0.23, 0.43, 0.33. This indicates that the surplus of 0.25 was equally divided among the three groups irrespective of the reservations made. However, GRP-RC still allocated the surplus capacity in proportion to the reserved rates, thereby, achieving, 0.20, 0.47, 0.33. Note that these results were consistent in spite of varying the relative periods of the three tasks. The mean rates achieved are shown in Table 5.5.

The main conclusions of the rate experiments are that,

- all the three schedulers were able to guarantee the reserved rates of execution to each group
- the schedulers differed in the way the surplus capacity was distributed. GRP-RC distributes it proportionate to the reservations made; GRP-RR and GRP-EDF distribute equally.
- These results conform to the expected behavior of the schedulers.

Scheduler	Mean Rate(Grp1)	Mean Rate(Grp2)	Mean Rate(Grp3)
GRP-EDF	0.33	0.33	0.33
GRP-RR	0.33	0.33	0.33
GRP-RC	0.33	0.33	0.33

Table 5.4: Mean rates achieved by three greedy groups in *Rate-Period-Sym* experiment. The reserved rates were respectively, 0.25 for each group and an unreserved capacity of 25%. The results had a confidence interval width of less than 0.001 for a 95% confidence level.

Scheduler	Mean Rate(Grp1)	Mean Rate(Grp2)	Mean Rate(Grp3)
GRP-EDF	0.23	0.43	0.33
GRP-RR	0.23	0.43	0.33
GRP-RC	0.200	0.467	0.333

Table 5.5: Mean rates achieved by three periodic groups in *Rate-Period-Asym* experiments. The reserved rates were respectively, 0.15, 0.35 and 0.25. The results had a confidence interval width of less than 0.001 for a 95% confidence level.

5.3.2 Delay Experiments

The main objective of these experiments was to evaluate the delay performance of the three schedulers. Two particular features are of interest. The first is that GRP-RR is expected to be sensitive to the choice of scheduler resolution parameter R . The second is the performance of GRP-RC which has no notion of deadlines and which assigns priority based only on the current rate of execution and the reserved rate.

The *mean rates* achieved for the above workloads are now presented. Table 5.6 presents the mean rate achieved by Group 1 when its reserved share was set to 0.1.

Observe that the mean rate achieved under GRP-RR rapidly falls below the reserved rate of 0.1 for group 1, as the scheduler resolution increases. This is because of the fact that the non-preemption interval size increases causing rate errors. GRP-RC and GRP-EDF achieve the mean rates very well.

The *deadline miss ratio* results for these experiments are shown in the Fig-

Scheduler Resolution	Mean Rate		
	GRP-EDF	GRP-RR	GRP-RC
1.0	0.100	0.100	0.099
40.0	0.100	0.100	0.099
80.0	0.100	0.099	0.099
120.0	0.099	0.093	0.098
160.0	0.099	0.087	0.098
200.0	0.099	0.082	0.098
240.0	0.099	0.075	0.097
280.0	0.099	0.071	0.097
320.0	0.099	0.067	0.097
360.0	0.100	0.064	0.096
400.0	0.100	0.060	0.097

Table 5.6: Mean rate achieved by Group 1 in *Delay-Period-Greed* experiments. Group 1 had reserved rate of 0.1. Group 1 is rate compliant and Group 2 is greedy. The results had a confidence interval width of less than 0.01 at 95% confidence level.

ures 5.2 - 5.4 which shows the plot of the deadline miss ratio against the scheduler resolution for three different values of reservations, 10%, 20% and 30%. GRP-RR clearly seems to be sensitive to the choice of R . The larger the R value, larger is its deadline loss ratio. This is most pronounced at lower reservation levels. This is only to be expected. In analyzing the behavior of GRP-RR the sensitivity of the worst-case delay for GRP-RR to the choice of R was pointed out. This arises out of the fact that a larger value of R produces longer intervals of non-preemptibility, in accordance with the logic of the GRP-RR scheduler. In addition, when the reserved share of the group is small, the window of time in which this group can immediately have access to the CPU in each round diminishes further. Thus a job arriving after this window will be likely to miss its deadline.

GRP-EDF has the best deadline loss performance. This is also to be expected given that it is after all, a deadline-based scheduler. However, two observations are surprising. The first is that GRP-RC policy's performance is comparable to that of GRP-EDF. However, as R increases, the deadlines missed by GRP-RC increases. The second is that for some range of values of R , GRP-EDF has more deadline

Figure 5.2: Fraction of deadlines missed by Group 1 activities with Group 1 reserved share of 10%.

Figure 5.3: Fraction of deadlines missed by Group 1 activities with Group 1 reserved share of 20%.

Figure 5.4: Fraction of deadlines missed by Group 1 activities with Group 1 reserved share of 30%.

loss than GRP-RC but otherwise it is insensitive to R . This can be seen for R between 125 and 150. This can possibly be explained as follows: Note that the task periods are chosen from three ranges of values, each uniformly distributed. The first activity has its period drawn from the range 75, 100, the second from 150, 200 and the third from 300, 400. As R value crosses the range 75 – 100, there is an increase in the deadline miss rate before stabilizing. This increase occurs because, more tasks will have periods lower than the R value and hence, have higher probability of missing deadlines. A similar increase does not arise when R value crosses the range 300, 400 because, these tasks with their relatively longer deadlines can be scheduled, especially by the slack that arises due to dropping of jobs in the other two period ranges.

Figures 5.5 to 5.7 show the results for the same experiments as a function of the group 1 reserved share for the scheduler resolution values (R) of 100ms, 300ms and 460ms. Again, these plots indicate that GRP-RR has large deadline miss ratios at lower reservation levels, particularly at larger scheduler resolution values. GRP-EDF

Figure 5.5: Fraction of deadlines missed by Group 1 activities when scheduler resolution was 100ms.

has consistently lowest deadline miss ratios for all reservation levels and is insensitive to the scheduler resolution parameter values. The reason for this is that GRP-EDF is highly preemptive and is not affected much by the scheduler resolution. GRP-RC has very good deadline miss ratios but, shows some variability to the reserved rate and small sensitivity to scheduler resolution parameter values.

Finally, the *preemption counts* for the three schedulers in the *Delay-Period-Greed* experiments are presented in Table 5.7. The table shows that for GRP-RR and GRP-EDF schedulers the preemption counts fall rapidly as scheduler resolution increases before saturating. For GRP-RR it remains nearly the same. This is because, unlike in GRP-RR and GRP-EDF, in GRP-RC, the scheduler resolution is used to compute the *RC-value* and it not necessarily causes a preemption if the *Rc-values* are same which is likely when the rate specification period is as high as 200ms in these experiments. GRP-EDF has the highest preemption count as would be expected because of its highly preemptible nature. GRP-RR has lower preemption count than GRP-RC, for $R > 100ms$.

Figure 5.6: Fraction of deadlines missed by Group 1 activities when scheduler resolution was 300ms.

Figure 5.7: Fraction of deadlines missed by Group 1 activities when scheduler resolution was 460ms.

Scheduler Resolution	GRP-EDF		GRP-RR		GRP-RC	
	Mean	Conf.	Mean	Conf.	Mean	Conf.
1.0	400207.40	14.69	400201.00	17.90	3672.65	70.02
20.0	20268.20	27.06	20197.13	5.47	3678.91	49.78
40.0	10430.40	42.70	10173.33	6.54	3676.31	38.90
60.0	7340.00	66.90	6789.58	9.04	3667.33	47.70
80.0	5935.40	88.57	5086.45	13.30	3667.99	43.16
100.0	5890.63	74.57	3946.05	19.05	3668.47	47.09
120.0	5737.05	37.51	3015.76	34.69	3655.91	55.43
140.0	5418.71	69.59	2543.55	56.09	3650.98	54.40
160.0	5284.97	65.76	2138.30	11.45	3639.72	65.00
180.0	5164.12	50.46	1928.71	21.90	3642.85	74.53
200.0	5033.96	42.89	1702.50	100.28	3603.60	48.01

Table 5.7: Preemption counts for the three schedulers in the *delay-period-greed experiment*.

The number of preemptions for GRP-RC are dramatically low in comparison with GRP-EDF. Interestingly, it does not vary significantly with R . This is possible because, after computing the finish time for each group, GRP-RC sets the *RC-value* to the next reservation period in which this finish time falls. The granularity of the reservation period parameter t is more influential in determining the *RC-val* than the scheduler resolution.

5.3.3 Mixed Load Experiments

In these experiments the scheduling schemes are evaluated simultaneously for the rate and deadline performance, for a mixed set of workloads. Consider first the results for *MM-Greed-Compete* experiment where the multimedia session with CBR audio and VBR video activities competes with a Greedy workload. We are interested in the deadlines missed by audio and video, the rate achieved by Greedy as well as the number of preemptions for each of the three schedulers.

The deadline miss ratio for session 1 for the three schedulers are shown in Table 5.8 and the mean rate achieved by the two groups are shown in Table 5.9, for

Grp 1 Share	Deadline Miss Ratio		
	GRP-EDF	GRP-RR	GRP-RC
0.3	0.296	0.394	0.481
0.35	0.258	0.391	0.388
0.4	0.218	0.247	0.295
0.45	0.180	0.197	0.201
0.5	0.140	0.162	0.101
0.55	0.090	0.128	0.002
0.6	0.054	0.098	0.001
0.65	0.019	0.069	0.001
0.7	0.003	0.037	0.000

Table 5.8: Deadline miss ratios suffered by three groups in *MM-Greed-Compete* experiment. The Confidence interval widths were less than 0.01 for a 95% confidence level.

various reservation levels of Group 1. Note that at reservation levels below 0.5 GRP-EDF and GRP-RR have lower deadline misses than GRP-RC. It should be noted that the combined utilization of Group 1 is at most 0.60, for the workload model chosen.

The deadline miss ratios suffered by the MM group under isolated processor with three different reservations are shown in Table 5.10.

Finally, the results for the mix of multimedia workload in competition with bursty workload as described in the experiments *MM-Burst-Nores* and *Mix-MM-Burst-Grpres* are presented in Table 5.11. When the multimedia workload was run with no reservation along with the bursty workload, the mean deadline miss ratio for the multimedia activities was 0.232 with a confidence interval width of 0.008. Compare this with the deadline miss ratio for the multimedia workloads under each of the group schedulers for various levels of reservations for the multimedia group shown in Table 5.11. Note that the mean utilization required for the group is 0.6. For all reservation levels from 0.3 to 0.7, the deadline miss ratio for group 1 were significantly lower than 0.232. However, for the GRP-RC scheduler, when the reservation levels were less than or equal 0.5, the group suffered worse deadline miss

Group 1 Share	Mean Rate for Group 1		
	GRP-EDF	GRP-RR	GRP-RC
0.3	0.400	0.300	0.332
0.35	0.447	0.350	0.388
0.4	0.488	0.399	0.444
0.45	0.524	0.480	0.499
0.5	0.553	0.510	0.555
0.55	0.574	0.538	0.599
0.6	0.590	0.561	0.599
0.65	0.596	0.578	0.599
0.7	0.599	0.591	0.600

Table 5.9: Mean rate achieved by Group 1 for the *MM-Greed-Comp* experiment. The confidence interval widths were less than 0.01 for a 95% confidence level. The scheduler resolution R was 10ms.

MM-Greed-Isol1		MM-Greed-Isol-2	
Share	Deadline Miss Ratio	Share	Deadline Miss Ratio
0.3	0.424	0.33	0.391
0.35	0.387	0.39	0.359
0.4	0.324	0.44	0.290
0.45	0.283	0.50	0.242
0.5	0.241	0.56	0.177
0.55	0.184	0.61	0.147
0.6	0.148	0.67	0.094
0.65	0.108	0.72	0.052
0.7	0.071	0.78	0.011

Table 5.10: Deadline miss ratios for Group 1 under isolated processor with the partition sizes as per experiments *MM-Greed-Isol*. The Confidence interval widths were less than 0.01 for a 95% confidence level.

ratios than with no reservation. This is interesting but not surprising, given that GRP-RC is biased towards groups with higher reserved shares.

5.3.4 Summary of Group Scheduler Performance

The evaluation experiments described above indicate that the group schedulers achieve the goals of isolation and fairness with varying degrees of success under different conditions. Under intense competition from greedy and persistent workloads, the three schedulers achieve the reserved rates equally well. In distributing any spare capacity the GRP-RC scheduler was distinct from the GRP-RR and GRP-EDF schedulers in that it distributed it proportional to the reservations made where as the latter two schedulers distributed it all to the next group in the scheduling order at a given instant and equally to all groups on an average. The rates achieved were insensitive to the relative task periods.

When a compliant group of workloads competed with a greedy persistent or bursty competition, the GRP-EDF and GRP-RC schedulers achieved the mean rates effectively. The rate achieved under the GRP-RR scheduler however, was sensitive to the choice of the scheduler resolution parameter. Similarly, the deadline miss ratio under GRP-RR was also found to be sensitive. At coarser resolutions, several deadlines were missed under GRP-RR. GRP-RC, although somewhat sensitive to the value of the scheduler resolution parameter, had low deadline miss ratios. However, as seen from the delay and mixed load experiments, GRP-RC was biased towards the group with higher reservation levels both in terms of rate achieved and deadlines missed. GRP-EDF had the best deadline miss ratio and mean rate under all scheduler resolutions. Due to the highly preemptive nature of GRP-EDF, it was insensitive to variations in the scheduler resolution. The schedulers also differed in the overheads as measured in terms of preemptions. GRP-EDF had the highest number of preemptions especially at finer scheduler resolutions. GRP-RR had lowest overheads at coarser resolutions. GRP-RC had low overheads at all resolutions.

From the above experiments, it appears that the appropriate choice of the scheduler will depend on several factors. From performance point of view GRP-EDF

Scheduler Resolution	GRP-EDF	GRP-RR	GRP-RC
0.3	0.1878	0.1645	0.4034
0.35	0.1702	0.1538	0.4047
0.4	0.1544	0.1325	0.4017
0.45	0.1314	0.1124	0.3448
0.5	0.1091	0.0832	0.2530
0.55	0.0784	0.0630	0.1397
0.6	0.0565	0.0428	0.0080
0.65	0.0345	0.0223	0.0004
0.7	0.0153	0.0071	0.0003

Table 5.11: Deadline miss ratios for MM group in competition with bursty workload *MM-Burst-Grpres*. The Confidence interval widths were less than 0.01 for a 95% confidence level.

scheduler appears to be the best. Even though it has higher overheads, by choosing coarser scheduler resolutions, this can be lowered significantly. GRP-RC scheduler has second best performance and lower overheads. It is also simpler to implement than GRP-EDF. GRP-RR scheduler, is the simplest among the three but has some drawbacks in terms of performance. It can still be used with finer scheduler resolutions if overheads is not a problem or at coarser resolutions if accuracy of isolation is not important.

5.4 Collaborative Sharing Experiments

In Chapter 3 the over all strategy of resource allocation in AURA was described as depicted in Figure 3.8. According to this strategy, the application's *Effective Required Bandwidth* is estimated. In addition a QOS model is derived which maps the focus level of an activity to a target QOS level based on a utility function. The goal of the collaborative sharing then, is to attain this target QOS so that the total utility value of the application session is maximized under the available resources.

In this section the reactive QOS control scheme is applied to a synergistic mul-

timedia session and its effectiveness in achieving the target QOS over the session duration as the focus state evolves with time is investigated. In particular, this reactive QOS control is studied in the presence of different load levels in the system and under reservation-less, reservation on a per-activity basis and with group reservation.

Although proactive and reactive QOS control strategies were described, only the reactive QOS control is investigated. The proactive QOS control is sufficient if the effective required bandwidth is exactly known and reservations are made accordingly. However, it does not have the flexibility to deal with inadequate resource availability or to exploit any spare capacity that may be available at a given time in the system. In contrast, the reactive QOS control is able to explore the QOS states under different load conditions by adaptively enhancing or decreasing the QOS of the activities. This property is important, especially when characterizing the effective required bandwidth of a multimedia session. By evaluating the target QOS achieved by the reactive collaborative sharing session and the total utility value to the session, under different capacity levels, the effective required bandwidth can be determined. Further, for a set of available capacity levels, the possible set of configurations of target QOS levels can also be explored and those with maximal utility value can be determined. Therefore, the evaluation of the effectiveness of the reactive collaborative sharing scheme is important.

5.4.1 Workload and QOS Model

In the following experiments, the system is assumed to have two concurrent groups which are competing with each other for the CPU. One of them is designated as a “foreground” group and the other is called a “background” group. The foreground group is principally of interest and the background group is representative of all sources of competition in the workstation. Such a workload mix is commonplace in multimedia workstation environments. For example, several computationally intensive workloads could be launched through a local area network on the given workstation. A single multimedia session comprising of different media types *and*

multimedia applications is present on this workstation, and usually is launched from the console.

There are several key points to be observed in this scenario. First of all, even though remote users and computations are termed “background” they do not have any lower priority or importance than the “foreground” computations. In fact, each remote computation also have quality of service requirements which are expressed in terms of a minimum required rate of execution. This rate of execution in contrast to ordinary time-shared systems, should be *guaranteed* as opposed to merely being provided on a best-effort basis. Thus, not only should the “foreground” session’s QOS requirements expressed in terms of frame rates etc., be met but also the “background” session’s rate of execution must be guaranteed. It should also be noted that not only can the console-based applications but also, remote computations can also be multimedia sessions. Examples of such scenarios were presented earlier in Chapter 1 when the scope of this research was laid out.

In the following experiments, the influence of the background load on the foreground, in terms of the QOS achieved by the foreground session only is stressed. This is because, what is being evaluated here is the performance of the collaborative sharing algorithm. The isolation performance of the group schedulers studied previously adequately assures that rate guarantees can be made for all the independent sessions. There fore, a second key point to be noted here is that only the foreground session consists of the collaborative sharing mechanisms. Although experiments were performed using multiple sessions with collaborative sharing within each were performed, in the reported results are for a single session with collaboration in the presence of competing workloads which are bursty and possibly greedy.

The background group is modeled as a two state Markov-modulated Poisson Process, in which, the state residence times for both the states are exponentially distributed with mean 50.0s. In the lower utilization state, the inter arrival time of jobs is 0.37037s and the service time is 0.0167s. In the higher utilization state the values are 0.24691s and 0.22223s respectively. The choice of these values were made in order to control the load levels. The experiments were repeated with other

combination of values but with no significant variations in the observed results.

The foreground group consists of three identical activities. These activities represent three constant bit-rate (CBR) video streams. For example, the session could represent a simple video-composition application where video streams from three different remote servers are gathered and a composite video is presented at the workstation. The composition process is such that at any given time one of the streams is at a highest focus level, the other at a moderate focus level and the third at the lowest focus level. This assignment changes with time and is captured by a finite state machine whose states represent a vector of focus levels. The session is modeled by a semi-Markov chain with general residence time distributions for the states. In the reported experiment, the residence times are deterministic with value 100s. A deterministic value was chosen because, we are interested in showing the trace of execution and the QOS achieved as a function of time.

The assumption that the video streams are CBR requires some justification. Compressed video is known to have variable bit rate, and can exhibit long range dependencies and self-similarity. These variations in the required demand are inherent to the scene complexity of the video sources. Although in the case of VBR video the variations are long term, they are still random and not easily associated with application or user state. Techniques such as imprecise computations [6] and hierarchical coding can be used to adaptively control the video quality, to achieve a constant bit rate. This technique has been applied to video compression [27]. In the AURA approach, a much higher-level phenomenon arising out of the user focus and application state is of primary concern. It is therefore assumed that such mechanisms exist which smooth out the random fluctuations in demand. The goal of collaborative QOS control is to deal with long range load fluctuations arising out of user and application state changes and hence they can be studied independently of the purely random fluctuations.

The QOS parameter used in the experiments is the video frame rate whose values depend on the focus level on the video stream. A set of five frame rates, [5, 10, 15, 20, 25] frames-per-second are allowed for each video activity. When any

given activity is at a high focus level, the optimal QOS level with respect to the QOS model's utility value function is assumed to be 20fps . The corresponding values for medium focus level and lowest focus levels are 15fps and 10fps respectively. These values are referred to as **Target QOS** for the given focus level. When an activity is at a given focus level, achieving less than the target QOS will be treated as QOS error. Video frames of the activities are assumed to have constant service times (in accordance with CBR video assumption). The per-frame service time is chosen to be 0.0167s , set so as to offer the desired load intensities. The experiment was also repeated with other values for the service times and no significant difference in the results were observed.

5.4.2 Observations and Conclusions

The reactive QOS control algorithm described in Chapter 4 is studied in three different environments – one in which no reservation mechanisms exist, one in which reservations are made for individual activities independently and finally in a group reservation-based environment. The collaborative load shedding in a reservation-less environment was studied by Compton and Tennenhouse [8]. Independent reservation-based QOS control but with no explicit notion of focus state has been proposed by Mercer and Rajkumar [45]. The objective in the experiments described below is more towards showing that effective isolation is important for effective collaboration. This can be seen by comparing the performance under reservation-less system and that under either the individual or group reservation system.

The collaboration experiments were performed for various reservation levels (for group reservation and independent reservation environments) and workload parameters for each of the no-reservation, independent-reservation and group reservation schemes. The frame rates achieved is compared against the target frame rate for a given focus level. In the no-reservation environment, all the three video activities as well as the background activity were treated as a single group with the entire system capacity being available to whichever activities can offer load. Only the three video activities of the foreground session are under collaborative sharing control.

Thus, when system overload is detected and the target QOS levels are not achieved, only these activities will be downgraded. Similarly, they may be upgraded when the system load is low. The background activity is not under any control and is offering load according to its state without any concern for whether the system is overloaded or whether its tasks are being dropped. In the independent reservation environment, each activity makes a reservation at the start of the session to meet its target QOS level as determined by the focus level of that activity at the start of the session. Subsequently, when the focus level shifts, the activities do not revise their reservations but simply attempt to share their reserved capacities together by downgrading and upgrading themselves, in accordance with the reactive collaborative sharing policy. In the group reservation environment, the three video activities together make a combined reservation for supporting their target QOS levels at the start of the session. Subsequently, they share the reserved capacity according to the reactive collaborative sharing policy as above. Again, the combined reserve for the group is not revised during the session. In the independent reservation and group reservation environments, the GRP-EDF algorithm was used to provide isolation. The use of other schedulers such as GRP-RR and GRP-RC is also acceptable, provided the scheduler parameters were chosen appropriately.

The over all performance objectives of the control strategy are :

- If the available resources are adequate, all the activities must attain their target frame rates.
- If surplus capacity is available, then the excess capacity should be used to enhance the QOS of the highest focus activity first.
- If the available resource is inadequate to meet the target frame rates of all the activities, then the available resources should be focused to meet the target frame rate of the highest focus activities.

How well these objectives are met under different system load conditions is being examined in these experiments for the three environments. Two sets of plots are

shown for each of group reservation, independent reservation and reservation-less environments. The nature of the competing load and the actual utilizations achieved by the foreground and background sessions is shown in the first set of plots. This utilization is measured over an interval of length 8s and is shown against time in the x-axis. The QOS achieved, expressed as frame rate, is also measured over an interval of length 8s and is plotted against time in the x-axis. In all the cases, the session focus dynamics was common. Initially, the the target QOS vector was [20, 15, 10] frames per second. The focus shifted every 100s and cycled through the sequence of target QOS vectors [20, 15, 10], [15, 10, 20], [10, 20, 15], [20, 15, 10] and so on.

Figure 5.8 shows the background load intensity and the actual background and foreground utilizations against time for the first 500s of the session. Note that for the first 150s the background load is very low and then has a short period of high load intensity between 150s and 180s. After 200 the background load varies between 0.5 and well beyond 1.0. At 320 there is a sharp spike following which the background load stays near 1.0 thus driving the system to persistent saturation. Observe now the actual utilizations by foreground and background sessions. The background session is able to utilize up to 80% of the CPU after time 320s. The foreground utilization which was initially between 0.75 and 1.0 stays as low as 0.125 at time 270 and after 360. This indicates that in the absence of reservation, a competing workload is able to saturate the system at will and consume a very high percentage of the CPU. The resultant loss of quality for the foreground system can be seen in Figure 5.9.

Initially, each of the activities achieve its target frame rate. Then, the highest focus activity is upgraded to 25 and then 30 frames per second and oscillates there until the focus change event occurs at 100. This shift in focus is successfully followed but the spike at 180 downgrades all the three activities. Around 250 when the background load peaks to 100%, the three activities get downgraded until 270. When the background is persistently saturating the system, the frame rates stay low.

In the independent reservation scheme, the load intensity of which is shown in Figure 5.10, the foreground utilization is preserved until time 220s after which it drops to about 50% due to the spike in the background load. When the background

Figure 5.8: Background load intensity, and foreground and background Utilizations in reservation-less collaboration.

Figure 5.9: Frame rates achieved by the three foreground activities under collaboration in a *reservation-less* system is shown as a function of time. The focus shift every 100 epochs. The corresponding target QOS vectors for the three groups cycles through the set of values: $[20, 15, 10]$, $[15, 10, 20]$ and $[10, 20, 15]$ respectively.

load stays persistently above 100%, the foreground is able to preserve up to 50% utilization due to the reservations made. However, a part of the reserved capacity is unused by the activity which was initially at the highest focus level and subsequently was downgraded because of focus shift. This unused capacity which was intended to be shifted to the current highest focus activity of the foreground session is being shared by the background as well. This is because, in the independent reservation scheme, any unused capacity in the system is shared among all competing workloads. The effect of this is seen in the frame rates achieved by the foreground activities as seen in Figure 5.11. Clearly, the frame rates which were satisfying the target QOS requirements initially, have been downgraded to less than the target later. It should be noted that if the independent reservation scheme was allowed to revise the reservations dynamically as focus shifts, this problem may not arise. However, there are several reasons why dynamic reservations are not practical. First of all, reservation operations may have overheads especially when optimization decisions are made. This is unattractive when focus shifts are dynamic. Secondly, when several reservations requests are pending, it is not immediate what the policy should be for handling the requests for revising reservations.

Contrast this with the performance of the group reservation case as seen in both utilization and QOS performance. Figure 5.12 shows that the foreground utilization stays steadily near the reserved level of 0.752 except for short periods when focus shift adaptation occurs. When the background load is low, the foreground gets as high as 0.855 and when background load is high, it still gets its reserved share. The effect of this on the frame rates achieved can be seen in Figure 5.13. The highest focus activity gets enhanced performance when the system load is not near saturation. Focus shifts are quickly followed. The activities always achieve their target QOS.

It can be concluded that collaborative sharing using focus levels is effective in achieving the QOS objectives provided adequate isolation from competition exists. This isolation, achieved through reservation and scheduling mechanisms is possible either on a per-activity basis or on a group basis. The latter form isolation is more

Figure 5.10: Background load intensity, and foreground and background utilizations in independent reservation collaboration.

Figure 5.11: Frame rates achieved by the three foreground activities under collaboration in an *independent reservation* system is shown as a function of time. The focus shift every 100s. The corresponding target QOS vectors for the three groups cycles through the set of values: $[20, 15, 10]$, $[15, 10, 20]$ and $[10, 20, 15]$ respectively.

Figure 5.12: Background load intensity and foreground and background utilizations in Group reservation collaboration.

Figure 5.13: Frame rates achieved by foreground session activities under *Group reservation*. The focus shifts every 100s. The corresponding target QOS vectors for the three groups cycles through the set of values: $[20, 15, 10]$, $[15, 10, 20]$ and $[10, 20, 15]$ respectively.

attractive when reservations can not be frequently revised dynamically during the progress of the session.

Chapter 6

Conclusions

This dissertation addressed the problem of allocating workstation CPU to multimedia applications. The summary of contributions made in this research together with main conclusions are presented here. Finally, further directions for continuing this research are outlined.

6.1 Contributions and Conclusions

A new two-level approach called Application and User-oriented Resource Allocation (AURA) was proposed and investigated. This approach was found to meet the objectives of fairness, flexibility, effectiveness and efficiency.

Several multimedia application scenarios were analyzed and it was found that application structural and temporal relationships as well as dynamic behavior of the session can be exploited in efficiently allocating resources. A phenomenon called *demand synergy* which reduces the resource demands for a given QOS level was identified. A QOS model which uses information about the focus state of a multimedia session to direct resources was proposed. Based on the synergy and QOS model, a two-level allocation strategy was formulated.

The key aspect of the AURA allocation strategy, was to distinguish between *competing* and *cooperating* workloads, and isolate competing workloads while allowing cooperating workloads to share resources. This distinction allowed AURA to

exploit structural and behavioral relationships in multimedia applications to focus available resources effectively to those activities which offer largest utility.

Fundamental to the success of the AURA allocation strategy was the *Group Reservation and Scheduling* mechanisms. This provided both a simple model of reservation and an effective means for isolation. Three scheduling algorithms, viz., GRP-RR, GRP-EDF and GRP-RC were proposed in Chapter 4 and investigated in Chapter 5. All the three were found to be adequate for providing isolation with proper choice of the scheduler parameter. This was evaluated by comparing the rate of execution achieved against the reservation made under various competing workloads typical in multimedia workstation environments. The algorithms were also found to be allocating any surplus capacity in a meaningful manner. GRP-RR and GRP-EDF allocated the surplus equally to the competing workloads and GRP-RC allocated it in proportion to the reservations made by them, as was shown in the *Rate Experiments* in Chapter 5. However, the three algorithms differed in their delay performance and overheads.

Both rate and delay performance of GRP-RR was found to be sensitive to the choice of the scheduler parameter, called the schedule round length R . As seen from Table 5.6 in the *Delay-Period-Greed* experiments, GRP-RR failed to allocate the required rate for large values of R . From the plots for the same experiments, it can be seen that the delay performance of GRP-RR was also poor at higher values of R especially, when the reservation is small. When R is greater than task periods, it resulted in deadline misses.

In contrast, both GRP-EDF and GRP-RC were found to be effective in providing rate guarantees as well as meeting deadlines, even for large R values. GRP-EDF had slightly better delay performance and rate performance than GRP-RC. However, it had very high preemption rate compared to GRP-RC, especially for small values of the scheduler round length R .

It can be concluded that the choice of the scheduler depends on several factors, since all the three schedulers meet the isolation requirements. If simplicity is needed, GRP-RR is a good choice. With sufficiently small R values, it can meet the isolation

and delay requirements. GRP-EDF had very good isolation and delay properties but has high overheads. However, since it was fairly insensitive to R value, a larger R value can be chosen with a corresponding reduction of preemption overheads. Finally, GRP-RC had sufficient simplicity, isolation and delay properties to be a good choice for group scheduling.

Given rate guarantees for groups, collaborative sharing policies were made possible. A simple reactive QOS control strategy was proposed and demonstrated to be effective when adequate resource was reserved for the group. This strategy exploited the information about current focus level of activities in scaling QOS required so as to deal with resource non-availability. This strategy was shown to be successful in meeting the target QOS of the activities in a dynamic focus change scenario of a multimedia application. This strategy exploited any spare system capacity to further enhance the QOS provided. However, it was observed that this strategy failed to support the target QOS when isolation was ineffective. In particular, when the reactive QOS control was applied in a reservation-less system, when there is persistent overload due to competition, the target QOS were not achieved. When reservations were made on a per-activity basis but without allowing for dynamical renegotiation of reservations, target QOS were achieved so long as the focus levels did not change dynamically. When group reservation was used, under all circumstances the target QOS were achieved, in a dynamic focus situation. This indicates the need for effective isolation in order to provide effective QOS control.

The AURA approach requires that the application as a whole be characterized for the required effective bandwidth, taking into consideration any temporal and synergy relationships among the activities. It also requires that the current focus state of the session be available through observable events in the operating system interface or be provided explicitly. Many possible interaction points were suggested in Chapter 3 where such information are available. Moreover, it is believed that these requirements will be reasonable when applications are developed with the objective of exploiting the benefits of synergy and the two-level allocation.

6.2 Future Directions

The AURA approach has shown the benefits of explicitly dealing with application-dependent information. Based on the initial investigations carried out in this research, it can be concluded that there is value in explicitly characterizing the resource requirements of multimedia applications including any state and structural information. This initial work can be extended in a few immediate ways.

- **Extensions to Group Scheduling**

This research concerned itself mainly to the allocation of a single CPU as a resource in workstation environments. In this respect, scheduling algorithms for partitioning the capacity were studied. Several immediate extensions can be thought of, which include allocation of symmetric and asymmetric multiprocessors and allocation of buffers and file systems using the principle of synergy. Techniques developed for multiprocessor job allocation, such as Gang Scheduling can be investigated in the light of multimedia applications.

- **Synergy Characterization**

It was assumed in the AURA approach that the synergy content of an application session and the corresponding effective required bandwidth information is readily available for the group reservation and collaborative sharing strategies described here. It is therefore important to study existing multimedia applications and their usage and characterize the synergy in them. Characterization of multimedia sessions will have three aspects. First is to be able to characterize the required bandwidth of the application session taking into consideration the structural and temporal relationships. Each layer in the temporal hierarchy can be modeled as a Markovian process which modulates the resource requirements of the lower level entities and is in turn modulated by a controlling entity at a higher layer. This leads to a layered controlling Markov processes. Such models have been studied in the context of network bandwidth allocation. For example, Hui [26] describes a layered required bandwidth computation for call level, burst level and cell level QOS support. The

second aspect is that of characterizing synergy content of a session. Attention focus distribution can be modeled as a semi-Markov model and the model parameters can be estimated. For stored multimedia applications, the structure of the documents is available in some modeled format such as OCPN. These models can be enhanced with the conditional branch probabilities of a particular path of progression of the presentation. Such probabilities can be used to estimate the effective required bandwidth as well.

- **Synergy in Distributed Multimedia Sessions** Synergy principle exploits a form of mutual exclusiveness of resource requirements among a set of concurrent workload sources. This principle is general enough to be extended to distributed systems. Within the end system software layers, the operating system deals with allocation of resources in the end system and communication layer deals with network Quality of service negotiation. Given the end-to-end nature of QOS, it is useful to investigate how far synergy in one subsystem could influence the resource allocation in the other. For example, an end system with synergistic session, would appear to the network as a smooth source of network traffic. Therefore, if this synergy states can be propagated onto all the resource allocating points such as networks, servers and remote workstations, they all can potentially exploit synergy. A second aspect of distributed sessions is that the behavior of an intermediate component such as network switch or multiplexor can induce synergy in the end system. Network-based information caching, and agent-based services are being proposed. It would be interesting to study the effect of these services on the resource demands placed on workstation resources. For example, the notion of QOS filters have been proposed for dealing with multiple grades of service in multipeer communications where network and end systems have diverse capabilities. Such systems will induce different kinds of behavior and roles among the participants of a teleconferencing or CSCW application. The impact of these on the end system resource requirements is also interesting.

References

- [1] J. A. Adam. Interactive multimedia: Applications, implications. *IEEE Spectrum*, 30(3):24–31, March 1993.
- [2] D. P. Anderson. Metascheduling for continuous media. *ACM Transactions on Computer Systems*, 11(3):226–252, August 1993.
- [3] D. P. Anderson, S. Y. Tzou, R. Wahabe, R. Govindan, and M. Andrews. Support for continuous media in the DASH system. In *Proceedings of the Tenth International Conference on Distributed Computing Systems*. pages 54-61. June 1990.
- [4] C. M. Aras, J. E. Kurose, D. S. Reeves, and H. Shulzrinne. Real-time communication in packet switched networks. *Proceedings of the IEEE*, 92(1), January 1994.
- [5] S. Casner and S. Deering. First IETF internet audiocast. *ACM Computer Communication Review*, 22(3):92–97, July 1992.
- [6] J. Y. Chung, J. W. S. Liu, and K. J. Lin. Scheduling periodic jobs that allow imprecise results. *IEEE Transactions on Computer*, 39(9):1156–1174, September 1990.
- [7] D. D. Clark, S. Schenker, and L. Zhang. Supporting real-time applications in an integrated services packet network: Architecture and mechanism. In *ACM SIGCOMM*. pages 14–26. October 1992.

- [8] C. L. Compton and D. L. Tennenhouse. Collaborative load shedding for media-based applications. In *Proceedings of the International Conference on Multimedia Computing and Systems*, pages 496–501. Boston, MA, May 1994.
- [9] G. Coulson, G. S. Blair, J. B. Stefani, F. Horn, and L. Hazard. Supporting the real-time requirements of continuous media in open distributed processing. *Computer Networks and ISDN Systems*, 27:1231–1246, 1995.
- [10] G. Coulson, G.S.Blair, P.Robin, and D.Shepherd. Supporting continuous media applications in a microkernel environment. Technical Report MPG-94-16, Department of Computing, Lancaster University, Lancaster, UK, 1994.
- [11] R. L. Cruz. A calculus for network delay, Part II: Network analysis. *IEEE Transactions on Information Theory*, 37(1):132-141, January 1991.
- [12] L. Delgrossi, C. Halstrick, D. Hehmann, R. G. Herrtwich, O. Krone, J. Sandvoss, and C. Vogt. Media scaling for audiovisual communication with the heidelberg transport system. In *ACM Multimedia*, pages 99–104, 1993.
- [13] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queuing algorithm. In *ACM SIGCOMM*. pages 3–12, September 1989.
- [14] H. Eriksson. MBONE – The multicast backbone. In *Proceedings of the International Networking Conference (INET)*, San Francisco, CA USA, August 1991. Internet Society.
- [15] D. Ferrari. Client requirements for real-time communication services. *IEEE Communications Magazine*, 28(11):65–72, November 1990.
- [16] D. Ferrari and D. C. Verma. Scheme for real-time channel establishment in wide-area networks. *IEEE Journal on Selected Areas in Communications*, 8(3):68–379, April 1990.
- [17] D. Ferrari and D. C. Verma. Delay jitter control for real-time communications in a packet switching network. In *TRICOMM*, Chapel Hill N.C., April 1991.

- [18] E. A. Fox. Advances in interactive digital multimedia systems. *Computer*, 24(11):9–19, October 1991. (Special issue on multimedia information systems).
- [19] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Mancheck, and V. Sunderam. *PVM: Parallel Virtual Machine A User's Guide and Tutorial for Networked Parallel Computing*. Scientific and Engineering Computation. MIT Press, 1994.
- [20] J. F. Gibbon. *Real-Time Scheduling for Multimedia Services Using Network Delay Estimation*. PhD thesis, College of Engineering, Boston University, Boston, MA USA, 1994.
- [21] S. J. Golestani. A framing strategy for congestion management. *IEEE Journal on Selected Areas in Communications*, 9(7), September 1991.
- [22] R. Govindan and D. P. Anderson. Scheduling and IPC mechanisms for continuous media. In *ACM SIGOPS*. pages 68–80, October 1991.
- [23] A. Gupta and M. Moran. Channel groups: A unifying abstraction for specifying inter-stream relationships. Technical Report TR-93-015, University of California, Berkeley, March 1993.
- [24] R. G. Herrtwich. The role of performance, scheduling, and resource reservation in multimedia systems. In A. Karshmer and J. Nehmer, editors, *Operating Systems of the Nineties and Beyond*, volume V.563 of *Lecture Notes in Computer Science*, pages 279–284. Springer-Verlag, Berlin, 1991.
- [25] A. Hopper. PANDORA – an experimental system for multimedia applications. *ACM Operating Systems Review*, 24(3), April 1990.
- [26] J. Y. Hui. Layered required bandwidth for heterogeneous traffic. In *Proceedings of the IEEE Information Conference (INFOCOM)*. 1992.
- [27] E. A. Hyden. *Operating System Support for Quality of Service*. PhD thesis, Wolfson College, University of Cambridge, Cambridge UK, February 1994.

- [28] J. Hyman, A. A. Lazar, and G. Pacifici. Real-time scheduling with QOS constraints. *IEEE Journal on Selected Areas in Communications*, 9(7):1052–1063, September 1991.
- [29] K. Jeffay. On kernel support for real-time multimedia applications. In *3rd IEEE Workshop on Workstation Operating Systems*, pages 39–46, Key Biscane, FL, April 1992.
- [30] K. Jeffay and D. Bennet. A rate-based execution abstraction for multimedia computing. In *Network and Operating Systems Support for Digital Audio and Video*, Durham, NH, April 1995.
- [31] E. D. Jensen, C. D. Locke, and H. Tokuda. A time-driven scheduling model for real-time operating systems. In *IEEE Real-Time Systems Symposium*, Montreal, Canada, December 1985.
- [32] A. Jones and A. Hopper. Handling audio and video streams in a distributed environment. *ACM SIGOPS Operating Systems Review*, December 1993.
- [33] M. B. Jones. Adaptive real-time resource management supporting modular composition of digital multimedia services. In *Workshop on Network and Operating Systems Support for Digital Audio and Video*, Lancaster, UK, April 1993.
- [34] M. B. Jones. Support for user-centric modular real-time resource management in the rialto operating system. In *Workshop on Network and Operating Systems Support for Digital Audio and Video*, Durham, NH, April 1995.
- [35] C. R. Kalmanek, H. Kanakia, and S. Keshav. Rate-controlled servers for very high-speed networks. In *GLOBECOMM*. Pages 300.3.1–300.3.9. December 1990.
- [36] H. Kanakia, P. P. Misra, and A. Reibman. An adaptive congestion control scheme for real-time packet video transport. In *ACM SIGCOMM'93: Conference Proceedings*, pages 20–31. September 1993.

- [37] D. D. Kandlur, K. G. Shin, and D. Ferrari. Real time communication in multi-hop networks. In *IEEE Transactions on Parallel and Distributed Computing*, 5(10):1044–1056. October 1994.
- [38] S. Khanna, M. Sebrée, and J. Zolonowsky. Real-time scheduling in Sun OS 5.0. In *USENIX Winter Conference*, San Francisco. January 1992.
- [39] J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. In *IEEE Real-Time Systems Symposium*, pages 166–171. 1989.
- [40] S. Leutenegger and R. Nelson. Analysis of spatial and temporal scheduling policies for semi-static and dynamic multiprocessor environments. Technical Report IBM RC#75594, IBM, IBM T.J. Watson Research Center, NY., August 1991.
- [41] C. L. Liu and J. M. Leyland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM*, 20(1), January 1973.
- [42] J. W. S. Liu, W. Shi, K. Lin, and R. Bettati. Imprecise computations. *Proceedings of the IEEE*, 82(1):83–94, January 1994.
- [43] S. McCanne and V. Jacobson. Vic: A flexible framework for packet video. In *ACM Multimedia*, pages 511–522. San Francisco, CA, USA, November 1995.
- [44] A. McKinlay, R. Procter, O. Masting, R. Woodburn, and J. Arnott. Studies of turn-taking in computer-mediated communications. *Interacting with Computers*, 6(2):151–171, 1994.
- [45] C. W. Mercer and R. Rajkumar. An interactive interface and RT-Mach support for monitoring and controlling resource management. In *Proceedings of the Real-Time Technology and Applications Symposium*, pages 134–139, May 1995.

- [46] C. W. Mercer, S. Savage, and H. Tokuda. Processor Capacity Reserves: An Abstraction for Managing Processor Usage. In *Proceedings of the Fourth Workshop on Workstation Operating Systems (WWOS-IV)*, October 1993.
- [47] C. W. Mercer, S. Savage, and H. Tokuda. Processor Capacity Reserves for Multimedia Operating Systems. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, pages 90–99, Boston, May 1994.
- [48] G. Miller, G. Baber, and M. Gilliland. News on-demand for multimedia networks. In *Proceedings of the ACM Multimedia*, pages 383–393, 1993.
- [49] A. K. Mok. *Fundamental Design Problems of Distributed Systems for the Hard Real-Time Environment*. PhD thesis, Dept. Electrical Engineering, MIT, 1983.
- [50] MPI Forum. MPI: A message-passing interface standard. *International Journal of Supercomputing Applications*, 8(3/4):165–416, 1994.
- [51] K. Nahrstedt and J. M. Smith. QOS negotiation in a robotics environment. In *Workshop on Distributed Multimedia Applications and QOS Verification*, Montréal, Canada. June 1994.
- [52] K. Nahrstedt and J. M. Smith. The QOS broker. *IEEE Multimedia*, 2(1):53–67, Spring 1995.
- [53] J. E. Neilson. PARASOL: A simulator for distributed and/or parallel systems. Technical Report SCS-TR-192, School of Computer Science, Carleton University, Carleton University, Ottawa, May 1991.
- [54] C. Nicolaou. An architecture for real-time multimedia communication systems. *IEEE Journal on Selected Areas in Communications*, 8(3), April 1990.
- [55] J. Nieh, J. D. Northcutt, J. G. Hanko, and G. Wall. SVR4 UNIX scheduler unsuitable for multimedia applications. In *Fourth International Workshop on Network and OS Support for Digital Audio and Video (NOSSDAV'93)*, Lancaster, UK, November 1993.

- [56] J. K. Ousterhout. Scheduling techniques for concurrent systems. In *Proceedings of the 3rd International Conference on Distributed Computing (ICDCS)*, pages 22–30, October 1982.
- [57] A. K. Parekh and R. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The multiple node case. *IEEE/ACM Transactions on Networking*, 2(2):137–150, 1994.
- [58] C. Partridge. *Gigabit Networking*. Addison-Wesley, Reading, Massachusetts, USA, 1993.
- [59] R. Philippe, G. Coulson, A. Campbell, G. S. Blair, and M. Papathomas. Implementing a QOS controlled ATM based communications system in CHORUS. Technical Report MPG-94-05, Department of Computing, Lancaster University, Lancaster, U.K., 1994.
- [60] G. Raghunath, J. Neilson, and M. Woodside. Scheduling support for QOS guarantees in end systems. In *Proc. Workshop on Distributed Multimedia Applications and Quality of Service Verification*, Montreal, May 1994.
- [61] K. K. Ramakrishnan, L. Vaitzblit, C. Gray, U. Vahalia, D. Ting, P. Tzelnic, S. Glaser, and W. Duso. Operating system support for a video-on-demand file service. *ACM/Springer Verlag Journal on Multimedia Systems*, 1994.
- [62] P. V. Rangan and D. C. Swinehart. Software architecture for integration of video services in the etherphone system. Technical Report CS90-189, Department of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, December 1990.
- [63] D. J. Reininger, D. Raychaudhuri, and J. Y. Hui. Bandwidth renegotiation for VBR video over ATM networks. *IEEE Journal on Selected Areas in Communications*, 14(6):1076–1085, August 1996.
- [64] P. Robin, G. Coulson, A. Campbell, G. Blair, and M. Papathomas. Implementing a QOS controlled ATM based communication system in CHORUS. In

- Proc. 4th IFIP International Workshop on Protocols for High Speed Networks*, Vancouver, Canada, August 1994.
- [65] S. Schenker. Fundamental design issues for the future Internet. *IEEE Journal on Selected Areas in Communications*, 13(7):1176-1188, 1996.
- [66] L. C. Schreier and M. B. Davis. System-level resource management for network-based multimedia applications. In *Workshop on Network and Operating Systems Support for Digital Audio and Video*, Durham, NH, April 1995.
- [67] H. Schulzrinne. Dynamic configuration of conferencing applications using pattern-matching multicast. In *NOSSDAV'95: Proc. Workshop on Network and Operating System Support for Digital Audio and Video*, Durham, NH, USA, April 1995.
- [68] L. Sha, J. P. Lehoczky, and R. Rajkumar. Solutions for some practical problems in prioritized preemptive scheduling. In *IEEE Real-Time Systems Symposium*, pages 181-191. December 1986.
- [69] R. Steinmetz. Analyzing the multimedia operating system. *IEEE Multimedia*, pages 68-84, Spring 1995.
- [70] I. Stoica, H. Abdel-Wahab, and A. Pothén. A microeconomic scheduler for parallel computers. In D. Feitelson. and L. Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, volume 949 of *Lecture Notes in Computer Science*, pages 200-218, Berlin, 1995. Springer-Verlag.
- [71] J. K. Strosnider. *Highly Responsive Real-Time Token Rings*. PhD thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA USA, August 1988.
- [72] C. Szyperski and G. Ventre. A characterization of multi-party interactive multimedia applications. Unpublished Technical Report, Information Systems Institute, University of California, Berkeley. 1993.

- [73] W. Tawbi and E. Horlait. Expression and management of QOS in multimedia communication systems. *Annals of Telecommunication*, 49(5–6):282–296, 1994.
- [74] W. Tawbi, F. Horn, E. Horlait, and J. B. Stéfani. Video compression standards and quality of service. *The Computer Journal*, 36(1):43–54, January 1993.
- [75] T.D.C.Little. A framework for synchronous delivery of time-dependent multimedia data. *Multimedia Systems*, 1(2):87–94, 1993.
- [76] T.D.C.Little and A. Ghafoor. Synchronization and storage models for multimedia objects. *IEEE Journal on Selected Areas in Communications*, 8(3), April 1990.
- [77] H. Tokuda, M. Kotera, and C. W. Mercer. An integrated time-driven scheduler for the ARTS kernel. In *8th IEEE Phoenix Conference on Computers and Communication*. March 1989.
- [78] H. Tokuda, Y. Tobe, S. T.-C. Chou, and J. M. F. Moura. Continuous media communication with dynamic QOS control using arts with an FDDI network. In *ACM SIGCOMM'92 Conference Proceedings*, pages 88–98. August 1992.
- [79] D. Venkatesh and T. Little. Dynamic service aggregation for efficient use of resources in interactive video delivery. In *NOSSDAV'95: Proc. Workshop on Network and Operating Systems Support for Digital Audio and Video*, Durham NH, USA, April 1995.
- [80] A. Vogel, B. Kerhervé, G. von Bochmann, and J. Gecsi. Distributed multimedia and QOS: A survey. *IEEE Multimedia*, pages 10–18, summer 1995.
- [81] A. Vogel, G. von Bachmann, R. Dssouli, J. Gecsei, A. Hafid, and B. Kerherve. On QOS negotiation in distributed multimedia applications. Technical Report 891, Université de Montréal, Dpt. d'IRO, January 1994.
- [82] C. Waldspurger and W. Weihl. Lottery scheduling: Flexible proportional-share resource management. In *Proceedings of the First Symposium on Operating*

- Systems Design and Implementation*, pages 1–11, Monterey, CA, November 1994.
- [83] G. A. Wall, J. Hanko, and J. D. Northcutt. Bus bandwidth management in a high resolution video workstation. In *Proceedings of the International Workshop on Network and Operating Systems Support for Digital Audio and Video*, Heidelberg, Germany, November 1992.
- [84] M. Woo and A. Ghafoor. Multichannel scheduling for communication of pre-orchestrated multimedia information (homogeneous channels case). In *IEEE INFOCOMM*, pages 920–927, Toronto, Canada, 1994.
- [85] D. K. Y. Yau and S. S. Lam. Adaptive rate-controlled scheduling for multimedia applications. In *ACM Multimedia Conference*, Boston, MA, November 1996.
- [86] L. C. Yun and D. G. Messerschmitt. Architectures for multi-source multi-user video compositing. In *ACM Multimedia*, pages 215–223, 1993.
- [87] H. Zhang and D. Ferrari. Rate-controlled static priority queuing. In *INFOCOM*. 1993.
- [88] L. Zhang. Virtual clock: A new traffic control algorithm for packet switching networks. In *Proceedings of ACM SIGCOMM'90*, pages 19–29, Philadelphia, PA, USA, September 1990.

IMAGE EVALUATION TEST TARGET (QA-3)

APPLIED IMAGE, Inc.
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989

© 1993, Applied Image, Inc., All Rights Reserved