COMPUTER TECHNOLOGY AND MULTIMEDIA OPERATING SYSTEM (MOS)

Unit 7

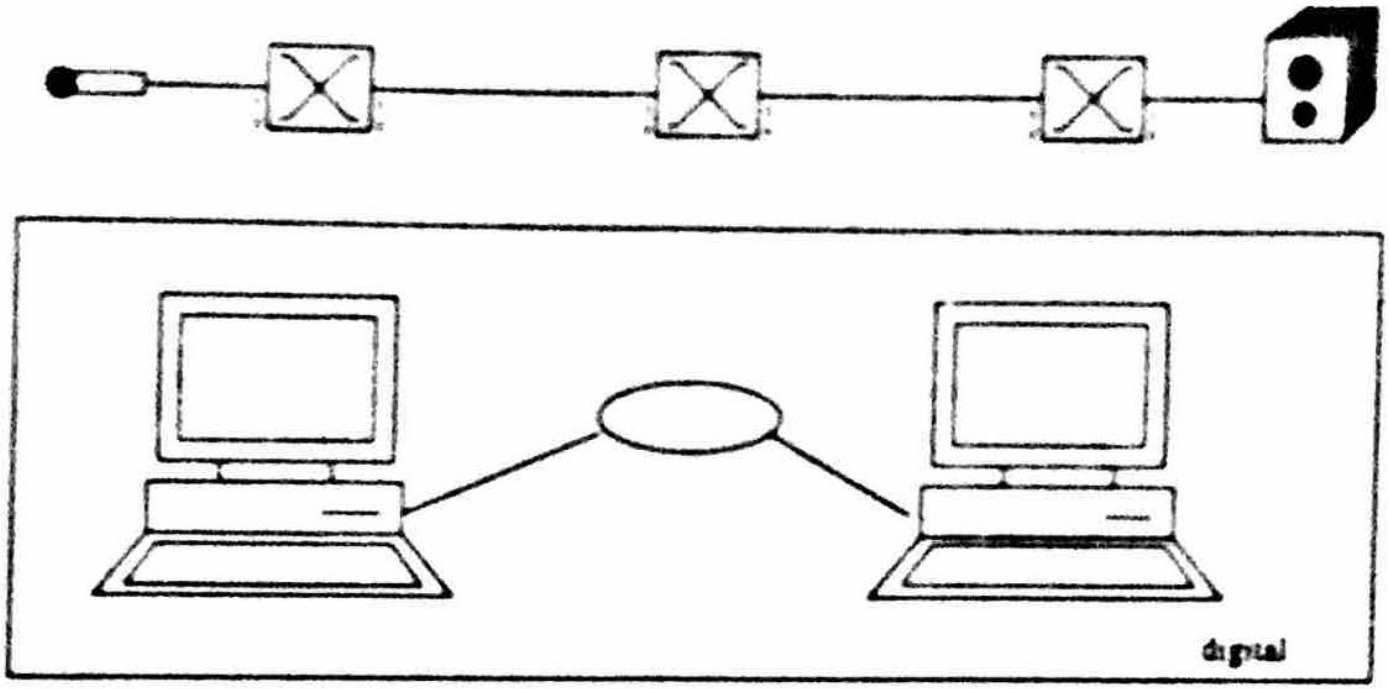
7.1. Communication Architecture

Local multimedia systems (i.e., multimedia workstations) frequently include a network interface (e.g., Ethernet card) through which they can communicate with each other. However, the transmission of audio and video cannot be carried out with only the conventional communication infrastructure and network adapters.

Until now, the solution was that continuous and discrete media have been considered in different environments, independently of each other. It means that fully different systems were built. For example, on the one hand, the *analog telephone system* provides audio transmission services using its original dial devices connected by copper wires to the telephone company's nearest *end office*. The end offices are connected to switching centers, called *toll offices*, and these centers are connected through high bandwidth intertoll trunks to *intermediate switching offices*. This hierarchical structure allows for reliable audio communication. On the other hand, *digital computer networks* provide data transmission services at lower data rates using network adapters connected by copper wires to switches and routers.

Even today, professional radio and television studios transmit audio and video

7.1. Communication Architecture



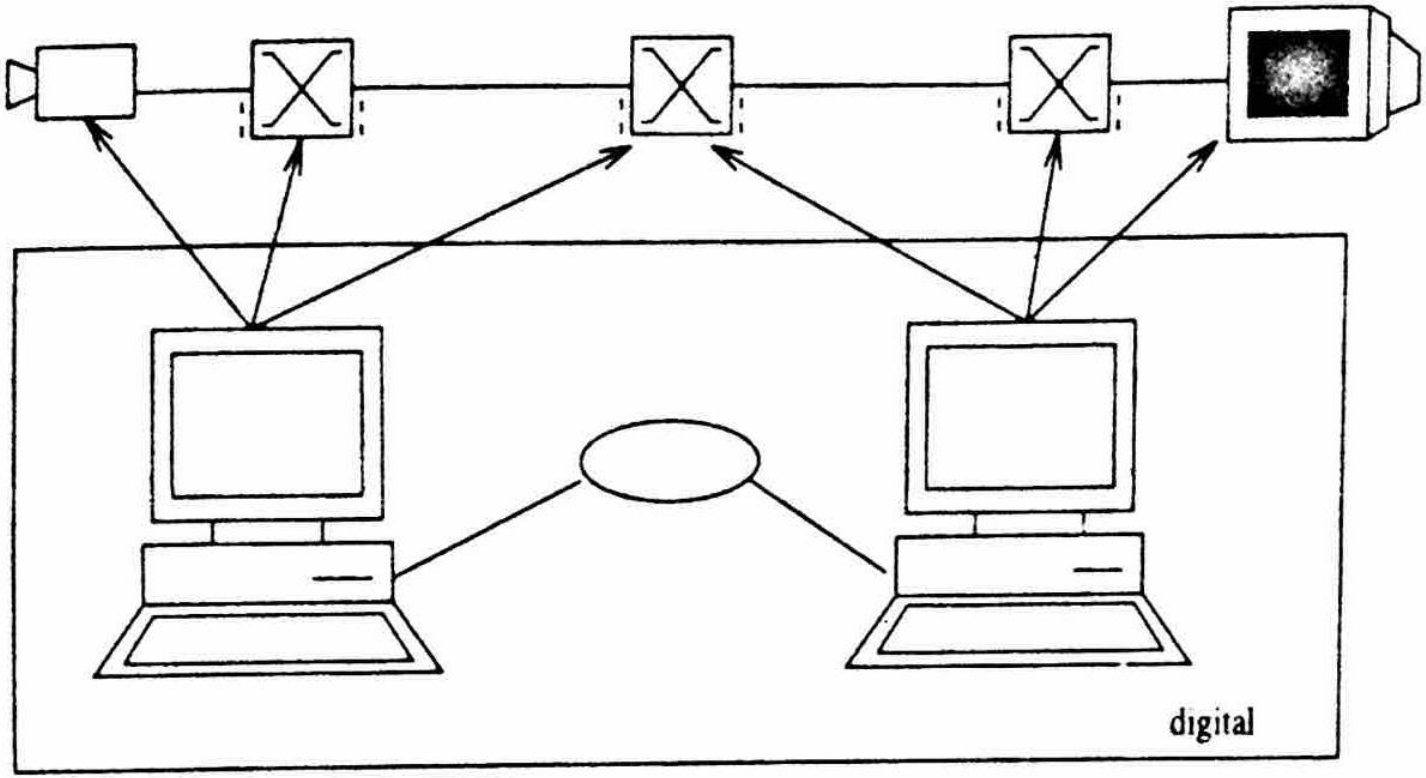
Analog and digital environments without interaction

7.1. Communication Architecture: Hybrid System

By using existing technologies, integration and interaction between analog and digital environments can be implemented. This integration approach is called the *hybrid approach*.

The main advantage of this approach is the high quality of audio and video and all the necessary devices for input, output, storage and transfer that are available. The hybrid approach is used for studying application user interfaces, application programming interfaces or application scenarios. The transmission techniques used in these cases are less important, although to meet the goal of full digital integration, this approach is not satisfactory.

7.1. Communication Architecture: Hybrid System



Computer control of all audio-video components.

7.1. Communication Architecture: Hybrid System

- This trend has emerged because with cooperative peers it is possible to asymptotically enhance the use of resources in sharing of data compared to the basic client-server architecture.
- The need for distribution of data is wide and one could argue that it is as fundamental a building block as the message passing of the Internet.
- As an answer to this need a new scalable architecture is introduced: **Hybrid Communication Architecture** (HCA), which provides both data sharing and message passing as communication primitives for applications.

7.1. Communication Architecture: Hybrid System

- HCA can be regarded as an abstraction layer for communication which is further encapsulated by a higher-level middleware.
- HCA is aimed at general use, and it is not designed for any particular application.
- One key idea is to combine data sharing with streaming since together they enable many applications not easily implementable with only one of these features.
- For example, a game application could share the game world state between clients and modify it by using streaming. The other distinctive feature of the system is the use of knowledge of the physical network topology in the optimization of the communication. With a feasible business model, fault-tolerance, and security features, HCA is aimed eventually for real-life adoption.

7.1. Communication Architecture: Hybrid System

The following interfaces and protocols form the skeleton of the HCA framework:

- Client interface is used by applications on top of HCA,
- DLL interface divides the client side implementation of HCA into dynamically and statically linked parts,
- Context interface passes information from the execution environment to the application,
- Security interface provides an access to cryptographic functions,
- OSLib interface provides an abstraction layer for different operating systems,
- Inter-domain protocol is used by communication nodes of the HCA network,
- Intra-domain protocol is used by communication nodes of the same domain as explained later.

7.1. Communication Architecture: Hybrid System

In a normal configuration, multiple applications with their HCA client interface static implementations and a single local node implementation are located on the same **client machine** at the edge of the HCA overlay network.

The **client machine** can be, for example, the user's home computer or office workstation which is shut down every now and then. The local node communicates with the nearest communication node using the inter-domain protocol.

This node probably resides on a different machine typically administered by a service provider organization or it could be located in a company intranet server.

7.1. Communication Architecture: Hybrid System

The HCA overlay network is formed by many communication nodes structured in the form of hierarchical domain tree.

The nodes are connected by the inter-domain protocol and typically reside in servers that are continuously online. Clients and persistence servers can be thought of residing at the edge of the overlay network.

Client interface, OSLib, Context interface and Security interface form together an operating-system- and hardware-independent platform for components on top of HCA.

7.1. Communication Architecture: Hybrid System

Client Interface

The HCA Client interface is the only part of the communication architecture visible to applications and higher-level layers of the middleware.

Because HCA is intended to be programming language neutral, there are language mappings of the client interface to each programming language supported.

The implementation of the client interface is normally divided to statically and dynamically linked parts so that it is possible to change the implementation without recompiling applications.

7.1. Communication Architecture: Hybrid System

DLL Interface and Local Node Implementation

HCA DLL (Dynamic link library) interface is an operating-system dependent application programming interface (API) for wrapping the local communication node to a separate process running on the client machine.

The DLL implementation shares the local communication node between multiple client processes and implements the communication between client processes and the local communication node process using fast inter-process communication, shared memory or other facilities provided by the operating system.

7.1. Communication Architecture: Hybrid System

OSLib

OSLib programming interface provides a system-wide hardware abstraction layer (HAL) for preventing the components of HCA of having unnecessary dependencies to particular operating system or hardware. With this modularization in implementations we gain the easy portability of HCA implementations over different platforms.

7.1. Communication Architecture: Hybrid System

Context Interface

Context interface is intended to pass information from the execution platform to the program. For example, communication node implementations can use it to read the configuration information of the node

Security Interface

Security interface contains a uniform interface for typical cryptographic algorithms. An abstract interface for both public key and secret key methods are specified.

Inter-domain Protocol

Inter-domain protocol is used between all communication nodes to form the overlay network of HCA

7.1. Communication Architecture: Hybrid System

Intra-domain Protocol

Each domain can use their own intra-domain protocol for coordination of the nodes inside the domain.

Nodes must implement the intra-domain protocol used by their enclosing domain. Small domains do not necessarily need an intra-domain protocol for managing nodes because it can be achieved by configuring each node manually.

7.1. Communication Architecture: Digital System

Digital systems are designed to store, process, and communicate information in digital form.

They are found in a wide range of applications, including process control, communication systems, digital instruments, and consumer products.

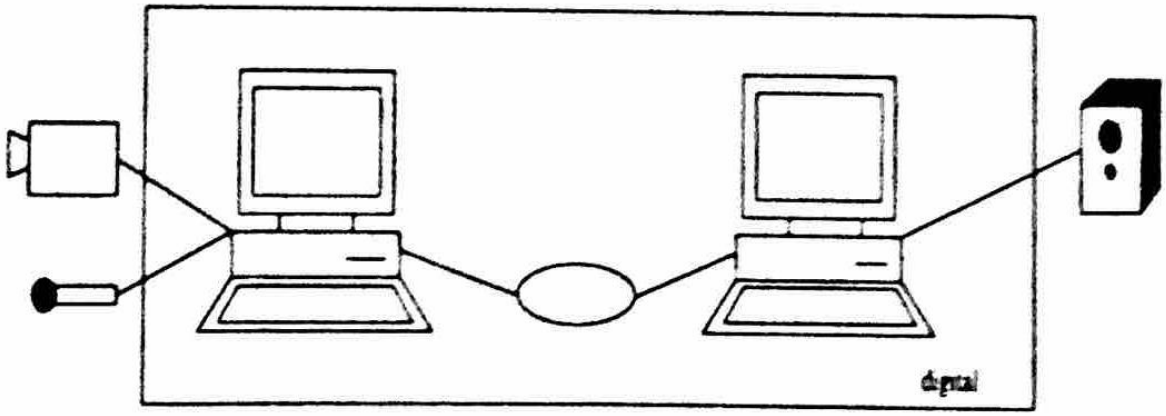
The digital computer, more commonly called the *computer*, is an example of a typical digital system.

7.1. Communication Architecture: Digital System

Connection to Workstations

In digital systems, audio-video devices can be connected directly to the computers (workstations) and digitized audio-video data are transmitted over shared data networks, Audio-video devices in these systems can be either analog or digital. Figure 8.5 shows an integrated system structure with analog devices and A/D and D/A interfaces. Figure 8.6 shows an integrated system structure with digital end-system devices and interfaces.

7.1. Communication Architecture: Digital System



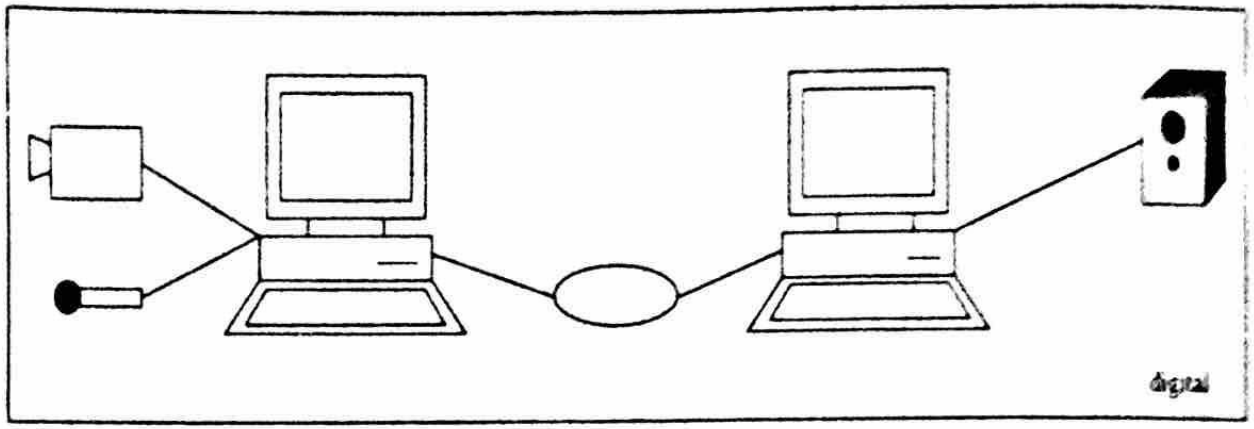
Integrated (with respect to hardware) system structure with analog end-system devices and A/D and D/A interfaces.

7.1. Communication Architecture: Digital System

An example of a digital system is the *Etherphone* system from Xerox PARC [Swi87]. A digital audio communication was demonstrated over an Ethernet, although not in a fully integrated form, i.e., the audio was not processed in the main memory.

Another example is an early project by AT&T in Naporville, which considered a similar system architecture to a Etherphone [LL89, LBH+90]. Here, a computer was directly connected to a *Fast Packet Switching* network. The processing of continuous media in the computer was allowed through extensions of the UNIX operating

7.1. Communication Architecture: Digital System



Integrated (with respect to hardware) system structure with digital end system devices and interfaces.

7.1. Communication Architecture: Digital System

A computer manipulates information in digital, or more precisely, binary form. A binary number has only two discrete values — zero or one.

7.2. Multimedia Workstation

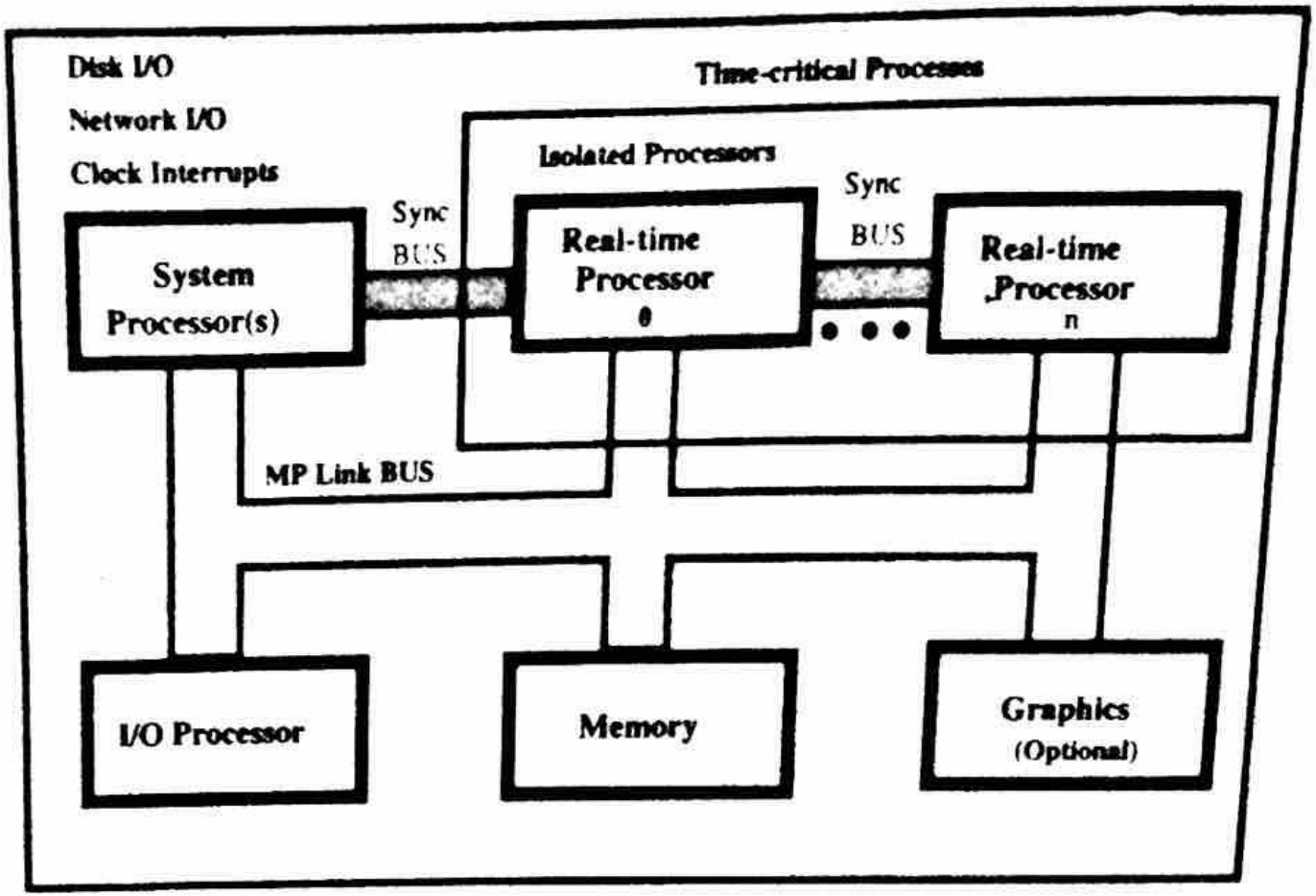
Current workstations are designed for the manipulation of discrete media information. The data should be exchanged as quickly as possible between the involved components, often interconnected by a common bus. Computationally intensive and dedicated processing requirements lead to dedicated hardware, firmware and additional boards. Examples of these components are hard disk controllers and FDDI-adapters.

7.2. Multimedia Workstation

A *multimedia workstation* is designed for the simultaneous manipulation of discrete and continuous media information. The main components of a multimedia workstation are:

- *Standard Processor(s)* for the processing of discrete media information.
- *Main Memory* and *Secondary Storage* with corresponding autonomous controllers.
- *Universal Processor(s)* for processing of data in real-time (signal processors).
- *Special-Purpose Processors* designed for graphics, audio and video media (containing, for example, a micro code decompression method for DVI processors) [Rip89, Tin89, Lut91].
- *Graphics and Video Adapters.*
- *Communications Adapters* (for example, the Asynchronous Transfer Mode Host Interface [TS93].
- Further *special-purpose adapters.*

7.2. Multimedia Workstation



Silicon Graphics workstation architecture (POWER Lock Processor Isolation).

7.3. Introduction to MOS / Function of MOS

The operating system provides a comfortable environment for the execution of programs, and it ensures effective utilization of the computer hardware.

The OS offers various services related to the essential resources of a computer: CPU, main memory, storage and all input and output devices.

In multimedia applications, a lot of data manipulation (e.g. A/D, D/A and format conversion) is required and this involves a lot of data transfer, which consumes many resources.

7.3. Introduction to MOS / Function of MOS

The integration of discrete and continuous multimedia data demands additional services from many operating system components.

The major aspect in this context is *real-time processing* of continuous media data.

7.3. Introduction to MOS / Function of MOS

Issues concerned:

- Process management: a brief presentation of traditional real-time scheduling algorithms.
- File systems: outlines disk access algorithms, data placement and structuring
- Interprocess communication and synchronization
- Memory management
- Database management
- Device management

7.3. Introduction to MOS / Function of MOS

Process management must take into account the timing requirement imposed by the handling of multimedia data.

- Concerns in process management (Scheduling):

	Traditional OS	MM OS
Timing requirement	No	Yes
Fairness	Yes	Yes

7.3. Introduction to MOS / Function of MOS

Single components are conceived as resources that are reserved prior to execution to obey timing requirements and this *resource reservation* has to cover all resources on a data path.

The *communication & synchronization between single processes* must meet the restrictions of real-time requirements and timing relations among different media.

7.3. Introduction to MOS / Function of MOS

Memory management has to provide access to data with a guaranteed timing delay and efficient data manipulation functions. (e.g. should minimize physical data copy operations.)

Database management should rely on file management services

7.5. Multimedia Real Time System

A real-time process is a process which delivers the results of the processing in a given time-span.

The main characteristic of real-time systems is the correctness of the computation.

- Errorless computation
- The time in which the result is presented

7.5. Multimedia Real Time System

Speed and efficiency are not the main characteristic of real-time systems. (e.g. the video data should be presented at the right time, neither too quickly nor too slowly)

Timing and logical dependencies among different related tasks, processed at the same time, must also be considered.

7.5. Multimedia Real Time System

Deadlines:

A deadline represents the latest acceptable time for the presentation of a processing result.

- Soft deadline:
  - a deadline which cannot be exactly determined and which failing to meet does not produce an unacceptable result.
  - Its miss may be tolerated as long as (1) not too many deadlines are missed and/or (2) the deadlines are not missed by much.

7.5. Multimedia Real Time System

- Hard deadline:
  - a deadline which should never be violated.
  - Its violation causes a system failure.
  - Determined by the physical characteristics of real-time processes.

7.5. Multimedia Real Time System

Characteristics of real time systems

The necessity of deterministic and predictable behavior of real-time systems requires processing guarantees for time-critical tasks.

A real-time system is distinguished by the following features:

- Predictably fast response to time-critical events and accurate timing information:
- A high degree of schedulability: to meet the deadlines.
- Stability under transient overload: critical task first.

7.5. Multimedia Real Time System

Real time and multimedia

The real-time requirements of traditional real-time scheduling techniques usually have a high demand for security and fault-tolerance. (Most of them involve system control.)

7.5. Multimedia Real Time System

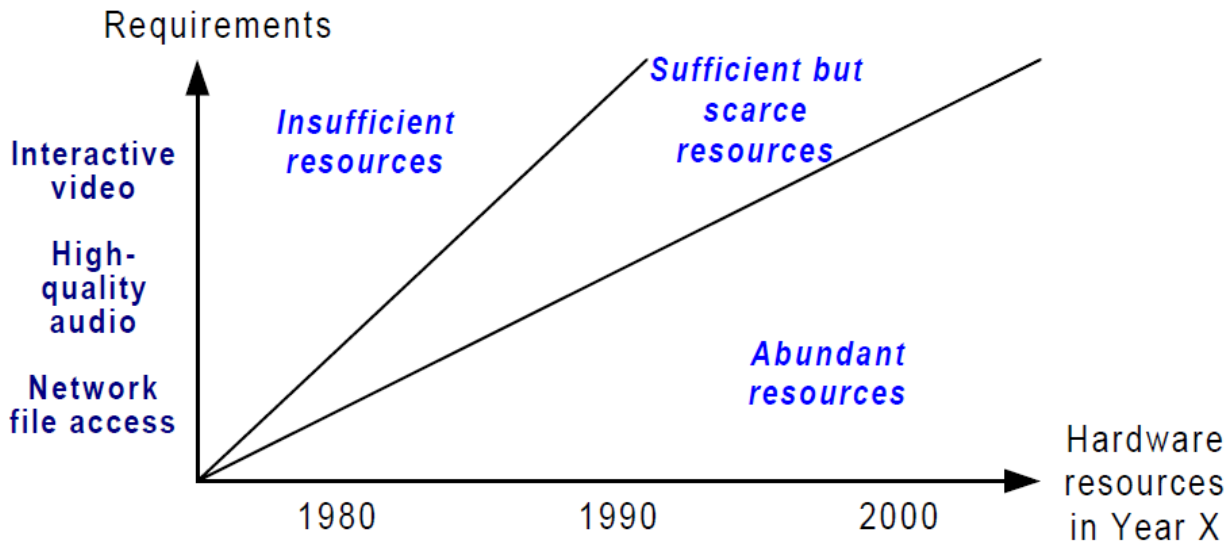
Real-time requirements of multimedia systems:

- The fault-tolerance requirements of multimedia systems are usually less strict than those of real-time systems that have a direct physical impact.
- For many multimedia system applications, missing a deadline is not a severe failure, although it should be avoided. (e.g. playing a video sequence)
- In general, all time-critical operations are periodic and schedulability considerations for periodic tasks are much easier.
- The bandwidth demand of continuous media is usually negotiable and the media is usually scalable.

7.5. Multimedia Real Time System
(Resource management)

Resource management

Multimedia systems with integrated audio and video processing are at the limit of their capacity even with data compression and utilization of new technology. (Demand increases drastically.)



7.5. Multimedia Real Time System

(Resource management)

No redundancy of resource capacity can be expected in the near future.

In a multimedia system, the given timing guarantees for the processing of continuous media must be adhered to along the data path.

The actual requirements depend on (1) the type of media and (2) the nature of the applications supported.

The shortage of resources requires careful allocation.

The resource is first allocated and then managed.

7.5. Multimedia Real Time System

(Resource management)

At the connection establishment phase, the resource management ensures that the new 'connection;' does not violate performance guarantees already provided to existing connections.

Applied to OS, resource management covers the CPU (including process management), memory management, the file system and the device management.

The resource reservation is identical for all resources, whereas the management is different for each.

7.5. Multimedia Real Time System (Resource management)

Resources

A resource is a system entity required by tasks for manipulating data.

A resource can be active or passive.

- Active resource:
  - e.g. the CPU or a network adapter for protocol processing;
  - it provides a service.
- Passive resource:
  - e.g. main memory, communication bandwidth or file systems;
  - It denotes some system capability required by active resources.

7.5. Multimedia Real Time System
(Resource management)

Resources

A resource can be either used exclusively by one process at a time or shared between various processes.

Active ones are often exclusive while passive ones can usually be shared.

Each resource has a capacity in a given time-span. (e.g. processing time for CPU, the amount of storage for memory and etc.)

For real-time scheduling, only the temporal division of resource capacity among real-time processes is of interest.

7.5. Multimedia Real Time System

(Resource management)

Requirements

The requirements of multimedia applications and data streams must be served.

The transmission/processing requirements of local and distributed multimedia applications can be specified according to the following characteristics:

- *Throughput*: Determined by the needed data rate of a connection to satisfy the application requirements.
- *Delay "at the resource"* (local): The maximum time span for the completion of a certain task at this resource.
- *End-to-end delay* (global): The total delay for a data unit to be transmitted from the source to its destination.
- *Jitter*: Determines the maximum allowed variance in the arrival of data at the destination.
- *Reliability*: Defines error detection and error correction mechanisms used for the transmission and processing of multimedia tasks.
  - How to handle errors: Ignored, indicated and / or corrected.
  - Retransmission may not be acceptable for time critical data.
- These requirements are known as *Quality of Service* (QoS) parameters.

7.5. Multimedia Real Time System
(Resource management)

Components and phases

Resource allocation and management can be based on the interaction between clients and their respective resource managers.

The client selects the resource and requests a resource allocation by specifying its QoS specification.

The resource manager checks its own resource utilization and decides if the reservation request can be served or not.

Performance can be guaranteed once it is accepted.

7.5. Multimedia Real Time System

(Resource management)

Components and phases (cont…)

Phases of the resource reservation and management process

1.Schedulability

- The resource manager checks with the given QoS parameters (e.g. throughput and reliability).

2.QoS calculation

- The resource manager calculates the best possible performance (e.g. delay) the resource can guarantee for the new request.

3.Resource reservation

- Allocates the required capacity to meet the QoS guarantees for each request.

4. Resource scheduling

- Incoming messages (i.e. LDUs) from connections are scheduled according to the given QoS guarantees.

7.5. Multimedia Real Time System
(Resource management)

Allocation Scheme

Reservation of resources can be made either in a pessimistic or optimistic way:

The pessimistic approach avoids resource conflicts by making reservations for the worst case. (It's very conservative.)

The optimistic approach reserves resources according to an average workload only.

7.5. Multimedia Real Time System
(Resource management)

Allocation Scheme (cont..)

	Pessimistic approach	Optimistic approach
Account for	Worst case	Average case
QoS	Guaranteed	Best effort
Utilization	Low	High
Remarks		May need a monitor to detect overload situation and act

7.5. Multimedia Real Time System
(Resource management)

Continuous media resource model

A model is frequently adopted to define QoS parameters and the characteristics of the data stream.

It is based on the model of linear bounded arrival process (LBAP).

A distributed system is decomposed into a chain of resources traversed by the messages on their end-to-end path.

The data stream consists of LDUs (messages). Various data streams are independent of each other.

The model considers a burst of messages consists of messages that arrived ahead of schedule.

7.5. Multimedia Real Time System
(Resource management)

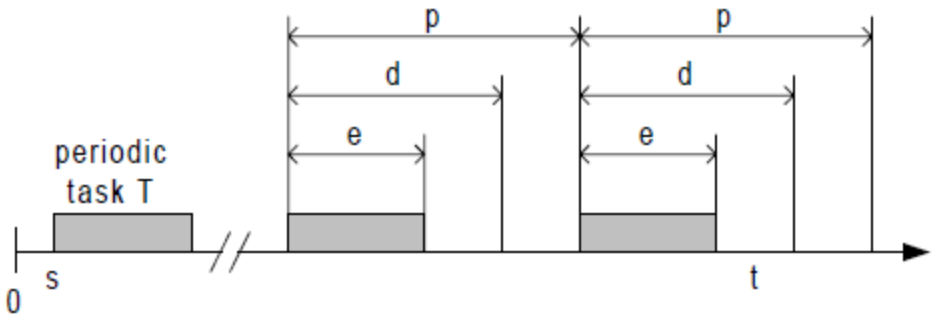
Real-time scheduling: system model

- A task is a schedule entity of the system.
- In a hard real-time system, a task is characterized by its timing constraints and its resource requirements.
- Here, only periodic tasks without precedence constraints are discussed. i.e. the processing of 2 tasks is mutually independent.

7.5. Multimedia Real Time System

(Resource management)/(Real-time scheduling: system model)

- The time constraints of the periodic task T are characterized by the following parameters (s,e,d,p)
  - s : Starting point
  - e : Processing time of T
  - d : Deadline of T
  - p : Period of T
  - r : Rate of T (r=1/p) whereby $0 \leq e \leq d \leq p$



Characterization of periodic tasks

7.5. Multimedia Real Time System

(Resource management)/(Real-time scheduling: system model)

For continuous media tasks, it is assumed that the deadline of the period (k-1) is the ready time of period k (i.e. d=p), which is known as *congestion avoiding deadlines*. All tasks processed on the CPU are considered as preemptive unless otherwise stated.

Performance metrics:

- *Guarantee ratio*: The total number of guaranteed tasks versus the number of tasks which could be processed.
- *Processor utilization*: The amount of processing time used by guaranteed tasks versus the total amount of processing time: $U = \sum_{i=1}^{n} \frac{e_i}{p_i}$

7.5. Multimedia Real Time System

(Resource management)/(Real-time scheduling: system model)

- Note, for each task *i*, the processor utilization is e_i / p_i . Many tasks are running in parallel, so we've

$$U = \sum_{i=1}^{n} \frac{e_i}{p_i}.$$

7.5. Multimedia Real Time System

(Earliest deadline first (EDF) algorithm)

(Earliest deadline first (EDF) algorithm)

- The highest priority is assigned to the task with the earliest deadline.
- EDF is an optimal, dynamic algorithm.
- At every new ready state, the scheduler selects the task with the earliest deadline among the tasks that are ready and not fully processed.
- At any arrival of a new task, EDF must be computed immediately leading to a new order and the new task is scheduled according to its deadline.
- The running task is preempted.
- No guarantee about the processing of any task can be given.
- It has a lot of overhead.

7.5. Multimedia Real Time System

(Earliest deadline first (EDF) algorithm)

Extension of EDF (*Time-Driven Scheduler* (TDS)):

- If an overload situation occurs the scheduler aborts tasks which cannot meet their deadlines anymore.
- If there is still an overload situation, the scheduler removes task which is less important for the system.

7.5. Multimedia Real Time System

(Earliest deadline first (EDF) algorithm)

Another variant of EDF:

- Every task is divided into a mandatory and an optional part.
- A task is terminated according to the deadline of the mandatory part, even if it is not completed at this time.
- Tasks are scheduled with respect to the deadline of the mandatory parts.
- The optional parts are processed if the resource capacity is not fully utilized.
- In an overload situation, the optional parts are aborted.
- This implementation favors scalable video.

7.5. Multimedia Real Time System (Rate monotonic algorithm)

- It is an optimal, static, priority-driven algorithm for preemptive, periodic jobs.
- It is optimal in a way that it can provide the same schedule any static algorithm can provide.
- There are 5 necessary prerequisites to apply the rate monotonic algorithm:
  - The requests for all tasks with deadlines are periodic.
  - Each task must be completed before the next request occurs.
  - All tasks are independent.
  - Run-time for each request of a task is constant
  - Any non-periodic task in the system has no required deadline.

7.5. Multimedia Real Time System

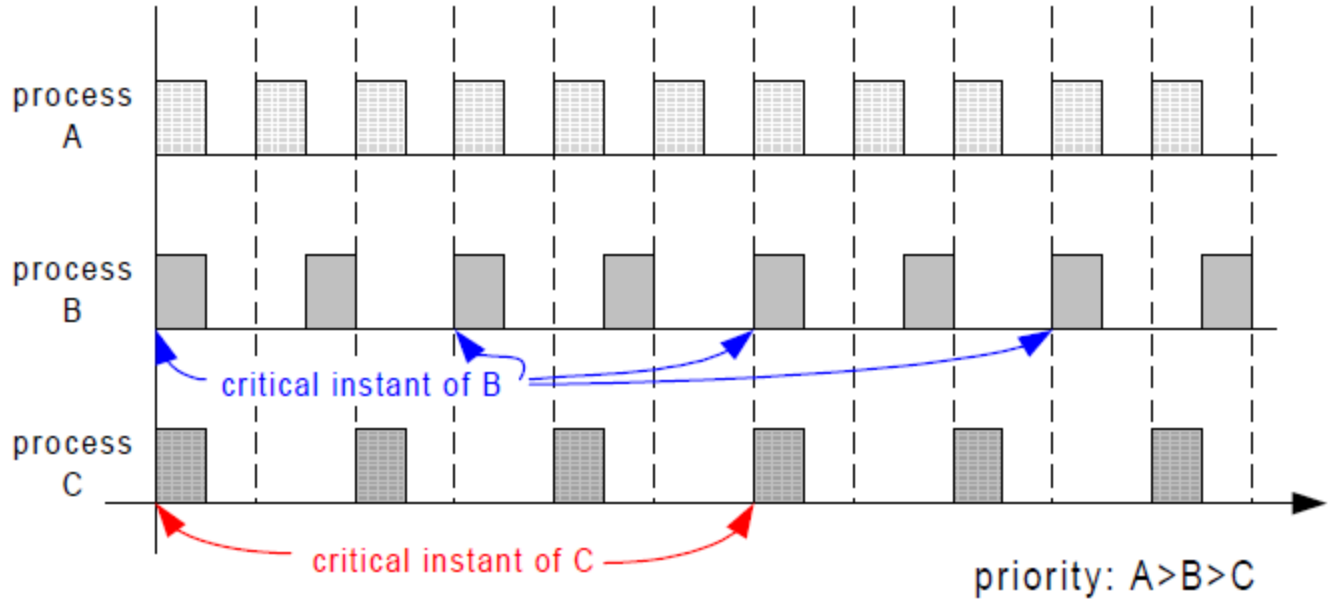
(Rate monotonic algorithm)

- A process is scheduled by a static algorithm at the beginning of the processing.
- Each task is processed with the priority calculated at the beginning.
- No further scheduling is required.
- Static priorities are assigned to tasks, once at the connection set-up phase, according to their request rates.

7.5. Multimedia Real Time System

(Rate monotonic algorithm)

- The time instant when this happens is known as the *critical instant*. (the fewer, the better.)

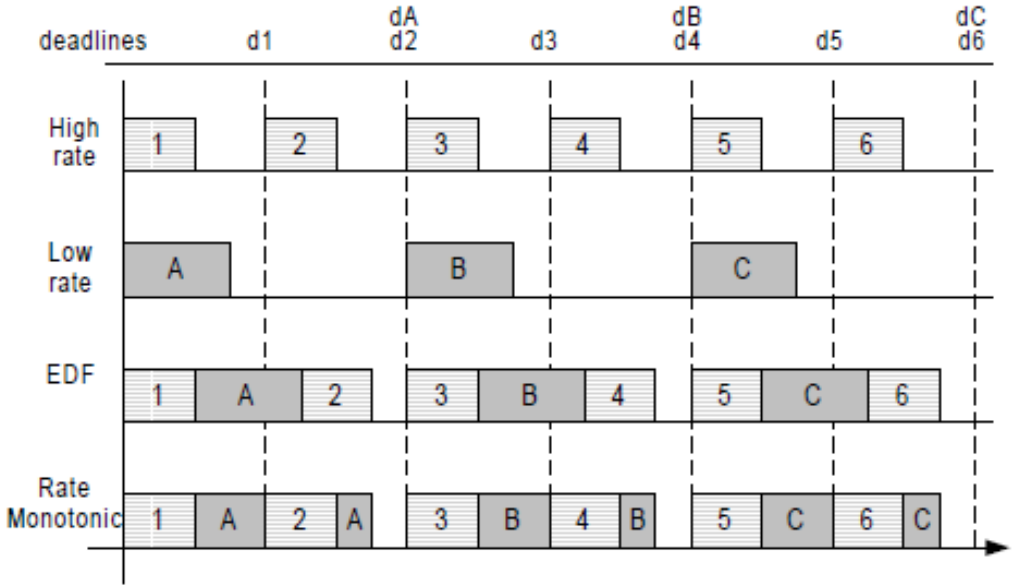


- The *critical time zone* is the time interval between the critical instant and the completion of a task. (the shorter, the better.)

7.5. Multimedia Real Time System

(EDF and Rate Monotonic: Context switches)

If more than one stream is processed concurrently in a system, it is very likely that there might be more context switches with a scheduler using the rate monotonic algorithm than one using EDF.



Rate monotonic vs. EDF: context switches in preemptive systems

7.5. Multimedia Real Time System

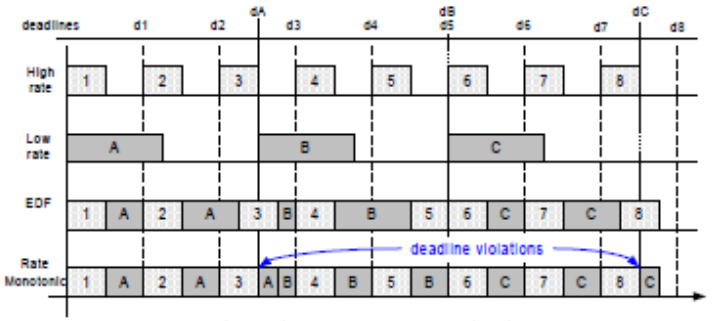
(EDF and Rate Monotonic: Processor Utilization)

- Rate monotonic algorithm
  - The processor utilization depends on the number of tasks which are scheduled, their processing times and their periods.
  - It is upper bounded.
  - The upper bound of the processor utilization is determined by the critical instant.
  - A set of m independent, periodic tasks with fixed priority will always meet its deadline if

    $$U(m) = m\left(2^{1/m} - 1\right) \geq \frac{e_1}{p_1} + \cdots + \frac{e_m}{p_m}$$

    Note: $U(m) = m\left(2^{1/m} - 1\right) \geq \ln 2 = 0.6931$
  - When a new job is coming, to save effort, just check if $U_k = U_{k-1} + \frac{e_k}{p_k} \leq \ln 2$, where $U_k = \sum_{i=1}^{k} \frac{e_i}{p_i}$. If yes, go ahead, else deny it.
  - The problem of underutilizing the processor is aggregated by the fact that, in most cases, the average task execution time is considerably lower than the worst case execution time.
  - The rate monotonic algorithm on average ensures that all deadlines will be met even if the bottleneck utilization is well above 80%.
  - No other static algorithm can achieve a higher processor utilization. (That's why it is optimal.)
- EDF
  - U(m) of 100% can be achieved with EDF as all tasks are scheduled dynamically according to their deadlines.

Example: Where the CPU can be utilized to 100% with EDF, but where rate monotonic scheduling fails.



Rate monotonic vs. EDF: processor utilization

7.5. Multimedia Real Time System (Extensions to Rate Monotonic Scheduling)

Extensions to Rate Monotonic Scheduling

- divides a task into a mandatory and an optional part.
- The processing of the mandatory part delivers a result which can be accepted by the user.
- The optional part only refines the result.
- The mandatory part is scheduled according to the rate monotonic algorithm.
- The optional part is scheduled with other polices.

7.5. Multimedia Real Time System (Extensions to Rate Monotonic Scheduling)

- What can we do if there are aperiodic tasks in some systems?
- If the aperiodic request is an aperiodic continuous stream, we transform it into a periodic stream if possible.
- If the stream is not continuous, we can apply a
- sporadic server to respond to aperiodic requests.
- Pros of the rate monotonic algorithm
  - It is particularly suitable for continuous media data processing because it makes optimal use of their periodicity.
  - No scheduling overhead as it is a static algorithm.
- Potential problems:
  - Problems emerge with data streams which have no constant processing time per message
  - The simplest solution is to schedule these tasks according to their maximum data rate, but this decreases processor utilization.

Finished

Chapter 7

Thank You