# Real Time Systems - Embedded Electronic A

## Handling Aperiodic Overloads

Yashodhan Vishvesh Deshpande
yashodhan-vishvesh.deshpande@stud.hshl.de

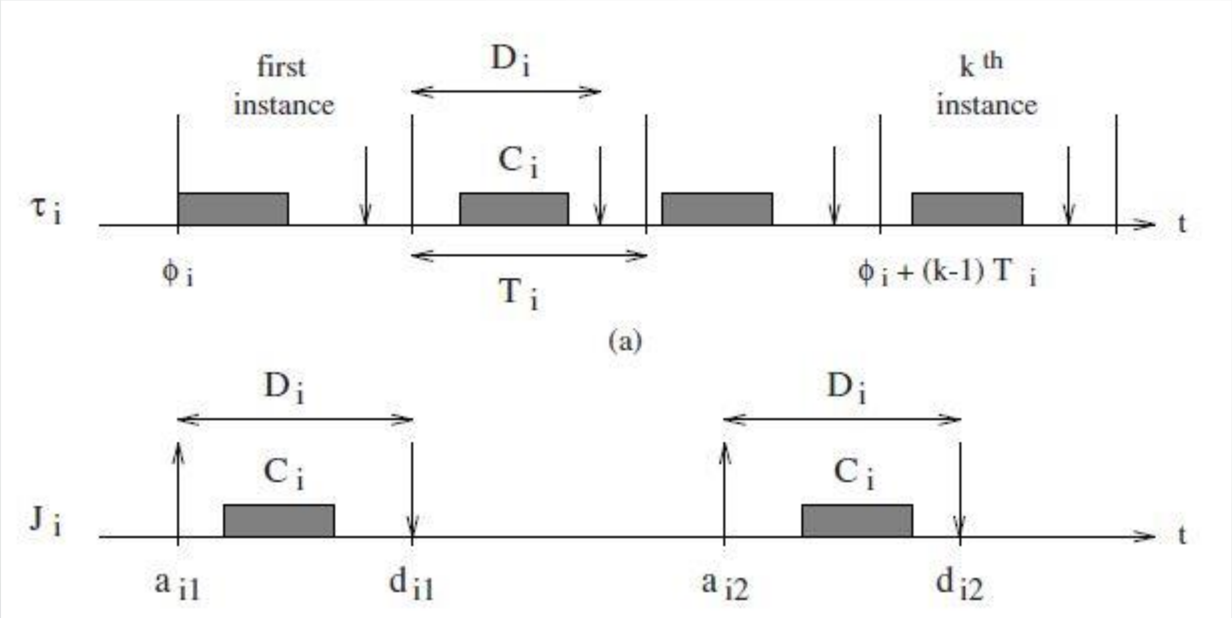
# Motivation

- Overload results in failure to complete task.
- Fatal consequences in hard real time systems.
- Lower performance in soft real time systems.
- Needs handling of overload.

# Contents

1. Periodic vs Aperiodic Overload
2. Computational Load
3. Overload and types
4. Cumulative value
5. Competitive factor
6. Classification of algorithms
7. Robust Earliest Deadline Algorithm

# Aperiodic vs Periodic Tasks

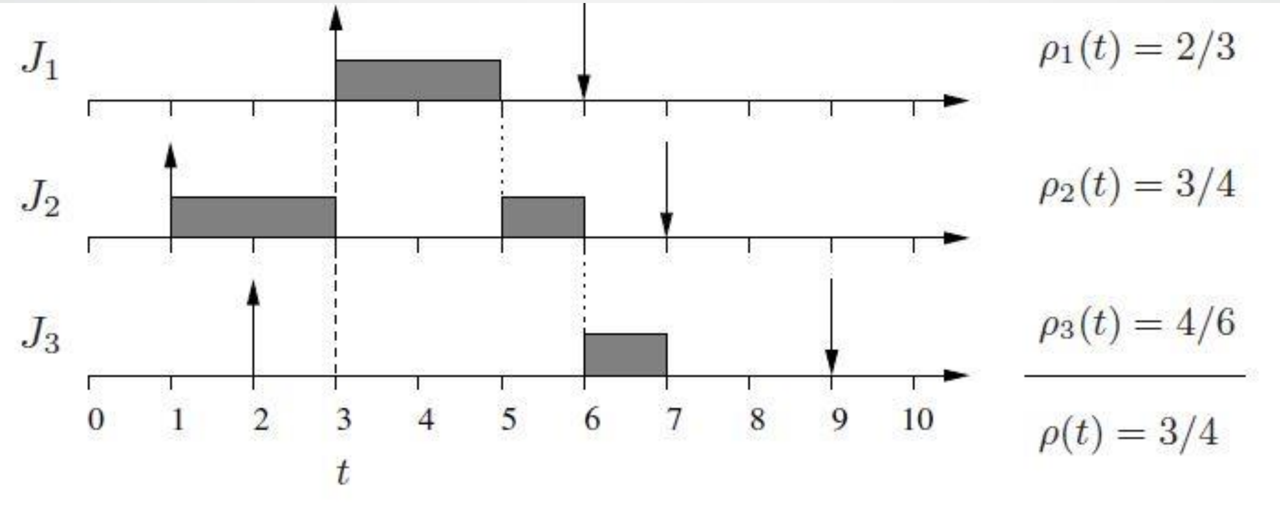



- Periodic Tasks - Sequence of tasks with regular activation times.
- Aperiodic Tasks - Sequence of tasks with irregular activation times.

# Instantaneous Computational Load

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

- P - load , c – computational time , d - deadline and t- current time
- Instantaneous load during a time interval is calculate to determine the load on the system



# Types of Overload

- When the **computational time** demand for a task **exceeds** the **available time** of a processor then it is said to **overload**.
- **Transient Overload** -

  Average load on the system is below overload conditions.

  Load during **specific time period** is above overload conditions.
- **Permanent Overload** - System is overloaded for unknown time duration.

# Cumulative Value of an algorithm

- Every **task** has an **arbitrary value** assigned to measure its importance.
- The arbitrary value can be decided by factors like:

  Computation time required.

  Ratio of arbitrary integer to required computational time - value density
- The performance of an algorithm is defined by **sum** of the **values of** each successfully completed **task**.
- Missing a deadline in **hard real time system** results in cumulative value equal to zero.

# Competitive Factor of a scheduling algorithm

- Competitive factor φ of an algorithm is a number between 0 to 1.
- is a **measure** of minimum cumulative **value** of that algorithm achieved for **φ times** the **cumulative value achieved by clairvoyant** scheduling algorithm.
- When load is greater than 2 , none of the online algorithm can guarantee a competitive factor of more than 0.25 .

# Classification of Algorithms for Overload

- **Best effort** - no prediction for overload conditions
- **With acceptance test** - verifies the schedule of the task set but **rejects tasks** which overload the system
- **Robust** - separate timing constraints and importance, send overloading tasks **back to the queue**.

# Robust Earliest Deadline Algorithm

- The residual laxity L of a task is defined as the interval between its estimated finishing time f and its primary deadline d.

$$L_i = L_{i-1} + (d_i - d_{i-1}) - c_i(t)$$

- Graceful degradation in overloads, deadline tolerance, and resource reclaiming.
- Maximum Exceeding Time

$$E_i = \max(0, -(L_i + M_i)).$$

**begin**

$E = 0;$ // *Maximum Exceeding Time*

$L_0 = 0;$

$d_0 = current\_time();$

$J' = J \cup \{J_{new}\};$

$k$ = <position of $J_{new}$ in the task set $J'$>;

**for** (each task $J'_i$ such that $i \geq k$) **do**

$L_i = L_{i-1} + (d_i - d_{i-1}) - c_i;$

**if** $(L_i + M_i \; < \; -E)$ **then** // *compute* $E_{max}$

$E = -(L_i + M_i);$

**end**

**end**

**if** $(E > 0)$ **then**

<select a set $J^*$ of least-value tasks to be rejected>;

<reject all task in $J^*$>;

**end**

**end**

# Conclusion

# Thankyou

Yashodhan Vishvesh Deshpande

yashodhan-vishvesh.deshpande@stud.hshl.de