SE3910 – REAL TIME SYSTEMS

Scheduling Theory Part 2

ROADMAP

- Wednesday
  - RTOS Scheduling (Continued)
- Friday
  - Communication between processes
- Monday
  - Gstreamer and OpenCV Introduction
  - Image processing / image handling

Image Stuff!

Prelab ⇒ two videos
⇒ one on Eclipse
⇒ one on the project itself.

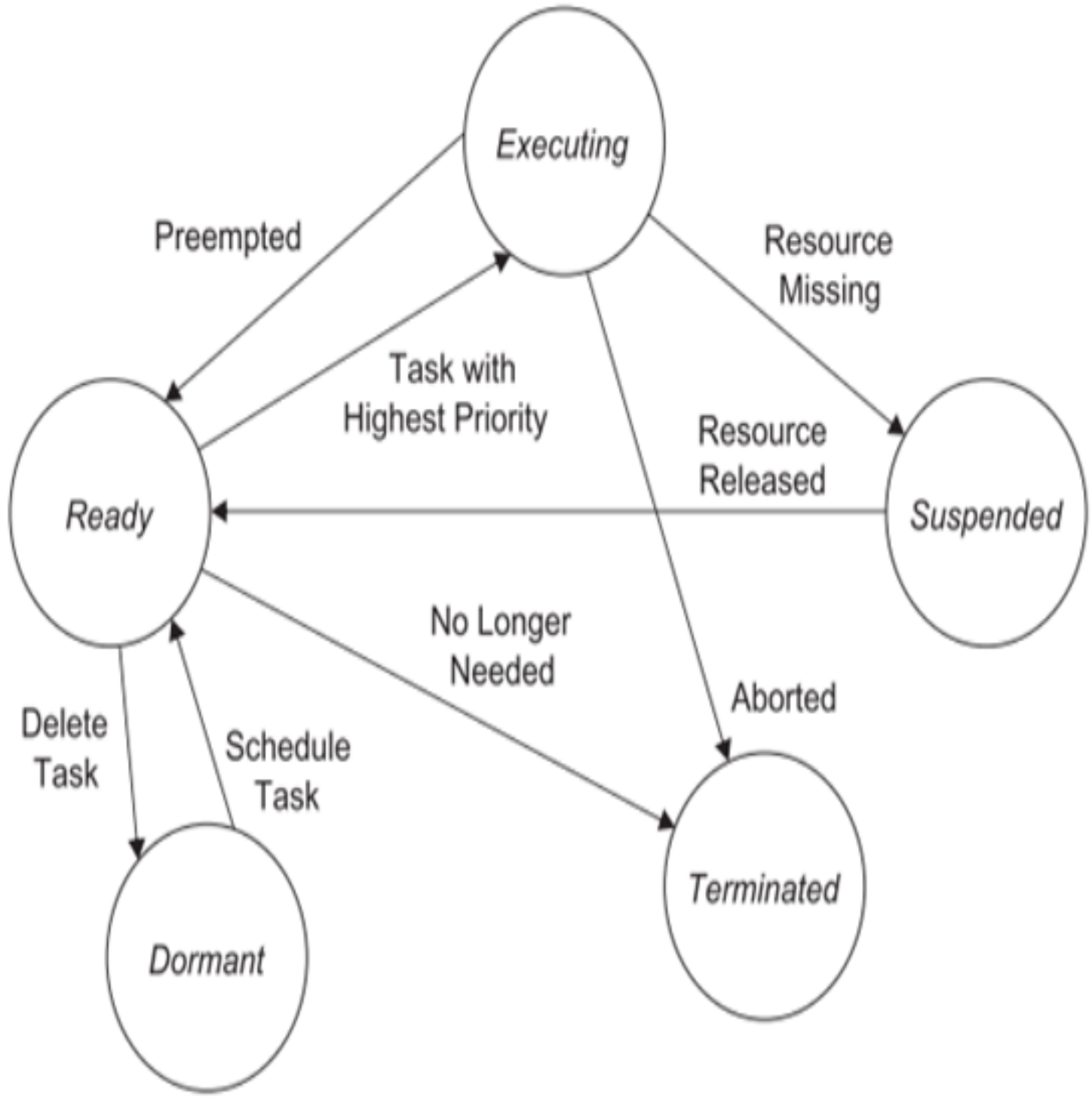
OBJECTIVES

- Draw the task state diagram
- Explain the difference between pre-runtime and runtime scheduling
- Explain the operation of round robin scheduling
- Explain how round robin scheduling may impact latency for a given process
- Explain cyclic code scheduling

"Review"

- Explain Rate Monotonic Analysis Scheduling.

TASK STATE DIAGRAM



SCHEDULING FRAMEWORK

- Pre-runtime ← "Design"
  - Create a feasible schedule offline prior to execution –
  - Requires prediction of the worst case performance for the system
  - Takes into account context switch overhead –
  - Tries to avoid resource conflicts –
- Runtime scheduling
  - Priorities (fixed or dynamic) are assigned and resources are allocated
  - Allows for tasks to be interrupted
  - Allows for resources to be demanded periodically, periodically, or sporadically

MATHEMATICAL DEFINITIONS

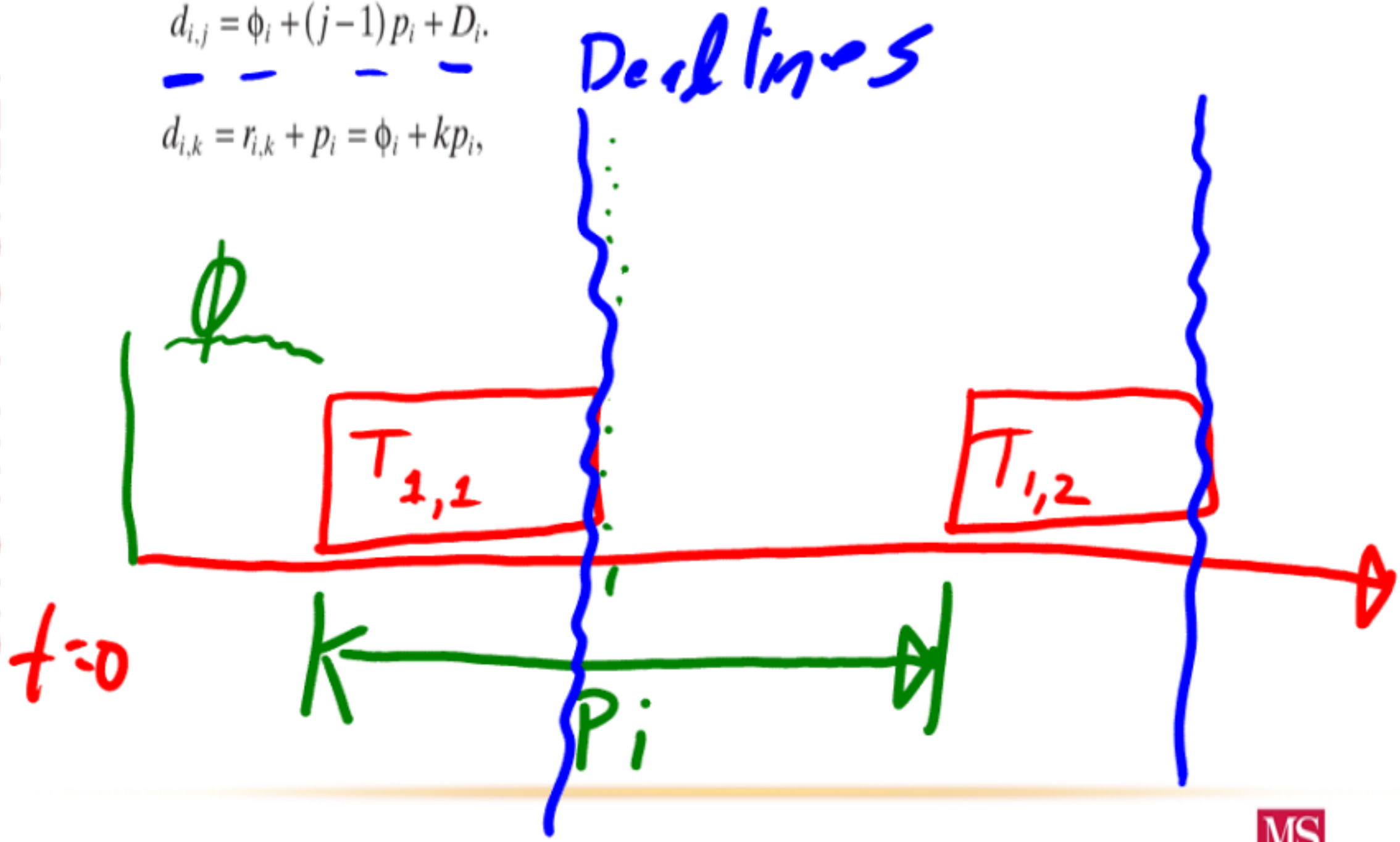
- *Precedence constraints:* Specify if any task needs to precede other tasks.
- *Release time $r_{i,j}$:* The release time of the jth instance of task τ_i.
- *Phase ϕ_i:* The release time of the first instance of task τ_i.
- *Response time:* The time span between the task activation and its completion.
- *Absolute deadline d_i:* The instant by which task τ_i must complete.
- *Relative deadline D_i:* The maximum allowable response time of task τ_i.
- *Laxity type:* The notion of urgency or leeway in a task's execution.
- *Period p_i:* The minimum length of interval between two consecutive release times of task τ_i.
- *Execution time e_i:* The maximum amount of time required to complete the execution of task τ_i when it executes alone and has all the resources it needs.

MATHEMATICAL RELATIONSHIPS SHOWN VISUALLY

$$\phi_i = r_{i,1} \quad \text{and} \quad r_{i,j} = \phi_i + (j-1)\,p_i.$$

$$d_{i,j} = \phi_i + (j-1)\,p_i + D_i.$$

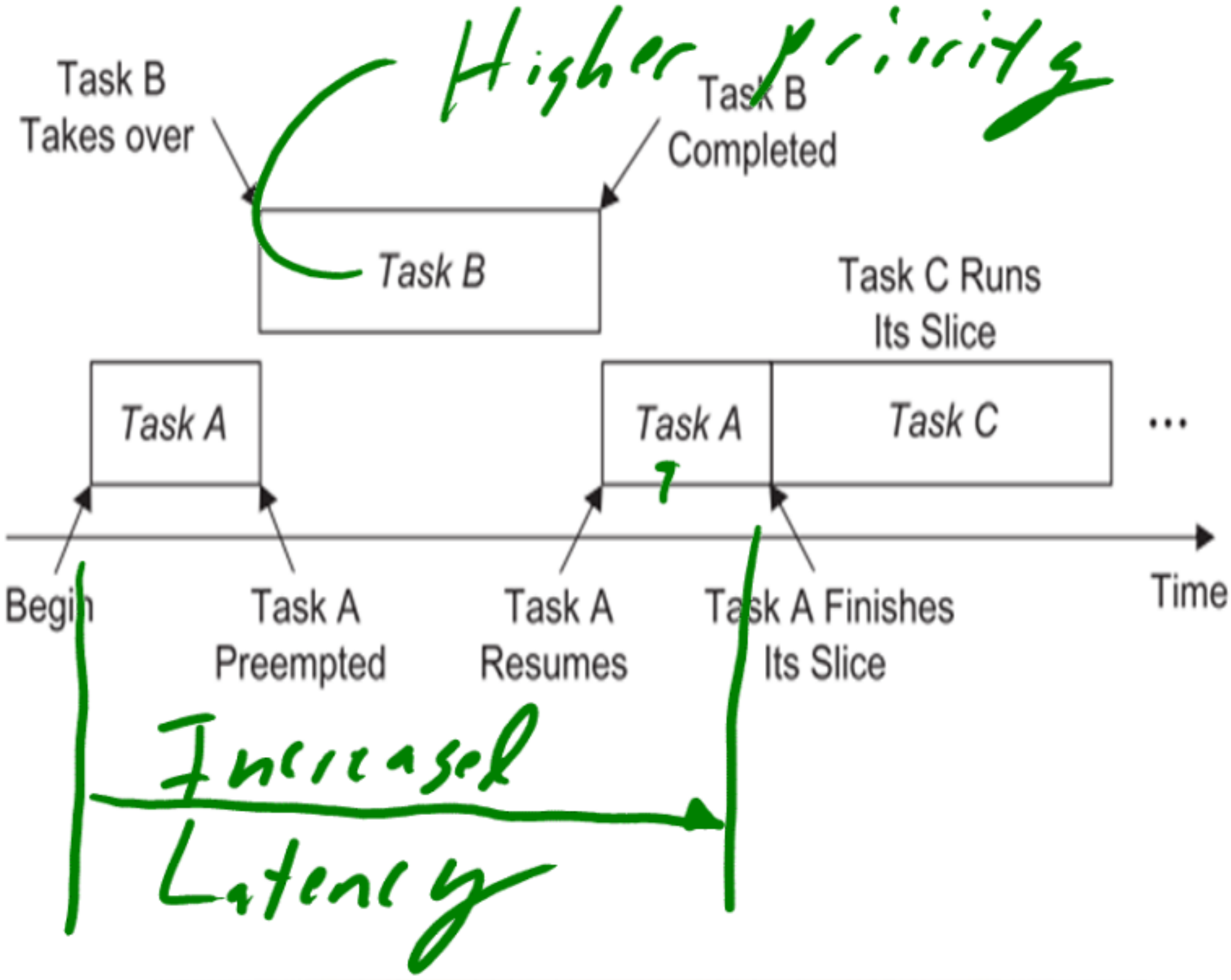
$$d_{i,k} = r_{i,k} + p_i = \phi_i + k p_i,$$



TASK MODEL ASSUMPTIONS

- All tasks in the task set considered are strictly periodic.
- The relative deadline of a task is equal to its period.
- All tasks are independent; there are no precedence constraints.
- No task has any nonpreemptible section, and the cost of preemption is negligible.
- Only processing requirements are significant; memory and I/O requirements are negligible.

ROUND ROBIN SCHEDULING



CYCLIC CODE SCHEDULING

- Scheduling decisions are made periodically rather than arbitrarily
  - Major cycle
    - The minimum time required to execute tasks allocated to the CPU
    - Equal to the lcm of the task periods
  - Frames
    - The locations where scheduling decisions are made
    - No premption within frames

RATE MONOTONIC ANALYSIS

- The priority of each periodic task is fixed relative to other tasks

Theorem: Rate-Monotonic

Given a set of periodic tasks and preemptive priority scheduling, then assigning priorities such that the tasks with shorter periods have higher priorities (rate-monotonic), yields an optimal scheduling algorithm.

EXAMPLE TASK SET

TABLE 3.2. Example Task Set for RM Scheduling

periods

Execution time

τ_i	p_i	e_i	u_i
τ_1	4	1	0.25
τ_2	5	2	0.4
τ_3	20	5	0.25

Highest Priority

Lowest priority

- Calculation: What is the processor utilization for this problem?

Utilization

- Based on RMA, what order will they execute and how will they execute?

What are the priorities?

$U = 90\%$ utilization

MSOE UNIVERSITY

TABLE 3.2. Example Task Set for RM Scheduling

τ_i	p_i	e_i	u_i
τ_1	4	1	0.25
τ_2	5	2	0.4
τ_3	20	5	0.25

- Max Utilization to be schedulable

$$U \leq n * (2^{\frac{1}{n}} - 1)$$

RMA BOUND

$$\lim_{n \to \infty} n * (2^{\frac{1}{n}} - 1) = ln2 \approx 0.69$$

THE CHART FROM LAST WEEK

TABLE 1.3. CPU Utilization (%) Zones

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 systems
>100	Overloaded	Stressed systems