Real-Time Systems and Applications

Prof. Dr. Jian-Jia Chen

LS 12, TU Dortmund

13,04,2015

Organization

- Instructor: Jian-Jia Chen, jian-jia.chen@cs.uni-dortmund.de
- Grading: Oral Exam (100%)
- Credit: 8
- Office hour: Wed., 10:30-11:30 AM. Please make appointments.
- `http://ls12-www.cs.tu-dortmund.de/daes/de/lehre/english-courses/ss15-real-time-systems.html`
- References
  - Textbooks:
    - Giorgio C. Buttazzo, “Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications”, Springer, Third Edition, 2011. ISBN: 0-387-23137-4
    - Jane W.S. Liu, “Real-Time Systems,” Prentice Hall, First Edition, 2000. ISBN:0130996513
  - Papers from conferences and journals

Course Description

Time: Monday: 10:15 - 11:45, Tuesday: 12:15 - 13:45
Place: OH12/E.003
Start: on 13.04.2015
Prerequisite: Computer Operating Systems or equivalent. A basic background in algorithm analysis, data structures, and discrete math will be assumed.
Course material: The slides contain material of the above textbooks, paper references, and course lectures from Steve Goddard, Kevin Jeffay, Nathan Fisher, Lothar Thiele, James Anderson, Kai Lampka, Alan Burns, and Sanjoy Baruah.
Course Calendar:

- The tentative schedule has been announced in the course website.
- The exercise sessions are handled by Dipl.-Inf. von der Brüggen.

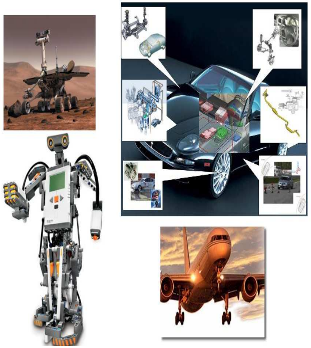
Embedded Systems

Complex "best effort" systems

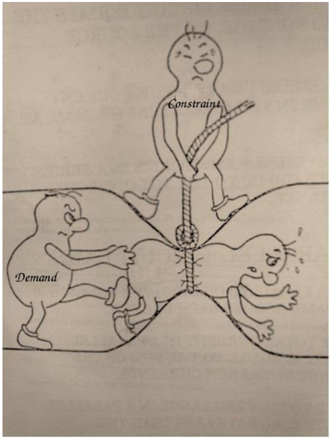
- Mobile telecommunications
- Consumer products, e.g., digital camera, digital video, etc.
- Good reactivity and good dependability

Critical control systems

- Automated aircraft landing systems
- Automotive control for gearing, ABS, airbag, etc.
- High reactivity and high dependability



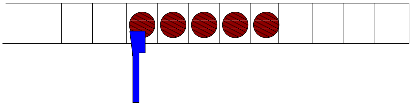
Characteristics of Embedded Systems (Illustrations)



- Even though the hardware improvement continues, the computing demand also increases
- Efficiency
  - Executing should be energy-efficient, code-efficient, and cost-efficient
- Dependability
  - Reliability, maintainability, and availability after deployment
  - Safety even for failure
- Timing guarantee

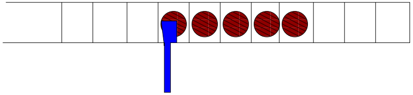
Real-Time Systems

- Dual notations of correctness:
  - Logical correctness ("the results are correct")
    - Requires functional analysis
  - Temporal correctness ("the results are delivered in/on time")
    - Requires non-functional analysis
- High reactivity and high dependability are more important than performance
- **Example:** A robot arm picking up objects from a belt:



Real-Time Systems

- Dual notations of correctness:
  - Logical correctness ("the results are correct")
    - Requires functional analysis
  - Temporal correctness ("the results are delivered in/on time")
    - Requires non-functional analysis
- High reactivity and high dependability are more important than performance
- **Example:** A robot arm picking up objects from a belt:



Examples for Real-Time Systems

- Chemical & Nuclear Power Plants
- Railway Switching Systems
- Flight Control Systems
- Space Mission Control
- Automotive Systems
- Robotics
- Telecommunications Systems
- Stock Market, Trading System,
- Information Access
- Multimedia Systems
- Virtual Reality
- ......

Hard Real-Time Systems

Catastrophic if some deadlines are missed

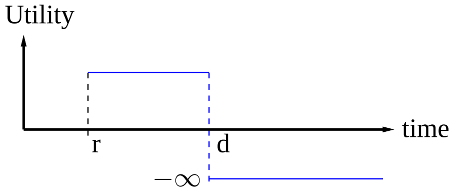
Firmed Real-Time Systems

The results are useless if the deadlines are missed

Soft Real-Time Systems

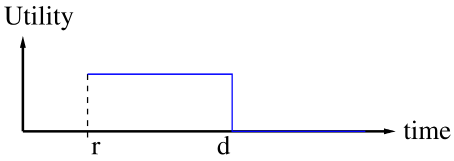
The results are not very useful if the deadlines are missed

Classifications of Real-Time Systems



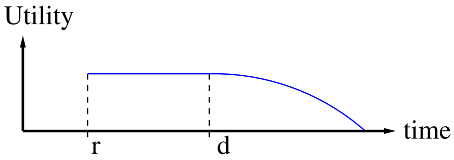
Hard Real-Time Systems

Catastrophic if some deadlines are missed



Firmed Real-Time Systems

The results are useless if the deadlines are missed



Soft Real-Time Systems

The results are not very useful if the deadlines are missed

Characteristics of Real-Time Systems

- Timeliness
- High cost of failure.
- Concurrency/multiprogramming
- Stand-alone/continuous operation
- Design for worst cases
- Reliability/fault-tolerance requirements
- Predictable behavior

Frequent Misconceptions

- “Real time” is performance engineering/tuning.

Frequent Misconceptions

- “Real time” is performance engineering/tuning.
    - Timeliness is more important in real-time systems.

Frequent Misconceptions

- “Real time” is performance engineering/tuning.
    - Timeliness is more important in real-time systems.
- Real-time computing is equivalent to fast computing.

Frequent Misconceptions

- “Real time” is performance engineering/tuning.
    - Timeliness is more important in real-time systems.
- Real-time computing is equivalent to fast computing.
    - Real-time computing means predictable and reliable computing.

Frequent Misconceptions

- "Real time" is performance engineering/tuning.
    - Timeliness is more important in real-time systems.
- Real-time computing is equivalent to fast computing.
    - Real-time computing means predictable and reliable computing.
- There is no science in real-time system design.

Frequent Misconceptions

- "Real time" is performance engineering/tuning.
    - Timeliness is more important in real-time systems.
- Real-time computing is equivalent to fast computing.
    - Real-time computing means predictable and reliable computing.
- There is no science in real-time system design.
    - Let's discuss this at the end of the semester.

Frequent Misconceptions

- “Real time” is performance engineering/tuning.
    - Timeliness is more important in real-time systems.
- Real-time computing is equivalent to fast computing.
    - Real-time computing means predictable and reliable computing.
- There is no science in real-time system design.
    - Let’s discuss this at the end of the semester.
- Advances in supercomputing hardware will take care of real-time requirements.

Frequent Misconceptions

- "Real time" is performance engineering/tuning.
    - Timeliness is more important in real-time systems.
- Real-time computing is equivalent to fast computing.
    - Real-time computing means predictable and reliable computing.
- There is no science in real-time system design.
    - Let's discuss this at the end of the semester.
- Advances in supercomputing hardware will take care of real-time requirements.
    - Buying a "faster" processor may result in timeliness violation.

Frequent Misconceptions

- "Real time" is performance engineering/tuning.
  - Timeliness is more important in real-time systems.
- Real-time computing is equivalent to fast computing.
  - Real-time computing means predictable and reliable computing.
- There is no science in real-time system design.
  - Let's discuss this at the end of the semester.
- Advances in supercomputing hardware will take care of real-time requirements.
  - Buying a "faster" processor may result in timeliness violation.
- It is not meaningful to talk about guaranteeing real-time performance when things can fail.

Frequent Misconceptions

- “Real time” is performance engineering/tuning.
  - Timeliness is more important in real-time systems.
- Real-time computing is equivalent to fast computing.
  - Real-time computing means predictable and reliable computing.
- There is no science in real-time system design.
  - Let’s discuss this at the end of the semester.
- Advances in supercomputing hardware will take care of real-time requirements.
  - Buying a ”faster” processor may result in timeliness violation.
- It is not meaningful to talk about guaranteeing real-time performance when things can fail.
  - Though hardwares may fail, the logic components, such as operating systems, should be solid when hardwares are still well functional.

Needs of Concurrency

Reasons for concurrency

- Functional
  - allow multiple users
  - perform many operations concurrently
- Performance
  - take advantage of blocking time
  - parallelism in multi-processor machines
- Expressive Power
  - many control application are inherently concurrent
  - concurrency support helps in expressing concurrency, making application development simpler

Multi-Tasking

- The execution entities (tasks, processes, threads, etc.) are competing from each other for shared resources
- Scheduling decision policy is needed
  - When to schedule an entity?
  - Which entity to schedule?
  - How to schedule entities?

General-Purpose Systems and Real-Time Systems

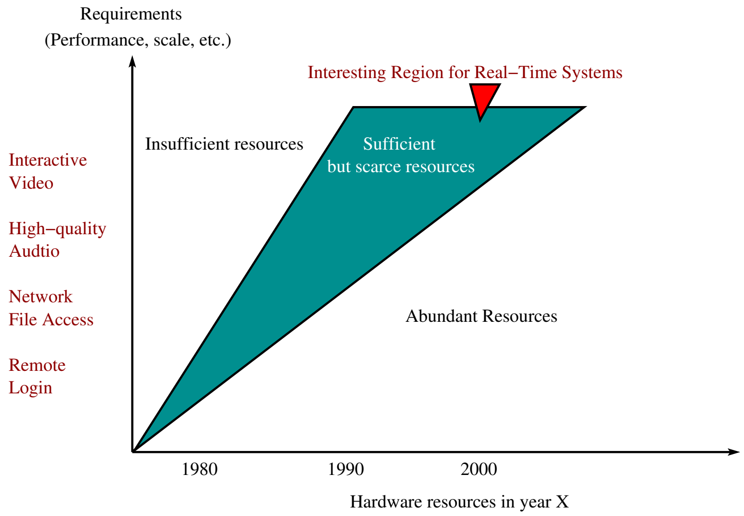
Real-Time Systems

- Applications are known a priori
- Programed by designers
- Timeliness
  - Worst-case response time guarantee
- Examples: traffic control systems, robotics, etc.

General-Purpose Systems

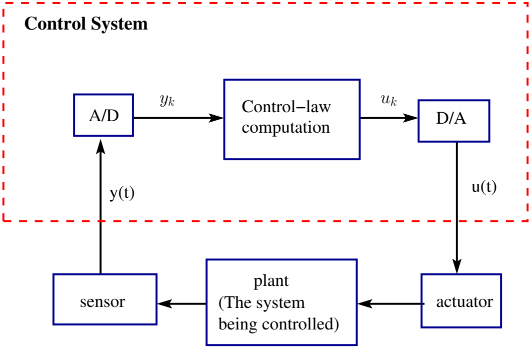
- Broad class of applications
- Programmable by end-users
- The faster, the better
  - Average-case response time
- Examples: Desktops, web servers, etc.

Scarcity of Resources



Example: Simple Control System

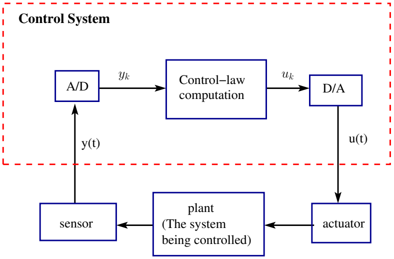
- Many embedded systems are control systems
  - Robotics
  - Automotive Systems
- A simple example: one-sensor and one-actuator control system.



Example: Simple Control System (cont.)

Pseudo-code for this system

```
set timer to interrupt periodically with period $T$;
at each timer interrupt **do**
do analog-to-digital conversion to get $y$;
compute control output $u$;
output $u$ and do digital-to-analog conversion;
**od**
```



T is called the sampling period, which is a key design choice.

Example: Multi-Rate Control System

More complicated control systems have multiple sensors and actuators and must support control loops of different rates.
Example: Helicopter flight controller.

Do the following in each 1/180-sec. cycle:

validate sensor data and select data source;
if failure, reconfigure the system

Every six cycles do::

keyboard input and mode selection;
data normalization and coordinate transformation;
tracking reference update control laws of the outer pitch-control loop;
control laws of the outer roll-control loop;
control laws of the outer yaw- and collective-control loop;

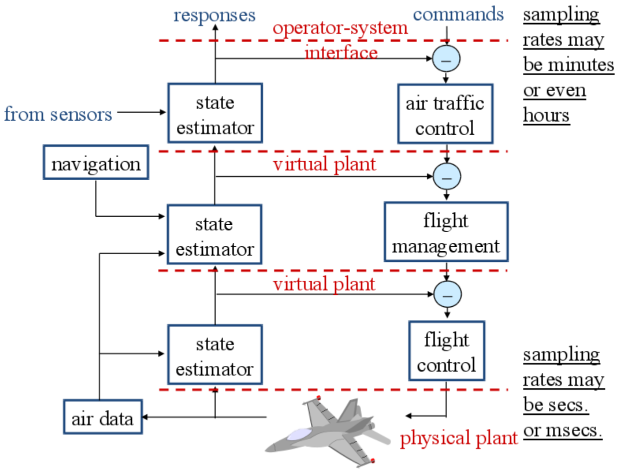
Every other cycle do:

control laws of the inner pitch-control loop;
control laws of the inner roll- and collective-control loop;
Compute the control laws of the inner yaw-control loop;

Output commands;
Carry out built-in test;
Wait until beginning of the next cycle;

This leads to periodic execution of each control task.

Example: Hierarchical Control System



Course Focus

- Basic tools and analytical methods for real-time systems and applications
  - Scheduling algorithms for real-time systems
    - earliest-deadline-first scheduling, rate monotonic scheduling, deadline monotonic scheduling, etc.
    - resource sharing and servers
    - multiprocessor scheduling
  - Basic analysis of timing satisfaction in real-time systems and applications
  - Real-time operating systems, communication and programming languages
- Several advanced topics related to the current research directions, including resource augmentation analysis, mixed criticality, and real-time calculus

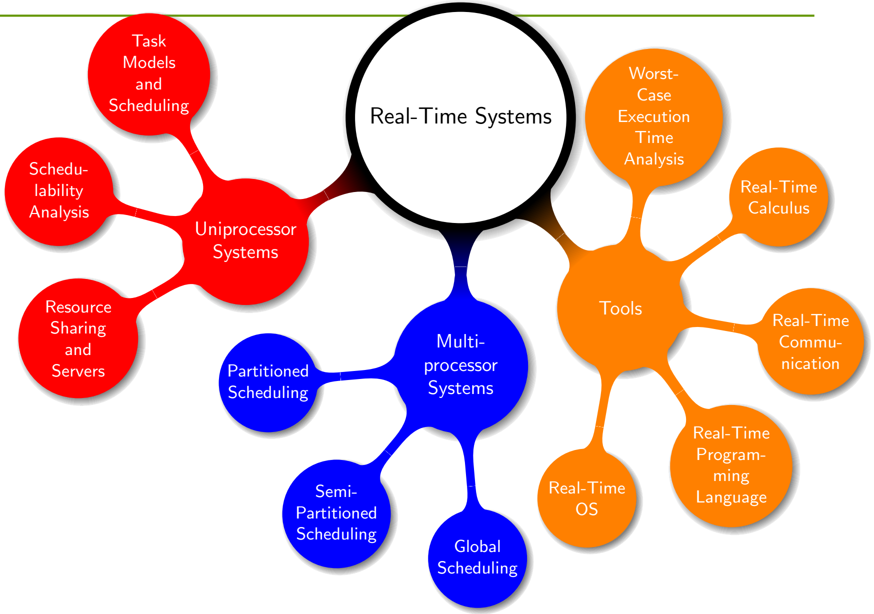
Course Goal

After the course, you are expected to know

- fundamental building blocks in real-time systems and applications, and
- schedulability analysis of scheduling algorithms in real-time systems and applications.

Application sectors:

- Safety-critical applications: avionics, automotive systems, etc.
- Best-effort applications with worst-case performance requirement: telecommunications, multimedia systems, etc.
- Embedded systems and cyber-physical systems


Real-Time Systems
Uniprocessor Systems
Task Models and Scheduling
Schedu-lability Analysis
Resource Sharing and Servers
Multi-processor Systems
Partitioned Scheduling
Semi-Partitioned Scheduling
Global Scheduling
Tools
Worst-Case Execution Time Analysis
Real-Time Calculus
Real-Time Commu-nication
Real-Time Program-ming Language
Real-Time OS

Course Calendar

announced in `http://ls12-www.cs.tu-dortmund.de/daes/de/lehre/english-courses/ss15-real-time-systems.html`

Real-Time Systems Community

For information on real-time research groups, conferences, journals, books, products, etc., have a look at:

- http://tcrts.org/