ECE 3140 / CS 3420
EMBEDDED SYSTEMS

LECTURE 17

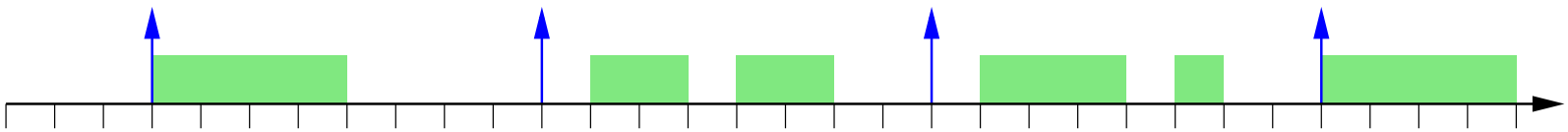
Prof. José F. Martínez

TR 1:25-2:40pm in 150 Olin

Cornell University
Computer Systems Laboratory

Periodic Tasks

Periodic tasks are those that have jobs that repeat at a regular interval in time.



For each periodic task τ_i:

- Each job $\tau_{i,j}$ is activated at $r_{i,j} = (j-1)T_i$
- Each job $\tau_{i,j}$ has a deadline $d_{i,j} = r_{i,j} + D_i$

Timeline Scheduling

Classic technique

- Used for decades in military systems, navigation, and monitoring
- Examples
  - Air traffic control
  - Boeing 777
  - Space Shuttle

TIMELINE SCHEDULING

Approach

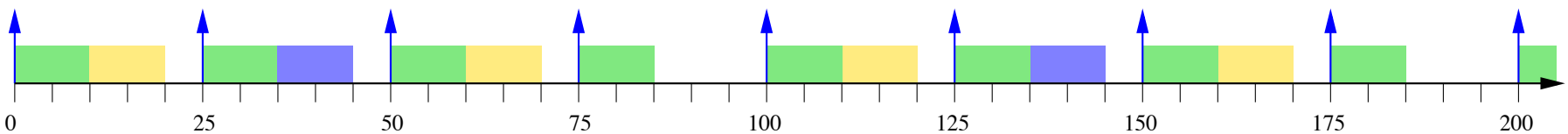
- The time axis is divided into intervals of equal length (slots)
- Each task is *statically* allocated in a slot in order to meet its desired request rate
- Timers are used to activate execution in each slot

TIMELINE SCHEDULING

Example:

Can be implemented via FSM using interrupts

Task	Period	WCET
A	25ms	10ms
B	50ms	10ms
C	100ms	10ms



- Δ = GCD (minor cycle) = 25ms
- T = LCM (major cycle) = 100ms

Timeline Scheduling

- Advantages
  - Simple implementation
  - Low run-time overhead
  - Jitter can be controlled
- Disadvantages
  - Not robust to overloads/overruns
  - Difficult to expand the schedule
  - Not easy to handle aperiodic activities

Timeline Scheduling

What happens if we have an overload?

- If the task continues, there can be a domino effect
- If the task is aborted, the system could be in an inconsistent state

What happens if a task is updated and now takes more time?

- May have to re-do the entire schedule
- Common approach: split the task into two sub-tasks, and then re-build the schedule

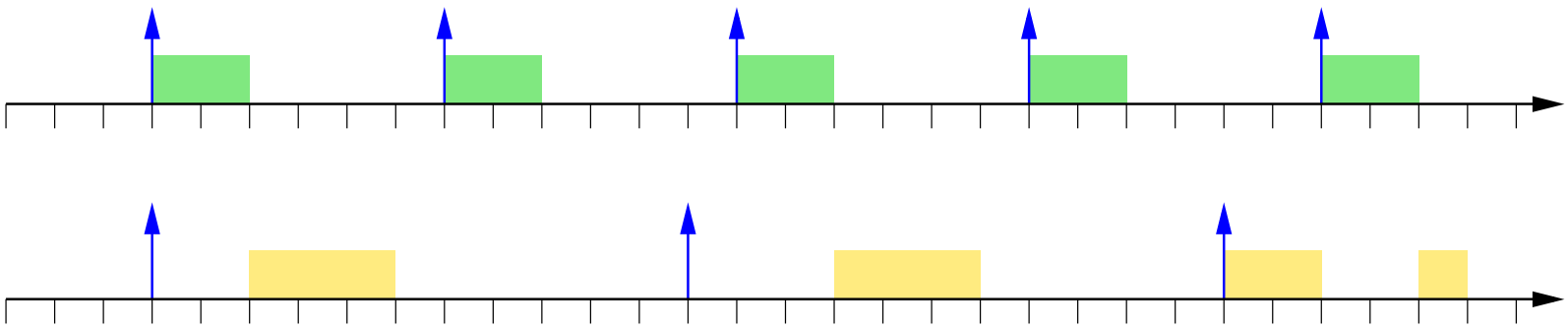
TIMELINE SCHEDULING

Task	Period	Period'	WCET
A	25ms	25ms	10ms
B	50ms	40ms	10ms
C	100ms	100ms	10ms

- Original
  - minor cycle: $\Delta = 25$ms
  - major cycle: $T = 100$ms
- New
  - minor cycle: $\Delta = 5$ms
  - major cycle: $T = 200$ms

Rate Monotonic Scheduling

Rate Monotonic (RM) Scheduling: Each task is assigned a fixed priority proportional to its rate.



Yellow does not run because of "weirdness"

Rate Monotonic Scheduling

How do we determine feasibility?

- Each task uses the processor for a fraction of time:

$$U_i = \frac{C_i}{T_i}$$

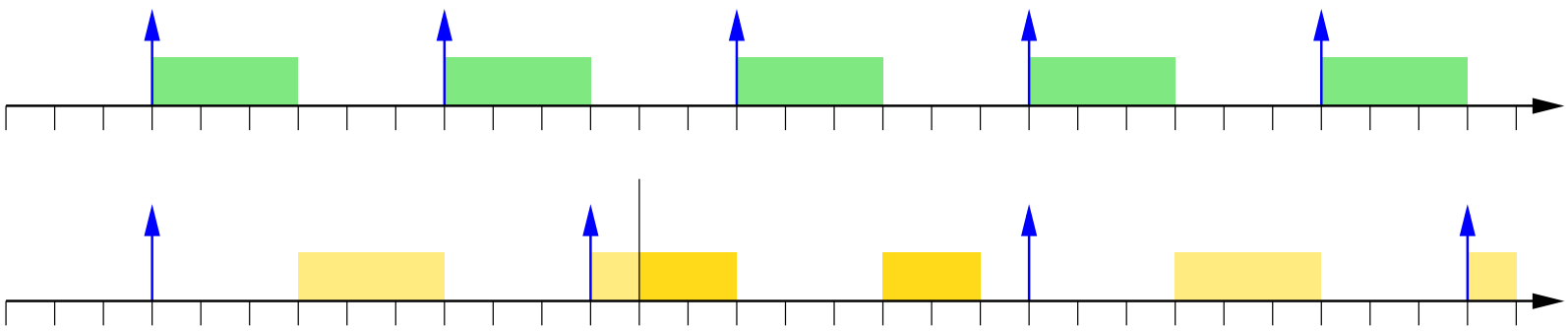
- The total *processor utilization* is given by:

$$U_{cpu} = \sum_i U_i$$

- U_{cpu} measures the processor *load*

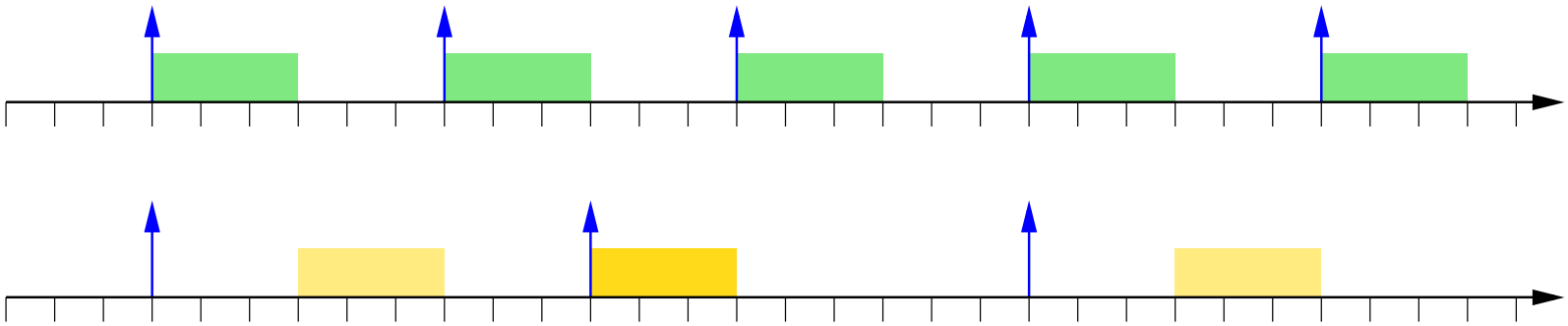
RATE MONOTONIC SCHEDULING

If $U_{cpu} > 1$, the processor is overloaded so the task set cannot be scheduled.

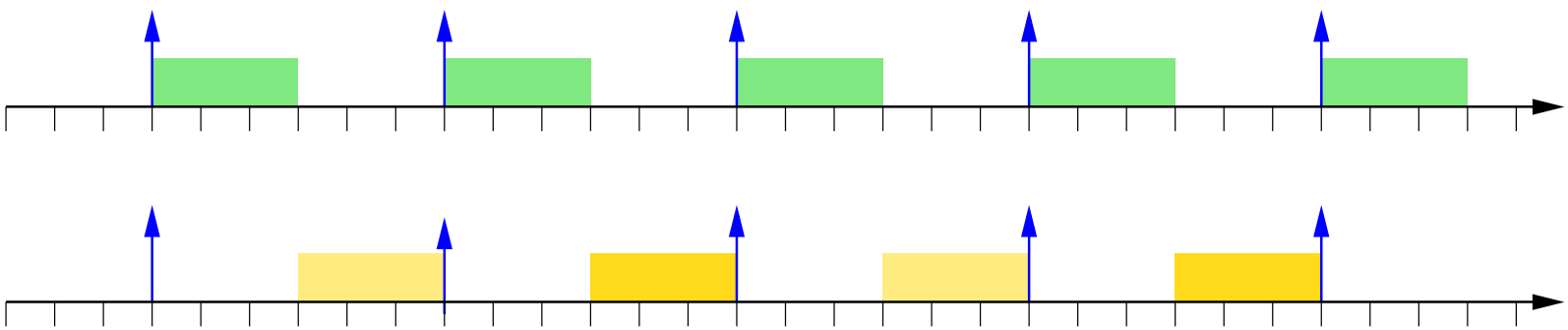


Here, $U_{cpu} < 1$ but the RM schedule is infeasible.

RATE MONOTONIC SCHEDULING



Increasing either C_1 or C_2 causes missed deadlines!



Here $U_{cpu} = 1$ and the schedule is feasible.

RATE MONOTONIC SCHEDULING

- Liu/Layland result (1973): for n periodic tasks, if

$$U_{cpu} \leq n(2^{1/n} - 1)$$

then RM will produce a feasible schedule.

In the limit $n \to \infty$, RHS is ln 2.

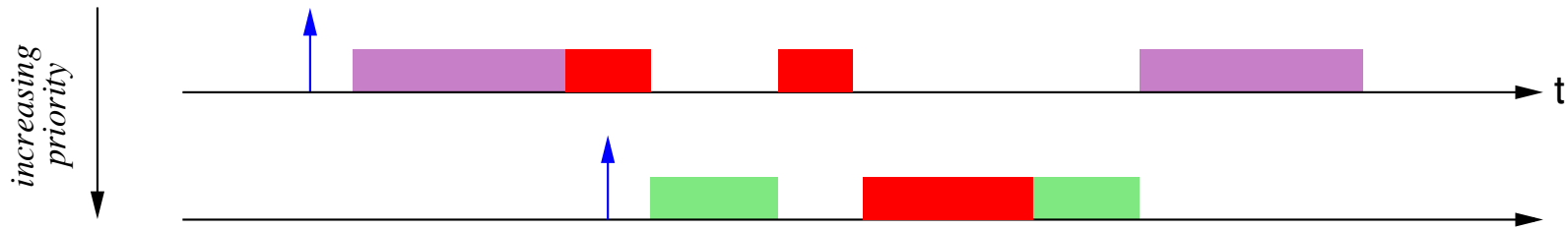
Also:

- If there exists a fixed priority assignment which leads to a feasible schedule, then RM produces a feasible schedule

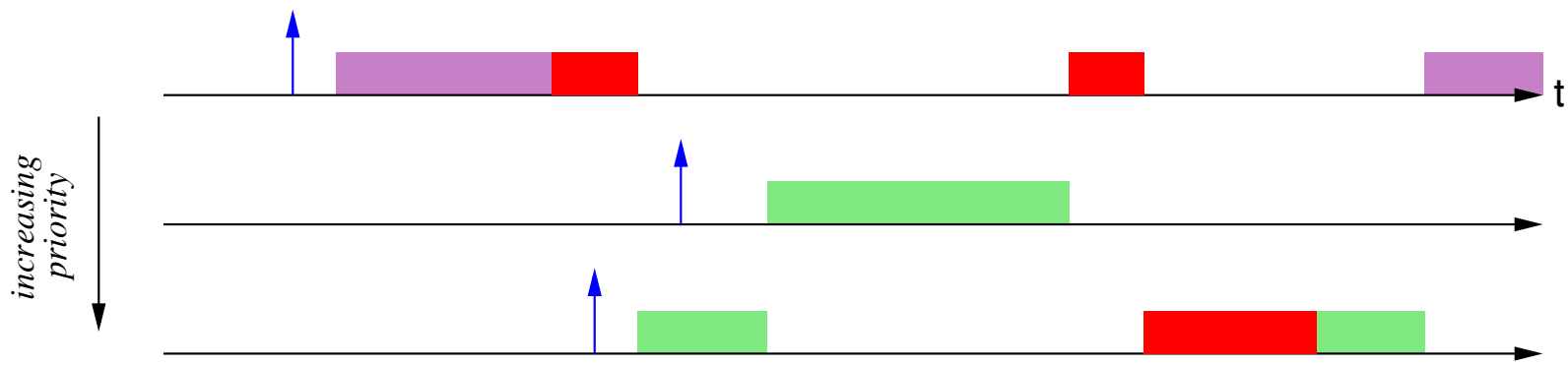
Resource Constraints

Blocking on a critical section can increase the delay...

- How long might we wait? The length of a critical section?



or . . .



Mars Mission

- July 4, 1997: landing on Mars
- July 15, 1997: the system self-resets on a timeout; the robot stops working
- July 17, 1997: the error is identified

Problem: a high-priority task was blocked by a lower priority task for an unbounded interval of time.

Phenomenon is referred to as *priority inversion*.

Resource Access Protocols: modifications to limit delays due to resource contention.