Real-Time Systems

Hermann Härtig

Introduction

Marcus Völp

WS 2012/13

TECHNISCHE UNIVERSITÄT DRESDEN

Organisation Issues

Web-Page http://os.inf.tu-dresden.de/Studium/RTS/

Subscribe to the mailing list !!!

Time 3 SWS: ca 2 lectures + ca 1 exercises

Thursday, 5th/6th DS

(whether or not 6th DS is needed is usually announced the day before on the Web-Site)

Real-Time Systems

Definition (strict)

Systems, whose correctness depends

(not only) on the correct logical results of computations

(but also) on meeting **all** *deadlines*.

Deadlines are dictated by the environment of the system.

Results, deadlines must be specified.

Real-Time Systems

Definition (weaker)

Systems, whose quality depend

- (not only) on the logical results of computations
- (but also) on the *time* these results are *produced*.

Requested timing characteristics originate from the environment of the system.

Some Examples for Weaker Notions

- Some deadlines are more important than others

  (Later: imprecise computations, mixed criticality)

- Occasional misses of deadlines are ok. (e.g., 3 in 10)

- The value of a result depends on the time it becomes available:
  - An imperfect result early may be better than a perfect result (too) late.
  - The more results can be obtained before a given deadline the better.
  - Explicit mapping of time to value

Real-Time Systems (weaker notions)

Specification needed for :

Results, deadlines PLUS

- "Importance" of certain deadlines OR
- How many deadlines per time period may be missed OR
- Mapping of time to values of results OR …

More Complex ?

A saying by Doug Jensen (?):

Hard real-time systems are hard to build,

soft real-time systems even harder.

We largely ignore: more terminology

Hard vs. Firm vs. Soft

- hard real-time systems
  - deadlines are strict,
    missing has fatal consequences for the controlled object or humans, must work under peak load
- firm real-time systems
  - deadlines are strict
- soft real-time systems
  - deadlines should be met,
    value of results decreases with time,
    graceful degradation under peak load is acceptable

Safety-Critical Systems

Failures (timing or functional) put humans, … at risk

→ Many Safety-Critical Systems are Real-Time

Embedded Systems (also see → Lecture Prof. Hochberger)

Computer systems as integral part of a larger system ...

Common characteristics:

- large numbers (mass market)
- static structure (changing)
- often: minimize mechanics
- software hard to change after deployment (ROM)

cars, consumer electronics, printer, cell phone, ...

Well over 90% of microprocessor production in terms of number of units (not value)

→ Many Embedded Systems are Real-Time

Key Abstraction: Recurrent Processes

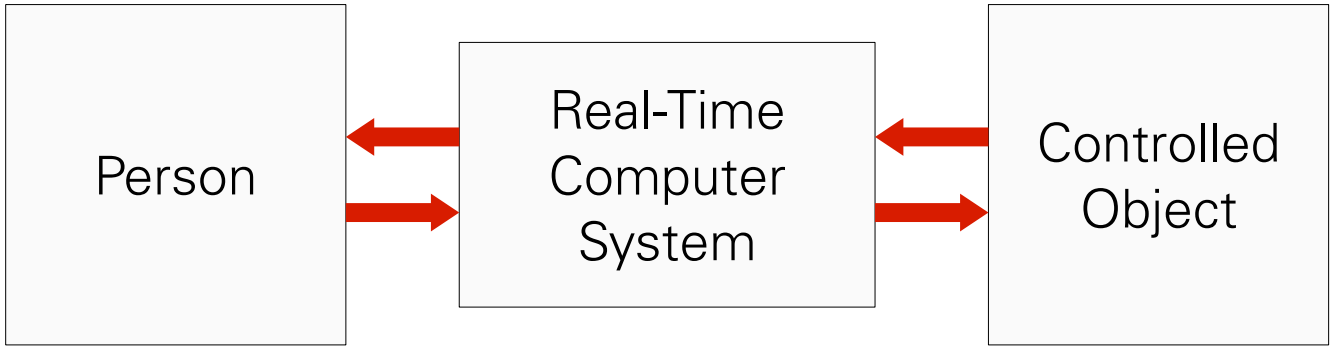
```
forever {
        Begin period (     )
            compute
        End period }
```

Next lecture on models:

- Two variants: periodic / sporadic
- Parameters (inter-arrival, deadline, execution times, ...)

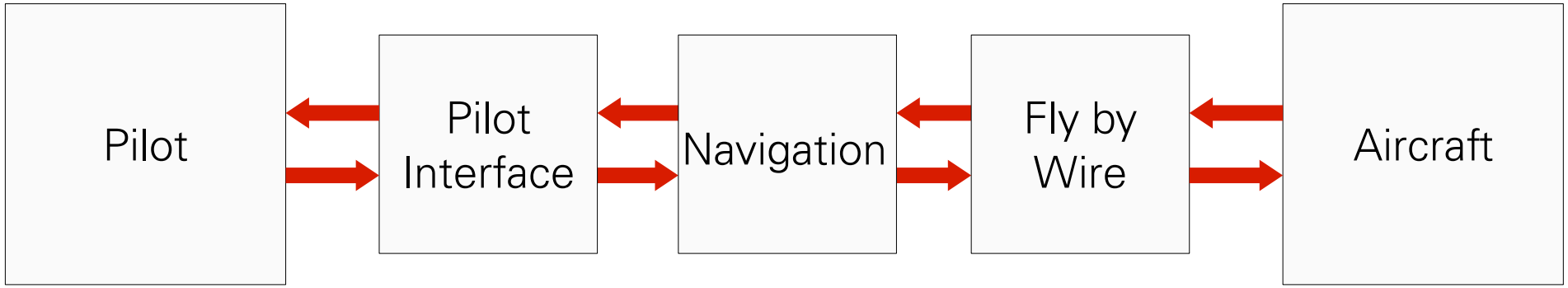
where do the times come from ...

Principle Interfaces



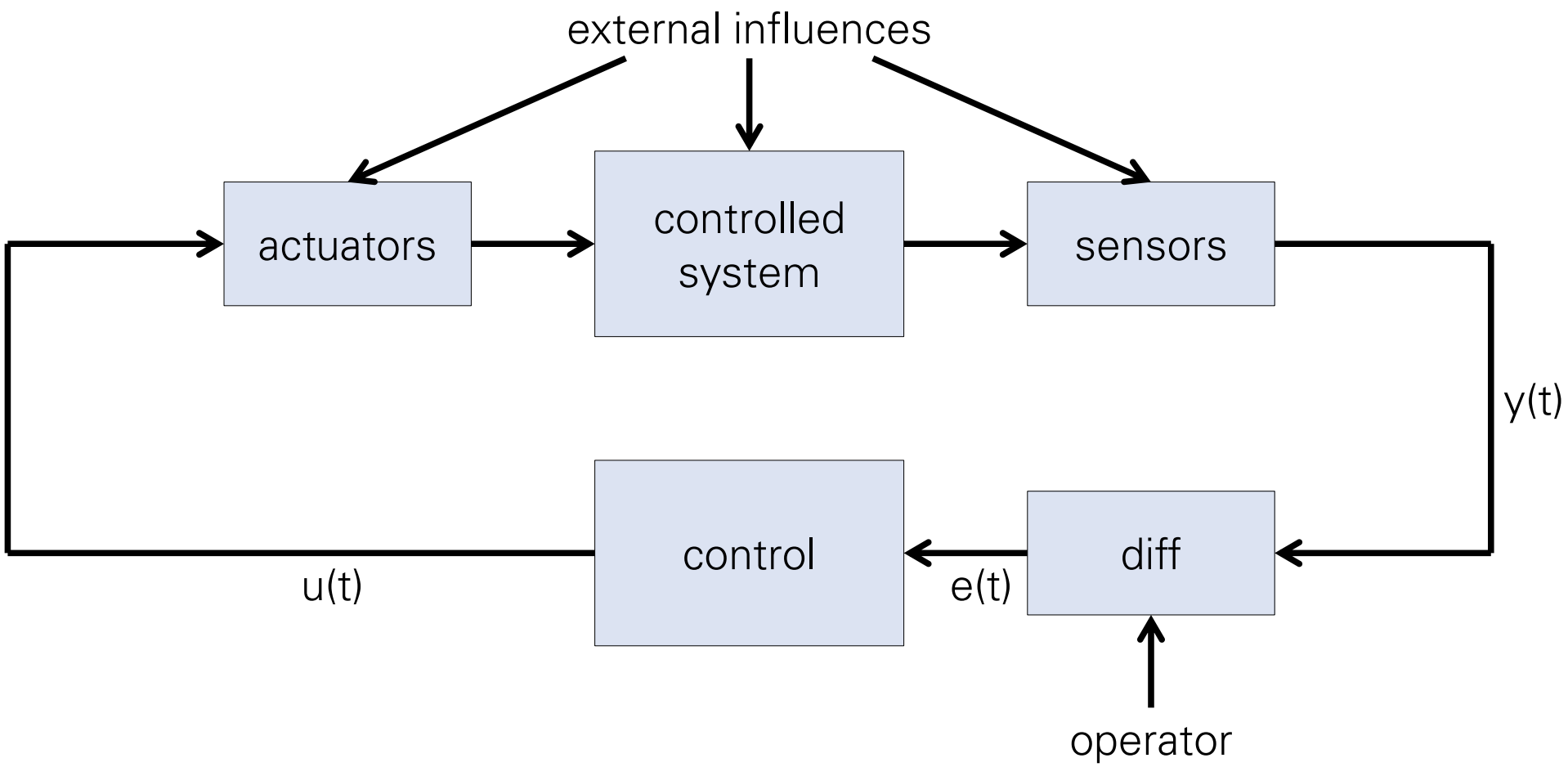
times induced thru both interfaces

Layers of Control

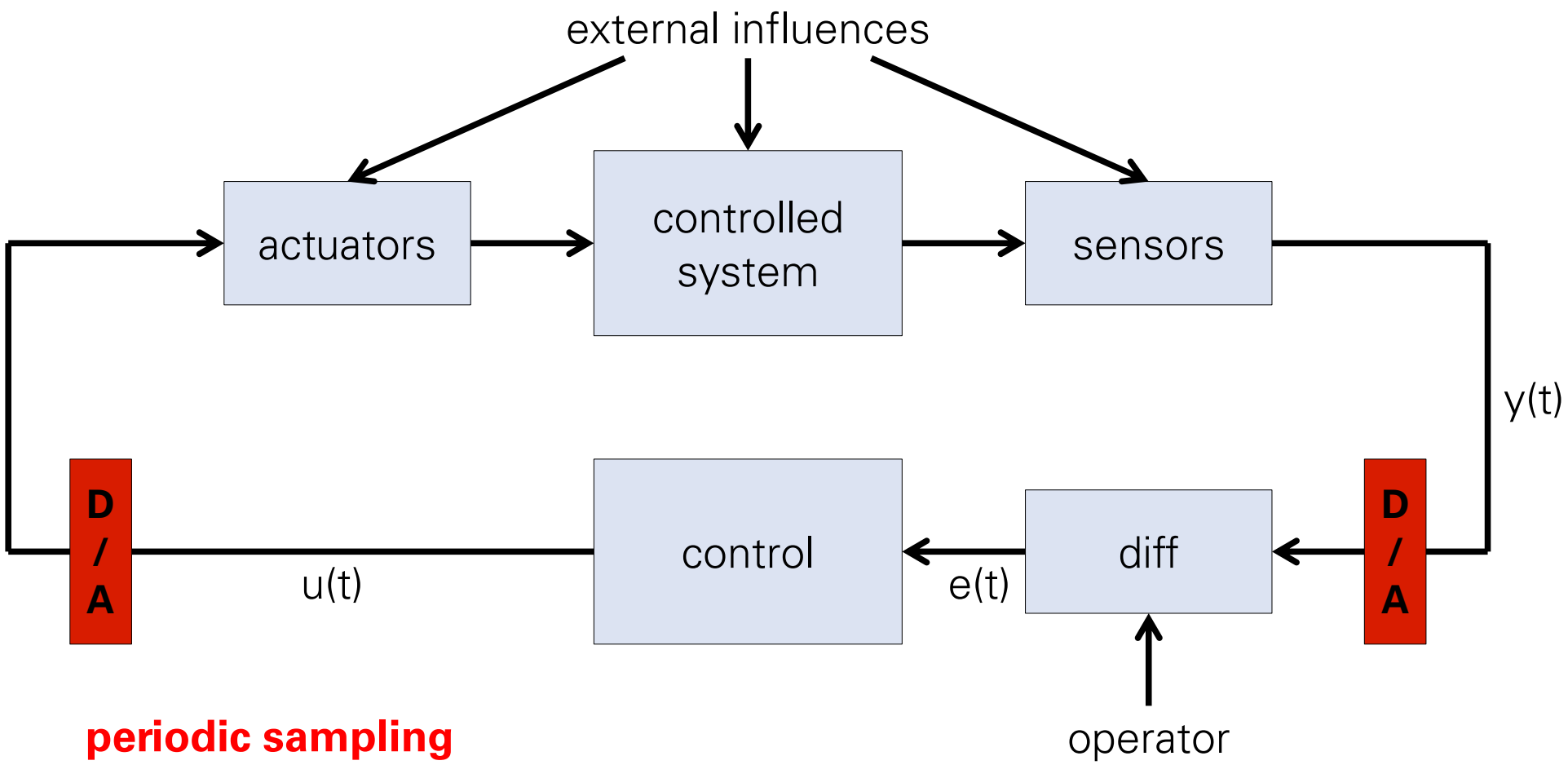


Multiples stages may induce different times

Simple Control System



Simple Digital Control System



“PID” controller

Continuous formula: $u = k_p e + k_i \int_{\tau=0}^{t} e(\tau)d\tau + T_d \frac{de}{dt}$

Approximation by periodic sampling (rate T)

Integral via Simpson's Rule: $\frac{T}{3} * (e_{k-2} + 4e_{k-1} + e_k)$

Differential: $\frac{e_k - e_{k-1}}{T}$

Then: $u_k := u_{k-2} + a * e_k + b * e_{k-1} + c * e_{k-2}$

With $a = k_p + \frac{k_i T}{3} + \frac{T_d}{T}$ $b = \frac{4k_i T}{3} - \frac{T_d}{T}$ $c = \frac{k_i T}{3}$

Digital Controllers

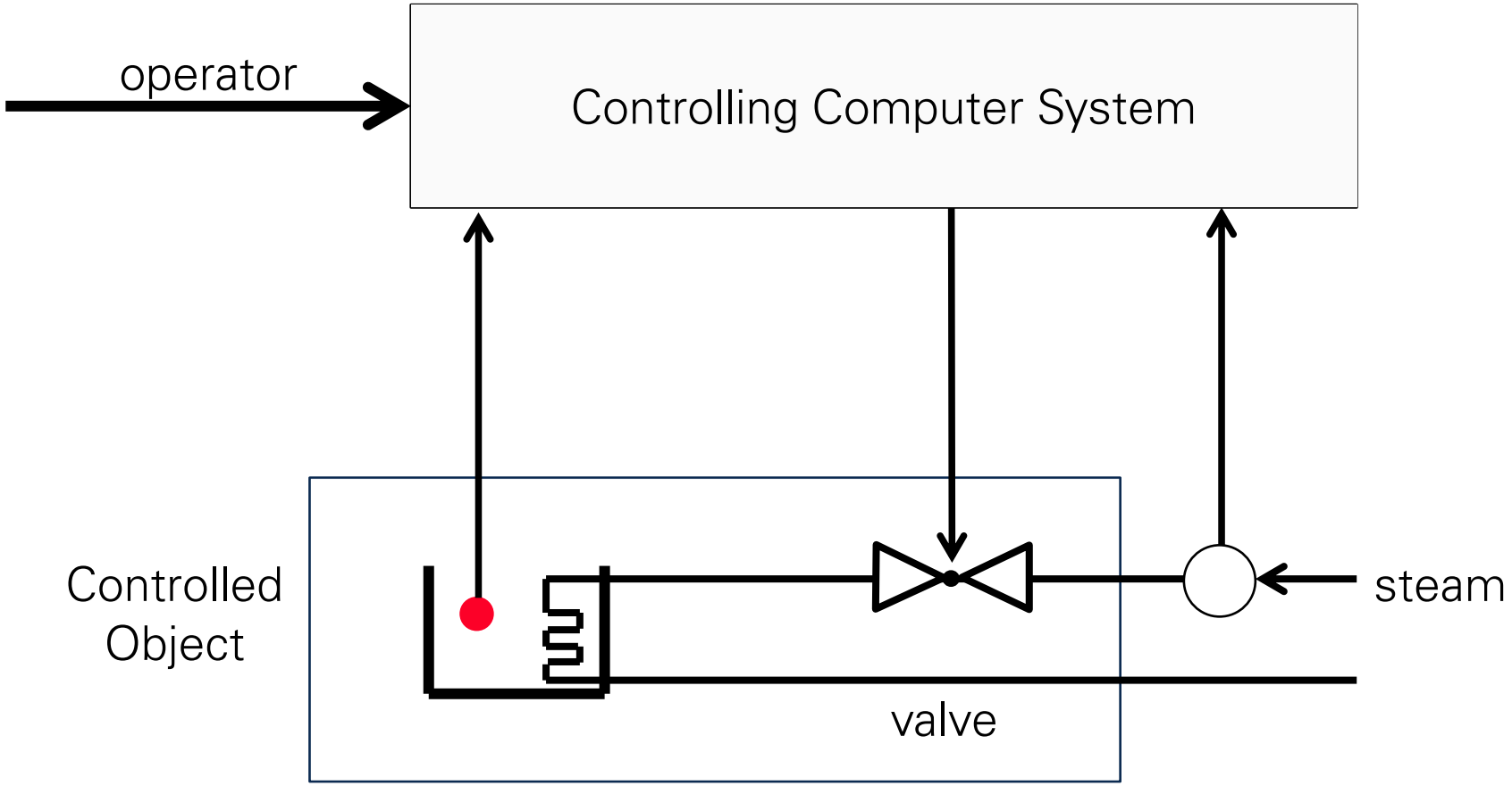
```
at every xx time units do
     read r,y
     u_k :=u_k-2+a*e_k +b*e_k-1+c*e_k-2
     write u
  done
```

sample period xx depends on:

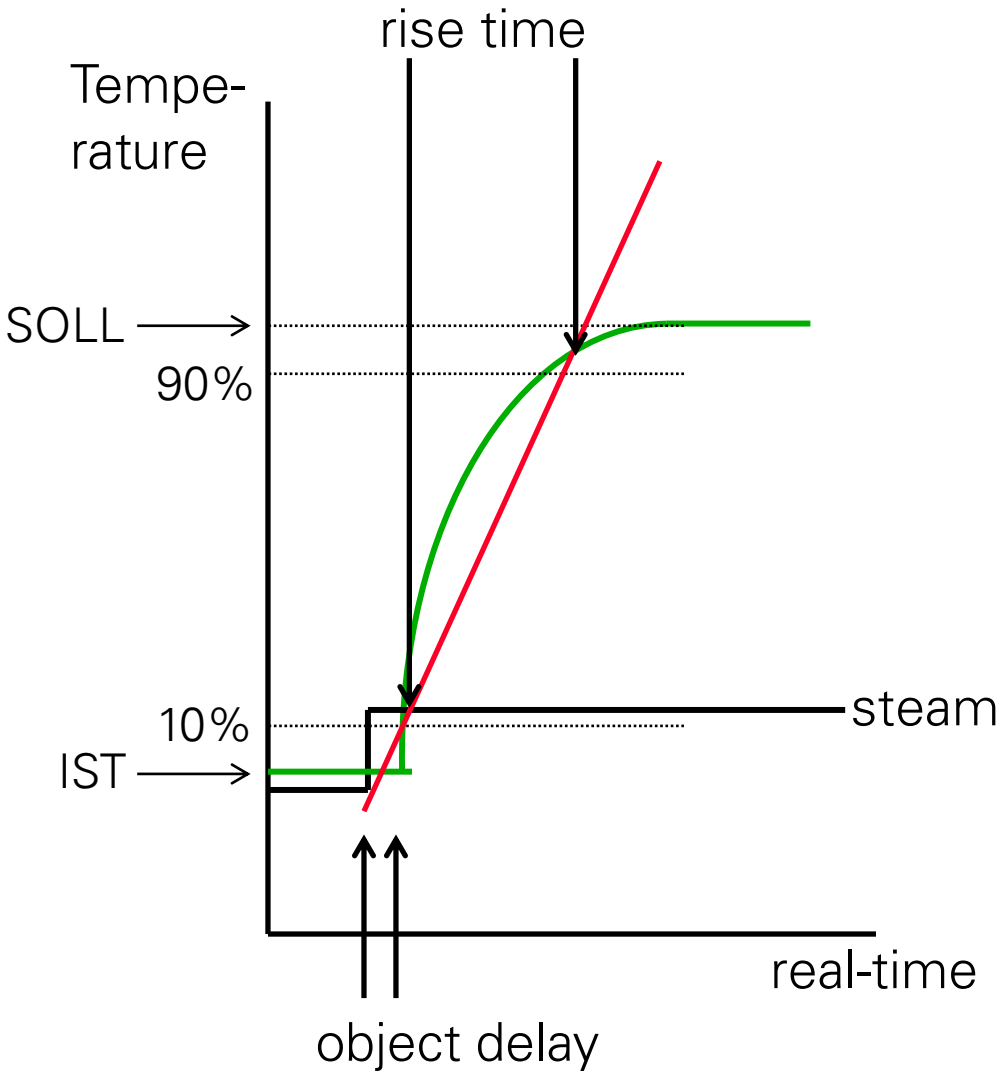
reactivity of person (<100 ms)

reactivity of controlled object (?)

Simple Control Loop (Kopetz)



Times



- rise time
  (10% or other small neighborhood)
- object delay
  (inertia of control process)
- sampling period
  rule of thumb < 1/10 to 1/20 rise time
- computation delay
  computation delay jitter
  (< sample period)
- deadtime = object delay + comp delay
- shorter sampling periods result in:
  - smoother operation
  - less oscillation
  - more resources used

Complications of simple model: state

complete state of

controlled object

is not

represented in

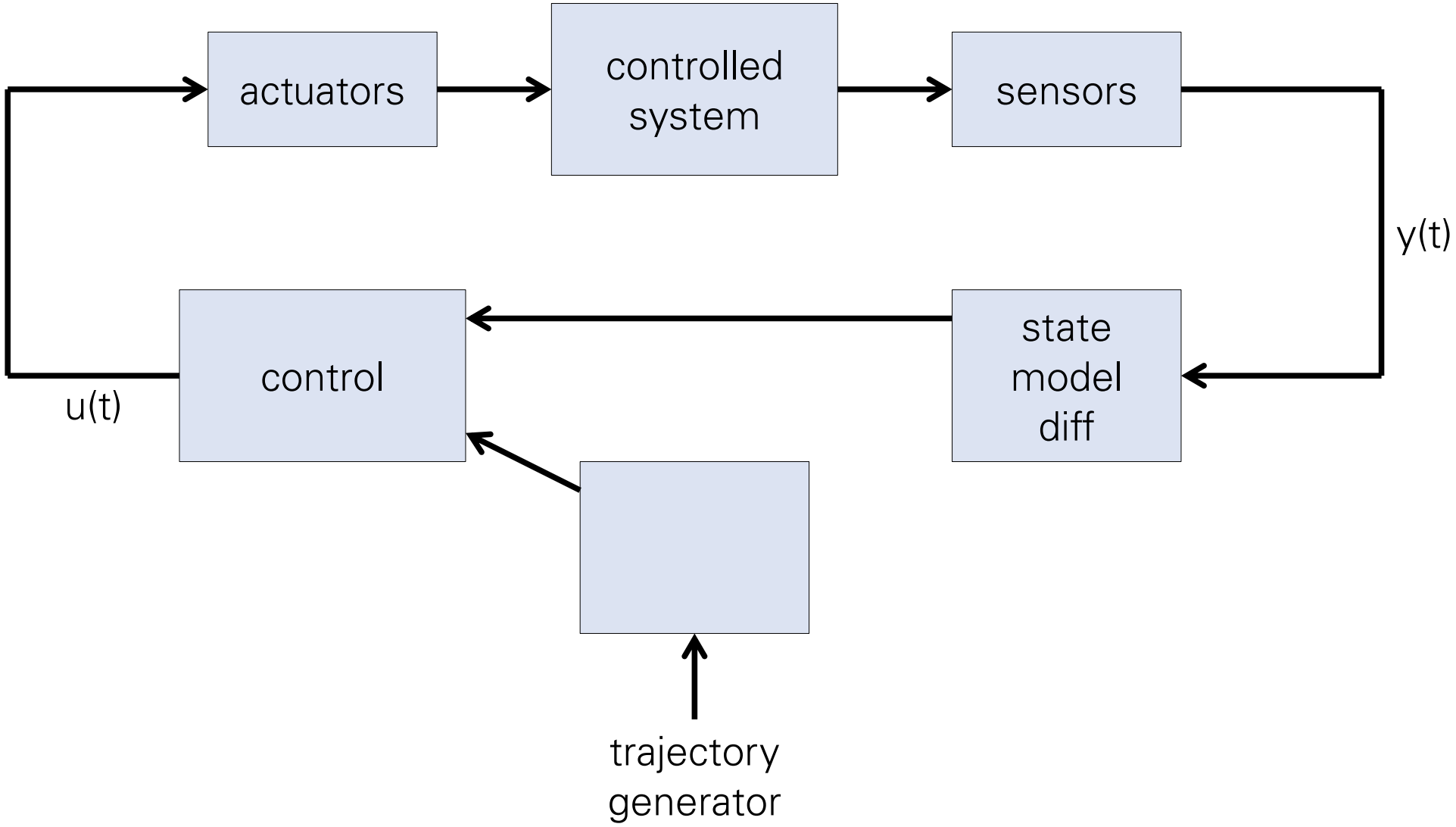
sampled data

(robot arm)

```
state ...
at every xx timeunits do
   read ...
   compute
      output and new state
      Use: samples and current state
   write ...
done
```

Highly dangerous situations:
Internal system model and physical reality not consistent
(Lauda Air and Warsaw plane crashes)

Digital Control System



Complications with simple model: Multirate Controllers

multiple sensors, actuators, and state variables

- with different sampling rates
- often the larger are integer multiples of smaller rates
- harmonic rates

example: rotation, temperature (engine control)

method (successive loop closure):

- start with highest rate sensor
- integrate it in system and consider it part of the controlled object
- determine next rates (as multiples of fastest)

Example: Engine Control

Controls

- amount of fuel
- precise time of injection (0.1 degree of *crankshaft*)

depending on ...

6000 rpm, 10 ms per 360 degree

→ ca. 3micro-sec precision

Examples of Real-Time Systems

- Hermann's next bicycle? - a segway?
- Aircraft: fly by wire, navigation, pilot interface
- Air traffic control (sweeping radar, interpretation, …)
- Measurement systems (sampling rate)
- Alarm Systems (alarm showers)
- Signal processing
- Multimedia Systems
- AWACS

Computer Science Areas

- Parallel programming
- Low Level I/O
- Computer architecture
- Communication
- Modeling techniques (Scheduling)
- Programming languages
- Fault Tolerance
- SW design
- Operations Research
- Computer science theory

Lecture Overview (not a temporal order)

Foundations:

Introduction (Org, Definitions, Examples, CS-Areas)
Time and Order: Time, Clocks (logical and physical)
Modeling Real-Time Systems: Tasks, Jobs, Periods, Resources
Time Driven Systems: Mars Video, Principles, cyclic executives
Event Driven Scheduling: RMS, EDF, Priority Inversion ...

Other areas:

Hardware: Busses, Caches, Catch and Compare, …
Operating Systems: general,case-studies: QNX, Posix, may be: AUTOSAR
Communication: near (CAN), far(ATM), alarm showers (Mars-Example)
HLL: Ada, Esterel
→ real-time literature research (exercises)

„May be“s:

Multiprocessor scheduling
Mixed criticality (per video from Sanjoy Baruah)
Fault Tolerance in Real-Time, Replica Determinism
Statistical Scheduling: SRMS, QRMS
Non Periodic RT-Systems (per video from Doug Jensen)

Material

Textbooks (available in library):

- Hermann Kopetz

  Real-Time Systems (Kluwer)

- Jane Liu

  Real-Time Systems (Prentice Hall)

Additional papers:

provided in lectures