UNIVERSIDAD DE CANTABRIA

Predictable response times in event-driven real-time systems

Automotive 2006 - Security and Reliability in Automotive Systems

Stuttgart, October 2006. Presented by:

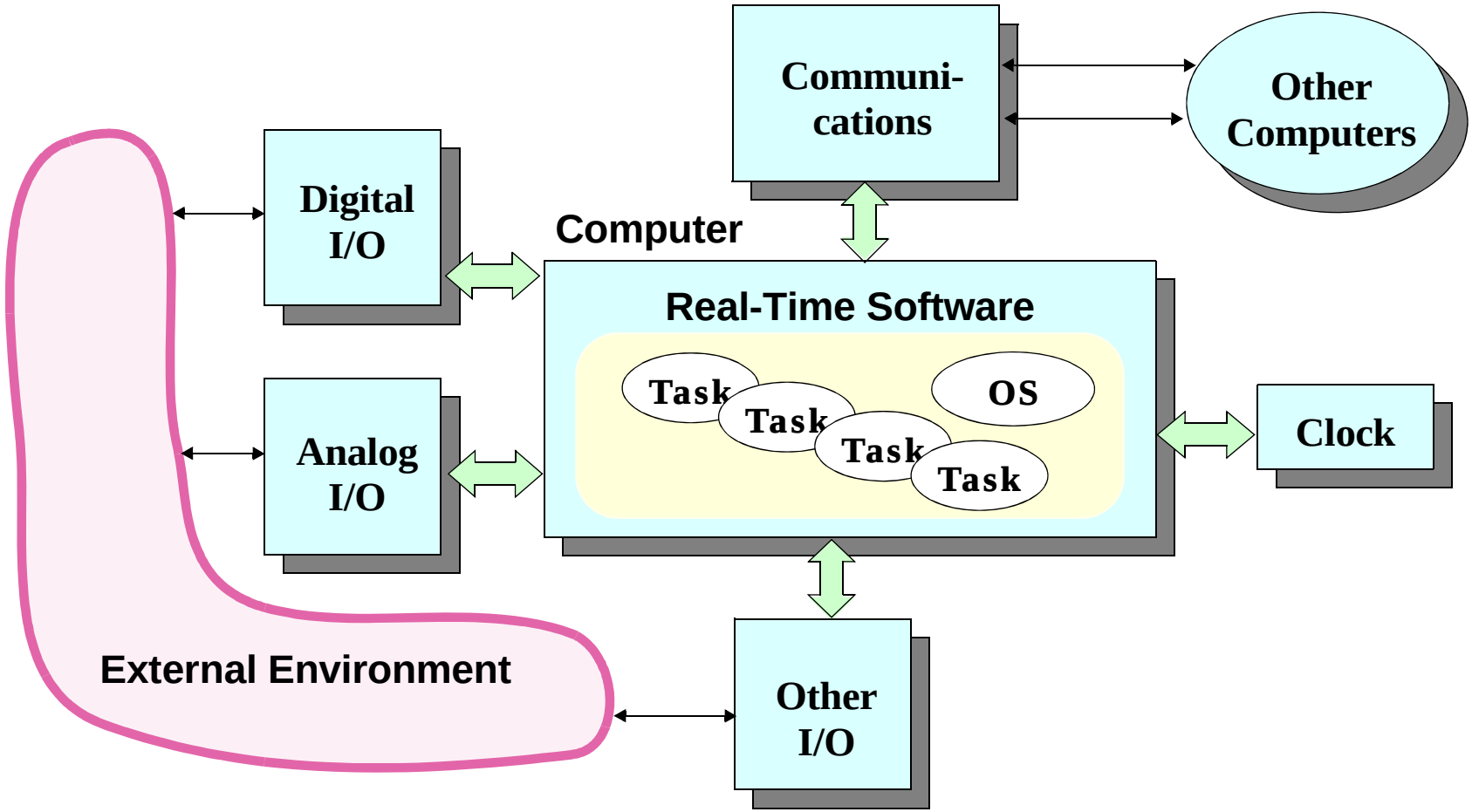
Michael González Harbour
mgh@unican.es
www.ctr.unican.es

Predictable response times in event-driven real-time systems

UNIVERSIDAD DE CANTABRIA

Elements of a real-time system

UNIVERSIDAD DE CANTABRIA



Real-time systems

UNIVERSIDAD DE CANTABRIA

A Real-time system is a combination of a computer, hardware I/O devices, and special-purpose software, in which:

- **there is a strong interaction with the environment**
- **the environment changes with time**
- **the system simultaneously controls and/or reacts to different aspects of the environment**

As a result:

- **timing requirements are imposed on software**
- **software is naturally concurrent**

To ensure that timing requirements are met, the system's timing behavior must be *predictable*

What's important in real-time

UNIVERSIDAD DE CANTABRIA

Predictability **of the response time**

Criteria **for real-time systems differ from that for time-sharing systems.**

	Time-Share Systems	Real-Time Systems
Capacity	High throughput	Ability to meet timing requirements: Schedulability
Responsiveness	Fast average response	Ensured worst-case latency
Overload	Fairness	Stability of critical part

Worst case cannot be checked by ***testing***

Options for managing time

UNIVERSIDAD DE CANTABRIA

Compile-time schedules:

- time triggered or cyclic executives
- predictability through static schedule
- logical integrity often compromised by timing structure
- difficult to handle aperiodic events & dynamic changes
- difficult to maintain

Run-time schedules:

- priority-based schedulers
- preemptive or non preemptive
- analytical methods needed for predictability
- separates logical structure from timing
- more flexibility

Fixed-priority scheduling (FPS)

UNIVERSIDAD DE CANTABRIA

Fixed-priority preemptive scheduling is very popular for practical applications, because:

- **Timing behavior is simpler to understand**
- **Behavior under transient overload is easy to manage**
- **A complete analytical technique exists**
- **High utilization levels may be achieved (typically 70% to 95% of CPU)**
- **Supported in standard concurrent languages or operating systems:**
  - **Ada 2005's RT-annex, Java RTSJ**
  - **Real-time POSIX**

Dynamic priority scheduling

UNIVERSIDAD DE CANTABRIA

Most common policy is earliest deadline first (EDF):

- **the priority varies with time, because the task with the closest absolute deadline is chosen first**
- **100% CPU utilization may be achieved**

However:

- **behavior is more complex, and so is the analysis**
- **treatment of transient overload is more complex**
- **not supported by standard operating systems**
  - **supported in Java RTSJ**
  - **recently added to Ada 2005**

Mixed EDF/FPS schemes are possibly the best approach

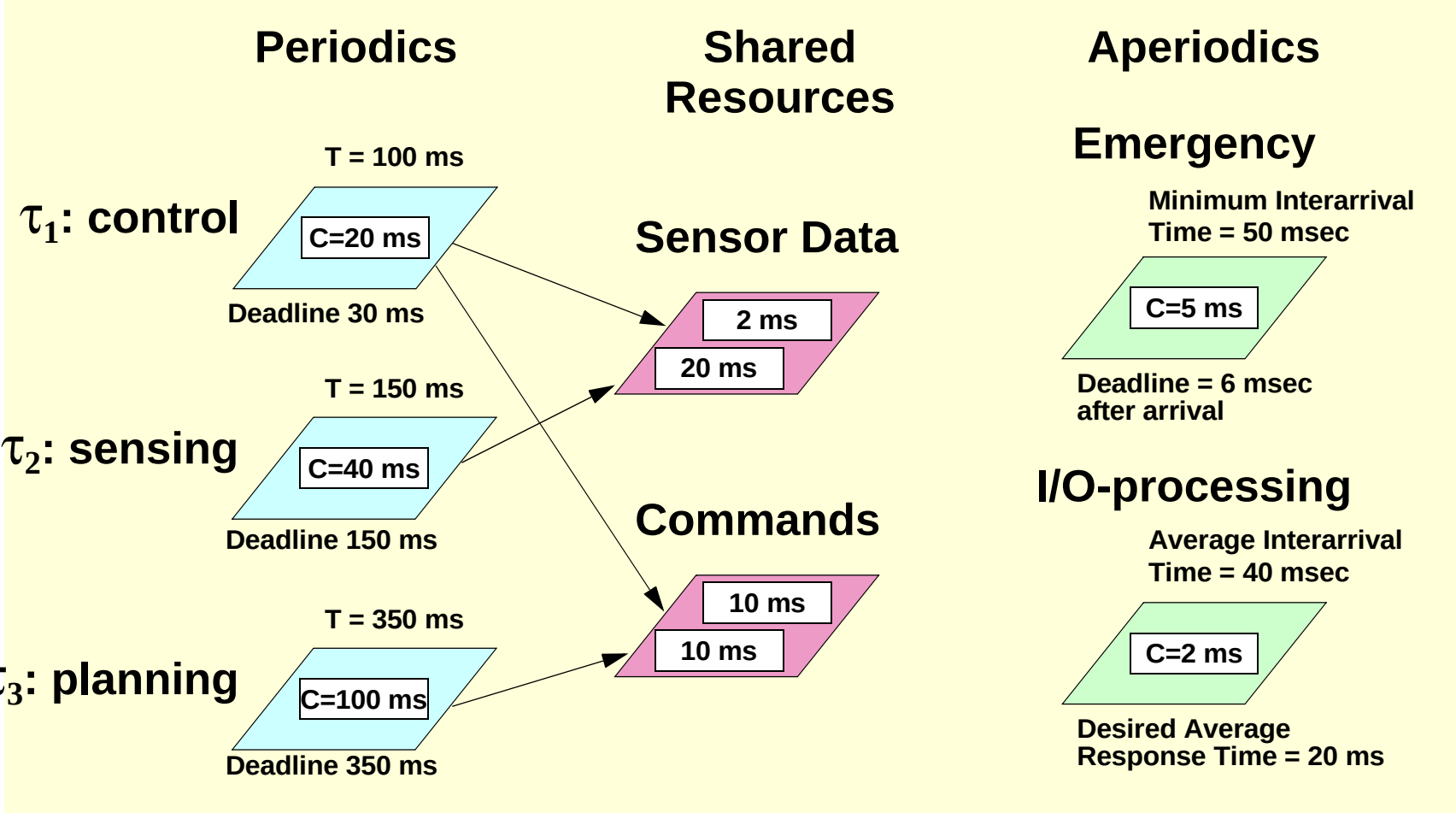
Predictable response times in event-driven real-time systems

UNIVERSIDAD DE CANTABRIA

1. Introduction
2. *Basic scheduling*
3. Mutual exclusion
4. Distributed systems
5. Hierarchical scheduling
6. Protection and flexible scheduling frameworks
7. Operating systems
8. Programming languages
9. Modeling and integration into the design process
10. Conclusions

Activities in a basic real-time system: example

UNIVERSIDAD DE CANTABRIA



Basic principles of real-time analysis

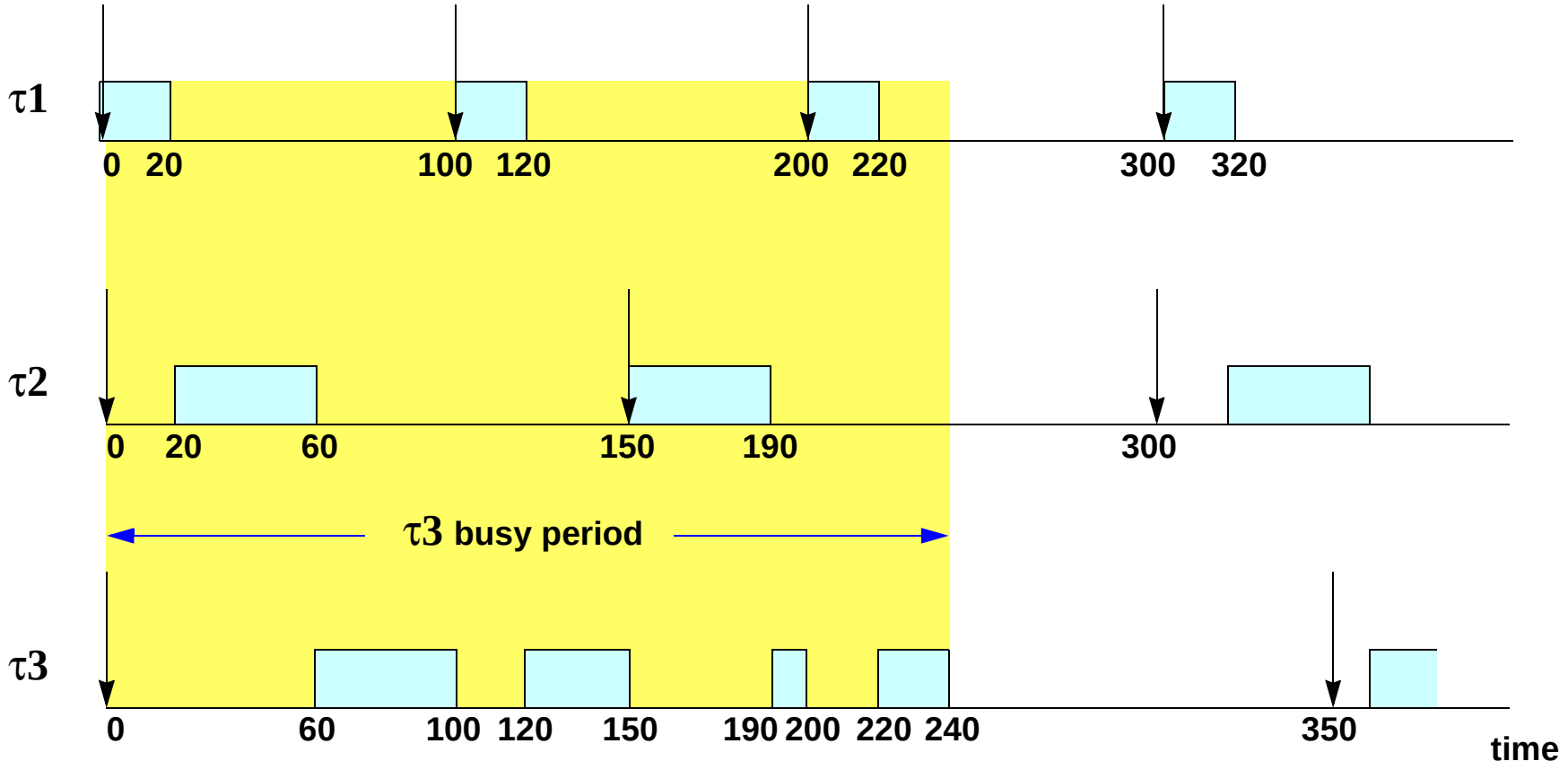
UNIVERSIDAD DE CANTABRIA

Two concepts help building the worst-case condition under fixed priorities:

- ***Critical instant*. The worst-case response time for all tasks in the task set is obtained when all tasks are activated at the same time**
- ***Checking only the deadlines in the worst-case busy period*.**
  - **for task: interval during which the processor is busy executing τ_i or higher priority tasks**

Example of a critical instant

UNIVERSIDAD DE CANTABRIA



Priority assignment

UNIVERSIDAD DE CANTABRIA

If $D_i \leq T_i$, *deadline monotonic* assignment

- **For a set of periodic independent tasks, with deadlines within the period, the optimum priority assignment is the deadline monotonic assignment:**
  - **Priorities are assigned according to task deadlines**
  - **A task with a shorter deadline is assigned a higher priority**

If $D_i > T_i$ for one or more tasks, Audsley's algorithm

- **iteratively apply analysis, successively ordering tasks by priority: O(n^2) times the analysis**

Utilization bound test (D=T) (Liu and Layland, 1973)

UNIVERSIDAD DE CANTABRIA

***Utilization Bound Test*: A set of *n* independent periodic tasks, with deadlines at the end of the periods, scheduled by the rate monotonic algorithm will always meet its deadlines, for all task phasings, if**

$$\frac{C_1}{T_1} + \ldots + \frac{C_n}{T_n} \leq U(n) = n(2^{1/n} - 1)$$

U(1) = 1.0U(4) = 0.756U(7) = 0.728
U(2) = 0.828U(5) = 0.743U(8) = 0.724
U(3) = 0.779U(6) = 0.734U(∞) = 0.693

For harmonic task sets, the utilization bound is *U(n)=1.00* for all *n*.

Response time analysis (Harter, 1984; Joseph and Pandya, 1986)

UNIVERSIDAD DE CANTABRIA

Iterative test (pseudopolynomial time):

$$a_0 = C_1 + C_2 + \ldots + C_i$$

$$a_{k+1} = W_i(a_k) = \left\lceil \frac{a_k}{T_1} \right\rceil C_1 + \ldots + \left\lceil \frac{a_k}{T_{i-1}} \right\rceil C_{i-1} + C_i + B_i$$

preemption

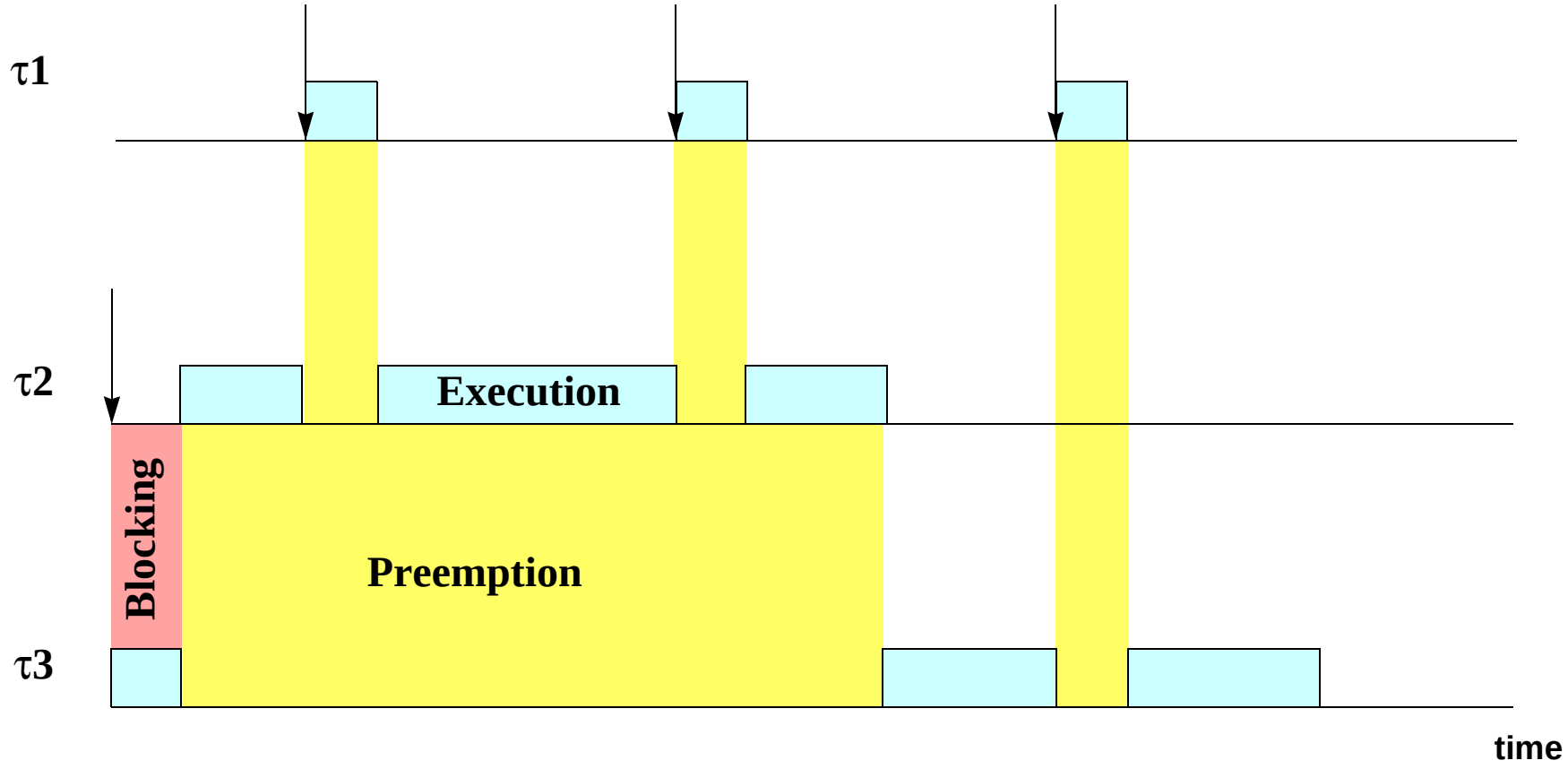
execution

blocking

Finish when two consecutive results are the same

Elements that influence the response time

UNIVERSIDAD DE CANTABRIA



Enhancements to response-time analysis (FPS)

UNIVERSIDAD DE CANTABRIA

Analysis with arbitrary deadlines (Lehoczky, 1990)

- **checking first deadline is not enough**
- **repeat analysis for multiple jobs in a busy period**

Analysis with release jitter (Tindell, 1992)

Analysis with overhead effects (context switch)

Methods for controlling input/output jitter

Response time analysis for EDF

UNIVERSIDAD DE CANTABRIA

***Busy period*: interval during which processor is not idle**

Worst case *response time* of a task: found in a busy period in which all other tasks:

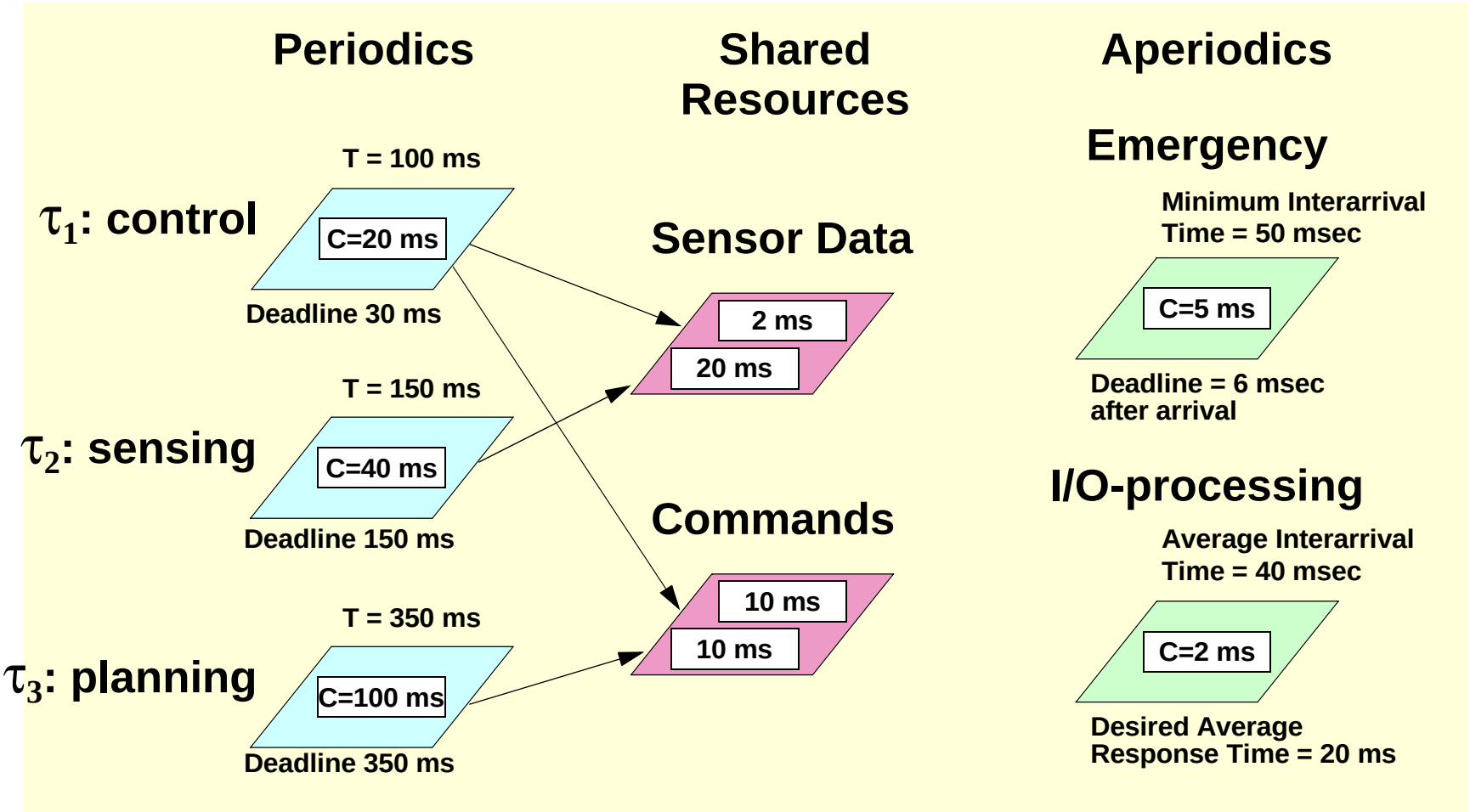
- **are released at the beginning of the busy period,**
- **and have experienced their maximum jitter**

Differences with fixed priorities:

- **The task under analysis does not necessarily start with the busy period**
- **The busy period is longer**

Handling aperiodic activities

UNIVERSIDAD DE CANTABRIA



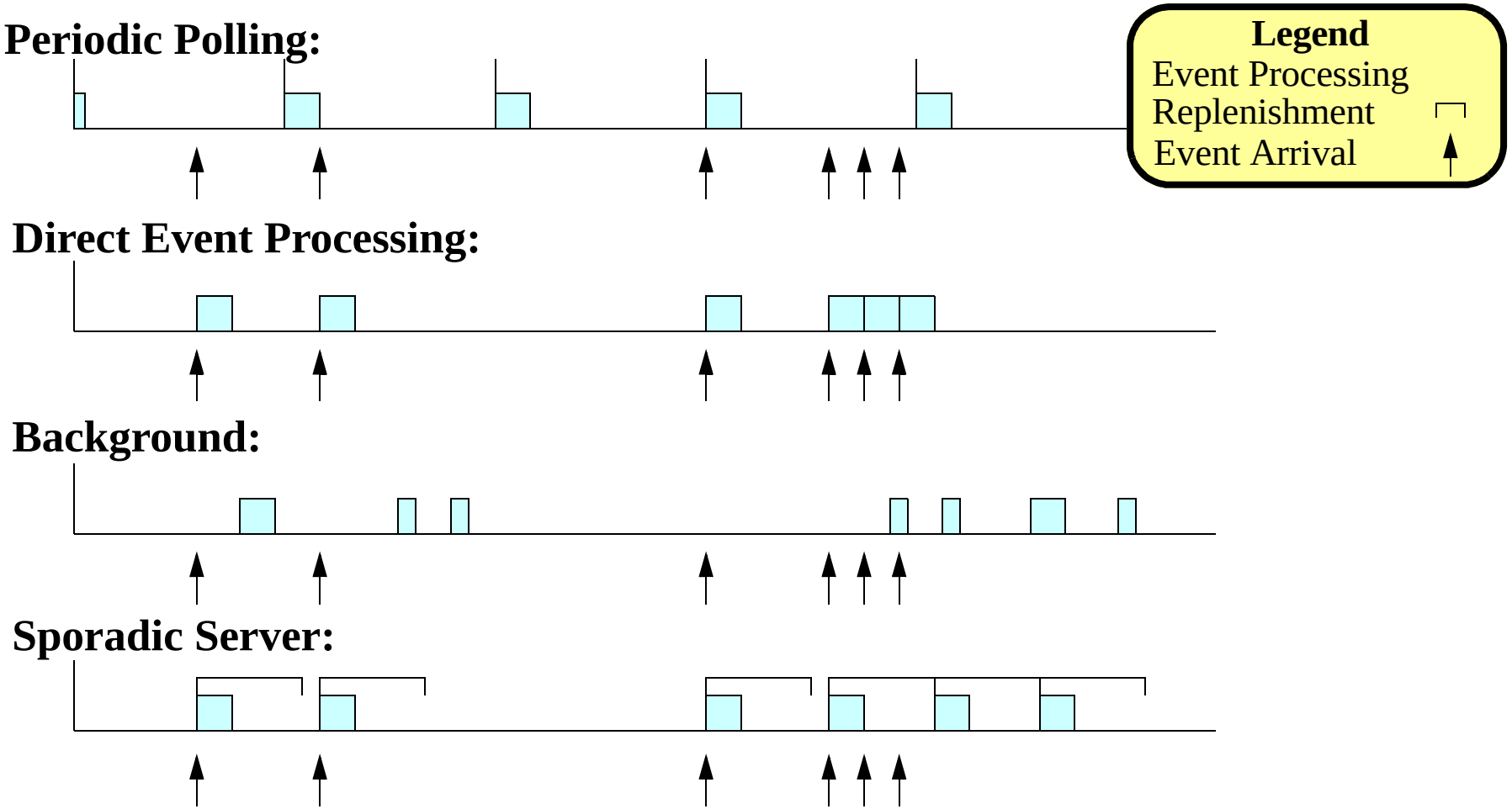
Handling aperiodic activities

UNIVERSIDAD DE CANTABRIA

The main problem is guaranteeing predictability in systems with unbounded aperiodic events

Scheduling aperiodic tasks

UNIVERSIDAD DE CANTABRIA



Periodic servers

UNIVERSIDAD DE CANTABRIA

Sporadic server (FPS)

- **Characterized by two parameters:**
  - **initial execution capacity (C_R)**
  - **replenishment period (T_R)**
- **It guarantees**
  - **a minimum bandwidth for aperiodic events**
  - **bounded preemption on lower priority tasks**

Servers also exist for EDF scheduling

- **Constant bandwidth servers (CBS)**

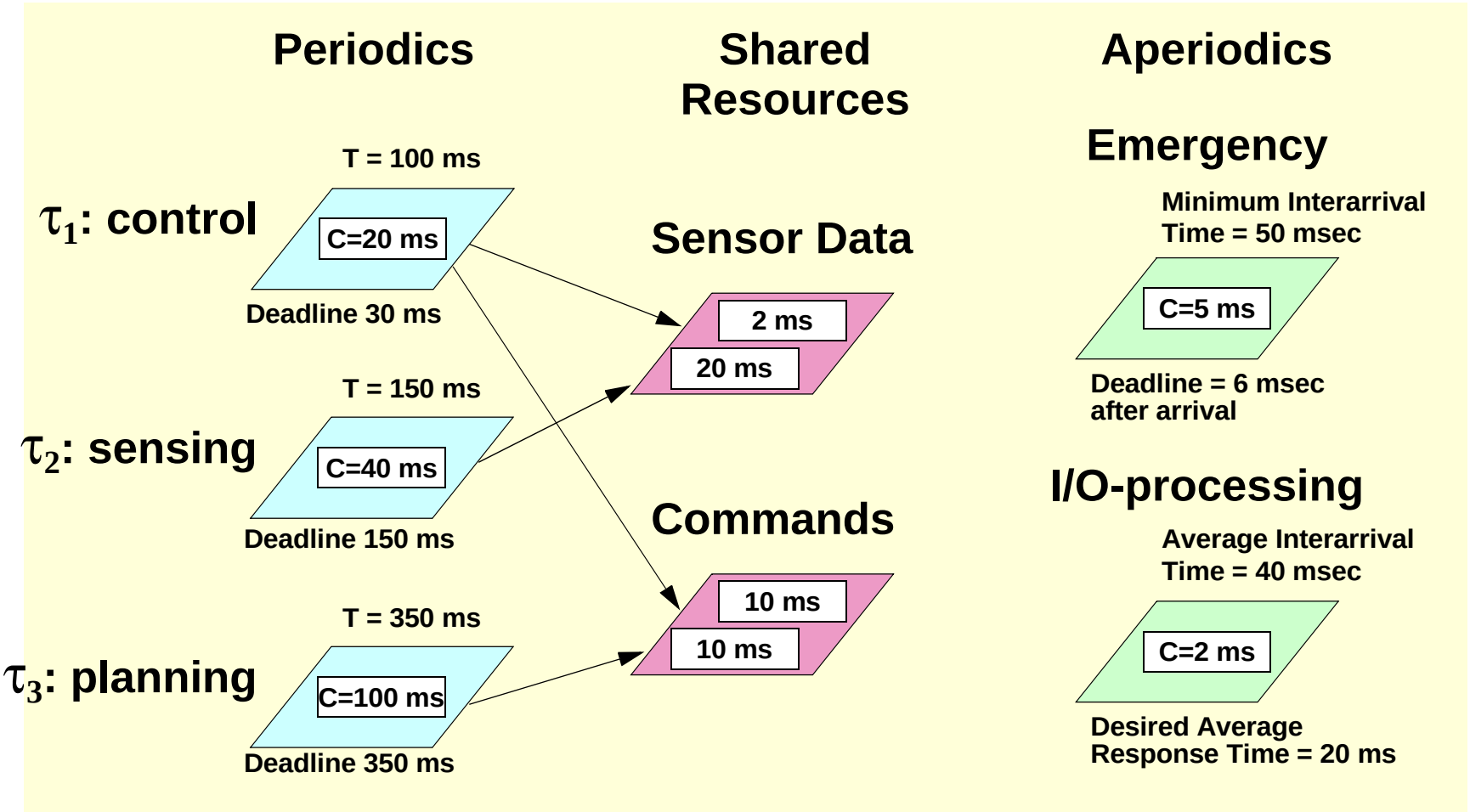
Predictable response times in event-driven real-time systems

UNIVERSIDAD DE CANTABRIA

1. Introduction
2. Basic scheduling
3. *Mutual exclusion*
4. Distributed systems
5. Hierarchical scheduling
6. Protection and flexible scheduling frameworks
7. Operating systems
8. Programming languages
9. Modeling and integration into the design process
10. Conclusions

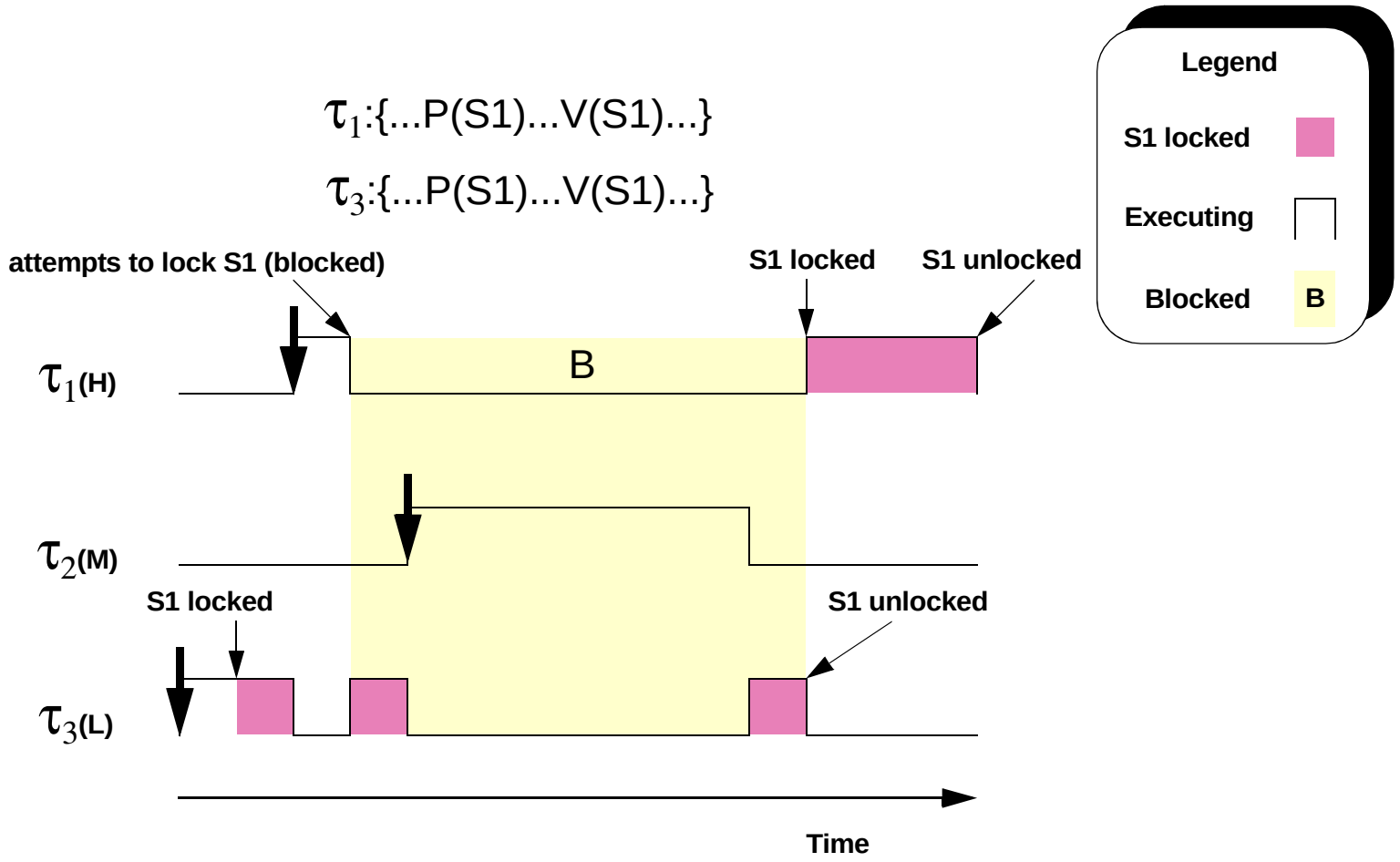
Example: synchronization

UNIVERSIDAD DE CANTABRIA



Unbounded priority inversion

UNIVERSIDAD DE CANTABRIA



Synchronization protocols

UNIVERSIDAD DE CANTABRIA

Protocols that prevent unbounded priority inversion:

- ***No preemption***
  - **equivalent to raising the priority of the critical section to the highest level**
- ***Immediate priority ceiling protocol***
  - **raises the priority of the critical section to a value called the "ceiling"**
  - **lowest blocking time, no extra context switches**
  - **best for static environments**
- ***Priority inheritance***
  - **raises the priority only if somebody is waiting**
  - **best for dynamic environments**

Synchronization in EDF

UNIVERSIDAD DE CANTABRIA

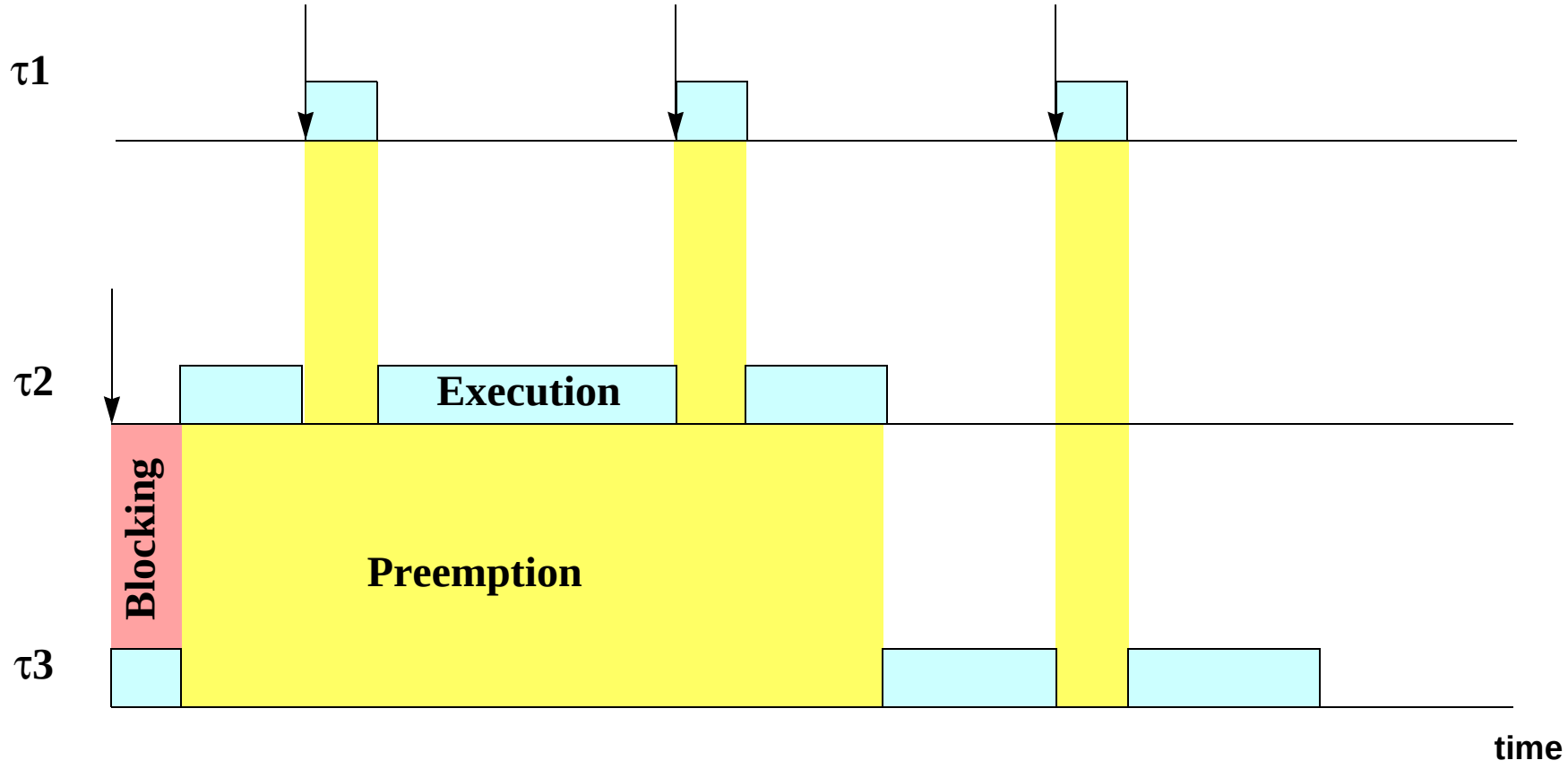
Blocking effects similar to unbounded priority inversion exist

Synchronization protocols

- **(dynamic) *priority inheritance***
- ***Baker's protocol***
  - **applicable to many scheduling policies**
  - **immediate priority ceiling is a special case**
  - **similar properties**

Elements that influence the response time

UNIVERSIDAD DE CANTABRIA



Lessons learned from basic real-time theory

UNIVERSIDAD DE CANTABRIA

In fixed-priority systems, the worst-case response time of a task is a function of:

- ***preemption*: time waiting for higher-priority tasks**
- ***execution*: time to do its own work**
- ***blocking*: time delayed by lower-priority tasks, usually because of mutual exclusion synchronization**

Deadlines are missed when utilization is over 100% (seems obvious, but happens in reality):

- **Solutions:**
  - **change the computation requirements**
  - **change the periods or interarrival times**
  - **buy a faster CPU**

Lessons learned from basic real-time theory (cont'd)

UNIVERSIDAD DE CANTABRIA

Deadlines are missed because there is too much preemption:

- **A task or a task's portion runs at a higher priority than it should**
  - **this happens typically with interrupt service routines**
- **There are not enough priorities**
- **Solutions:**
  - **preemption can be minimized by choosing an optimum priority assignment**
  - **sometimes a task or an ISR must be split into parts whose priorities can be independently assigned**
  - **buy a system with user-defined priorities for I/O drivers**
  - **buy a system with at least 32 priority levels**

Lessons learned from basic real-time theory (cont'd)

UNIVERSIDAD DE CANTABRIA

Deadlines are missed because there is too much blocking:

- **Normal semaphores suffer from unbounded priority inversion**
  - **this causes horrendous worst-case blocking times**
- **Solutions**
  - **disable preemption in the critical sections (may still have too much blocking)**
  - **priority inheritance protocol**
  - **priority ceiling or Baker's protocol**
  - **split long critical sections**

Predictable response times in event-driven real-time systems

UNIVERSIDAD DE CANTABRIA

1. Introduction
2. Basic scheduling
3. Mutual exclusion
4. *Distributed systems*
5. Hierarchical scheduling
6. Protection and flexible scheduling frameworks
7. Operating systems
8. Programming languages
9. Modeling and integration into the design process
10. Conclusions

Real-time networks

UNIVERSIDAD DE CANTABRIA

Very few networks guarantee real-time response

- **many protocols allow collisions**
- **no priorities or deadlines in the protocols**

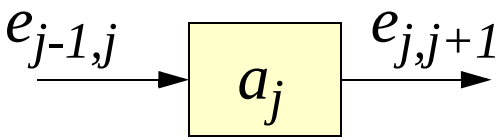
Some solutions

- **CAN bus and other field busses**
- **Priority-based token passing**
- **Time-division multiplexed access**
- **Point to point networks**

Distributed system model

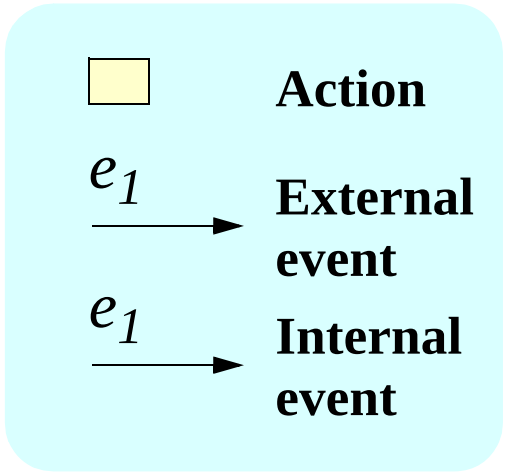
UNIVERSIDAD DE CANTABRIA

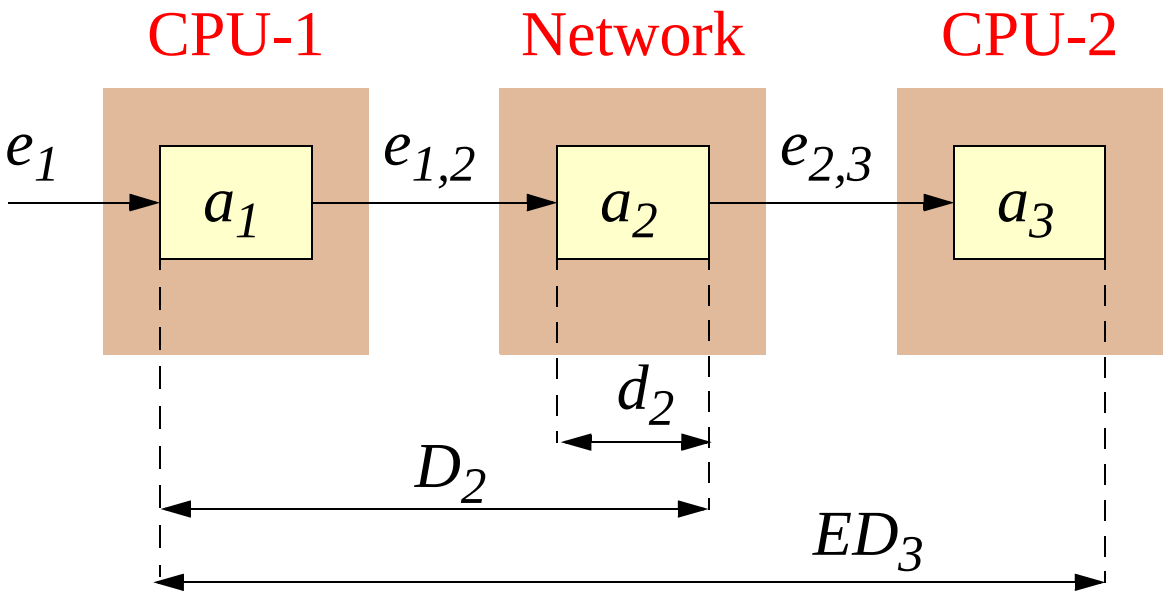
Linear Action:



$$T_{j-1,j} = T_j = T_{j,j+1}$$

Linear Response to an Event:

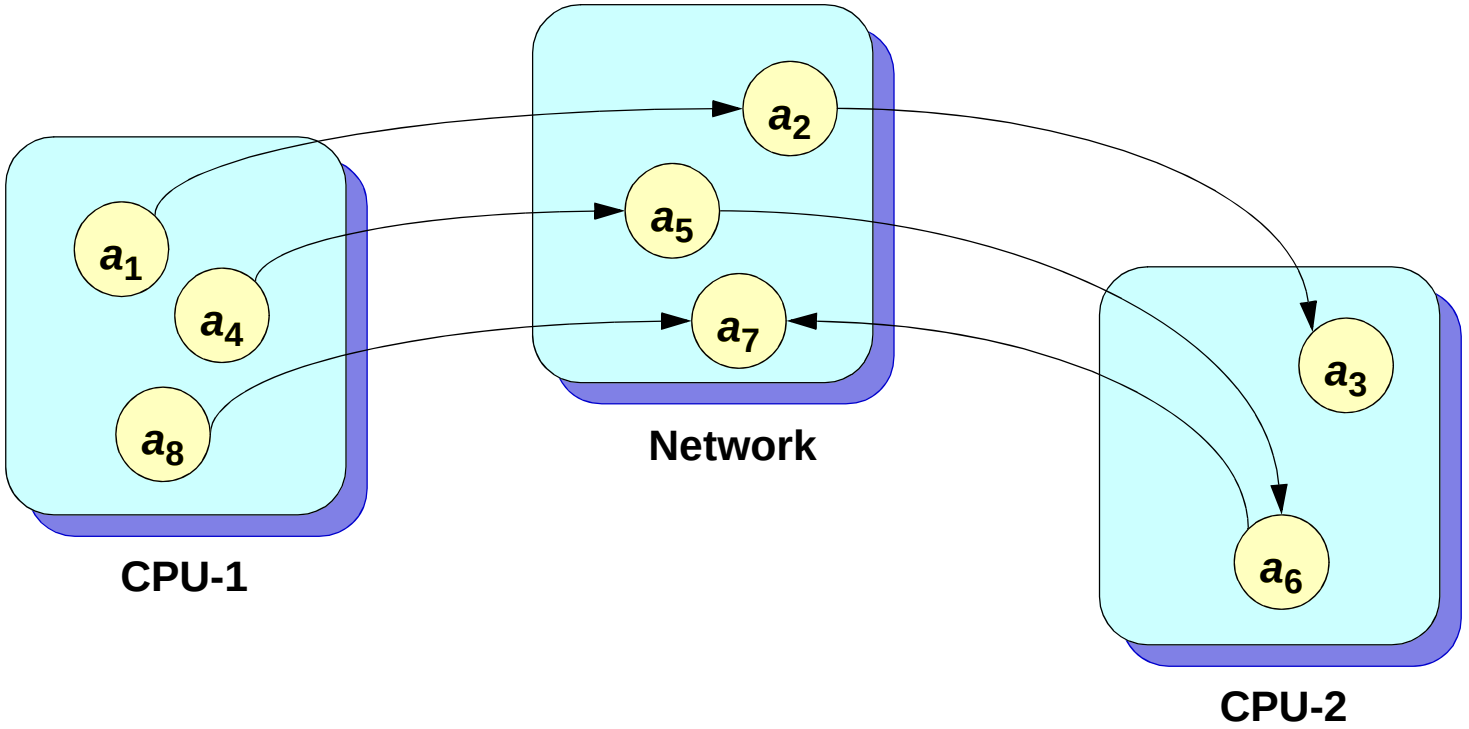




Release jitter in distributed systems

UNIVERSIDAD DE CANTABRIA

In the example below, actions a_2,a_3 and a_5,a_6,a_7,a_8 have jitter, even if a_1 and a_4 are purely periodic:



The problem

UNIVERSIDAD DE CANTABRIA

Release jitter in one resource depends on the response times in other resources

Response times depend on release jitter

Holistic analysis technique

UNIVERSIDAD DE CANTABRIA

Developed at the University of York

Each resource is analyzed separately (CPU's and communication networks):

- **the analysis is repeated *iteratively*, until a stable solution is achieved**
- **the method converges because of the *monotonic* relation between jitters and response times**

Analysis provides guarantees on schedulability, but is very pessimistic

- **independent critical instants may not be possible in practice**

Offset-based techniques

UNIVERSIDAD DE CANTABRIA

They reduce the pessimism of the worst-case analysis in multiprocessor and distributed systems:

- by considering offsets in the analysis

Exact analysis is intractable:

- The task that generates the worst-case contribution of a given transaction is unknown
- The analysis has to check all possible combinations of tasks

Upper bound approximation

- This technique is pessimistic, but polynomial
- In 93% of the tested cases, the response times were exact

Priority assignment techniques

UNIVERSIDAD DE CANTABRIA

No known optimum priority assignment for distributed systems

Simulated Annealing

- **Standard optimization technique**

HOPA

- **Heuristic algorithm based on successively applying the analysis**
- **Much faster than simulated annealing**
- **Usually finds better solutions**

None of them guarantees finding the solution

Summary

UNIVERSIDAD DE CANTABRIA

Kind of Analysis	Deadlines	Number of processors	Fixed Priorities	EDF
Utilization test	D=T	1	☑	☑
Response Time Analysis	Arbitrary	1	☑	☑
Holistic Analysis	Arbitrary	Many	☑	☑
Offset-Based Analysis	Arbitrary	Many	☑	☑

Distributed model also applicable to:

- **signal & wait synchronization**
- **activities that suspend themselves (i.e., delays, I/O, ...)**

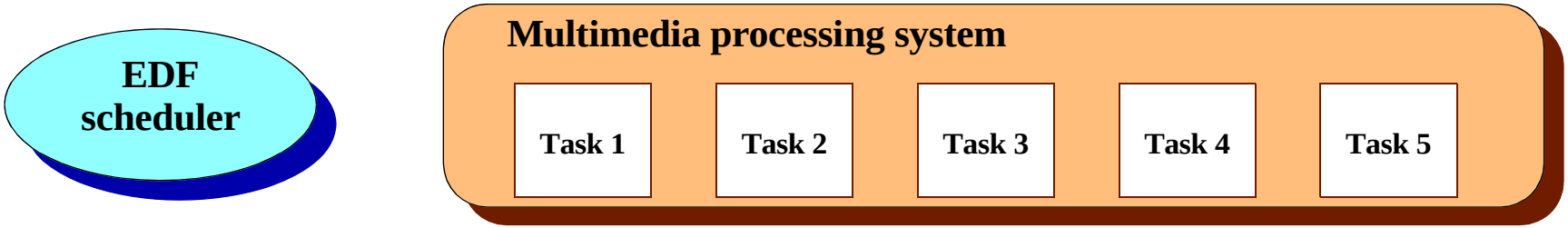
Predictable response times in event-driven real-time systems

UNIVERSIDAD DE CANTABRIA

1. Introduction
2. Basic scheduling
3. Mutual exclusion
4. Distributed systems
5. *Hierarchical scheduling*
6. Protection and flexible scheduling frameworks
7. Operating systems
8. Programming languages
9. Modeling and integration into the design process
10. Conclusions

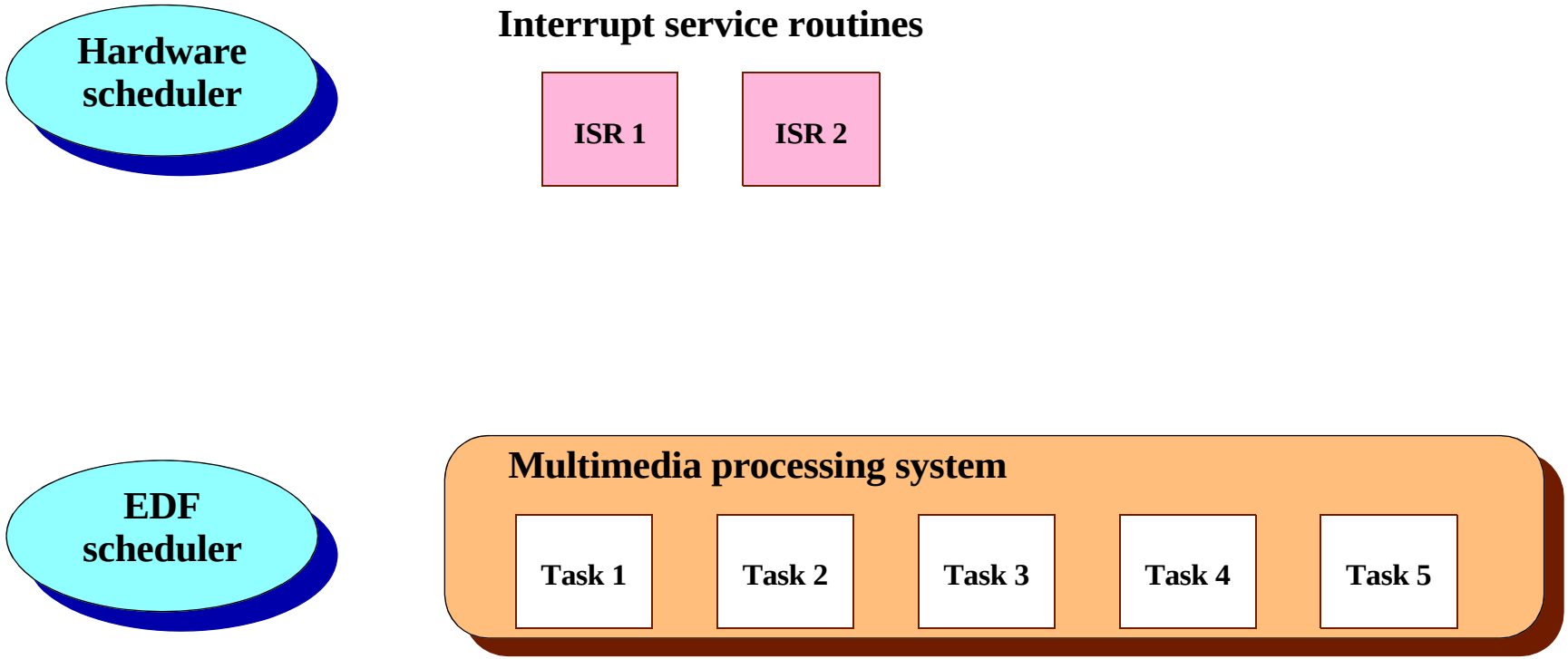
Mixing EDF and fixed priorities

UNIVERSIDAD DE CANTABRIA



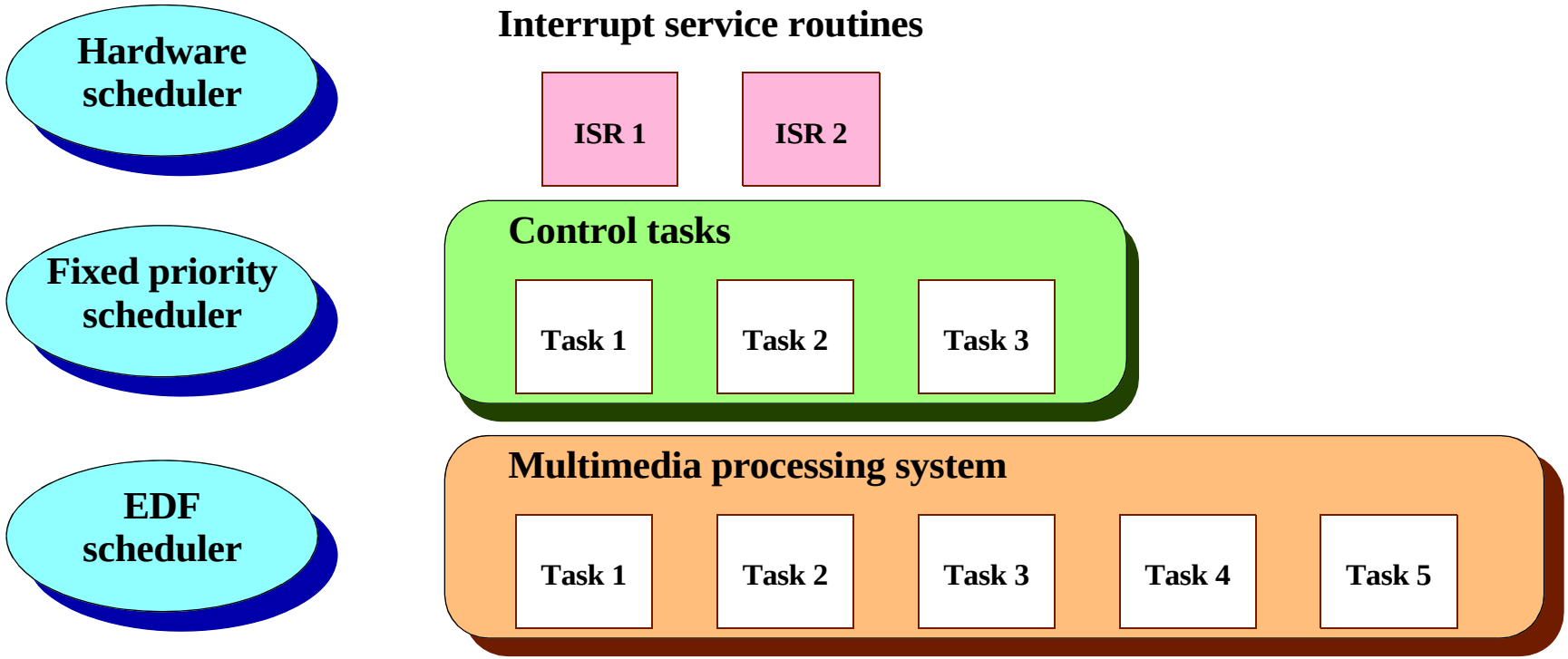
Mixing EDF and fixed priorities

UNIVERSIDAD DE CANTABRIA



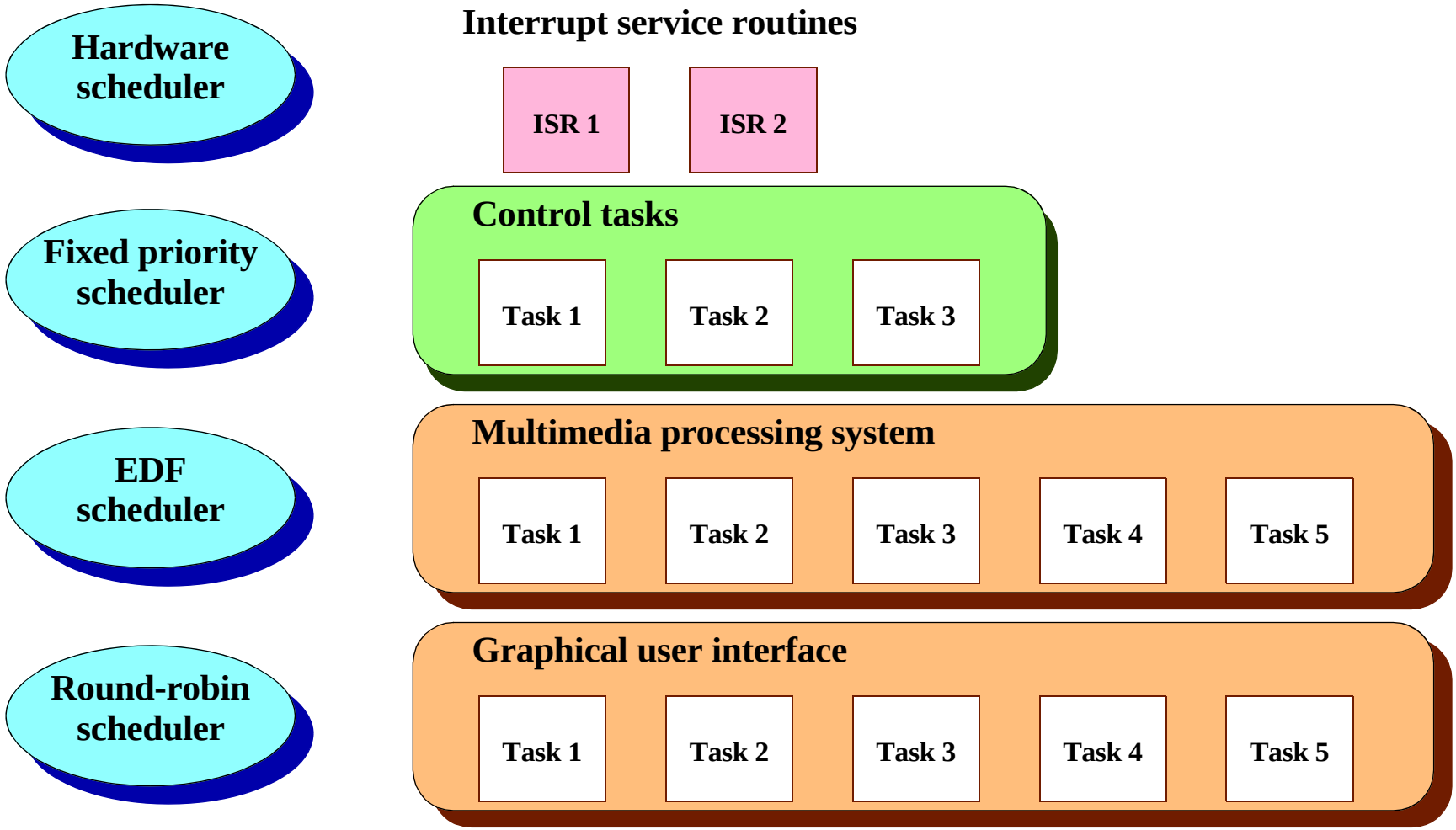
Mixing EDF and fixed priorities

UNIVERSIDAD DE CANTABRIA



Mixing EDF and fixed priorities

UNIVERSIDAD DE CANTABRIA



RTA for EDF-within-priorities

UNIVERSIDAD DE CANTABRIA

Techniques have developed to obtain a worst-case response time analysis in systems with hierarchical schedulers

- **underlying fixed priority scheduler**
- **EDF schedulers running tasks at a given priority level**

We call this approach "*EDF within fixed priorities*"

Predictable response times in event-driven real-time systems

UNIVERSIDAD DE CANTABRIA

1. Introduction
2. Basic scheduling
3. Mutual exclusion
4. Distributed systems
5. Hierarchical scheduling
6. *Protection and flexible scheduling frameworks*
7. Operating systems
8. Programming languages
9. Modeling and integration into the design process
10. Conclusions

Motivation for time protection

UNIVERSIDAD DE CANTABRIA

Response-time analysis is heavily based on knowing the worst-case execution times (WCET) of each individual task

- **tools based on precise processor models**
- **tools based on measurement + probabilistic methods**

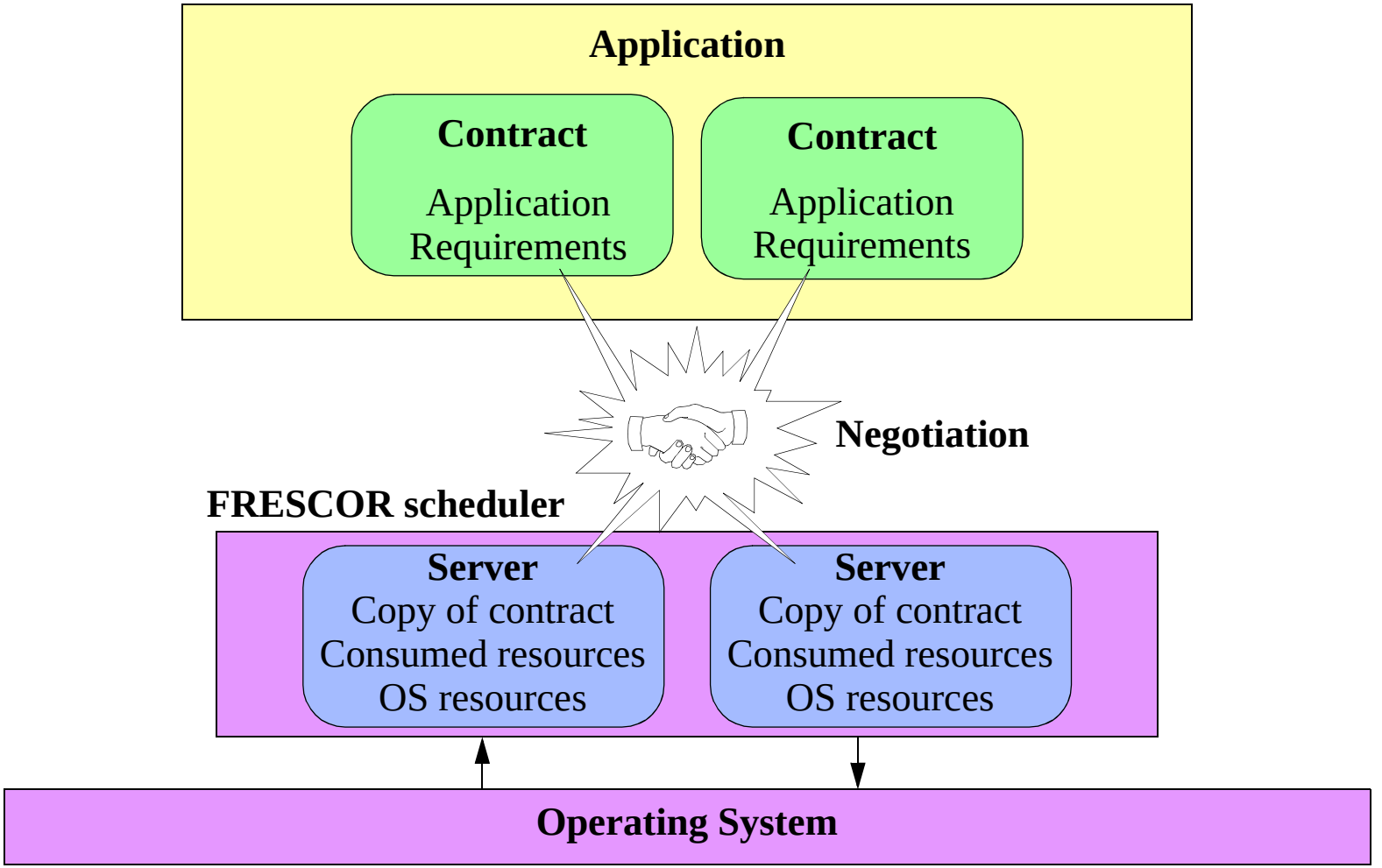
Precise WCET is very different from average case

Time protection helps by detecting execution-time overruns

Independently developed components sharing resources require time protection

Contract-based scheduling

UNIVERSIDAD DE CANTABRIA



The FRESCOR flexible scheduling framework project

UNIVERSIDAD DE CANTABRIA

A contract specifies:

- **the minimum resource usage requirements**
- **how to make use of spare resources**

FRESCOR Increases the level of abstraction for RT services

- **automatic admission test**
- **independence of scheduling algorithm**
- **dynamic reclamation of unused resources**
- **coexistence of hard real-time, quality of service**

Temporal encapsulation of subsystems

Closes the gap between theory and practice

Predictable response times in event-driven real-time systems

UNIVERSIDAD DE CANTABRIA

1. Introduction
2. Basic scheduling
3. Mutual exclusion
4. Distributed systems
5. Hierarchical scheduling
6. Protection and flexible scheduling frameworks
7. *Operating systems*
8. Programming languages
9. Modeling and integration into the design process
10. Conclusions

Real-time operating systems

UNIVERSIDAD DE CANTABRIA

In the past, many real-time systems did not need an operating system

Today, many applications require operating system services such as:

- **concurrent programming, communication networks, file system, etc.**

The timing behavior of a program depends strongly on the operating system's behavior

POSIX definition of real-time in operating systems:
"The ability of the operating system to provide a required level of service in a *bounded response time*."

What is POSIX?

UNIVERSIDAD DE CANTABRIA

Portable Operating System Interface

Based on UNIX operating systems

The goal is portability of

- **applications (at the source code level)**
- **programmers**

Sponsored by the IEEE and The Open Group

POSIX real-time profiles (POSIX.13)

UNIVERSIDAD DE CANTABRIA

The POSIX Standard:

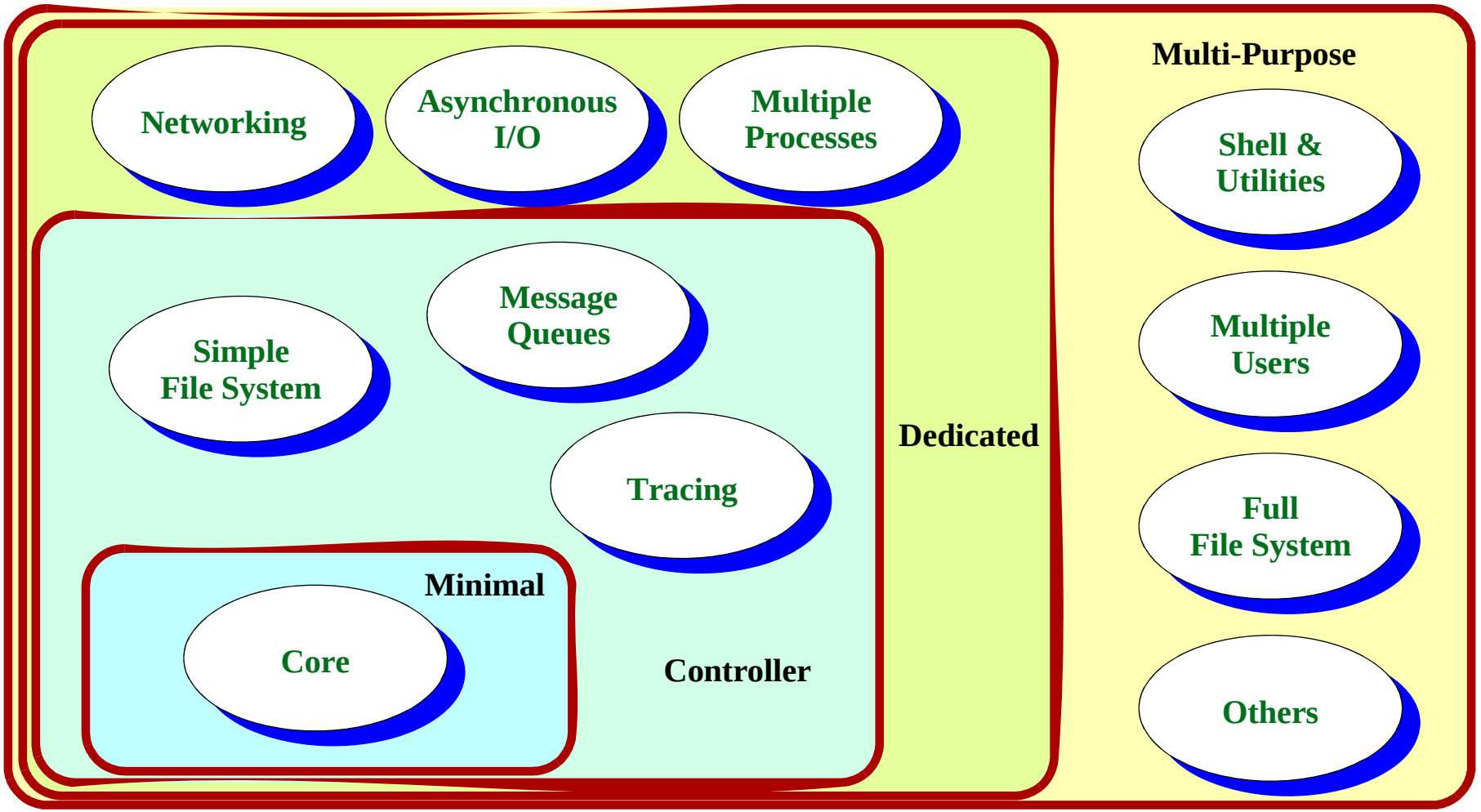
- **Allows writing portable real-time applications**
- **Very large: inappropriate for embedded real-time systems**

POSIX.13:

- **Defines four real-time system subsets (profiles)**
- **C and Ada language options**

Summary of RT Profiles

UNIVERSIDAD DE CANTABRIA



Predictable response times in event-driven real-time systems

UNIVERSIDAD DE CANTABRIA

1. Introduction
2. Basic scheduling
3. Mutual exclusion
4. Distributed systems
5. Hierarchical scheduling
6. Protection and flexible scheduling frameworks
7. Operating systems
8. *Programming languages*
9. Modeling and integration into the design process
10. Conclusions

Programming languages for real-time systems

UNIVERSIDAD DE CANTABRIA

Some languages do not provide support for concurrency or real-time

- **C, C++**
- **support achieved through OS services (POSIX)**

Other languages provide support in the language itself

- **Java (concurrency) and RTSJ (real-time)**
  - **API based**
- **Ada: concurrency and real-time**
  - **integrated into the core language**

Programming languages for real-time systems

UNIVERSIDAD DE CANTABRIA

Language support helps in

- **increased reliability**
- **more expressive power**
- **reduced development and maintenance costs**

Predictable response times in event-driven real-time systems

UNIVERSIDAD DE CANTABRIA

1. Introduction
2. Basic scheduling
3. Mutual exclusion
4. Distributed systems
5. Hierarchical scheduling
6. Protection and flexible scheduling frameworks
7. Operating systems
8. Programming languages
9. *Modeling and integration into the design process*
10. Conclusions

Motivation

UNIVERSIDAD DE CANTABRIA

The latest schedulability analysis techniques are difficult to apply by hand

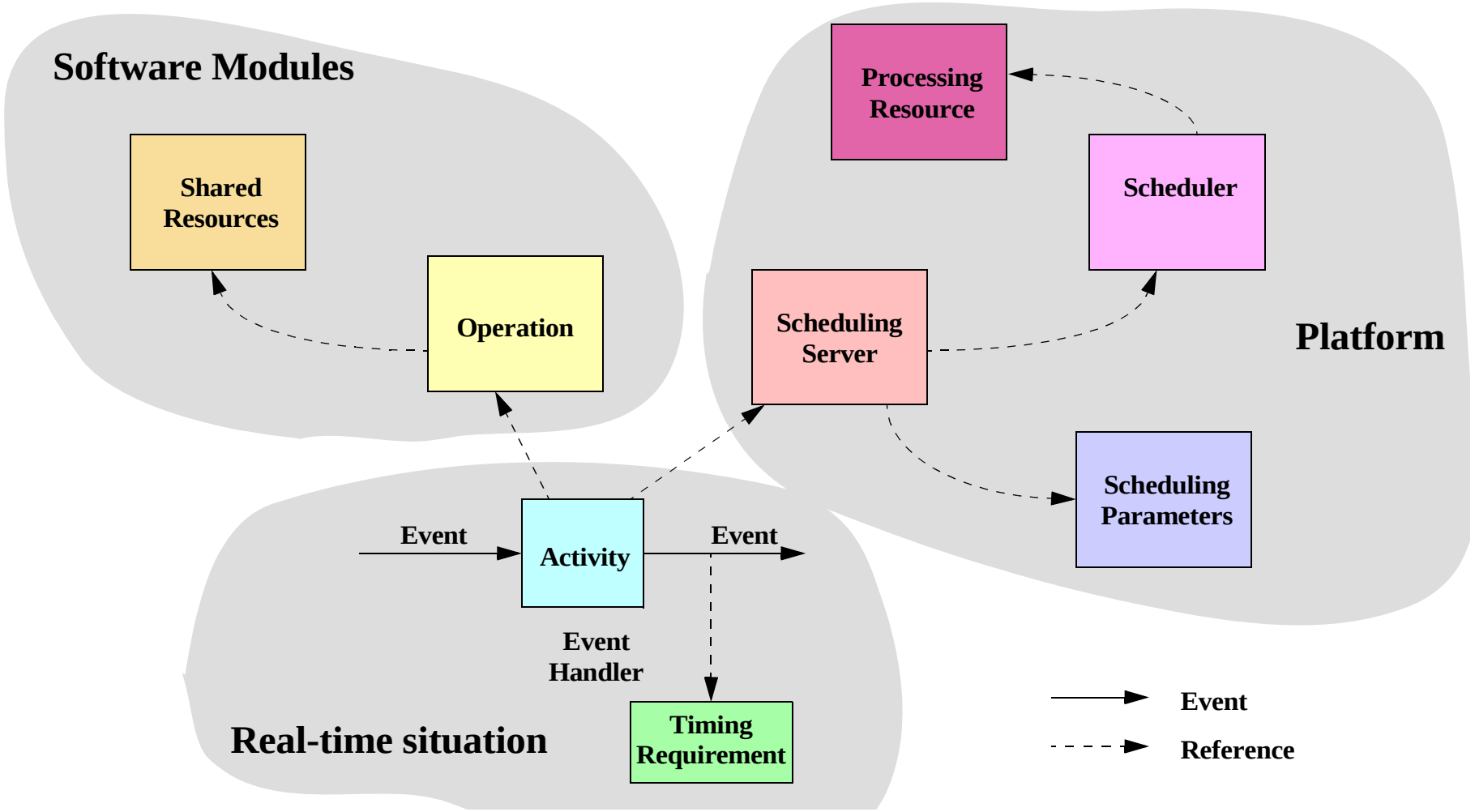
Need to integrate all the techniques in a single tool suite

- **schedulability analysis tools**
- **priority assignment**
- **sensitivity analysis**

A model of the real-time behavior is necessary

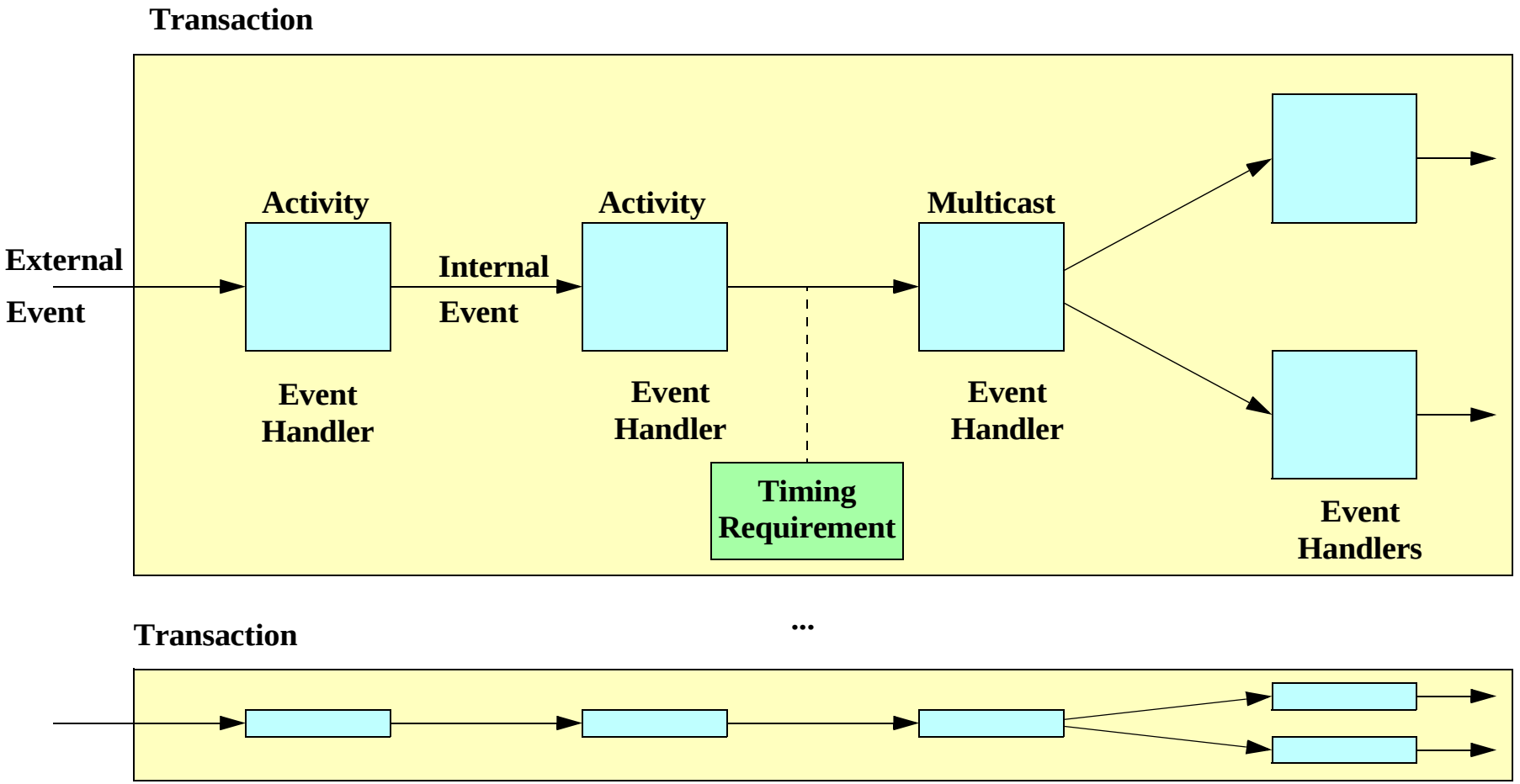
Real-time system model

UNIVERSIDAD DE CANTABRIA



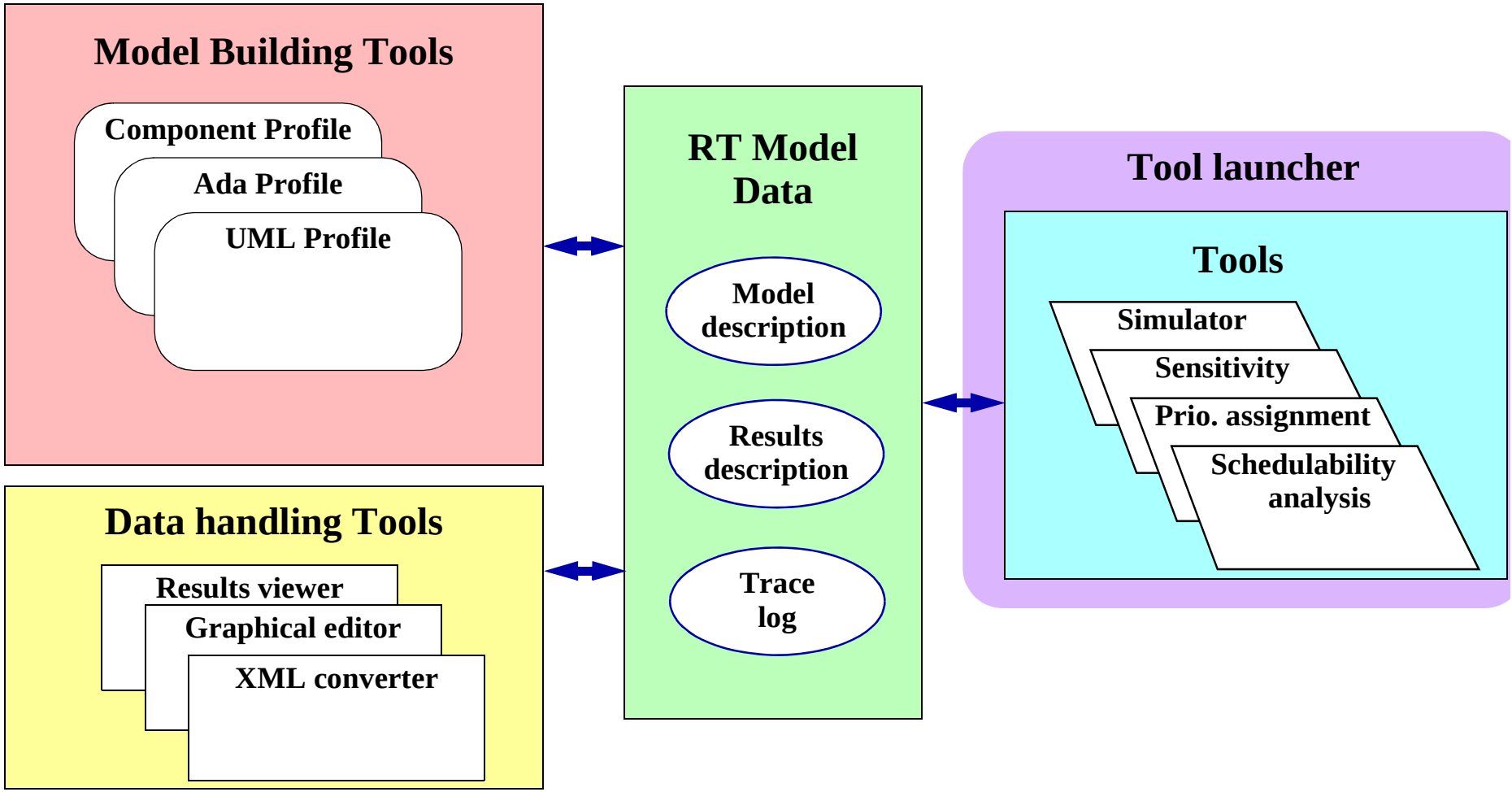
Real-time situation

UNIVERSIDAD DE CANTABRIA



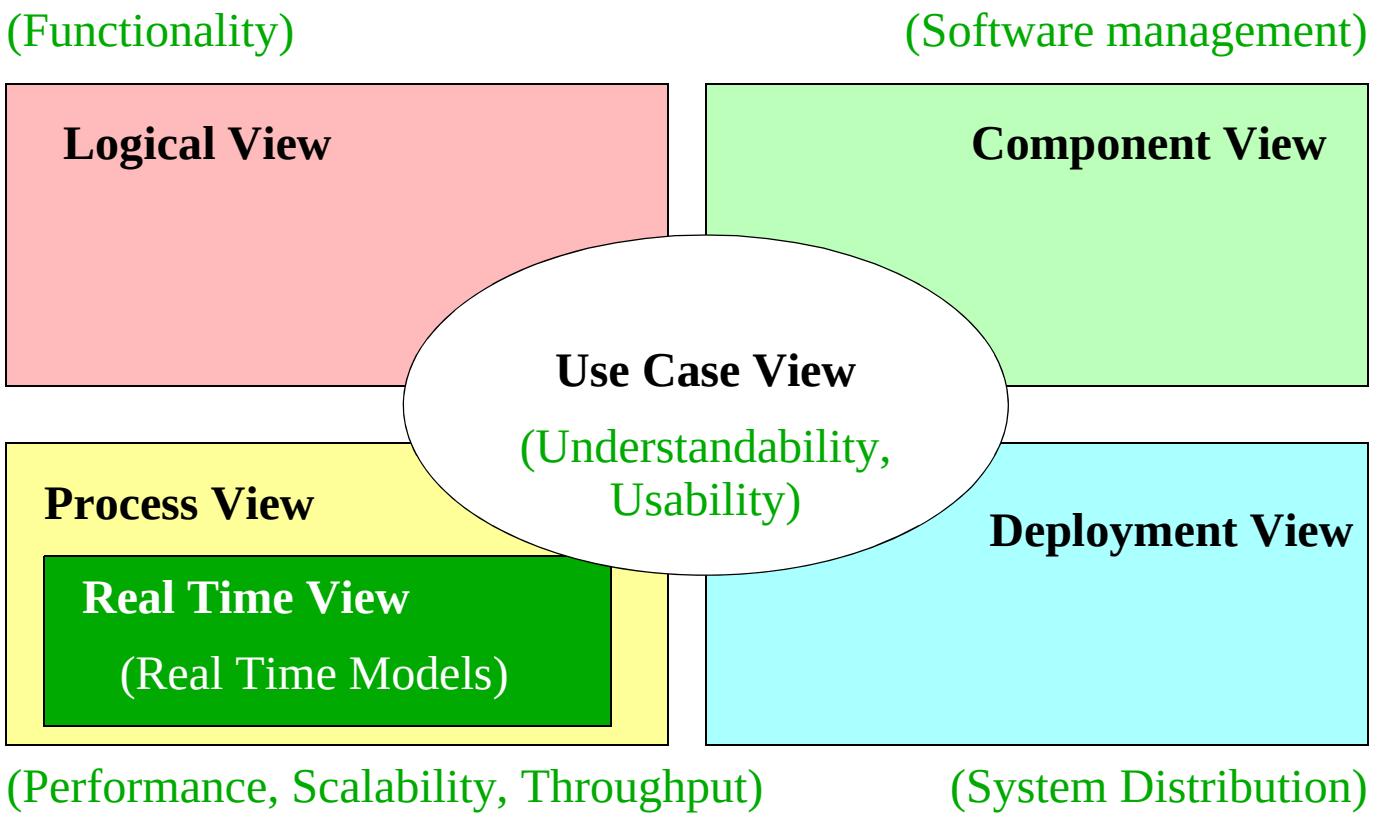
MAST environment

UNIVERSIDAD DE CANTABRIA



The MAST_RT_View in the "4+1 View"

UNIVERSIDAD DE CANTABRIA



Predictable response times in event-driven real-time systems

UNIVERSIDAD DE CANTABRIA

1. Introduction
2. Basic scheduling
3. Mutual exclusion
4. Distributed systems
5. Hierarchical scheduling
6. Protection and flexible scheduling frameworks
7. Operating systems
8. Programming languages
9. Modeling and integration into the design process
10. *Conclusions*

Conclusions

UNIVERSIDAD DE CANTABRIA

Priority-based run-time scheduling supports

- **predictability of response time, via static analysis**
- **flexibility and separation of concerns**
- **dynamic and fixed priorities can be combined together**

Real-time theory is now capable of analyzing complex real-time systems

Tools are available to

- **automatically analyze a system**
- **provide sensitivity analysis**
- **design space exploration**

Conclusions (cont'd)

UNIVERSIDAD DE CANTABRIA

Flexible scheduling frameworks

- **integrate hard real-time with quality of service requirements**
- **increased use of system resources**
- **higher level of abstraction**

Future goals

UNIVERSIDAD DE CANTABRIA

Integration of schedulability analysis with design tools and methods

- **component-based approaches**

Improved methods for WCET estimation

New networks and protocols that use real-time scheduling

Integration of real-time response into general-purpose operating systems

- **kernel support, including I/O drivers**
- **flexible scheduling frameworks**