ECE3140 / CS3420
Embedded Systems

Introduction to Real-Time Scheduling

Prof. José F. Martínez

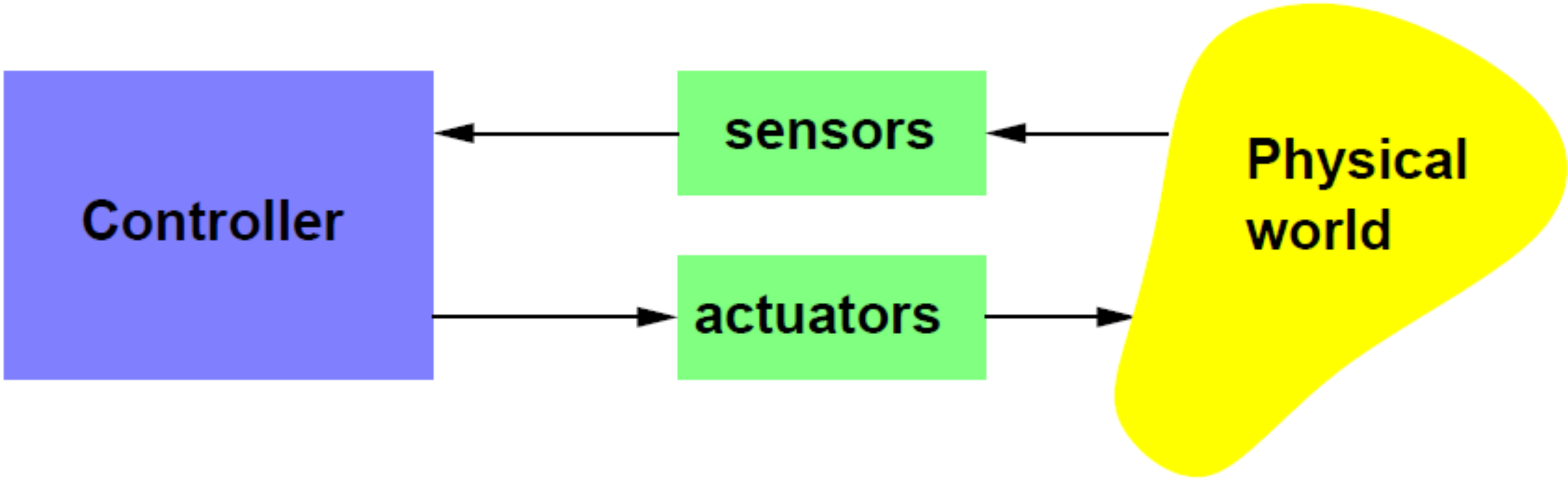
Cornell University

Outline: Real-Time Scheduling

- Real-time systems
- Terminologies
  - Tasks and jobs
  - Parameters
- Scheduling
  - Problem
  - Considerations
  - Types
- Reference
  - Chapter 2 (2.1-2.3), "Hard Real-Time Computing Systems Predictable Scheduling Algorithms and Applications" by Giorgio C. Buttazzo (Free electronic copy through Cornell library)

Introduction to Real-Time

- Many computing systems need to respond to events within precise timing constraints



- Tight interaction between sensing and actuation
  → need predictable timing of operations

- We won't cover the details of how a system is controlled.

Real-Time Systems

Computing system that is able to respond to events within *precise timing constraints* is a real-time system

- Correct operation depends on
  - Usual properties (producing the correct output, etc)
  - Also on the time at which the output is produced

- Some interesting observations:
  - Time between different entities must be synchronized
    - Note: time synchronization is not a simple problem
  - Systems often run multiple tasks with varying criticality levels
  - Real time is not the same as fast!

Importance of Tasks

- Hard real-time tasks: must meet their deadlines. Missing a deadline has a catastrophic effect.
  - Low-level control
  - Sensor-actuator interactions for critical functions
    - Example: airbag, engine control, etc.
- Soft real-time tasks: Missing deadlines is undesirable, but only causes performance degradation
  - Reading keyboard input
  - Displaying a message
  - Updating graphics
- Tasks can be assigned priorities

Performance vs. Predictability

Real-time is different from high performance

- Real time: have to guarantee timing properties
- Performance: minimize average response time

- Source of unpredictability:
  - Architecture: cache, pipelining, . . .
  - Run-time system: scheduling, other tasks, . . .
  - Environment: Bursty information flow, extreme conditions, . . .
  - Input: no explicit notion of time in most languages

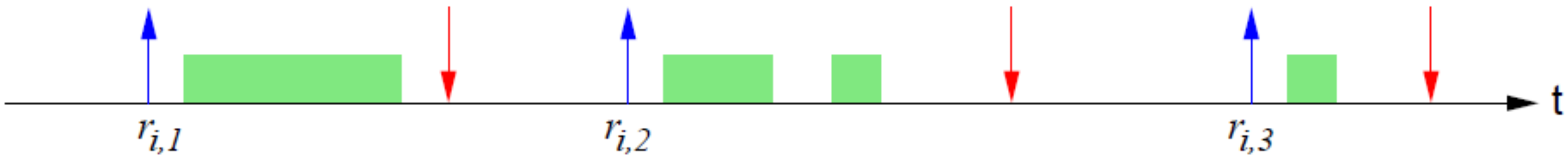
Real-Time Systems Terminology

- A job is a unit of work that is scheduled and executed by the system
- A task is a sequence (possibly infinite) of jobs, which jointly provide some system functions
- A job has:
  - A request time r_i (arrival time)
  - A start time s_i
  - A finishing time f_i
  - A computation/execution time C_i
  - An absolute deadline d_i

Tasks and Jobs

- A single job:

- A task:



Periodic vs. Aperiodic Tasks

- A task can be time-driven (periodic) or event-driven (aperiodic)

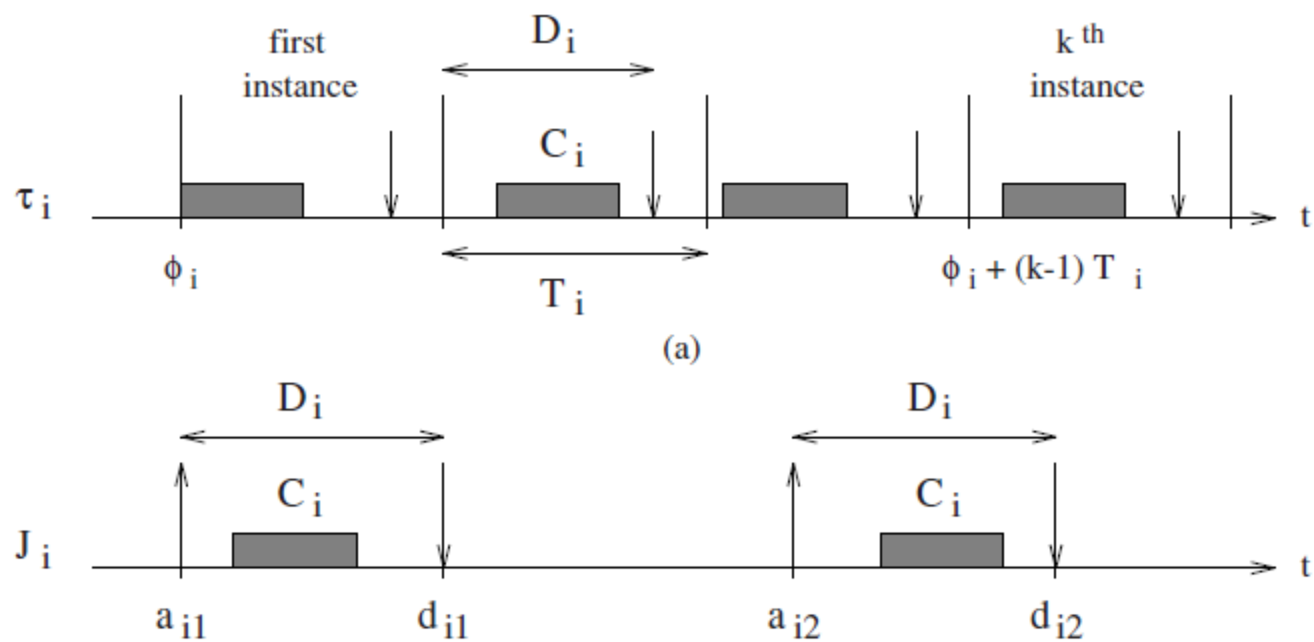
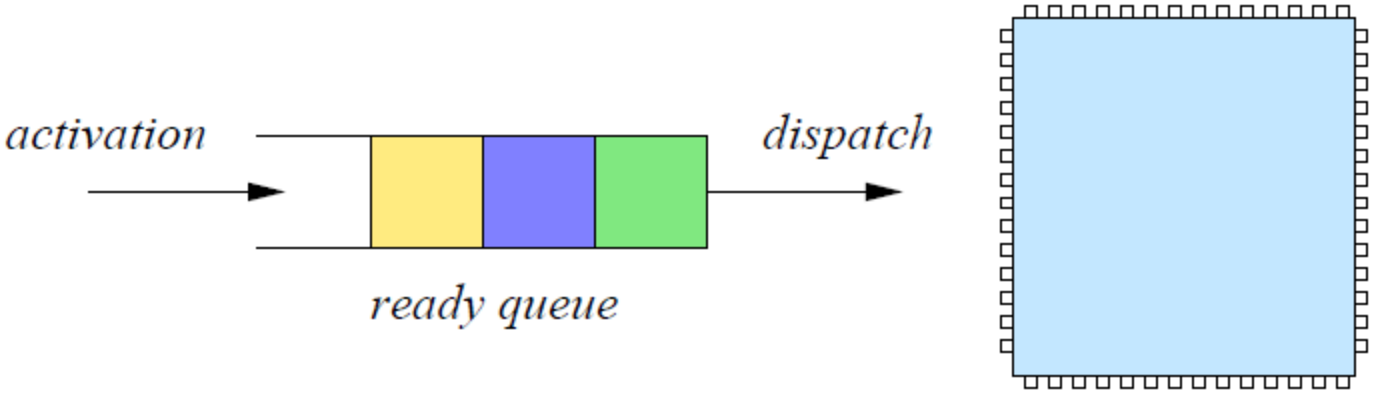


Diagram source: Buttazzo book

Scheduling Algorithm

- Scheduling algorithm: the strategy used to pick a ready task for execution



- Two categories:
  - *Preemptive*: The running task can be temporarily suspended to execute another task
  - *Non-preemptive*: The running task cannot be suspended until completion or until it is blocked

Schedule

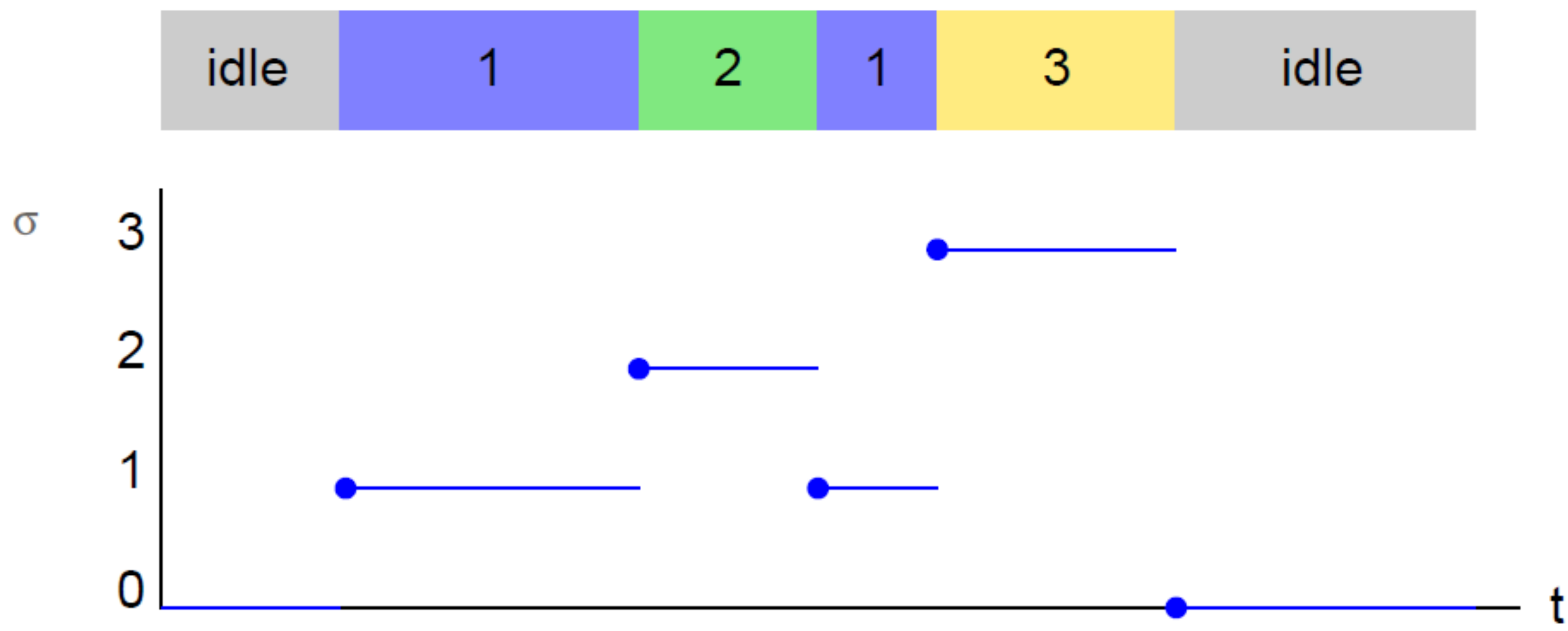
- A schedule is a particular assignment of tasks (jobs) to the processor

Given a set of tasks $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$, a schedule is a mapping $\sigma : \mathbb{R}_{>0} \rightarrow \{0, 1, \ldots, n\}$ such that:

$$\sigma(t) = \begin{cases} k > 0 & \text{if } \tau_k \text{ is running} \\ 0 & \text{if the processor is idle} \end{cases}$$

and in any interval $[t_1, t_2) \in \mathbb{R}_{>0}$ $\sigma(t)$ can only change value a finite number of times.

Scheduling Example

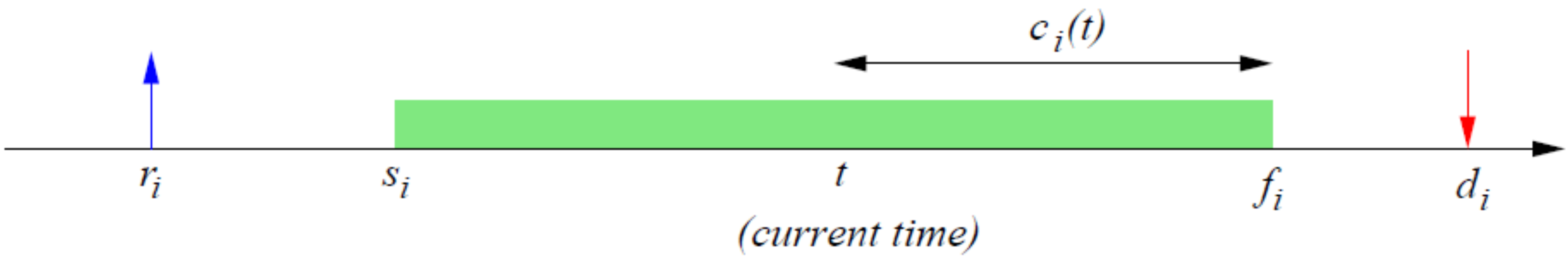


- The points at which σ changes value is where a context switch occurs. Each interval $[t_i, t_{i+1})$ is a time slice.

Feasible Schedule

- A schedule σ is feasible if all tasks are able to complete with their set of constraints
- A set of tasks Γ is schedulable if a feasible schedule exists
- General problem: given Γ, a set of processors *P*, and a set of resources *R*, find an assignment of *P* and *R* that produces a feasible schedule

Derived Parameters

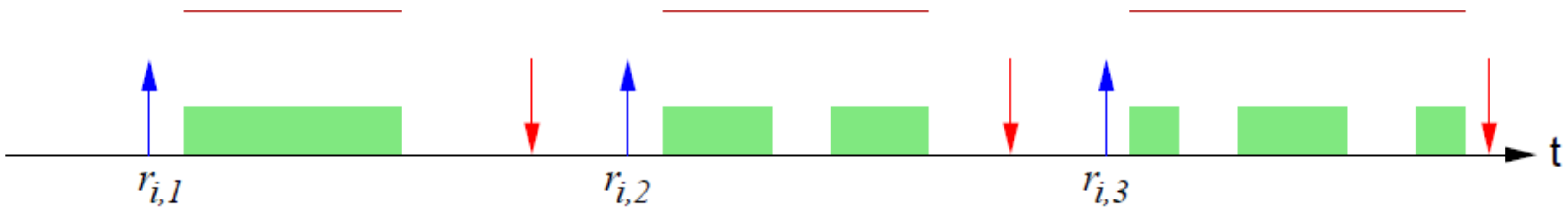


- Lateness: $L_i = f_i - d_i$
  - Tardiness: $\max(0, L_i)$
- Computation/execution time $C_i = f_i - s_i$ (assume continuous)
  - Residual computation time: $c_i(t)$
- Slack: $X_i(t) = d_i - c_i(t)$
- Response time: $R_i = f_i - r_i$

Jitters

- Jitter is the time variation of a periodic event

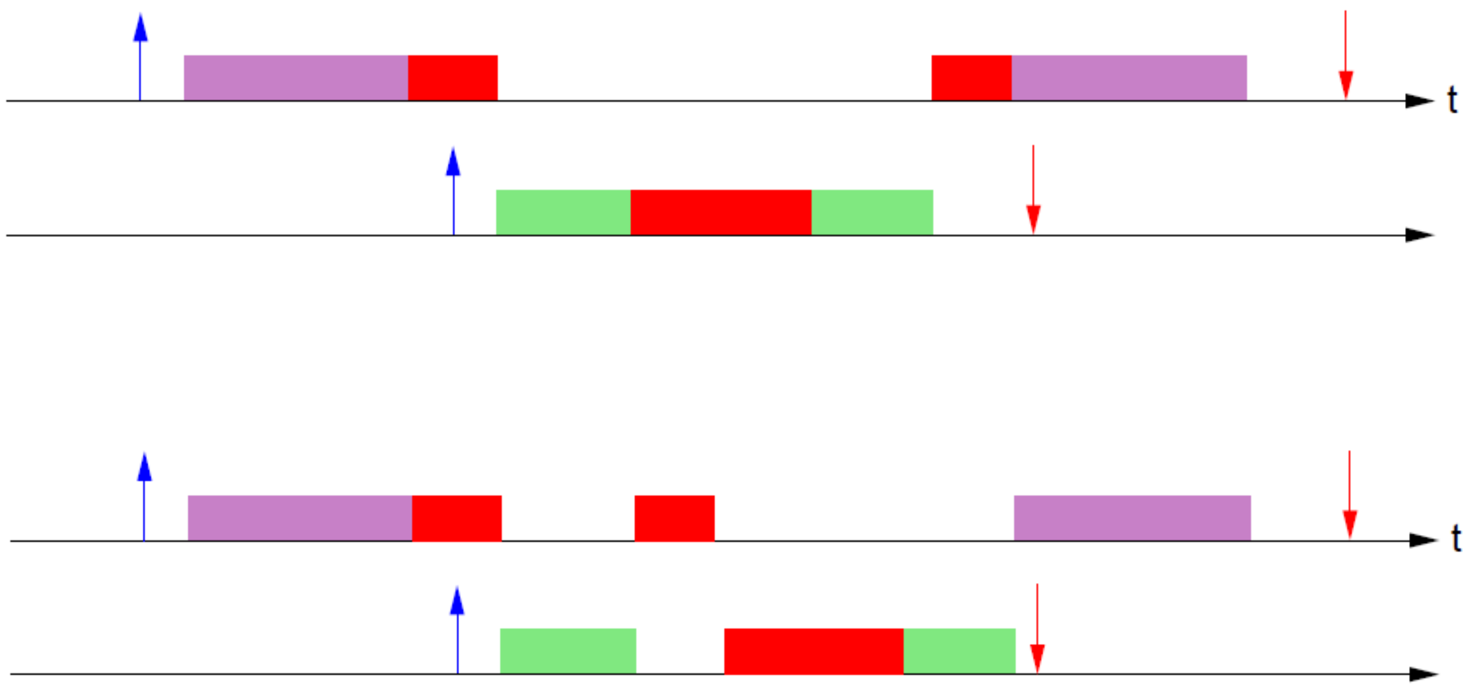
- Example: completion-time jitter



$$\max_k(f_{i,k} - s_{i,k}) - \min_k(f_{i,k} - s_{i,k})$$

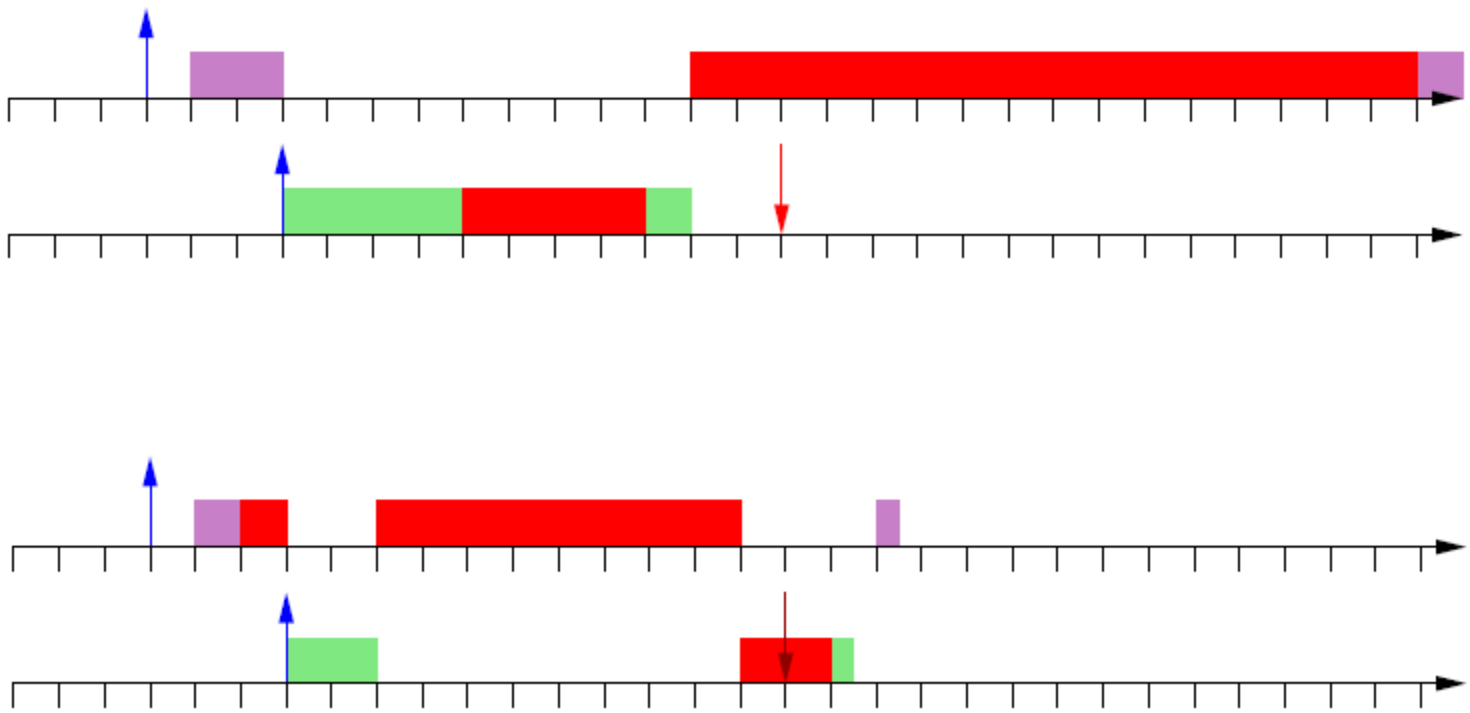
Resource Constraints

- Resources may be limited or even unavailable
- Shared resources may require mutual exclusion



Faster Processor

- Having a faster processor doesn't automatically mean it is easier to meet deadlines



Scheduling Algorithms

- Preemptive or Non-preemptive
- Static or Dynamic
  - Are the scheduling decisions based on parameters that change with time?
  - Fixed-priority vs. dynamic-priority
- Online or Offline
  - Are the decisions made a priori with knowledge of task activations, or are they taken at run time based on the set of active tasks?
- Optimal or Heuristic
  - Can you prove that the algorithm is optimal in terms of a certain criteria or not?