Padma: An Architecture for Adaptive Multimedia Systems

Shahadat Khan, Kin F. Li and Eric G. Manning
Department of Electrical and Computer Engineering
University of Victoria
Victoria, BC, Canada V8W 3P6
{skhan,kinli,emanning}@ece.uvic.ca

Abstract

We present Padma *architecture – a system architecture for multi-session adaptive multimedia systems (AMSs) where the perceived 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 quality adaptation in AMSs, and uses metaspaces to encapsulate the machinery of quality adaptation. The former provides dynamic adaptation and improved resource utilization, and the latter provides the programmers of distributed multimedia applications freedom from the concerns of low-level resource management issues.*

1 Introduction

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. The quality of a service may be expressed using as a set of parameters, where each parameter relates to one property of the service [7]. For example, a video stream may have QoS parameters such as refresh rate, resolution and color depth (bits/pixel).

In a dynamic system environment where the load and available resources may change at run-time, multimedia applications have inherent potential to become adaptive. For example, one can have real-time video transmitted at a 30 frame/sec refresh rate when there is enough bandwidth available; or one may reduce the bandwidth requirement to half by reducing the rate to 15 frame/sec. How do we design a multimedia system to exploit the potential for adaptation of multimedia applications?

In this article, we present *Padma* – a system architecture for multi-session adaptive multimedia systems (AMSs) where the perceived quality of each session is dynamically adapted to changes of available resources and user preferences. The article is organized as follows. Section 2 presents some design issues. Section 3 presents the Padma system architecture with an example application. Section 4 discusses some related research, and, section 5 provides the conclusion.

2 Some 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, 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 architecture is based on the following design decisions:

- **Utility Model for Admission and Adaptation:** We use the Utility model for quality adaptation in multisession AMSs. In this model, each session specifies its quality preferences using a *quality profile* (from minimum acceptable quality to highest quality desired), and the system dynamically adapts the quality of individual sessions in order to maximize a *system utility function* (such as revenue) under a system resource constraint. The model provides a unified approach to both admission control of new

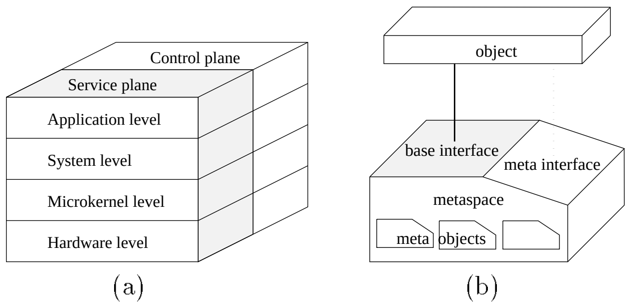

Figure 1: Service-control separation using metaspaces: implement service plane using base interface (service interface) and implement control plane using meta-interface (control interface).

multimedia sessions and the dynamic quality adaptation of existing sessions. Please refer to [4] for details on the Utility model.

- **Service-control Separation:** Service and control are distinct activities, and should be separated in the system architecture[1]. *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 1 (a) shows the 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 Metaspace:** 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. Metaspace concept was introduced at the Sony CSL for designing a flexible and dynamic operating system called Apertos [9]. 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 1 (b), a metaspace has two types of interfaces [5]: base interface for accessing the service functions and meta-interface for changing or controlling the service behavior.

  Metaspaces provide an elegant mechanism for implementing service-control separation[Figure 1]: service plane is implemented using base interface (also called service interface) and control plane is implemented using meta-interface (also called 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.

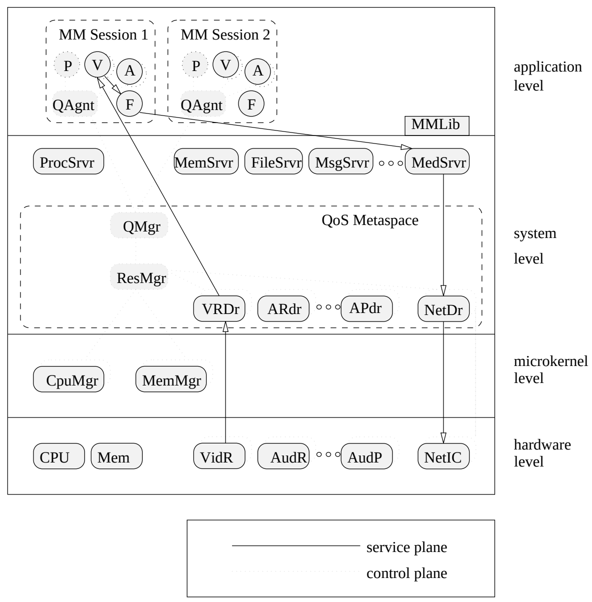

Figure 2: The Padma system architecture.

3 The Padma System Architecture

Figure 2 presents the Padma architecture for AMSs. The architecture consists of four levels:

1. Application Level: A multimedia library (MMLib) provides the media objects such as audio(A), video(V), media filters(F)[2], quality profile(P), and QoS agent(QAgnt).

2. System Level: It consists of servers such as QoS manager(QMgr), process server (ProcSrvr), memory

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

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

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 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 dotted lines for control objects and meta-interface (control interface) calls.

3.1 Quality Management Hierarchy

The Padma architecture uses a hierarchical quality management and adaptation scheme. In Figure 2, we observe three domains of quality management:

1. Application domain: The quality control and adaptation issues of each session are handled by a quality agent QAgnt. QAgnt negotiates with its peers and with the local QMgr to find an appropriate operating quality for the session. It finds and 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.

2. 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 deals with the admission of new sessions, and takes quality adaptation decisions using the Utility model. The QMgr's quality control decisions are then enforced by the ResMgr by sending suitable control signals to the resource managers such as CpuMgr, MemMgr, NetMgr and VRDr.

3. 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.

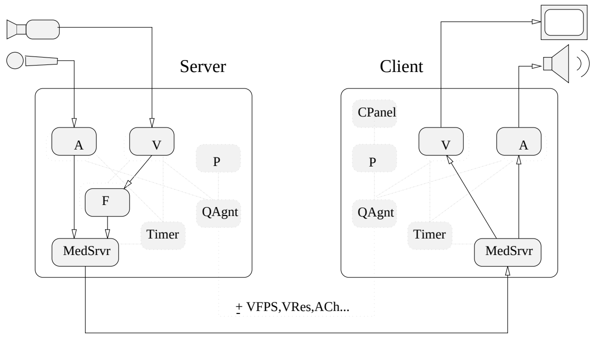

Figure 3: The client-server based AVTS system (Application level only).

3.2 An Example

Figure 3 illustrates an audio-visual transmission system (AVTS). It is 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.

When a user wants a new AVTS session, it triggers the following sequence of events:

- The client requests a new AVTS session with two media: audio and video.
- 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 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 find a suitable operating quality for the session.
- 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 levels of available resources which is then enforced by the the server and client QAgnts by proper adjustments 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.

4 Related Work

System design for distributed multimedia applications is a very active and popular research field [1, 8, 6]. Most of the architectures in the literature deal with the establishment and management of a single multimedia session. On the other hand, in Padma architecture, we target the quality adaptation problem in a multisession dynamic end-system. Two novelties distinguish the Padma architecture from others reported in the literature: (1) hierarchical quality management based on the Utility model, and (2) the use of metaspaces to encapsulate the machinery of quality adaptation.

The Padma architecture has some points of similarity with the Keio MMP architecture [3] because both architectures target the same general problem of adapting among multiple concurrent sessions. However the two architectures focus on complementary issues: the MMP architecture focuses on how adaptive quality can be enforced using reservation-based resource management, and the Padma architecture focuses on how adaptation decisions can be derived using the Utility model, and how adaptation can be abstracted from the application programmer using metaspaces.

5 Conclusion

We have presented Padma architecture for adaptive multimedia systems (AMSs) where the perceived 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 quality adaptation in AMSs, and uses metaspaces to encapsulate the machinery of quality adaptation.

Currently we are working on a prototype implementation of the Padma architecture. We are also working on the design and implementation of a multimedia object library. Finally we want to evaluate the Padma architecture and the multimedia library using some real multimedia applications such as distributed video conference.

6 Acknowledgments

We wish to thank Dr. Mario Tokoro of Sony Corporation, who first observed that metaspaces might provide an elegant way to encapsulate QoS mechanisms. We also thank our friends at the UVic Panda Lab for many stormy debates and improvement suggestions.

References

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

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

[3] 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.

[4] Shahadat Khan. Quality Adaptation in a Multisession Adaptive Multimedia System: Model and Architecture. Ph.D. Proposal, June 1997.

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

[6] 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.

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

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

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