Introduction to Real-Time Systems

Peter Puschner

slides credits: H. Kopetz, P. Puschner

VO Echtzeitsysteme

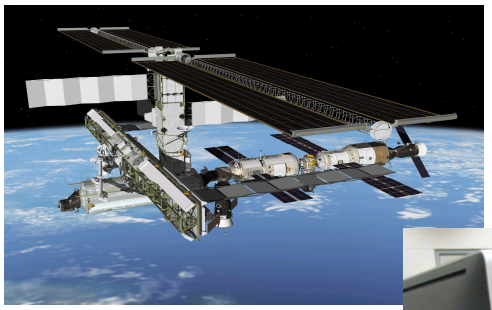
SS 2017

What is a Real-Time System?

- Definition 1: RT-systems are systems in which the correctness of the system behavior depends
  - on the logical results of the computations, and
  - on the physical time when these results are produced

- Definition 2: RT-systems are systems that have to be designed according to the dynamics of a physical process

What is a Real-Time System? (2)



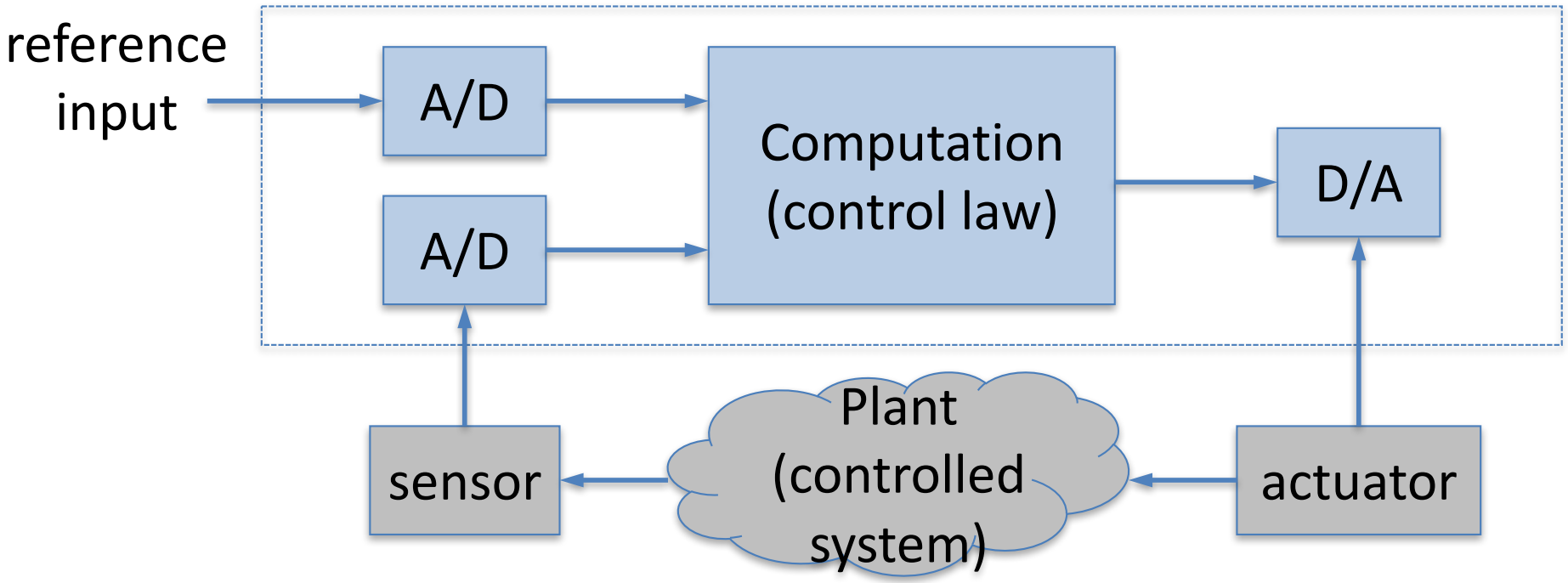
What is a Real-Time System? (3)

- Often part of an embedded or cyber-physical system
  - Computer system performs a specific task (not general purpose)
  - Tight interaction with physical environment (sensors, actuators)
  - Dependability
  - Resource efficiency (cost are critical)
  - Increasing importance of security

Example Real-Time Application

Many real-time systems are control systems

Example: simple one-sensor, one-actuator control system



Example Real-Time Application – Pseudo Code

```
Initialize periodic interrupt timer with period T

Interrupt service routine:
      do analog-to-digital conversion for input value
      compute control output from reference and input value
      do digital-to-analog conversion for control output
```

- T … sampling period
- T is application dependent, chosen by system designer
- Range of T: milliseconds to seconds

Misconceptions about Real-Time Systems

(Stankovic, IEEE Computer, 1988)

- “Real-time computing is equivalent to fast computing.”
  - “real-time” sounds cool/good – term often used to advertise products
- “Real-time programming is assembly coding,…”
  - Proper models, design and development process

Challenges – What is Difficult about RTS?

1. Reactive behavior
   - Continuous operation
   - Pace is controlled by environment
2. Concurrency
   - Devices operate in parallel in the real-world
   - Conflicts with sequential execution on controller
   - Hard to maintain deterministic, reproducible behavior
3. Guaranteed response times
   - Predictability is essential – still efficiency is important
   - Worst case must be predictable
   - Response times on system level

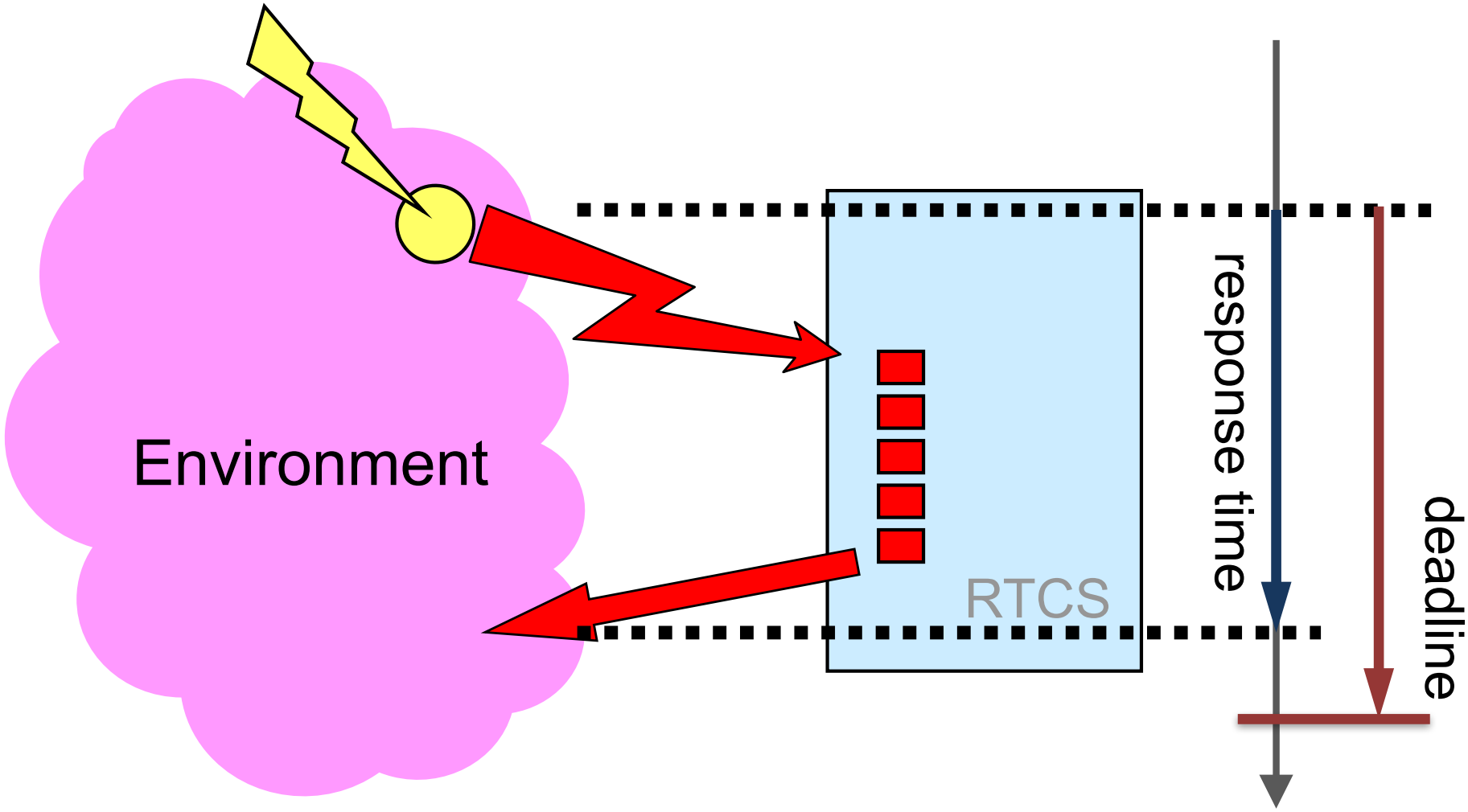
What is Difficult about RTS?

4. Interaction with special purpose hardware
   - Devices must be programmed in a reliable and abstract way
   - Interfaces, device drivers are often a large development-time sink
5. Maintenance usually difficult
   - Hardly maintenance loop
   - Instead: “First time right”
6. Harsh environment
   - Temperature, EMI, radiation, etc.
7. Constrained resources
   - Processing power, memory, power, etc.

What is Difficult about RTS?

8. Often cross development
   - Target platform ≠ development platform
9. Size and complexity
   - Few lines of assembler code … x100 million lines of code (car, plane)
10. Reliability and safety requirements
    - Embedded systems control the environment in which they operate
    - Control failures can result in
      - enormous damage to environment
      - substantial financial loss
      - the loss of human life

Deadline



Deadline

- The time at which a real-time systems has to produce a specific result is called a deadline.
- Deadlines are dictated by the environment.
- What happens if an RTS misses a deadline?

Classification of Real-Time Systems

- Soft RTS
  - The result has utility after the deadline.
  - Respective deadline is called a soft deadline.
- Firm RTS
  - The result has zero utility after the deadline.
- Hard RTS
  - Missing a deadline may be catastrophic.
  - Critical deadline is called hard deadline.
  - HRTS has at least one hard deadline
- Hard and Soft RTS design are fundamentally different!

Fail-Safe versus Fail-Operational Applications

Fail-safe system: has a safe state in the environment that can be reached in case of a system failure (e.g., train signaling).

- Fail safeness is an application property.
- High *error detection coverage* is critical.
- Use of watchdog, heart-beat signal.

Fail-operational system: no safe state can be reached in case of a system failure (e.g., a flight control system of airplane).

- Computer system has to provide a minimum level of service, even after the occurrence of a fault.
- Active redundancy

Guaranteed Timeliness versus Best Effort

Guaranteed timeliness of a system implementation

- Load and fault hypothesis is available
- Temporal correctness can be shown by analytical arguments
- Assumption coverage is critical

Best effort system implementation

- Analytical argument for temporal correctness cannot be made.
- The temporal verification relies on probabilistic arguments, even within the specified load- and fault hypothesis.

Hard real-time systems must be based on guaranteed timeliness.

Resource Adequacy

In order to provide timing guarantees a system has to

- provide sufficient computational resources to handle
- the specified peak load and
- fault scenarios.

In the past, resource adequacy has been considered *too expensive*.

Today, decreasing hardware cost make the implementation of resource adequate designs economically viable.

For hard real-time applications,
there is no alternative to resource adequate designs.

Predictability in Rare-Event Situations

Rare Event

- important event that
- occurs very infrequently during the lifetime of a system (e.g., the rupture of a pipe in a nuclear reactor).
- can give rise to many correlated service requests (e.g., an alarm shower).

In a number of applications

- the utility of a system depends on the predictable performance in rare event scenarios (e.g., flight control system).
- In many cases, workload testing will not cover the rare event scenario.

Hard versus Soft RTS

Characteristic	Hard Real Time	Soft Real Time
Deadlines	hard	soft
Pacing	environment	computer
Peak-Load Perform.	predictable	degraded
Error Detection	system	user
Safety	critical	non-critical
Redundancy	active	standby
Time Granularity	millisecond	second
Data Files	small/medium	large
Data Integrity	short term	long term

Points to Remember

- RT is not about performance (fast is not real-time)
- Hard RT systems are safety critical
- Predictability is important
- RT does not imply ad-hoc, low-level design
- RT design has to be systematic
  - Timing is central
  - Architecture (hardware and software)
  - Design, implementation and verification process