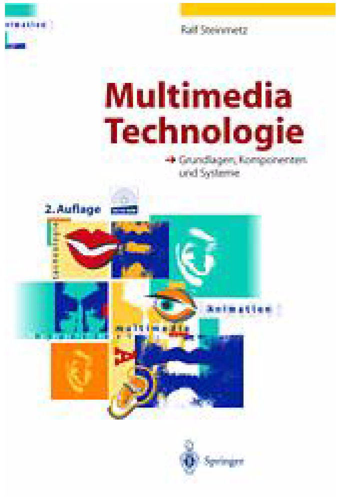
Multimedia-Systems:
Operating Systems



Prof. Dr.-Ing. **Ralf Steinmetz**
Prof. Dr. **Max Mühlhäuser**

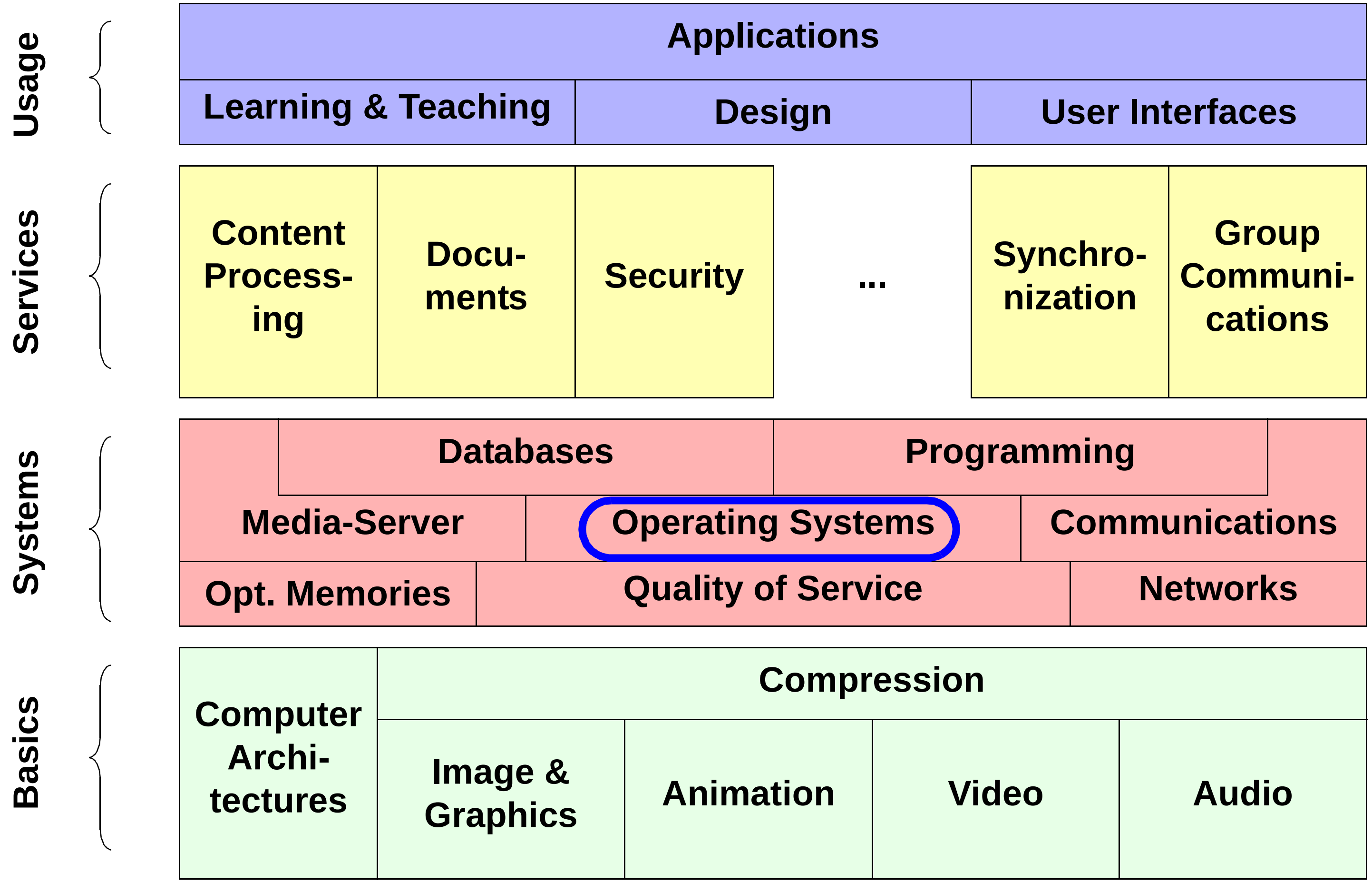
MM: *TU Darmstadt - Darmstadt University of Technology,*
Dept. of of Computer Science
TK - Telecooperation, Tel.+49 6151 16-3709,
Alexanderstr. 6, D-64283 Darmstadt, Germany, max@informatik.tu-darmstadt.de Fax. +49 6151 16-3052

RS: *TU Darmstadt - Darmstadt University of Technology,*
Dept. of Electrical Engineering and Information Technology, Dept. of Computer Science
KOM - Industrial Process and System Communications, Tel.+49 6151 166151,
Merckstr. 25, D-64283 Darmstadt, Germany, Ralf.Steinmetz@KOM.tu-darmstadt.de Fax. +49 6151 166152
GMD -German National Research Center for Information Technology
httc - Hessian Telemedia Technology Competence-Center e.V

Scope

Scope

Scope



Usage
Services
Systems
Basics
Applications
Learning & Teaching
Design
User Interfaces
Content Process-ing
Docu-ments
Security
...
Synchro-nization
Group Communi-cations
Databases
Programming
Media-Server
Operating Systems
Communications
Opt. Memories
Quality of Service
Networks
Computer Archi-tectures
Compression
Image & Graphics
Animation
Video
Audio

Contents

Scope

Scope

1. Real-Time Characterization

Real-time process:

> ***"A process which delivers the results of the processing in a given time-span."***

Real-time system:

> ***"A system in which the correctness of a computation depends not only on obtaining the right result, but also upon providing the result on time."***

Real-time application:

- **Example: Control of temperature in a chemical plant**
  - Driven by interrupts from an external device
  - These interrupts occur at irregular and unpredictable intervals
- **Example: control of flight simulator**
  - Execution at periodic intervals
  - Scheduled by timer-service which the application requests from the OS

Scope

Scope

Deadlines

A deadline represents the <u>latest acceptable time</u> for the presentation of a processing result

Soft deadlines:

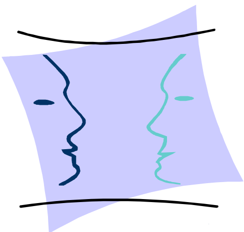
- **in some cases the deadline is missed**
  - not too many deadlines are missed
  - deadlines are not missed by much
- **presented result has still some value**
- **Example: train/plain arrival-departure**

Hard deadlines:

- **should never be violated**
  - violation means system failure
- **too late presented result has no value**

Critical:

- **violation means severe (potentially catastrophic) system failure**



Scope

Real-Time Operating System – Requirements

Real Time

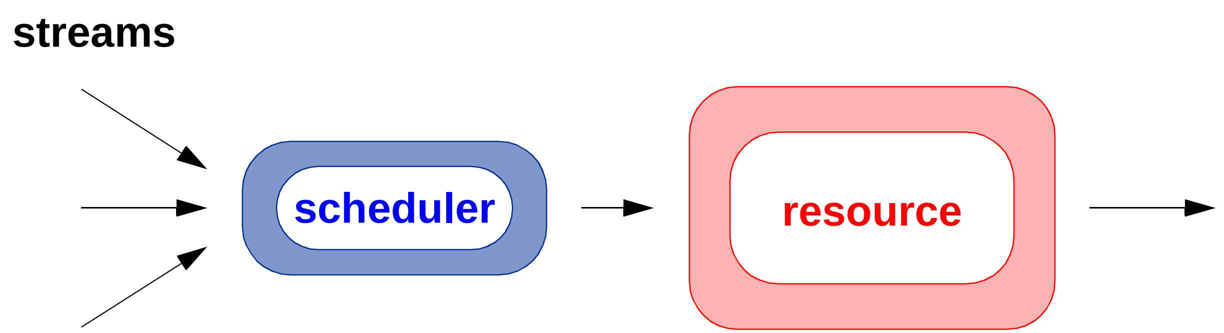
- Multi-tasking capabilities
- Short interrupt latency
- Fast context switch
- Control of memory management
- Proper scheduling
- Fine-granularity of timer services
- Rich set of interprocess communication mechanisms

Real-Time and Multimedia

- Typically soft real-time and
- Not critical
- Periodic processing requirements
- Large bandwidth requirements

Scope

2. Resource Scheduling: Motivation



Resource:

- **active:** **like CPU, network protocol, ...**
- **passive:** **like bandwidth, memory, ...**

Scheduler:

- **One for each active resource: esp. CPU, network**
- **Multiplexes resource between:**
  - Processing requests from different multimedia streams
  - Other processing requests
- **Determines order by which requests are serviced**

  ? scheduling algorithm

Scope
Scope

Scheduler Requirements

Support QoS scheme:

- **Allow calculation of QoS guarantees**
- **Enforce given QoS guarantees**
  - support high, continuous media data throughput
  - take into account for deadlines

Account for stream-specific properties:

- **Streams with periodic processing requirements**
  - real-time requests
- **Streams with aperiodic requirements**
  - should not starve multimedia service
  - should not be starved by multimedia service

Scope

Overall - Approach

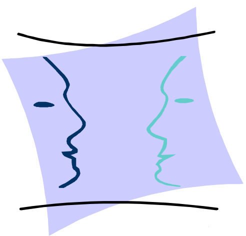
Adapt real-time scheduling to continuous media

- **Deadline-based (EDF) and rate-monotonic (RM)**
- **Preemptive and non-preemptive**

Exploit resource-specific properties, e.g.:

- **CPU:priority scheme supported by operating system**
- **Token Ring:MAC priority scheme**
- **FDDI:synchronous mode traffic**

? Priority-based schemes are of special importance



Scope

Priorities

Overall priorities account for importance of traffic, e.g.:

1. Multimedia traffic with guaranteed QoS

2. Multimedia traffic with predictive QoS

3. Other processing requests

Within classes 1 and 2:
Second-level scheduling scheme **to distinguish between streams,**
e.g. EDF, RM, fine-grained priorities

Preemptions

Preemptive scheduling:

- **Running process is preempted when process with higher priority arrives**
- **For CPU scheduling: often directly supported by operating system**
- **Overhead for process switching**

Non-preemptive scheduling:

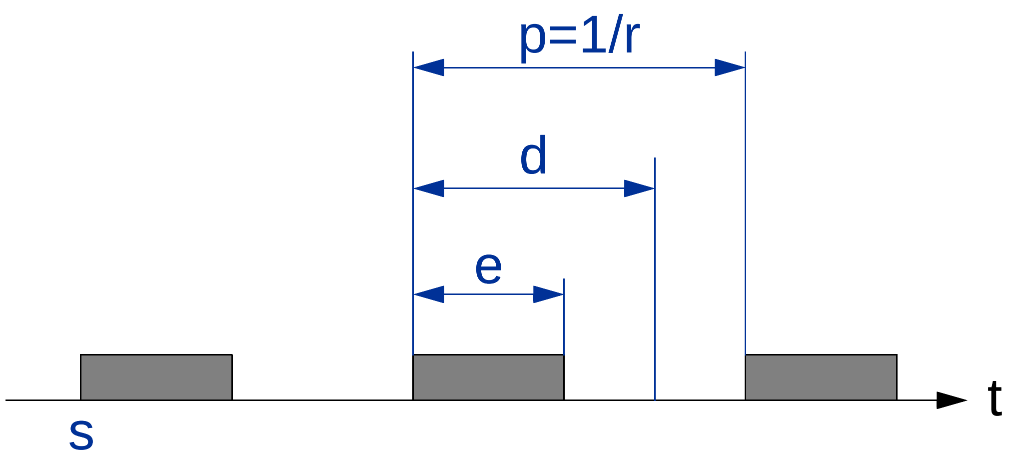
- **High-priority process must wait until running process finishes**
- **Inherent property of, e.g., the network**
- **Less frequent process switches**

? Non-preemptive scheduling can be the better choice if processing times are short

Scope

3. Properties of Multimedia Streams

Periodic stream model:



Packets of stream i:

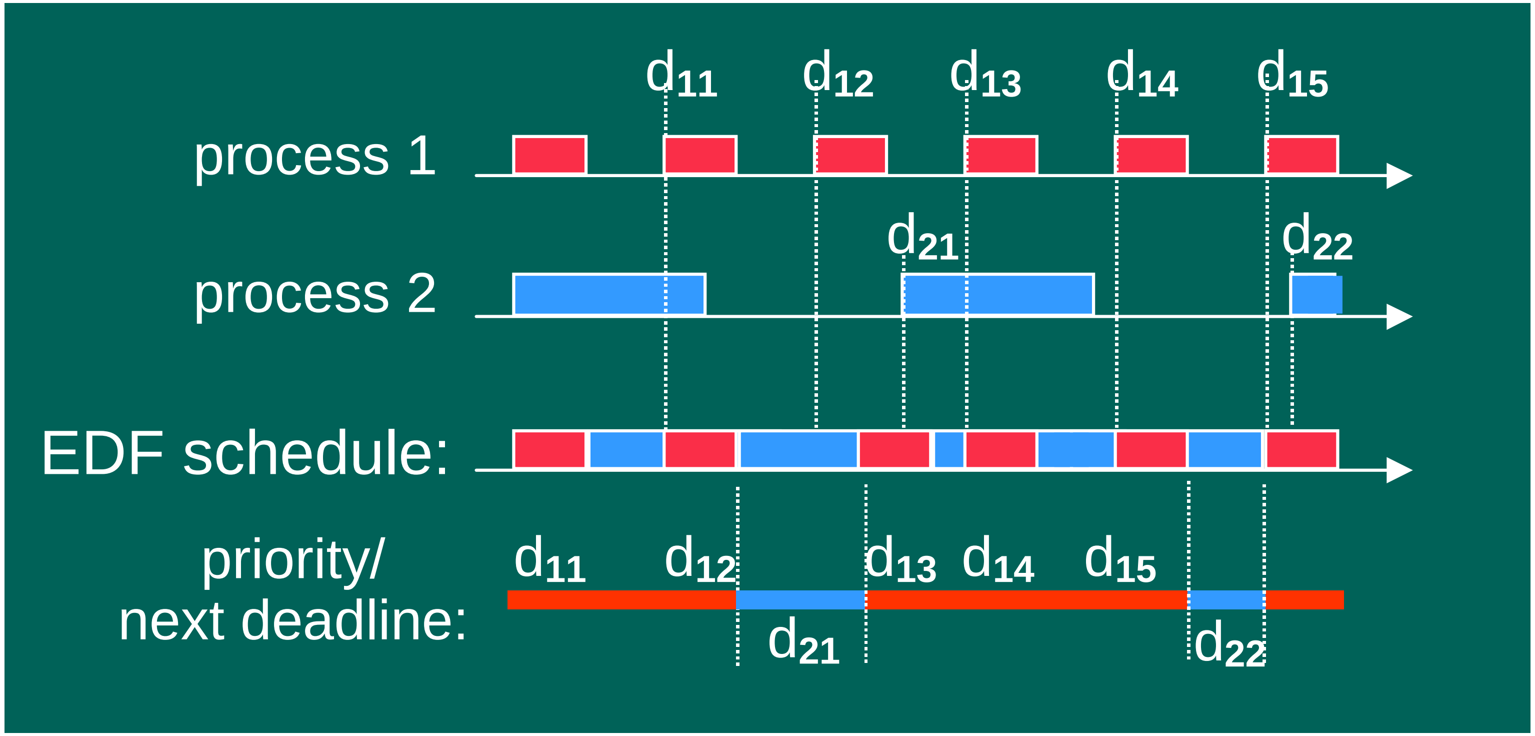
- **Begin at time s_i**
- **Arrive with rate r_i (i.e. r_i packets per time unit)**
- **Require processing time e_i**
- **Must be finished at deadlines d_{ij}**

? Scheduling algorithm must account for these properties

4. Deadline-Based Scheduling – EDF

Process priority determined by process deadline:

- **Process with closest deadline has highest priority**



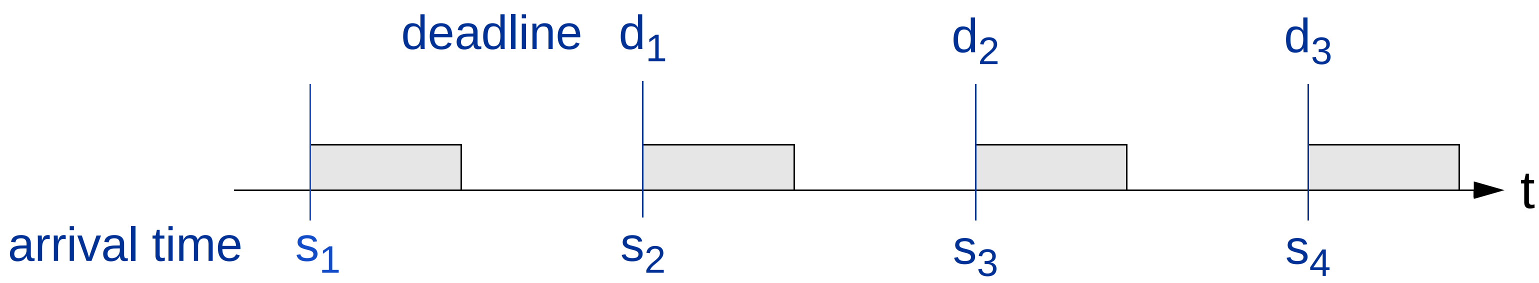
? Stream priorities vary with time

Scope

Scope

Deadline-Based Scheduling

(Assumption for most research projects): deadline = end of period:



QoS calculation:

- **Preemptive scheduling (Liu / Layland, 1973):**
  - maximum allowable throughput (limit for accepting scheduling requests):

$$\sum_{\text{all streams } i} \frac{e_i}{p_i} \leq 1$$

  - packet delay $\leq p_i$
- **Non-preemptive scheduling (Nagarajan / Vogt, 1992):**
  - same throughput as above
  - packet delay <= $1/p_i + e$, where e is the (unique) processing time for a packet

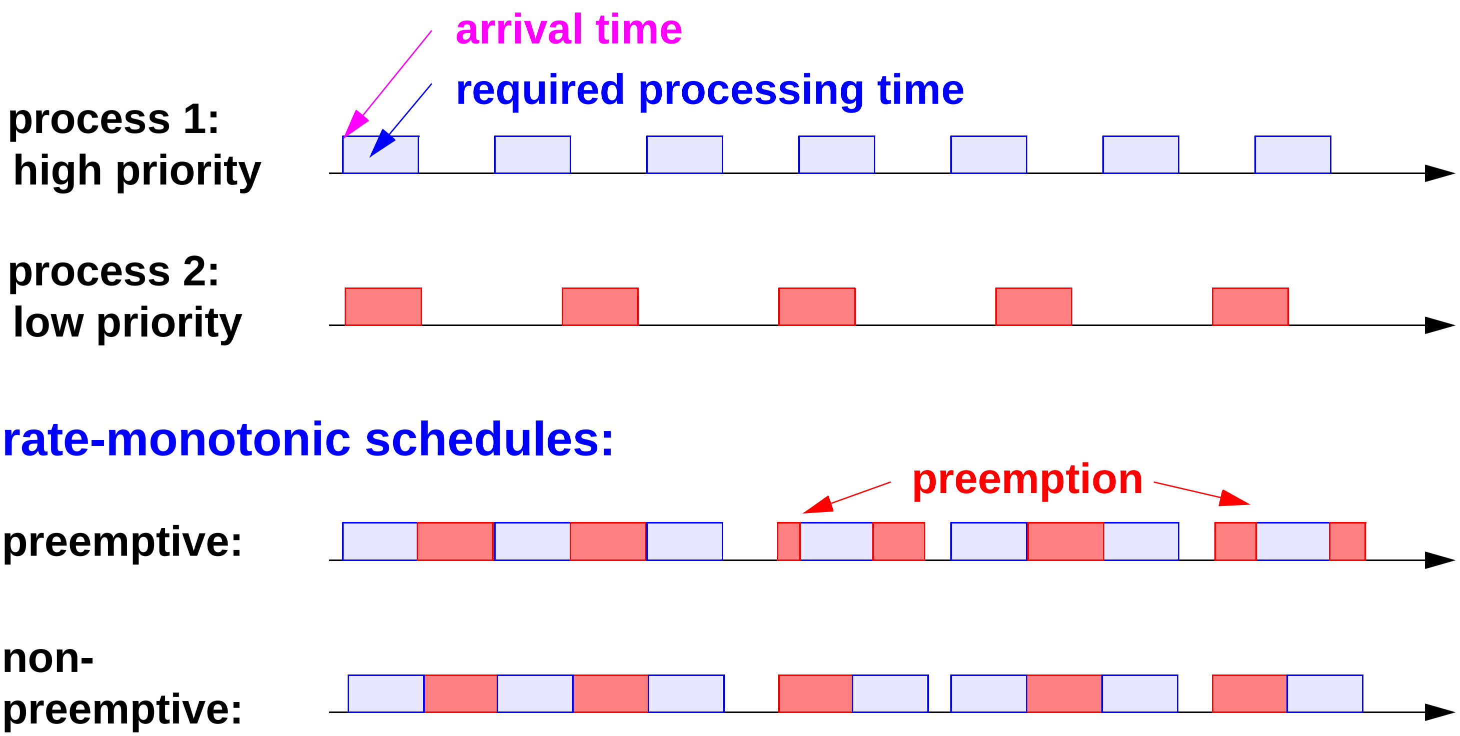
Scope

Scope

5. Rate-Monotonic Scheduling

Process priority determined by packet rate:

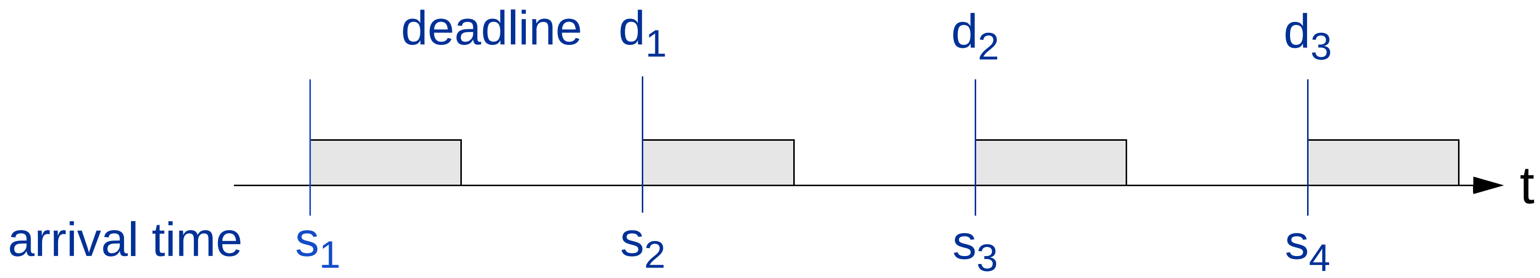
- **Process with highest rate has highest priority**



? Relative stream priority is fixed

Rate-Monotonic Scheduling

Deadline = end of period (same as for deadline-based sched.):



QoS calculation:

- **Preemptive scheduling (Liu / Layland, 1973):**
  - maximum allowable throughput:

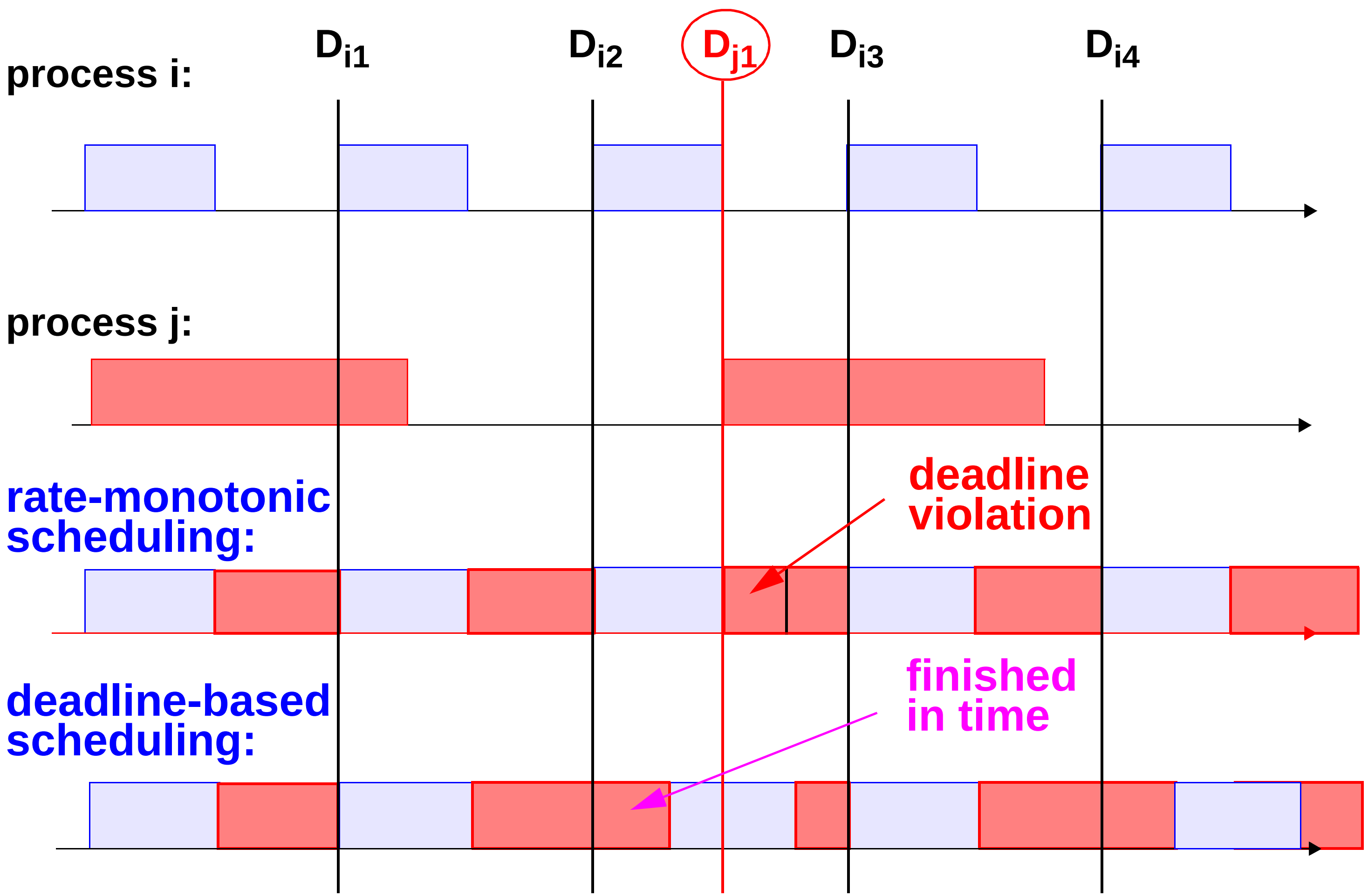
$$\sum_{\text{all streams } i} \frac{e_i}{p_i} \leq \ln 2$$

  - packet delay $\leq p_i$
- **Non-preemptive scheduling (Nagarajan / Vogt, 1992):**
  - formulae significantly more complex
  - guaranteed throughput significantly lower

Scope

Scope

6. Deadline-Based vs. Rate-Monotonic Scheduling

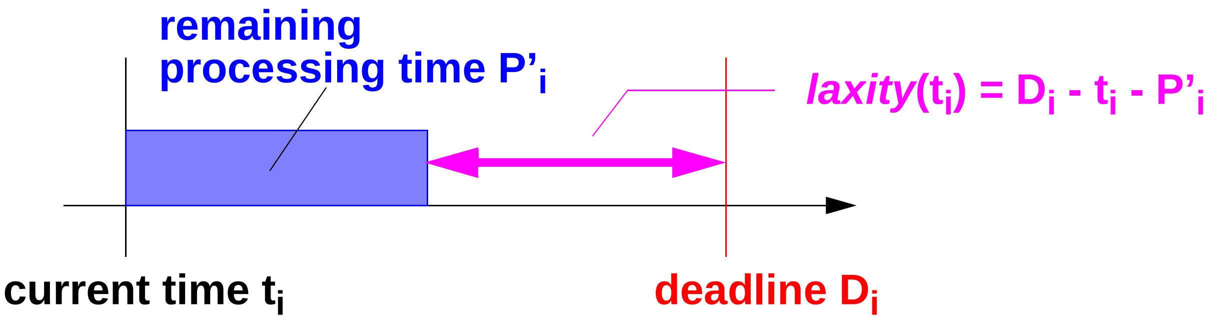


7. Some Other Scheduling Algorithms

Fixed-priority scheduling:

- **For each stream: fixed priority**
  - Rate-monotonic scheduling is special case
- **Delay calculation for one stream based on worst-case assumptions about streams with higher priority**

Laxity-based scheduling:



- **Stream with lowest laxity has highest priority**
- **Dynamically changing priorities**
- **Improvement over EDF in cases where s_i (process n) = s_j (process m)**

Other examples:

- **"shortest-job-first" SJF: improvement for overload conditions**
- **Scheduling for Imprecise Computations**
- **Sporadic Servers**

8. Execution Architecture – System Structure

Problem:

- **How to structure software that is to be scheduled?**

Distinction of functions into separate environments:

- **Non-real-time**
- **Real-time**

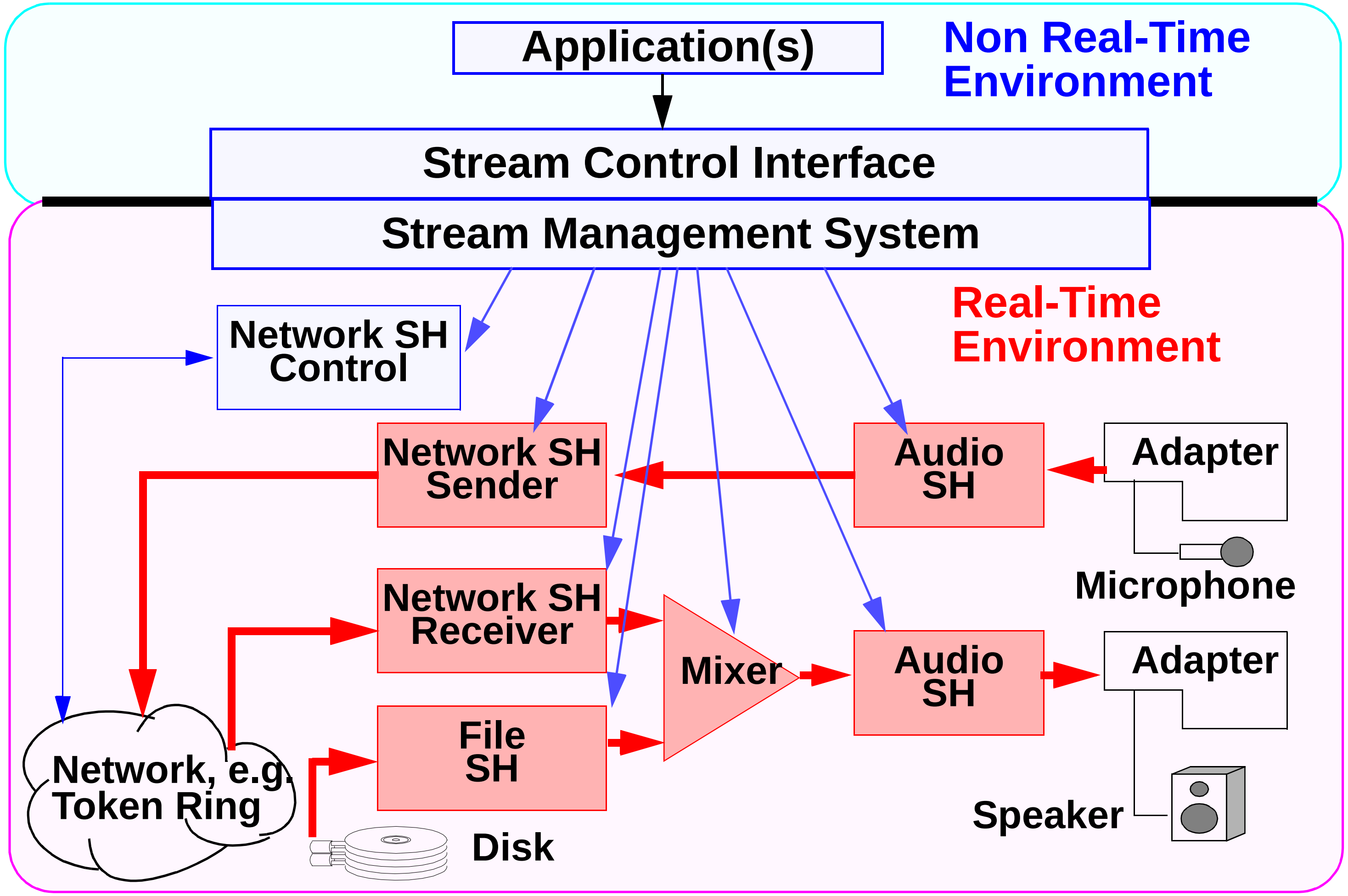
Structuring into software modules:

- **Application builds user interface**
- **Software modules (Stream Handlers „SH“) perform real-time functions**
- **I.e.: application is based on system of connected stream handlers**
- **Advantages:**
  - predefined stream handlers can be better controlled than user-written software
  - application-writing is easier

Scope

Scope

System Structure Example

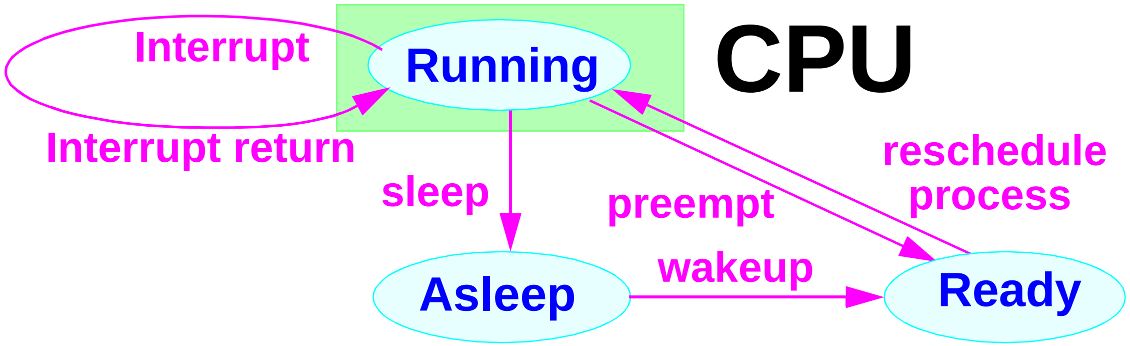


9. CPU Requirement Estimation: CPU Utilization of Software Modules

QoS calculation needs knowledge of required CPU capacity

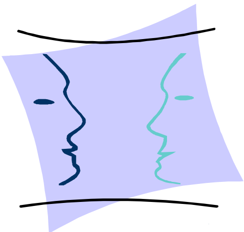
Definition:

> ***The <u>CPU utilization</u> t_i of a software module is the time during which the processor executes code of this module or code for the management of this execution.***

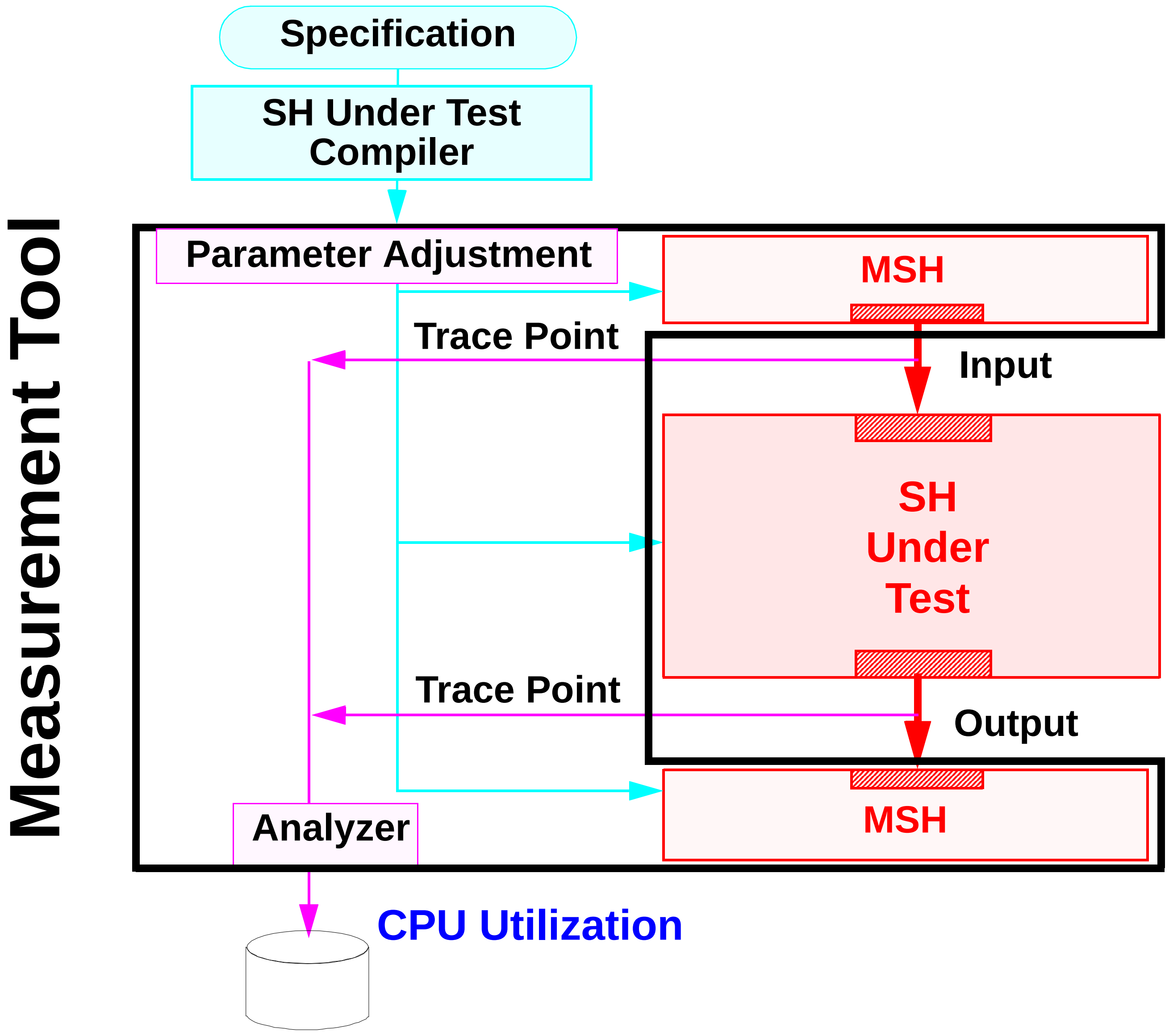


- **Analytical calculation is very difficult**
- **Measurement tool required**

http://www.kom.e-technik.tu-darmstadt.de
http://www.tk.informatik.tu-darmstadt.de



Measurement Tool Architecture



Scope

Scope

10. Operating System Support

Operating system manages local resources:

- **CPU**
- **Memory space**
- **File system**

? To be distinguished from network resources used for data transmission

Operating system support required for:

- **Real-time processing**
- **Memory management**

Scope

Scope

10.1 Issues in Operating System Support - Examples

Fixed-priority scheduling:

- High fixed priorities for multimedia streams
- Management by special multimedia scheduler
- No impact of operating system (non-real-time) scheduler

Timer support:

- Clock with high granularity
- Event scheduling with high accuracy

Kernel preemption:

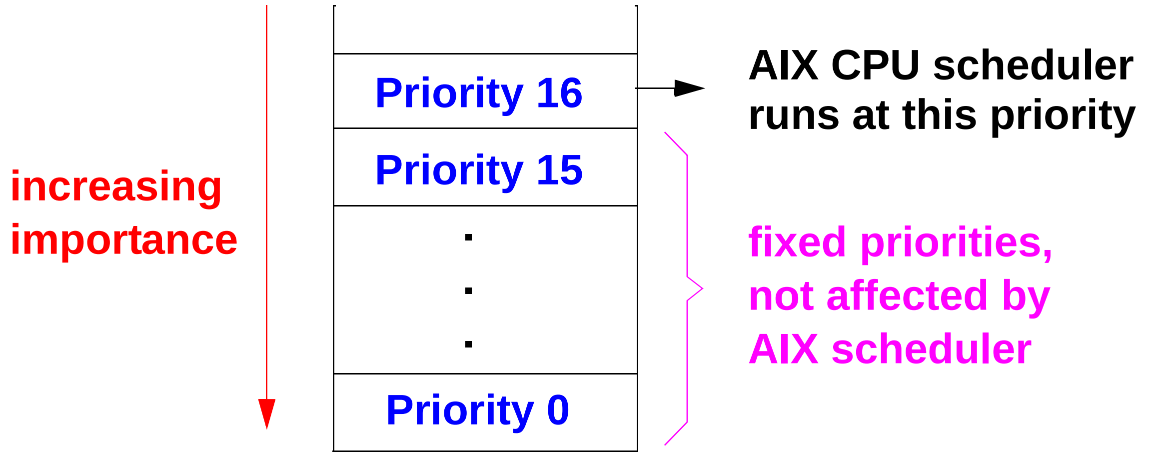
- Avoid long periods where a low-priority process cannot be interrupted

Memory pinning:

- Prevents code for real-time programs from being paged out

Operating System Example: AIX™

Fixed CPU priorities:



High-granularity timers:

- Logical granularity: 1 ns
- Current implementation: 256 ns

Preemptive kernel

Pinning of pages in main memory

Scope

Operating System Example: Windows NT

Fixed CPU priorities:

- **Real-time scheduler can be implemented**
- **Dominates the original scheduler**

High-granularity timers:

- **Granularity of 1 ms**

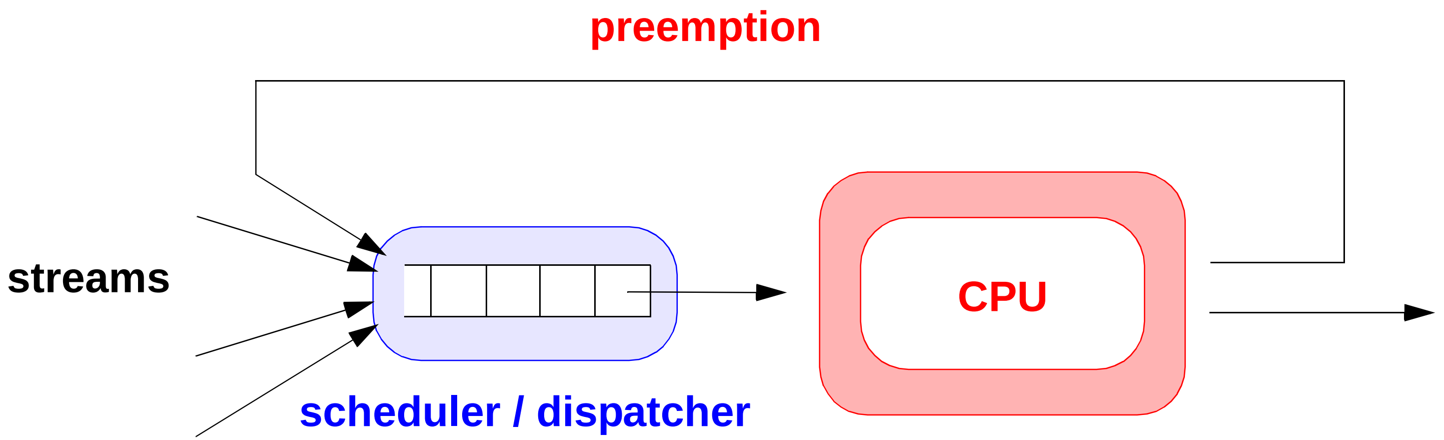
Preemptive kernel

Pinning of pages in main memory

Scope

Scope

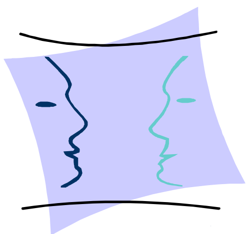
10.2 CPU Management: Scheduling Scenario



Packets on a number of streams wait for local processing (e.g., execution of protocol stack, compression algorithms)

Scheduler / Dispatcher:

- **Assigns relative priorities to waiting packets**
- **Submits packet with highest priority for execution**
- **Preempts current execution when more urgent packet arrives**



CPU Management: Scheduling Algorithms

Rate-Monotonic Scheduling:

- **Implementation:**
  - relative priority of a stream remains fixed
  - map stream priorities to fixed operating system priorities (as in AIX)
- **QoS calculation based on Liu / Layland formulae**

Deadline-Based Scheduling:

- **Implementation:**
  - dynamic process priorities require frequent priority switches
  - considerable overhead in operating systems with static system priorities
- **QoS calculation based on Liu / Layland formulae**

Scope

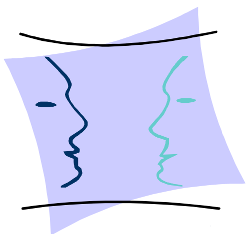
10.3 Memory Management

Main memory is needed for several purposes:

- **to store code of applications and system components such as OS kernel,**
- **to store data structures, e.g., for state of this software,**
- **to store data on which processing is done, e.g., a video frame**

Required features:

- **page faults take too much time & introduce large variations into processing times**
- **thus: pinning of memory**
  - not only application code, but functions used by it inside libraries, OS kernel, etc. as well
  - not always possible and pinning large memory areas reduces overall performance
  - also contrary to trend in workstation OS to provide for paging of kernel code
- **Reservation of main memory (“buffer”) space to avoid data loss**



Scope

Scope

Buffer space calculation:

- **Depends on input traffic& packet delay**

Actual reservation by operating system functions

Example for a periodic stream:

Rate: R [packets per second]
Burst: B [packets in excess to rate]
Maximum packet size:S [bytes]
Maximum local delay:D [seconds]
$\Rightarrow$ Required buffer space= $(R*D + B) * S$ bytes

Memory Management (2)

Data movement costs should be kept small

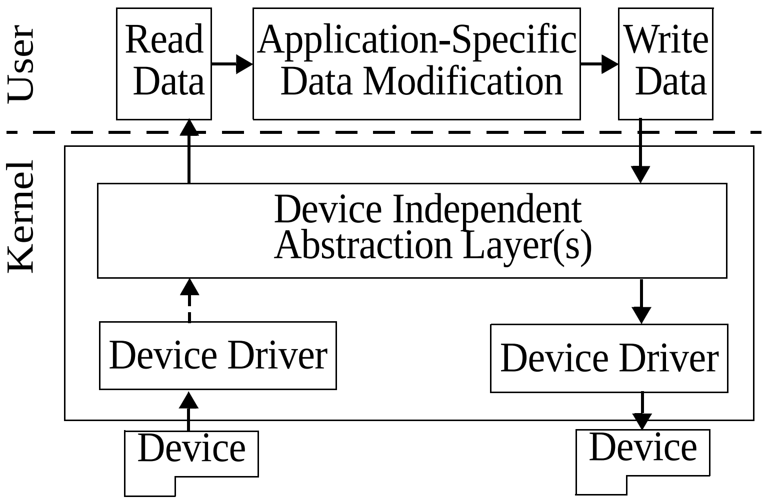
- **handle continuous-media data carefully**
- **avoid unnecessary physical data movements**
- **apply buffer management schemes which use, e.g., scatter/gather techniques**
- **potentially also between kernel and user level (or use remapping by virtual memory operations)**

In future, ’streaming mode’ might be offered

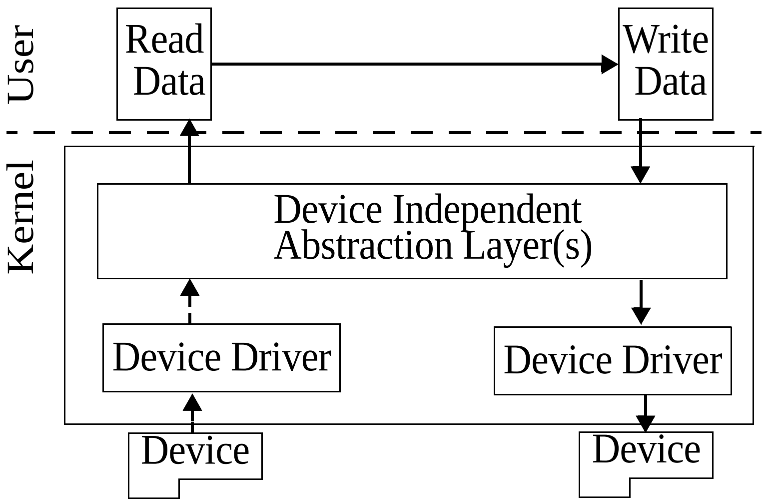
- **data flows directly from source to sink device in application specified manner**
- **two different approaches possible**
    - ’application streaming’: new system calls (read_stream, write_stream)
        - read data from device into kernel buffer (and leave the data inside the kernel)
        - write it from that buffer to a device
        - application is responsible for timing of I/O operations
    - ’kernel streaming’: create new kernel thread per stream
        - performs read and write operations
        - application specifies timing of stream and thread ensures that this is met
        - application mainly controls the thread

Memory Management (3) – Streaming Modes

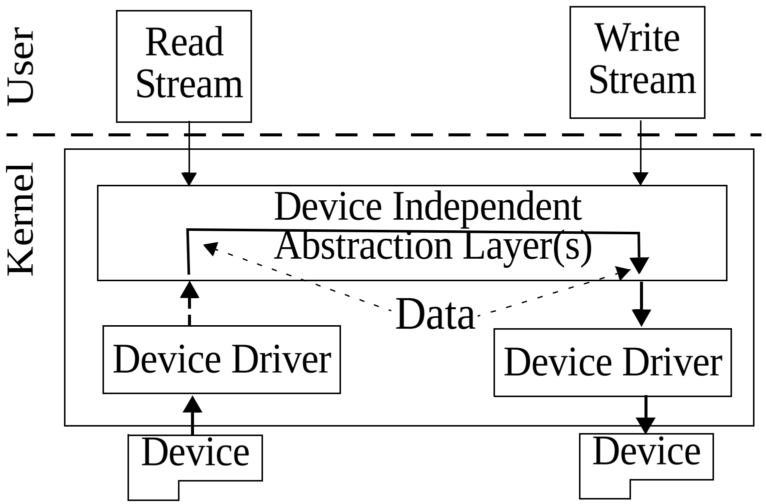
Traditional Application



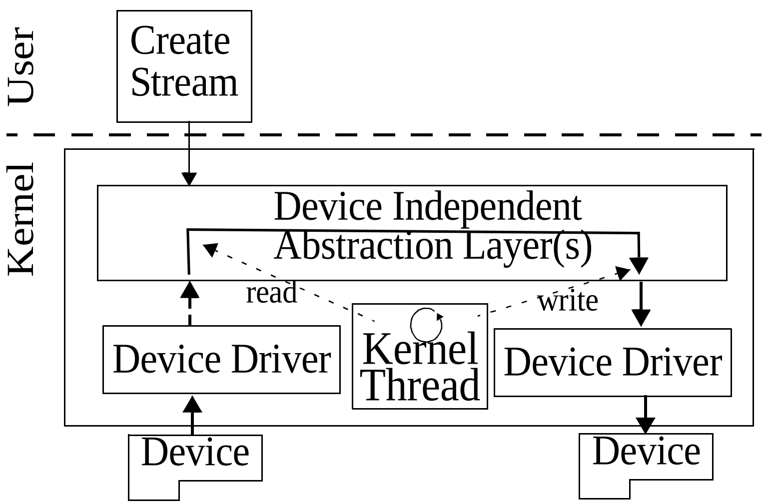
Streaming Application



Application-Streaming



Kernel-Streaming



Scope
Scope

10.4 Existing Operating Systems: Difficulties (1)

Extensions have been developed for

- **real-time processing, stream-handling, etc.**
- **to handle audio-visual data streams**

but problems remain especially for resource accounting

- **what happens when, by whom, and how**
  - which user, which application, and which task …
  - … uses how much resources
  - with fine granularity
  - and low overhead and influence on the system performance
- **necessary for exact**
  - admission control, schedulability tests
  - QoS monitoring, resource control, charging
  - better scheduling decisions with adaptive schemes

Restrictions due to the basic design and structure of the OS

Scope

Existing Operating Systems: Difficulties (2)

Reasons:

- **Processing in OS kernel, interrupt handlers, server processes, …**
- **Current OS do not provide sufficient support for fine granular measurements**
    - typically not more than start and stop times of tasks in a period (often with coarse granularity in the order of several milliseconds only)
    - not resource usage time – differences due to other tasks / system activities in meantime

A relatively simple and cheap approach:

- **introduce a task state variable Di which contains the run-time of the task i**
    - System-wide variable E holds time stamp of last context switch or interrupt
    - As part of the creation of a new task *j* the variable *Dj* is set to 0
    - while performing a context switch from task *k* to *l*
- **helps for determination of processing time requirements of tasks**
- **allows to check whether tasks stay (reasonable) within their specifications**
- **But: no support to accumulate resource usage in summary for particular appl.**
- **Resource usage of server tasks (executing on behalf of this application) must also be taken into account**

Scope

10.5 New Architectures for 'Multimedia Operating Systems'

Entirely new operating system

- **geared to support time-sensitive applications requiring consistent QoS**
- **provides fine-grained guaranteed levels of all system resources including**
    - CPU, memory, network bandwidth and disk bandwidth

Majority of code could executes in the application process itself:

- **extremely small lightweight kernel**
- **most OS functions in shared libraries which execute in user's process**
- **vertically-structured operating system**

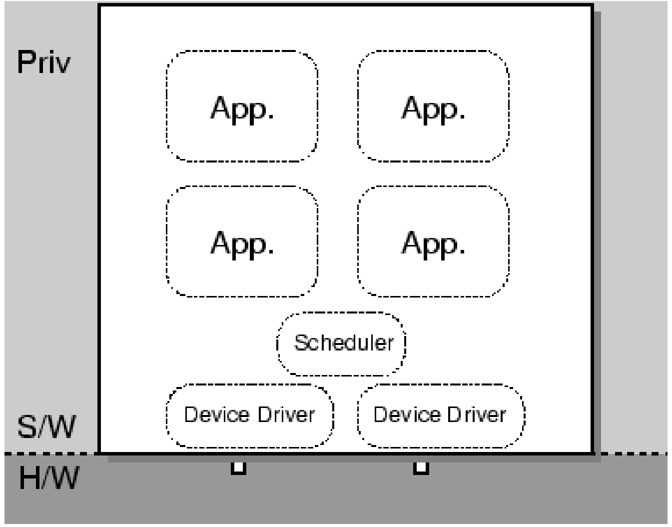
Use of single address space:

- **greatly reduces memory-system related context-switch penalties**
- **removes the need to copy high-bandwidth multimedia data**
- **memory protection is still performed on a per-process basis**

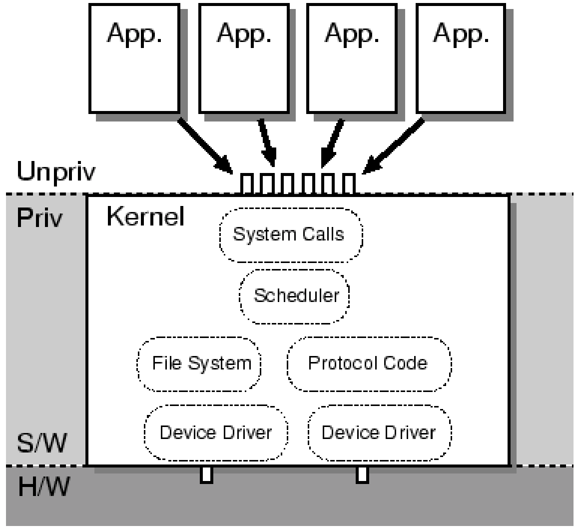
Scope

Scope

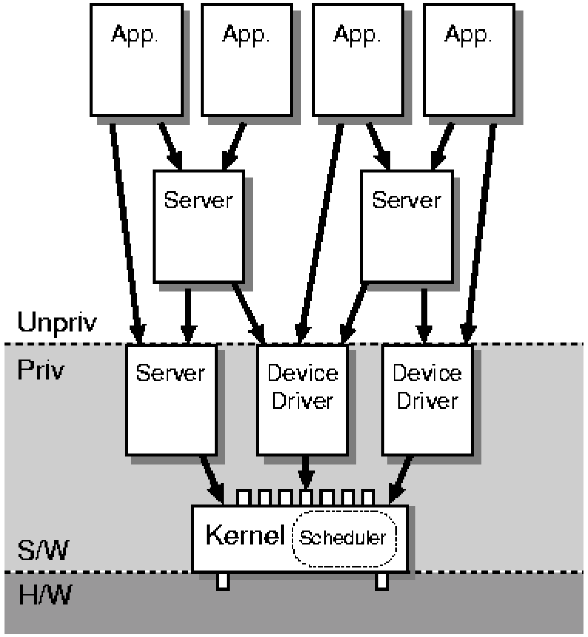
Comparison of OS Structures



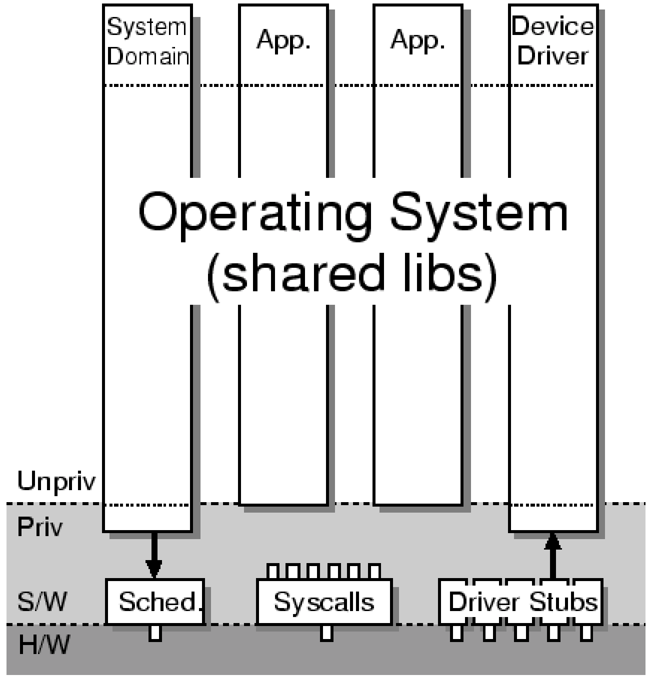
Monolithic



Kernel



Microkernel



Nemesis

11. Conclusions

Scheduling mechanisms have to:

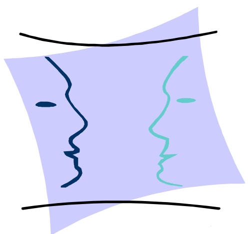
- Consider real-time requirements of multimedia applications
- Be implementable
- Provide good resource utilization

Resources to be scheduled:

- Local resources (esp. CPU): by operating system
- Network resources: Network protocols, network adapters

Memory management:

- Reservation of „buffer“ memory to avoid data losses
- Pinning data and program code in physical storage



Scope