Scheduling Algorithm and Analysis

EDF
(Module 28)

Yann-Hang Lee
Arizona State University
yhlee@asu.edu
(480) 727-7507

Summer 2014
Priority-Driven Scheduling of Periodic Tasks

- **Why priority-driven scheduling**
  - use priority to represent urgency
  - easy implementation of scheduler (compare priorities and dispatch tasks)
  - tasks can be added or removed easily
  - no direct control of execution instance

- **How can we analyze the schedulability if we don’t know when a task is to be executed**

- **Let’s begin a deterministic case in one processor**
  - independent periodic tasks
  - deadline = period
  - preemptable
  - no overhead for context switch
Priority-Driven Schedules

- **Assign priority when jobs arrive**
  - static -- all jobs of a task have a fixed priority
  - dynamic -- different priorities to individual tasks of a task
  - relative priorities don’t change while jobs are waiting

- **Static priority schedules**
  - Rate-monotonic -- the smaller a task period, the higher its priority
  - Deadline-monotonic – if deadline \( \neq \) period, the smaller a task’s deadline (relative), the higher its priority

- **Dynamic priority schedules**
  - EDF -- earliest deadline (absolute) first

- **Schedulable utilization:**
  - a scheduling algorithm can feasibly schedule any sets of priority tasks if the total utilization is equal to or less than the schedulable utilization of the algorithm
Earliest-deadline First (EDF) Schedule

- **Priority preemptive scheduling**
  - a job with earliest (absolute) deadline has the highest priority
  - does not require the knowledge of execution time

- **Optimal if**
  - single processor, no resource contention, preemption
  - why it is optimal: assume a feasible schedule

\[
\begin{align*}
(\text{non-EDF}) & \quad J_i & J_k & d_k & d_i \\
& \quad r_k & (r_k) & & \\
(\text{EDF}) & \quad J_k & J_i & d_k & d_i
\end{align*}
\]
Non-optimality of EDF

- Non-preemptive or multiple processors
- Scheduling anomaly --- the schedule fails after we reduce job execution times

( all jobs meet their deadline under EDF after increasing e₁ )
EDF Schedule

- A optimal algorithm under single processor and preemptable tasks
- How do we know a set of periodic tasks are schedulable under EDF?
- If we know the schedulable utilization $SU$ of EDF, then any sets of tasks are schedulable when $U \leq SU$
- Theorem: A set of $n$ periodic tasks can be scheduled by EDF iff

$$U = \sum_{i=1}^{n} \frac{e_i}{P_i} \leq 1$$

- This schedulability utilization is no long true if deadline < period.
Example of EDF Schedule

- **A digital robot with EDF schedule**
  - control loop: $e_c \leq 8\text{ms}$ at 100Hz
  - BIST: $e_b \leq 50\text{ms}$
  - given
    
    
    
    \[ u_c + u_b = \frac{8}{10} + \frac{50}{p_b} \leq 1 \]
  
    - BIST can be done every 250ms

- **Add a telemetry task to send and receive messages with $e_t \leq 15\text{ms}$**
  - if BIST is done every 1000ms
  - given
    
    
    
    \[ u_c + u_b = \frac{8}{10} + \frac{50}{1000} + \frac{15}{D_t} \leq 1 \]
  
    - the telemetry task can have a relative deadline of 100ms
    - sending or receiving must be separated at least 100ms
Supplementary Slides
On-line vs. Off-line Scheduling

- **Off-line scheduling**: the schedule is computed off-line and is based on the knowledge of the release times and execution times of all jobs.
  - For deterministic systems: with fixed set of functions and job characteristics does not vary or vary only slightly.

- **On-line scheduling**: a scheduler makes each scheduling decision without knowledge about the jobs that will be released in the future.
  - there is no optimal on-line schedule if jobs are non-preemptive
  - when a job is released, the system can serve it or wait for the future jobs

![Diagram showing scheduling decisions](image)