REAL-TIME ISSUES IN DISTRIBUTED MULTIMEDIA SYSTEMS

Borko Furht
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, Florida 33431
email: borko@cse.fau.edu

Abstract

The fundamental characteristics of multimedia systems is that they incorporate continuous media, such as voice, video, and animated graphics. This implies the need for multimedia systems to handle data with strict timing requirements and at high rate. This article discusses important real-time issues in distributed multimedia systems including media synchronization, operating system support for continuous media, scheduling issues for on-demand multimedia applications, and real-time architectures of future interactive TV-converter boxes. Several solutions of these problems are presented in the article.

1. ARCHITECTURE OF A DISTRIBUTED MULTIMEDIA SYSTEM

A distributed multimedia system (DMS), shown in Figure 1, consists of one or more multimedia servers, a network, and a number of multimedia user stations (multimedia clients).

A multimedia server typically stores different media databases (such as video, audio, images, text, etc), and handles various functions applied on these databases. These functions include multimedia objects query, retrieval, integration, and others. The distributed multimedia systems can be interconnected by traditional local area networks, however the fiber optics and Broadband Integrated Services Digital Networks (BISDN) provide much better environment for transmitting continuous media.

Typical applications of distributed multimedia systems include video conferencing, video telephony, collaborative work, multimedia mail, and distance learning. Some recent applications include on-demand multimedia services, such as in entertainment, video news distribution services, distribution of video rental services, interactive television, and digital multimedia libraries.

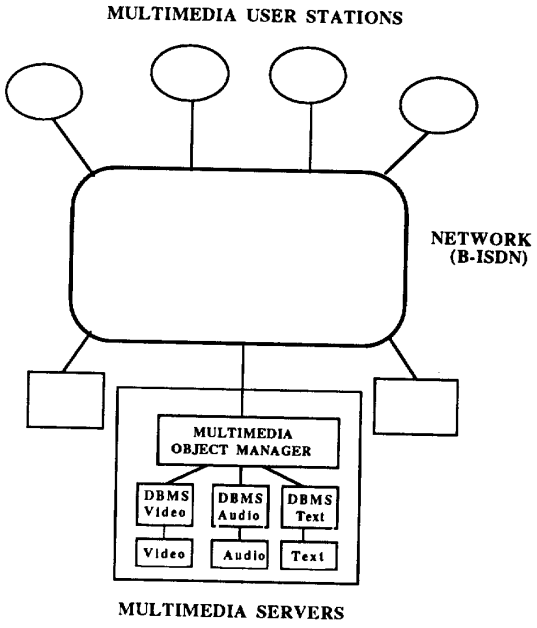

Figure 1. The architecture of a distributed multimedia system

In a typical distributed multimedia application, the multimedia data (e.g. audio and video) must be compressed at the server site, transmitted over the network to its destination, and decompressed and synchronized for playout at the receiving site. Continuous media (CM), such as video, audio,

0-8186-6420-7/94 $03.00 © 1994 IEEE

and animated graphics, are characterized by: (1) strict timing requirements, (2) high data rate, and (3) continuous transfer over long periods of time. Therefore, real-time techniques must be applied at various sites in distributed multimedia systems in order to meet these requirements. Table 1 presents elements of a distributed multimedia system and related real-time issues that must be addressed.

However, there is a significant difference between traditional real-time systems and real-time systems required to support multimedia data [1]. Traditional real-time systems are often static in nature, which means that the system is configured with a known workload which is never changed. Multimedia systems are typically dynamic, which assumes dynamic multiprogrammed workload. In such systems system overload is possible due to a long term commitment of resources. In processing continuous media this is an error that can be avoided by denying the resource allocation to the application. Unlike in traditional real-time systems, overloads in multimedia systems may be short term. In dynamic real-time systems the main issues are predictability and guaranteed access to resources. In the case of continuous media these guarantees are probabilistic rather than absolute. By knowing the applications it is possible to reduce the total demand on the system. For example, in a video phone application, it is easy to reduce the total demand on the system by reducing video quality and frame rate.

In this paper we will not address all the issues from Table 1. The emphasis is given to multimedia synchronization, operating system support, and on-demand multimedia services including the needs for real-time architecture of interactive cable converters. Compression/ decompression techniques and standards and real-time requirements for multimedia networks are studied in [2].

2. MULTIMEDIA SYNCHRONIZATION

One of the difficulties in supporting multimedia data delivery in a distributed multimedia system is the synchronization of related media. When multimedia data is transmitted via different channels and its synchronization is required, a synchronization techniques is necessary to ensure synchronous presentation or playout of these media. A typical example is a lip-synchronization of audio and video data, however synchronization of other media can also be required in a multimedia application.

TABLE 1
Real-Time Issues in Distributed Multimedia Systems

ELEMENTS OF DMS	REAL-TIME ISSUES
Multimedia compression and decompression	Development of real-time compression/decompression algorithms
Multimedia networks	High-speed, predictable network
Providing synchronized presentation of multiple media	Development of real-time synchronization algorithms
Operating systems	Operating systems support for CM - real-time scheduling - light weight processes/threads
Disk storage systems	Real-time disk scheduling for CM
Multimedia databases	Real-time data retrieval
On-demand services	Real-time program scheduling and caching
Interactive cable convertors	Real-time architecture

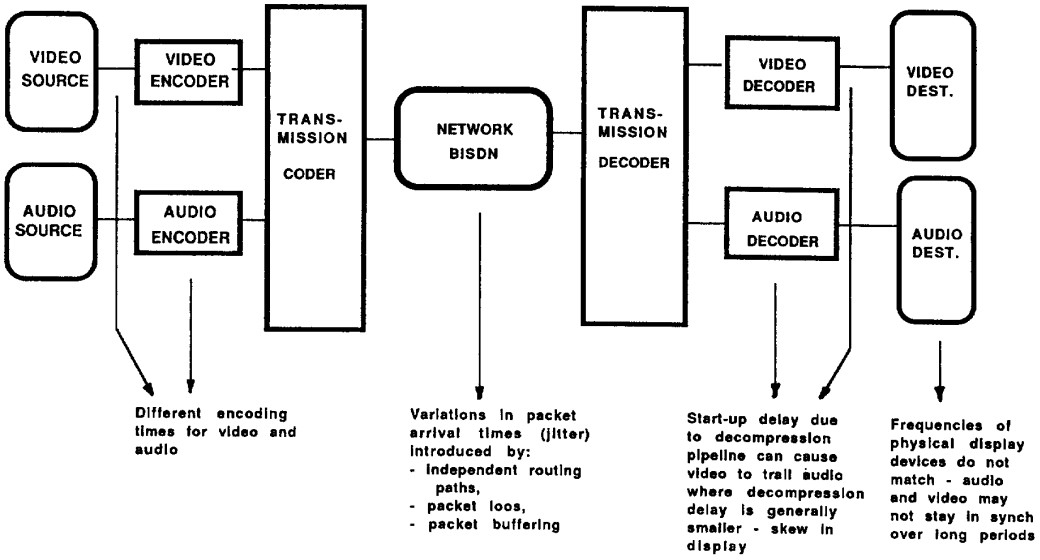

Figure 2. Video telephony system: various causes of asynchrony

Figure 2 illustrates an example of video telephony [3], and specifies various places within the systems which contribute to asynchrony. The task of a synchronization technique, which can be implemented in the network or in the receiver, is to eliminate all the variations and delays occured during the transmission of multiple media streams, and to maintain synchronization among these media streams.

For a such distributed multimedia system, an end-to-end- delay can be defined between the source and destination. This delay consists of all the delays created at the source site, network, and receiver site, and is slightly different for real-time video and audio, and for stored multimedia objects, as shown in Figure 3 [3].

When implementing a synchronization algorithm for a specific application, a requirement is specified for Quality Of Service (QOS) for multimedia communications. The QOS is defined as a set of parameters, which include: speed ratio, utilization, average delay, jitter, bit error rate, and packer error rate [3]. Figure 4 illustrates how some of these parameters can be calculated.

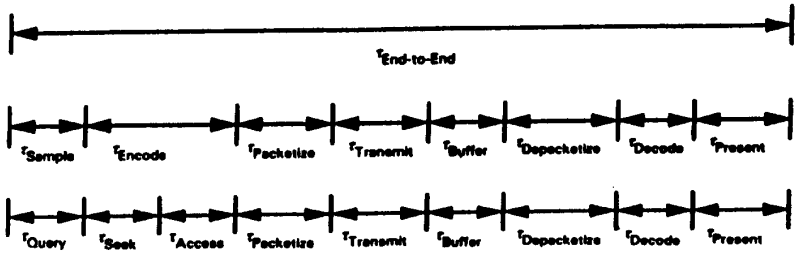

Figure 3. Definition of the end-to-end delay

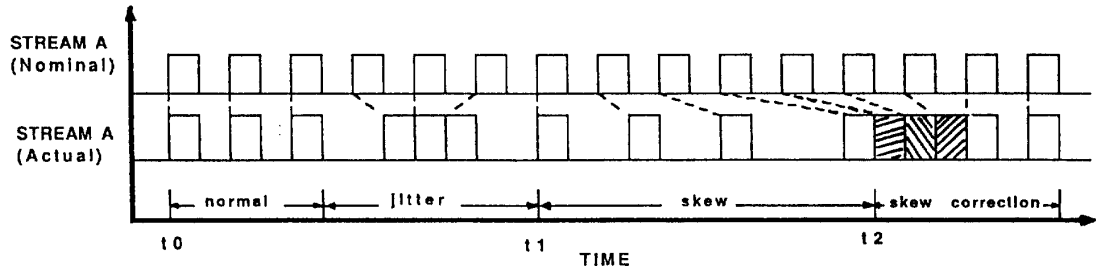

Figure 4. Nominal and actual streams: example of skews, jitter, utilization and speed [3]

The definitions of these parameters are as follows:

$$\text{Speed ratio} = \frac{\text{Actual presentation rate}}{\text{Nominal presentation rate}}$$

$$\text{Utilization} = \frac{\text{Actual presentation rate}}{\text{Available delivery rate}}$$

For the interval $\{t0,t1\}$ in Fig. 4 speed ratio = 6/6=1, while for the interval $\{t1,t2\}$ speed ratio =5/7=0.71. Ideally, both speed ratio and utilization are equal to 1. Duplication of frames will cause U>1, while frame dropping will cause U<1.

Skews is defined as the average difference in presentation times between two synchronized objects over n synchronization points [3]. If a data frame is duplicated, it causes the stream to retard in time (stream leg). Data losses results in gaps in playout and causes streams to advance in time (stream lead). In Fig. 4, for the time interval $\{t0,t1\}$ the skew is 4/6. Jitter is defined as the instantaneous difference between two synchronized streams. Figure 4 also illustrates how jitter can be corrected by dropping (or duplicating) frames.

Another two parameters which comprises QOS, Bit Error Rate (BER) and Packet Error Rate (PER), specify the required reliability of the network. For illustration purpose, the Quality of Service for video telephony and for JPEG video transmission are specified in Table 2.

The process of synchronization consists of: (a) evaluation of the temporal characteristics of the data streams to be synchronized, and (b) corrections off all delays and other anomalies [4].

Let's begin with the synchronization of a single event X, with the playout time instant π, playout interval μ, and total latency time Ω due to all the delays defined in Fig. 3. For this triple $\{\Omega,\pi,\mu\}$, synchronization consists of finding a time, called the control time T, such that $T \geq \Omega$, and then scheduling the retrieval at time T units prior to π

TABLE 2
SPECIFICATION FOR QUALITY OF SERVICE

	Video Telephony	JPEG Video Transmission
Speed ratio	1.0	1.0
Utilization	1.0	1.0
Average delay	0.25 s	0.2 s
Maximum jitter	10 ms	5 ms
Maximum BER	0.01	0.1
Maximum PER	0.001	0.01

will guarantee successful synchronization. Otherwise, if such a time T cannot be found, the deadline π will be missed. Timing of the single event synchronization is shown in Figure 5. From Fig. 5, the retrieval time, or packet production time f is defined as $f=\pi$-T.

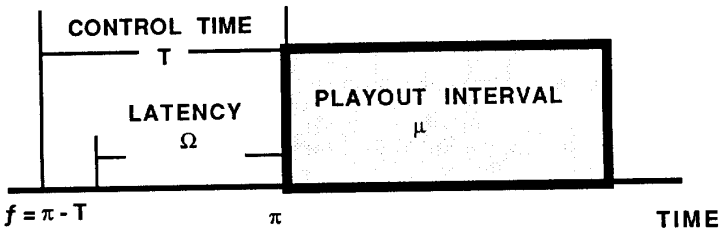

Figure 5. Timing for single event synchronization

The case of synchronizing one event is simply the problem of meeting a single real-time deadline, which is subset of the real-time scheduling problem [4]. The single event synchronization can be extended to the case of a single data stream synchronization, as illustrated in Figure 6, and further to the general case of multiple data streams synchronization [4].

Packets of data, created at a source and transmitted over the network, are received with a random distribution p(t). The reduction of arriving packet delay variance can be achieved by using buffering or some other techniques, so the distribution w(t) can be obtained, as illustrated in Fig. 6a. Then, the synchronization consists of calculating the control time T, based on multiple deadlines $\{\pi_i\}$, playout times $\{\mu_i\}$, and latencies $\{\Omega_i\}$. Time T represents the time required for buffering at the receiving site to smooth variations in latencies $\{\Omega_i\}$.

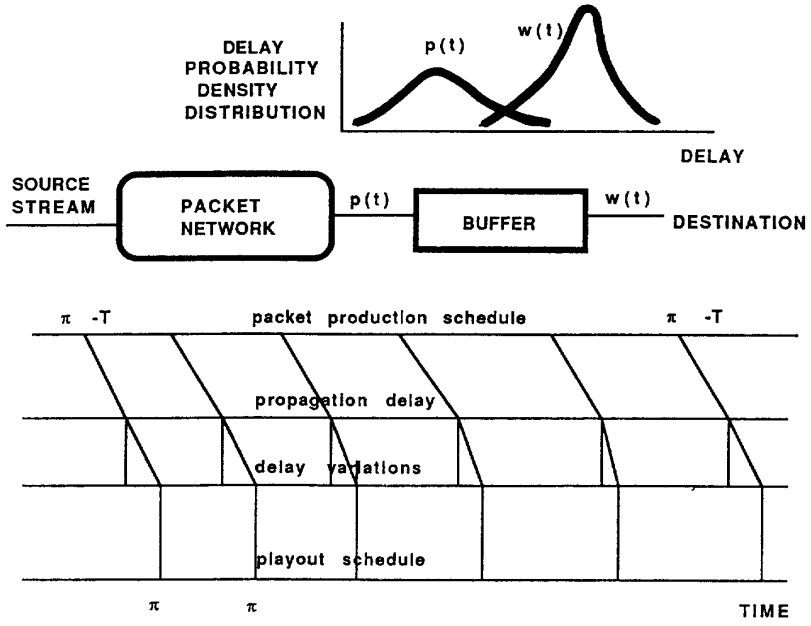

Figure 6. Single stream synchronization example: (a) the reduction arriving packet delay variance, (b) the creation of the playout schedule

Further, the case of synchronization of single streams of packetized audio and video can be extended to the general case of synchronization of multiple streams as well as nonstream data (e.g. still images and text) [4]. Then, various synchronization techniques have been proposed to determine the delay and buffering required in order to establish synchronization of multiple events for given packet loss probabilities and network delay distributions.

A synchronization technique, proposed in [4], uses an object composition Petri net (OCPN) model to specify synchronization and temporal relationships between data elements. This technique first applies a **serialize-net** algorithm to simulate an OCPN, and to determine deadlines for data retrieval and identify the necessary buffering. Once serialization is complete, a playout schedule is decomposed into subschedules for each resource by applying the **resource-decomposition** algorithm. Finally, after the net has been serialized and decomposed into classes, the control time is determined for each subschedule [4].

3. OPERATING SYSTEM SUPPORT FOR CONTINUOUS MEDIA

Traditional general-purpose operating systems are not well suited to handle continuous media. On existing operating systems, such as UNIX, integrated continuous media applications suffer from poor performance. The results, reported in [5], on ACME continuous media I/O server implemented on a Sun SPARCstation running SunOS 4.1, show that the system suffers from timing errors and lost data, when there is a concurrent system activity.

These problems are partly due to the overhead of user/kernel mechanisms by which user-level programs invoke system functions such as CPU scheduling and I/O functions [5]. This overhead includes user/kernel domain switches and mapping switches between different user virtual address spaces. For example, the UNIX asynchronous I/O mechanism requires up to ten domain switches and two mapping switches to read a block of data.

In general, a multimedia operating system (MMOS) should provide preemptive multitasking, easy expendability, format-independent data access, and support for real-time scheduling. The MMOS should be object-oriented and capable of synchronizing data streams. User interface must be highly sophisticated and flexible, supported at the OS level.

Several distinct approaches are proposed that can overcome the problems of traditional operating systems and that provide support for continuous media. One approach is based on modifying traditional operating systems, such as UNIX and incorporate functions needed to support continuous media [5,6]. Another approach consists of developing a totally new MMOS with a better scheduler, a lighter-weight I/O system, and a more flexible user/system interface [1]. The last approach is based on using a standard UNIX operating system linked to a coprocessor which will provide functions for multimedia support [7]. Examples of these approaches are discussed next.

Support for continuous media can be achieved by modifying a traditional operating system and incorporating new scheduling mechanisms and new interprocess communication (IPC) facilities [5]. The scheduler, proposed in [5,8], is based on split-level scheduling in which each user virtual address space contains multiple leightweight processes. The scheduler is partitioned into user-level and kernel-level parts, which communicate via shared memory. The information in shared memory is used to correctly prioritize lightweight processes in different virtual address spaces, and thus avoid domain and mapping switches where possible. The split-level scheduler combines advantages of threads and lightweight processes, and minimize user/kernel interaction. An example of using the split-level scheduling is shown in Figure 7 [5]. The kernel-level scheduler decides which user virtual address space should execute. Each virtual address space has a user-level scheduler that manages lightweight processes in that virtual address space.

In the example in Fig. 7, the kernel-level scheduler chooses virtual address space S2 to run because it has the globally earliest deadline. The user-level scheduler in that virtual address space,

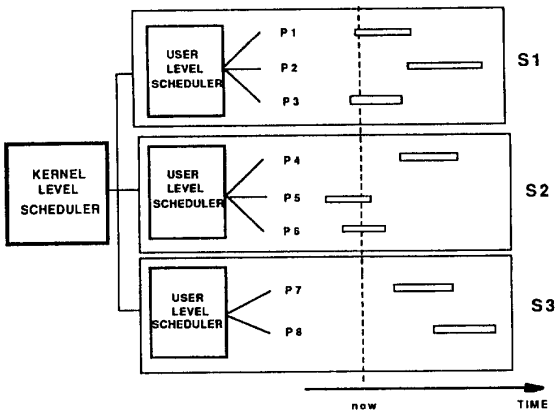

Figure 7. An example using split-level scheduling

which is based on deadline scheduling policy, will execute P5, which has the earliest deadline. User/kernel interactions are reduced, because the context switching to P6, which has the next earliest deadline, can be done without a kernel call.

Another example of providing multimedia support on a Ultrix 4.2 operating system is proposed in [6]. The mechanisms have been developed, which provide real-time scheduling for multimedia applications with the goal to reduce and bound the overall delay in the response times to both external and internal events.

The second approach consists of developing a new MMOS, and an example of this approach is Pegasus operating system developed jointly by universities of Cambridge and Enshede [1]. The Pegasus operating system provides capturing, rendering, storage, and interchange of multimedia data by user-level applications, while keeping all of the desirable properties of distributed systems, such as resource sharing, data sharing, security, and fault tolerance.

Another approach to provide multimedia support on UNIX systems consists of adding a multimedia coprocessor as a component of a workstation architecture [7]. In such a system UNIX operating system runs on the host workstation, while the multimedia coprocessor (MmCP) is driven by a separate distributed operating system that provides communication support between its hosting workstation and other workstations in a distributed environment, as shown in Figure 8.

The MmCP can execute arbitrarily complex processing sequences, and has access to the whole host's memory. The MmCP controls local I/O and provides synchronization mechanism. This approach provides a balance by keeping the benefits of a standard UNIX development environment, and providing new, high performance services for continuous media.

4. REAL-TIME ISSUES IN ON-DEMAND MULTIMEDIA SERVICES

Advances in distributed multimedia systems began to have significant impact on the development of on-demand multimedia services, such as interactive entertainment, video news distribution services, distribution of video rental services, digital multimedia libraries, and others. Entertainment, cable, and phone companies realized that fiber optic networks coupled with improved computing and compression techniques would soon be capable of delivering digital movies on demand. Over the last year, a number of alliances have been formed between entertainment, cable, phone, and computer companies with the main focus on video on demand applications.

On-demand multimedia services will use a hierarchical configuration of multimedia servers and network switches [9,10], as shown in an example in Figure 9.

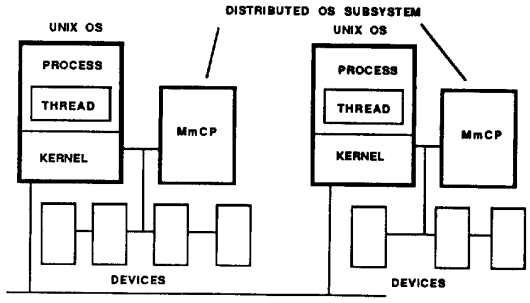

Figure 8. Distributed operating system configuration with multimedia coprocessors [7]

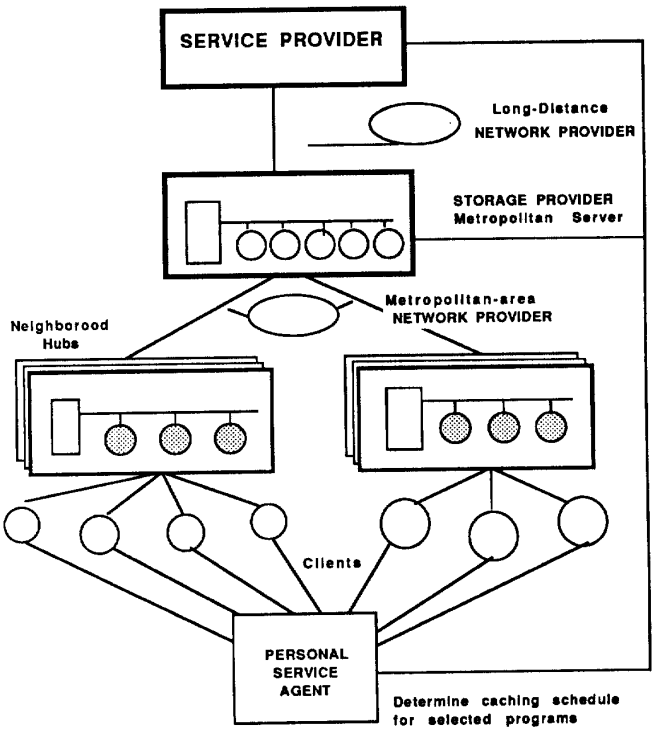

Figure 9. Hierarchical configuration of multimedia servers and network switches for on-demand multimedia services [9]

The system consists of *storage providers* (such as publishing houses, entertainment houses, and television stations), who offer various services, *network providers* who are responsible for media transport over integrated networks, and several levels of *storage providers* who manage multimedia data storage in multimedia servers.

In such systems, there are many real-time problems which have to be addressed, including the development of predictable, high-speed network, real-time multimedia data retrieval, real-time video decompression, media synchronization, and others. In this section, we present two challenging problems dealing with the development of an inexpensive interactive TV set top box for these systems, and real-time program scheduling and caching algorithms.

The information superhighway, digitally compressed audio and video, interactive movies and games, access to databases and services all on the TV, will change the culture and habits of the future generations. An interactive TV set top box is required to enable these functions and services. This box will incorporate many functions, capabilities, and peripherals used by the personal computer industry today. In addition, unlike the personal computers, these systems will perform many real-time functions. The main challenge facing the designers of the interactive TV set top box is how to get all the functions for the right, acceptable price, which, according to some analyses, should be about $300.

Interactive TV services to be supported by the interactive TV set top box should include: basic TV, subscription TV, pay per view, video on demand, electronic guide, electronic navigator,

electronic yellow pages, promotional services, interactive network games, electronic newspaper, digital audio, and downloadable games. These services translate into a number of functions that must be supported by this box. Although all the functions are not yet clearly defined, it is obvious that the basic interactive TV set top box will have to receive an analog signal from coaxial cable, decode it into a digital signal, decompress the digital signal into video and sound, and provide interfaces to remote controls and upstream communication for the interactive part. The interactive TV set top box will have interfaces to a number of devices, which implies a variety of I/O signals, including analog, digital, rf, and infrared signals, making the integration of these functions very complex from a semiconductor standpoint. Figure 10 shows the interactive box including the I/O requirements.

By incorporating all necessary functions: communication, video and audio decompression, games, upstream communication, descrambling, program storage, infrared, keyboard inputs, and computer line peripherals, the interactive TV set top box becomes similar to a networked multimedia personal computer and even more. In addition, the system will include many functions of a standard cable TV set box, and required real-time features. The architecture of the future interactive TV set top box, shown in Figure 11, will be real-time in its nature. It will consists of a powerful RISC processor running a real-time, multimedia operating system, large memory capable of temporarily storing movies or games, and several I/O interface components, as indicated in Fig. 11. And the challenging question will remain, will be really be able to get this system for $300?

Personalized service agents (PSA), indicated in Figure 9 and proposed in [9,12], will be capable of castomizing services to meet the needs of individual clients. The PSA will employ content analysis techniques to select programs that closely match clients' interest, and then schedule the selected programs for client's preferred viewing times by caching the programs at neighborhood servers. The PSA will perform schedule of the program retrieval for storage providers, programs transmission via network providers, and their caching at neighborhood servers or at clients' sites.

The proposed algorithms for program retrieval and caching will be capable of minimizing the total cost of programs storage and transmission by implementing resource optimization [9]. These program scheduling algorithms can be real-time in their nature, because the schedules may need to be dynamically altered due to real-time requests from clients. In addition, they must be adaptable to any change that may occur in the costs of servers and networks, which can even occur on-the-fly. An optimal scheduling and caching strategy in a unrestricted network configuration of storage servers and clients, and a fast, on-line scheduling and cache strategy are presented in [9].

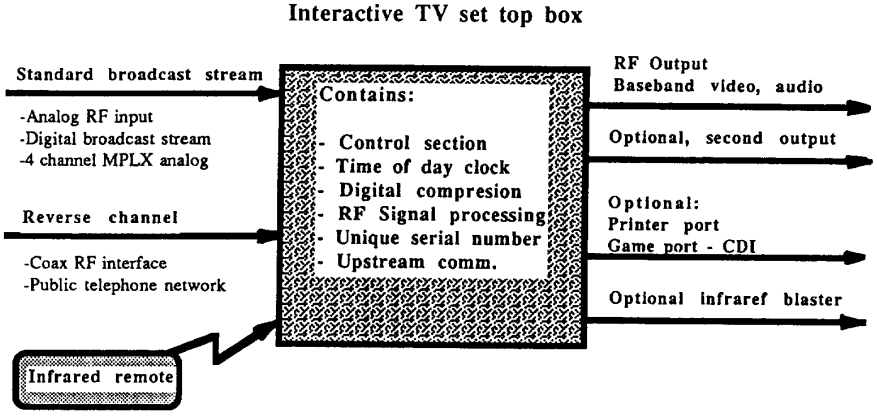

Figure 10. Interactive TV set top box and its I/O requirements

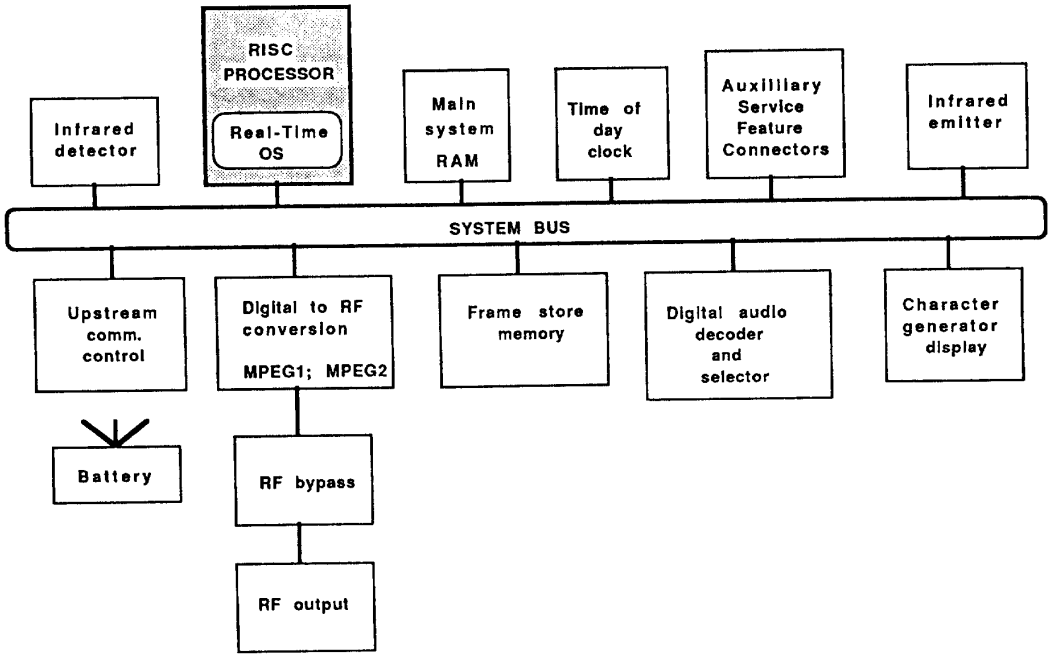

Figure 11. The real-time architecture of a future interactive TV set top box

References

[1] I.M. Leslie, D. McAuley, and S.J. Mullender, "Pegasus - Operating System Support for Distributed Multimedia Systems", *Operating Systems Review*, 1991, pp. 69-77.

[2] B. Furht, "Multimedia Systems: An Overview", to appear in *IEEE Multimedia Magazine*, Vol. 1, No. 1, 1994.

[3] T.D.C. Little and A. Ghafoor, "Network Considerations for Distributed Multimedia Object Composition and Communication", *IEEE Network*, November 1990, pp. 32-49.

[4] T.D.C. Little and A. Ghafoor, "Multimedia Synchronization Protocols for Broadband Integrated Services", *IEEE Journal on Selected Areas in Communications*, Vol. 9, No. 9, December 1991, pp. 1368-1382.

[5] R. Govindan and D.P. Anderson, "Scheduling and IPC Mechanisms for Continuous Media", *Proc. of 13th Symposium on Operating Systems Principles*, 1991, pp. 68-80.

[6] T. Fisher, "Real-Time Scheduling Support in Ultrix 4.2 for Multimedia Communication", *Proc. of the Third International Workshop on Network and Operating System Support for Digital Audio and Video*, San Diego, CA, November 1992, pp. 282-288.

[7] D.C.A. Bulterman, and R. van Liere, "Multimedia Synchronization and UNIX", *Proc. of the 2nd International Workshop on Network and Operating System Support for Digital Audio and Video*, Heidelberg, Germany, November 1991, pp.108-119.

[8] D.P. Anderson, "Metascheduling for Continuous Media", *ACM Transactions on Computer Systems*, Vol. 11, No. 3, August 1993, pp. 226-252.

[9] S. Ramanathan and P. V. Rangan, "System Architectures for Personalized Multimedia Services", to appear in *IEEE Multimedia Magazine*, 1994.

[10] P.V. Rangan, H.M. Vin, and S. Ramanathan, "Designing an On-Demand Multimedia Service", *IEEE Communication Mgazine*, July 1992, pp. 56- 64.

[11] G. Morse, "The Technologies of the Interactive TV Set Top Box", Multimedia project report, Florida Atlantic University, Boca Raton, 1994.

[12] K.P. Davies, "Digital Television Broadcasting Dreams, Decisions, and Destinies", *SMPTE Journal*, January 1993, pp. 32-36.