Real-time Systems

Week 10:
Handling overload condition

Ngo Lam Trung

Dept. of Computer Engineering

Contents

- Introduction
- Types of overload
- Strategies to handle overload

Overload condition

- Critical situations in which the computational demand requested by the task set exceeds the processor capacity

  ➔not all tasks can complete within deadlines.

- Reasons
  - Bad system design
  - Simultaneous arrival of events
  - Unpredicted runtime condition: system malfunction, exception,…
- Problem:
  - Which task will miss deadline?

Load definition

- For periodic task set with $D_i = T_i$

$$\rho = U = \sum_{i=1}^{n} \frac{C_i}{T_i}$$

- For dynamic task set, load is calculated for time slices

$$\rho(t_a, t_b) = \max_{t_1, t_2 \in [t_a, t_b]} \frac{g(t_1, t_2)}{t_2 - t_1}$$

- Practical load definition: time slice (t, di)

$$\rho_i(t) = \frac{\sum_{d_k \leq d_i} c_k(t)}{(d_i - t)}$$

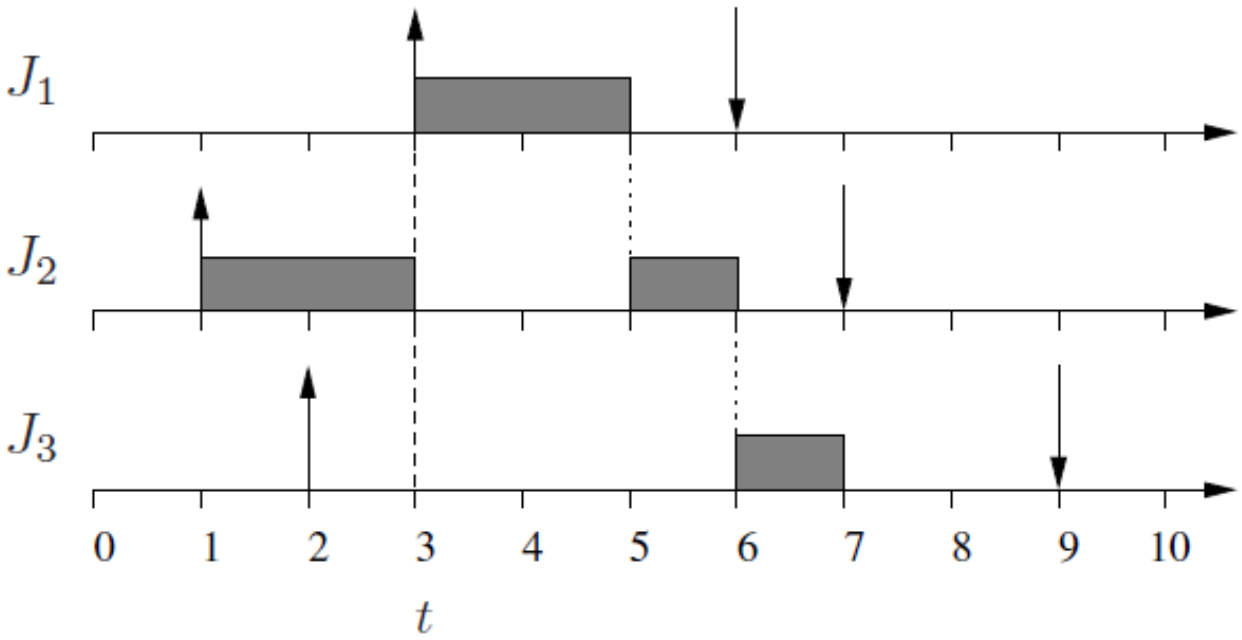
$$\rho(t) = \max_i \rho_i(t)$$

Load definition

- Calculate the load at $t = 3$

$$\rho_i(t) = \frac{\sum_{d_k \le d_i} c_k(t)}{(d_i - t)}$$

$$\rho(t) = \max_i \rho_i(t)$$



$\rho_1(t) = 2/3$

$\rho_2(t) = 3/4$

$\rho_3(t) = 4/6$

$\rho(t) = 3/4$

Overload condition

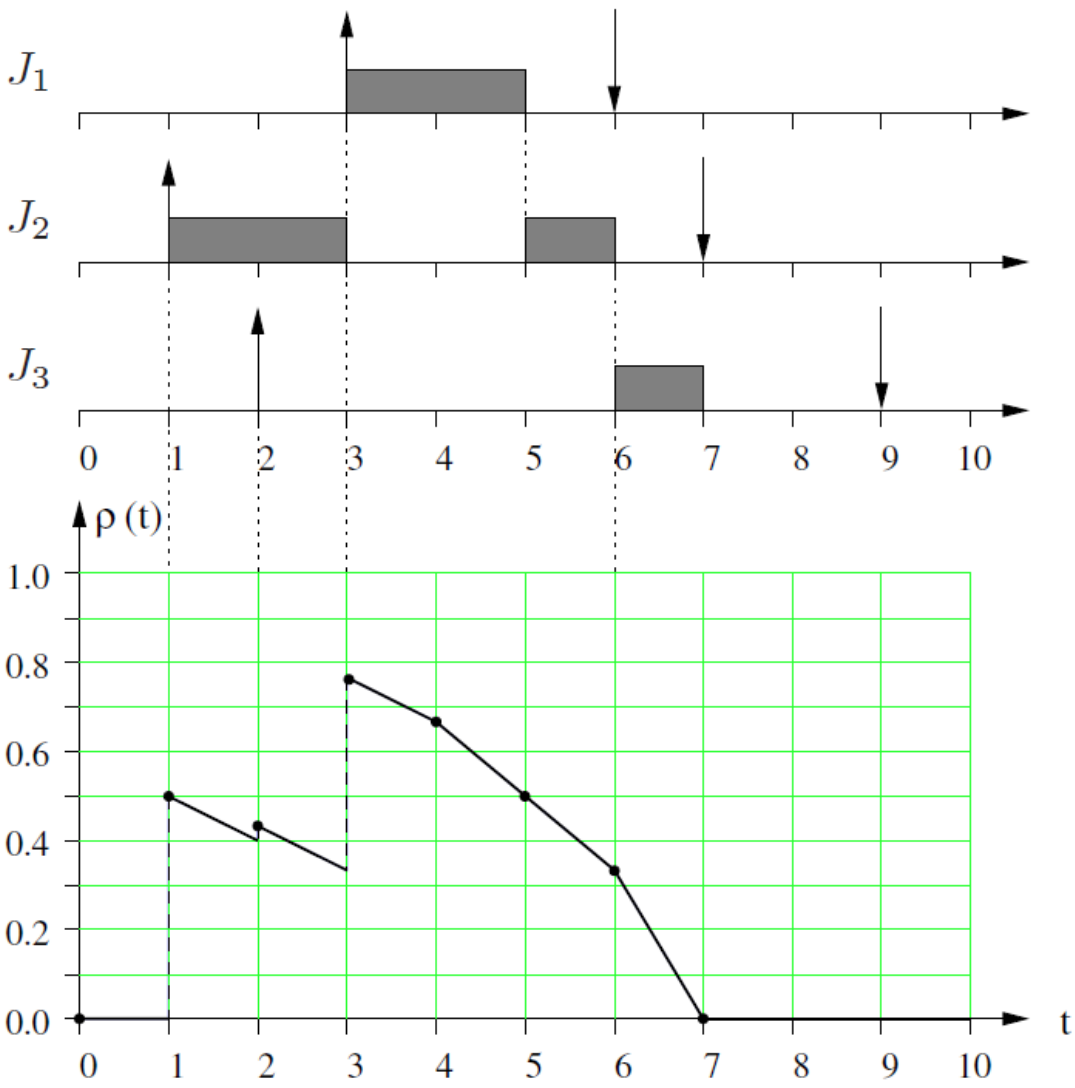
- Overload: computation time demand exceed processing time
- Transient overload

$$\overline{\rho} \leq 1$$

$$\rho^{max} > 1$$

- Permanent overload

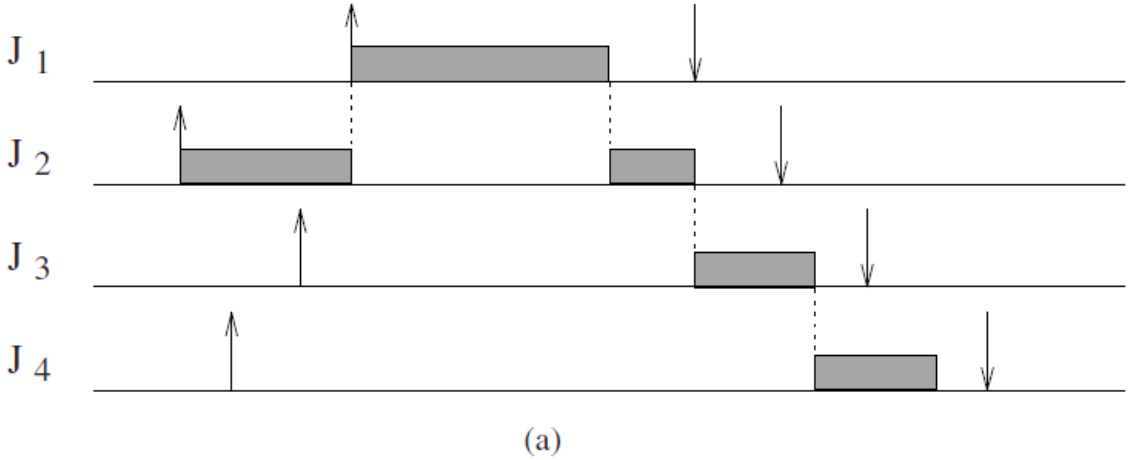
$$\overline{\rho} > 1$$



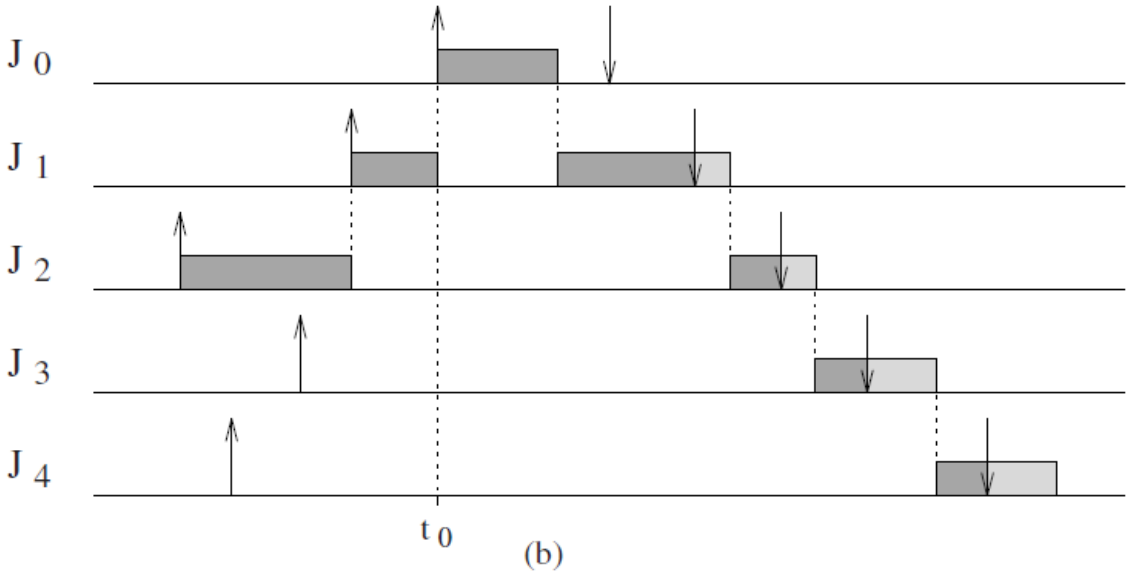
System load over time

Transient overload from aperiodic tasks

a. Normal condition



b. Domino effect

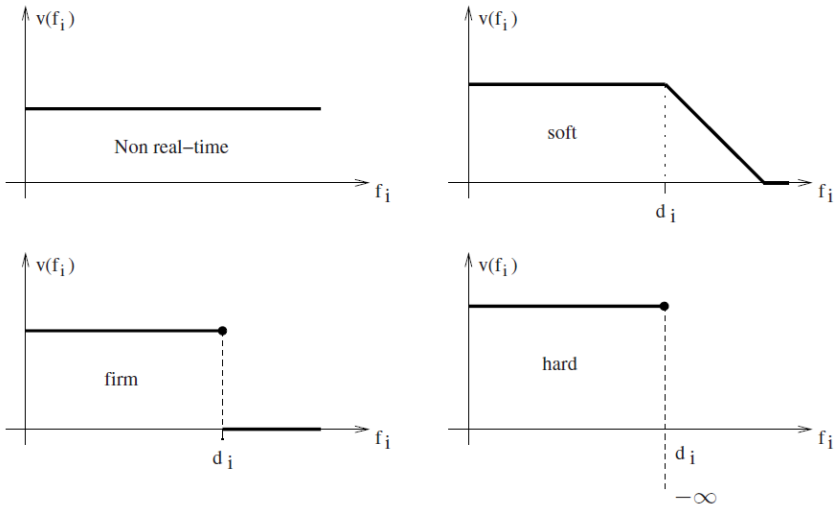


Handling aperiodic overload

- Given a set of *n* jobs $J_i(C_i, Di, Vi)$,
  - Ci: worst case computation time
  - Di: relative deadline
  - Vi: the task's value when task finishes on-time
  - v_i: value function over finishing time
- Cumulative value of algorithm A: $\Gamma_A = \sum_{i=1}^{n} v(f_i)$

Scheduling problem under overload:

Find algorithm A to maximize Γ_A

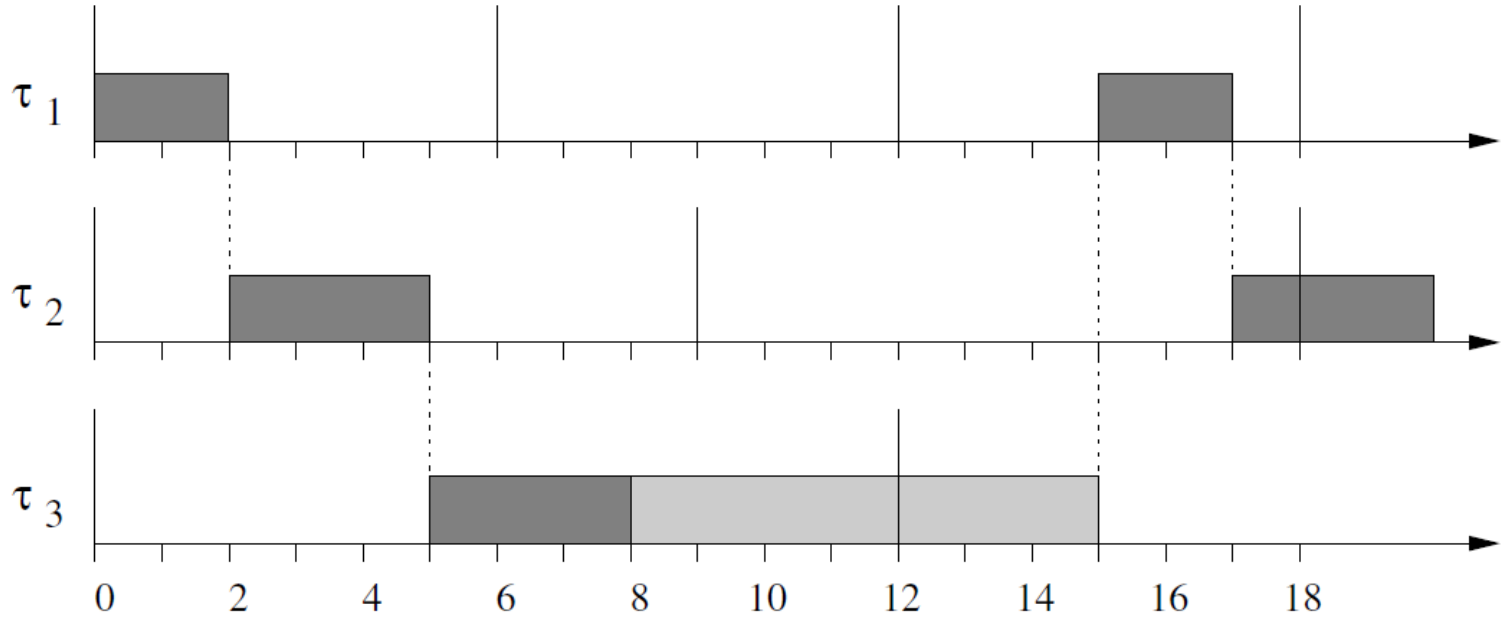


Handling aperiodic overload

- Optimal online scheduling does not exist → task activation time should be known/estimated *a priori*
- Competitive factor: minimum cumulative value that can be achieved by algorithm A
- Main scheduling schemes
  - EDF: best effort approach
  - GED: guarantee based
  - RED: robust EDF with overload prediction

Transient overload from task overrun

- Task overrun: a task (suddenly) runs longer than expected

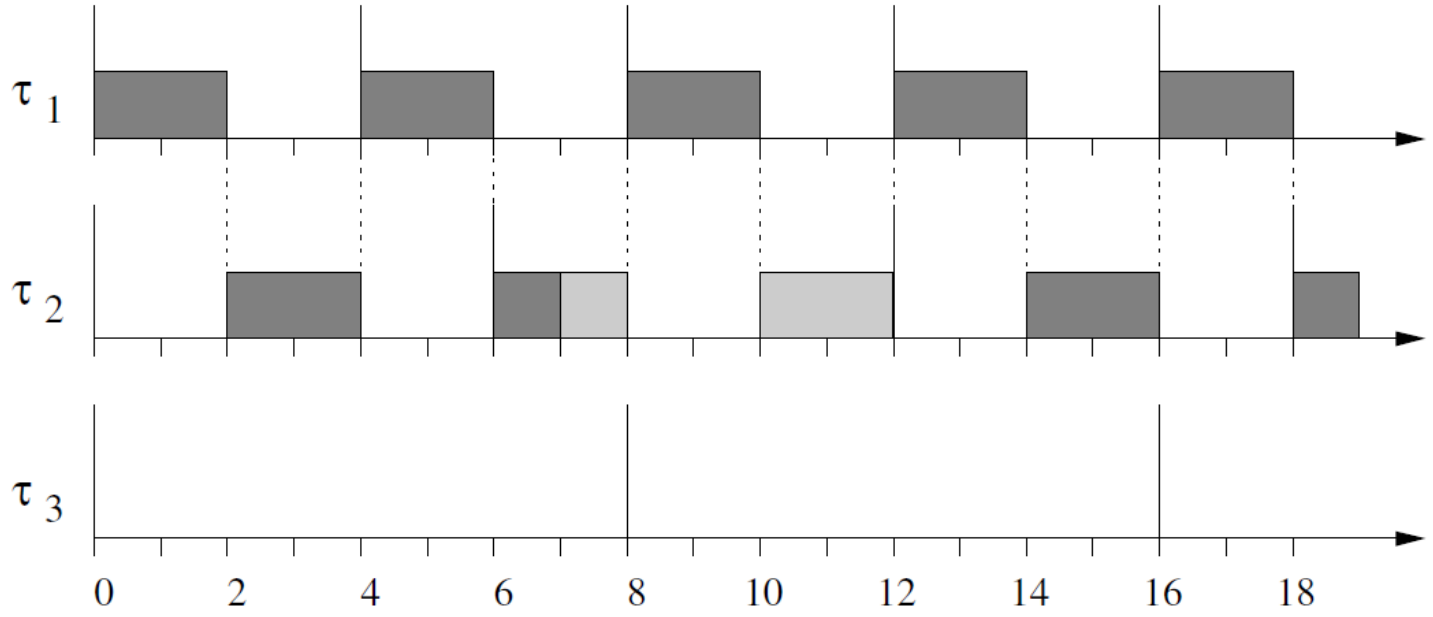


Handling overrun

- Two main schemes
  - Naïve solution: halting the overrun task: dangerous!!!
  - Better solution: allows the task to continue with lower priority
- Resource reservation approach
  - Assign each task a fraction of requested computation time for reservation
  - CPU reservation = reserved resource
  - Reserved resource can be shared between tasks

Handling permanent overload

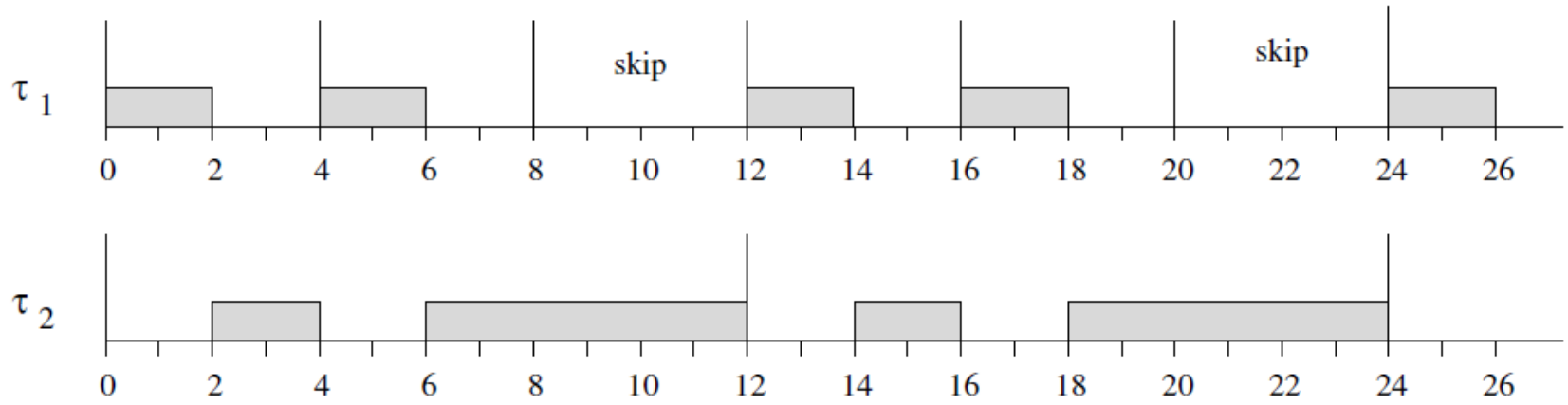
- Requested computation time is larger than available time



T3 will not be executed

Handling permanent overload

- Job skipping: periodically skip a job's instance
- Period adaptation: increase period to reduce activation rate
- Service adaptation: reduce service quality to gain (reduce) computation time, thus reducing system load



Job skipping

The end