Quality Adaptation in a Multisession Multimedia System: Model, Algorithms and Architecture

by

Md. Shahadatullah Khan

B.Sc. (Electrical & Electronic Engineering), Bangladesh University of Engineering & Technology, Dhaka, 1992
M.A.Sc. (Electrical & Computer Engineering), University of Victoria, 1994

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Electrical & Computer Engineering

We accept this dissertation as conforming to the required standard

---

Dr. K.F. Li, Supervisor (Dept. of Elec. & Comp. Eng.)

Dr. E.G. Manning, Supervisor (Depts. of Comp. Sc. and Elec. & Comp. Eng.)

Dr. N.J. Dimopoulos, Departmental Member (Dept. of Elec. & Comp. Eng.)

Dr. F. El Guibaly, Departmental Member (Dept. of Elec. & Comp. Eng.)

Dr. H. Tokuda, External Examiner (Faculty of Env. Inf., Keio Univ., Japan)

Supervisors: Dr. Kin F. Li and Dr. Eric G. Manning

Abstract

Flexible and adaptive quality of service (QoS) is desirable for real-time multimedia applications. Suppose a multimedia system is supporting a 30 frame/second video stream which is using a network bandwidth of 2 Mbps, and due to network congestion the network bandwidth is reduced to 1 Mbps. It is desirable that the system supports graceful adaptation of quality of the video stream, for example, by reducing the frame rate to 15 frame/second. The focus of this dissertation is to investigate the design of an adaptive multimedia system (AMS) with multiple concurrent sessions, where the quality of individual sessions is dynamically adapted to the available resources and to the run-time user preferences.

We propose the *Utility Model* – a mathematical model to capture the issues of resource management within multisession AMSs. In this model, each session provides a quality profile, which is a set of operating qualities arranged from the minimum acceptable quality to the maximum desired quality. Any operating quality may be mapped to the required resources using a quality-resource mapping, and also to a session utility using a quality-utility mapping. The main problem in a multisession AMS is to find an operating quality for each session such that the overall system utility (e.g. system revenue) is maximized under the system resource constraints. This is called the adaptive multimedia problem (AMP). The Utility Model formulates the AMP as the multiple-choice multi-dimension 0-1 knapsack problem (MMKP). It provides a unified and computationally feasible way to solve the admission problem for new multimedia sessions, and the dynamic quality adaptation and integrated resource allocation problems for existing sessions.

We present the *Padma Architecture* – a system architecture for multisession AMSs. This architecture has two novelties: (1) integrated and adaptive management of system resources based on the Utility Model, and (2) the use of metaspaces to encapsulate the machinery of quality adaptation. The former provides improved resource utilization and dynamic quality adaptation, and the latter provides the application programmers freedom from the concerns of low-level resource management issues while developing multimedia applications.

Finally, we present the Utility Model Demonstration Prototype (UMDP) – a prototype which demonstrates the capability of the Utility Model to handle admission control, quality adaptation and integrated resource allocation in a unified way. We evaluate the performance of UMDP using random sequences of events, and show that the system utility achieved by the UMDP is significantly higher than that of a simple reservation model prototype (SRMP). For applications such multimedia service providers, it means that UMDP will generate more revenue than SRMP from the same amounts of system resources provisioned.

Examiners:

Dr. K.F. Li, Supervisor (Dept. of Elec. & Comp. Eng.)

Dr. E.G. Manning, Supervisor (Depts. of Comp. Sc. and Elec. & Comp. Eng.)

Dr. N.J. Dimopoulos, Departmental Member (Dept. of Elec. & Comp. Eng.)

Dr. F. El Guibaly, Departmental Member (Dept. of Elec. & Comp. Eng.)

Dr. H. Tokuda, External Examiner (Faculty of Env. Inf., Keio Univ., Japan)

Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my heartiest appreciation to my supervisors Dr. Kin Li and Dr. Eric Manning for their continuous support and active guidance during this research work. Their visions and experiences were vital to shape my ideas into this dissertation.

I would like to thank members of my dissertation committee Dr. Nikitas Dimopoulos and Dr. Fayez El Guibaly for taking time from their busy schedule, and for the valuable suggestions they provided.

Discussions with the fellow researchers at the LAPIS and PANDA laboratories have proved to be very valuable and enlightening. John Foxgord and Glen Chen provided valuable comments on some earlier drafts. The cooperation of Vassilios Dimakopoulos, Michael Horie, Kenichi Murata, James Pang, and Dr. Ali Shoja deserve special mention. Thanks for bearing with me and my camera!

The unending love, inspiration and confidence of my family members and friends provided strong support for this research. Ma, what is the source of your strength to support all of us? Had I not had friends like Nahar Vabi, Nargis Vabi, Mitu, Nita, Ali Vai, Mahmood Vai, Mizan and Siddiquee Vai, life in Victoria would have been very difficult. Thank you all.

The financial support of the Canadian Commonwealth Scholarship and NSERC is gratefully acknowledged.

To my father who showed me the light,

and to my mother who guided me through the dark.

Chapter 1

Introduction

Recent advances in computing and telecommunications have opened up a plethora of new applications. Computers are getting faster and cheaper, networks are providing higher bandwidth and lower error rates, internetworking is pervasive, and multimedia technology is expanding the domain of applications. All these developments have enabled the technology of distributed multimedia systems (DMS), where physically remote users interconnected by a network enjoy real-time communication employing speech and images as well as text.

We note two interesting properties of distributed systems and multimedia applications:

- Distributed systems are inherently *dynamic*; that is they exhibit strongly time-varying behavior. Some of the causes are variation of user demand and hence load, change of available resources, mobility of hosts, and unpredictable failures.

- Multimedia applications are potentially *adaptive* [5, 12]. For example, one can have real-time video transmitted at a 30 frame/sec refresh rate when there is enough network bandwidth available, or one may reduce the bandwidth requirement to half by reducing the rate to 15 frame/sec.

This observation raises the following question: How do we design a system to exploit the adaptive property of multimedia applications? The adaptive property of multimedia applications makes them very suitable candidates for a dynamic environment as provided by distributed systems. The objective of our research is to design an adaptive multimedia system (AMS) where the perceived quality of the multimedia applications is dynamically adapted to the state of the system, in particular to the changes of resource availability and user preferences.

1.1 Motivation

Multimedia applications require system support with quality of service (QoS) guarantees. For instance, in order to support a video stream at a refresh rate of 10 frame/sec, the stream handler has to be scheduled onto the CPU once every 0.1 second. Or, to support a telephone quality audio conversation, the data transport system should be able to provide a data-rate of about 16 kbps with end-to-end delay in the range of 50 msec[36].

The QoS may be expressed using a set of qualitative or quantitative parameters, where each parameter relates to one property of the service [46]. For example, the QoS perceived by a user of a session depends on the user's subjective evaluation. The user may express her perception using expressions such as good, OK, bad, or choppy but understandable. On the other hand, the QoS for a network connection may be expressed using parameters such as network bandwidth, delay, loss and jitter.

It is well-known that current systems, such as the Unix operating system or the Internet, cannot deal with the QoS requirements of multimedia applications, especially the interactive ones with strong end-to-end latency requirements. It is very important to investigate the issues of system design which can provide QoS guarantees.

One may think that having sufficient resources will solve the QoS guarantee problem. However, will resources ever be enough? In most of the cases, it appears that demand for resources is increasing at a higher rate than the increase in supply of resources. To enforce QoS guarantee effectively, resources in a multimedia system must be managed in an integrated way. In a traditional video transmission system, it is possible that the encoded video stream is getting sufficient network bandwidth, but cannot use this bandwidth effectively because the system cannot allocate sufficient CPU cycles to decode and render the stream. An integrated resource management system should eliminate problems like this.

Designing an adaptive system with integrated resource management requires a comprehensive understanding of the problems, issues and dynamics within such a system.

1.2 Adaptive Multimedia Systems

Figure 1.1 illustrates an audio-visual transmission system (AVTS), based on the classical client-server model. The server manages the generation and transmission of audio and video streams, while the client manages reception and presentation. On the server side, a video camera records the time sequences of video images and a microphone records audio samples at periodic intervals. The signals are then encoded into representations which can be stored, transmitted, and later presented. The server transmits this representations using a transport system. The client decodes the media, and passes the video and audio streams to the display and speaker systems for presentation.

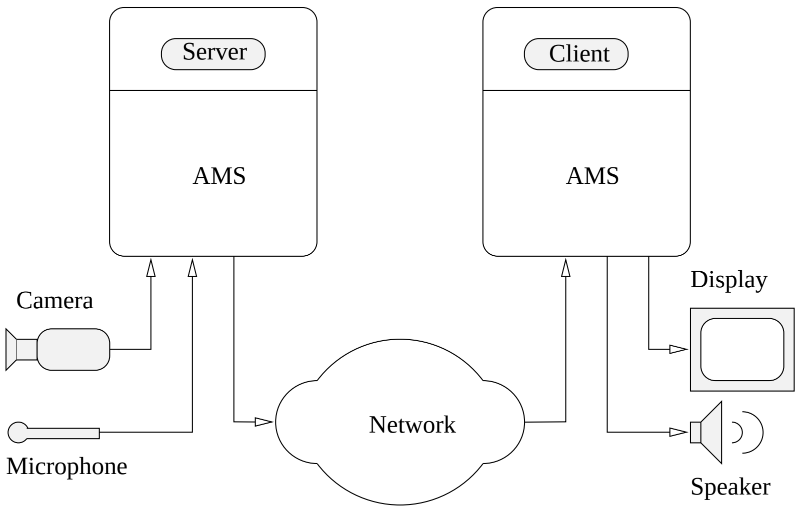


Figure 1.1: *An Audio-Visual Transmission System: an adaptive multimedia system (AMS) application.*

Suppose an AVTS session is using a network bandwidth of 3 Mbps [Figure 1.1]. Now suppose, due to network congestion, that the bandwidth available to the session drops from 3 Mbps to 1 Mbps. How should the system adapt to this problem? It must *scale* some or all of its media[1] components, e.g. scale-down[2] video to a lower

[1]A *medium* can be defined as a representation of information. Examples of media are video, audio, images, and text.

[2]For a *scalable* medium, the quality may be adjusted depending on the availability of resources. When sufficient resources are available, the system should provide the maximum desired quality. However, when available resources cannot support the maximum desired quality, quality may be scaled down. Here *scale-down* means compromising media quality as compared to the desired quality. Thus different lev-

resolution, and/or scale-down stereo audio to mono quality.

1.3 The Adaptive Multimedia Problem

Let us consider a multiuser multimedia system configuration as shown in Figure 1.2. We assume a small group of users, such as a family or a research group. Each user uses her own multimedia terminal for media recording and playout, and contacts the media server for the processing, storage and transmission of media. At any time, the media server may have to support multiple concurrent sessions from multiple users.

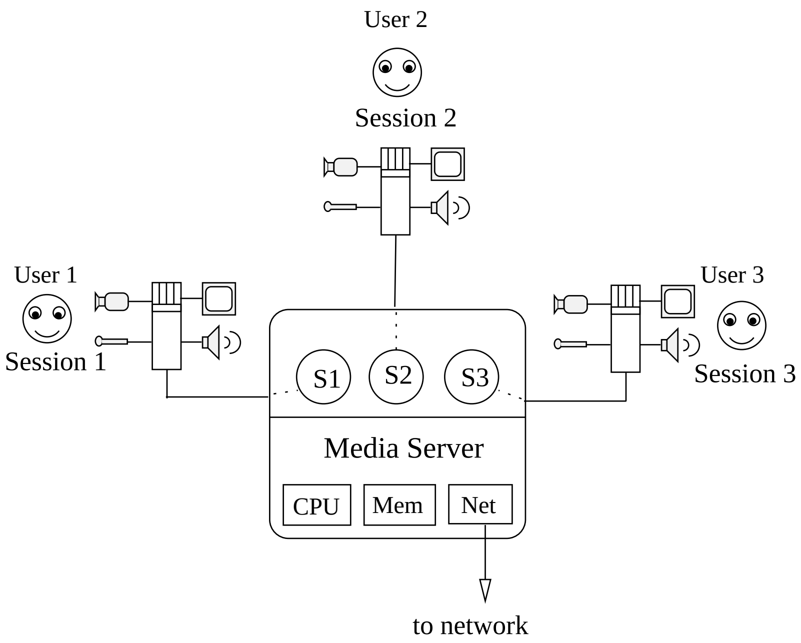


Figure 1.2: *A multiuser multimedia system with a multisession media server.*

Suppose user 1 is participating in a video conference, user 2 is watching a movie,

els of scale-down imply different levels of quality, and no scale-down implies the desired quality.

and user 3 is visiting a virtual entertainment park. At times, the resources of the media server may not be sufficient to provide the maximum desired quality of media to all sessions, for example because the CPU is overloaded, or the network is congested. How should this system adapt to these dynamic situations? When sufficient resources are available, the system should provide every user the maximum desired quality. However, when available resources cannot support the maximum desired quality, the system may compromise the quality. For video, this may mean reduced refresh rate or reduced resolution, and for audio it may mean mono quality instead of stereo. Thus in Figure 1.2, the server should be able to adapt the QoS of an individual session, based on policies concerning relative importance and other constraints. For example, when a higher-priority video-conference session needs better quality, compromising the quality of a lower-priority cyberspace session is an option.

In a multisession system with dynamically changing users' demands and dynamically changing amounts of unallocated resources – *resource availability* – the problem of determining the QoS of individual sessions, in order to maximize some objective function, subject to a set of system resource constraints, is called the *Adaptive Multimedia Problem (AMP)*.

Adaptation in a multimedia system may be triggered by many events, such as change of system load, change of network load or congestion/failure in the network, and dynamic change of users' requirements. An AMS should provide an adaptive and integrated resource management scheme in order to make best use of the system resources. Effective solution of the AMP involves many factors such as the system objective, the relative importance of sessions, the relative importance and interrelations of media within a session, and users' requirements/constraints.

1.4 Research Challenges

Some of the challenges involved in the design of an adaptive multimedia system are as follows:

1. **Understand *adaptive* multimedia systems:**

   The first challenge is to develop a comprehensive understanding of the issues and dynamics of the adaptive multimedia systems. This involves the following questions.

   - What are the requirements of an adaptive multimedia system? (What do the users of an adaptive multimedia system require?)
   - What are the sources and characteristics of system dynamics? What are the forces behind adaptation, and what are the possible results? What is the relation of the forces to the results?
   - How do media qualities relate to resource requirements? How do we distribute system resources among the concurrent sessions, and how do we distribute each session's resources to its media components? How do we take care of dependence and interrelations among media components and sessions?
   - How do we achieve a sufficiently general but clean design and implementation, to meet the real-time performance requirements?

2. **System architecture design:**

   The system architecture for an AMS has to support adaptive QoS-based service requirement of the users by managing the system resources using a QoS-based discipline. Some of the key questions here are:

- What kind of layering should the architecture use, if any? What are the functionalities and adaptation issues at different layers?
- What are the components required in an AMS? How should they be organized for adaptive multimedia? How do the components cooperate to provide adaptive quality?
- How does the user specify adaptive multimedia requirements?
- How does the system map qualities from one layer to another? For example, how does the system map users' requirements to system QoS parameters, and then to resource requirements?
- How does the system monitor quality of individual media components, or the overall quality of a multimedia session?
- When and how does the system make the adaptation decisions? How does the system enforce these decisions?

3. **Reservation-based resource management:**

   In order to provide the service with QoS guarantees required by multimedia applications, we require a reservation-based resource management scheme, where some resources are reserved to guarantee the minimum QoS of a session. For example, to support end-to-end bandwidth guarantee of a connection, we have to reserve resources at every switch or router along the connection path, and at the end computers. In order to provide adaptive service quality, the resource management scheme must support adaptation. The most important resources of an AMS are

   - processor cycles,

- main memory, and
- network bandwidth.

Some of the specific questions that have to be answered are as follows:

- How do we enforce discipline in resource management to make best use of the resources, as opposed to the wholly inadequate best-effort resource management of the Internet or of traditional Unix-like operating systems?
- How is the CPU scheduled? Does the system support both QoS-based and non-QoS-based (best-effort) tasks at the same time?
- Does the system support locking of main memory pages? It is important, because otherwise swapping may lead to prohibitive memory access latencies for time-critical data.
- How much bandwidth should be reserved and how much should be used for best-effort services?
- How does the system monitor the resource availability, and when does it trigger adaptation?
- Enforcement: Once resources are reserved, how does the system ensure that the users do not consume more than their share of resources?

1.5 Our Focus and Approach

In the previous section, we have identified three main challenges, namely, understanding adaptive multimedia systems, designing a system architecture for AMSs, and finally the management of the system resources for providing the QoS-based services required by multimedia applications. Dealing with all these challenges is beyond the scope of this dissertation; our research focuses on the first two challenges. Research at Keio University [21], University of Pennsylvania [35], IBM ENC at Heidelberg [51] and University of Kentucky [24] is addressing the third issue.

We have taken a top-down approach – from the users' requirements to the system model, and finally to the system architecture. At first, we analyze the adaptive requirements of users, and study the challenges of satisfying them in a dynamic environment. We develop a mathematical model for the adaptive multimedia system which relates users' requirements to system dynamics. It may be used to make admission and adaptation decisions for the system. Finally we use this model to design a system architecture for adaptive multimedia. Work was done in three phases:

- The first phase was to understand the factors and forces within an adaptive multimedia system. We developed a new mathematical model, the *Utility Model*, which captures the dynamic requirements, issues and goals of a multimedia system using concepts such as quality profile, quality-resource mapping and session and system utilities. This model expresses the adaptive multimedia problem as a multiple-choice multi-dimension knapsack problem (MMKP). We also provide two solutions for the MMKP.

- The second phase was to derive a suitable end-system architecture for adaptive multimedia. We propose a layered end-system architecture called the Padma Architecture. This architecture is based on the Utility Model.

  We believe that multimedia application programmers should not be concerned with the quality adaptation and low level resource management issues. We propose an end-system architecture using the metaspace concept for abstracting the machinery of adaptation. The Metaspace concept was originally proposed for designing a flexible and adaptive operating system called Apertos [53]. To our knowledge, no other project has used the metaspace construct to design a QoS-based adaptive system.

- The final phase was to validate the concepts of the Utility Model, using prototype implementation and experimental evaluation.

1.6 Outline

This dissertation contains seven chapters, organized as follows:

- Chapter 1 provides the motivation of this research, introduces the adaptive multimedia problem, and explains our objective and approach.
- Chapter 2 presents background material and surveys related work.
- Chapter 3 proposes the Utility Model – a mathematical model for adaptive multimedia systems. Here the adaptive multimedia problem is mapped to the multiple-choice multi-dimension knapsack problem (MMKP).
- Chapter 4 presents two solutions for the MMKP: a branch and bound algorithm for optimal solution and a heuristic for fast but near-optimal solution.
- Chapter 5 describes the Padma end-system architecture for distributed multimedia with quality adaptation.
- Chapter 6 describes the prototype implementation of the Utility Model, and presents our experimental results.
- Finally, Chapter 7 summarizes our contributions, and provides directions for future work.

Chapter 2

Background

In this chapter, we provide definitions and explain fundamental concepts. We also survey related work, to explain the context of our research.

2.1 Multimedia

A *medium* is a representation of information such as audio, video, images or text[1]. A medium is digitally represented by a binary sequence called a media *stream*. A medium implies *sources* such as microphones and cameras, and *sinks* such as speakers and display monitors. Media representations can be stored in storage systems, and transmitted over communication *channels* (such as wireless, coaxial

[1]Unfortunately the multimedia literature defines medium in a way different from the standard definition in electrical engineering, where medium implies a transmission channel such as twisted pair, coaxial cable, optical fiber and wireless.

cable, and fiber) using *coding* techniques (such as PCM[2], ADPCM[3], JPEG[4], and MPEG[5]) and *modulation* techniques (such as SSB[6] and FM[7]).

Media may be of two types [46]:

- *Time-dependent or continuous media:* Information in these media is a function of time. For example, sound and video information change over time, and as such they are time-dependent. The processing of these media is time-critical because the validity and correctness of the data depend on a time constraint. Generally, time-dependent media are represented using periodic and continuous sequences. For example, video may be represented as a periodic sequence of static images, and audio may be represented as a sequence of periodic samples. These media are therefore called *continuous media*. Continuous media may be of two types: (1) *interactive*, such as real-time video conference, where there is a tight timing relationship between the source and sink of the media; and (2) *playback*, such as multimedia mail, where the relationship is looser [44].

- *Time-independent or discrete media:* Information in these media consists exclusively of a sequence of individual elements, and does not depend on time. Examples of time-independent media are text and graphics. These media are also called discrete since they are not constrained to the time continuum.

Multimedia suggests combinations of media. For this work, *multimedia* implies integration of various media which may include continuous media such as audio

[2]PCM: Pulse Code Modulation.
[3]ADPCM: Adaptive Differential Pulse Code Modulation.
[4]JPEG: Joint Photographic Experts Group.
[5]MPEG: Motion Pictures Experts Group.
[6]SSB: Single Side Band.
[7]FM: Frequency Modulation.

and video of interactive or playback type, and discrete media such as still images, text and graphics. Examples of multimedia applications include video-on-demand, multimedia email, video conference, hypermedia course-ware [46], and distributed virtual environments (DVEs) [3].

2.1.1 Characteristics of Continuous Media

Due to the time-dependent nature of continuous media, dealing with continuous media is more challenging than dealing with discrete media. Continuous media (CM) applications impose special requirements on the underlying system:

- Timely service: Timely service is essential to QoS perception. For instance, for acceptable quality of interactive video the end-to-end delay should not be more than 250 msec, and for lip synchronization of audio and video streams the synchronization skew should not be more than 80 msec [35].

- High throughput: The stream-like behavior of CM demands relatively high data throughput. For example, CD-quality audio requires a bandwidth of 1.5 Mbps, while HDTV compressed MPEG video requires a bandwidth of 20 Mbps [44].

CM applications also have some favorable properties [17, 35]:

- The resource requirements of CM are not always stringent – they may potentially be adjusted to resource availability. For example, one can have real-time video transmitted at 30 frame/sec refresh rate when there is enough bandwidth available, or one may reduce the bandwidth requirement by half, by reducing the rate to 15 frame/sec. That is, video may be *scalable* (if the coding technique is).

- Raw CM data usually show high redundancy. This property enables coding algorithms to achieve high compression ratios.

- Generally the timing requirements of CM applications are soft as opposed to hard real-time applications where a deadline miss may be catastrophic. For example, if an uncompressed video frame (or parts of it) is not available, it may simply be omitted. The viewer hardly notices the omission of a frame unless it happens for a long contiguous sequence of frames.

- A sequence of CM data is usually the result of a periodic sampling of a sound or image signal. For this reason, most of the time-critical operations for processing CM streams are periodic in nature, and are much easier to handle than aperiodic operations.

2.2 Levels of QoS in AMS

In an AMS, QoS may arise at many levels such as [35]:

- *Perceived QoS*: The QoS perceived by a user of a session depends on the user's subjective evaluation. The user may express her perception using expressions such as good, OK, bad, or choppy but understandable. It is highly influenced by psychological and physiological factors such as hearing, vision, taste and training. For example, a certain audio quality may be acceptable to a general user, but may be totally unacceptable to a trained musician.

- *Media QoS*: The perceived QoS of a session depends on the media QoS, which involves the quality parameters of individual media, and the relation among media components. For example, video refresh rate (frame/sec), video resolution (pixel/cm^2), color content (number of color or gray-scale bits), end-to-end delay (msec), audio sampling rate, number of audio channels (mono, stereo or surround), and video-audio synchronization (msec) are all relevant.

- *System QoS*: It describes requirements placed on the communication services and other operating system services. For example, the system transport system should support throughput and delay guarantees[8], and the system scheduler should guarantee timely processing of time-critical data.

- *Network and Device QoS*: The network QoS parameters include network bandwidth, delay, loss and jitter. Example of device QoS include data bandwidth to the video output, and sampling rate of the audio device.

One interesting research problem here is how to translate QoS parameters at one level to QoS parameters at another level. This is called *QoS translation*. Research at Keio University [38], University of Pennsylvania [35] and Washington University at St. Louis [15] is addressing this problem. For this research, we assume that such translations are available.

2.3 Media Scaling

To reduce the resource requirements of a media stream, it is sometimes possible to subsample it to get another valid media stream of poorer quality. This operation is called *media scaling*, and encodings for which this is possible are called *scalable*.

[8]The present Internet, with its best-effort datagram service, is fundamentally unable to provide such guarantees. ATM networks, on the other hand, can.

For example, dropping the odd frames of a Motion-JPEG video stream produces another Motion-JPEG video stream but with half the refresh rate.

Scaling a stream can be done either at the source or at the sink. For example, refresh rate reduction of a video stream is usually done at the source whereas scaling using hierarchical coding may be done at the sink.

Continuous and Discrete Scaling

Based on the granularity of quality control, scaling may be of two types: continuous or discrete[9] [12]. In *continuous scaling* the stream quality can be adjusted finely, whereas in *discrete scaling* the choice of stream quality is limited to a small set. Suppose an audio stream is composed of two sub-streams (L+R) and (L-R), where L is the stream for the left speaker and R is the stream for the right speaker. When sufficient resources are available, both the sub-streams are transmitted, and the receiver separates the L and R streams out for stereo quality sound. Now suppose, due to a network congestion the network bandwidth available for this audio stream has to be reduced. In this situation, either continuous or discrete scaling may be used.

- Continuous scaling: Change the sampling rate of the sub-streams such that the aggregate bandwidth of the (L+R) and (L-R) sub-streams matches the available network bandwidth. The receiver now gets a stereo stream but with a poorer quality.
- Discrete scaling: Drop the (L-R) sub-stream. Mono audio can be obtained at the receiver by using (L+R)/2.

[9]To comply with the multimedia literature, here we use the terms continuous and discrete to imply fine-grained and course-grained respectively.

The choice between continuous or discrete scaling depends on media characteristics and encoding techniques. Both continuous and discrete scaling have their own strengths. Continuous scaling provides more control over the scaling than does discrete scaling. However, discrete scaling is usually simpler to implement. In the above example, implementing discrete scaling is straightforward, and the only choice of bandwidth reduction is by 50%. On the other hand, implementing continuous scaling requires changing the sampling rate, but it can adapt to any reasonable reduction of bandwidth leading to the possibility of better resource utilization.

There may be cases where continuous scaling is not possible or meaningful. For example, for applications involving stored MPEG stream, changing sampling rate is not an option. Also there are cases, such as choosing among mono, stereo or surround for audio, or choosing between VHS, super VHS or HDTV for video, where discrete scaling is more natural for users.

Hierarchical Coding Techniques

Recently, new coding techniques have been developed to simplify media scaling [5]. For example, the MPEG-2 standard uses hierarchical coding techniques where the media are coded using several layers corresponding to importance levels. Media scaling may be applied by just dropping the least important levels. An MPEG-2 video stream may be represented using an essential base layer (also called the *main profile*), and one or two optional enhancement layers (alternately called *scalability profiles*). The base layer constructs the coarse or base representation of the stream, and then the enhancement layers successively improve it.

Layer name	Profile	Frame size	Bit rate	Subjective QoS
Base layer (BL)	main	304x112	0.32 Mbps	VHS
Enhancement 1 (E1)	spatial	608x224	0.83 Mbps	Super VHS
Enhancement 2 (E2)	SNR	608x224	1.85 Mbps	Laser Disc

Table 2.1: *MPEG-2 hybrid scalable bit-stream using spatial and SNR scalability.*

Table 2.1 from [40] shows the scalability of MPEG-2 with one base layer and two enhancement layers. Here the base layer (BL) gives VHS quality video with a frame resolution of 304x112 and requires a data rate of 0.32 Mbps. The enhancement layer (E1) improves resolution to the super VHS resolution of 608x224 pixels with an additional data rate requirement of 0.83 Mbps. Finally the second enhancement layer (E2) improves the video to laser disc quality by improving the quantization step size; it requires an additional data rate of 1.85 Mbps.

Filtering

Filtering is a general concept which may mean arbitrary manipulation of a data stream [42][10]. Filtering may be done at any part of the distributed system such as source, intermediate node/router or sink. Some of the operations done by media filters include discarding parts of the stream (scaling), encrypting the data for security and protection, or even recoding the previously coded data stream into a new format. The compression of audio/video media into compressed format, and decompression of compressed data into audio/video media can also be considered

[10]Unfortunately this definition is also different from the standard one used in electrical engineering, where it means discarding part of the signal (low-pass filters) or refining the signal (noise reduction).

as filtering. Active research on media scaling and filtering is going on at IBM ENC, Heidelberg [12], Columbia University [14], Lancaster University [52] and Princeton University [54].

This work studies the dynamics of adaptive multimedia systems where quality of media streams are adapted to available system resources, and proposes algorithms and system architecture to realize such systems. Parts of our discussion assume that media streams are scalable. Chapter 3 presents a mathematical model on how to take scaling decisions for media streams, and Chapter 5 provides an end-system architecture using media scaling objects such as filters.

2.4 Resource Management

Multimedia applications pose new challenges to system design including protocols, processor scheduling, memory management, network and transport system, and device handling. In order to support the QoS guarantees required by distributed multimedia applications, we require new reservation-based resource[11] management mechanisms to administer the available resources. Research in reservation-based resource management has made considerable progress in the last few years. In the following subsections, we introduce some of the fundamental concepts. Our discussion is adapted from contributions of many research projects such as the Omega project at the University of Pennsylvania [35], the DiME project at IBM ENC,

[11] A *resource* is a system entity required by processes for manipulating data [46]. A resource can be used exclusively (e.g. loudspeaker), or shared among various processes (e.g. bandwidth). Services for distributed multimedia applications need resources for their operation. Of special interest are resources which are shared among applications, system and network, such as CPU cycles, system memory and network bandwidth.

Heidelberg [51], and the MMP project at Keio University [21].

There are two approaches to resource reservation:

- The *pessimistic approach* reserves resources for the worst case workload such as longest CPU processing time, highest bandwidth, and so on. This approach can lead to resource under-utilization, but avoids resource conflicts and guarantees quality. The telephone network uses it.

- The *optimistic approach* reserves resources according to an average workload and meets QoS guarantees as much as possible. In an overload situation, the reservation system may fail to guarantee the quality of service of a particular session. The system usually implements some monitoring functions to detect overloads, and solves the problem by preempting processes according to their importance.

The mechanisms for reservation-based resource management are based on three fundamental steps [36, 51]:

- Admission and reservation: check whether the QoS demands of a session can be satisfied; if sufficient resources are available, reserve the amounts of resources required for the QoS guarantee; if not, either renegotiate the QoS or reject the session;

- Resource allocation [12] and policing: ensure that the given QoS guarantees are

[12] In this dissertation, we distinguish resource *reservation* and *allocation*. When the system admits a session with a minimum QoS guarantee, the system *reserves* sufficient resources in order to ensure this minimum QoS guarantee. Once a session is admitted, the system cannot revoke the reserved resource during the life-time of the session. However at run-time, the system may *allocate* more resources than

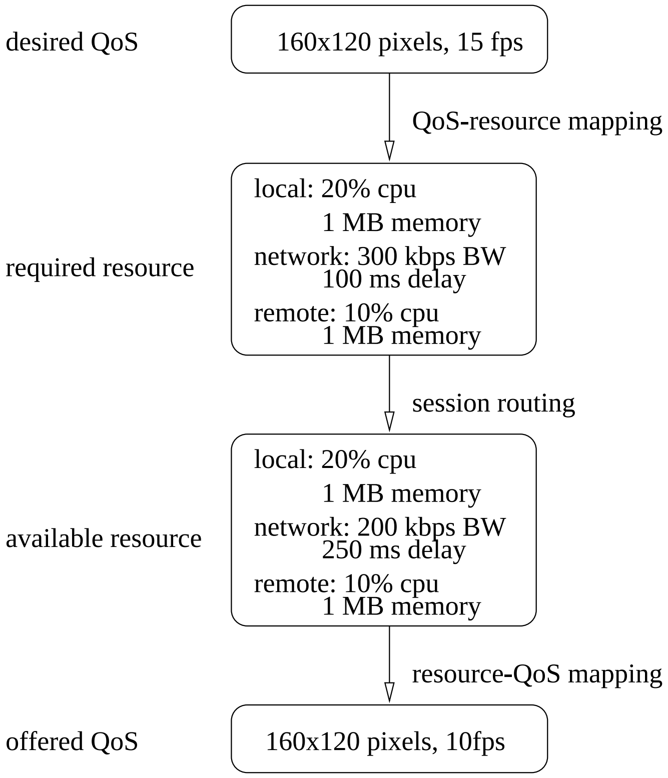


Figure 2.1: *Establishing a video transmission session: a scenario.*

satisfied during the session by appropriate allocation and policing of resource access;

- Resource adaptation: adapt multimedia sessions to dynamic resource changes.

The resource management subsystem includes resource managers (RMs) at the end-systems as well as at the network nodes. Resource management protocols are used to exchange information about resources among resource managers.

---

the reserved resources to provide a better QoS to the session than the guaranteed minimum. Thus the resources allocated to a session may have both reserved and unreserved components.

2.4.1 Establishing a Multimedia Session

When a user requests a new multimedia session, the user first specifies the QoS requirement (QoS specification). The user's specification is then translated to system and network level QoS parameters (QoS translation). The resource managers at the end-systems and network nodes determine the resource requirements of the session, and negotiate to determine whether sufficient resources are available to support the specified quality (admission control). If sufficient resources are available, the session is accepted, and the system reserves the required end-system and network resources to provide service according to the QoS specification.

If the available resources cannot support the specified quality, the system may either reject the request or suggest a compromise of quality.

Figure 2.1 illustrates a scenario where a video transmission session is being established. Suppose the user requests a video session with a desired quality of 160x120 resolution and 15 fps refresh rate. Suppose that this quality requires 20% of the processor cycles and 1 Mbyte of main memory from the local end-system, a network connection of 300 kbps bandwidth and 100 msec end-to-end delay, plus 10% of the processor cycles and 1 Mbyte of main memory from the remote end-system[13]. Suppose that there are sufficient resources at the end-systems, and after routing the session request through the network it is found that the network cannot support a connection quality better than 200 kbps and 250 msec end-to-end delay. Suppose that these network QoS parameters map to a video session with 160x120 resolution and 10 fps refresh rate. Now the user has to decide whether she accepts this quality or not. If the user accepts the quality, the RMs then send the signals to reserve the required resources in all the end-systems and network nodes relevant to this session,

[13]These numbers are adapted from performance results of [38].

and confirm to the user that the session is now established.

2.4.2 Managing Resources During Transmission

After a multimedia session is established, the system has to enforce the given QoS guarantees by appropriate allocation of the resources. The system must ensure appropriate policing of resource usage to avoid unexpected interference among sessions. This involves the management of all end-system and network resources such as process management, memory management and network management.

- Process management: The task of the process manager (scheduler) is to map tasks onto the processor according to a specified scheduling policy, so that all tasks meet their constraints. Several admission and scheduling algorithms are suitable for multimedia tasks. Two of the most relevant are the *earliest deadline first* and *rate monotonic* algorithms [47].

- Memory management: Multimedia applications involve large-scale data movement and real-time deadlines, implying a need for large primary memory and large memory bandwidth. To avoid the prohibitive latency of paging and swapping techniques, multimedia data must sometimes be pinned (locked) to the real main memory. Efficient buffer management techniques such as offset management and scatter/gather systems may also be employed [46].

- Network management: Network management involves scheduling and management of the bandwidth, service time, and buffer space available both at the hosts and the switches (or routers). Since multimedia data requires network resources at certain negotiated rates, the communication protocols must include flow scheduling disciplines which provide users with minimum service rate, independent of other users' traffic characteristics. Combined with a proper admission policy, rate-based scheduling schemes, such as Virtual Clock, Delay Earliest Deadline First and Weighted Fair Queuing, can provide throughput, delay, delay jitter, and loss rate guarantees [35].

2.4.3 Resource Adaptation

Resource availability may change dynamically in a distributed system, and the system should be able to adjust the resource allocation to multimedia sessions in order to make best use of the available resources. For a multimedia system to be adaptive, it has to be able to monitor resource availability of the system and/or the QoS of the sessions.

Adaptation should be triggered in two cases:

- The system does not have sufficient resources to support the current QoS of the sessions, and
- the system has free resources[14], and may be capable of improving the QoS of one or more sessions.

Examples of events which may trigger adaptation include arrival or removal of sessions, user-request for change of QoS, and start or end of a network congestion.

Adaptation may be accomplished in many ways. We give a few examples:

- The resource usage of a session may be changed by adjusting the data rate at the source. For example, if the available bandwidth for a session is doubled, the video refresh rate may be doubled. If the bandwidth is reduced to half, the refresh rate may be reduced to half.

[14]Here free resources mean resources which are not allocated to any session.

- For network related changes such as network congestion or overloads, the adaptation may be accomplished by network mechanisms such as monitoring for overload, load balancing, and dynamic re-routing of connections.

- If the QoS guarantee of the system cannot be maintained, the system may notify the user, and a renegotiation may be initiated to find another QoS contract suitable for the current system state.

2.5 Metaspaces and AMS

In [53], Yokote *et al.* of the Sony Computer Science Laboratory (CSL) proposed the *metaspace*[15] *architecture* for designing a flexible and dynamic operating system called Apertos. Although the first implementation was too inefficient for practical use, the Apertos operating system demonstrated many desirable features such as late binding, dynamic reconfiguration, service specialization on the fly, and run-time object mobility. The metaspace architecture can be characterized by object/metaobject separation, meta-hierarchy and object migration.

Figure 2.2 shows the layered object/metaspace framework used in Apertos. Each application level object is supported by a set of metaobjects, called its *metaspace*. Metaobjects can be shared by different metaspaces. Now each metaobject itself is an object; and therefore needs meta-metaobjects for support. In this way, the object/metaspace relation generates a hierarchical structure called a *metahierarchy*.

[15]Suppose O is an object, and M is another object which implements part of O's execution environment. Then M is called a *metaobject* of object O. The set of metaobjects $\{M\}$ that determines object O's execution environment is called the *metaspace* of object O. We note that meta is a relation between a pair of objects. For example, M may be a metaobject of O, and N may be a metaobject of M.

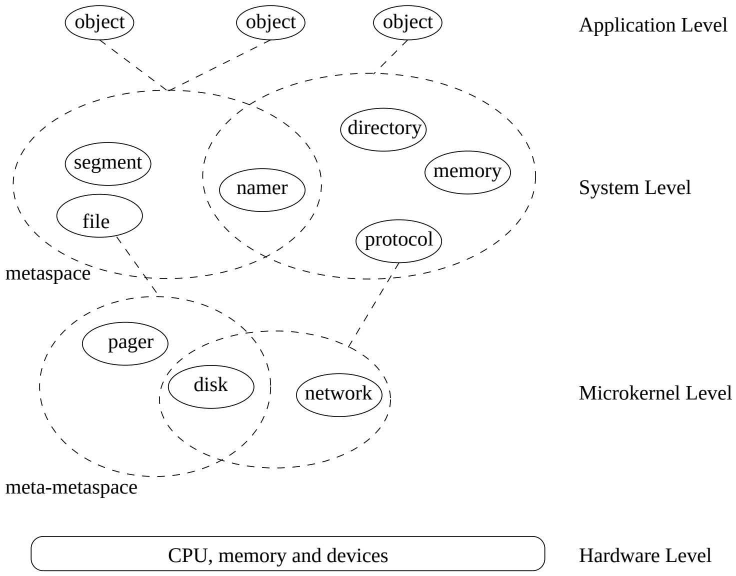


Figure 2.2: *Metaspace architecture for system organization.*

Conceptually, the metahierarchy may extend to infinite depth, but for practical purposes it can be restricted to three levels providing for objects, metaspaces (operating system), and meta-metaspaces (kernel).

The Apertos operating system uses the metaspace architecture for the organization of system components. The system is composed of four levels[16]:

- The *application level* implements the application algorithms, interacts with the users, and may optionally control some of the resource management policies.
- The *system level* provides an interface to the applications and performs low level resource management and adaptation to provide a flexible service interface to the application level.
- The *microkernel level* provides the core (minimum) services to access the hardware resources.

[16]The terminology used in this section is somewhat different from that used in [53], in order to simplify presentation by using standard terms whenever possible.

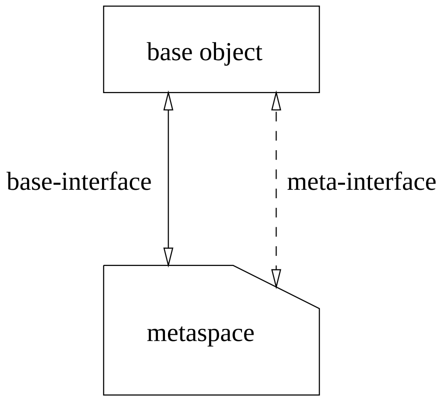


Figure 2.3: *A metaspace system has two interfaces: the base interface for service and the meta-interface for control and adaptation.*

- The *hardware level* consists of system hardware such as CPU, memory, and input/output devices such as media and network devices.

The metaspace architecture provides a systematic approach to adaptation. As shown in Figure 2.3, a metaspace has two interfaces [22]: the base interface for accessing system services and the meta-interface for querying or controlling the behavior of a service or for dynamically changing the implementation of a service. For example, an application program may request allocation of a memory page using the base interface, and the mechanism of page allocation may be controlled/changed dynamically by meta-interface calls. Further discussion of the use of the base- and meta- interfaces is presented in section 5.1.

The metaspace concept was originally proposed by Yokote for the design and implementation of a flexible and reconfigurable operating system. To our knowledge, no other research project has considered using this concept for the management and adaptation of QoS in a dynamic system. We envision that the machinery of quality adaptation in a multimedia system can be neatly encapsulated in metaspaces[17]. The

[17] This possibility was first suggested by Dr Mario Tokoro of Keio University.

metaspace of an object should be a suitable mechanism to monitor performance, take adaptation decisions, and make adjustments for adaptation enforcement. System services can be accessed using the base-interface calls, and quality control may be enforced separately using the meta-interface calls.

2.6 Some Related Projects

In recent years, there has been significant progress in the design of distributed multimedia systems [35, 4, 50, 26, 21]. A very good survey is provided in [2], and Table 2.2 presents a summary of some of the interesting projects.

In [35], Nahrstedt presents an end-to-end QoS architecture called the *Omega Architecture*. This architecture targets multimedia applications with hard real-time requirements, and examines the QoS requirements of the applications. The essence of the Omega Architecture is the reservation and management of end-to-end resources using a Brokerage model which incorporates QoS translation, negotiation, and renegotiation. The resource management activities are managed by a layer-crossing end-system entity, called a Broker. A Broker orchestrates the local resources at different layers, and negotiates with the remote Brokers in order to provide the QoS services to the users.

Campbell *et al.* at University of Lancaster [4] have proposed the *QoS Architecture (QoS-A)* which defines a set of configurable interfaces, services and mechanisms to meet the end-to-end QoS requirements of applications. They have developed a QoS-based transport system, called Multimedia Enhanced Transport System (METS), for an ATM LAN. METS provides a QoS-based service API using three mechanisms: QoS provision, QoS control and QoS management. The user requests a flow with a

Table 2.2: Summary of some related projects.

Projects	Contributions
UPenn Omega	resource management model for end-to-end QoS (Broker model) layered QoS and translation orchestrate local resources and negotiate with remote Brokers resource adaptation using renegotiation implemented in AIX on ATM
Lancaster QoSA	framework for end-to-end QoS management multimedia-enhanced transport system (METS) QoS enforcement by monitoring and adjustment dynamic QoS management using adaptors, filters and groups adaptive video using substream priority (B1, E1 and E2) implemented in Chorus and Linux on ATM
Keio MMP	QoS-Ticket model for QoS adaption on available resources QoS manager allocates resource tickets dynamically CPU reservation using Q-Thread thread package network reservation using ST-II protocol implemented in RT-Mach on ATM
Heidelberg DiME	resource management protocol HeiRAT with multicast support support from ST-II agents for QoS computation multilevel preemptive scheduling regulation by delaying work-ahead packets rate-controlled transport system HeiTS upcall-structure for events and errors media scaling and filtering using substreams implemented in AIX and OS/2 systems on IBM Token Ring
Columbia XRM	extended reference model (XRM) for DMS CORBA-based integration of networking and multimedia (binding) end-to-end QoS with monitoring and adaptation (qStack) simple API for application and transparent QoS management implemented in Solaris on ATM

QoS specification. QoS provision maps the user's specification to system resources, performs admission tests, and allocates resources for the flow. QoS control provides the real-time traffic control of an ongoing flow using techniques such as flow shaping at the source, flow scheduling at the end-system and network nodes, flow policing for overloading and flow synchronization at the receiver. The purpose of QoS management is to sustain the contracted QoS, but it works in a longer time-scale than QoS control. The system tries to maintain the contracted QoS by a monitor-compare-and-adjust loop, and if it fails to maintain the contracted QoS, the user is notified of the degradation.

Recent work at Lancaster has concentrated on ways to support adaptive flows (such as MPEG-coded video flows) in the QoS Architecture [5]. It introduces the concept of *dynamic QoS management (DQM)* which controls and manages multi-layer coded flows operating in a heterogeneous, multicast, distributed multimedia environments. Two techniques are proposed to support scalable video over multimedia networks. These are (1) end-to-end rate shaping which adapts the rates of multilayer flows to the available network resources while minimizing the distortion observed at the receiver, and (2) an adaptive network service which offers a combination of *hard* guarantees to the base layer of the multilayer coded flows, and *fairness* guarantees to the enhancement layers.

Kawachiya *et al.* of Keio University's MMP (MultiMedia Platform) project have proposed a dynamic QoS architecture for multimedia systems with multiple concurrent sessions [19]. This architecture provides mechanisms to communicate and enforce resource-allocation decisions. It is based on a reservation-based resource management model called QoS Ticket [21]. When a CM session is initiated, it registers its QoS requirements with a system server, called the QoS Manager. The QoS

Manager computes the resource requirement, allocates resources for the session, and issues a QoS Ticket to the session. The QoS Ticket contains the resource rights and restrictions of a session, and the session can use it to get preferential service. The resource management required for the QoS Ticket model is handled by the operating system. For the CPU resource, the QoS Ticket model has been implemented on a flexible real-time thread package, called Q-Thread, running on RT-Mach microkernel [20]. For network resources, a reservation based protocol server using the ST-II protocol [48] is being developed.

Wolf *et al.* of IBM European Networking Center at Heidelberg have developed a QoS based transport system called Heidelberg Transport System (HeiTS) [50], based on the ST-II network protocol. The concepts of flow filtering and media scaling for adapting to resource availability were introduced by this project [12].

The Comet group at Columbia University has presented the XRM model and Binding Architecture for the design and management of multimedia networks [26]. It provides an object oriented approach where the issues of user, control, monitoring, management and abstraction are separated into different planes. The design and implementation of a transport layer for meeting QoS guarantees by end-to-end QoS monitoring and adaptation is given in [16].

Our project is based on recent developments in QoS-based system design using resource reservation. However, while most of the above research projects (except Keio MMP) focus on resource reservation, and scheduling and adjustment for individual sessions over a communication network, our primary focus is the general problem of a multisession distributed adaptive multimedia system, where each session specifies a flexible QoS requirement, and the adaptation algorithm adjusts the quality of each session in order to maximize an objective function.

Our project is similar to the Keio MMP project because both consider quality adaptation among multiple concurrent sessions. However the two projects are fundamentally different in their focus and approach:

- The MMP project focuses on the adaptation enforcement and resource management mechanisms, and does not address how the adaptation decisions are to be made. On the other hand, the focus of our project is to understand the issues and factors of an adaptive multimedia system, to devise a mathematical model, and to find a computational algorithm and an architecture for a dynamically adaptive system. Thus the two projects are complementary.

- For our system architecture, we use a metaspace-based hierarchical organization for abstracting the machinery of adaptation, whereas the MMP project uses other abstractions and structures.

2.7 The Knapsack Problems

To help the presentation of Chapter 3 and 4, let us introduce an important class of combinatorial optimization problems, known as the knapsack problems. Many practical problems, such as capital budgeting, industrial production, menu planning, cargo loading, information systems and resource allocation, may be expressed as variants of knapsack problems [32]. For an in-depth discussion of the knapsack problems please refer to [32].

The *classical 0-1 Knapsack Problem (KP)* is posed as follows[18]. Suppose there are n items, and a knapsack. The ith item is worth v_i dollars and weighs r_i kilograms, and the maximum weight the knapsack can carry is R kilograms. The problem is

[18] This definition of the KP is adapted from [9].

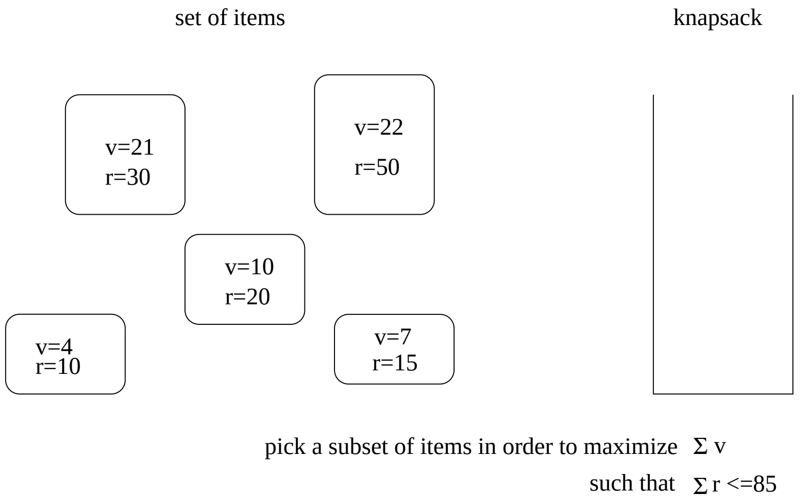


Figure 2.4: *The classical 0-1 knapsack problem.*

to pick a set of items in order to maximize the total value of the pick[19] such that the total weight of the pick does not exceed the capacity of the knapsack.

The classical 0-1 KP can also be expressed as a resource allocation problem where value v_i denotes value (or profit) provided by item i, weight r_i denotes resource required by item i, and R denotes the amount of available resource. Here the problem is to allocate resource to a subset of items in order to maximize the total value such that the total allocated resource does not exceed the available resource. For the rest of the dissertation, we use the resource allocation terminology of the KP and its variants.

Mathematically the problem is stated as follows:

$$V = \text{maximize} \sum_{i=1}^{n} x_i v_i, \tag{KP}$$

[19]The set of picked items are called *the pick* for short.

such that

$$\sum_{i=1}^{n} x_i r_i \le R,$$

$$x_i \in \{0,1\}, \quad i = 1, \dots, n.$$

Here x_i's for $i = 1, 2, \dots, n$ are variables. The problem is called the 0-1 knapsack problem because variable x_i can either take a value of 0 implying item i is not picked, or a value of 1 implying item i is picked. Any pick of items which satisfy the constraint is called a *feasible solution* of the problem. The solution of the 0-1 knapsack problem is the feasible solution which maximizes the sum of the value of the picked items. The classical 0-1 KP is illustrated in Figure 2.4.

If we relax the integrality restriction of the above problem it becomes a linear program which is called the LP relaxation of KP, denoted LP(KP):

$$V = \text{maximize} \sum_{i=1}^{n} x_i v_i, \qquad \text{(LP(KP))}$$

such that

$$\sum_{i=1}^{n} x_i r_i \le R,$$

$$\boxed{0 \le x_i \le 1, \quad i = 1, \dots, n.}$$

Since the constraints of LP(KP) are more relaxed than that of P,

$$V_{LP(KP)} \ge V_{KP}.$$

Thus $V_{LP(KP)}$ may be used as an upper bound [20] for solving KP [10, 32].

[20] Here *upper bound* means a bound which the optimal value of the objective function can never exceed.

The classical 0-1 KP can be generalized for multiple resource constraints (or dimensions). A knapsack problem with multiple constraints is called *multi-constraint knapsack problem* or *multi-dimension knapsack problem (MDKP)*. Suppose there are n items, and m resources. Item i has a value of v_i, and resource required by this item is expressed using resource vector $\mathbf{r}_i = (r_{i1}, r_{i2}, ..., r_{im})$[21]. The amounts of available resources are given by resource vector $\mathbf{R} = (R_1, R_2, ..., R_m)$. Then the following problem is an MDKP:

$$V = \text{maximize} \sum_{i=1}^{n} x_i v_i, \qquad \text{(MDKP)}$$

such that

$$\sum_{i=1}^{n} x_i r_{ik} \leq R_k, \quad k = 1, \ldots, m,$$

$$x_i \in \{0, 1\}, \quad i = 1, \ldots, n.$$

Figure 2.5 illustrates a multi-dimension knapsack problem with $m = 2$.

The variant of knapsack problems most interesting for this work is called the *multiple-choice multi-dimension knapsack problem (MMKP)*, defined as follows. Suppose there are n groups (stacks[22]) of items. The ith group has l_i items. Item j of group i has value v_{ij}, and requires resource $\mathbf{r}_{ij} = (r_{ij1}\ r_{ij2}, ..., r_{ijm})$. The amounts of available resources are given by $\mathbf{R} = (R_1, R_2, ..., R_m)$. The MMKP is to pick exactly one item from each group in order to maximize the total value of the pick, subject to the resource constraints.

---

[21] We use **boldface** symbols to express vector quantities and non-boldface symbols to express scalar quantities.

[22] Here *stack* means *pile of items*, and it is not related to the stack data-structure.

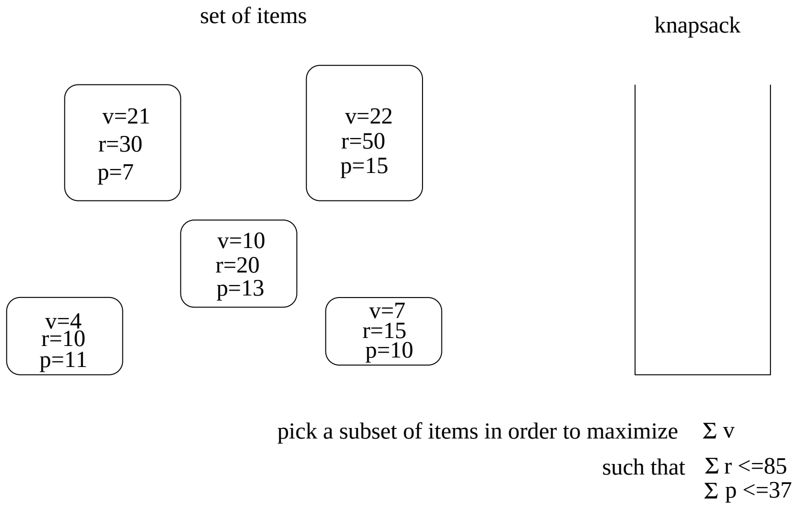


Figure 2.5: *A multi-dimension 0-1 knapsack problem.*

Formally the MMKP is expressed as follows:

$$V = \text{maximize} \sum_{i=1}^{n} \sum_{j=1}^{l_i} x_{ij} v_{ij}, \tag{MMKP}$$

such that

$$\sum_{i=1}^{n} \sum_{j=1}^{l_i} x_{ij} r_{ijk} \leq R_k, \quad k = 1, \ldots, m,$$

$$\sum_{j=1}^{l_i} x_{ij} = 1, \quad i = 1, \ldots, n,$$

$$x_{ij} \in \{0, 1\}, \quad i = 1, \ldots, n; j = 1, \ldots, l_i.$$

Figure 2.6 illustrates a multiple-choice multi-dimension knapsack problem.

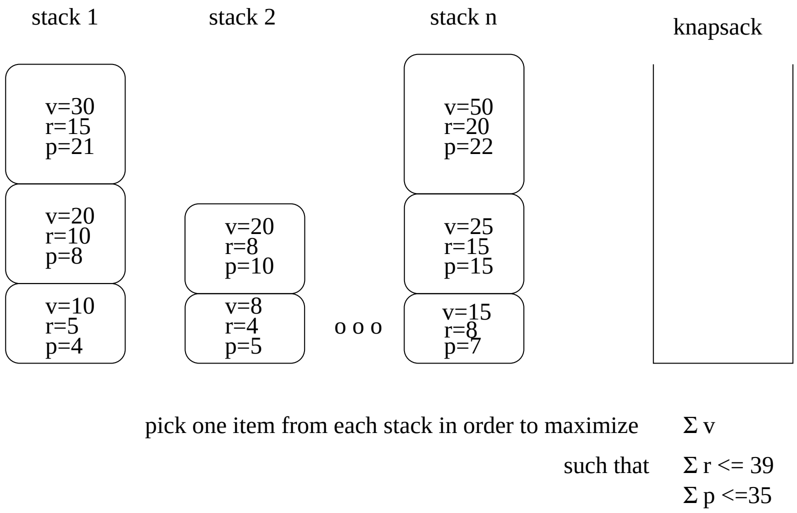


Figure 2.6: *The multiple-choice multi-dimension knapsack problem.*

The special case of MMKP, where there is only one resource constraint (i.e. $m = 1$), is called a *multiple-choice knapsack problem (MCKP).*

Variants of knapsack problems comprise an important class of combinatorial optimization problems. The classical 0-1 knapsack problem is one of the most studied problems in operations research or combinatorial optimization. Please refer to [31] and [7] for good surveys of algorithms for the knapsack problems and some of their variants.

Since the 0-1 knapsack problems are NP-hard [32], the worst-case computation time of the optimal solutions grows exponentially with the size of the problem. For this reason, there are two types of solutions proposed for the KP and its variants: *optimal solutions*, and *near-optimal solutions*. The objective of the near-optimal solutions is to provide solutions which are close to optimal values, but require computation times which are much shorter than those of the optimal solutions.

Most of the optimal algorithms for the variants of the KP uses a general search method, called the *branch and bound method.*

2.8 Branch and Bound Method

Branch and bound is a general and popular method for solving combinatorial optimization problems. In this method the optimal solution is found using iterative generation of a tree, called the *search-tree*. The following discussion of the branch and bound method is based on its application for solving the classical 0-1 KP.

A node in the search-tree represents a solution state where there may be some variables which are known (values are assigned), and some others which are unknown (values are not assigned). A node may be expanded based on a variable whose value is unknown at the current node. For example, expanding a node based on binary variable x_i may generate two nodes: one for $x_i = 0$ and the other for $x_i = 1$. No node will be generated for $x_i = 1$ if it is not feasible (that is picking item i violets the resource constraint). A node which has been generated and whose children have not yet been generated is called a *live* node. The branch and bound method is based on the assumption that we can compute an upper bound of the objective function from the known values at each node.

The branch and bound method uses the following iterative procedure:

1. Start with a single live node where all variables are unknown.

2. Find node e which has the largest upper bound among the live nodes. This node is called the *branching node*, the *expanding node* or simply the *e-node*.

3. If node e does not have any unknown variable (i.e. all the variables are known), then this node represents the optimal solution, and the method terminates.

4. If node e has at least one unknown variable, expand the node on one of the unknown variables. This variable is called the *branching variable*.

5. Go back to step 2.

The branch and bound method finds a global optimum solution [23], and the optimality does not depend on the choice of the branching variables in step 4. However, intelligent choice of the branching variable may potentially reduce the size of the search-tree implying reduction of computation time. Most of the branch and bound algorithms choose the branching variable using some problem-specific heuristic where the objective is to reduce computation time.

2.8.1 A Simple Branch and Bound Example

The general branch and bound method may be explained using a simple example illustrated in Figure 2.7. Here we consider an optimization problem whose objective function has three binary variables x_1, x_2 and x_3.

The branch and bound method starts with one live node (node A) representing all variables as unknown. Suppose the upper bound at this node is 100. Since there is only one live node, it becomes the e-node. Suppose we expand node A based on variable x_1[24]. This generates two more live nodes: node B with upper bound 100 for $x_1 = 0$, and node C with upper bound 90 for $x_1 = 1$.

Next node B becomes the e-node. Expansion of node B on variable x_3 generates node D with upper bound 80, and node E with upper bound 95. Node E becomes the next expanding node. Expansion of node E on variable x_2 generates node F with upper bound 85, and node G with upper bound 92.

[23] An informal proof of the optimality of the branch and bound method is given in section 2.8.2.

[24] For this example, we have chosen an arbitrary sequence (1 3 2) to choose the branching variables.

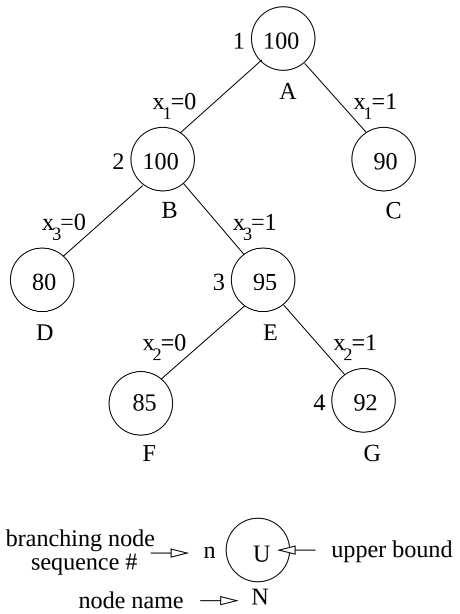


Figure 2.7: *An example search-tree for the branch and bound method.*

At this point, node G becomes the e-node. We note that at node G, since all the variables are already decided (i.e. there is no unknown variable), the upper bound must be equal to the value of the objective function. Since this node is chosen as e-node, it must have the largest upper bound among all the live nodes. It implies that expanding no other live node can provide an objective value which is larger than that already achieved by node G. Thus node G yields the optimal solution of the current optimization problem.

This example illustrates that branch and bound employs an intelligent enumeration technique [41]: enumeration because it performs an exhaustive search in the solution space, and intelligent because it performs this exhaustive search without generating all the nodes. For instance, in our example the optimal solution has been found without expanding live nodes C and D.

2.8.2 Optimality of the Branch and Bound Method

The branch and bound method finds a global optimum solution subject to the condition that an upper bound can be computed at each node of the search-tree. Let us give an informal proof here.

We observe two properties of the branch and bound search method:

1. Once a node becomes a live node, it remains a live node until it is expanded, and every time an e-node is chosen all the live nodes are considered.

2. If T denotes a feasible solution, then T is either a live node, or it can be reached by further expansion of one of the live nodes. This is true at the start because the method starts with only one live node where all variables are unknown. This property is maintained every time a node is expanded because after expansion of an e-node e, all the feasible descendants of node e become live nodes.

To prove the optimality of the branch and bound method by contradiction, suppose it fails to find a global optimum. Suppose in a maximization problem, node X is the solution found by the branch and bound method, and there exists another feasible solution Y with $V(Y) > V(X)$ where V(Y) and V(X) denotes the values of the objective function for feasible solutions Y and X, respectively. Since there is no unknown variable at X, we can assume that $U(X)=V(X)$ where U(X) denotes the upper bound of the objective function at node X. Thus $V(Y) > U(X)$.

Now due to properties 1 and 2, when node X is chosen for expansion, there existed a live node Z where either Y=Z or node Y could be reached by further expansion of node Z. According to the definition of upper bound, $U(Z) \geq V(Y)$. Thus $U(Z) > U(X)$. This contradicts with the choice of X as e-node when node Z is also a live node, and the proof is complete.

2.8.3 Branch and Bound Algorithms for the Variants of KP

Kolesar gave the first branch-and-bound algorithm for the classical 0-1 KP [23]. This algorithm uses a greedy-like strategy where at any e-node, it branches on the not-yet-decided item which provides a highest value per unit of required resource (v_i/r_i). This algorithm uses the solution to the LP relaxation of KP, that is LP(KP), for the computation of upper bound. It is to be noted that a simple greedy method solves the single-constraint LP(KP) optimally.

Shih presented a branch-and-bound algorithm for the MDKP [45]. For upper bound estimation, this algorithm treats the MDKP problem as m single-constraint KPs, and calculates the optimal value of the objective function in each case. The minimum of these objective function values is then used as an upper-bound.

Branch and bound algorithms for the MCKP were proposed by Nauss [37], Armstrong *et al.* [1] and Dudzinski *et al.* [13].

Even though branch and bound algorithms have been proposed for many variants of KP, no such work is found on the MMKP. In Chapter 4, we propose a branch and bound algorithm for the MMKP.

2.9 Near-optimal Solutions

For the classical 0-1 KP, a greedy approach to get a near-optimal solution to the problem is as follows: pick the item with the largest value of v_i/r_i, then pick the item with the second largest value of v_i/r_i, and so on, until no more item can be picked either because the available resource is not enough or because no item is left. For certain classes of knapsack problems a greedy algorithm is indeed optimal. Magazine *et al.* describes some of these classes in [29].

Now if one wants to apply the greedy method to the MDKP, it is not very clear how this approach can be generalized for multiple resource constraints. Toyoda proposed a simple solution to this problem using the concept of a scalar resource index, called the *aggregate resource* [49]. Here the main idea is to penalize the not-yet-picked items depending on the current resource state. Suppose in a m-resource MDKP instance, the current resource usage vector is given as $\mathbf{C} = (C_1, C_2, \ldots, C_m)$, and resource requirement of item i is given by $\mathbf{r}_i = (r_1, r_2, \cdots\ r_m)$. Then the aggregate resource required by item i is computed as follows:

$$a_i = (r_1 C_1 + r_2 C_2 + \cdots + r_m C_m)/|\mathbf{C}| = \frac{\mathbf{r}_i \bullet \mathbf{C}}{|\mathbf{C}|},$$

where $|\mathbf{C}|$ denotes the magnitude of vector $\mathbf{C}$, and $\bullet$ denotes dot product of vectors. Geometrically a_i is the projection of $\mathbf{r}_i$ on $\mathbf{C}$, as illustrated in Figure 2.8.

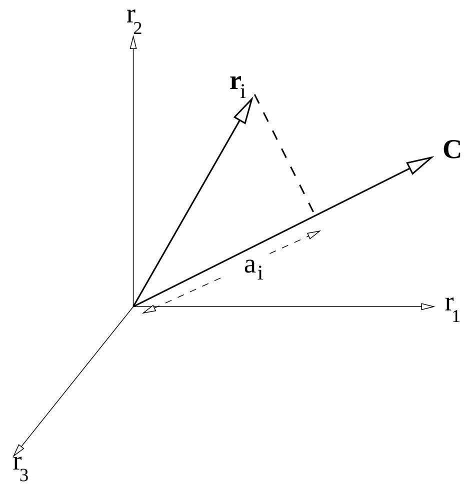


Figure 2.8: *Aggregate resource is a projection of resource vector on the current resource usage vector.*

For example, consider an MDKP with two resources. Item 1 requires resource $\mathbf{r}_1 = (0.6, 0.4)$ and item 2 requires $\mathbf{r}_2 = (0.3, 0.7)$. The current resource usage vector is given as $\mathbf{C} = (0.7, 0.3)$. Then the aggregate resource required by each of the two items is obtained as:

$$|\mathbf{C}| = \sqrt{0.7^2 + 0.3^2} = 0.76,$$

$$a_1 = (0.6, 0.4) \bullet (0.7, 0.3)/0.76 = (0.42 + 0.28)/0.76 = 0.92, \text{and}$$

$$a_2 = (0.3, 0.7) \bullet (0.7, 0.3)/0.76 = (0.21 + 0.21)/0.76 = 0.55.$$

We note that the computation of aggregate resource uses the elements of $\mathbf{C}$ as penalty factors. In the above example, a penalty factor of 0.7 per unit usage of resource 1, and a penalty factor of 0.3 per unit usage of resource 2 are applied. Thus it applies a large penalty for a heavily used resource, and a small penalty for a lightly used resource.

Toyoda's heuristic for MDKP starts with no items, and adds one item at a time iteratively as long as the solution is feasible. At any iteration, the heuristic picks the item which provides the maximum value per unit of aggregate resource (v_i/a_i) among the not-yet-picked items. In the above example, suppose item 1 has value 3, and item 2 has value 2, then the value per unit of aggregate resource is computed as follows:

$v_1/a_1 = 3/0.92 = 3.26$, and $v_2/a_2 = 2/0.55 = 3.63$.

Clearly the heuristic will prefer to pick item 2 to item 1. Computational experiments show that this heuristic provides very good near-optimal solutions to the MDKP using very short computation time, and it can be applied to large problem instances, such as problems with more than a thousand variables [49].

The MMKP is one of the harder variants of the 0-1 knapsack problem [34]. It is a combination of two well-known variants of the knapsack problems: the multi-dimensional knapsack problem (MDKP) and the multiple-choice knapsack problem

(MCKP). Even though the standard 0-1 knapsack problem or its MDKP and MCKP variants are well-studied problems, a recent heuristic by Moser [34] seems to be the only solution available on the MMKP.

In [34], Moser generalizes the Lagrange Multiplier based heuristic for the MDKP proposed by Magazine and Oguz [30]. This work also analyzed the computational complexity of the heuristic, and reported computational experiments. However, we found some problem instances where the heuristic failed to achieve any feasible solution, even though there existed one or more feasible solution(s).

In Chapter 4, we propose a heuristic to provide a near-optimal solution to the MMKP. Our solution uses the concept of aggregate resource proposed by Toyoda.

Chapter 3

The Utility Model for Adaptive Multimedia Systems

In this chapter, we present the *Utility Model* — a mathematical model to capture the dynamics of adaptive multimedia systems (AMSs). It is based on the concepts of quality profile, quality-resource mapping, session and system utility, and of system resource constraints. The Utility Model formulates the adaptive multimedia problem as a multiple-choice multi-dimension 0-1 knapsack problem. This model provides a unified and computationally feasible approach to make session admission, quality adaptation and resource allocation decisions within a multisession multimedia system.

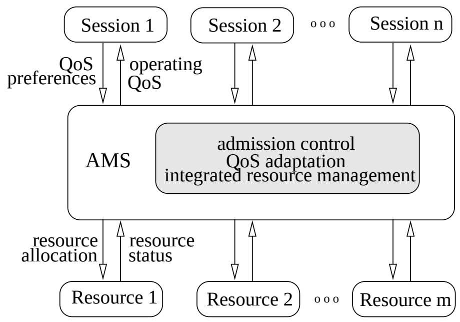


Figure 3.1: *Adaptive multimedia system requirements.*

3.1 AMS Requirements

The quality perceived by the user of a multimedia session depends on the quality of individual media components, and this in turn depends on such things as video refresh rate, number of sound channels (mono, stereo or surround), and the relation among media components such as video-audio synchronization. In an AMS, the QoS of individual sessions is dynamically adapted in order to maximize some objective function (e.g. total revenue earned), also taking into account the users' preferences and the system resource status.

Let us consider an adaptive multimedia system with n sessions as shown in Figure 3.1. Some of the requirements of this system are as follows:

- The user of a session should be able to specify an adaptive quality requirement or preference. This is done using a *quality profile* ($\mathbf{P}_i$) as defined below.

- The system should dynamically adapt the *operating quality* ($\mathbf{q}_i$) of each session based on the quality profiles of the sessions and the system resource status. To do this, the system must know the dynamic resource status of the system.

3.2 The Utility Model

The Utility Model is a mathematical model for AMS based on the concepts of quality profile, quality-resource mapping, session and system utility, and of system resource constraints. These concepts are explained in the following subsections.

3.2.1 Quality Profile

The quality profile specifies the quality preferences of the user of a particular session. For simplicity, we consider three media: audio (a), video (v) and image (i). The *operating quality* $\mathbf{q}_i$ of session i is the vector (3-tuple) of individual media qualities,

$$\mathbf{q}_i = (q_{ia}, q_{iv}, q_{ii}). \tag{3.1}$$

The *quality profile* of a session is a sequence of acceptable operating qualities in increasing order of preference (from minimum acceptable quality to highest desired quality)[1]. Mathematically the quality profile of session i can be expressed as a vector[2],

$$\mathbf{P}_i = (\mathbf{q}_{i1}, \mathbf{q}_{i2}, \mathbf{q}_{i3}, \ldots, \mathbf{q}_{il_i}), \tag{3.2}$$

[1]Here the order of preference is chosen by the user.

[2]When a scalar index is available for operating quality, it may be possible to express the quality profile as a continuous range by specifying the lower and upper limit (that is minimum acceptable quality and maximum desired quality). For example, the effective bandwidth of an application may be used as a quality index. However in this dissertation, we focus our attention on discrete profile set for two reasons: (1) Discrete profile is suitable for expressing QoS in a qualitative way, which is more natural for the users. For instance, audio may have one of the following qualities: mono, stereo or surround sound, where the qualities are discrete valued quantities. (2) From a technological point of view, varying media quality in a continuous range is more difficult than varying it in discrete coarse-grained steps [12].

where l_i is the number of qualities in $\mathbf{P}_i$, $\mathbf{q}_{i1}$ is the minimum acceptable quality and $\mathbf{q}_{il_i}$ is the highest desired quality. For example, Figure 3.2 shows a session with quality profile $\mathbf{P}_i = (\mathbf{q}_{i1}, \mathbf{q}_{i2}, \mathbf{q}_{i3}, \mathbf{q}_{i4})$ where $\mathbf{q}_{i1}$ indicates low-resolution video and mono audio, $\mathbf{q}_{i2}$ indicates low-resolution video and stereo audio, $\mathbf{q}_{i3}$ indicates medium-resolution video and stereo audio, and $\mathbf{q}_{i4}$ indicates high-resolution video and surround sound audio.

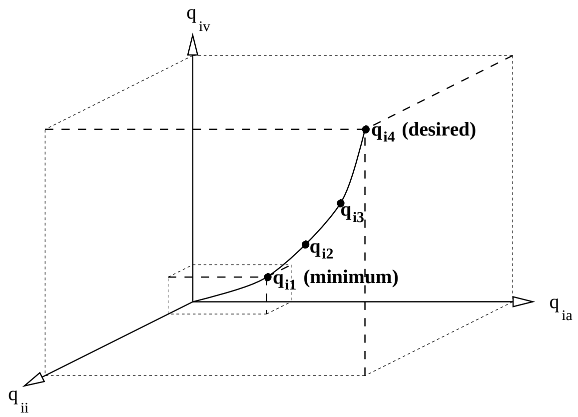


Figure 3.2: *Quality profile of a session.*

3.2.2 Quality-resource Mapping

We assume the existence of a mapping from an operating quality to the resources required to provide that quality. This is called the *quality-resource mapping*. Quality-resource mapping is addressed in research projects such as at Keio University and at Washington University at St. Louis [38, 15].

In this model, we assume that a media QoS may be mapped uniquely to the required resources[3]. We note that *unique* does not imply static. Let us assume

[3]However, in practice, quality-resource mappings may not be unique. For example, if we use a simple compression algorithm a certain perceived QoS may require 10% of the total CPU cycles and 3 Mbps of network bandwidth. On the other

only three resources within a system: processor cycles (p), main memory (m), and network bandwidth (b). Then the required resources $\mathbf{r}_i$ of session i can be expressed by a three-tuple,

$$\mathbf{r}_i = (p_i, m_i, b_i) \tag{3.3}$$

where

$$\begin{aligned} p_i &= p(\mathbf{q}_i) = p(q_{ia}) + p(q_{iv}) + p(q_{ii}), \\ m_i &= m(\mathbf{q}_i) = m(q_{ia}) + m(q_{iv}) + m(q_{ii}), \\ b_i &= b(\mathbf{q}_i) = b(q_{ia}) + b(q_{iv}) + b(q_{ii}). \end{aligned}$$

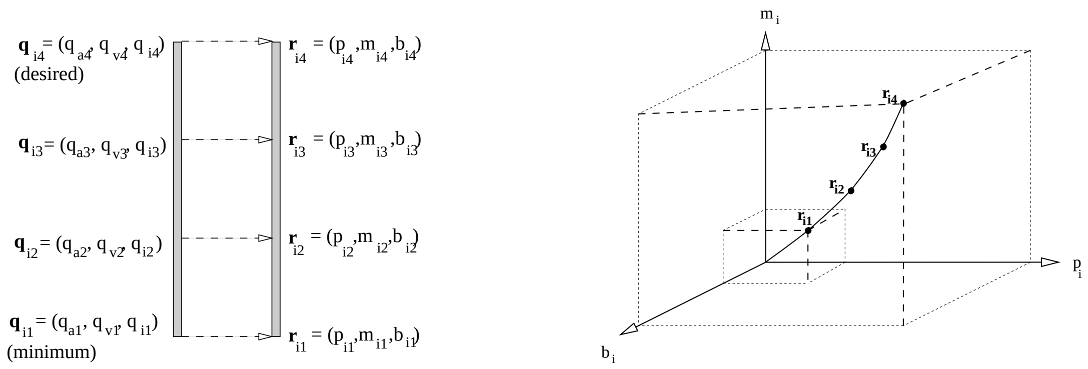
Here $p(\cdot)$, $m(\cdot)$ and $b(\cdot)$ are quality-to-resource mapping operators for processor, memory and bandwidth resources respectively. In vector notation, the quality-resource mapping can be expressed as

$$\mathbf{r}_i = \mathbf{r}(\mathbf{q}_i). \tag{3.4}$$

The quality-resource mapping is illustrated in Figure 3.3. Figure 3.3 (a) shows how each operating quality of the quality profile is mapped to a resource vector. For example, operating quality $\mathbf{q}_{i1}$ would require resource vector $\mathbf{r}_{i1}$, i.e. $\mathbf{r}_{i1} = \mathbf{r}(\mathbf{q}_{i1})$. The quality-resource mapping transforms the quality profile to the session resource profile, which is a list of required resources for different choices of operating qualities. The resulting *resource profile* is illustrated in Figure 3.3 (b). Intuitively, one may expect that better quality implies more resources. However, practically, the resource

hand, if we use a more sophisticated compression algorithm, the same perceived QoS may require 40% of CPU cycles and 1 Mbps of network bandwidth. We note the processor-bandwidth trade-off depending on the choice of compression algorithm. In such cases, we may achieve uniqueness by fixing the choice of compression algorithm, or by arbitrarily choosing one of several resource values.

profile may not always show this monotonic behavior. For example, it is possible that a video transmission system improves quality by using more network bandwidth but fewer CPU cycles.



(a) Quality-resource mapping. (b) Resource profile.

Figure 3.3: *Mapping operating quality to required resources, and resource profile.*

3.2.3 Session and System Utility

Any adaptive system must have an objective, which decides the direction of adaptation in any given situation. We assume that the objective of an AMS can be expressed as a *system utility objective*[4], and session i's utility can be obtained from its operating quality $\mathbf{q}_i$ by using the *session utility function* $u_i(\cdot)$.

For simplicity, we assume that the system utility objective is to maximize the system utility function U given by

$$U = \sum_{i=1}^{n} u_i(\mathbf{q}_i), \tag{3.5}$$

[4]The concept 'utility' is taken from economics, where it means the capacity of a commodity or a service to satisfy some human want, and the goal of an economic system is to allocate resources to maximize utility.

a linear function of the session utilities. We note that the Utility Model is not restricted to the above system utility function. More generally, the system utility function defines the objective of the adaptive system, and may be derived from a set of resource usage policies based on issues such as revenue, fairness and priority. For example, for a multimedia service provider, a natural choice for the system utility objective would be to maximize revenue, where each session pays for the service it receives; for a friendly research lab, the goal might be to provide fair share of resources to the sessions; whereas for a military system, the goal may be to provide a strict priority-based scheme where the generals' requests receive the highest priority and the private soldiers' requests receive the lowest priority.

3.2.4 System Resource Constraints

At any moment, the system state has to obey the following *system resource constraints*: for each resource, the sum of the quantities of the resource allocated to all the sessions cannot exceed the total available quantity of the resource. Suppose the available system resources are expressed as a vector $\mathbf{R} = (P, M, B)$. The resource constraints are expressed as

$$\sum_{i=1}^{n} \mathbf{r}(\mathbf{q}_i) \leq \mathbf{R}. \tag{3.6}$$

3.3 The Adaptive Multimedia Problem

The main concepts of the Utility Model are illustrated in Figure 3.4:

1. Each user specifies a quality profile which is the set of acceptable operating qualities for the session.

2. A session's operating qualities are assumed to be mapped uniquely to required resources.

3. The system is subject to the system resource constraints.

4. A session's operating qualities are assumed to be mapped uniquely to session utilities.

5. The system utility is the sum of all session utilities.

Now the AMP is to *find* the operating quality $\mathbf{q}_i$ of each session i which *maximizes* the system utility U under the system *resource constraints*.

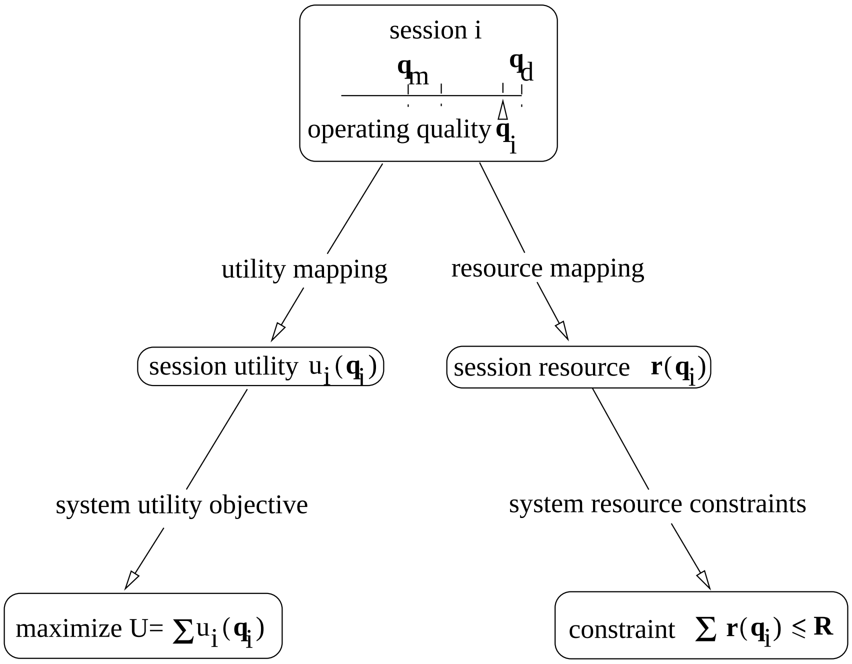


Figure 3.4: *The main concepts of the Utility Model.*

3.3.1 AMP as a Knapsack Problem

In this subsection, we show a mapping of AMP to a multiple-choice multi-dimension 0-1 knapsack problem (MMKP). In section 2.5, we defined the MMKP as a resource allocation problem as follows. Suppose there are n groups (stacks) of items. The ith group has l_i items. Item j of group i has value v_{ij}, and requires resource $\mathbf{r}_{ij} = (r_{ij1}, r_{ij2}\ ..., r_{ijm})$. The amounts of available resources are given by $\mathbf{R} = (R_1, R_2, ..., R_m)$. The MMKP is to pick exactly one item from each group in order to maximize the total value of the pick, subject to the resource constraints. Formally:

$$V = \text{maximize} \sum_{i=1}^{n} \sum_{j=1}^{l_i} x_{ij} v_{ij},$$

$$\text{such that}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{l_i} x_{ij} r_{ijk} \leq R_k, \quad k = 1, \ldots, m,$$

$$\sum_{j=1}^{l_i} x_{ij} = 1, \quad i = 1, \ldots, n,$$

$$x_{ij} \in \{0, 1\}, \quad i = 1, \ldots, n; j = 1, \ldots, l_i.$$

The AMP can be viewed as an MMKP as follows.

- The quality profile $\mathbf{P}_i = (\mathbf{q}_{i1}, \mathbf{q}_{i2}, \mathbf{q}_{i3}, \ldots, \mathbf{q}_{il_i})$ can be viewed as a stack of l_i items. Thus n quality profiles map to n stacks of items, and item j of stack i denotes operating quality $\mathbf{q}_{ij}$ of session i.

- When session i operates at quality $\mathbf{q}_{ij}$, session utility u_{ij} is denoted as value

v_{ij}, and the amount of resource k consumed is denoted as r_{ijk} (quality-resource mapping).

- The system utility U of AMP is mapped to the value of the pick V.
- The resource constraints of AMP are equivalent to the resource constraints of the MMKP.
- Finding an operating quality for each session can be viewed as picking (exactly) one item from each stack.

Figure 3.5 illustrates the mapping with two system resources: CPU time p and main memory q.

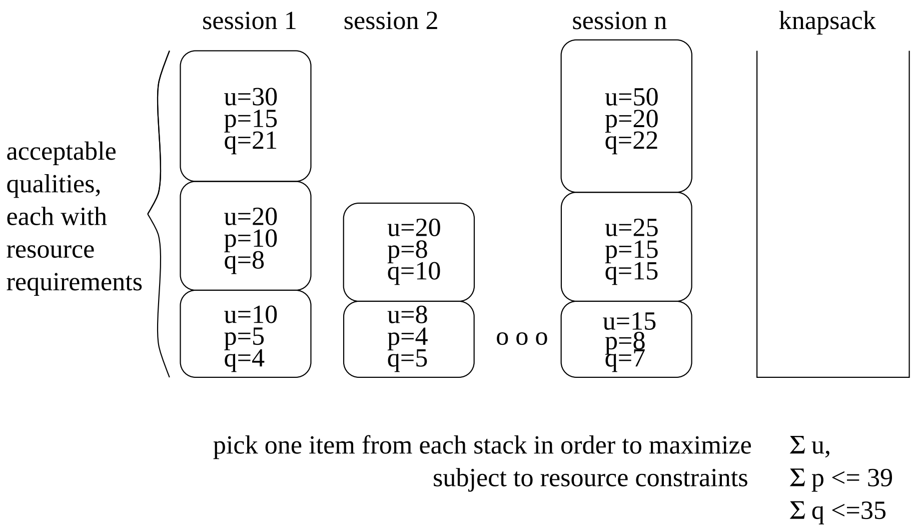


Figure 3.5: *AMP as a multiple-choice multi-dimension 0-1 knapsack problem (MMKP).*

3.4 Admission Control of New Sessions

When a user wants to start a new multimedia session, the system should admit the session if and only if it has the available resources to support the user's minimum quality requirement. This is called *admission control.* Admission control is necessary for service with quality guarantees because the system has to ensure that enough resources are available at run-time to meet the minimum quality guarantee. Mere fairness is not sufficient.

Suppose the system currently has n sessions, and the current system utility is U_n. Now when a user requests a new session with a quality profile, the admission decision requires two steps:

1. The system must check whether any feasible solution of the AMP exists, that is whether there exists any solution where the $n+1$ sessions can share the currently available resources. If such a solution does not exist the new session must be rejected.

2. If there exist one or more feasible solution(s) with $n+1$ sessions, suppose the maximum system utility any feasible solution may achieve is U_{n+1}. If $U_{n+1} \leq U_n$, the new session should be rejected as unprofitable (i.e. no increase in system utility). Otherwise the session should be accepted.

When a user is admitted, the system must reserve the resources required for guaranteeing at least the minimum operating quality. We note that if the user of a session wants to increase the minimum acceptable quality on the fly, another admission control process will be required to change the reserved resources. However, if the user changes her quality profile without increasing (some components of) the minimum quality, the reservation does not have to be affected.

3.5 Applications

Although the Utility Model is proposed to capture the dynamics within adaptive multimedia systems, the concepts developed are general, and may potentially be applied to many diverse applications. Some examples follow.

End-System Architecture: The concepts of the Utility Model may be applied in the design of end-system architecture for adaptive multimedia systems. We present one such architecture in Chapter 5. This architecture uses the Utility Model to provide an integrated and adaptive management of system resources.

Multimedia Service Provider: For a multimedia service provider, suppose the objective of the system is to maximize revenue. Consider a system with n current sessions, and m resources. Session i has an operating quality $\mathbf{q}_i$ and requires resource $\mathbf{r}_i = r(\mathbf{q}_i)$. If session i's utility $u_i(\mathbf{q}_i)$ is interpreted as the revenue paid by (the bill of) the session in return for the resources of resource vector $\mathbf{r}_i$), then maximizing the utility function

$$U = \sum_{i=1}^{n} u_i(\mathbf{q}_i)$$

under the resource constraint

$$\sum_{i=1}^{n} \mathbf{r}_i \leq \mathbf{R}$$

implies maximization of total revenue which the system could earn by allocating its resources.

The Utility Model may also be used to maximize the profit of a service provider. Consider a system with n current sessions, and m resources. Session i has an operating quality $\mathbf{q}_i$ and requires resource $\mathbf{r}_i = \mathbf{r}(\mathbf{q}_i)$. System profit can be found as total revenue minus total cost. Thus system profit,

$$P = \sum_{i}^{n} u(\mathbf{q}_i) - \sum_{i}^{n} c(\mathbf{r}_i),$$

where $u(\mathbf{q}_i)$ denotes the revenue earned from user i for operating quality $\mathbf{q}_i$, and $c(\mathbf{r}_i)$ denotes the cost for resource $\mathbf{r}_i$.

Suppose for each resource k, the cost for a session can be found from the cost per unit of allocated resource k. Then for session i, the session cost $c(\mathbf{r}_i)$ can be uniquely found from the session operating quality $\mathbf{q}_i$ using the resource mapping $\mathbf{r}_i = r(\mathbf{q}_i)$ and cost per unit of each allocated resource. Suppose $u'(\mathbf{q}_i)$ denotes this session quality-cost mapping. Then system profit can be found as,

$$P = \sum_{i}^{n} u(\mathbf{q}_i) - \sum_{i}^{n} u'(\mathbf{q}_i) = \sum_{i}^{n} u''(\mathbf{q}_i),$$

where $u''(\mathbf{q}_i) = u(\mathbf{q}_i) - u'(\mathbf{q}_i)$ denotes system profit from session i for operating quality $\mathbf{q}_i$. We note that, here $u(\mathbf{q}_i)$ maps a session quality to the user's bill (session-bill) which can be seen as part of the user-system contract. On the other hand, function $u''(\mathbf{q}_i)$ maps session quality to session-profit which is derived from the session-bill and session-cost.

Under the above assumptions, the problem of finding the quality of each session in order to maximize system profit of the multimedia service provider subject to system resource constraints can be mapped to an MMKP. Once again, the Utility Model may be used both for the session admission and quality adaptation in such a system.

Adaptive Virtual Environments: The Utility Model may be used for designing adaptive distributed virtual environments (DVEs). A DVE is a three dimensional setting where multiple users can interact with the setting and among themselves [3]. Usually the users are represented by avatars. An avatar may move from one point to another, and may communicate with other avatars. For example, consider a virtual theme park where avatars move around and participate in the activities. Applications of DVEs include simulated training, distance learning, and entertainment.

Let us consider a client-server structure of a DVE. Each avatar is represented by a client, and the DVE is represented by the server. The clients and servers exchange the dynamic state information so that both the DVE and each avatars view of the DVE is consistent. For example, when an avatar opens a door, all of the other avatars who have an unobstructed view should see it open [3]. However, an avatar who is too far away from the door or whose view is obstructed by a wall should not see this state.

Ideally the clients and server should always have a consistent view of the DVE. However in a distributed environment this will be too expensive in terms of resource requirement for exchanging and processing of the high volume of information this would require. Practically the level of consistency a DVE can support will be limited by the availability of resource. Now how should a DVE behave in a dynamic resource environment?

The Utility Model may be used for adapting the consistency level within a DVE. For example, when enough bandwidth is available avatars may send facial expressions updates as well as position updates. However when the network is congested, the avatars may only send position updates. An architecture for scaling virtual worlds is suggested in [28].

Dynamic Markets: The Utility Model may be used in single-seller multiple-buyer *dynamic markets* where each buyer provides a set of choices, and the seller finds the appropriate choice for each buyer in order to optimize the seller's objective.

For example, consider a dynamic airline ticketing system. Suppose an airline provides three kinds of services: s choices for seats, m choices for meals, and d choices for drinks. The airline sells tickets of two types:

- Dynamic Tickets: A dynamic client provides a ticket profile with l composite choices. Each composite choice represents a ticket configuration with a choice for seat, meal and drink. The client also puts a price offer for each of this composite choice. Once a ticket profile is admitted (accepted), the airline must guarantee that the client will get a ticket chosen from her l choices.
- Static Tickets: The airline provides a desired price for each choice. Any client can buy any service element at this desired price on a 'first come first serve' (FCFS) basis. This is like the traditional way of buying tickets where a client rests assured after she has bought a ticket.

Now suppose the plane is ready to fly. All the passengers are checked in, and foods and drinks are already loaded. Ticket profiles of the dynamic clients, ticket configurations of the static clients and cancellations are known. The problem of the airline is to find the appropriate ticket configurations for all the clients of this flight in order to maximize the total revenue. We note that the Utility Model may be used to represent this problem.

In a dynamic airline ticketing system, the incentive for a dynamic client is to get a upgraded ticket at lower price than the price of a static ticket with same configuration, and the incentive for the airline is to maximize revenue by selling last moment ticket upgrades. This system seem to be particularly suitable for future web-based ticketing systems where the automated systems will replace the traditional travel agencies.

It will be interesting to see whether the concepts of the Utility Model may be extended for use in dynamic markets with multiple sellers and buyers.

3.6 Usage Issues

3.6.1 User-System Protocol

The quality profile may be interpreted as an adaptive service contract between the system and the session. Let us consider the multimedia service provider with revenue maximization objective. From the session's side, a quality profile as given by expression 3.2 means that the minimum quality the session is ready to accept is $\mathbf{q}_{i1}$; that the best quality it desires and is willing to pay for is $\mathbf{q}_{il}$; and that it is ready to accept and pay for any quality from within the quality profile set. From the system's side, when such a session is admitted, the system has to guarantee the resources such that the session's operating quality never falls bellow $\mathbf{q}_{i1}$, and depending on resource availability it can provide the session with any quality as long as it is within the quality profile set.

The user side:

- When a user requests a new session, she specifies a quality profile which is a list of acceptable qualities arranged from minimum acceptable quality to maximum desired quality.

- The user specifies the utility function which maps any acceptable quality to the amount she pays[5]. The user pays more for better quality but not beyond the maximum desired quality.
- The user may change the quality profile, but it becomes effective only after the system accepts it.
- When the user is done with the session, she requests the system to terminate the session.

The system side:

- The system decides whether to admit a new session or to reject it. The system may not admit a session unless it increases the system utility.
- Once a session is admitted, the system must provide at least the minimum operating quality during the life-time of the session. This requires reservation of resources for the session.
- The system may change the operating quality of a session as long as it is within the quality profile, without prior notice. However the system should inform the session after it changed the operating quality.
- The system cannot drop any existing session in order to maximize revenue.
- When the session changes its quality profile, the system decides whether to accept the new profile or not. The system may not accept it unless it increases the system revenue.

[5] How such a mapping can be obtained is discussed later in section 3.6.4

- When the system receives a termination request from a session, the system releases the resources allocated to this session.

3.6.2 Getting the Quality Profile

We assume that application programs are QoS-transparent in order to simplify the application design and allow the use of legacy applications in QoS-based environments. The actual quality of the media is decided at run-time by the system with a possibility of user (not application program) control. The user can specify her quality requirements using a quality profile at run-time.

The user must be able to specify her quality profile which expresses her QoS requirements. This can be achieved using a static table of acceptable qualities, or it may be automatically derived by profiling her usage history. For instance, Table 3.1 illustrates a simple quality profile for a session. It has only three discrete qualities: Bronze, Silver and Gold. The session's minimum acceptable quality is Bronze, and the maximum desired quality is Gold.

Quality	Bronze	Silver	Gold
Audio	mono	stereo	surround
Video	low resolution no color	high resolution no color	high resolution color
Image	low resolution	medium resolution	high resolution

Table 3.1: *A simple table for a quality profile.*

An application may have a default quality profile which may be loaded at the initial setup time. However, the user should be able to control the profiles according to her choice at run-time. This may be achieved using a GUI-based interface where the quality profile is generated based on the user's set of media choices and the relative importance of each medium.

3.6.3 Quality-resource Mapping

The Utility Model assumes the existence of an operating-quality to required-resource mapping. However, how such a mapping can be obtained is a research issue being addressed in projects such as at Keio University and at Washington University at St. Louis [38, 15].

The quality-resource mapping may be obtained using off-line experimental evaluation. These evaluations are usually platform (both hardware and software) dependent, and proper scaling is necessary for use in different platforms. In [38], Nishio *et al.* present a quality-resource table based on measurement of their RT-Mach system. It is desirable to have the quality-resource mapping available as empirical models which can be used to compute required resources from the desired QoS parameters.

3.6.4 Quality-utility Mapping

Mapping an operating quality to a utility value is a nontrivial problem. Should the user provide this mapping? Or should it be provided by the system? There are no easy answers to these questions which may be very much dependent on the application.

For example, in our service provider example, the session utility represents the user's bill, and the system utility represents the system revenue. In this case, when the user specifies the QoS requirements using the quality profile, she could also offer a dollar value she is ready to pay for each operating quality. The system then uses these values to maximize its revenue both for admission control and run-time quality adaptation.

Implementation of the above strategy however does not imply that the users have to be directly involved in quality-utility mapping. For example, an intelligent utility agent may be used to derive such a mapping for the user dynamically. The utility agent may also be aided by system default setups, empirical modeling based on off-line measurements, and dynamic learning. The system can also help the utility agent by providing hints on dynamic price based on the current demand-supply state. However, since this mapping represents money coming out of the user's pocket, the user should have the final decision.

3.6.5 Resource Allocation Policy and the System Utility

The quality profile contract between the sessions and the system requires a reservation based resource management scheme. The system reserves resources to ensure at least the minimum quality to each session. At any time, part of the system resources are reserved, and it is the remaining unreserved resources which the system can allocate in order to provide adaptive quality to the sessions. In the following we discuss three resource allocation policies: revenue or profit maximization, fair share, and priority, and investigate their implications for the system utility objective.

- *Revenue or Profit Maximization:* Revenue or profit maximization is discussed in section 3.5. We note that, under this policy, the system tends to provide more resources to sessions which are prepared to pay more.

- *Fair Share Policy:*

  What does a fair share policy mean in a multisession multiresource environment? We note that the system utility objective with revenue maximization interpretation is fair in the sense that it allocates resources using a fair competition among the sessions' utilities (or offered value in the sense of revenue). If session A offers more money than session B for the same resource vector, session B will not get this resource vector unless session A also gets that or better.

- *Priority Policy:*

  The system utility objective with revenue maximization interpretation can easily be extended to a priority-based policy. This can be done by using a system utility objective which is to maximize U given by,

  $$U = \sum_{i=1}^{n} w_i u_i(\mathbf{q}_i), \tag{3.7}$$

  where w_i is a weight factor for session i's utility. Larger w_i means higher priority for resource allocation.

  In the above system, if session A has higher priority than session B and they both have an operating quality $\mathbf{q}_x$ which maps to the same resource vector $\mathbf{r}_x$, then session B will not get quality $\mathbf{q}_x$ unless session A also gets that or better quality. In the system utility expression (3.7), priority means relative importance in resource allocation, and does not mean guarantee. However if the weight factor of a session is sufficiently high compared to others', priority may imply a resource allocation guarantee for that user.

  We assume that the session utility function u_i is given by the user of the session, whereas the weight factor w_i is determined by the provider as part of the system utility objective which reflects the system resource allocation policy. Adjusting weight factors requires changing the resource allocation policy among the sessions.

3.6.6 Solving the MMKP

In order to use the Utility Model for dynamic quality adaptation in a real-time multimedia system, we have to solve the corresponding MMKP instance. The MMKP is one of the harder variants of the knapsack problem [34]. Since this is an NP-hard problem, we do not expect to have an algorithm which will find an optimal solution in less than exponential time. We have developed two solutions for the MMKP: a branch and bound algorithm for optimal solutions, and a heuristic for fast and near-optimal solutions. These solutions are described in Chapter 4.

For effective solution of the adaptive multimedia problem, we may have to compromise the optimality of the solution so that the computation time does not exceed acceptable limits. For such a time-critical application, we believe that the heuristic solution will be more suitable than the optimal solution. This belief will be supported in Chapter 4 by performance data.

3.7 Related Work

Moser presented an Optimally Graceful QoS Degradation (OGQD) Model where the realized QoS of a *single session* multimedia service is gracefully degraded to conform to changes in resource availability [33]. Suppose a multimedia service consists of g elementary services such as audio input, audio output, video input and video output. Elementary service i has n_i implementations $\mathbb{I}_i = (I_i^1, I_i^2, \ldots, I_i^{n_i})$ where each implementation has m_i parameters. The desired QoS of elementary service

i is expressed as a QoS vector, $\mathbf{q}_i = (d_{i1}, d_{i2}, \ldots, d_{im_i})$. Similarly the utility of elementary service i is expressed as a utility vector, $\mathbf{u}_i = (u_{i1}, u_{i2}, \ldots, u_{im_i})$. The user specifies the service requirements expressed as $\mathbb{Q} = ((\mathbf{q}_1, \mathbf{u}_1), (\mathbf{q}_2, \mathbf{u}_2), \ldots, (\mathbf{q}_g, \mathbf{u}_g))$ where $\mathbf{q}_i$ is the desired QoS of elementary service i and $\mathbf{u}_i$ is the corresponding utility.

For each elementary service, the system has to find a suitable implementation and the operating QoS of that implementation depending on the available resources. The goal of the OGQD system is to provide multimedia service with minimum degradation from desired QoS.

Moser formulates the OGQD problem as follows [33].

Find

$$\mathbf{q}(I_i) = (q_{i1}, q_{i2}, ..., q_{im_i}),\ \ I_i \in \mathbb{I}_i \text{ for } i = 1, 2, \ldots, g, \tag{3.8}$$

to maximize

$$\sum_{i=1}^{g} \left[\frac{1}{m_i} \sum_{l=1}^{m_i} u_{il} - \frac{1}{m_i} \sum_{l=1}^{m_i} u_{il} \left(\frac{d_{il} - q_{il}}{u_{il}} \right)^2 \right]^2$$

subject to

$$\sum_{i=1}^{g} \mathbf{r}(I_i) \leq \mathbf{R}.$$

The OGQD Model and the Utility Model have some high level similarities: both models view the dynamic quality adaptation as an optimization problem, and they assume a similar quality-resource mapping. However the two models are fundamentally different in their scope, objective, problem formulation, and mapping:

- The Utility Model is a simple mathematical model for *multisession* adaptive multimedia systems. Each session provides a quality profile with *multiple* operating qualities, and the system finds the dynamic QoS of each session in order to maximize some system utility function. This model degrades the QoS of one or more sessions when there is a scarcity of available resources.

On the other hand, the OGQD Model is a powerful model to express the QoS degradation of a *single multimedia session*. The session provides a desired QoS, and the system finds a realized QoS in order to maximize a session utility function. The model tries to exploit resource-resource tradeoffs for providing the desired QoS (e.g. processor-bandwidth tradeoff by changing compression algorithm), and when it fails, it degrades the QoS by exploiting resource-QoS tradeoffs.

- The Utility Model provides a unified approach for dealing with session admission, quality adaptation and resource allocation. However, since OGQD deals only with a single session; session admission is beyond its scope.

- The Utility Model expresses the multisession adaptive multimedia system, where the QoS of individual sessions may be dynamically adapted to the resource availability of the system and to the choice of the users, as an MMKP. Moser also maps the OGQD problem to an MMKP instance, but how an optimization problem with a *nonlinear* objective function may be mapped to an MMKP is not clear from his article [33].

- In the Utility Model, the session and system utility values may be related to the user's bill and the system revenue, respectively. Thus this model may be applied to a service provider system or server system. The OGQD Model does not deal with multiuser systems, and does not have any notion of billing or revenues.

Kawachiya *et al.* of the Keio MMP project proposed a system architecture for quality adaptation among multimedia sessions [19]. In this architecture, a system-wide server, called QoS Manager, controls the quality of each session by using a QoS Ticket abstraction. When a session is initiated, it registers its QoS requirements with the QoS Manager. The QoS Manager computes the resource requirement, allocates resources for the session, and issues a QoS Ticket to the session. The QoS Ticket incorporates the resource usage privilege of a particular session. A reservation-based resource management scheme distributes resources according to the QoS Ticket privileges of the sessions.

The Utility Model and the MMP Architecture share the same general goal of designing an adaptive multimedia system with multiple concurrent sessions. However the foci of the two projects are fundamentally different and rather complementary: the focus of the Utility Model is to find the adaptive quality of each session at any given system state; whereas the focus of the MMP Architecture is to provide a mechanism to enforce quality adaptation using a reservation-based resource management scheme. Thus, an interesting research project would be to investigate how to integrate the concepts of the Utility Model and the MMP Architecture. For example, one idea would be to implement the Utility Model on the MMP Architecture to design an adaptive multimedia system where the MMP provides the support for QoS specification, monitoring and enforcement, and the Utility Model performs the computation for session admission, quality adaptation and integrated resource allocation.

In the context of networked multimedia system design, the admission of multimedia calls is dealt with in [27] using the idea of multimedia capacity region (MCR). An MCR represents a region in a multidimensional space where each dimension denotes a media stream (such as CD quality audio, JPEG video or MPEG video). Suppose a system load requires 3 CD quality audio, 2 JPEG video and 1 MPEG video streams. These requirements represent a point in the multidimensional space, and if that point is within the MCR, then the system can support this load. Otherwise the system cannot support this load.

The multimedia admission method using the Utility Model is different from that of the MCR approach. In the Utility Model method, we deal with the admission control problem for an adaptive system where the quality of individual sessions may be adjusted to the available resources. We use a unified approach for admission control of new sessions and quality adaptation of existing sessions. However the MCR approach does not have any notion of quality adaptation.

If a system does not support quality adaptation, as in the case of the MCR approach, the concepts of the Utility Model may still be used for admission of new sessions. When a user requests a new session, the system maps its QoS requirement to the requirement of basic end-system resources such as CPU time, memory and network bandwidth. If there are sufficient system resources to support this session, the request can be accepted; otherwise it must be rejected. We note that the admission control using the Utility Model is a generalization of the MCR method because:

- The MCR method is not suitable where the characteristics of media objects are different. For example, it is not suitable for admitting multiple MPEG streams where each stream has a different refresh rate, because then the expression '2 MPEG streams' loses its meaning. However the mathematical method of the Utility Model is still suitable as long as we know the resource requirements of each media component.

- If the set of media objects with different characteristics is small, the Utility Model admission control becomes equivalent to the geometric method of the MCR approach.

3.8 Discussion

3.8.1 Model Features

The Utility Model has the following features:

- The Utility Model provides a unified and computationally feasible way to solve the admission problem of new multimedia sessions, dynamic quality adaptation and resource allocation problem of existing sessions[6].

- This model provides an integrated approach to allocate system resources in order to maximize some system objective. Thus this model enables integrated resource management.

- The concept of quality profile allows the users to specify their service requirements in a flexible manner. For example, the user request may be like "Silver or better quality" (profile has Silver as the minimum acceptable quality) or "Gold quality always" (either the profile has only one choice which is Gold, or each choice of the profile is Gold).

- The users may negotiate with the system to find a suitable (and personalized) quality-utility mapping.

[6]This point will later be validated in Chapter 4.

- The Utility Model allows flexible management policies for service providers. The management policies of a service provider may either be expressed indirectly in the system utility function and parameters, or they may be expressed as additional system constraints. For example, an implementation of the Utility Model may either use an upgrade-only adaptation policy or it may use a bidirectional adaptation policy. In *upgrade-only adaptation*, a session quality can only be upgraded, but it can never be downgraded. However, in *bidirectional adaptation*, both upgrade and downgrade of session quality is allowed.

- The Utility Model provides increased system utility (e.g. revenue or profit) as compared to static reservation approach[7].

3.8.2 Big Questions

In the preceding sections, we have presented the Utility Model for adaptive multimedia systems. It is based on the concepts of quality profile, resource mapping, session and system utility, and system resource constraint. This model incorporates the dynamics of resource management and the interplay of media components within a multisession multimedia environment.

The Utility Model raises the following questions for the system designer:

1. For the Utility Model to be useful in session admission and QoS adaptation, we need to solve the MMKP. How do we solve the MMKP?

2. How can we use the concepts of the Utility Model to design an end-system architecture?

[7]This point will later be validated in Chapter 5.

3. Is it feasible to use the Utility Model for dynamic quality adaptation in real multimedia systems? Or is it just a theoretical model? How do we evaluate the performance of such an implementation?

We address these questions in the three subsequent chapters:

1. Chapter 4 presents two solutions for the MMKP, and evaluates and compares their performance.

2. Chapter 5 presents an end-system architecture, called the Padma Architecture, for adaptive multimedia systems based on the Utility Model.

3. Chapter 6 presents a prototype implementation of the Utility Model, and evaluates its performance.

3.9 Summary

We have defined the adaptive multimedia problem within an AMS, and formulated it as a multiple-choice multi-dimension 0-1 knapsack problem. We have shown how this model may be used in admission control of new sessions, and in dynamic quality adaptation and resource allocation of existing multimedia sessions in a unified way. It provides a computationally feasible real-time method to optimize the utility (e.g. revenue or profit) of real multimedia systems.

Chapter 4

Two Solutions of the MMKP

In this chapter, we present two solutions for the MMKP: a branch and bound algorithm and a heuristic. The branch and bound algorithm produces the optimal solution, whereas the heuristic provides a fast and near-optimal solution. We report computational experiences, and compare the two approaches for practical applications.

4.1 The MMKP

In section 2.5, we defined the MMKP as the following resource allocation problem. Suppose there are n groups (stacks) of items. Group i has l_i items. Item j of group i has value v_{ij}, and requires resource $\mathbf{r}_{ij} = (r_{ij1}, r_{ij2}, ..., r_{ijm})$. The amounts of available resources are given by $\mathbf{R} = (R_1, R_2, ..., R_m)$. The MMKP is to pick exactly one item from each group in order to maximize the total value of the pick, subject to the resource constraints.

Formally MMKP is expressed as follows:

$$V_P = \text{maximize} \sum_{i=1}^{n} \sum_{j=1}^{l_i} x_{ij} v_{ij},$$

such that

$$\sum_{i=1}^{n} \sum_{j=1}^{l_i} x_{ij} r_{ijk} \leq R_k, \quad k = 1, \ldots, m, \tag{P}$$

$$\sum_{j=1}^{l_i} x_{ij} = 1, \quad i = 1, \ldots, n,$$

$$x_{ij} \in \{0, 1\}, \quad i = 1, \ldots, n; j = 1, \ldots, l_i.$$

Here x_{ij} is either 0 implying item j of group i is not picked, or x_{ij} is 1 implying item j of group i is picked.

The MMKP is an important combinatorial problem with applications such as resource allocation, configuration problems and dynamic quality adaptation within multimedia systems. For example, consider a menu selection problem. A person wants to have a meal where she has to choose a beverage, an appetizer, a main dish, and a dessert. For each choice, her preference is expressed as a satisfaction value. The problem is to find a meal which maximizes the total satisfaction subject to constraints such as maximum values of calories, cholesterol, and cost.

MMKP is a generalization of the classical 0-1 KP. To prove this, consider an MMKP instance with n stacks of items and a single resource ($m = 1$). Each stack i has two items: one item has value $v_{i0} = 0$ and requires resource $r_{i0} = 0$, and the other item has value v_{i1} and requires resource r_{i1}. The amount of available resource is R. We note that this MMKP instance is equivalent to the classical 0-1 KP with n items where item i has value v_{i1} and requires resource r_{i1}, and the amount of available resource is R. Now since the classical 0-1 KP is NP-hard [32], MMKP is also NP-hard.

In the context of real-time multimedia, the previous chapter proposed the Utility Model for adaptive multimedia systems where the dynamic quality adaptation within a multisession multimedia system is expressed as an MMKP.

4.2 A Branch and Bound Algorithm

As already mentioned in Chapter 2, the branch and bound method provides a popular approach for many variants of the KPs. Even though branch and bound algorithms have been proposed for many variants of the KP, no such work has been found on the MMKP. This section develops a branch and bound algorithm for the MMKP.

At any solution state of the MMKP, a group is either *free* denoting no item is picked from this group, or it is *fixed* denoting an item is picked from this group. The fixed/free status of the groups are indicated using the group status vector $\mathbf{g}$. If group i is free, g_i is 0; otherwise, the group is fixed and g_i is 1. The solution state of a node is represented by a solution vector $\mathbf{x} = \{x_{ij}\}$ where $i = 1, ..., n$ and $j = 1, ..., l_i$. For a fixed group i, if $x_{ij} = 1$, it implies that item j is picked, and otherwise $x_{ij} = 0$ implying that item j is not picked.

4.2.1 Algorithm BBLP

The branch and bound algorithm for the MMKP involves the iterative generation of a search-tree. The basic algorithm works as follows:

1. Start with a solution state where all groups are free. Compute the upper

bound, select the branching group[1], and initialize the search-tree with this node as the only live node.

2. Find the e-node e which is the live node with the largest value of upper bound.

3. If node e does not have any free group (i.e. all the groups are fixed), then this node represents the optimal solution, and the algorithm terminates.

4. If node e has at least one free group, then this node is *expanded* by *fixing* the branching group b. Fixing group b involves the following steps for each item j of this group:

   - form a new node t where the picked items are the picked items of node e and the item j of group b,
   - compute the upper bound at node t,
   - select the branching group if there exists any free group in t, and
   - if node t is feasible, put node t as a live node into the search-tree.

5. Go back to step 2.

We note that this algorithm is based on the general branch and bound method of section 2.8, and the main distinction of this algorithm lies in dealing with groups of items.

Figure 4.1 presents the pseudocode of the branch and bound algorithm for the MMKP. It may be characterized by two basic decision rules: how to compute an upper bound of the objective function, and how to select the branching group.

[1]Branching group b of a node t denotes the free group of node t which would be fixed when node t becomes the e-node. Computation of the upper bound and selection of the branching group is presented shortly.

/* A branch and bound algorithm for MMKP $*$/

/* Legend:

n: groups, l_j: items in group j,

$\mathbf{v}$: value vector, $\mathbf{r}$: required resource vector, $\mathbf{R}$: total resource vector,

$\mathbf{x}$: solution vector, $\mathbf{g}$: group status vector, $\mathbf{R_2}$: available resource vector,

V_1: objective value, n_2: free groups, b: branching group and U: upper bound */

procedure BBLP()

{

$V_1 = 0,\ n_2 = n, \mathbf{R}_2 = \mathbf{R}, \mathbf{g} = \mathbf{0}$; /* initialize with all groups free */

$U = \text{SolveLP}(\mathbf{x}, \mathbf{g}, \mathbf{v}, \mathbf{r}, \mathbf{R}_2, V_1)$; /* find upper bound U */

find $x[b][j'] = \max_{\substack{i=1,\ldots,n \\ j=1,\ldots,l_i \\ g[i]=0}} x[i][j]$; /* find branching group b */

$\text{TreeInsert}(\mathbf{x}, \mathbf{g}, V_1, n_2, R_2, b, U)$; /* insert first node to data-tree*/

while(1){

  $t = \text{TreeExtractMax}()$; /* extract live node with highest U */

  $\mathbf{x} = t \rightarrow \mathbf{x},\ \mathbf{g} = t\rightarrow\mathbf{g},\ b = t\rightarrow b$;

  if $(t\rightarrow n_2 \leq 0)$

    return $t\rightarrow\mathbf{x}$; /* return if no free variable */

  $g[b] = 1,\ \mathbf{x}[b] = \mathbf{0},\ n_2 = (t\rightarrow n_2) - 1$; /* fix branching group b */

  for$(j = 1, ..., l_b)$ {

    $\mathbf{y} = \mathbf{x},\ \mathbf{R}_2 = t\rightarrow\mathbf{R}_2$;

    $y[b][j] = 1$; /* pick item j of group b */

    $\mathbf{R}_2 = \mathbf{R}_2 - \mathbf{r}[b][j],\ V_1 = t\rightarrow V_1 + v[b][j]$; /* update $\mathbf{R}_2$ and V_1 */

    $U = \text{SolveLP}(\mathbf{y}, \mathbf{g}, \mathbf{v}, \mathbf{r}, \mathbf{R}_2, V_1)$; /* find upper bound U */

    if $(U > 0)${

      find $x[b'][j'] = \max_{\substack{i=1,\ldots,n \\ j=1,\ldots,l_i \\ g[i]=0}} y[i][j]$; /* find next branching group b' */

      $\text{TreeInsert}(\mathbf{x}, \mathbf{g}, V_1, n_2, R_2, b', U)$; /* insert new live node to data-tree */

    }

}

}

Figure 4.1: *Procedure* BBLP*: A branch and bound algorithm for MMKP.*

Computation of Upper Bound: Suppose at a certain node t, there are n_1 fixed groups, and $n_2 = n - n_1$ free groups. The value of the objective function achieved for the decisions already taken to reach node t can be found from the fixed groups:

$$V_1(t) = \sum_{\substack{i=1,\ldots,n \\ j=1,\ldots,l_i \\ g[i]=1}} x_{ij} v_{ij}.$$

The amount of resource available for free groups can be expressed using the vector $\mathbf{R}_2$ where

$$\mathbf{R}_2 = \mathbf{R} - \sum_{\substack{i=1,\ldots,n \\ j=1,\ldots,l_i \\ g[i]=1}} \mathbf{r}_{ij}.$$

Now consider the MMKP instance $P_2(t)$ where the n_2 free groups of P are considered as the groups of items, and vector $\mathbf{R}_2$ denotes the amount of available resources. Suppose

$$U(t) = V_1(t) + V_{LP(P_2(t))}$$

where $LP(P_2(t))$ is the LP relaxation of $P_2(t)$. Since $V_{LP(P_2(t))} \geq V_{P_2(t)}$, $U(t)$ indicates an upper bound of the objective value of P achievable from node t. For an efficient solution of the LP(P) problem, we use the Simplex Method[2] [11, 8] for linear programming.

[2]The Simplex Method is a very powerful technique for solving linear programs. The design of the Simplex Method by G.B. Dantzig in 1957 has strongly influenced the field of operations research. Even though the worst case computational complexity of the Simplex Method grows exponentially with the problem size, this method is very efficient practically, and can be used for linear programs up to hundreds of variables. Details of the Simplex Method can be found in many textbooks such as [25] and [8].

Selection of Branching Group: The solution of the Simplex Method is also used for selecting the branching group. After the Simplex Method computation, the free group which has the maximum value of x_{ij} is used as the branching group. This is based on the hypothesis that a high fractional value of x_{ij} in the solution of LP(P) would also lead us to $x_{ij} = 1$ in the solution of P. However, the optimality of the algorithm does not depend on this hypothesis.

group status vector (n bits)	solution vector x (nl floats)	objective value (float)	# of free groups (integer)	available resource vector (m floats)	branching group (integer)	upper bound (float)	lchild (ptr)	rchild (ptr)

Figure 4.2: *The structure of a data-tree node.*

During the search, we use a tree data-structure, called *data-tree*, for maintaining the live nodes[3]. Each node of the data-tree has the data structure of Figure 4.2. It contains the following fields:

1. Group status vector ($\mathbf{g}$): a vector of n binary digits to indicate the fixed or free status of the groups in the current solution.

2. Solution vector ($\mathbf{x}$): a vector of floating point numbers to store x_{ij} for $i = 1, ..., n; j = 1, ..., l_i$. For a fixed group i, if $x_{ij} = 1$, it implies that item j is picked, and otherwise $x_{ij} = 0$ implying that item j is not picked. For free group i, the value of x_{ij} may also be fractional because the vector $\mathbf{x}$ is also used to store the solution of the upper bound computation using the Simplex Method.

[3]We note the distinction of the search-tree and the data-tree: the former is used for conceptualizing the solution steps where a branch represents picking of an item; whereas the latter is used as a data-structure to maintain the live nodes, and to find the next e-node.

3. Objective value (V_1): a floating point number indicating the value of objective function achieved from the fixed groups.

4. Number of free groups (n_2): an integer indicating the number of free groups.

5. Available resource vector ($\mathbf{R}_2$): is a vector of m floating point numbers indicating the amount of available resources.

6. Branching group (b): an integer indicating the group to be fixed if current node becomes e-node.

7. Upper bound (U): a floating point number indicating the achievable upper bound value of the objective function from the current node.

8. Left and Right Child Pointers (lchild and rchild): two pointers to the data-tree nodes.

The data-tree is maintained in such a way that a parent node would always have a larger upper bound than any of its children. The branch and bound method requires two functions on this tree: insert a node into the data-tree (function TreeInsert()) and extract the node with the largest upper bound from the data-tree (function TreeExtractMax()).

In algorithm BBLP, the data-tree is first initialized by inserting a node where all the groups are free. The upper bound and branching group are based on the Simplex Method. Here function SolveLP() solves the linear program LP(P). If the LP is feasible and bounded, SolveLP returns the upper bound achievable from the current node and the solution of the LP is available in vector $\mathbf{x}$. However, if the LP is unbounded or infeasible, SolveLP returns a negative value.

The main loop of the algorithm involves iterative expansion of live nodes until the optimal solution is found (lines 5–20). Each iteration starts by extracting the data-tree node t which has the largest value of upper bound. If node t does not have any free group ($n_2 = 0$), the algorithm terminates, and the current value of $\mathbf{x}$ yields the optimal solution of P. Otherwise, we *fix* the branching group b. For each item j within group b, we do the following: (1) extend $\mathbf{x}$ by picking item j to generate a new partial solution $\mathbf{y}$; (2) compute upper bound $U(\mathbf{y})$ by solving $LP(P_2(\mathbf{y}))$ using the Simplex Method; (3) select a branching group using the results of the Simplex Method; (4) if the Simplex Method solution is optimal, which implies picking item j is feasible, we insert a new node with partial solution $\mathbf{y}$ into the data-tree.

4.2.2 A BBLP Example

We consider the MMKP instance illustrated in Figure 4.3.

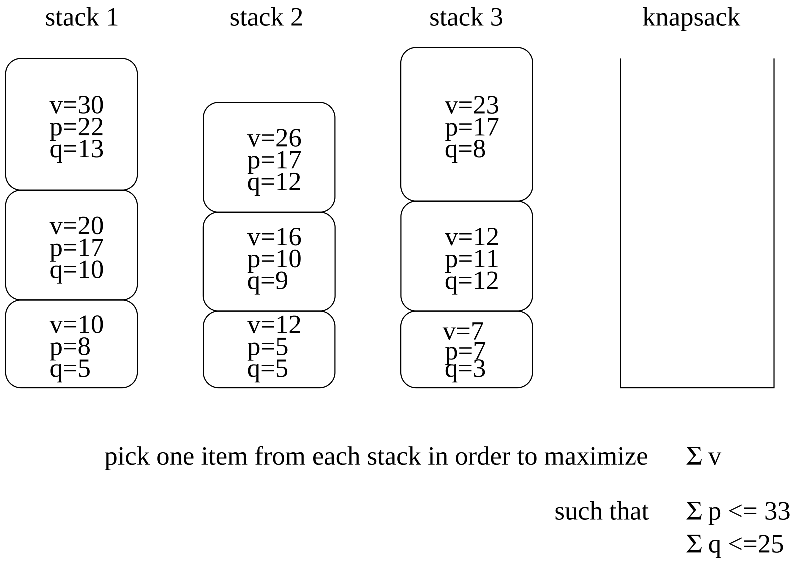


Figure 4.3: *The MMKP instance for the* BBLP *example.*

There are 3 groups of items ($n = 3$), 3 items in each group ($l_1 = l_2 = l_3 = 3$), and 2 resource constraints ($m = 2$). Formally the problem may be expressed as follows:

Maximize

$$10x_{11} + 20x_{12} + 30x_{13} + 12x_{21} + 16x_{22} + 26x_{23} + 7x_{31} + 12x_{32} + 23x_{33},$$

subject to

$$8x_{11} + 17x_{12} + 22x_{13} + 5x_{21} + 10x_{22} + 17x_{23} + 7x_{31} + 11x_{32} + 17x_{33} \leq 33,$$

$$5x_{11} + 10x_{12} + 13x_{13} + 5x_{21} + 9x_{22} + 12x_{23} + 3x_{31} + 12x_{32} + 8x_{33} \leq 25,$$

$$x_{11} + x_{12} + x_{13} = 1,$$

$$x_{21} + x_{22} + x_{23} = 1,$$

$$x_{31} + x_{32} + x_{33} = 1,$$

$$x_{ij} = \{0, 1\} \text{ for } i = 1, ..., 3;\ j = 1, ..., 3.$$

The solution of this problem instance using algorithm BBLP is illustrated in the search-tree of Figure 4.4. Each node of this tree shows three quantities (from top to bottom): the upper bound, the value of the objective function, and the branching group. The sequence at which the nodes are expanded is given beside the nodes. The label x_{ij} written beside a branch indicates $x_{ij} = 1$, that is item j is picked from group i.

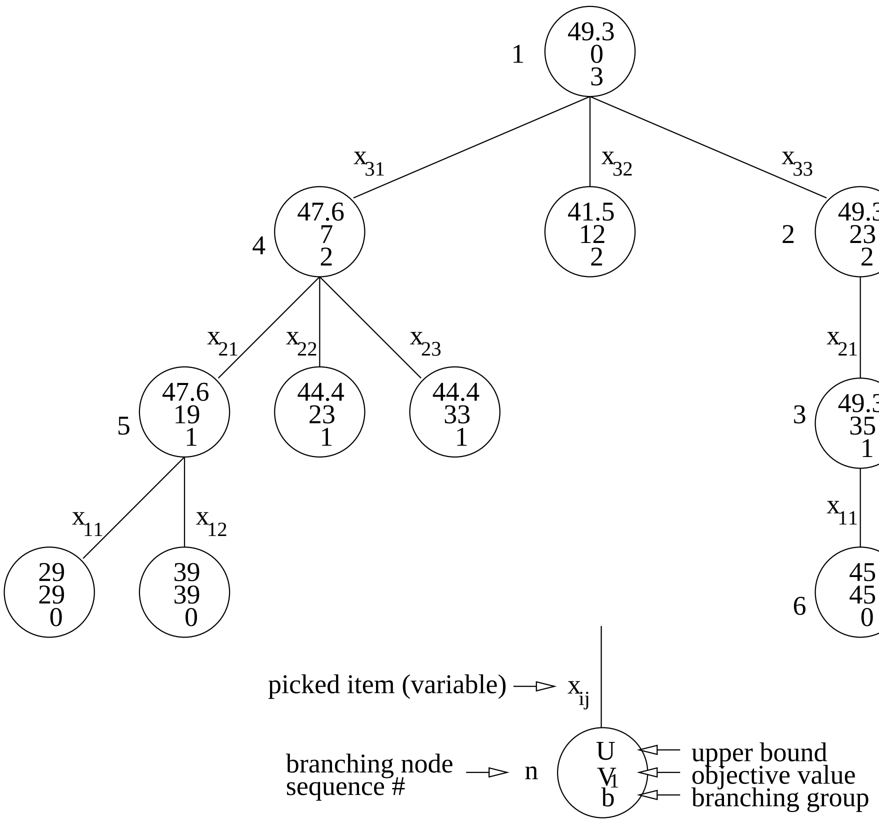


Figure 4.4: *The search-tree for the* BBLP *example.*

The algorithm BBLP starts with the solution state where all groups are free. It computes the upper bound of the objective function and selects the branching group using the Simplex Method, and initializes the data-tree with this node as the only live node. This node becomes the e-node, and the algorithm expands it by fixing the branching group ($b = 3$). Expansion of this group generates three more nodes. Since node 2 has the highest upper bound among the live nodes, it becomes the e-node. We note that expansion of node 2 provided only one branch. This is because the other branches are not feasible. After this expansion, node 3 has the highest upper bound among the live nodes. Thus it is expanded next. Expansion continues in this fashion until node 6 becomes the e-node. Since there is no more free group at this node, it represents the optimal solution.

4.3 A Heuristic Solution

Since MMKP is an NP-hard problem, the computation time for any optimal algorithm, such as BBLP, may grow exponentially with the size of the problem instance in the worst case. This may not be acceptable for time-critical applications such as admission control and dynamic resource allocation in a multimedia system. These applications are forced to accept a near-optimal solution if the computational time for optimal solution exceeds real-time bounds.

4.3.1 Heuristic HEU

Procedure HEU of Figure 4.5 presents a heuristic for a fast and near-optimal solution of the MMKP. Here are the main concepts of the heuristic:

- It starts with a solution where from each group the item with the smallest value (v_{ij}) is picked[4], and iteratively improve the solution by gradually replacing items of smaller values with those of larger values as long as the solution remains feasible[5].

- It uses Toyoda's concept of aggregate resource where the required resource vector of an item is converted to a scalar index using penalty factors taken from the current resource usage vector[49]. Here the main idea is to penalize the use of resources depending on the current resource state. It applies a large

[4]We assume that such a solution is feasible.

[5]Another approach could be to start with the item with the largest value from each group, and then iteratively lower value of the solution until feasibility is achieved. At times of resource abundance, this approach may converge to solution faster than the approach described here.

/* A heuristic for MMKP */

/* Legend:

n: groups, l_j: items in group j, m: system resources,

$\mathbf{v}$: value vector, $\mathbf{r}$: required resource vector, $\mathbf{R}$: total resource vector,

ρ: solution vector, $\mathbf{C}$: resource usage vector,

Δr: aggregate resource saving, and Δp: value gain per unit of extra resource. */

procedure HEU()

{

for $(i = 1, ..., n)$ $\rho[i] = 1$; /* start with items with smallest values */

$\mathbf{C} = \sum_{i=1}^{n} \mathbf{r}[i][\rho[i]]$; /* compute resource usage vector $\mathbf{C}$ */

while(1){ /* iterative improvement */

  $\Delta r_{\max} = 0, \Delta p_{\max} = 0$;

  for $(i = 1, ..., n; j = \rho[i] + 1, ..., l_i)$ {

    if $(\exists k : k = 1, ..., m, C[k] - r[i][\rho[i]][k] + r[i][j][k] > R[k])$ continue;

    $\Delta r = \frac{(\mathbf{r}[i][\rho[i]] - \mathbf{r}[i][j]) \bullet \mathbf{C}}{|\mathbf{C}|}$; /* compute aggregate resource saving */

    if $(\Delta r > \Delta r_{\max})$

      $\Delta r_{\max} = \Delta r$, $i' = i, j' = j$; /* upgrade with max resource saving */

    if $(\Delta r_{\max} \leq 0)${

      $\Delta p = \frac{v[i][\rho[i]] - v[i][j]}{\Delta r}$;

      if $(\Delta p > \Delta p_{\max})$

        $\Delta p_{\max} = \Delta p$, $i' = i, j' = j$; /* upgrade with max value/aggr. resource */

    }

  }

  if $(\Delta r_{\max} \leq 0$ and $\Delta p_{\max} \leq 0)$ return ρ;

  $\mathbf{C} = \mathbf{C} - \mathbf{r}[i'][\rho[i']] + \mathbf{r}[i'][j']$; /* update $\mathbf{C}$ and upgrade solution */

  $\rho[i'] = j'$;

}

}

Figure 4.5: *Procedure* HEU*: A heuristic for MMKP.*

penalty for a heavily used resource, and a small penalty for a lightly used resource.

- To find the next item to be picked, it chooses the one which maximizes the savings in aggregate resource. But if no such item is found, it chooses the one which maximizes the value gain per unit of aggregate resource.

Procedure HEU assumes that, within each group i, the items $j = 1, ..., l_i$ are arranged in nondecreasing order of values, that is if $j' < j''$ then $v_{ij'} \leq v_{ij''}$. This does not affect the generality of the heuristic because even if this is not true for a given instance of MMKP, it may be achieved by a little preprocessing.

Replacing an item with a smaller value with another item from the same group but with a larger value is called a solution upgrade or simply an *upgrade*, because it implies an increase in the value of the solution. An upgrade is called *feasible* if the solution after the upgrade is feasible (that is, it does not violet any of the resource constraints); otherwise the upgrade is called infeasible.

The main loop of the heuristic HEU tries to find a feasible upgrade (lines 3–19). If an upgrade is found, the solution is updated, and the heuristic starts another iteration. Find a feasible upgrade involves the following steps:

1. Find the extra aggregate resource (Δr) for all feasible upgrades.

2. If there exists at least one feasible upgrade which provides savings in aggregate resource (that is the extra aggregate resource computed for the upgrade is negative), then HEU chooses the upgrade which maximizes the savings in aggregate resource.

3. However, if there does not exist feasible upgrade which provides savings in

aggregate resource, then HEU chooses the upgrade which maximizes the value gain per unit of extra aggregate resource ($\Delta p = \frac{\Delta v}{\Delta r}$).

If the heuristic fails to find a feasible upgrade in an iteration, it returns the current solution, and terminates.

4.3.2 Computational Complexity

Heuristic HEU starts with an initial solution where the item with the smallest value (v_{ij}) is picked from each group, and iterates in the loop of lines 3–19 as long as it finds a feasible upgrade. Now what is the maximum number of upgrades HEU may find? For group i, there may be at most $(l_i - 1)$ upgrades. Thus the maximum number of possible upgrades is given by $\sum_{i=1}^{n}(l_i - 1)$, which implies that the loop of lines 3–19 can have at most $\sum_{i=1}^{n}(l_i - 1)$ iterations.

Now in each iteration of loop of lines 3–19, the heuristic HEU uses the loop of lines 5–15. The maximum number of iterations of the latter loop is given by $\sum_{i=1}^{n}(l_i - 1)$. Within the loop of lines 5–15, steps 6 and 7 are the most expensive ones, and each of these two has a computational complexity of $O(m)$. Thus the computational complexity of loop of lines 5–15 is $O(m\sum_{i=1}^{n}(l_i - 1))$.

If we combine the above two results, we find computational complexity of heuristic HEU to be $O(m(\sum_{i=1}^{n}(l_i - 1))^2)$. If we assume that all groups are of equal size, that is $l_1 = \cdots = l_n = l$, then the computational complexity of heuristic HEU becomes $O(mn^2(l-1)^2)$.

4.4 Computational Experience

We implemented procedures BBLP and HEU using the C programming language. For simplicity of implementation, we assumed that each group has the same number of items (that is $l = l_1 = ... = l_n$). In order to solve the linear programs, we used the Simplex Method code from [43].

We tested the implementations using randomly generated MMKP instances. For our experiments we used a Pentium 200 MHz personal computer with 32 MB of RAM running Linux OS (RedHat Package Version 4.2, Kernel 2.0.30). Table 4.1 shows the performance of the procedures. Here the solution quality is presented after normalizing it with the optimal value (as found by BBLP), and computation time is presented in milliseconds. For each set of parameters n, l and m, we generated 10 random MMKP instances. We tested BBLP and HEU on all 10 instances, and tabulated the average value of solution quality and execution time. Table 4.1 also shows the standard deviation ($\delta(V_{\text{HEU}})$) of the solution quality of the heuristic HEU.

We observe that heuristic HEU produced near-optimal solutions which are very close to the optimal solutions provided by algorithm BBLP. The table shows that, on average HEU achieved a value of the objective function which was within 4% of the optimal value with a maximum standard deviation of 1.5%.

On comparison of computation time, HEU was 17 to 28152 times faster than the optimal algorithm BBLP. For example, for $n = 10$, $l = 5$ and $m = 5$ algorithm BBLP produced an optimal solution in 180.3 msec, and heuristic HEU produced an objective value which is 4% less than the optimal value in just 0.45 msec. Here HEU executed 400 times faster than BBLP.

To test the performance of the heuristic HEU, we tested it with problems of varying sizes. For this test, we varied the number of groups n from 5 to 500, and

MMKP parameters			solution quality (normalized)			execution time (msec)	
n	l	m	V_{BBLP}	V_{HEU}	$\delta(V_{\text{HEU}})$	t_{BBLP}	t_{HEU}
5	3	5	1.00	0.99	0.007	1.4	0.08
5	5	5	1.00	0.98	0.011	5.9	0.13
7	3	5	1.00	1.00	0.004	6.6	0.09
7	5	5	1.00	0.97	0.013	33.6	0.23
7	7	5	1.00	0.96	0.015	111.9	0.42
10	3	5	1.00	0.99	0.006	30.5	0.17
10	5	5	1.00	0.96	0.011	180.3	0.45
10	7	5	1.00	0.98	0.007	1734.7	0.89
10	10	5	1.00	0.98	0.004	4692.5	2.10
15	3	5	1.00	0.98	0.006	367.6	0.36
15	5	5	1.00	0.97	0.007	2875.7	1.14
15	7	5	1.00	0.96	0.009	14884.3	2.03
15	10	5	1.00	0.96	0.009	105714.5	4.71
20	3	5	1.00	0.98	0.005	2425.2	0.64
20	5	5	1.00	0.98	0.009	23073.9	1.97
20	7	5	1.00	0.98	0.005	106166.6	3.60
20	10	5	1.00	0.97	0.007	238737.5	8.48

Table 4.1: *Performance of procedures* BBLP *and* HEU.

kept each of parameters l and m constant at a value of 5. Figure 4.6 shows the variation of average computation time with the number of groups. We note that the graph looks like a section of a parabola. This is compliant with the n^2 term in the theoretical computational complexity of $O(mn^2(l-1)^2)$.

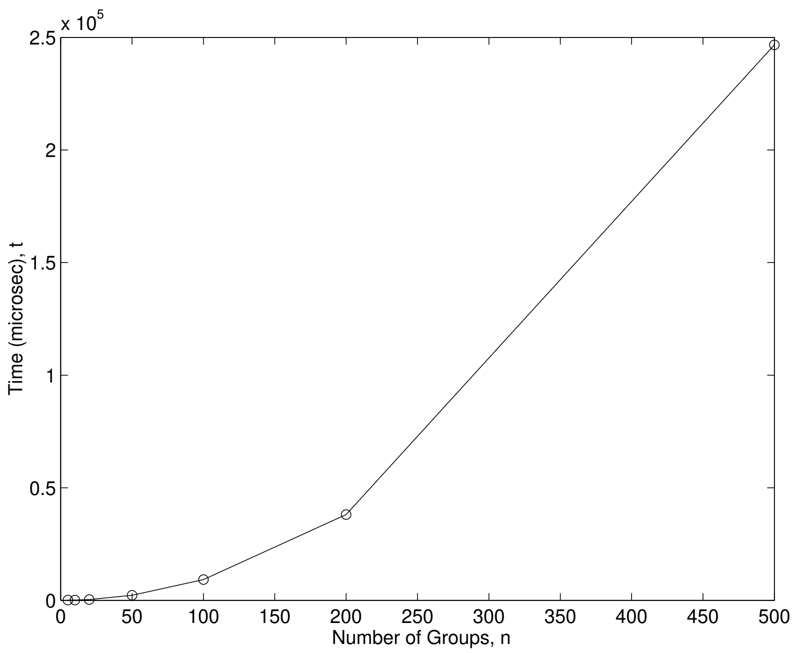


Figure 4.6: *Variation of computation time of* HEU *with number of groups n.*

We believe that heuristic HEU will be a very good candidate for solving MMKP in time-critical or real-time applications such as adaptive multimedia systems. Session admission decisions in multimedia systems need to be made in times well under 1 second; say 500 msec. Table 4.1 shows that BBLP is unsuitable for many cases; HEU is suitable for all cases tried. However, in non-real-time applications such as project planning or long term resource allocation problems, algorithm BBLP will be more suitable to guarantee an optimal solution.

4.5 Summary

We presented two solutions for the MMKP: a branch and bound algorithm BBLP for optimal solutions, and a heuristic HEU for fast and near-optimal solutions. We presented experimental results comparing the two solutions. The heuristic HEU is a very good candidate for time-critical applications where a near-optimal solution is acceptable, and fast computation is more important than guaranteeing optimal value. However in applications where optimality of solution cannot be compromised, BBLP will be more appropriate.

Chapter 5

The Padma End-System Architecture

In this chapter, we present the Padma Architecture – an end-system architecture for multisession adaptive multimedia systems (AMSs) where the quality of each session is dynamically adapted to the change of resource availability and user preferences. This architecture is based on the Utility Model for integrated and adaptive management of system resources, and uses metaspaces to encapsulate the machinery of quality adaptation. The former provides improved resource utilization and dynamic quality adaptation, and the latter provides the programmers of distributed multimedia applications freedom from the concerns of low-level resource management issues.

5.1 Design Issues

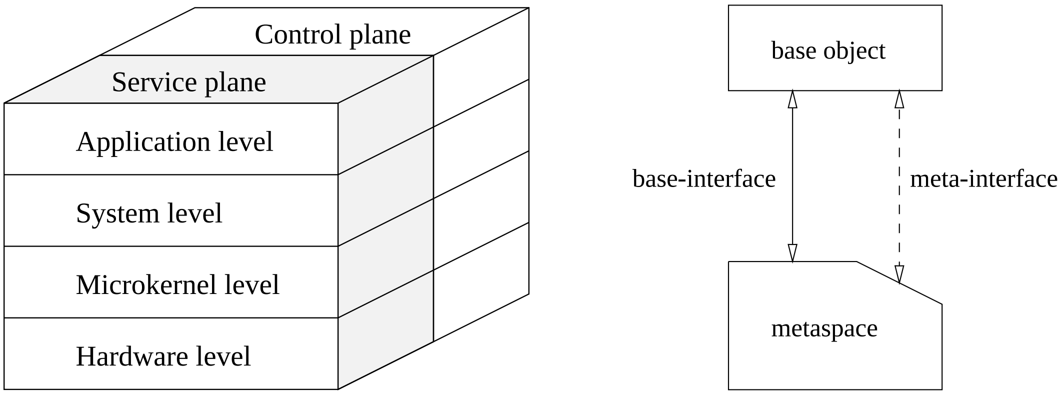
Our goal is to design a QoS-based system architecture to support adaptive quality multimedia sessions. We oppose QoS-aware applications: we believe that multimedia applications should not be concerned with quality of service (*QoS-transparent applications*), and the actual QoS should be dynamically decided by the system at run-time. This approach has two desirable features: The application programmers do not have to worry about the run-time dynamics of the system (separation of concerns), and dynamic quality adaptation can also be provided to legacy applications.

We assume that multimedia programs are composed of media objects such as audio, video and channel objects, supported by an adaptive multimedia object library. This library provides an application programming interface incorporating media objects supporting recording, processing, transmission, synchronization, presentation, and quality control and adaptation of media objects. The multimedia object library and the underlying system cooperate to provide quality of service adaptation to applications.

Our design is guided by the following decisions:

- **Utility Model for Integrated Resource Management:** We use the Utility Model for an integrated and adaptive management of system resources[1]. In this model, each session specifies its user preferences using a *quality profile* (from minimum acceptable quality to highest quality desired), and the system

[1] We note that Utility Model is proposed to be a general mathematical model to express the dynamics within an adaptive multimedia system. In contrast, Padma Architecture is our proposal to realize the Utility Model. Thus the dependency between the model and the architecture is unidirectional: Padma Architecture uses the Utility Model, but the realization of the Utility Model is not restricted to the Padma Architecture.



(a) (b)

Figure 5.1: *Service-control separation using metaspaces: implementation of the service plane using the base interface (service interface) and of the control plane using the meta-interface (control interface).*

finds the operating quality of (and the amounts of resources to be allocated to) each existing session in order to maximize some *system utility function* under system resource constraints. The model provides a unified and computationally feasible approach to make session admission, quality adaptation and resource allocation decisions within a multisession system.

- **Service-control Separation:** Service and control are distinct activities, and should be separated in the system architecture[2]. *Service* is related to the activities invoked by requests such as an application/system programming interface call, whereas *control* is related to activities invoked by service implementation mechanisms, such as reservation of network bandwidth or control signals denoting network congestion or system load drop. Figure 5.1 (a) shows the

[2]This concept originated in the context of the B-ISDN model and ATM networking where service, control and management issues are distinguished [18]. We port this concept to operating system design in a simplified form. Here we consider management within control, and thus only service and control are distinguished.

layered structuring of the Padma Architecture where service and control are separated into different planes. For example, in a video transmission system, sending the next frame is a service operation and changing the refresh rate is a control operation.

- **Abstraction using Metaspaces:** As already mentioned, application programs are composed of objects from an adaptive media library, and the quality control and adaptation is achieved by the media objects with support from the underlying system without QoS-awareness of the application programs. In order to encapsulate the quality control and adaptation issues from the application programs, we use the metaspace concept. The metaspace concept was introduced at the Sony CSL for designing a flexible and dynamic operating system called Apertos [53]. Here the execution of an object is supported by a *metaspace* consisting of some *metaobjects*. A metaobject is a flexible object which can accommodate changes in its behavior/implementation. As shown in Figure 5.1 (b), a metaspace has two types of interface [22]: the base interface for accessing the service functions and the meta-interface for changing or controlling the service behavior.

  Metaspaces provide an elegant mechanism for implementing service-control separation[Figure 5.1]: the service plane is implemented using the base interface (also called the service interface) and the control plane is implemented using the meta-interface (also called the control interface). Thus application programmers deal only with the base interfaces of the media objects, whereas the designers of the object library and the system have to deal with both the base and meta-interfaces in order to support dynamic quality adaptation.

5.2 The Padma System Architecture

Figure 5.2 presents the Padma Architecture for AMSs. The architecture consists of four levels:

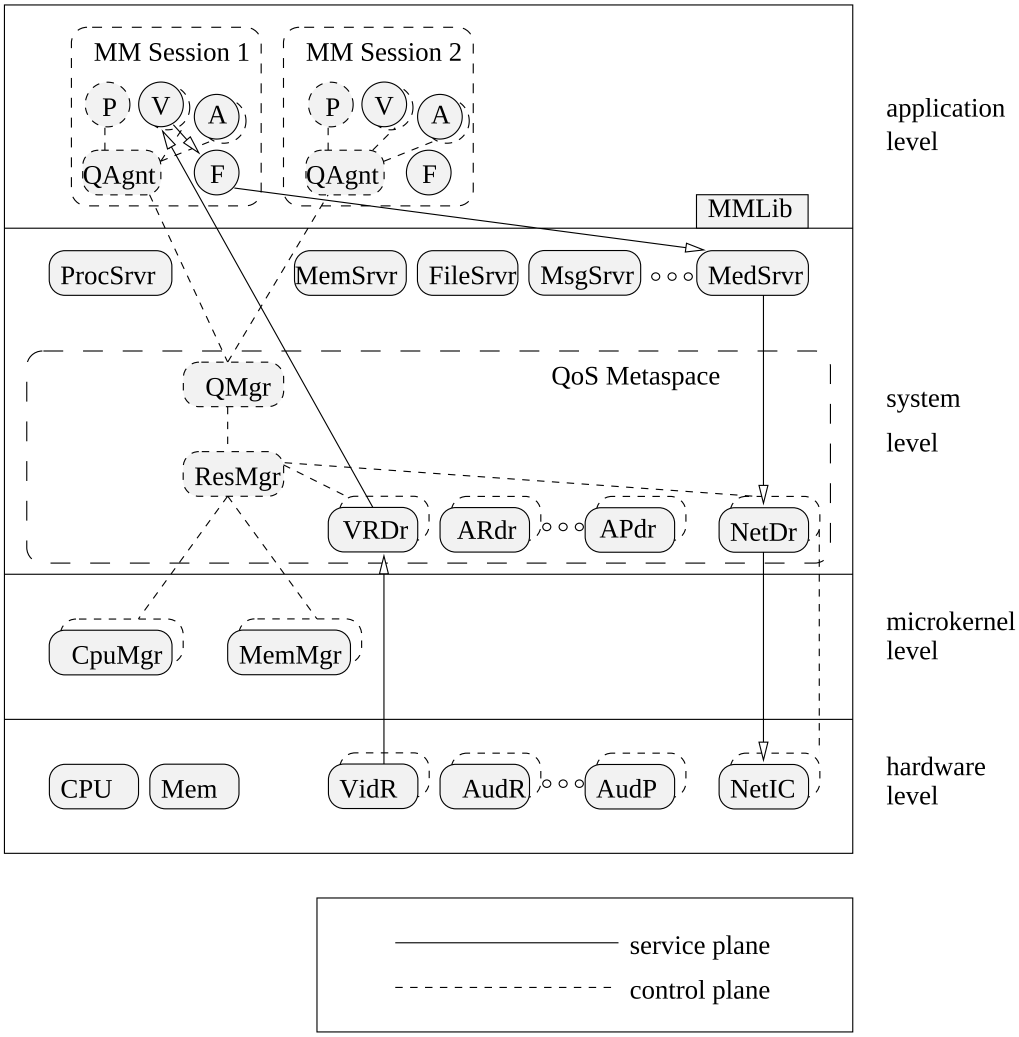


Figure 5.2: *The Padma system architecture.*

1. **Application Level:** A multimedia library (MMLib) provides the media objects such as audio(A), video(V), media filters(F)[3], quality profile(P), and QoS

[3]For instance, a filter may remove the E2 enhancement layer from a layered MPEG-2 stream.

agent(QAgnt).

2. **System Level:** It consists of servers such as QoS manager(QMgr), process server (ProcSrvr), memory server (MemSrvr), file server(FileSrvr) and media server(MedSrvr). This is supported by the resource management scheme involving the resource manager(ResMgr), and device drivers such as video recorder driver(VRDr) and audio player driver(APDr) and the network driver(NetDr).

3. **Microkernel Level:** The microkernel level provides access to hardware resources such as CPU and main memory.

4. **Hardware Level:** The hardware level consists of the system components such as CPU and memory; the interfaces and devices for recording, encoding or presentation of different media (such as audio/video recorder/player), and for accessing the communication network (e.g. network interface card).

For simplicity, Figure 5.2 does not show all the metaspaces. It only shows the system level QoS metaspace for illustration. We use solid lines for service objects and base interface (service interface) calls, and (small-spaced) dashed lines for control objects and meta-interface (control interface) calls.

5.3 Quality Management Hierarchy

The Padma Architecture proposes integrated and adaptive resource management using the Utility Model. It uses a hierarchical quality management and adaptation scheme. This is suitable for a multimedia end-system where the relation of objects is naturally hierarchical: the end-system resources are allocated to the individual sessions, and the resources of each session are allocated to the individual media

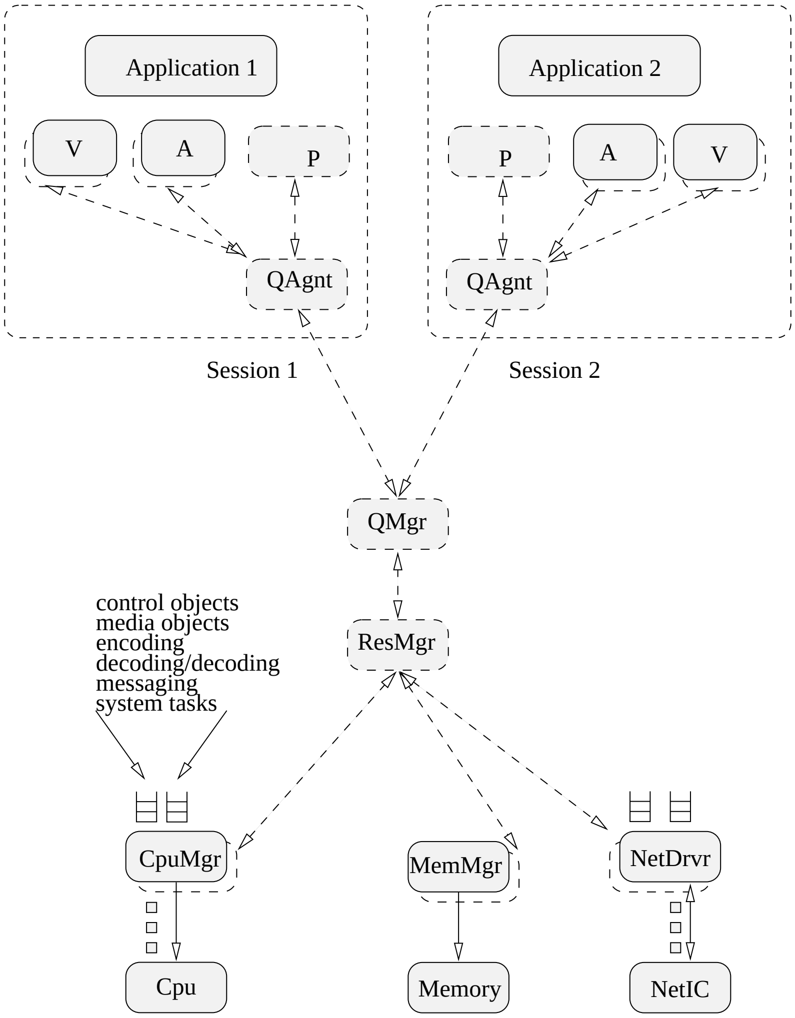


Figure 5.3: *Hierarchical Quality Management in Padma.*

components. Figure 5.3 shows the quality management hierarchy in the Padma Architecture. We can identify three domains of management:

Application domain: The quality control and adaptation issues of each session are handled by a quality agent QAgnt. QAgnt negotiates with its peers to maintain a consistent view of the session operating quality. For example, a server should not send video at 30 fps when the client is expecting only 15 fps. It enforces the proper QoS parameters for the media objects A, V and F in order to utilize the session's share of the system resources.

System domain: The QMgr and ResMgr at the system level cooperate to provide appropriate qualities to all the sessions. The QMgr is the heart of the Padma Architecture. It works in close cooperation with the QAgnts at application level and the ResMgr at the same level. It implements the concepts of the Utility Model to make session admission, quality adaptation and resource allocation decisions. The QMgr's resource allocation decisions are then enforced by the ResMgr by sending suitable control signals to the resource managers such as CpuMgr, MemMgr, NetMgr and VRDr.

Resource domain: Any quality of service management scheme must start with resources. Individual resource managers such as CpuMgr, MemMgr, NetMgr and VRDr support mechanisms for admission, reservation, scheduling and policing of resources to enforce the signals from the ResMgr.

5.3.1 The Role of QoS Manager

Figure 5.4 shows the operation of the QMgr. It executes an event-based algorithm, called the Utility engine, which implements the concepts of the Utility Model. The system management policy is specified using some system utility function by the system administrator. QMgr receives control signals such as new session request or session drop (± session) and change of quality profile from QAgnts, and change of resource availability signals (± P,M,B) from the ResMgr. It derives the session admission, quality adaptation and resource allocation decisions, and issues signals such as change operating quality or admit/reject session to QAgnts, and change of resource allocation (± P,M,B) to the ResMgr.

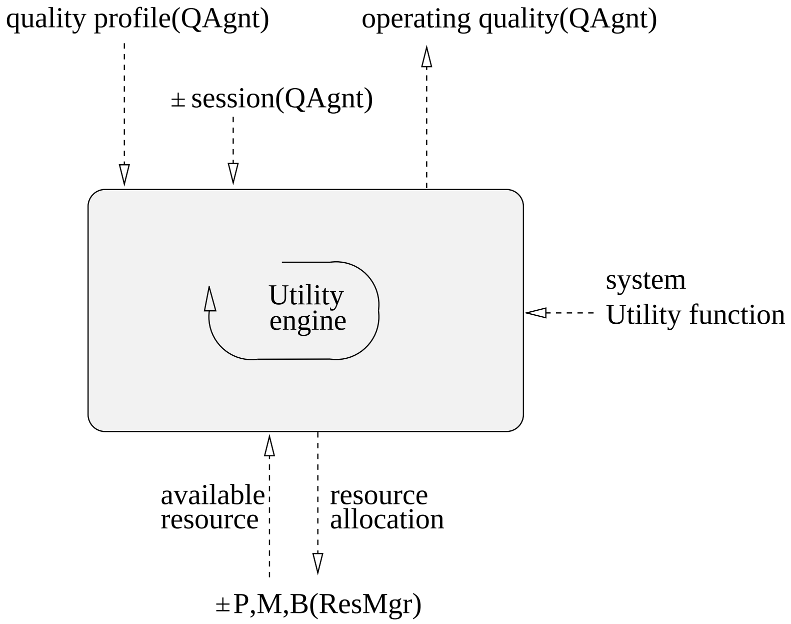


Figure 5.4: *Operation of the QMgr.*

- **Session Admission:** Admission procedure starts from the application level when a QAgnt requests a new session. QAgnt specifies the user preferences using a quality profile. The QMgr negotiates with the QAgnt and the ResMgr, and admits the session only when (a) the system can reserve enough resources to guarantee the minimum acceptable quality, and (b) there is enough incentive to admit this session (e.g. increase in system utility objective).

- **Quality Adaptation and Resource Allocation:** The QMgr is responsible for the adaptation management of the system based on the dynamic resource status, and the signals it receives from other components. It receives all the signals which may potentially trigger adaptation (such as network congestion, resource scarcity/surplus and service failure). It derives the adaptation decisions such as when and how to change the quality of individual sessions, and what should be the corresponding change in the amounts of resources allocated to the sessions. When the QMgr decides to change the operating quality of a session, it informs the corresponding QAgnt of this decision, and issues the appropriate resource allocation signals to the ResMgr in order to enforce it.

5.4 The AVTS Example

To illustrate the operation of the Padma end-system architecture, we consider the AVTS introduced in section 1.1. As already mentioned, it is a client-server system where the client receives live audio and video from the server.

Figure 5.5 shows the AVTS system in the Padma Architecture. The server gets media input from a video camera and a microphone system, and the client plays the media on a display and speaker system. The components on the server side are QAgnt for overall quality management, P for tracking the client's quality profile, V and A for the video and audio media components, and finally filter(s) if there is a scaling requirement[4]. The components at the client are QAgnt, P, CPanel, V, A and Synch. CPanel is the user's GUI[5] control panel where the user can provide her quality requirements, and control the direction of adaptation. Synch takes care of the timing and synchronization requirement among audio/video media. We note that the media objects have both service components (A, V and F) and control components (A′, V′ and F′, shown as objects with dashed outline). The service components participate in the media data generation and transmission, while the control components participate in quality control, connection setup and run-time adjustment with the help of QAgnt.

[4]Here we assume only one client. If there are more than one client, the server objects are replicated.

[5]GUI: Graphical User Interface.

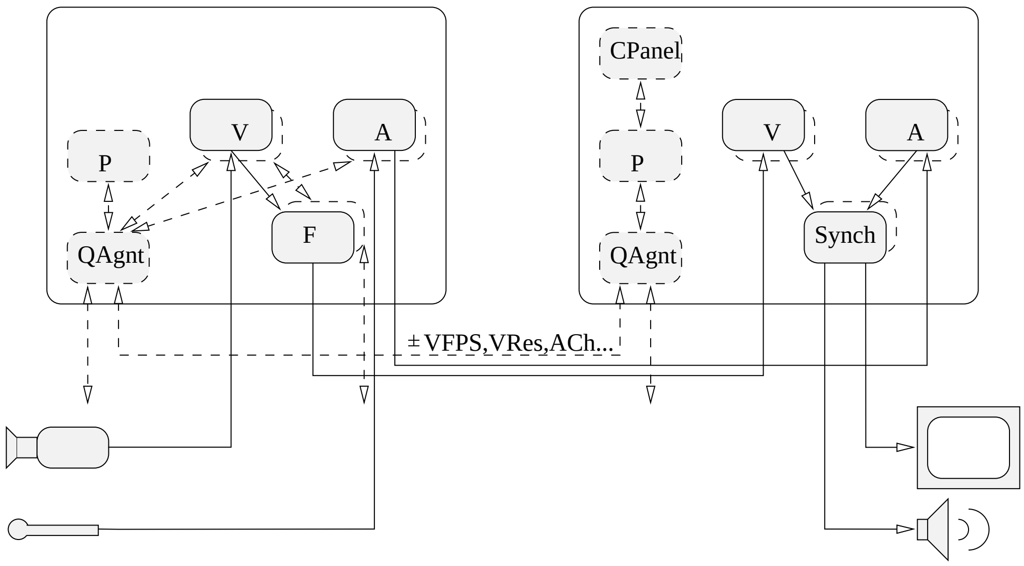


Figure 5.5: *The client-server based AVTS system (Application level only).*

Figure 5.5 also shows some service and control paths both local and end-to-end. A QAgnt, with the help of P, negotiates with the QMgr to establish a QoS contract. It issues the control signals to the media objects to make best utilization of the resources allocated to the session. Examples of these control signals include 'reduce the sampling rate', 'send only the mono audio', and 'cut the E2 layer out from the MPEG-2 video stream'. The QAgnts at the client and server communicate end-to-end to maintain a consistent view of the session operating quality. For example, if the user requests better resolution video using the CPanel, the client QAgnt passes this request to the server QAgnt. In reply, the server QAgnt may ask whether the client is ready to accept a lower frame rate in return.

It is to be noted that we have shown a filter on the server side in the illustration, but none at the client side. This is because it is usually preferable to scale a stream down as early in the transmission path (i.e. as close to the source) as possible in order to avoid wastage of resources such as transmission bandwidth. However, this may not always be true, especially in cases such as resource constrained servers or heterogeneous multicast receivers. For example, in [5], Campbell *et al.* illustrate an organization for a scalable MPEG-2 flow where some filters are placed at the client ends and others are scattered around the transmission network.

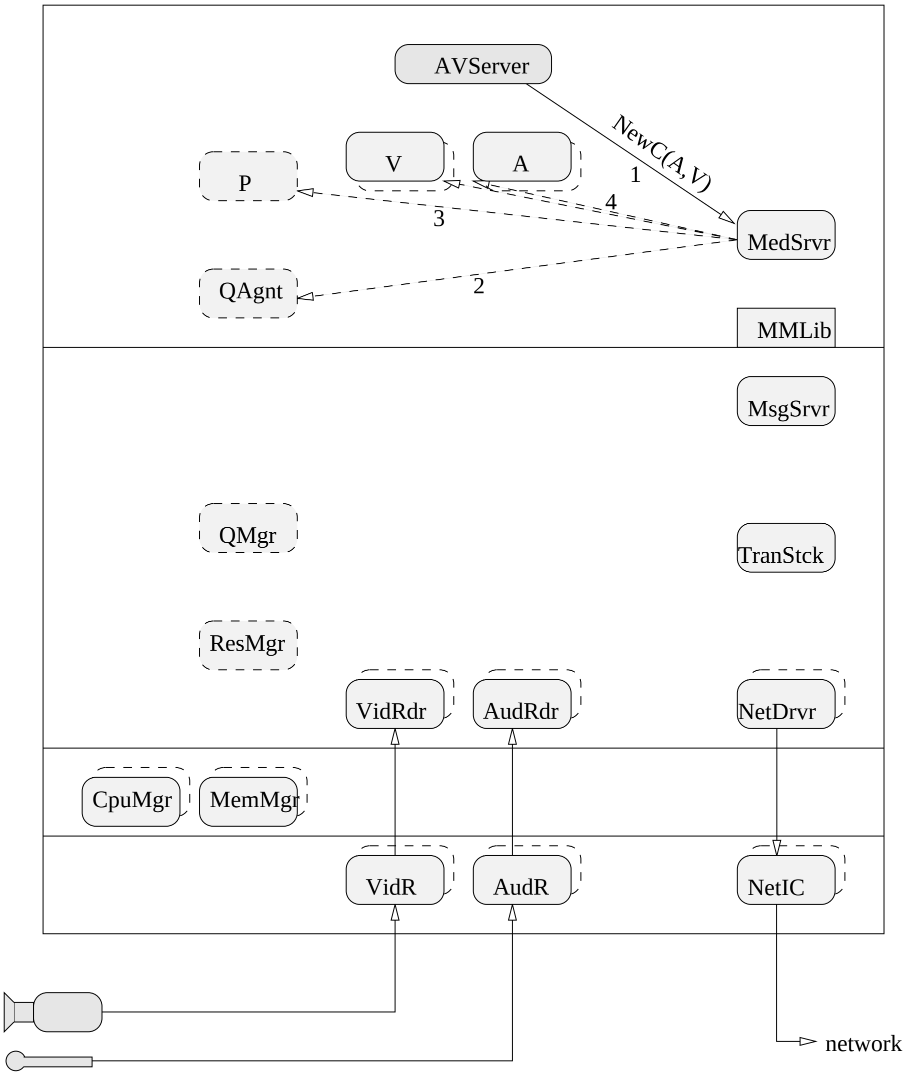


Figure 5.6: *Events to start a session for a new client: Part 1.*

When a user wants a new AVTS session, it triggers the following sequence of events [Figures 5.6 and 5.7]:

- The client requests a new AVTS session with two media: audio and video.

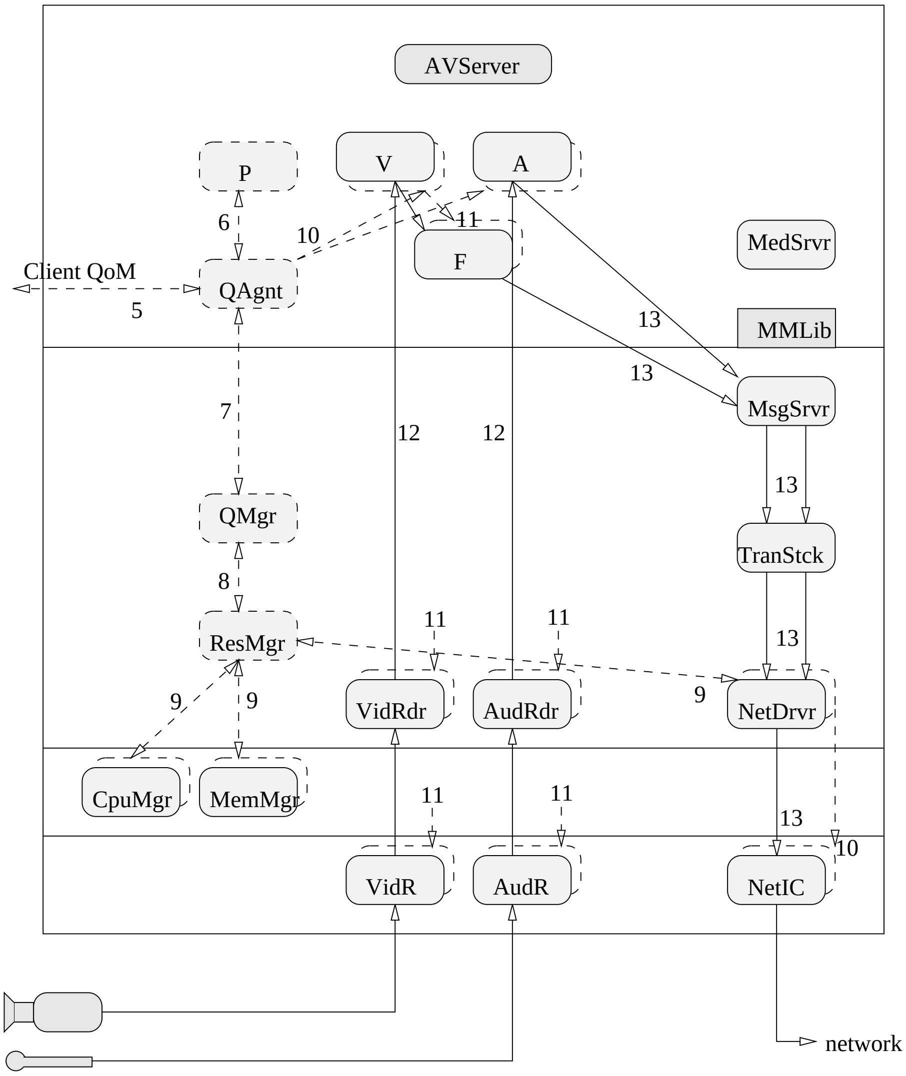


Figure 5.7: *Events to start a session for a new client: Part 2.*

- The media server (MedSrvr) instantiates two media objects A and V, and some control objects such as a quality profile (P) to maintain the user's quality preferences, a control panel (CPanel) for the user interface, and a QoS agent (QAgnt) for quality management of the session.

- The client QAgnt establishes a control channel with the server, and in turn, the server instantiates a QAgnt and a profile P at the server end. The client P and server P are kept consistent to reflect the user's preferences.

- The client and the server QAgnts negotiate between themselves, and also with the local QMgrs, to maintain a consistent view of the session operating quality.

- The media server establishes a media transfer channel from the server to the client which may involve resource reservation in the network nodes.

- The QAgnts enforce the operating quality both at the server and the client. This may involve setting the correct QoS parameters for the sampling, encoding or filtering of the stream at the server, and timing and inter-media synchronization at the client.

Suppose an AVTS session is using a network bandwidth of 3 Mbps. Now suppose, due to network congestion, that the bandwidth available to the session drops from 3 Mbps to 1 Mbps. The Padma Architecture handles this situation as follows. The client QAgnt observes the delivered QoS periodically, for example by observing the packet loss rate. When the loss rate increases above some acceptable threshold, the QAgnt initiates an adaptation process. The system finds a new operating quality suitable for the current level of resource availability which is then enforced by the server and client QAgnts, by proper adjustment of the QoS parameters of the media objects. For example, the server QAgnt may drop the video refresh rate and/or transmit mono quality audio instead of stereo.

Control and Adaptation Signals

The paths for some of the control and adaptation signals within the server end-system are shown in Figure 5.8. Control signals may originate at the client QAgnt due to the dynamic choice of the users, or due to dynamic adaptation at the client end-system. Control messages exchanged between the server and client QAgnts include 'change the video frame/sec' (±VFPS) or 'change the number of audio channels' (ACh). The control signals propagate within the system, with their paths determined by the origin of the signal and the format of the input media. For example, a change video sampling rate is propagated from QAgnt to the video object, then to the video driver, and finally to the video recorder device if this implies changing a parameter of the encoding mechanism. On the other hand, if one has to discard the E2 layer of a MPEG-2 stream, a filter is put between the video object and the MsgSrvr.

5.5 Relation to the Utility Model

Th Utility Model provides the basis of the Padma Architecture. It provides the Padma Architecture with the following strengths:

- The dynamic choices and preferences of the users can be expressed using the quality profiles. Quality profiles are vital for finding the suitable direction of adaptation for a particular session.

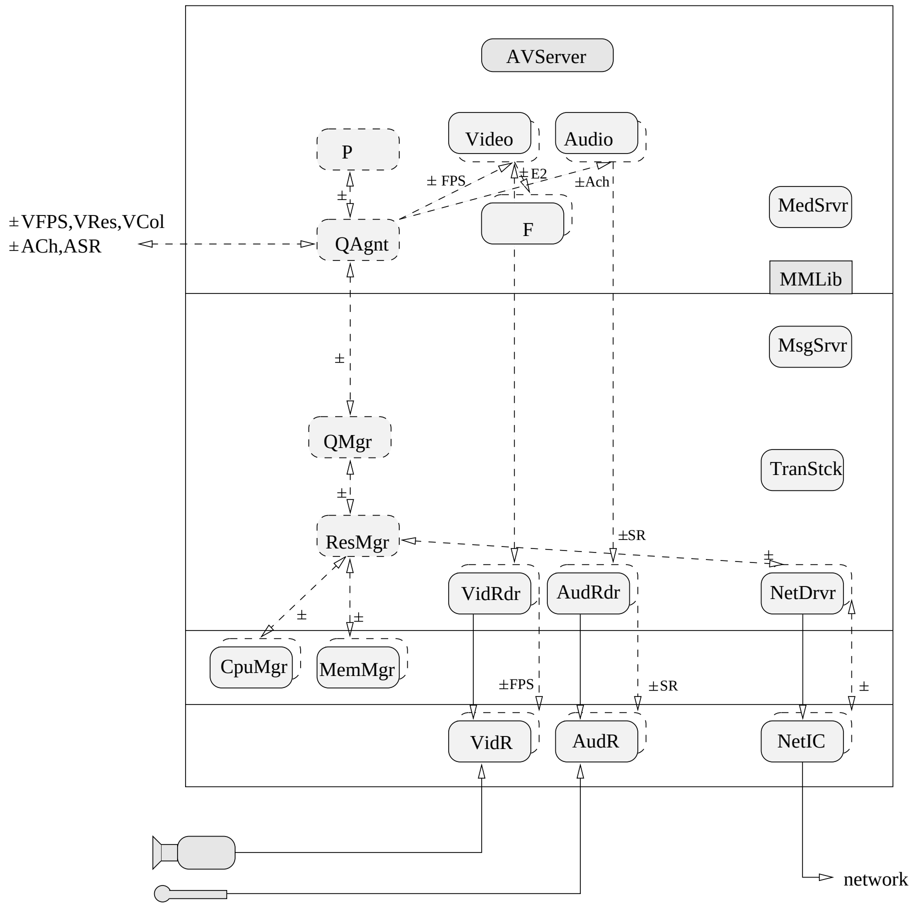


Figure 5.8: *Some control and adaptation signals and messages at AVServer.*

- The management policy of the end-system can be expressed using a system utility function, and then the goal of the end-system is to dynamically optimize this system utility function subject to system resource constraints.

- The Padma Architecture uses the mathematical formulation and the corresponding solutions of the Utility Model to make session admission, quality adaptation, and resource allocation decisions. The Utility Model provides a unified and computationally feasible way to make these decisions.

- The Utility Model allocates all the system resources in a unified way based on the current resource availability and user preferences. This enables an integrated and adaptive resource management scheme for the Padma Architecture.

- The management policy of a system can be changed on the fly just by changing the system utility function.

5.6 Summary

We have presented the Padma Architecture for multisession adaptive multimedia systems, where the operating quality of each session is dynamically adapted to the change of resource availability and user preferences. This architecture has two novelties: (1) integrated and adaptive management of system resources based on the Utility Model, and (2) the use of metaspaces to encapsulate the machinery of quality adaptation. The former provides improved resource utilization and dynamic quality adaptation, and the latter provides the programmers of distributed multimedia applications freedom from the concerns of low-level resource management issues.

Chapter 6

Utility Model Prototype Implementation and Evaluation

In this chapter, we present a prototype implementation of the Utility Model which we call the Utility Model Demonstration Prototype (UMDP). This prototype demonstrates the fundamental concepts proposed in the Utility Model. It shows the capability of the Utility Model to deal with session admission, quality adaptation and integrated resource allocation in a unified way. We have two implementations of the UMDP: UMDP1 using the optimal algorithm BBLP, and UMDP2 using the heuristic HEU. We present traces of UMDP1 and UMDP2 using a random sequence of session requests and drops, and compare the performances with the performance of a simple reservation model prototype (SRMP).

6.1 Prototype Introduction

The UMDP is a GUI-based prototype implemented to demonstrate the concepts proposed in the Utility Model for adaptive QoS systems. Specifically it demonstrates the ability of the model to deal with:

- new session request,
- drop (termination) of an existing session,
- change in total resources available to the system[1], and
- change in quality profile of an existing session.

To simplify the implementation, the UMDP assumes three resources and nine levels of quality denoted as L1 ,L2,..., L9. We also assume that the quality-resource mapping is available as a table, where each of the nine quality levels is mapped to a three-tuple resource vector. We use the resource mapping of Table 6.1 for our experiments. In Table 6.1, for any resource, a higher quality level always maps to an equal or larger amount of required resource. However, this is not a requirement of the Utility Model or of the prototype implementation.

The main window of the UMDP is shown in Figure 6.1. The left side of this window shows the overall state of the multisession multiresource system. It shows the scheduling status of the system resources: reserved, allocated, free[2] and total.

[1]In practice, the available resources in a system may change due to change in non-QoS system load, or due to dynamic change of distributed resources, such as connection bandwidth, where the system does not have direct control, or due to failure or unpredictable behavior of some part of the system.

[2]The system resource not allocated to any session is called free resource.

QoS level	Resource 1	Resource 2	Resource 3
L1	3	4	5
L2	6	7	8
L3	10	8	11
L4	12	8	14
L5	20	10	20
L6	30	25	25
L7	35	30	40
L8	39	35	45
L9	45	40	52

Table 6.1: *Quality resource mapping assumed for demonstration.*

At the right side of this window, a graphical scale shows the system utility. The main window provides buttons for requesting new sessions and changing the total amounts of resources. Choosing the new session button opens a quality profile window. Choosing the change resource button brings up three scales, which may be used to change the total amounts of resources available to the system. For any system resource, the system does not allow reduction of resource quantity below the currently reserved quantity.

The quality profile window is used to specify the quality requirement and the utility values for the new session as shown in Figure 6.2. Each session is described using three *service levels*, namely Gold, Silver and Bronze. It is assumed that an user prefers Gold service to Silver service, and Silver service to Bronze service. For each service level, the user chooses a quality level from the nine quality levels L1,

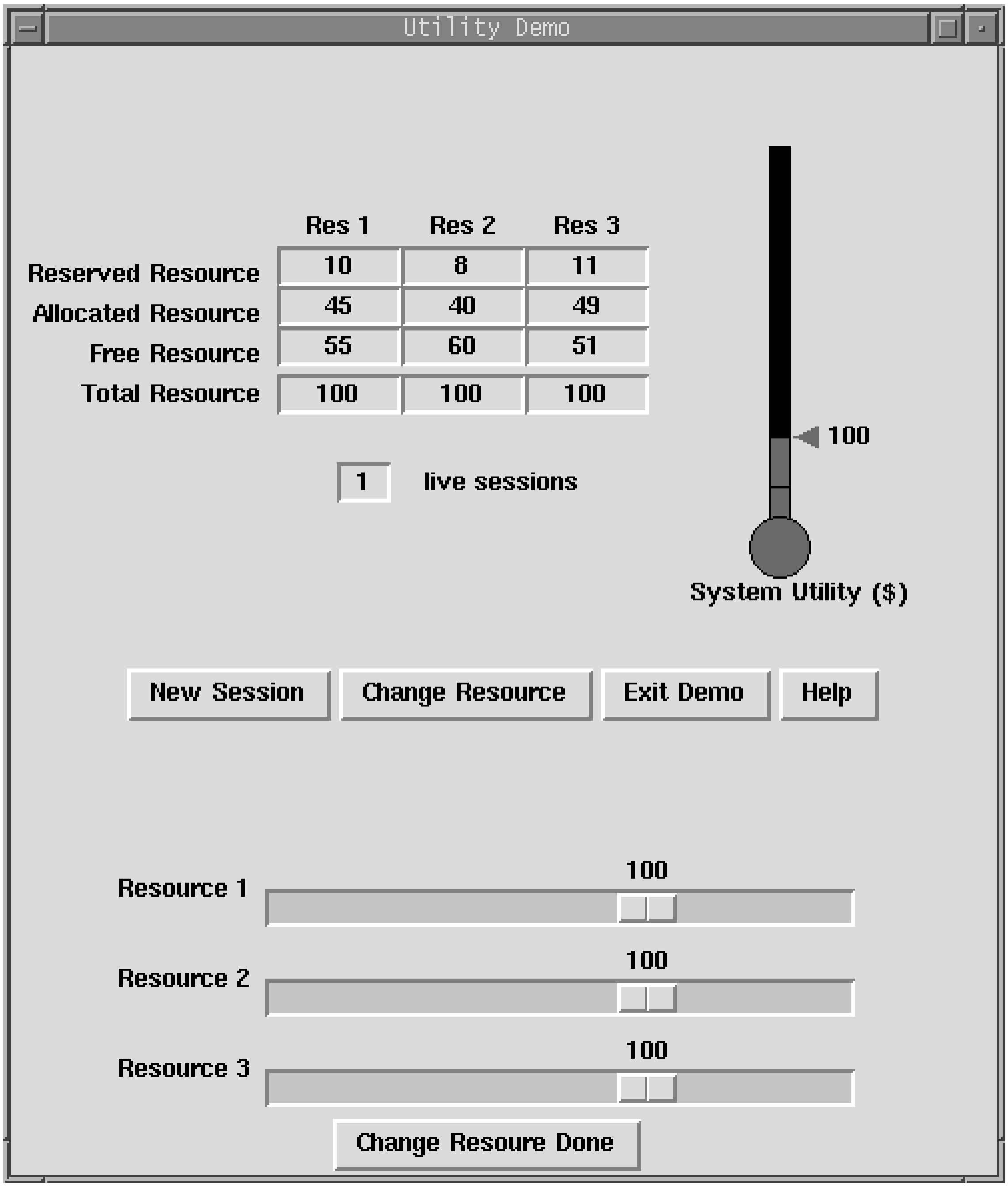


Figure 6.1: *The main window of the UMDP.*

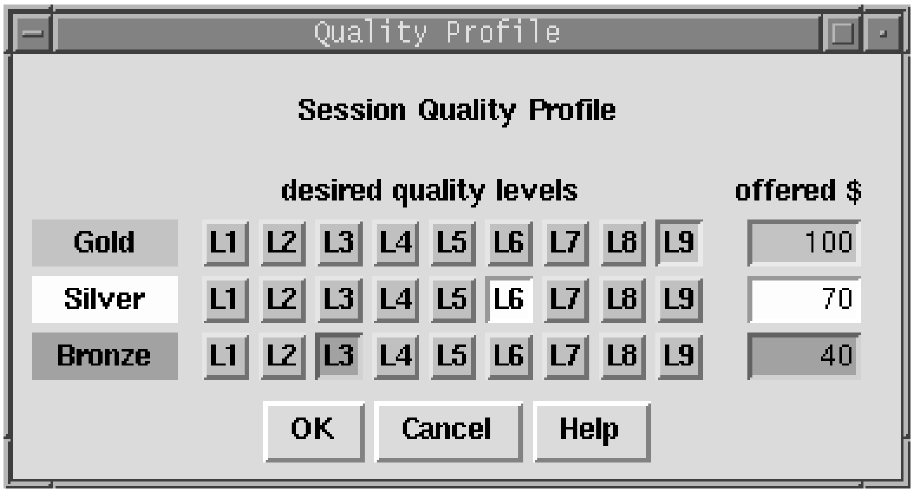


Figure 6.2: *Quality Profile window for session request.*

Figure 6.3: *Three normal session windows.*

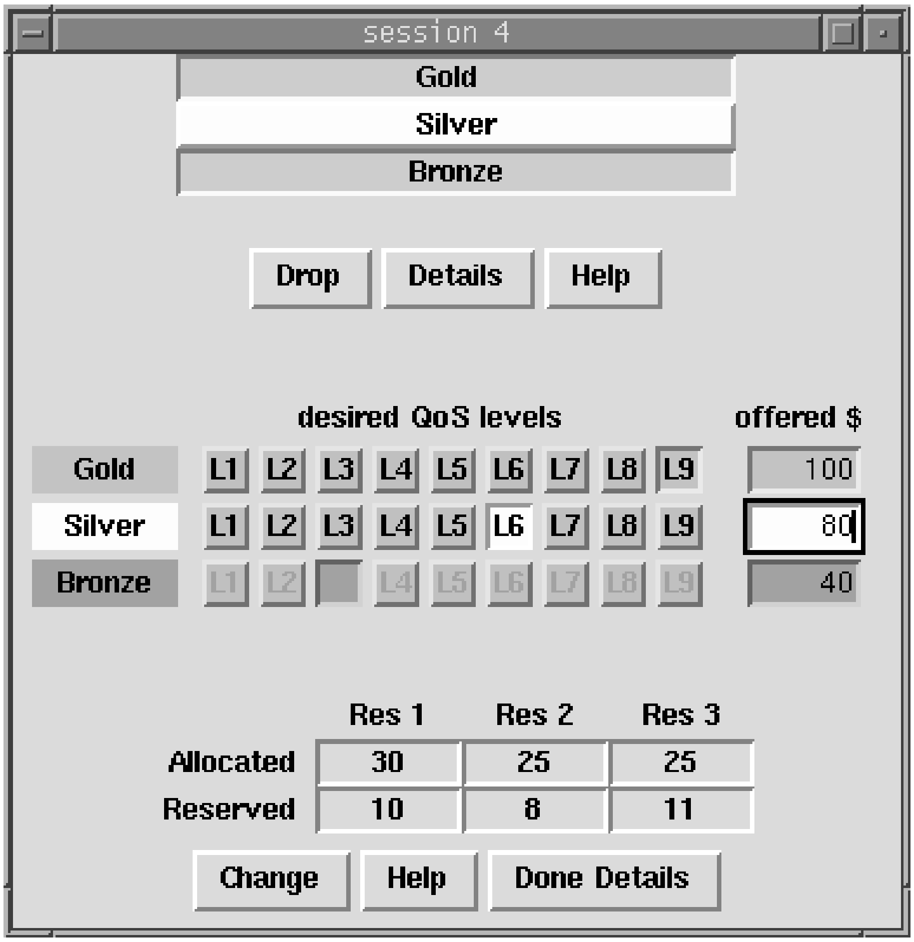


Figure 6.4: *Session window: details.*

L2, ..., L9, and provides a utility value (offered price)[3].

When a new session is requested, the UMDP provides a set of default values, both for quality levels and utility values. These default values could be obtained using a system-wide default policy or could be personalized based on off-line configuration or on the usage history of the user. However the user may override the default values at the session request time, using the quality profile window. The user may also change the quality profile during the life-time of the session using the session window. The UMDP requires that the utility value of Gold is larger than that of Silver, and the utility value of Silver is larger than that of Bronze.

When a session is admitted, the UMDP creates a new window called the session window. Figure 6.3 shows three session windows in their normal form. The window shows the current service level (Gold, Silver or Bronze) of the session. The drop button is used to terminate the session. The 'Details' button changes the session window to its detailed form as shown in Figure 6.4. Here the session window shows the current quality profile and resource usage of the session. Using this window, the user of an existing session may change the quality profile, and the system will adapt to this change. This is used to demonstrate the capability of the Utility Model to adapt to dynamic changes in user preferences. However, once a session is admitted, the user is not allowed to change the minimum acceptable quality level or the corresponding utility value, because allowing such a change would invalidate the admission of the session, and the session would have to go through another admission control procedure with no guarantee of success.

[3]For most of our discussion in this chapter, we assume the service provider example where the objective is to maximize system revenue subject to system resource constraints. Here session utilities denote the amounts paid by the users, and system utility denotes the total revenue earned by the system.

```
/* Procedures for UMDP implementation */

/* global variables */
solution *S;
int m=3;                              /* 3 resources */
int d=3;                              /* 3 choices for each profile */
int n;                                /* number of live sessions */
int R[3];                             /* total resource */
int l2r[9][3];                        /* QoS level to resource map */

procedure UMDP ()
{
    n=0;                              /* start with no session */
    S→U = 0;                          /* system utility */
    S→Rr=[0 0 0];                     /* reserved resource */
    S→Ra=[0 0 0];                     /* allocated resource */
    read_resmap(l2r);                 /* read QoS-resource map */
    read_resource(R);                 /* get total resource vector*/
    do{
        e = read_event();
        switch (e→type){
            case "req":               /* session request event */
                req_session(e);
            case "drop":              /* session drop event */
                drop_session(e);
            case "rchange":           /* resource change event */
                resource_change(e);
            case "pchange":           /* QoS profile change */
                profile_change(e);
        }
    } while(e→type == "quit")
}
```

Figure 6.5: *Pseudocode for UMDP implementation.*

```
procedure req_session(event e)
{
    session *s;
    solution *Sd;
    read_profile(s);
    if(S→Rr + l2r(s→Bronze) > R){
        print "reject for insufficient resource";
        return;
    }
    enlarge_MMKP(s);
    Sd=solve_MMKP(S, n + 1);
    if(Sd→U <= S→U){
        reduce_MMKP(s);
        print "reject for no profit";
        return;
    }
    S→Rr = S→Rr - l2r[s→Bronze];
    n = n + 1;                          /* admit new session */
    S = Sd;                             /* accept new solution */
}
```

```
procedure drop_session(event e)
{
    session *s;
    solution *Sd;
    s = e→sid;
    S→Rr = S→Rr + l2r[s→Bronze];
    reduce_MMKP(s);
    Sd=solve_MMKP(S, n-1);
    n = n − 1;
    S = Sd;
}
```

Figure 6.6: *Pseudocode for request session and drop session events.*

```
procedure resource_change(event e)
{
    int Rd[3];
    solution *Sd;
    read_reasource_change(Rd);
    if(Rd < S→Rr){
        print "panick: resource cannot be less than reserved";
        return;
    }
    R = Rd;
    Sd=solve_MMKP(S, n);
    S = Sd;
}
```

```
procedure profile_change(event e)
{
    session *s;
    solution *Sd;
    s = e→sid;
    read_profile_change(s);
    change_MMKP(s);
    Sd=solve_MMKP(S, n);
    S = Sd;
}
```

Figure 6.7: *Pseudocode for resource change or profile change events.*

6.2 The UMDP Implementation

The UMDP demonstrates the capability of the Utility Model to provide a unified solution of the session admission and quality adaptation problems within a multisession multiresource system. It is implemented using a mix of C and Tcl/Tk languages [39]. We used C for the main computation, and Tcl/Tk for the graphical user interface. We have two implementations of the UMDP based on the two solutions of the MMKP: UMDP1 using the optimal algorithm BBLP, and UMDP2 using the heuristic HEU.

In section 3.3.1, we discussed how the adaptive multimedia problem can be expressed as an MMKP instance. In our prototype implementation, the system maintains an MMKP instance where the number of existing sessions in the system represents the size of the MMKP instance. The system also maintains a system solution data-structure (S) which consists of the following fields: U indicating the system utility, Rr indicating the reserved resource vector, and Ra indicating the allocated resource vector.

Pseudocodes of Figures 6.5-6.7 present the implementation of the UMDP[4]. Figure 6.5 implements the main loop of the prototype, and Figures 6.6-6.7 implement the processing of the different events: request-session, drop-session, change-resource, and change-profile.

Request-session Event:

When a new session is requested, the UMDP first reads the quality profile of the session. Each session chooses three desired service levels: Gold, Silver and Bronze. Each service level is specified using a quality level and a utility value.

[4] In this chapter, UMDP implies both UMDP1 and UMDP2 unless we mention the specific MMKP solution used.

Here the utility value denotes the amount of revenue the user has offered to pay or has agreed to pay for the quality level.

After the quality profile is obtained, the UMDP first performs a resource mapping to map the quality profile to a resource profile according to Table 6.1. Then the system performs an admission test to check whether the system has enough unreserved resources[5] to guarantee the minimum acceptable quality to the new session. If the answer is affirmative, the system enlarges the MMKP instance, and solves the enlarged MMKP instance.

If the new system solution provides higher system utility than the current system utility, then:

- the session is admitted,
- necessary resources are reserved to guarantee the minimum acceptable quality for the session, and
- the new system solution becomes the current system solution.

Here new system solution implies new resource allocation, and therefore new operating quality, for the existing sessions.

However if the new solution does not provide higher system utility than the current system utility, then:

- the session is rejected,
- the MMKP instance is reduced back to the original size (the size before the new request-session), and
- the system solution remains unchanged.

[5]Here unreserved resources mean total resources minus reserved resources.

Drop-session event:

When an existing session is dropped, the allocated resources (both reserved and non-reserved) are returned to the system, the MMKP instance is reduced by one, the modified MMKP instance is solved, and the new system solution becomes the current one.

Change-resource event:

The system does not allow reduction of resource quantity below the reserved quantity because otherwise the system can no longer satisfy the quality contracts with the existing sessions. When the amount of resource available to the system changes, the MMKP instance is changed. The system solves the new MMKP instance, and the new system solution becomes the current one.

Change-profile event:

The UMDP also adapts to changes of the quality profile (service levels and/or utility values) of an existing session. However, it does not allow changes of the minimum acceptable quality level (i.e. Bronze service level), and the corresponding utility value. When the profile of a session is changed, the new MMKP instance is solved, and the new system solution becomes the current one.

6.3 Tests with Random Sequences of Events

In this section, we test the UMDP using randomly generated sequence of events, and evaluate its performance[6]. We compare the performance of the UMDP with that of a

[6]For random number generation, we used Linux random() function which uses a non-linear additive feedback pseudo-random number generator with normal dis-

simple reservation model prototype (SRMP). We use the system utility provided by the prototype under test as the main performance index. Since in the Utility Model, the goal of the adaptive system is expressed as a constrained maximization problem of the system utility function, a higher system utility obtained by a prototype implies better performance. We also compare the prototypes in terms of system utility and computation time.

The SRMP is based on a reservation based QoS model with no adaptation. Here all resource allocation is done by reservation. Thus no distinction is made between reserved and allocated resources. Once a session is admitted with certain amounts of resources allocated to it, the system cannot reduce or increase the amounts until the session terminates.

For our implementation of the SRMP, we reused the concepts of quality profile and quality-resource mapping from the UMDP. When requesting a new session, the user provides a quality profile to specify her requirements. The system maintains the status of the amount of free (not allocated to any session) resources. When it receives a new session request, it admits the session at the highest quality level which can be supported by the amount of free resources. The system allocates the appropriate amount of resources to the session, and the amount of free resources is reduced accordingly. When a session is dropped, the system revokes the resources allocated to this session which adds to the amount of free resources.

The use of SRMP for our evaluation experiments may be justified as follows.

- In its simplest form where there is only one service level, SRMP is like the current telephone system. In this system, a session, called a 'call' in telephony, is admitted only when the system is capable of reserving enough resources to

---

tribution.

support this session. When a session terminates, the resources are returned to the system.

- The SRMP expresses the current trend of B-ISDN systems, such as ATM, where a user may specify a desired quality of service for a session. For example, to request a session, a call in B-ISDN terms, the user may specify the required bandwidth of the session. However, once a session is established, the bandwidth cannot be changed during the life-time of the session.

- In its general form the SRMP allows the user to specify multiple choices for desired quality levels, and the system may choose the most suitable one for the current system state. In this sense, SRMP is more generalized and flexible than the B-ISDN model.

To compare the UMDP with SRMP, we tested their performance of the models using randomly generated (temporal) sequences of events.

6.3.1 Event Sequence Generation

We have implemented the event generation program 'genevents', which generates quasi-random events using four input parameters: number of events n, average size of time steps t_s, average size of utility steps u_s, and average duration of a session d_s. Here the parameter t_s is used to control the average time-difference between two consecutive session requests, and the parameter u_s is used to control the average difference between the utility values of two consecutive service levels of a session. The parameter d_s is used to control the average duration between the request and drop of a session. Table 6.2 shows the sequence of events produced by command 'genevents 10 5 10 40'. Each row indicates a session request with its quality profile specification and the session duration. For example, event 1 describes a session request event triggered at time 2 with quality levels (3 4 5) and corresponding utility values (4 24 38), and duration 24 time units. Thus if the session is admitted, it will trigger a session drop event at time 26.

Seq.	Time	QoS Profile			Utility Value			Duration
no.	t	$q[0]$	$q[1]$	$q[2]$	$u[0]$	$u[1]$	$u[2]$	d
1	2	3	4	5	4	24	38	24
2	8	1	5	6	17	27	36	43
3	10	1	3	6	8	23	38	11
4	17	2	4	7	12	22	36	58
5	27	1	2	3	2	22	32	51
6	29	3	6	9	16	32	42	71
7	35	6	7	9	13	27	28	25
8	40	1	4	9	19	23	39	75
9	43	1	3	9	2	14	34	16
10	52	1	3	4	3	16	26	56

Table 6.2: *Random sequence of events generated by 'genevents 10 5 10 40'.*

We note that the quality to utility mapping is generated randomly which may vary from instance to instance. For example, for event 1, quality level 3 is mapped to a utility value of 4, whereas for event 6, quality level 3 is mapped to a utility value of 16.

6.3.2 Test Algorithm

Our objective here is to compare the temporal variation of the system utilities provided by the three prototype implementations UMDP1, UMDP2 and SRMP

```
/* Algorithm to test the prototypes: UMDP and SRMP */

procedure TestPrototype()
{
   read_events(list);                      /* put events into link list */
   while(1){
        e = extract_min_t_event(list);
        if(e==NULL) break;
        t = e→t;
        if(e →type == "req"){              /* request event */
             s = req_session(e);
             if(s != NULL)                 /* session admitted */
                  insert(t + d, s);
        }
        else                               /* drop event */
             drop();
        print "t  U  e";
   }
}
```

Figure 6.8: *Procedure* TestPrototype*: Algorithm for testing prototypes.*

using random sequences of events. A high level description of the algorithm used for testing UMDP and SRMP is given in Figure 6.8. We use a doubly linked list to implement the time driven simulation algorithm. The linked-list uses time t as the key, and elements are maintained in such a way that the key is nondecreasing from head to tail. Two operations required are insertion of a node, and extraction of the node with a lowest key value. Events in the list may be of two types: session request event ("req") or session drop event ("drop"). The session request events are obtained from the input sequence of events, and if the session is admitted, a session drop event is inserted by the system.

The prototypes UMDP1, UMDP2 and SRMP are invoked using command formats 'umdp1 r1 r2 r3', 'umdp2 r1 r2 r3' and 'srmp r1 r2 r3' where r1, r2 and r3 are integers indicating total amount of available resource for resource 1, resource 2 and resource 3 respectively.

The algorithm TestPrototype works as follows. At first the input sequence of events is read into the linked-list. The processing of these events proceeds in a loop until the list is empty. In each iteration, the system extracts the event with the lowest key from the list, and checks the type of the event and processes it using the prototype under test (UMDP1, UMDP2 or SRMP).

6.3.3 Test Results

Tables 6.3 provides a temporal trace of the events, and their effect on the system utility obtained by the prototypes UMDP1, UMDP2 and SRMP for the sequence of events given in Table 6.2 and for a value of 60 for each of the three system resources. The temporal variation of the system utility is plotted in Figure 6.9.

We note that due to difference in admission policies of the implementations[7], it is possible that, one prototype admits a session, but another rejects it. For example, UMDP1 and UMDP2 admitted session 8, but SRMP rejected it.

In Table 6.3, we note that the system utilities obtained by UMDP1 and UMDP2 is always higher than that of SRMP. We also note that, even though UMDP2 uses the heuristic Heu, its performance is comparable to that of UMDP1 using the optimal algorithm Bblp. UMDP2 provided optimal system utility most of the time, and provided close to optimal system utility rest of the time (that is, during periods 10–21 and 26–27).

[7] This is because admission decisions depend on the implementation.

time	UMDP1		UMDP2		SRMP	
t	V	Event	V	Event	V	Event
0	0	startup	0	startup	0	startup
2	38	session 1 added	38	session 1 added	38	session 1 added
8	74	session 2 added	74	session 2 added	74	session 2 added
10	97	session 3 added	93	session 3 added	82	session 3 added
17	105	session 4 added	100	session 4 added	94	session 4 added
21	87	session 3 dropped	87	session 3 dropped	86	session 3 dropped
26	63	session 1 dropped	58	session 1 dropped	48	session 1 dropped
27	90	session 5 added	90	session 5 added	80	session 5 added
29	103	session 6 added	103	session 6 added	96	session 6 added
35	103	session 7 rejected	103	session 7 rejected	96	session 7 rejected
40	122	session 8 added	122	session 8 added	96	session 8 rejected
43	122	session 9 rejected	122	session 9 rejected	96	session 9 rejected
51	105	session 2 dropped	105	session 2 dropped	60	session 2 dropped
52	115	session 10 added	115	session 10 added	86	session 10 added
75	109	session 4 dropped	109	session 4 dropped	74	session 4 dropped
78	81	session 5 dropped	81	session 5 dropped	42	session 5 dropped
100	49	session 6 dropped	49	session 6 dropped	26	session 6 dropped
108	39	session 10 dropped	39	session 10 dropped	0	session 10 dropped
115	0	session 8 dropped	0	session 8 dropped		

Table 6.3: *Output traces of prototypes UMDP1, UMDP2 and SRMP on random sequence of events presented in Table 6.2.*

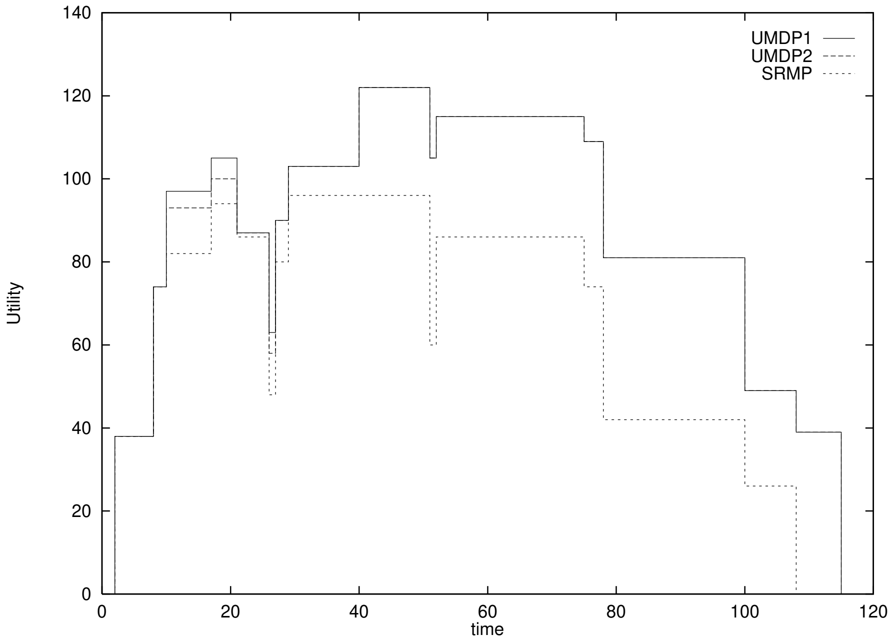


Figure 6.9: *Temporal variation of system utility obtained by UMDP1, UMDP2, and SRMP for the events of Table 6.2.*

The utility-axis of this figure indicates utility or revenue (e.g. dollars) per unit time, and the area under the system utility curve between two time instants t_1 and t_2 indicates the revenue earned in this time period. Thus in revenue terms, UMDP1 and UMDP2 will generate more revenue than SRMP for the same amounts of system resources. We measured the system revenues for the three implementations using Table 6.3. Here UMDP1 produced a revenue of 10152, UMDP2 produced a revenue of 10099, and finally SRMP produced a revenue of 7468. We note that the revenue produced by UMDP2 is very close to that of UMDP1, and either of these two is significantly higher than that of SRMP.

Figure 6.10 shows the variation of system utility for a random sequence of 100 events. Here UMDP1 produced a revenue of 735401, UMDP2 produced a revenue of 728276, and finally SRMP produced a revenue of 589619.

We note that, during certain periods, the system utility of UMDP2 exceeds that

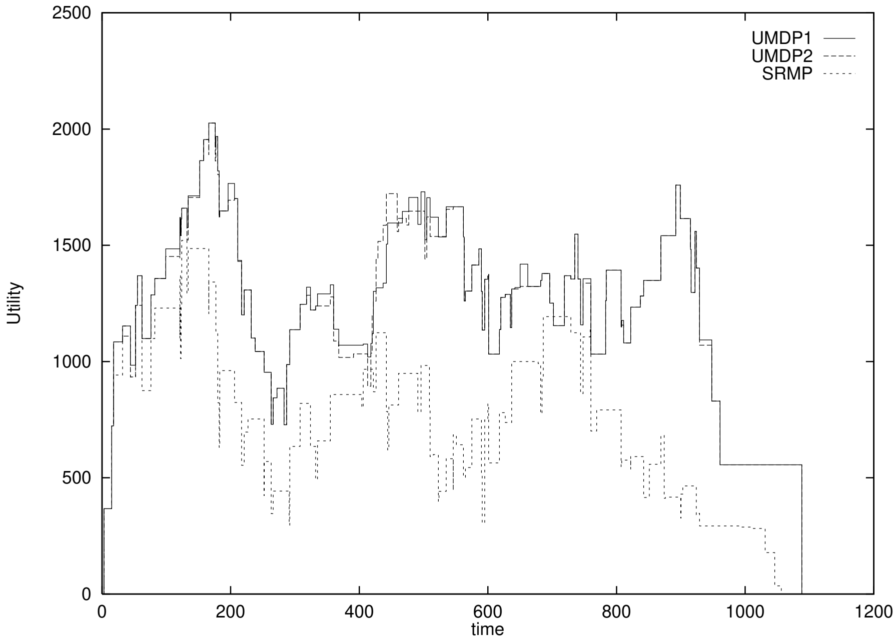


Figure 6.10: *Temporal variation of system utility obtained by UMDP1, UMDP2, and SRMP for events generated by 'genevents 100 10 100 100'.*

of UMDP1. This may be misleading, because UMDP1 uses an optimal algorithm for solving the MMKP, whereas UMDP2 uses a heuristic solution. However this is not unexpected due to the difference of admission decisions: The sessions admitted by UMDP1 may be different from those of UMDP2. Thus at certain times, the MMKP instance for the UMDP1 may be different from that of the UMDP2, and the system utility provided by UMDP2 may not always be smaller than that UMDP1.

We have also compared the total computation time required by the implementations to process the sequences of events. For the sequence of 10 events, the computation times required by implementations UMDP1, UMDP2 and SRMP are 28411, 1289 and 331 microseconds, respectively. For the sequence of 100 events, the corresponding numbers are 1856318, 24695 and 2947.

Table 6.4 compares the performance of UMDP1, UMDP2 and SRMP in terms of system revenue and computation time. Here performance numbers are normalized

Prototype	Sequence of 10 events		Sequence of 100 events	
	system revenue	computation time	system revenue	computation time
UMDP1	1.00	1.000	1.00	1.000
UMDP2	0.99	0.045	0.99	0.013
SRMP	0.74	0.011	0.80	0.002

Table 6.4: *Performance of UMDP1, UMDP2 and SRMP in terms of system revenue and computation time. Here performance numbers are normalized using those of UMDP1.*

with respect to the relevant numbers of UMDP1. We note that the system revenue provided by UMDP2 is 99% of the system revenue provided by UMDP1, and the computation time of UMDP2 is less than 5% of the computation time of UMDP1. On the other hand, the system revenue provided by SRMP is significantly lower than that of either UMDP1 or UMDP2.

We note that UMDP2 produced at least 19% more revenue than that of SRMP, and the computation time of UMDP2 is seven times that of SRMP. Since the computation time is still in the order of milliseconds, this extra computation time appears to be well justified by the extra revenue it earns.

6.4 Summary

In this chapter, we have described the Utility Model Demonstration Prototype (UMDP). This demonstrates the capability of the Utility Model to provide a unified solution of the session admission and quality adaptation problems within a multisession multiresource system.

We have two implementations of the UMDP: UMDP1 using the optimal algorithm Bblp, and UMDP2 using the heuristic Heu. We tested the UMDP with random sequences of events, and present output trace of the program. We compare the UMDP with a simple reservation model prototype (SRMP), and show that the system utility achieved by the UMDP is significantly higher than that of the SRMP. For applications such multimedia service providers, it means that UMDP will generate more revenue than SRMP for the same amounts of system resources.

Chapter 7

Conclusion

In this chapter, we summarize our contributions, and list future work and work in progress.

7.1 Contributions

The major contributions of this dissertation are as follows:

1. *Utility Model for AMS:*

   We have presented the Utility Model – a mathematical model to capture the dynamics of adaptive multimedia systems. This model is based on the concepts of quality profile, quality-resource mapping, session and system utility, and resource constraints. In this model, each session provides a quality profile which is a set of operating qualities arranged from the minimum acceptable quality to the maximum desired quality. Any operating quality may be mapped to the required resources using a quality-resource mapping, and to a session utility using a quality-utility mapping. The goal of the adaptive multimedia problem is then to maximize some system utility, subject to the system resource constraints. This model formulates the adaptive multimedia problem as the multiple-choice multi-dimension 0-1 knapsack problem (MMKP). It incorporates the dynamics of resource management and the interplay of media components of the existing sessions.

The Utility Model has the following features:

- It provides a unified and computationally feasible way to solve the admission problem of new multimedia sessions, dynamic quality adaptation and integrated resource allocation problems of existing sessions.
- This model provides an integrated approach to allocate all the system resources which enables integrated resource management.
- The concept of quality profile allows the users to specify their service requirements in a flexible manner. For example, the user request may be like "Silver or better quality" or "Gold quality always".
- The users may negotiate with the system to find a suitable (and personalized) quality-utility mapping.
- The model allows flexible management policies for service providers. The management policies of a service provider may either be expressed indirectly in the system utility function and parameters, or they may be expressed as additional system constraints.
- It provides increased system utility (e.g. revenue or profit) as compared to static reservation approach.

Although the Utility Model was developed to capture the dynamic behavior of AMSs, it is very general in nature, and may potentially be applied in many diverse application. For example, it may be used to model single-seller multiple-buyer *dynamic markets* where each buyer provides a set of choices, and the seller finds the appropriate choice for each buyer in order to optimize the seller's objective. For example, it may be used to maximize the revenue of a multimedia service provider or to model a dynamic airlines system where a client may have a customized ticket with business class seat and economy class meal.

2. *Solutions of MMKP:*

   The MMKP is one of the harder and less well-studied variations of the 0-1 Knapsack Problem. We presented two solutions for the MMKP: the branch and bound algorithm BBLP and the heuristic HEU. Algorithm BBLP produces the optimal solution, whereas the heuristic HEU provides a fast but near-optimal solution. We have analyzed the computational complexity of the heuristic HEU. We have reported computational experiences, and have compared the two approaches for practical applications, finding that HEU solutions are usually within 4% of the optimum but at a much reduced computational cost. HEU is suitable for time-critical applications such as real-time admission and quality adaptation decisions in multimedia systems.

3. *Padma End-System Architecture:*

   We have presented the Padma Architecture – a system architecture for multi-session adaptive multimedia systems where the quality of each session is dynamically adapted to the changes of resource availability and user preferences.

This architecture has two novelties: (1) integrated and adaptive management of system resources based on the Utility Model, and (2) the use of metaspaces to encapsulate the machinery of quality adaptation. The former provides improved resource utilization and dynamic quality adaptation, and the latter provides the application programmers freedom from the concerns of low-level resource management issues while developing multimedia applications.

4. *Prototype Implementation and Evaluation:*

   We have implemented a GUI-based Utility Model Demonstration Prototype (UMDP) to demonstrate its capability to handle admission control, quality adaptation and integrated resource allocation in a unified way. We have evaluated the performance of UMDP using random sequences of events, and compared the results with those from a simple reservation model prototype (SRMP). We have shown that the system utility achieved by the UMDP is significantly higher than that of the SRMP. For applications such as multimedia service providers, it means that UMDP will generate more revenue than SRMP from the same amounts of system resources.

7.2 Future Work and Work in Progress

This work has opened many problems and issues which require further research. Let us list some of them.

- *Resource Mapping:* Further research is required for mapping operating qualities to required resources. Ongoing research at Keio University and at Washington University at St. Louis is addressing this problem [38, 15].

- *Design of Adaptive Library and Applications:* We and our friends at the PANDA Laboratory have designed an adaptive multimedia object library. The library has been partially implemented, and we are in the process of incorporating some of the concepts of the Utility Model to support dynamic quality adaptation based on the network statistics collected by a statistics server. Here our goal is to demonstrate dynamic quality adaptation using real multimedia applications, such as the AVTS system described in section 5.4.

  We have implemented the multimedia library and the Utility Model on a best-effort datagram-based network using IP and RTP[1]. We have implemented a prototype system where the model is used to adapt quality of two contending sessions: a video transmission system and a ftp session [6]. In future, it will be interesting to port some of these implementations to a reservation-based network such as ATM. ATM network is particularly interesting because it has the provision for quality of service specification and guarantee.

- *Implement an Adaptive Multimedia Server:* One good way to demonstrate the features of the Utility Model will be to implement it in an adaptive multimedia server (AMSrvr) such as video-on-demand server, where the quality of real multimedia streams are dynamically adapted based on the user choices and resource availability.

  Our research group at the PANDA Laboratory is currently working on such an implementation. We assume a distributed client-server model where the clients execute in the users' machines, and the server executes in the service provider's machine. When an user starts a session, the server initiates a quality of service agent (QAgnt) at the server machine who enforces the quality of

[1]RTP: Real Time Protocol.

service adaptation of this session. We note that all the admission control and adaptation issues will be handled by a single server.

For our implementation, we are assuming an IP-based best-effort service in our network. Thus for the network bandwidth resource, we are working on a reactive adaptation model, where the system monitors the availability of the network bandwidth and adjusts the usage of other resources and/or quality of service parameters to adapt to the changes. This approach can be contrasted with a pro-active adaptation model where resources can be reserved, and the reservation can be policed and enforced using predictive scheduling.

- *Apply to Economic Systems:* The Utility Model may potentially be extended to capture dynamic markets where the price of a resource changes dynamically with change of demand and supply. In the Extended Utility Model (EUM), the system consists of a set of user agents, a provider agent and a set of resources. The objective of the user agents is to maximize utility-price ratio. On the other hand, the objective of the provider agent is to maximize system utility (revenue or profit) subject to the resource constraints and user constraints. Here the main problem is to allocate resources in order to resolve the conflicting objectives of the users and the provider. This will also involve designing a protocol for dynamic negotiation between the user agents and the provider agent. The EUM may farther be extended to incorporate a set of providers instead of just one provider.

- *Extend the Utility Model for Distribution:* Currently, the Utility Model incorporates the dynamic behavior of multiple sessions within a single end-system. It will be interesting to see how the model has to be extended to apply to distributed systems whose resources are scattered around a communication network.

Currently, we are working on a two layer model where the higher layer deals with allocation of distributed resources, and the lower layer handles the allocation of local resources. Here the idea is to use one Utility Model engine (UME) at each site. An UME manages the resources local to its site, and communicates with other UMEs to handle the distributed resources. Such an extension has to address the following issues:

- What protocol should a UME use to talk to each other?
- How do UMEs maintain the dynamic state of the overall system?
- How do we formulate the adaptive multimedia problem with both local and distributed resources?
- Should we use a simple client-server model where the higher layer adaptation computation is done at a single server? Or should we use a distributed peer-peer model where the higher layer adaptation computation is performed using some distributed algorithm?

- *Monitoring and Signaling:* In any adaptive system, there has to be some way of signaling such that the system knows when and how to react to dynamic state changes. For example, if the network is congested or if it fails to meet a certain bit-rate, this information must be passed back in order to achieve any adaptation. The design of a monitoring and signaling system includes the design of required protocols and the corresponding mechanisms. Monitoring is also required for a connection user to check that the provider is providing the quality of service as per commitment.

- *Eventual Fair Share Scheduling:* The Utility Model addresses the dynamic resource allocation problem within multisession multimedia systems. It tells how the system resources should be shared among the live sessions. However it cannot tell when and how to schedule the system resources in order to implement the resource allocation. Thus we require an adaptive resource scheduling algorithm for a multiresource multisession system. For example, we consider a department of an office or a research laboratory. The system is required to support a mix of QoS loads such as multimedia conferencing, and non-QoS (best-effort) loads such as electronic mail. To support the QoS loads, the system must support reservation-based resource allocation and scheduling. The system should also be fair to its users. Now what does fairness mean in the context of reservation-based resource scheduling?

  To solve the resource scheduling problem in adaptive systems we are working on an idea which we call *eventual fairness*. In eventual fairness, the system implements a policy which is fair to its users if the resource allocation is viewed over a long period of time, for example an hour or a day. It is contrary to the traditional fairness policy where the resource allocation tends to be fair at all times. For our scheduling policy, we are working on an adaptive algorithm using the concepts of reservation tokens and dynamic priority. At any time, the priority of a task is determined dynamically based on the importance of the user, status of the user's token account, user's priority tag (Gold, Silver or Bronze), and system load.

- *Modeling the Distributed Virtual Worlds:* The Utility Model may be extended to distributed virtual worlds where the consistency level among virtual worlds is adapted, based on available resources. For example, when enough bandwidth is available avatars may send facial expression updates as well as position updates. However, when the network is congested, the avatars may only send position updates. In this context, an architecture for scaling virtual worlds is suggested in [28].

Bibliography

[1] R.D Armstrong, D.S. Kung, P. Sinha, and A.A. Zoltners. A Computational Study of a Multiple-choice Knapsack Algorithm. *ACM Transactions on Mathematical Software*, 9:184–198, 1983.

[2] C. Aurrecoechea, A. Campbell, and L. Hauw. A Review of Quality of Service Architectures. *ACM Multimedia Systems Journal*, November 1995.

[3] Robert Braham and Richard Comerford. Sharing Virtual Worlds. *IEEE Spectrum*, 34(3), March 1997.

[4] A. Campbell, G. Coulson, and D. Hutchinson. A Quality of Service Architecture. *ACM Operating Systems Review*, 24, April 1994.

[5] A. Campbell, G. Coulson, and D. Hutchison. Supporting Adaptive Flows in Quality of Service Architecture. *ACM Multimedia Systems Journal*, 1996.

[6] Lei Chen. Utility Model Applied to Layered-coded Sources. M.Sc. Proposal, Department of Computer Science, University of Victoria, March 1998.

[7] Paul C. Chu. *A Genetic Algorithm Approach for Combinatorial Optimization Problems*. PhD Thesis, Imperial College of Science, 1997.

[8] Vašek Chvátal. *Linear Programming*. W.H. Freeman and Co., New York, 1983.

[9] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introductions to Algorithms.* The MIT Press and McGraw-Hill, 1990.

[10] G.B. Dantzig. Discrete Variable Extremum Problems. *Operations Research*, 5:266–277, 1957.

[11] G.B. Dantzig. *Linear Programming and Extensions.* Princeton University Press, Princeton, New Jersey, 1962.

[12] L. Delgrossi, Ch. Halstrick, D. Hehmann, R. G. Herrtwich, O. Krone, J. Sandvoss, and C. Vogt. Media Scaling for Audiovisual Communication with the Heidelberg Transport System. Technical Report 43.9305, IBM ENC Heidelberg, Heidelberg, Germany, 1993.

[13] K. Dudzinski and S. Walukiewicz. A Fast Algorithm for the Linear Multiple-choice Knapsack Problem. *Operational Research Letters*, 3:205–209, 1984.

[14] A. Eleftheriadis. *Dynamic Rate Shaping of Compressed Digital Video.* PhD Thesis, Columbia University, 1995.

[15] R. Gopalakrishnan and G. Parulkar. Efficient quality of service support in multimedia computer operating systems. Technical Report WUCS-94-26, Dept. of Computer Science, Washington University, St. Louis, 1994.

[16] Jean-François Huard, Ichiro Inoue, Aurel A. Lazar, and Hideaki Yamanaka. Meeting QOS Guarantees by End-to-End QOS Monitoring and Adaptation. In *Proceedings of the Fifth IEEE International Symposium On High Performance Distributed Computing (HPDC-5)*, Syracuse, NY, August 1996.

[17] E. A. Hyden. *Operating System Support for QoS.* PhD Thesis, Wolfson College, University of Cambridge, February 1994.

[18] ITU-T Recommendation I.321. BISDN Protocol Reference Model, 1990.

[19] Kiyokuni Kawachiya, Masanobu Ogata, Nobuhiko Nishio, and Hideyuki Tokuda. QoS Control of Continuous Media: Architecture and System Support. Technical Report RT0108, IBM Tokyo Research Lab, Kanagawa, Japan, April 1995.

[20] Kiyokuni Kawachiya and Hideyuki Tokuda. QoS-Thread: A New Execution Model for Dynamic QoS Control of Continuous-Media Processing. In *Fifth International Workshop on Network and Operating System Support for Digital Audio and Video, NOSSDAV 95*, Durham, NH, April 1995.

[21] Kiyokuni Kawachiya and Hideyuki Tokuda. QOS-Ticket: A New Resource-Management Mechanism for Dynamic QOS Control of Multimedia. In *Proceedings of Multimedia Japan '96*, pages 14–21, April 1996.

[22] Gregor Kiczales. Foil for the Workshop on Open Implementation, 1994. Available at http://www.xerox.com/PARC/spl/eca/oi/workshop-94/default.html.

[23] P.J. Kolesar. A Branch and Bound Algorithm for the Knapsack Problem. *Management Science*, 13:723–735, 1967.

[24] K. Lakshman and Raj Yavatkar. AQUA: An Adaptive End-System Quality of Service Architecture. In W. Effelsberg, O. Spaniol, A. Danthine, and D. Ferrari, editors, *High Spped networking for Multimedia Applications*, 1996. To appear. Also available at http://yellow.ccs.uky.edu/ lakshman/papers/dag.ps.

[25] Eugene L. Lawler. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart & Winston, N.Y., 1976.

[26] A.A. Lazar, S. K. Bhonsle, and K.S. Lim. A Binding Architecture for Multimedia Networks. *IEE Parallel & Distributed Computing*, 1995.

[27] A.A. Lazar, L.H. Ngoh, and A. Sahai. Multimedia Networking Abstractions with Quality of Service Guarantees. In *Proceedings of the SPIE Conference on Multimedia Computing and Networking*, pages 24–33, San Jose, CA, February 1995.

[28] Rodger Lea, Pierre G. Raverdy, Yasuhiko Honda, and Kouichi Matsuda. Scaling a Shared Virtual Environment, 1997. Available at http://www.csl.co.jp/person/rodger.

[29] M.J. Magazine, G.L. Nemhauser, and L.E. Trotter. When the Greedy Solution Solves a Class of Knapsack Problem. *Operations Research*, 23:207–217, 1975.

[30] M.J. Magazine and Osman Oguz. A Heuristic Algorithm for the Multidimensional Zero-One Knapsack Problem. *European Journal of Operational Research*, 16(3):319–326, June 1984.

[31] S. Martello and P. Toth. Algorithms for Knapsack Problems. *Annals of Discrete Mathematics*, 31:70–79, April 1987.

[32] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations.* John Wiley & Sons, Chichester, 1990.

[33] Martin Moser. Declarative Scheduling for Optimally Graceful QoS Degradation. In *Proceedings of IEEE Multimedia Systems*, Hiroshima, Japan, 1996.

[34] Martin Moser, Dušan P. Jokanović, and Norio Shiratori. An Algorithm for the Multidimensional Multiple-Choice Knapsack Problem. *IEICE Transactions on Fundamentals of Electronics*, 80(3):582–589, March 1997.

[35] K. Nahrstedt. *An Architecture for End-to-End Quality of Service Provision and its Experimental Validation.* PhD Thesis, Department of Computer and Information Science, University of Pennsylvania, August 1995.

[36] K. Nahrstedt and R. Steinmetz. Resource management in networked multimedia systems. *IEEE COMPUTER*, pages 52–63, May 1995.

[37] R.M. Nauss. The 0-1 Knapsack Problem with Multiple Choice Constraints. *European Journal of Operational Research*, 2:125–131, 1978.

[38] Nobushiko Nishio and Hideyuki Tokuda. QOS Translation and Session Coordination Techniques for Multimedia Systems. In *Proceedings of Sixth International Workshop on Network and Operating System Support for Digital Audio and Video*, Jushi, Japan, 1996.

[39] John K. Ousterhout. *Tcl and Tk Toolkit.* Addison-Wesley, 1994.

[40] S. Paek, P. Bocheck, and S.F. Chang. Scalable MPEG-2 Video Servers with Heterogeneous QoS on Parallel Disk Arrays. In *Proceedings of Fifth International Workshop on Network and Operating System Support for Digital Audio and Video*, Durham, New Hamshire, USA, 1995.

[41] Christos H. Papadimitriou and Keneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice Hall Inc., Englewood Cliffs, New Jersey, 1982.

[42] J. Pasquale. Filter Propagation in Dissemination Trees: Trading off Bandwidth for Processing in Conitinuous Media Networks. In *Fourth International Workshop on Network and Operating System Support for Digital Audio and Video, NOSSDAV 93*, Lancaster University, November 1993.

[43] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, Cambridge, UK, second edition, 1992.

[44] Henning Schulzrinne. Operating System Issues for Continuous Media. *Multimedia Systems*, 4:269–280, May 1996.

[45] Wei Shih. A Branch and Bound Method for the Multiconstraint Knapsack Problem. *Journal of the Operational Research Society*, 30:369–378, 1979.

[46] R. Steinmetz and K. Nahrstedt. *Multimedia: Computing, Communications, and Applications.* Prentice Hall, Inc., 1995.

[47] Ralf Steinmetz. Analyzing Multimedia Operating System. *IEEE Multimedia*, 2(1):68–84, Spring 1995.

[48] C. Topolcic. Experimental Internet Stream Protocol, Version 2 (ST II). Internet Network Working Group, RFC 1190, October 1990.

[49] Yoshiaki Toyoda. A Simplified Algorithm for Obtaining Approximate Solution to Zero-one Programming Problems. *Management Science*, 21:1417–1427, 1975.

[50] L. Wolf and R. G. Herrtwich. The System Architecture of the Heidelberg Transport System. *ACM Operating Systems Review*, 28(2), April 1994.

[51] Lars Cristian Wolf. *Resource Management for Distributed Multimedia Systems.* Kluwer Academic Publishers, Norwell, MA, 1996.

[52] Nicholas Yeadon. *Quality of Service Filters for Multimedia Communications.* PhD Thesis, Lancaster University, Lancaster, UK, may 1996.

[53] Yasuhiko Yokote. The Apertos Reflective Operating System: The Concept and Its Implementation. Technical Report SCSL-TR-92-014, Sony CSL, Japan, June 1992.

[54] Wenjun Zeng and Bede Liu. Rate Shaping by Block Dropping for Transmission of MPEG-precoded Video over Channels of Dynamic Bandwidth. In *Proceedings of Fourth ACM International Multimedia Conference*, Boston, 1996.

VITA

Surname: Khan — Given Names: Md. Shahadatullah

Place of Birth: Dhaka, Bangladesh — Date of Birth: April 14, 1967

Educational Institutions Attended:

Bangladesh University of Engineering and Technology, Dhaka	1986 to 1992
University of Victoria	1992 to 1997

Degrees Awarded:

B.Sc. (Electrical & Electronic Eng.)	
Bangladesh University of Engineering & Technology, Dhaka	1991
M.A.Sc. (Electrical & Computer Eng.)	
University of Victoria, Canada	1994

Honours and Awards:

Canadian Commonwealth Scholarship	1992 to 1997
University of Victoria President's Award	1992 to 1997

Publications:

1. Mohammod Kaykobad and Shahadat Khan, "On the Performance of Simplex Algorithm", I.P.E. Research Bulletin, BUET, Dhaka, Bangladesh, Volume 6, Pages 39–57, 1995.

2. Shahadat Khan and Kin F. Li, "Fast Graph Partitioning Algorithms", IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing, Pages 337–342, Victoria, BC, August 1995.

3. Eric G. Manning, Shahadat Khan, Rodger Lea, Gholamali C. Shoja and Michael

M.J. Zastre, "Metaspaces and Mobile Computing: Promises and Challenges", Invited Paper, International Conference on World Wide Computing and Its Applications, Tsukuba, Japan, Pages 350-362, March 1997.

4. Shahadat Khan, Kin F. Li and Eric G. Manning, "Padma: An End-System Architecture for Distributed Adaptive Multimedia", IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing, Victoria, Pages 105–108, August 1997.

5. Shahadat Khan, "Quality Adaptation in a Multi-Session Adaptive Multimedia System: Model and Architecture", Technical Report ECE-97-1, Department of ECE, University of Victoria, September 1997.

6. Shahadat Khan, Kin F. Li and Eric G. Manning, "Two Solutions for the Multiconstraint Multiple-choice Knapsack Problem", Technical Report ECE-97-3, Department of ECE, University of Victoria, September 1997.

7. Shahadat Khan, Kin F. Li and Eric G. Manning, "The Utility Model for Adaptive Multimedia Systems", The International Conference on Multimedia Modeling, Singapore, Pages 111–126, November 1997.

8. Shahadat Khan, Mahmood Chowdhury, Eric Manning and Kin Li, "Communication Algorithms for One Dimensional Hypercycles", National Conference on Computer and Information Systems (NCCIS'97), Dhaka, Bangladesh, Pages 283-289, December 1998.

9. Shahadat Khan, Kin F. Li and Eric G. Manning, "The Utility Model for Adaptive Multimedia Systems", Submitted to ACM Multimedia.

10. Shahadat Khan, Kin F. Li and Eric G. Manning, "Two Solutions for the Multiconstraint Multiple-choice Knapsack Problem", In preparation for journal submission.