Real Time Scheduling
Basic Concepts

Radek Pelánek

Tento projekt je spolufinancován Evropským sociálním fondem a státním rozpočtem České republiky.

EVROPSKÁ UNIE

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

OP Vzdělávání pro konkurenceschopnost

UNIVERSITAS MASARYKIANA BRUNENSIS

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Model of RT System

- abstraction
- focus only on timing constraints
- idealization (e.g., zero switching time)

Basic Notions

task	process, something that needs to be done (sequentially)
job	a) same as task, b) task is a series of jobs (i.e., periodic task)
resource	active (processor), passive (e.g., mutex, file, database lock)
schedule	assignments of resources to tasks
scheduling algorithm	creates schedule

Basic Notions II

feasible schedule	all tasks can be completed according to a set of specified constraints
schedulable set of tasks	there exists at least one algorithm that can produce feasible schedule

Model of Execution

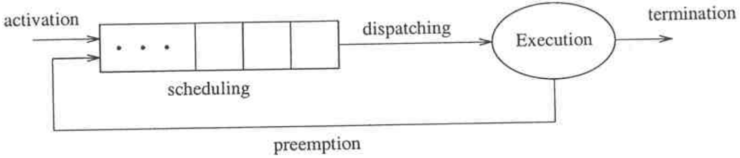


Figure 2.1 Queue of ready tasks waiting for execution.

Preemption

Preemption

temporarily interrupting a task (without requiring its cooperation), with the intention of resuming the task at a later time

Why preemption?

Preemption

Reasons for preemption (examples):

- different levels of criticality (e.g., brakes control vs. radio tuning)
- more efficient schedules

A Simple Case

- set of tasks $J = \{J_1, \ldots, J_n\}$
  1 processor
  no other resources
- schedule: function $\sigma : R^+ \to N$
- $\sigma(t) = k, k > 0$ means that the task J_k is active at time t
- $\sigma(t) = 0$ means that the processor is idle

Example of a Schedule

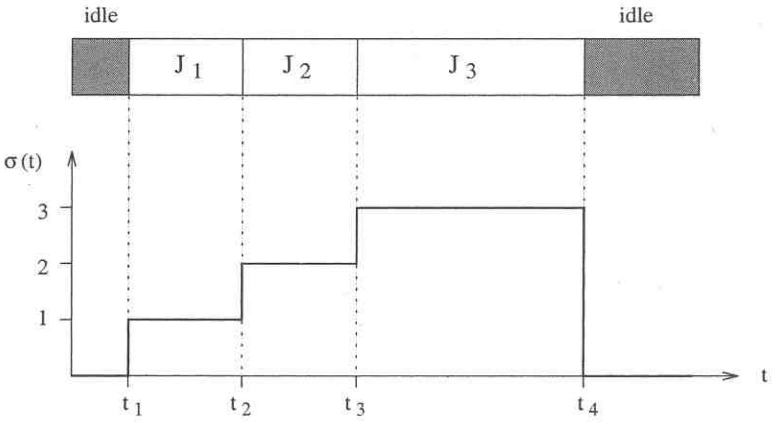


Figure 2.2 Schedule obtained by executing three tasks J_1, J_2, and J_3.

Example of a Preemptive Schedule

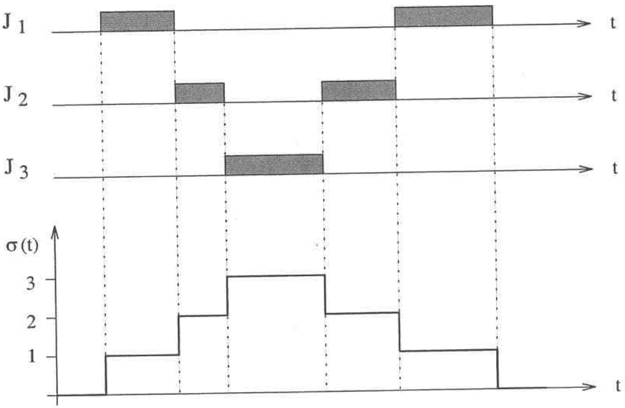


Figure 2.3 Example of a preemptive schedule.

Basic Parameters

arrival time	a_i time when a task becomes ready for execution also denoted release time, request time: r_i
computation time	C_i time necessary for completion of a task
absolute deadline	d_i time before which a task should be completed
start time	s_i time at which a task starts its execution
finishing time	f_i time at which a task finishes its execution
criticality	typically hard/soft

Illustration of Timing Parameters

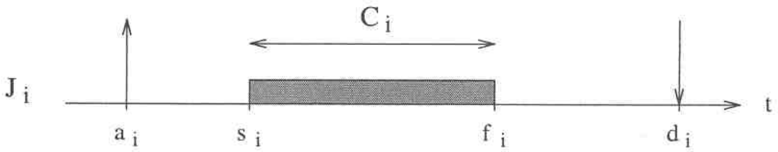


Figure 2.4 Typical parameters of a real-time task.

Derived Parameters

- **relative deadline** $D_i = d_i - a_i$
- **response time** $R_i = f_i - a_i$
- **lateness** $L_i = f_i - d_i$ delay of a task (can be negative)
- **tardiness** (exceeding time) $E_i = max(0, L_i)$
- **slack time** (laxity) $X_i = d_i - a_i - C$ maximum time a task can be delayed on its activation to complete within deadline

Example

	J_1	J_2	J_3
a_i	1	2	2
C_i	2	2	3
d_i	6	5	11

- determine slack time of each job
- find a feasible schedule (with, without preemption)
- determine response time, lateness of each job (in your schedule)

Periodicity

periodic infinite sequence of identical *jobs*, activated at a constant rate; denoted τ_i

- *phase* ϕ_i activation time of the first job
- *period* T, activation time of the k-th job is $\phi_i + (k-1) \cdot T_i$

aperiodic a single job, or a sequence without regular activation, denoted J_i

- *sporadic*: consecutive jobs separated by minimum inter-arrival time

Periodicity: Examples

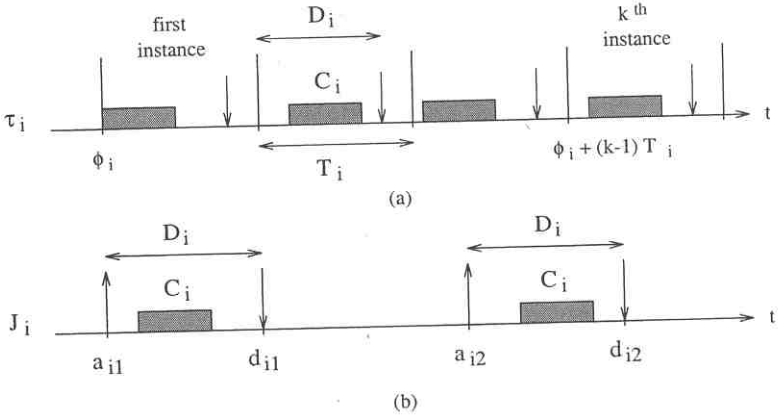


Figure 2.5 Sequence of instances for a periodic (a) and an aperiodic task (b).

Precedence Relations

- used to capture dependencies among tasks
- described by precedence graph
- $J_a < J_b$ means J_a is a predecessor of J_b
- $J_a \rightarrow J_b$ means J_a is an immediate predecessor of J_b

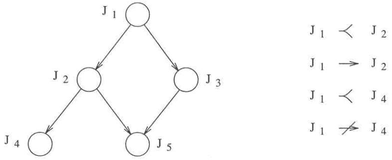


Figure 2.6 Precedence relations among five tasks.

Resources

resource
: something needed to advance execution of a task

shared resource
: resource used by several tasks

mutually exclusive resource
: shared resource that can be used by only one task at a time

critical section
: piece of code executed under mutual exclusion constraint

Blocking on Shared Resources

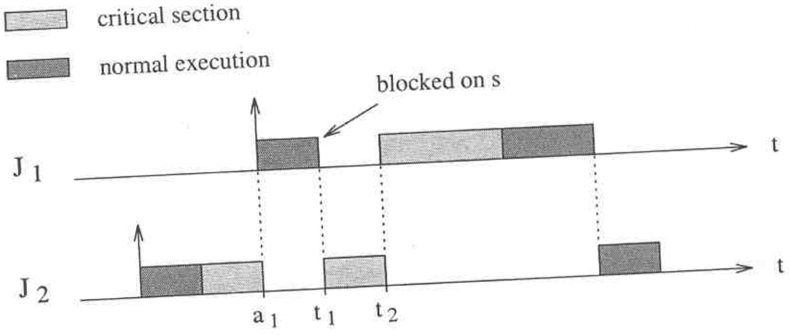


Figure 2.10 Example of blocking on a mutually exclusive resource.

Task Status

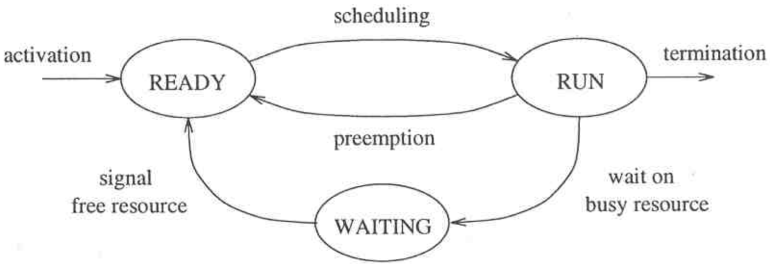


Figure 2.11 Waiting state caused by resource constraints.

General Problem

Input

- set of tasks $J = \{J_1, \dots, J_n\}$
- set of processors $P = \{P_1, \dots, P_m\}$
- set of resources $R = \{R_1, \dots, R_r\}$
- constraints: timing, precedence, ...

Output

Assignment of processor P and resources R to tasks from J such that the given constraints are satisfied.

In general NP-complete.

Example 1

5 periodic tasks, relative deadline D_i = period T_i, phase of all tasks is 0

	A	B	C	D	E
T_i	2	2	2	8	8
C_i	1	1	1	6	6

Is there a feasible schedule on three processors (with preemption, without preemption).

Example 2

9 tasks, arrival time 0 , deadline 12, computation time:

	A	B	C	D	E	F	G	H	J
C_i	3	2	2	2	4	4	4	4	9

precedence constraints: $A \rightarrow J$; $D \rightarrow E, F, G, H$

- Is there a feasible schedule on three processors (without preemption)?

Example 2

9 tasks, arrival time 0 , deadline 12, computation time:

	A	B	C	D	E	F	G	H	J
C_i	3	2	2	2	4	4	4	4	9

precedence constraints: $A \to J$; $D \to E, F, G, H$

- Is there a feasible schedule on three processors (without preemption)?
- Assume that jobs have priorities $p_A > p_B > \ldots > p_J$. What is the schedule based on priorities? Are all deadlines met?

Preemption

preemptive running task can be interrupted

non-preemptive task, once started, is executed by processor until completion

Static vs Dynamic

static scheduling decision based on fixed parameters, assigned before their activation
typical example: “highest priority” scheduling

dynamic scheduling decisions based on dynamic parameters that may change during system evolution
typical example: earliest deadline scheduling

Off line vs On line

off line
: the schedule is generated for the entire task set before the task activations (must be static)

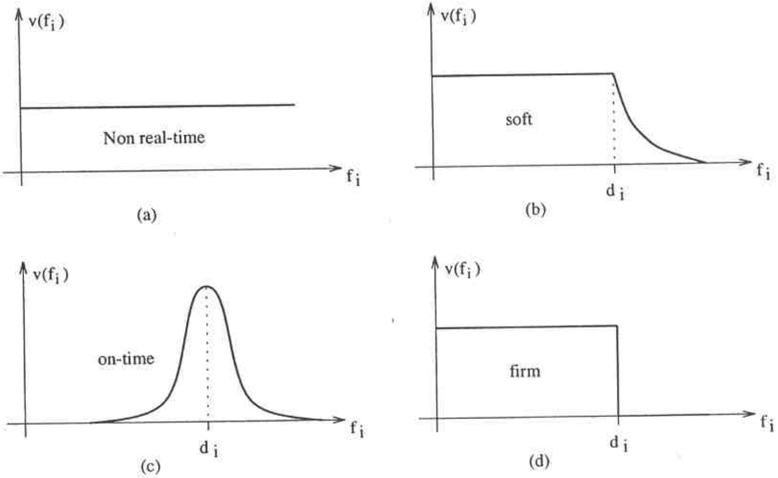
on line
: scheduling decisions are taken at runtime every time a new task enters the system (may be static or dynamic)

Optimal vs Heuristic

optimal algorithm minimizes a given cost function; if no cost function is given then it always finds a feasible schedule if there exists one

heuristic algorithm tries to find feasible schedule; no guarantees of optimality

Cost Functions for Tasks



Cost Functions for Schedule

average response time

$$\frac{1}{n}\Sigma_{i=1}^{n}(f_i - a_i)$$

total completion time

$$max_i(f_i) - min_i(a_i)$$

maximum lateness

$$max_i(f_i - d_i)$$

number of late tasks

Example: Different Cost Functions

All jobs have arrival time 0.

	J_1	J_2	J_3	J_4	J_5
C_i	6	4	4	3	9
d_i	3	8	13	16	24

Find a schedule which:

1. minimizes the number of late tasks
2. minimizes the maximum lateness

Example: Different Cost Functions

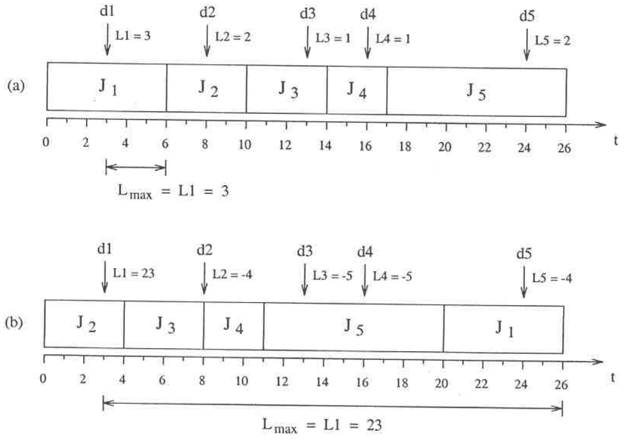


Figure 2.14 The schedule in a minimizes the maximum lateness, but all tasks miss their deadline. The schedule in b has a greater maximum lateness, but four tasks out of five complete before their deadline.

Outlook

- we will study algorithms for specific scheduling problems
- given: type of tasks, type of schedule, cost function
- find: scheduling algorithm