Real time System: CPU scheduling

Tej Bahadur Shahi

CDCSIT,TU

Overview

- System Characteristics
- Features of Real-Time Systems
- Implementing Real-Time Operating Systems
- Real-Time CPU Scheduling

Introduction

- A real-time system requires that results be produced within a specified deadline period
- An embedded system is a computing device that is part of a larger system (I.e. automobile, airliner)
- A safety-critical system is a real-time system with catastrophic results in case of failure

Hard Vs Soft RTS

- A hard real-time system guarantees that real-time tasks be completed within their required deadlines.
- A soft real-time system provides priority of real-time tasks over non real time tasks"
- **Periodic and aperiodic tasks."**
- **Starting deadline/ completion deadline."**

RTS Characteristic

- Single purpose
- Small size
- Inexpensively mass-produced
- Specific timing requirements

RTS Characteristic

- Many real-time systems are designed using system-on-a-chip (SOC) strategy.
- SOC allows the CPU, memory, memory-management unit, and attached peripheral ports (i.e., USB) to be contained in a single integrated circuit.

Real Time Kernels

- Most real-time systems do not provide the features found in a standard desktop system

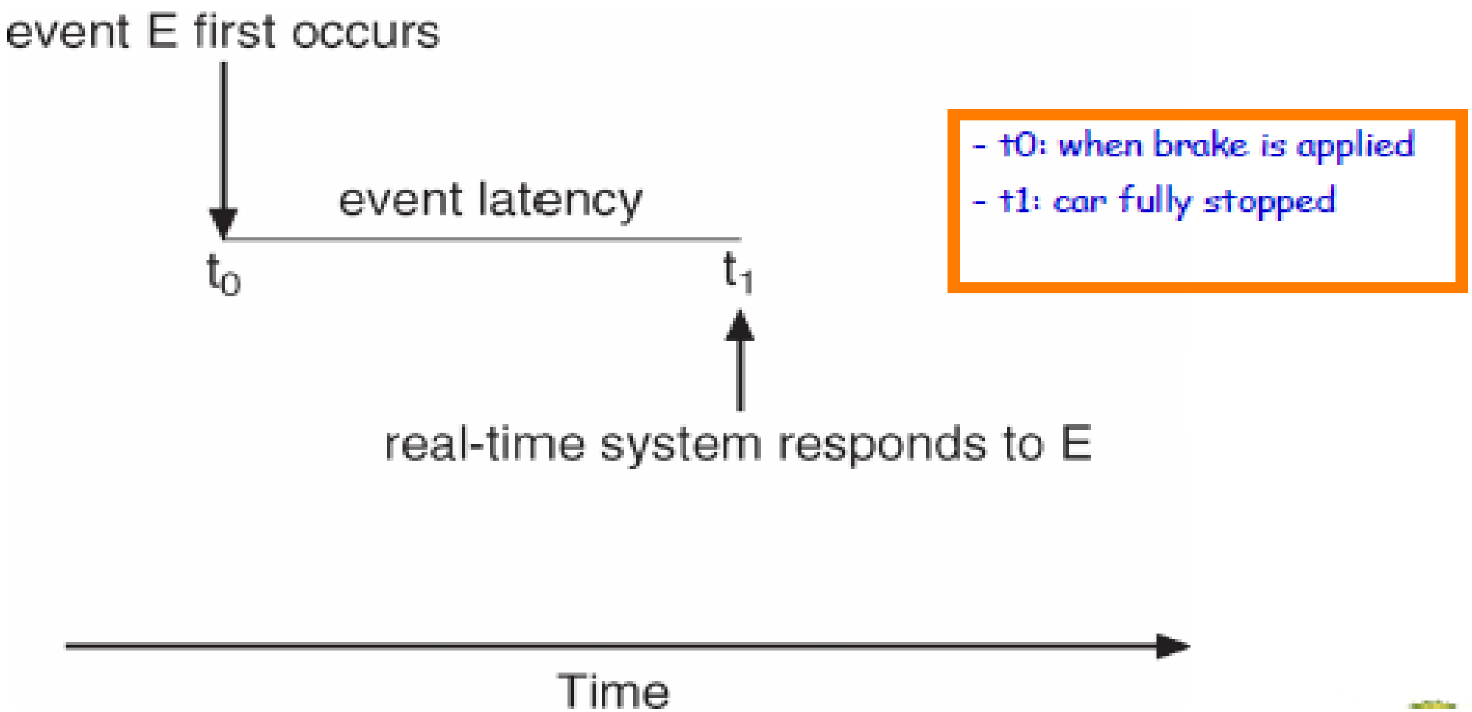
- Because:
  - Real-time systems are typically single-purpose
  - Real-time systems often do not require interfacing with a user

Implementing Real-Time Systems

- In general, real-time operating systems must provide:
  - Preemptive, priority-based scheduling.
  - Preemptive kernels.
  - Latency must be minimized

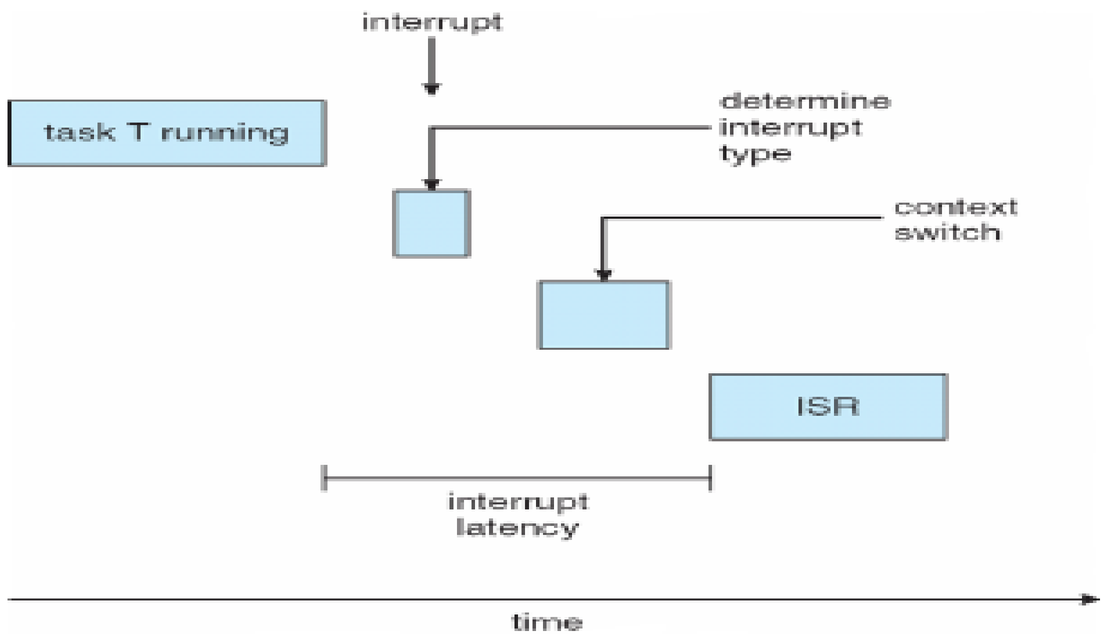
Minimizing Latency

- Event latency is the amount of time from when an event occurs to when it is serviced



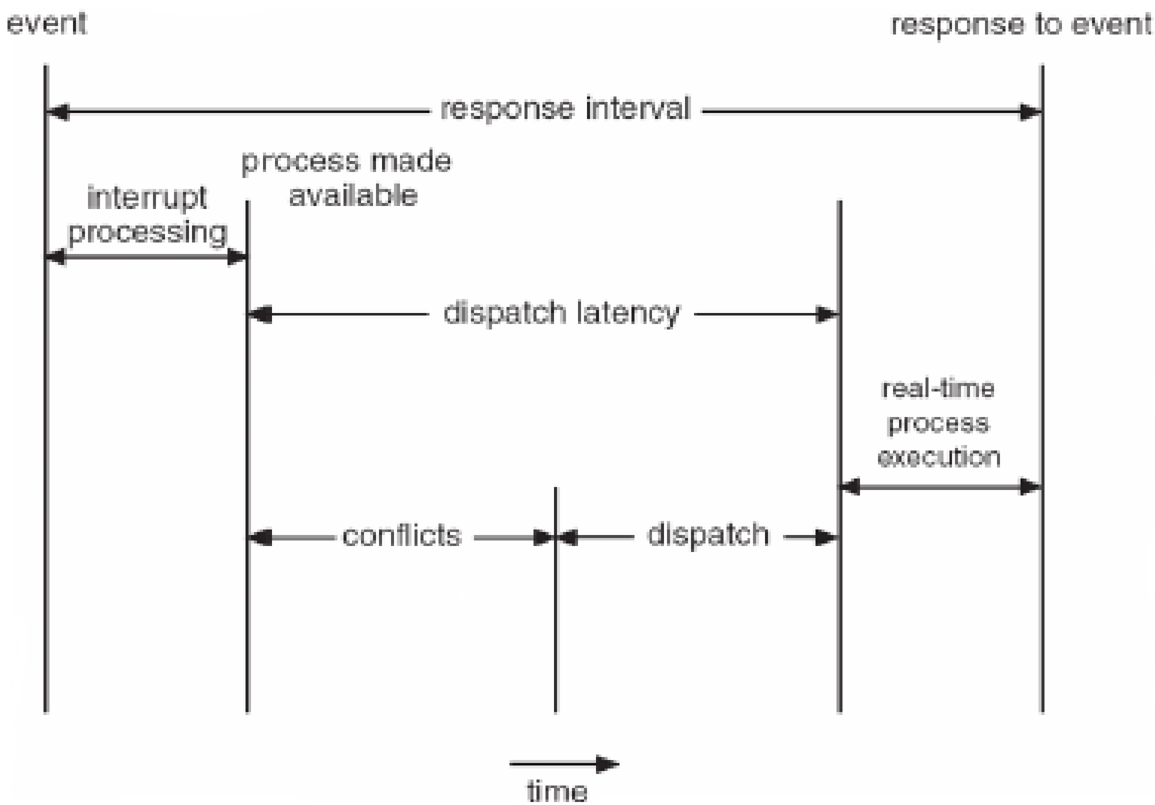
Interrupt Latency

- Interrupt latency is the period of time from when an interrupt arrives at the CPU to when it is serviced



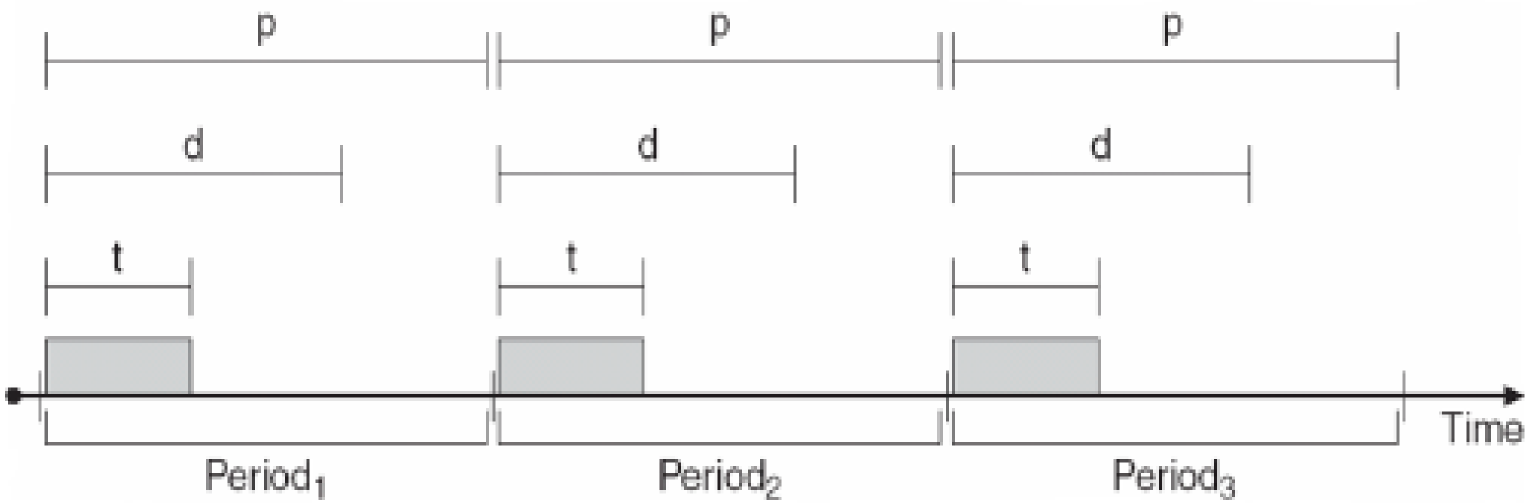
Dispatch Latency

- Dispatch latency is the amount of time required for the scheduler to stop one process and start another"



RTS scheduling

- How to schedule the Tasks such that given timing constraints are satisfied?



d may be the same as p.

Task models

- Non periodic/Aperiodic (three parameters)
  - A: arriving time
  - C: computing time
  - D: deadline (relative deadline)
- Timing constraints: deadline for each task,
  - Relative to arriving time or absolute deadline

Scheduling Problem

- Given a set of tasks (ready queue)
  - Check if the set is schedulable
  - If yes, construct a schedule to meet all deadlines
  - If yes, construct an optimal schedule e.g. minimizing response times

Tasks with the same arrival time

- Assume a list of tasks
  - (A,C1, D1)(A,C2, D2) ...(A,Cn,Dn) that arrive at the same time i.e. A
- How to find a feasible schedule?
  - (there may be many feasible schedules)

Earliest Due Date first (EDD) [Jackson 1955]

- EDD: order tasks with non-decreasing deadlines.
  - Simple form of EDF (earliest deadline first)
  - Example: (1,10)(2,3)(3,5)
  - Schedule: (2,3)(3,5)(1,10)
- FACT: EDD is optimal
  - If EDF cann’t find a feasible schedule for a task set, then no other algorithm can, i.e. The task set is non schedulable.

Tasks with different arrival times

- Assume a list of tasks
  - S= (A1,C1, D1)(A2,C2, D2) ...(An,Cn,Dn)
- Preemptive EDF [Horn 1974]:
  - Whenever new tasks arrive, sort the ready queue according to earliest deadlines first at the moment
  - Run the first task of the queue if it is non empty

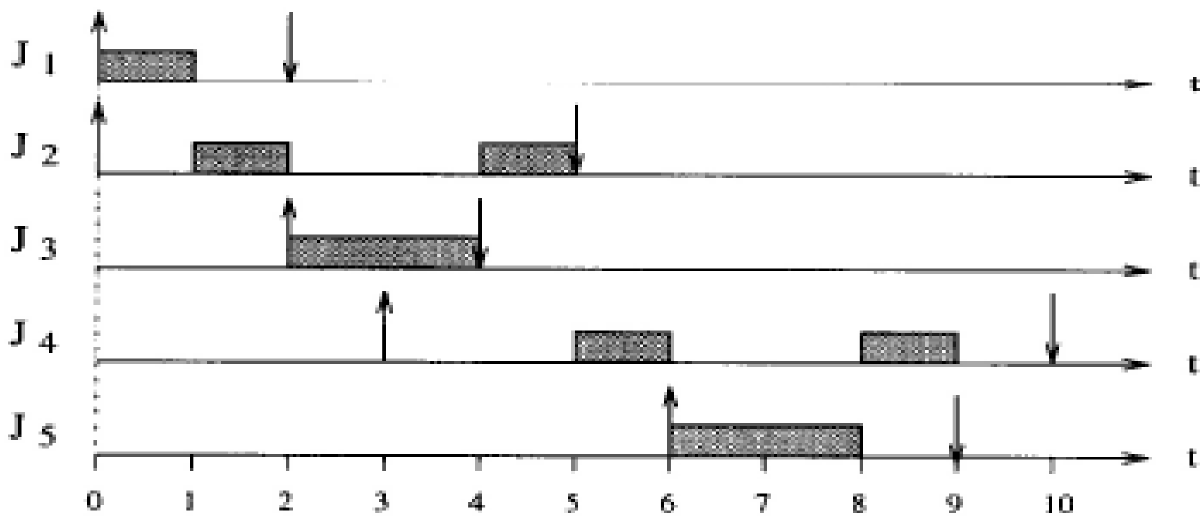
Earliest Deadline First Scheduling

- Priorities are assigned according to deadlines:
  - the earlier the deadline, the higher the priority;
  - the later the deadline, the lower the priority

EDF: Example

- ***Example***:

	J_1	J_2	J_3	J_4	J_5
a_i	0	0	2	3	6
C_i	1	2	2	2	2
d_i	2	5	4	10	9



Least slack scheduling

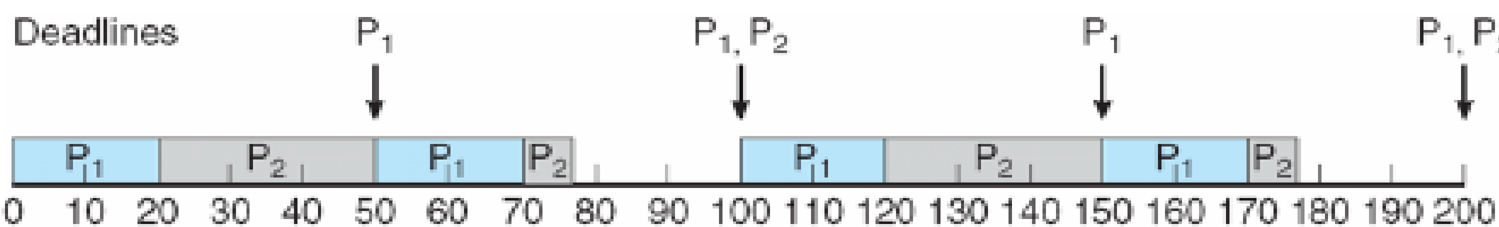
- similar to shortest remaining time scheduling with the concept of a deadline thrown in.
- pick the process that we can least afford to delay.
- Least slack is computed as the time to the deadline minus the amount of computation.
- For example, suppose that our remaining computation time, *C*, is 5 msec. and the deadline, *D*, is 20 msec from now. The slack time is *D* - *C*, or 15 msec.
- The scheduler will compare this slack time with the slack times of other processes in the system and run the one with the lowest slack time.

Least Slack Time scheduling

- With earliest deadline, we will always work on the process with the nearest deadline, delaying all the other processes.
- With least slack scheduling, we get a balanced result in that we attempt to keep the differences from deadlines balanced among processes.

Rate Monotonic Scheduling

- priority is assigned based on the inverse of its period (its rate)"
  - Shorter periods = higher priority;"
  - Longer periods = lower priority“
- P1 is assigned a higher priority than P2.



Periodic tasks with completion deadline

Rate: 1/period, the higher the rate, the higher the priority (the smaller P is)

Schedulability

???????

Thank you