NAZARBAYEV UNIVERSITY
SCHOOL OF SCIENCE AND TECHNOLOGY

10 September, 2015

ROBT305 – Embedded Systems

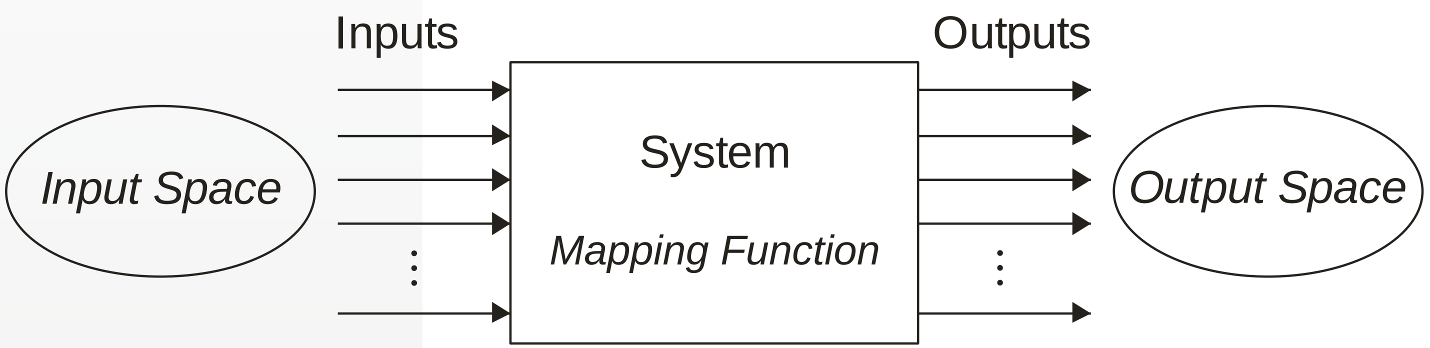
Lecture 7 – Real-Time Systems. Basic definitions

Course Logistics

Reading Assignment:
Chapter 1 of the Real-Time Systems Design and Analysis textbook
Homework Assignment #1 is out in Moodle and due to end of 13 September (Sunday)
Quiz #2 is on 17 September – Pthreads, Mutexes

Definition: System

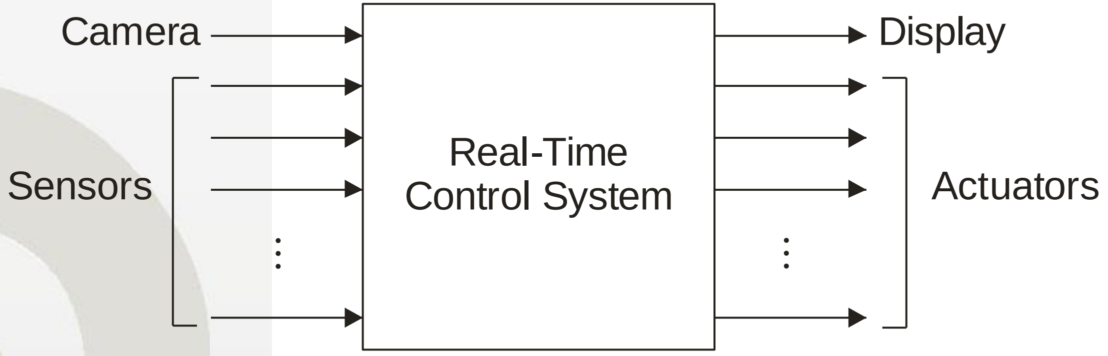
A system is a mapping of a set of inputs into a set of outputs



1. A system is an assembly of components connected together in an organized way
2. A system is fundamentally altered if a component joins or leaves it
3. It has a purpose
4. It has a degree of permanence
5. It has been defined as being of particular interest

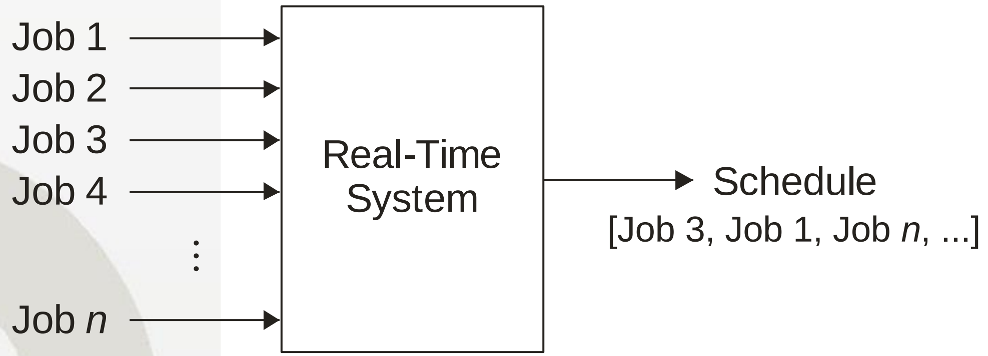
Example: A Real-Time Control System

- Inputs are *excitations* and outputs are corresponding *responses*
- Inputs and outputs may be digital or analog
- Inputs are associated with sensors, cameras, etc.
- Outputs with actuators, displays, etc.



Classic Representation

- A sequence of jobs to be scheduled and performance to be predicted
- Ignores the usual fact that the input sources and hardware under control may be highly complex



Definition: Response Time

The time between the presentation of a set of inputs to a system and the realization of the required behavior, including the availability of all associated outputs, is called **the response time** of the system

- How fast and punctual does it need to be?
  - Depends on the specific real-time system
- But what is a real-time system?

Definitions: Real-Time System

- **A real-time system** is a computer system that must satisfy bounded response-time constraints or risk severe consequences, including failure

- **A real-time system** is one whose logical correctness is based on both the correctness of the outputs and their timeliness

Definition: Failed System

A failed system is a system that cannot satisfy one or more of the requirements stipulated in the system requirements specification

- Hence, rigorous specification of the system operating criteria, including timing constraints, is necessary

Definition: Embedded System

An embedded system is a system containing one or more computers (or processors) having a central role in the functionality of the system, but the system is not explicitly called a computer

- A real-time system may be embedded or non-embedded
- But it is always reactive
  - Task scheduling is driven by ongoing interaction with the environment

Degrees of "Real-Time"

- All practical systems are ultimately real-time systems
- Even a batch-oriented system—for example, grade processing at the end of a semester—is real-time
- Although the system may have response times of days, it must respond within a certain time
- Even a word-processing program should respond to commands within a reasonable amount of time
- Most of the literature refers to such systems as soft real-time systems

Soft, Hard, and Firm “Real-Time”

Definition: **Soft Real-Time System**

A soft real-time system is one in which performance is degraded but not destroyed by failure to meet response-time constraints

Definition: **Hard Real-Time System**

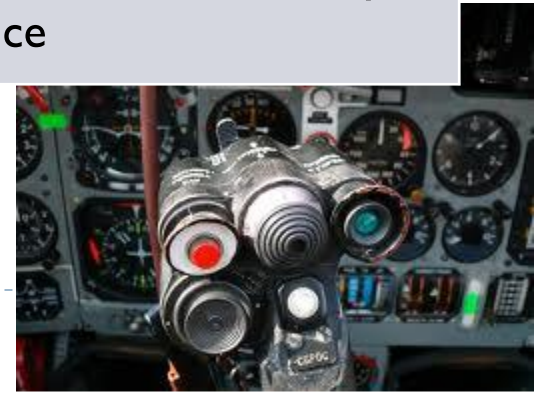
A hard real-time system is one in which failure to meet even a single deadline may lead to complete or catastrophic system failure

Definition: **Firm Real-Time System**

A firm real-time system is one in which a few missed deadlines will not lead to total failure, but missing more than a few may lead to complete or catastrophic system failure

Example: Real-Time Classification

System	Real-Time Classification	Explanation
Avionics weapons delivery system in which pressing a button launches an air-to-air missile	Hard	Missing the deadline to launch the missile within a specified time after pressing the button may cause the target to be missed, which will result in a catastrophe
Navigation controller for an autonomous weed-killer robot	Firm	Missing a few navigation deadlines causes the robot to veer out from a planned path and damage some crops
Console hockey game	Soft	Missing even several deadlines will only degrade performance



Where Do Deadlines Come from?

- Deadlines are based on the underlying physical phenomena of the system under control
- Punctuality is another measure related to response times
  - Particularly important in periodically sampled systems with high sampling rates (e.g., in audio and video signal processing)
- Rule of thumb for cost-effective real-time systems:
  - Process everything as slowly as possible and repeat tasks as seldom as possible

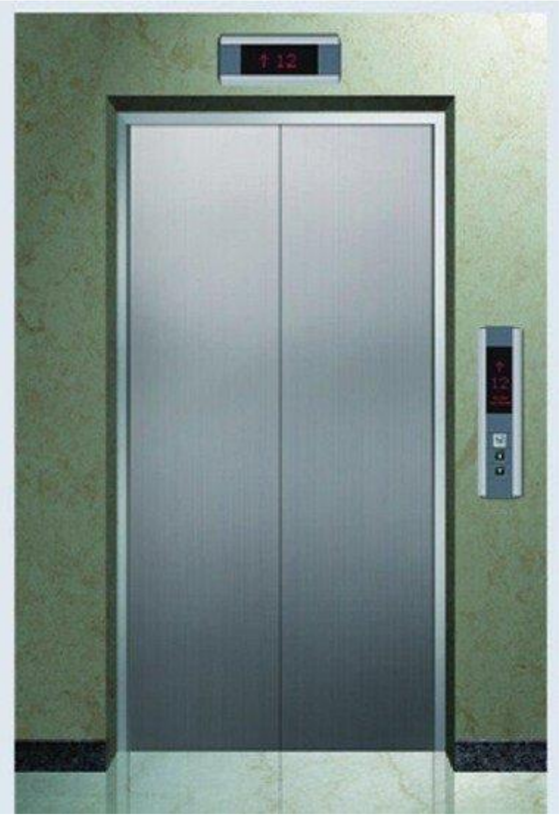
Definition: Real-Time Punctuality

Real-time punctuality means that every response time has an average value, t_{R}, with upper and lower bounds of $t_{\mathrm{R}} + \varepsilon_{\mathrm{U}}$ and $t_{\mathrm{R}} - \varepsilon_{\mathrm{L}}$, respectively, and $\varepsilon_{\mathrm{U}}, \varepsilon_{\mathrm{L}} \rightarrow 0^{+}$

- In all practical systems, the values of ε_{U} and ε_{L} are nonzero, though they may be very small
- The nonzero values are due to cumulative latency and propagation-delay components (hardware/software)
- Such response times contain jitter within the interval $t \in [-\varepsilon_{\mathrm{L}}, +\varepsilon_{\mathrm{U}}]$

Example: Where a Response Time Comes from?

- An elevator door is automatically operated and it may have a capacitive safety edge for sensing possible passengers between the closing door blades
- Thus, the door blades can be quickly reopened before they touch the passenger and cause discomfort or even threaten the passenger's safety
- What is the required system response time from when it recognizes that a passenger is between the closing door blades and starting to reopen the door?



Door Reopening Example Cont'd

This response time consists of five independent latency components:

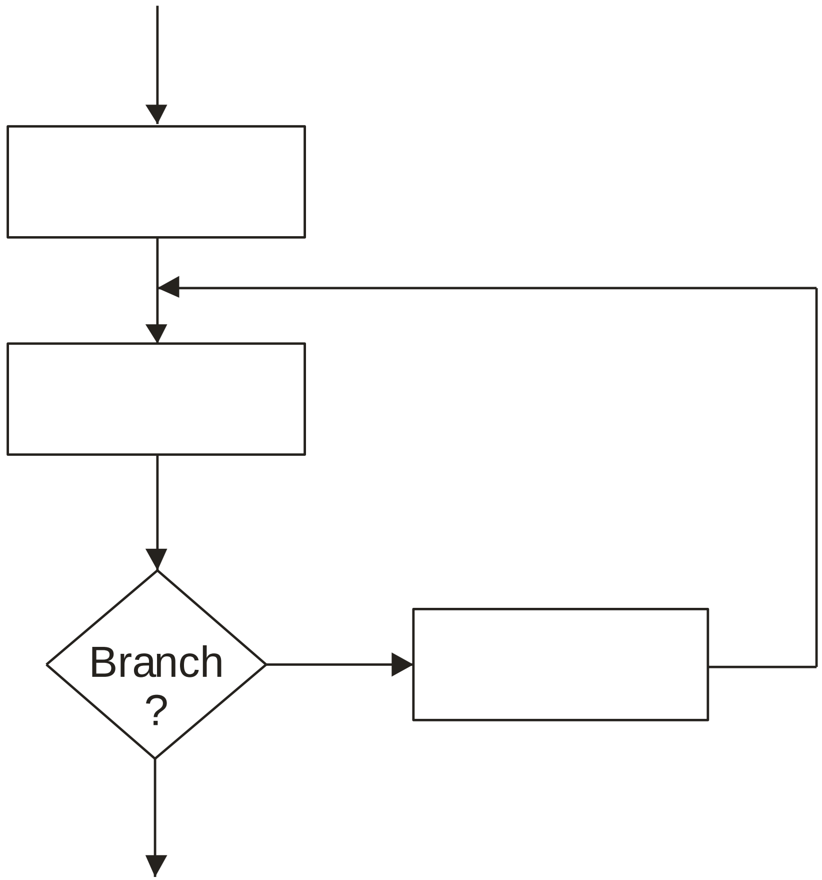
Sensor:	$t_{S\_min} = 5\ \text{ms}$	$t_{S\_max} = 15\ \text{ms}$
Hardware:	$t_{HW\_min} = 1\ \mu\text{s}$	$t_{HW\_max} = 2\ \mu\text{s}$
System software:	$t_{SS\_min} = 16\ \mu\text{s}$	$t_{SS\_max} = 48\ \mu\text{s}$
Application software:	$t_{AS\_min} = 0.5\ \mu\text{s}$	$t_{AS\_max} = 0.5\ \mu\text{s}$
Door drive:	$t_{DD\_min} = 300\ \text{ms}$	$t_{DD\_max} = 500\ \text{ms}$

Now, we can calculate the minimum and maximum values of the composite response time: $t_{\min} \approx 305\ \text{ms},\ t_{\max} \approx 515\ \text{ms}$

The overall response time is dominated by the door drive's response time containing the deceleration time of the moving door blades.

Change in Flow of Control

- In software systems a change in state results in a change in the flow-of-control
- The decision block suggests that the program flow can take alternative paths (see *right*)
- `case`, `if-then`, and `while` statements represent a possible change in flow-of-control
- Invocation of procedures in C represent changes in flow-of-control



Definitions: Event and Release Time

Definition: **Event**

Any occurrence that causes the program counter to change non-sequentially is considered a change of flow-of-control, and thus an event

Definition: **Release Time**

The release time is the time at which an instance of a scheduled task is ready to run, and is generally associated with an interrupt

Taxonomy of Events

- An event can be either synchronous or asynchronous
  - *Synchronous* events are those that occur at predictable times in the flow-of-control
  - *Asynchronous* events occur at unpredictable points in the flow-of-control and are usually caused by external sources
- Moreover, events can be periodic, aperiodic or sporadic
  - A real-time clock that pulses regularly is a *periodic* event
  - Events that do not occur at regular periods are called *aperiodic*
  - Aperiodic events that tend to occur very infrequently are called *sporadic*

Example: Various Types of Events

Type	Periodic	Aperiodic	Sporadic
Synchronous	Cyclic code	Conditional branch	Divide-by-zero (trap) interrupt
Asynchronous	Clock interrupt	Regular, but not fixed-period interrupt	Power-loss alarm

Deterministic Systems

- For any physical system, certain states exist under which the system is considered to be out of control
- The software controlling such a system must therefore avoid these states
- In embedded real-time systems, maintaining overall control is extremely important
- Software control of any real-time system and associated hardware is maintained when the next state of the system, given the current state and a set of inputs, is predictable

Definition: Deterministic System

A system is deterministic, if for each possible state and each set of inputs, a unique set of outputs and next state of the system can be determined

Deterministic Systems Cont'd

- *Event determinism* means the next states and outputs of a system are known for each set of inputs that trigger events
- Thus, a system that is deterministic is also event deterministic
- However, event determinism may not imply determinism
- While it is a significant challenge to design systems that are completely event deterministic, it is possible to inadvertently end up with a system that is non-deterministic
- Finally, if in a deterministic system the response time for each set of outputs is known, then, the system also exhibits *temporal determinism*
- A side benefit of designing deterministic systems is that guarantees can be given that the system will be able to respond at any time, and in the case of temporally deterministic systems, when they will respond

CPU Utilization or Time-Loading Factor

- The final term to be defined is a critical measure of real-time system performance
- Because the CPU continues to execute instructions as long as power is applied, it will more or less frequently execute instructions that are not related to the fulfillment of a specific deadline
- The measure of the relative time spent doing *non-idle processing* indicates how much real-time processing is occurring

Definition: CPU Utilization Factor

> The CPU utilization or time-loading factor, U, is a relative measure of the non-idle processing taking place

CPU Utilization Zones

- A system is said to be time-overloaded if U > 100%
- Systems that are too highly utilized are problematic
  - Additions, changes, or corrections cannot be made to the system without risk of time-overloading
- On the other hand, systems that are not sufficiently utilized are not necessarily cost-effective
  - The system was over-engineered and that costs could likely be reduced with less expensive hardware
- While a utilization of 50% is common for new products, 80% might be acceptable for systems that do not expect growth
- However, 70% as a target for U is one of the potentially useful results in the theory of real-time systems where tasks are periodic and independent

CPU Utilization Zones Cont’d

Utilization %	Zone Type	Typical Application
< 26	Unnecessarily safe	Various
26 – 50	Very safe	Various
51 – 68	Safe	Various
69	Theoretical limit	Embedded systems
70 – 82	Questionable	Embedded systems
83 – 99	Dangerous	Embedded systems
100	Critical	Marginally stressed system
> 100	Overloaded	Stressed system

Calculation of *U*

- *U* is calculated by summing the contribution of utilization factors for each task
- Suppose a system has $n \geq 1$ periodic tasks, each with an execution period of p_i
- If task *i* is known to have a *worst-case* execution time of e_i, then the utilization factor, u_i, for task *i* is

$$u_i = e_i / p_i \qquad (1.1)$$

- Furthermore, the overall system utilization factor is

$$U = \sum_{i=1}^{n} u_i = \sum_{i=1}^{n} e_i / p_i \quad (1.2)$$

- In practice, the determination of e_i can be difficult, in which case estimation or measuring must be used
- For aperiodic and sporadic tasks, u_i is calculated by assuming a worst-case execution period

Example: Calculation of *U*

Suppose, an individual elevator controller in a bank of elevators has the following tasks with execution periods of p_i and worst-case execution times of $e_i, i \in [1,2,3,4]$:

Task 1: Communicate with the group dispatcher.

Task 2: Update the car position information and manage floor-to-floor runs as well as door control.

Task 3: Register and cancel car calls.

Task 4: Miscellaneous system supervisions.

i	e_i	p_i
1	17 ms	500 ms
2	4 ms	25 ms
3	1 ms	75 ms
4	20 ms	200 ms

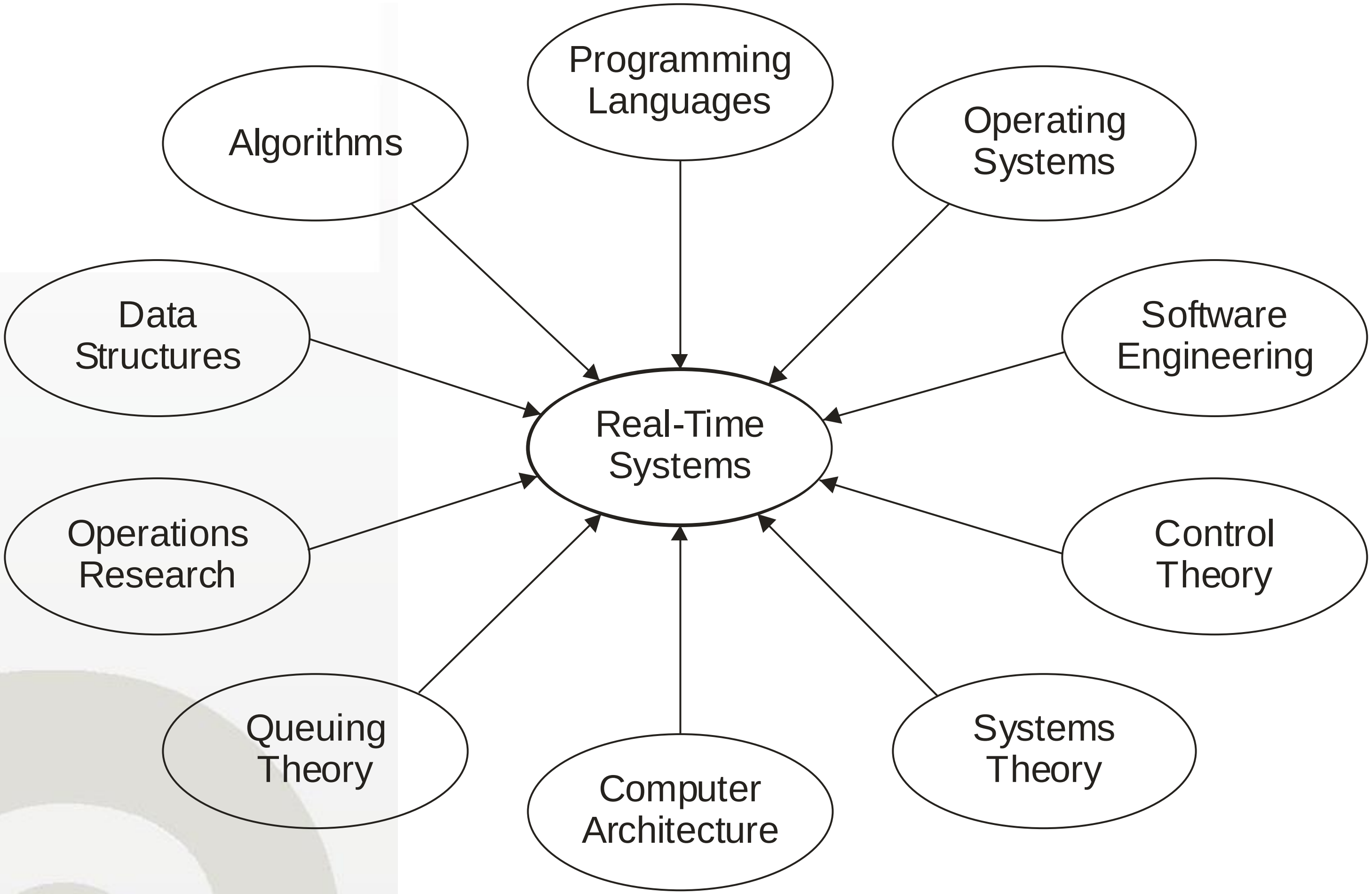
$$U = \sum_{i=1}^{4} e_i/p_i = 0.31$$

31% (Very safe zone)

Usual Misconceptions

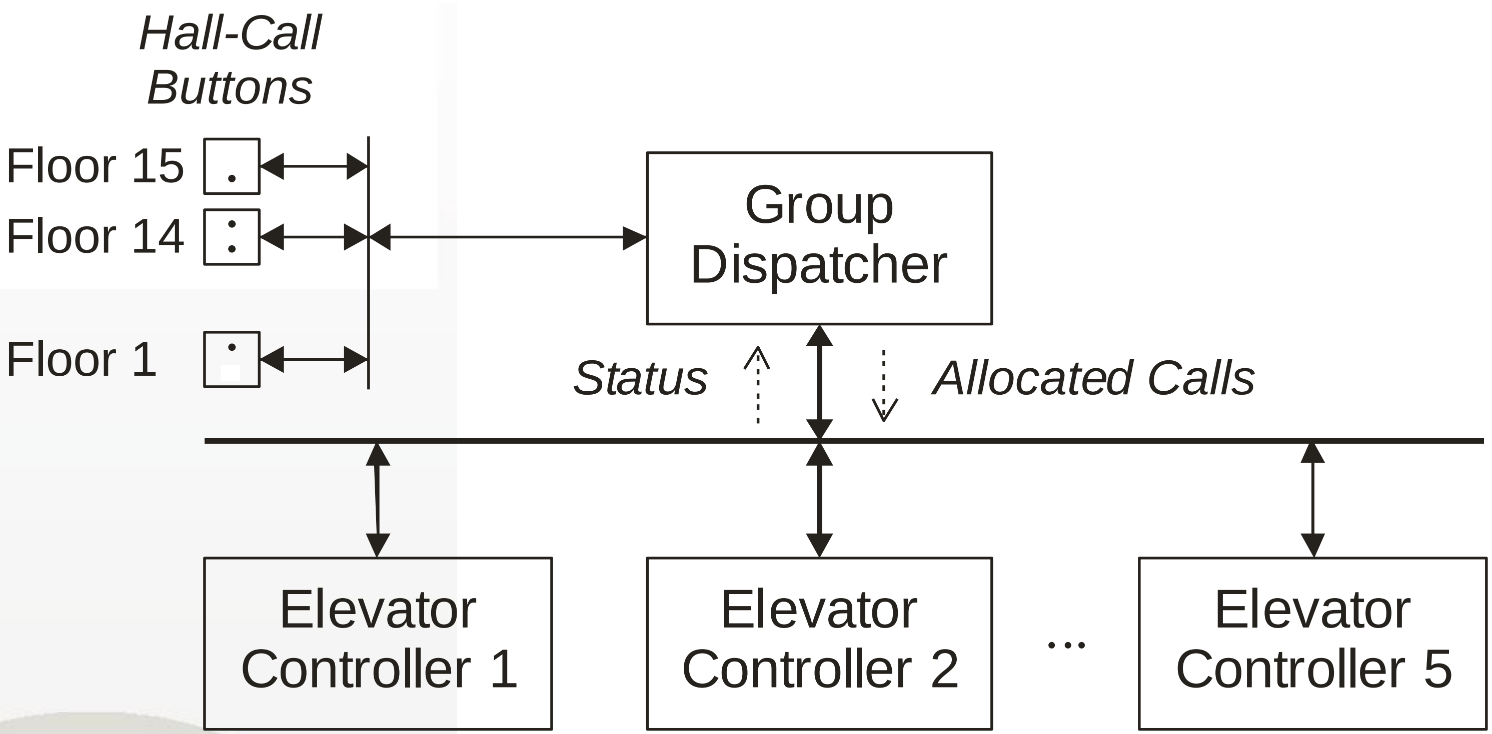
- **Real-time systems are synonymous with "fast" systems**
  - Many (but not all) hard real-time systems deal with deadlines in the tens of milliseconds
- **There are universal, widely accepted methodologies for real-time systems specification and design**
  - There is still no methodology available that answers all of the challenges of real-time specification and design all the time and for all applications
- **There is no more a need to build a real-time operating system, because many commercial products exist**
  - Commercial solutions have certainly their place, but choosing when to use an off-the-shelf solution and choosing the right one are continuing challenges

Rich Variety of "Real-Time" Disciplines



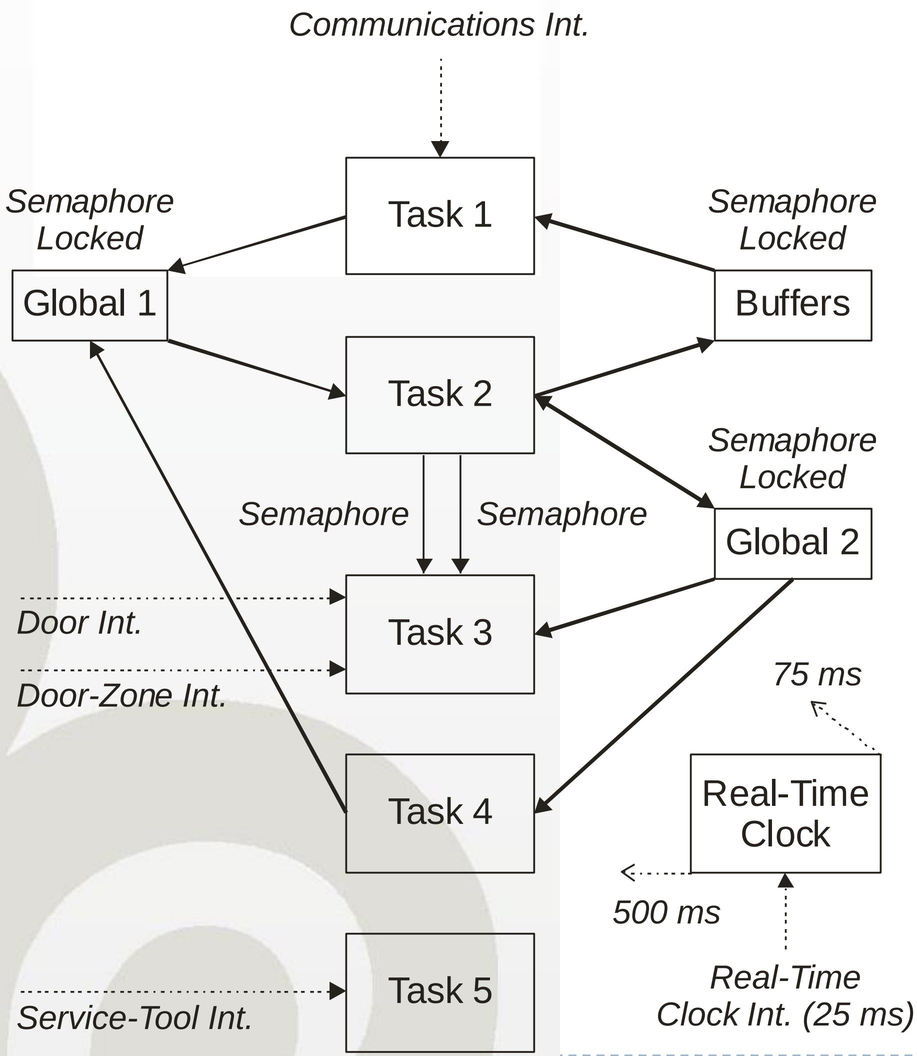
- Real-time systems is a truly multi-dimensional subdiscipline of computer systems engineering

Example: Architecture of an Elevator Bank Control System



- The group dispatcher register and cancel hall calls, and allocates registered calls dynamically to the most suitable elevators depending on their current status (e.g. occupancy, car position, running direction, etc.) with minimum waiting time
- The group dispatcher needs to periodically collect status information from each individual elevator

Example: Real-Time Structure of an Elevator Controller



Task/ Priority	Purpose	Rate
1	Group dispatcher communication. Pass data to Task 2 via global variable Global 1. Task execution time -15 ms	500 ms
2	Update car position, register car calls, determine destination floors Task execution time -17 ms	75 ms
3	Create floor run schedule, control door opening and closing and update displays. Variable execution time It is a Finite State Machine	Aperio -dic
4	Backup services in case of communications failure	500 ms
5	Diagnostics and debugger. Runs when CPU does not have anything more urgent to do	Backg-round

Any Questions?