Real-Time Programming

Lecture 10

Farhang Nemati

Spring 2016

Periodic Task Scheduling

- A set of periodic tasks
- Independent (No resource sharing)
- No precedence constraints
- A periodic task is repeated in specific rate
  - Sensor acquisition
  - Monitoring
  - Control loops

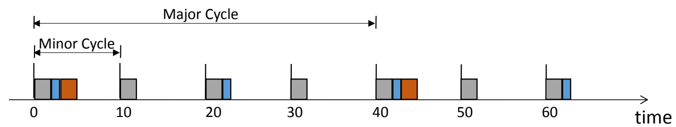
Periodic Task Scheduling; Timeline Algorithm (Cyclic Executive)

- Is used widely
- Simple and easy
- Algorithm: A global loop in equal iterates in equal time slots and in each time slot one or more tasks are executed

Timeline Algorithm (Cyclic Executive)

- Example

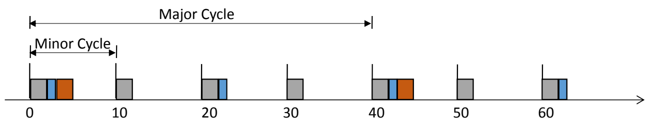
	e_i	p_i
τ_1	2	10
τ_2	1	20
τ_3	2	40



Major Cycle = Least Common Multiply (LCM) of task periods

Timeline Algorithm (Cyclic Executive); Schedulability Test

- A task set could be schedulable if: For each minor cycle the summation of execution times of tasks executing during that cycle <= length of Minor Cycle



Timeline Algorithm (Cyclic Executive)

- + Simple.
  - A main program within each interval equal to the minor cycle will execute the appropriate tasks. This is repeated at each interval equal to the major cycle.
- + Sequential
  - No context switch
  - No need to protect shared data, resources, etc.
- - Not flexible; if the period or execution time of a task changes the whole schedule might need to be changed
  - If the periods are not multiple of each other the time table could be too big, e.g., major cycle for tasks with periods 7, 11, 27 is 7*11*27 = 2079
- - If a task is malfunctioning it will affect other tasks

Fixed-Priority Scheduling Algorithms

- A fixed priority is assigned to each task offline
- The tasks are scheduled according to their fixed priorities
- Static (priority is fixed) online scheduling

Rate Monotonic Scheduling Algorithm (RMS)

- The rule for assigning priorities to the periodic tasks: Assign priorities to tasks according to their arrival rate; the shorter the period of a task is the higher its priority will be.
  - E.g., the task with the shortest period gets the highest priority and the task with the longest period will get the lowest priority.
- Online static (fixed priority) scheduling algorithm
- Among fixed-priority preemptive algorithms RMS is optimal

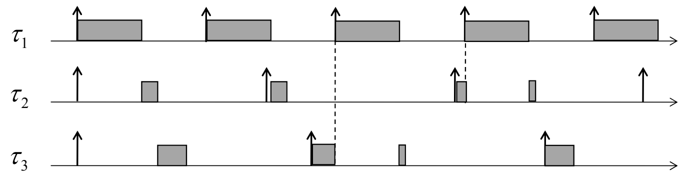
Rate Monotonic Scheduling Algorithm (RMS)

- Task Model
  - Independent tasks
  - Preemptive
  - Tasks are specified as follows:

$$\tau_i(e_i, p_i)$$

Rate Monotonic Scheduling Algorithm (RMS);

- Example



$$\rho_1 = 1,\ \rho_2 = 2,\ \rho_3 = 3$$

RMS

- Processor Utilization Factor, U:

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

- Example

	e_i	p_i
τ_1	4	10
τ_2	6	20
τ_3	9	40

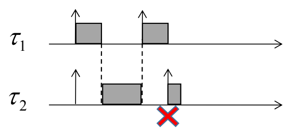
$$U = \frac{4}{10} + \frac{6}{20} + \frac{9}{40} = 92.5\%$$

RMS;
Schedulability Test

- U < 1 does not necessarily mean the task set is schedulable
- Example

	e_i	p_i
τ_1	2	5
τ_2	4	7

$$U = \frac{2}{5} + \frac{4}{7} \approx 97.14\%$$



RMS;
Schedulability Test

- Assuming n is the number of tasks of a periodic task set, the task set is schedulable if:

$$U \le n \times (2^{1/n} - 1)$$

- The upper bound is only dependent on the number of tasks

RMS;
Schedulability Test

- The test is sufficient but not necessary. Let $U_{\text{lub}} = n \times (2^{1/n} - 1)$:
  - If $U \le U_{\text{lub}}$ the task set is schedulable
  - If $U > 1$ the task set is not schedulable
  - If $U_{\text{lub}} \le U \le 1$ no conclusion!

RMS;
Schedulability Test

- Example of upper (U_{lub}) bounds for RMS

n	U_{lub}
1	1
2	0.828
3	0.780
4	0.757
5	0.743
6	0.735
7	0.729
∞	≈0.693

RMS;
Schedulability Test

- Shortest repeating cycle = Least Common Multiple (LCM) of periods
- We can test the schedule in LCM
  - Too difficult when the number of tasks is high
- It's enough if we only test the Critical Instant
  - Critical Instant of a task: The instant of a task where the task has worst response time
  - Critical Instant of a task set: The instant where all tasks arrive at the same time. The worst case scenario happens in the critical instant, i.e., all tasks have their worst response time [M. Joseph and P. Pandya, 1986]

RMS; Response Time

- Calculating Maximum Response Times
- Let denote response time of task τ_i with R_i and denote the set of tasks having priority higher than τ_i with H_i

$$R_i = e_i + \sum_{\tau_j \in H_i} \left\lceil \frac{R_i}{p_j} \right\rceil e_j$$

RMS; Exact Schedulability Test

- Response Time Analysis: For each task τ_i
  - 1) $R^{(0)}{}_i = e_i + \sum_{\tau_j \in H_i} e_j$
  - 2) $R^{(k+1)}{}_i = e_i + \sum_{\tau_j \in H_i} \left\lceil \frac{R^k{}_i}{p_j} \right\rceil e_j$
  - Using numerical calculations we continue the equation in step 2 until it either reaches a fix value (does not increase anymore) or it exceeds the deadline (d_i)
  - If the final response time exceeds the deadline the task is unschedulable
  - If the response time reaches a fix value before it exceeds the deadline the task is schedulable. This value is the maximum response time of the task.

RMS; Response Time Analysis

- Example

	e_i	p_i
τ_1	1	3
τ_2	1	4
τ_3	2	6
τ_4	1	20

$$R^{(1)}{}_1 = R^{(0)}{}_1 = e_1 = 1$$

RMS; Response Time Analysis

- Calculate R_2

	e_i	p_i
τ_1	1	3
τ_2	1	4
τ_3	2	6
τ_4	1	20

  - Step 0: $R^{(0)}{}_2 = e_2 + e_1 = 1 + 1 = 2$
  - Step 1: $R^{(1)}{}_2 = e_2 + \left\lceil \frac{R^{(0)}{}_2}{p_1} \right\rceil e_1 = 1 + \left\lceil \frac{2}{3} \right\rceil 1 = 2$
  - $R^{(1)}{}_2 = R^{(0)}{}_2 = 2 \le d_2 = 4 \;\Rightarrow\; \tau_2$ is schedulable

RMS;
Response Time Analysis

- Calculate R_3

	e_i	p_i
τ_1	1	3
τ_2	1	4
τ_3	2	6
τ_4	1	20

  - Step 0: $R^{(0)}{}_3 = e_3 + e_2 + e_1 = 2 + 1 + 1 = 4$
  - Step 1: $R^{(1)}{}_3 = e_3 + \left\lceil \frac{R^{(0)}{}_3}{p_1} \right\rceil e_1 + \left\lceil \frac{R^{(0)}{}_3}{p_2} \right\rceil e_2 = 2 + \left\lceil \frac{4}{3} \right\rceil 1 + \left\lceil \frac{4}{4} \right\rceil 1 = 5$
  - Step 2: $R^{(2)}{}_3 = e_3 + \left\lceil \frac{R^{(1)}{}_3}{p_1} \right\rceil e_1 + \left\lceil \frac{R^{(1)}{}_3}{p_2} \right\rceil e_2 = 2 + \left\lceil \frac{5}{3} \right\rceil 1 + \left\lceil \frac{5}{4} \right\rceil 1 = 6$
  - Step 3: $R^{(3)}{}_3 = e_3 + \left\lceil \frac{R^{(2)}{}_3}{p_1} \right\rceil e_1 + \left\lceil \frac{R^{(2)}{}_3}{p_2} \right\rceil e_2 = 2 + \left\lceil \frac{6}{3} \right\rceil 1 + \left\lceil \frac{6}{4} \right\rceil 1 = 6$

RMS;
Response Time Analysis

- $R^{(3)}{}_3 = R^{(2)}{}_3 = 6 \le d_3 = 6 \quad \Rightarrow \quad \tau_3$ is schedulable

- Exercise: Calculate R_4

	e_i	p_i
τ_1	1	3
τ_2	1	4
τ_3	2	6
τ_4	1	20

Earliest Deadline First(EDF)

- At arrival times of tasks sort the ready queue according their deadlines; earliest deadline first
- Online dynamic scheduling algorithm
- EDF is optimal

Earliest Deadline First(EDF)

- Task Model
  - Independent tasks
  - Preemptive
  - Tasks are specified as follows:

$$\tau_i(e_i, d_i)$$

EDF; Schedulability Test

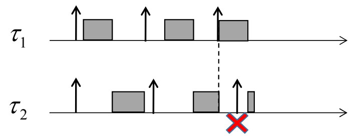
- Assuming n is the number of tasks of a periodic task set, the task set is schedulable if and only if:

$$U \leq 1 \qquad where \quad U = \sum_{i=1}^{n} \frac{e_i}{p_i}$$

- The condition is sufficient and necessary

Jitter

- In practice it can happen that a task does not arrives precisely at it beginning of its period



Jitter

- Let
  - $delay_{\max}$ = maximum delay of task τ_i from its period start
  - $delay_{\min}$ = minimum delay of task τ_i from its period start

$$jitter_i = delay_{\max} - delay_{\min}$$

Jitter

- Example