AARHUS
UNIVERSITY
DEPARTMENT OF ENGINEERING

Architecture and Design of Embedded Real-Time Systems (TI-AREM)

Basic Concepts and definitions of Embedded Real-Time Systems

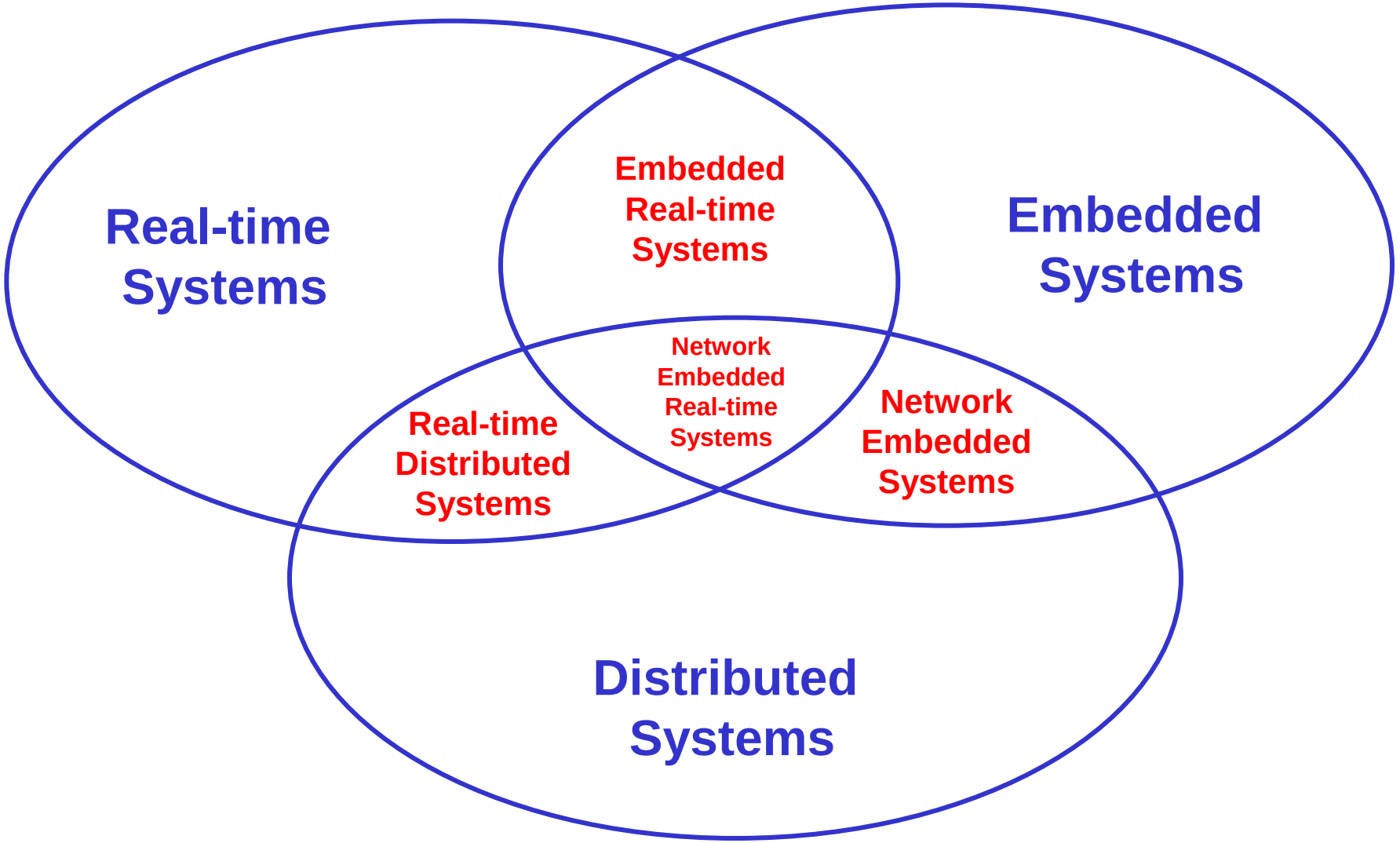
Ref. Bruce P. Douglas “Doing Hard Time”
Ref. Hermann Kopetz: ”Real-Time Systems”

Version: 8-1-2015

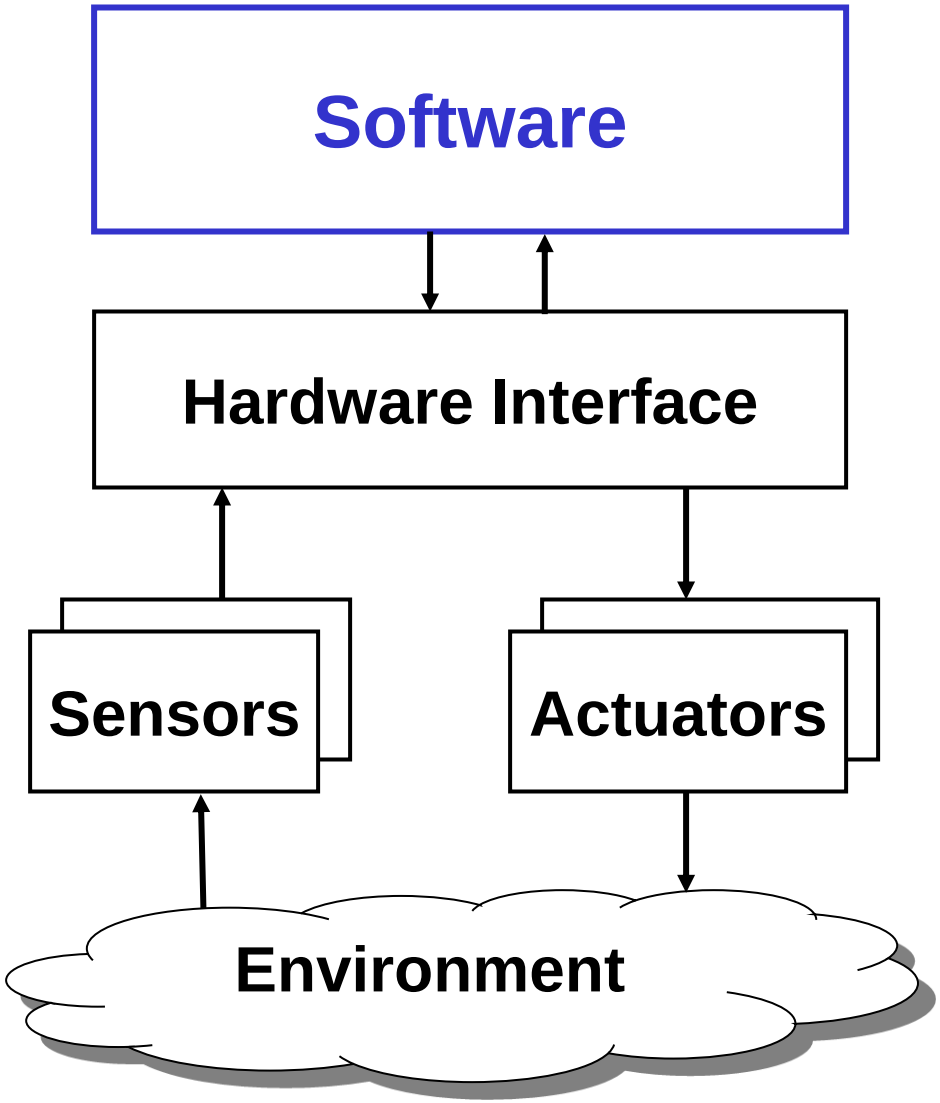
Agenda

- Introduction to Real-time embedded systems
- Presentation of basic concepts
  - Time in RT systems
  - Scheduling
  - Deadlock
  - Robust systems

System Characteristics



Basic Elements of a Real-time System



Characteristics of Real-Time Systems (1)

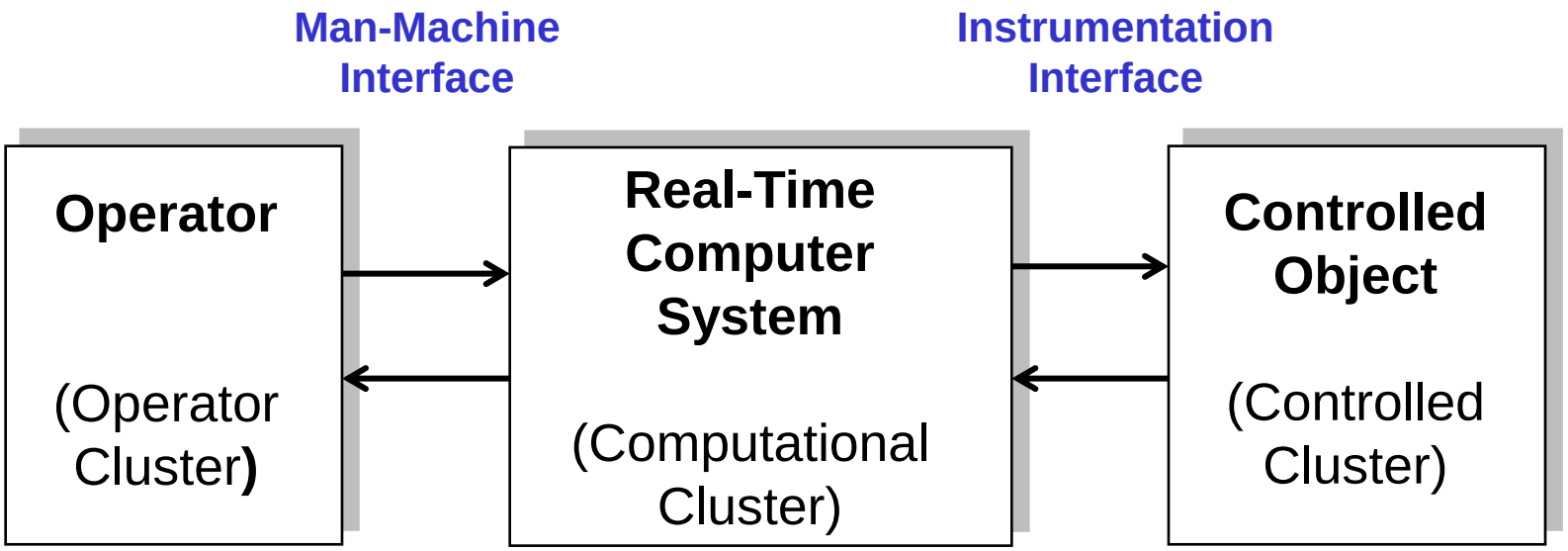
- A Real-time system has **performance deadlines** on its computations and actions
- A Real-time systems **interacts with its environment** through sensors and actuators
- Real-Time systems are subdivided in:
  - ***Hard** Real Time* systems
    - systems with very tight and hard deadlines, where **missing a deadline is an error**
  - ***Soft** Real Time* systems
    - systems with more soft deadlines, where **missing a deadline can be accepted**

Characteristics of Real-Time Systems (2)

- Real-time systems can be:
  - **Reactive** or **event-driven** systems
    - is a system where behavior is caused **by reactions to external events**
  - **Time-driven** systems
    - is a system whose behavior is driven by the passage of time
  - or a combination of these
- Real-time systems should react on many independent and possible concurrent events
  - A solution is to use ***concurrency***:
    - with concurrent executing tasks managed by a real-time operating system or a real-time kernel

When is a Computer System Real-Time?

”A real-time computer system is a computer system in which the correctness of the system behavior depends not only on the **logical results** of the computations, but also on the **physical instant** at which these results are produced”

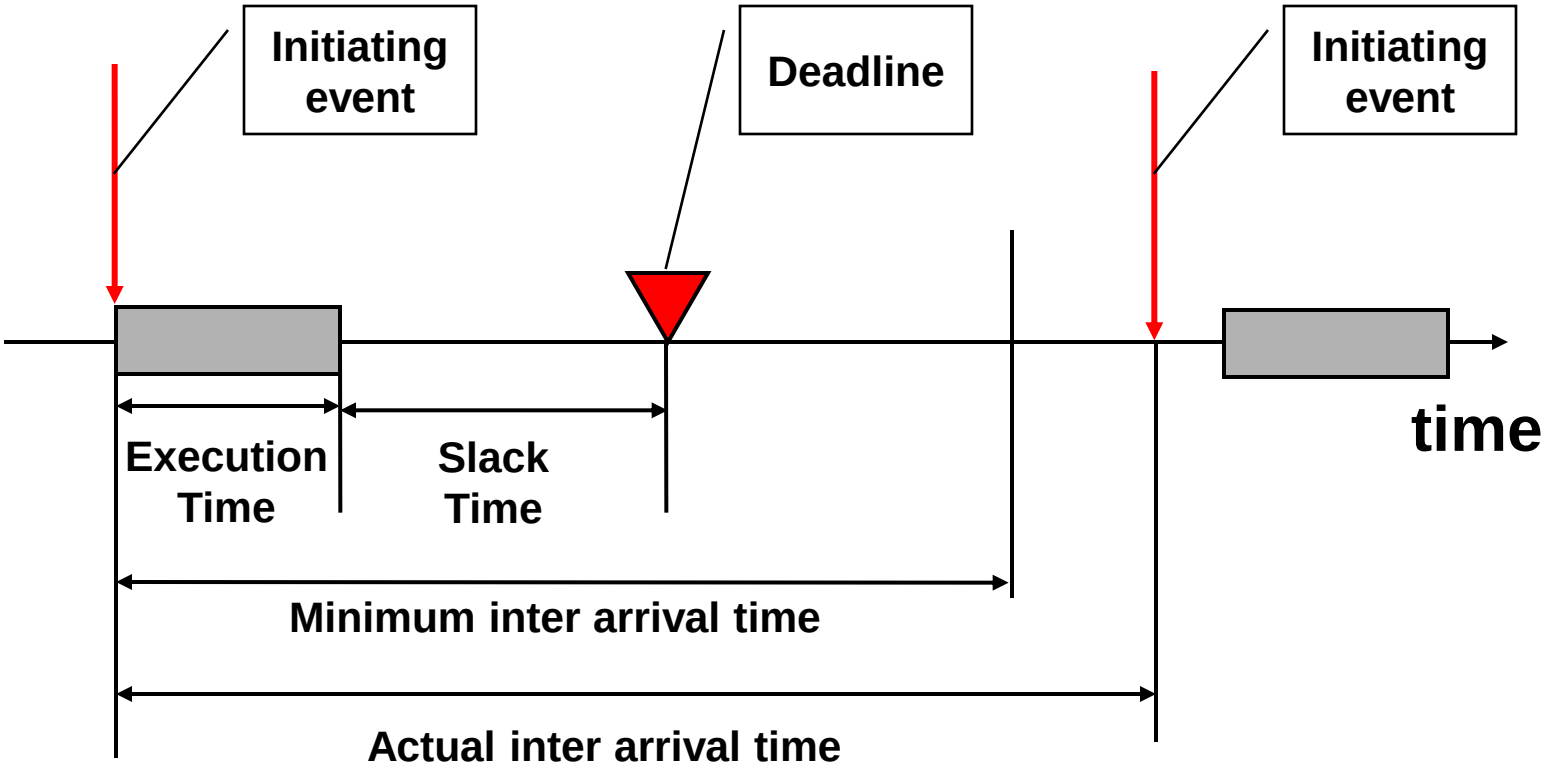


When is a Computer System Real-Time?

- A real-time computer system **must react** to stimuli from the controlled object (or the operator) **within time intervals dictated by its environments**
- A **deadline**: the instant at which a result must be performed
- A **hard** deadline:
  - if a catastrophe could result if a deadline is missed
  - A system with at least one hard deadline is called a **hard real-time system** or a **safety-critical real-time computer system**
  - else a **soft real-time computer system**
- A **soft** deadline:
  - if a result has **utility** even after passing the deadline

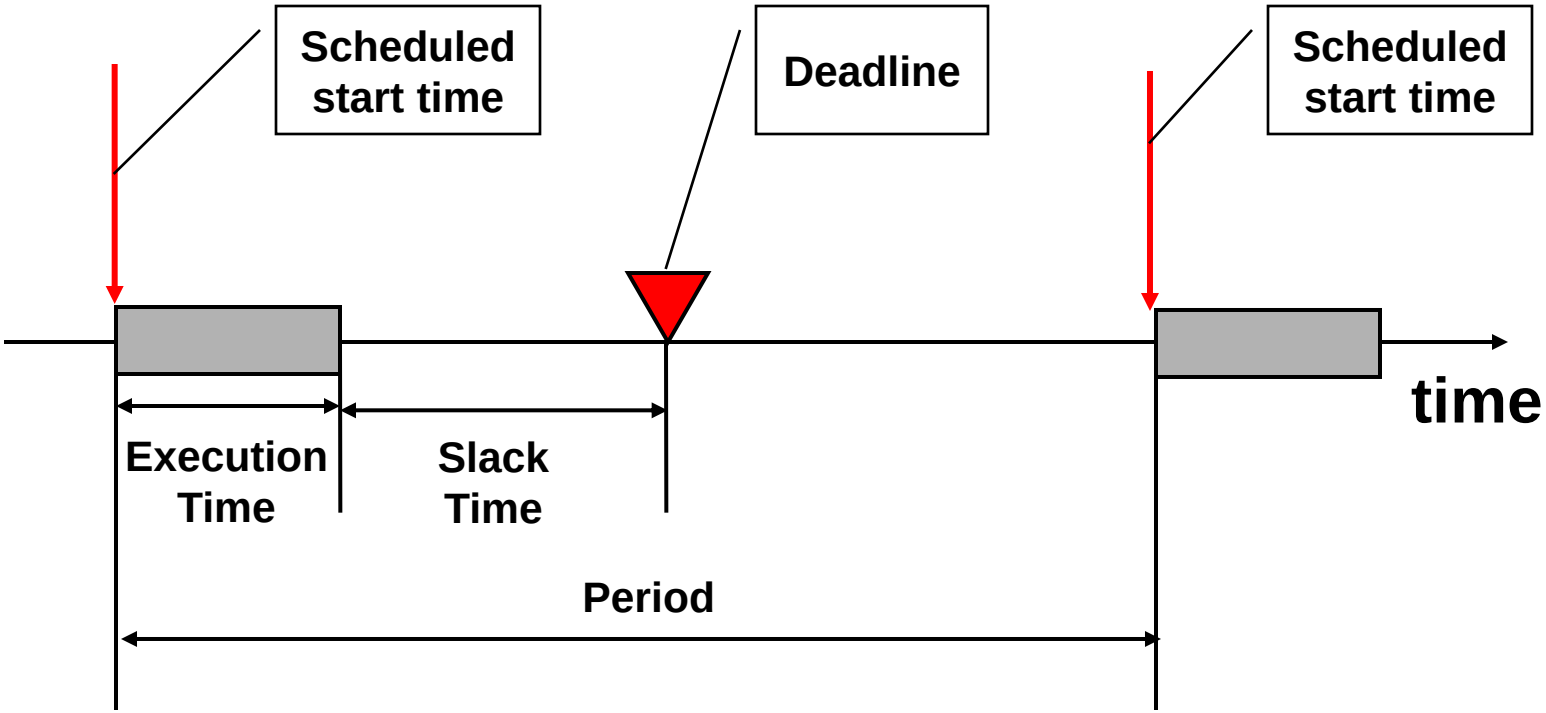
Definition of terms - 1

Event-Driven task



Definition of terms - 2

Time-Driven task

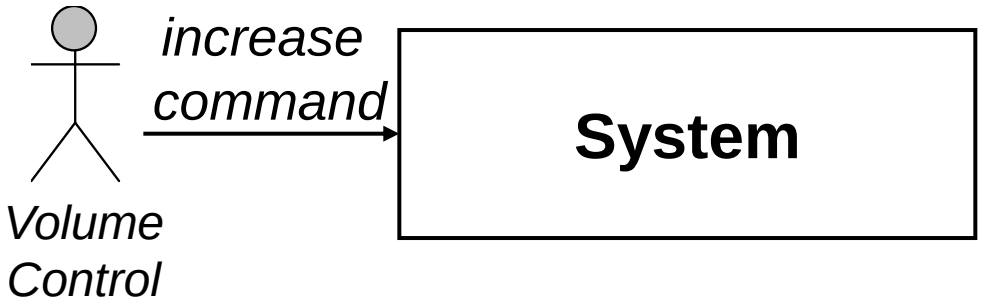


Handling of Time

- Time requirement can be:
  - Hard
    - deadlines must not be missed – e.g. a pacemaker
  - Soft
    - minor exceeding of deadlines are accepted – e.g. response time in a reservation system
  - Firm
    - a combination of soft and hard – e.g. a ventilator, where a hard long time requirement is to ventilate the patient before a hard maximum timespan

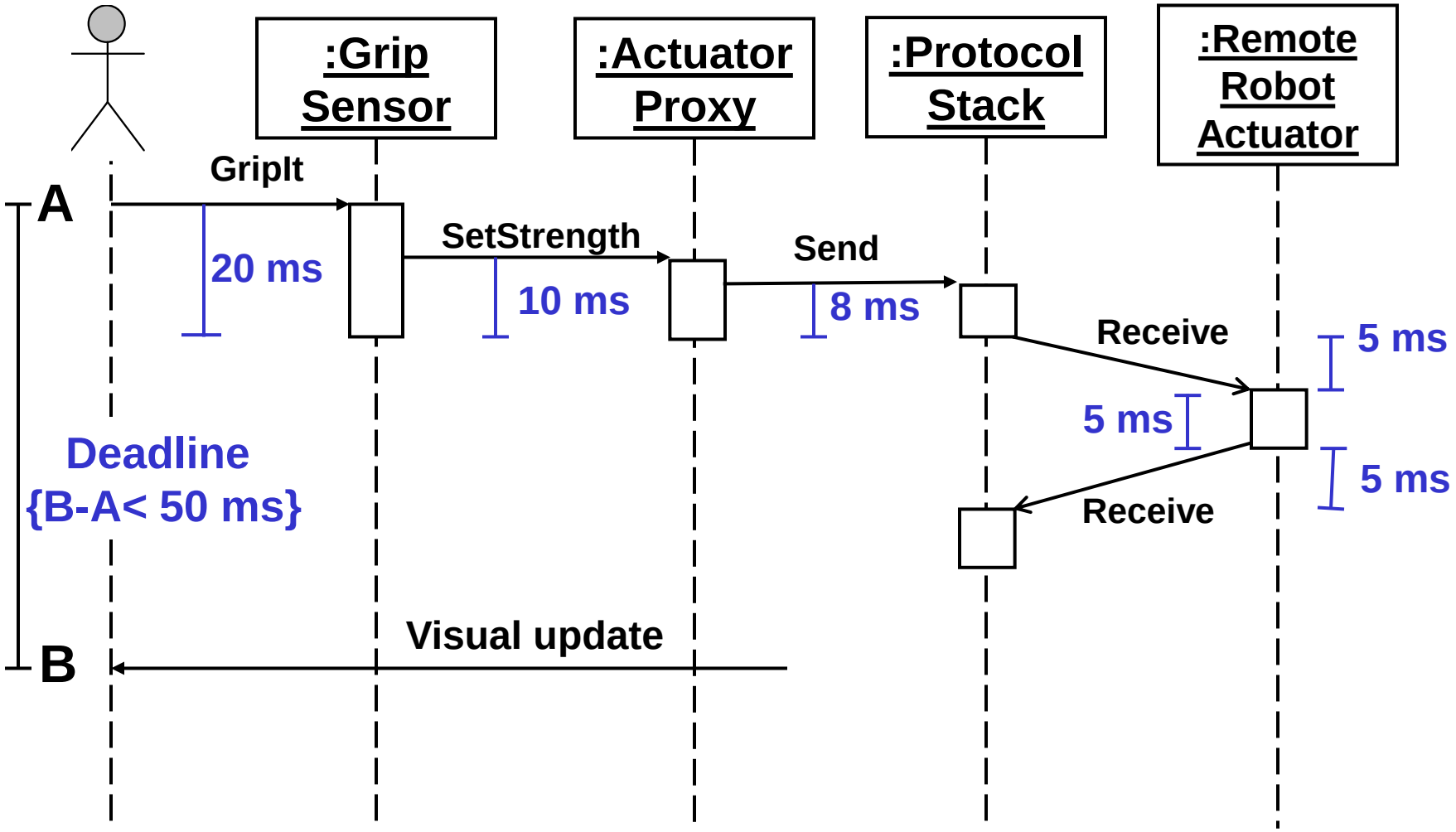
Timeliness

- Example:
  - *When the actor VolumeControl sends an increase command, the system shall respond within 10 ms*



Performance budgets are first computed from a black-box perspective – later on these deadlines propagates into performance budgets on individual operations and actions

Assignment of Time Budgets



Event-Triggered versus Time-Trigged

- Two distinctly different approaches to the design of real-time computer applications can be identified:
  - ***Event-triggered approach (ET)***
    - all communication and processing activities are initiated whenever a significant change of state is noted, i.e. an event
    - signaling of events can be realized by the interrupt mechanism
  - ***Time-Triggered approach (TT)***
    - all communication and processing activities are initiated at predetermined points in time
    - initiated by a periodic clock interrupt

Event - versus State Information

- An **event**:
  - any occurrence that happens at a given point in time
  - information describing an event is called **event information**
- A **state attribute**:
  - any property of an RT entity that remain valid during a finite duration
  - information describing a state attribute is called **state information**

Event Arrival patterns

- Arrival patterns may be either **periodic** or **aperiodic**
- **Periodic** events has a fixed period T plus or minus a small variation (called **jitter**)
- The timing of **aperiodic** events may be:
  - **Irregular** (a known, but varying sequence of intervals)
  - **Bursty** (can occur arbitrarily close to each other)
  - **Bounded** (with a known minimum inter arrival time)
  - **Bounded average rate** (clusters around a mean)
  - **Unbounded** (can only be predicted statistically)

Finite State Machines (FSM)

- **Reactive systems** are often modeled as Finite State Machines (FSM)
  - An FSM can be in only one state at a time and must be in exactly one state at all times
  - Transitions between states are not interruptable and run to completion
  - Actions are atomic and takes app. zero time
  - Actions may be executed:
    - on entry to a state
    - on exit from a state
    - During the transition from one state to another
- FSMs are modeled with UML State Diagrams

Scheduling Concurrent Threads

- Most prevalent scheduling strategies:
  - FIFO run-to completion event handling
    - as an independent set of interrupt handlers
  - Cyclic executive
    - run to completion of a set of ordered threads
  - Non-preemptive task switching
    - where the threads voluntarily release control to the kernel
  - Preemptive: time-slicing round robin
    - where a task is preempted when the fixed allocated timeslot is exceeded
  - Preemptive: priority-based preemption
    - where the current executing task is suspended when a higher priority task becomes ready to run

Deadlock problem

- The following **four conditions** must be true for a **deadlock** to occur:
  1. Tasks claims exclusive control over shared resources
  2. Tasks hold resources while waiting for other resources to be released
  3. Tasks cannot be forced to relinquish control
  4. A circular waiting condition exists

Robust Systems

- *”A robust system is one that continue to do the right thing even in the presence of system faults”*
- A robust system must:
  - Identify the fault
  - Take ”evasive actions”, such as
    - Correct the failure and continue processing
    - Repeat the previous computation to restore the correct system state
    - Enter a **fail-safe state**

Summary

- Definitions, terms and characteristics of embedded real-time systems

References

- *”Doing Hard Time – Developing Real-Time Systems using UML, Objects, Framework and Patterns”* – Bruce Powel Douglass, Addison-Wesley 1999
- *”Real-Time Systems, Design Principles for Distributed Embedded Applications”* – Hermann Kopetz, Kluwer Academic Publishers, 1997.