CIS 721 - Real-Time Systems
Lecture 7: Response Time Analysis

Mitch Neilsen
neilsen@ksu.edu

Outline

- Commonly Used Approaches For Real-Time Scheduling (Ch. 4)
  - Clock-Driven Scheduling (Ch. 5)
  - **Priority-Driven Scheduling**
    - **Periodic Tasks (Ch. 6)**
      - **Utilization-Based Test**
      - **Response Time Analysis**
      - **Arbitrary Start Times**
    - Aperiodic and Sporadic Tasks (Ch. 7)

Periodic Task Model

- **Periodic task set:** $\{T_1, \ldots\ldots, T_n\}$, each task consists of a set of **jobs**: $T_i = \{J_{i1}, J_{i2}, \ldots\ldots\}$
- ϕ_i: p**hase** of task T_i = time when its first job is released
- p_i: p**eriod** of T_i = inter-release time
- e_i or C_i: e**xecution time** of T_i
- u_i: **utilization** of task T_i is given by $u_i = e_i / p_i$
- D_i: (relative) **deadline** of T_i, typically $D_i = p_i$

Schedulability Analysis

- **Utilization-Based Tests**
  - Not exact (sufficient, but not necessary).
  - Not applicable to more general task models.
- **Time-Based Tests (Response Time Analysis)**
  - Use analytic approach to predict worst-case response time of each task.
  - Compare computed worst-case response times with deadlines.
  - Exact (sufficient and necessary)

Time-Based Tests (Response Time Analysis)

- M. Joseph and P.K. Pandya, “Finding Response Times in a Real-Time System”, The Computer Journal, Vol. 29, No. 5, pp. 390-395, 1986.

Example

Task τ_i		Period T_i	Deadline D_i	Run-Time C_i	Phase φ_i
A	(High Priority)	7	7	3	0
B		12	12	3	0
C	(Low Priority)	20	20	5	0

- $U = 3/7 + 3/12 + 5/20 = 13/14 \approx 0.93$
- $U_{RM} = 3\ (2^{1/3} - 1) \approx 0.78$
- Since $U_{RM} < U \leq 1.0$, no conclusion can be drawn using the Utilization-Based Test.

Response Time

- The **response time** (R_i) for task T_i is given by $R_i = e_i + I_i$ where:
  - e_i is the execution time of each job in T_i, and
  - I_i is the **maximum interference** caused by higher priority tasks in any interval $[t, t + R_i)$.

Maximum Interference (I_i)

- The maximum number of releases of task T_j in the time interval [t, t + R_i) is given by:

$$\left\lceil \frac{R_i}{p_j} \right\rceil$$

- So, the interference caused by task T_j is $$\left\lceil \frac{R_i}{p_j} \right\rceil * e_j$$

- The maximum interference caused by all higher priority tasks is given by

$$I_i = \sum_{j \in hp(i)} \left\lceil \frac{R_i}{p_j} \right\rceil * e_j$$

where *hp(i)* = set of all tasks with priority greater than task T_j .

Response Time Analysis

- The **(worst-case) response time** (w_i) for task T_i is given by the implicit equation:

$$R_i = e_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{p_j} \right\rceil * e_j$$

- Which is solved by forming a recurrence relation:

$$w_i^{n+1} = e_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^n}{p_j} \right\rceil * e_j$$

$$w_i^0 = e_i$$

Solving Recurrence

- The sequence $w_i^0, w_i^1, w_i^2, \ldots, w_i^n$

  is clearly non-decreasing:

  - If $w_i^{n+1} = w_i^n$, then a fixed point (solution) has been found.

  - If $w_i^{n+1} > D_i$, then no solution exists.

Algorithm

Input: $e_1,..,e_m,\ p_1,..,\ p_m, D_1,..,D_m$

Output: $R_1, R_2,...,R_m$

for $i = 1$ to m

$n = 0$

$w_i^n = e_i$

loop

$$w_i^{n+1} = e_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^n}{p_j} \right\rceil * e_j$$

if $w_i^{n+1} = w_i^n$ then

$R_i = w_i^n$

break out of loop {solution found}

if $w_i^{n+1} > D_i$ then

break out of loop {no solution}

$n = n + 1$

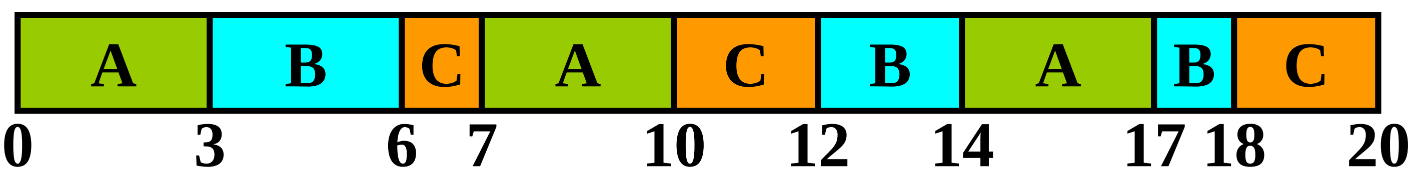
end loop

end for

Example – Rate-Monotonic Priorities

Task τ_i	Period T_i	Run-Time C_i	Worst-Case Response Time R_i
A (high)	7	3	3
B	12	3	6
C (low)	20	5	20

Gantt Chart



Example

$$w_A^0 = e_A = 3$$

$$w_A^1 = e_A + \sum_{j \in hp(A)} \left\lceil \frac{w_A^0}{p_j} \right\rceil * e_j = 3 + 0 = 3$$

$$=> R_A = 3$$

$$w_B^0 = e_B = 3$$

$$w_B^1 = e_B + \sum_{j \in hp(B)} \left\lceil \frac{w_B^0}{p_j} \right\rceil * e_j = e_B + \left\lceil \frac{w_B^0}{p_A} \right\rceil * e_A = 3 + \left\lceil \frac{3}{7} \right\rceil * 3 = 6$$

$$w_B^2 = e_B + \sum_{j \in hp(B)} \left\lceil \frac{w_B^1}{p_j} \right\rceil * e_j = e_B + \left\lceil \frac{w_B^1}{p_A} \right\rceil * e_A = 3 + \left\lceil \frac{6}{7} \right\rceil * 3 = 6$$

$$=> R_B = 6$$

Example (cont.)

$$w_C^0 = e_C = 5$$

$$w_C^1 = e_C + \sum_{j \in hp(C)} \left\lceil \frac{w_C^0}{p_j} \right\rceil * e_j = e_C + \left\lceil \frac{w_C^0}{p_A} \right\rceil * e_A + \left\lceil \frac{w_C^0}{p_B} \right\rceil * e_B$$

$$w_C^1 = 5 + \left\lceil \frac{5}{7} \right\rceil * 3 + \left\lceil \frac{5}{12} \right\rceil * 3 = 11$$

$$w_C^2 = e_C + \sum_{j \in hp(C)} \left\lceil \frac{w_C^1}{p_j} \right\rceil * e_j = e_C + \left\lceil \frac{w_C^1}{p_A} \right\rceil * e_A + \left\lceil \frac{w_C^1}{p_B} \right\rceil * e_B$$

$$w_C^2 = 5 + \left\lceil \frac{11}{7} \right\rceil * 3 + \left\lceil \frac{11}{12} \right\rceil * 3 = 14$$

$$w_C^3 = 5 + \left\lceil \frac{14}{7} \right\rceil * 3 + \left\lceil \frac{14}{12} \right\rceil * 3 = 17$$

$$w_C^4 = 5 + \left\lceil \frac{17}{7} \right\rceil * 3 + \left\lceil \frac{17}{12} \right\rceil * 3 = 20$$

$$w_C^5 = 5 + \left\lceil \frac{20}{7} \right\rceil * 3 + \left\lceil \frac{20}{12} \right\rceil * 3 = 20$$

$$=> R_C = 20$$

Response Time Analysis

- For each task, T_i , compute worst-case response time (R_i).
- If ($R_i \leq D_i$) for each task T_i , then the task set is feasible (schedulable).
- Response Time Analysis is both necessary and sufficient.

Example

Task τ_i	Period T_i	Deadline D_i	Run Time C_i	Response Time R_i
A	20	5	3	3
B	15	7	3	6
C	10	10	4	10
D	20	20	3	20

Since ($R_i \leq D_i$) for each task $\mathbf{T_i}$ **,** the task set is feasible (schedulable). Note: U = 3/20 + 3/15 + 4/10 + 3/20 = 0.9, so the Utilization-Based Test is inconclusive.

Time-Demand Analysis

- Compute total demand on processor time of job released at a critical instant and by higher-priority tasks as function of time from the critical instant.
- Check whether demand can be met before deadline.
- Determine whether T_i is schedulable:
  - Focus on a job in T_i, suppose release time is critical instant of T_i:

    $w_i(t)$: Processor-time demand of this job and all higher-priority jobs released in (t_0, t):

$$w_i(t) = e_i + \sum_{k=1}^{i-1} \left\lceil \frac{t}{p_k} \right\rceil e_k$$

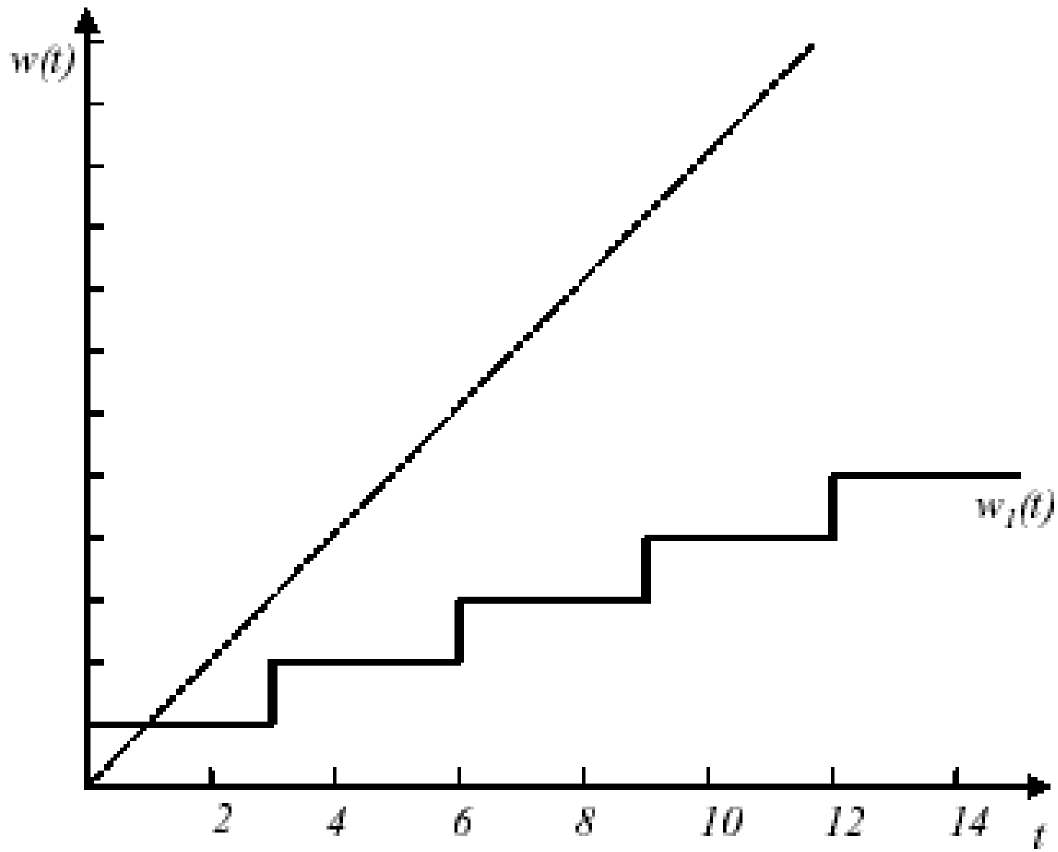
- This job in T_i meets its deadline if, for some

$$t_1 \le D_i \le p_i \quad : \quad w_i(t_1) \le t_1$$

- If this does not hold, job cannot meet its deadline, and system of tasks is not schedulabe by given static-priority algorithm.

Example

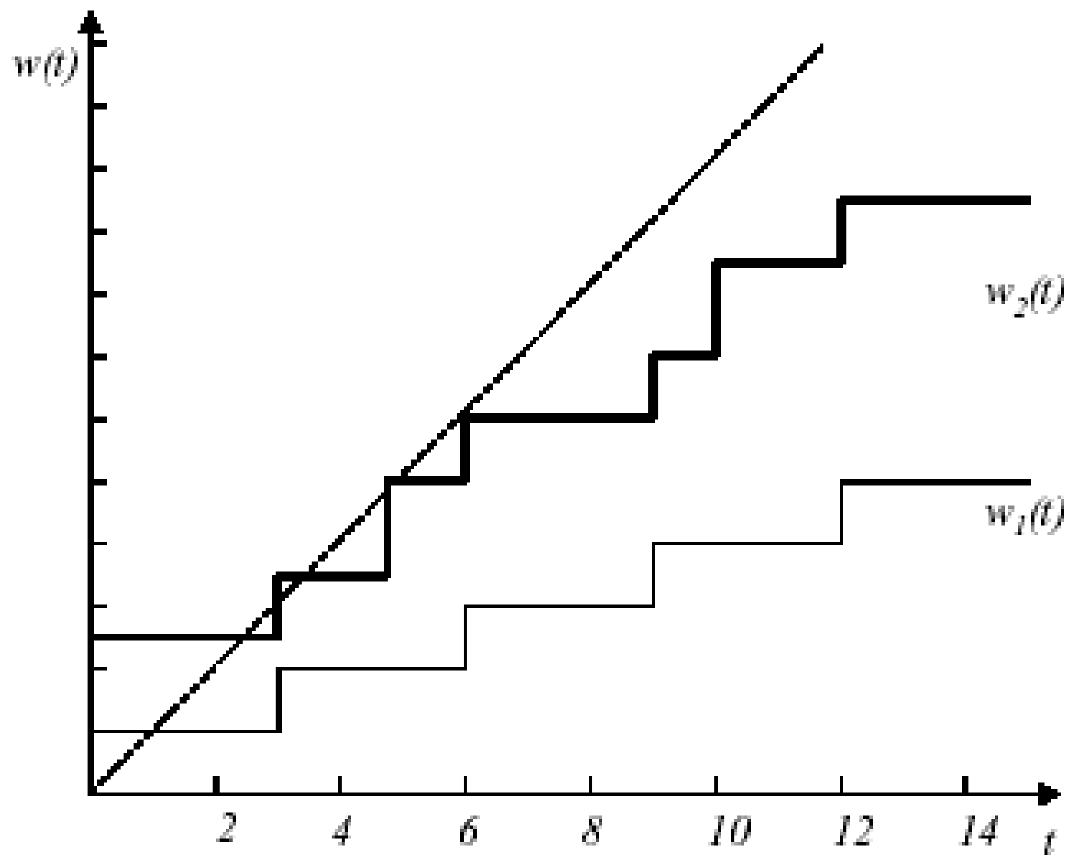
$$\boxed{T_1 = (\;3,\; 1\;)}$$
$$T_2 = (\;5,\; 1.5\;)$$
$$T_3 = (\;7,\; 1.25\;)$$
$$T_4 = (\;9,\; 0.5\;)$$



Example

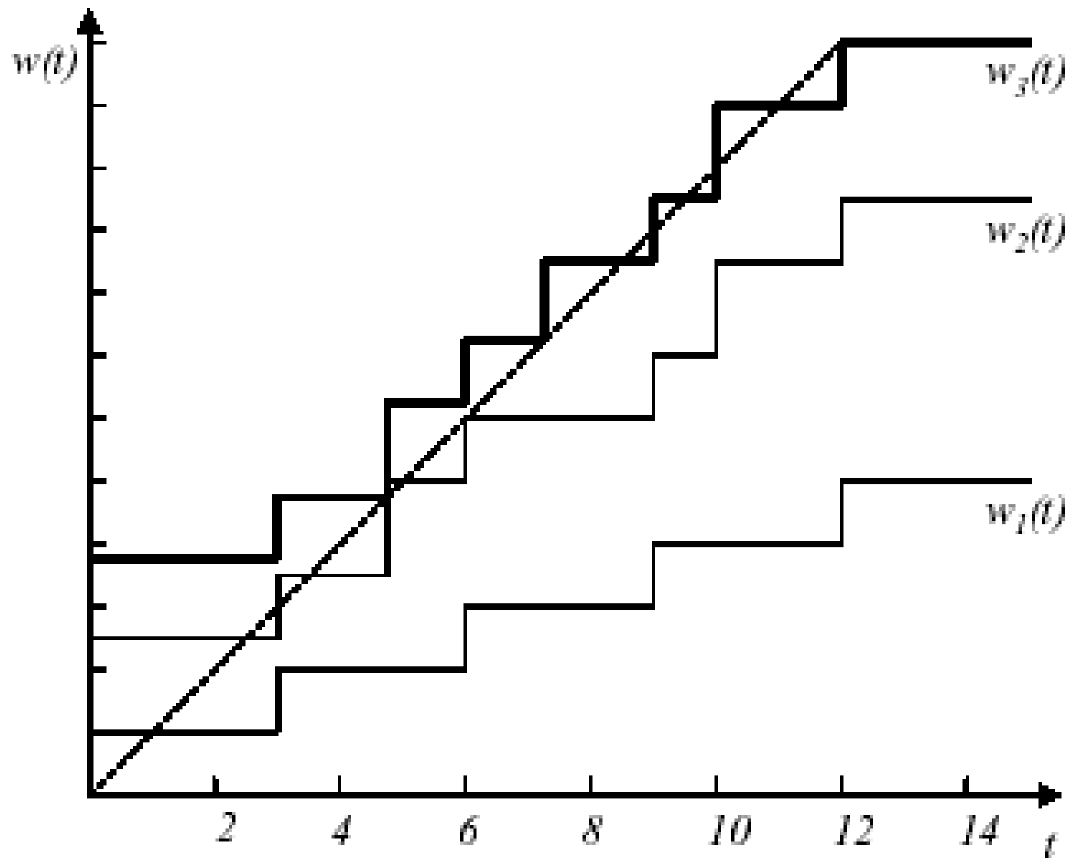
$$\boxed{\begin{array}{llll} T_1 & = & (3, & 1) \\ T_2 & = & (5, & 1.5) \end{array}}$$

$$\begin{array}{llll} T_3 & = & (7, & 1.25) \\ T_4 & = & (9, & 0.5) \end{array}$$



Example

$$T_1 = (3, 1)$$
$$T_2 = (5, 1.5)$$
$$T_3 = (7, 1.25)$$
$$T_4 = (9, 0.5)$$



Arbitrary Start Times (Phasing)

- N.C. Audsley, “Optimal Priority Assignment and Feasibility of Static Priority Tasks with Arbitrary Start Times”, Tech. Report YCS 164, University of York, York, England, 1991.

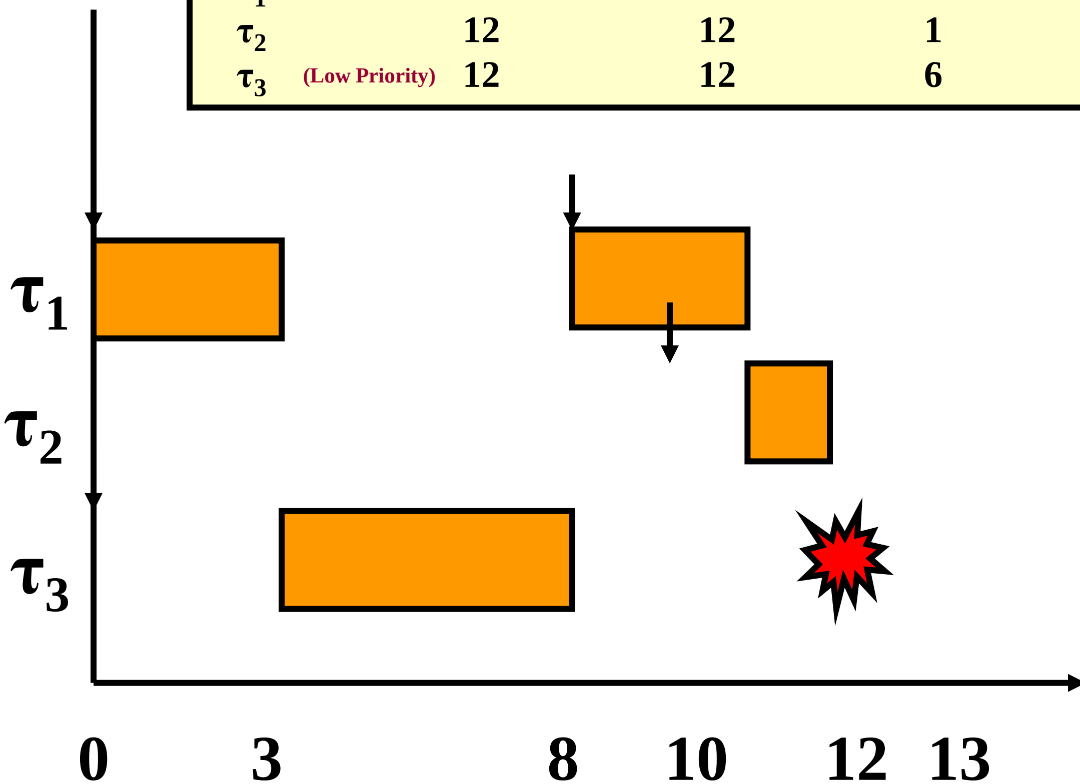
Example #1

- If phasing ($\boldsymbol{\varphi_i}$ or $\mathbf{O_i}$ **)** is allowed to be greater than 0, then a Rate Monotonic (RM) priority assignment may not be optimal.

Task τ_i	Period T_i	Deadline D_i	Run-Time C_i	Phase φ_i
τ_1	8	8	3	0
τ_2	12	12	1	10
τ_3	12	12	6	0

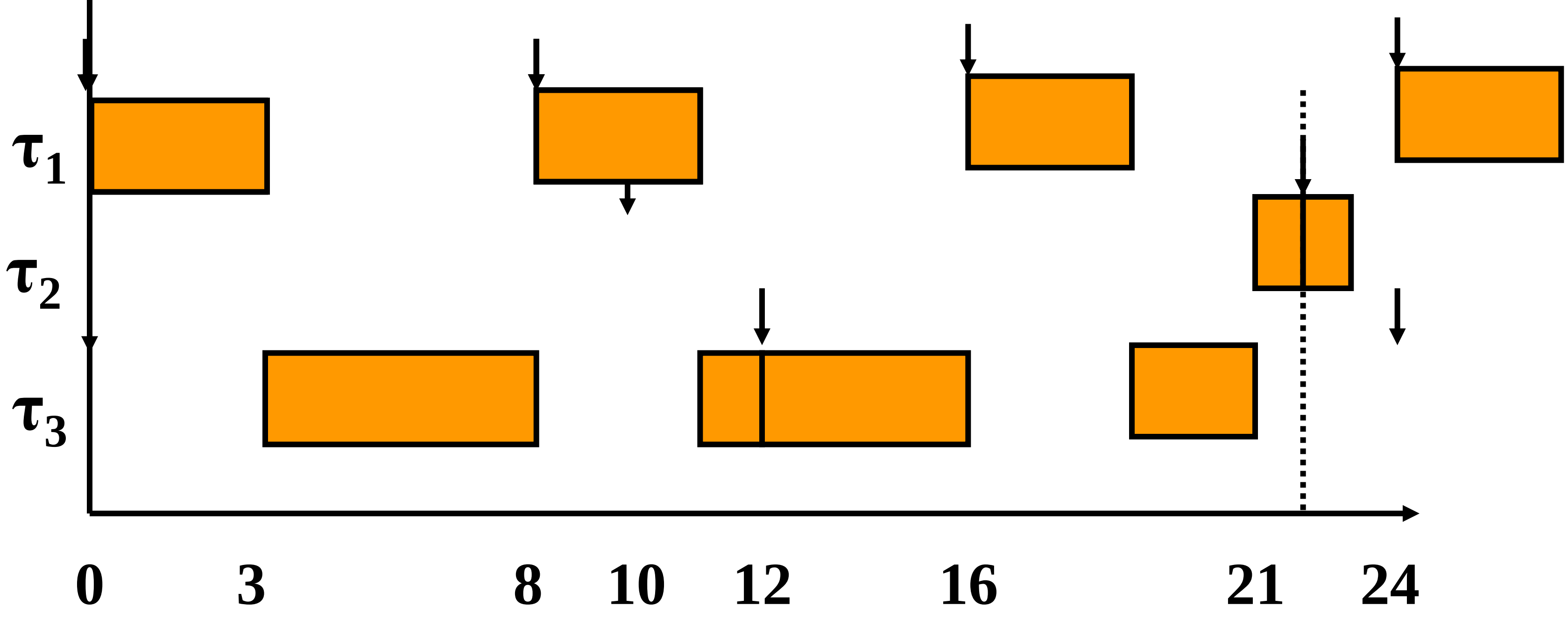
Example #1 (cont.)

Task τ_i		Period T_i	Deadline D_i	Run-Time C_i	Phase φ_i	Priority π_i
τ_1	(High Priority)	8	8	3	0	3
τ_2		12	12	1	10	2
τ_3	(Low Priority)	12	12	6	0	1



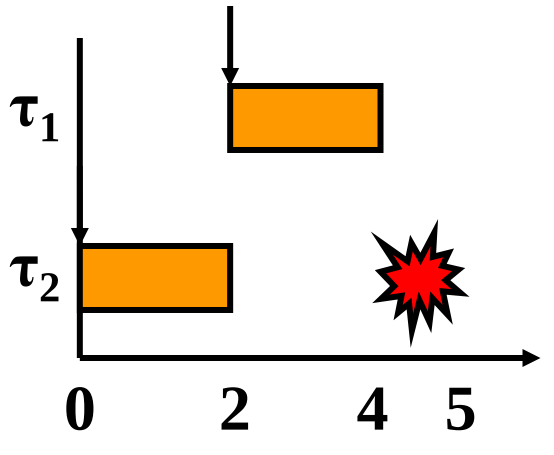
Example #1b

Task τ_i		Period T_i	Deadline D_i	Run-Time C_i	Phase φ_i	Priority π_i
τ_1	(High Priority)	8	8	3	0	3
τ_2	(Low Priority)	12	12	1	10	2
τ_3		12	12	6	0	1



Example #2

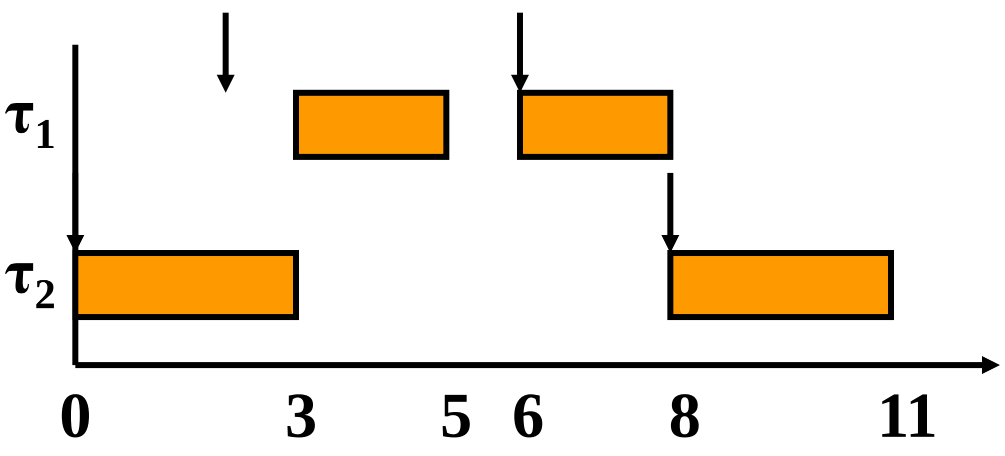
Task τ_i	Period T_i	Deadline D_i	Run-Time C_i	Phase φ_i
τ_1 (High Priority)	4	3	2	2
τ_2 (Low Priority)	8	4	3	0



If phasing (φ_i) is allowed to be greater than 0, then a Deadline Monotonic (DM) priority assignment may not be optimal.

Example #3

Task τ_i	Period T_i	Deadline D_i	Run-Time C_i	Phase φ_i
τ_1 (Low Priority)	4	3	2	2
τ_2 (High Priority)	8	4	3	0



Leung's Test

- J. Leung and J. Whitehead, "On the Complexity of Fixed Priority Scheduling of Periodic Real-Time Tasks", Performance Evaluation, 2(4):237-250, 1982.
- A task set is **feasible** if all deadlines are met in the interval $[\, s, 2P\,)$ where
  - $s = \max \{\varphi_1, \varphi_2, \ldots, \varphi_n\}$
  - $P = lcm \{ T_1, T_2, \ldots, T_n \}$
- Implicit assumption: $s \leq P$

Simple Test

- Compute $P = lcm\ \{ T_1, T_2, \ldots , T_n \}$.
- Set $s = \max \{\varphi_1, \varphi_2, \ldots , \varphi_n\}$.
- If ($s \leq P$), set $S = 0$; otherwise, set $S = \lfloor s/P \rfloor\ P$.
- Construct a schedule for the interval [S, $S + 2P$).
- Check the schedule to see if all deadlines are met.

Example #4

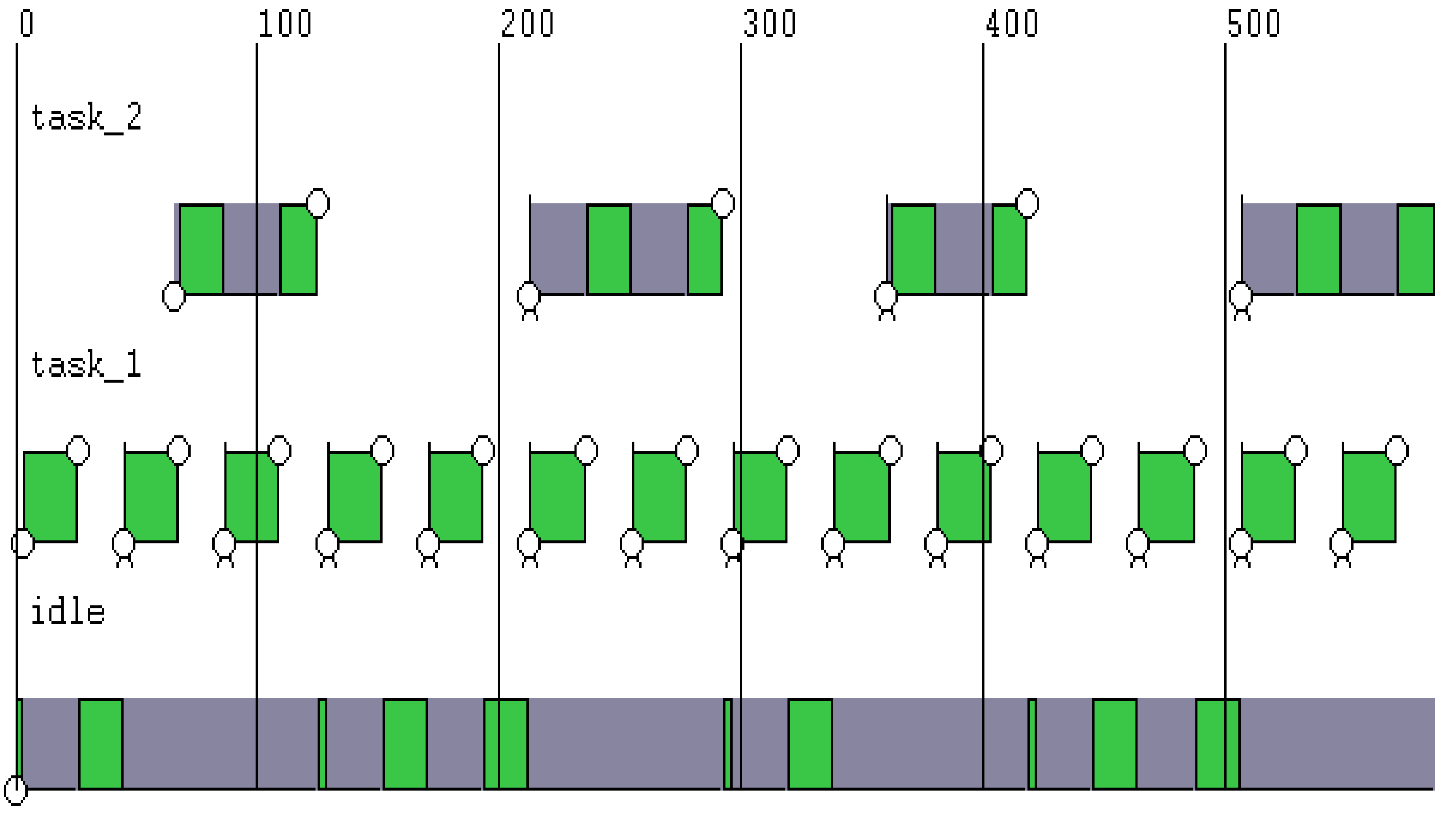
Task τ_i	Period T_i	Deadline D_i	Run-Time C_i	Phase φ_i
τ_1	42	42	23	3
τ_2	147	147	34	66

- Since $gcd(42, 147) = 21$, $P = lcm(42, 147) = 294$.
- Also, $s = 66$, so $S = 0$.
- Check for missed deadlines in [0, 588).

Stress Program Input

```
/* audsley1.str: Example From Audsley's
Paper */
system
  node node_1
    processor proc_1
      periodic task_1
        period 42 deadline 42 offset 3
        priority 1
        [23,23]
      endper
      periodic task_2
        period 147 deadline 147 offset 66
        priority 2
        [34,34]
      endper
    endpro
  endnod
endsys
```

Example: No Missed Deadlines



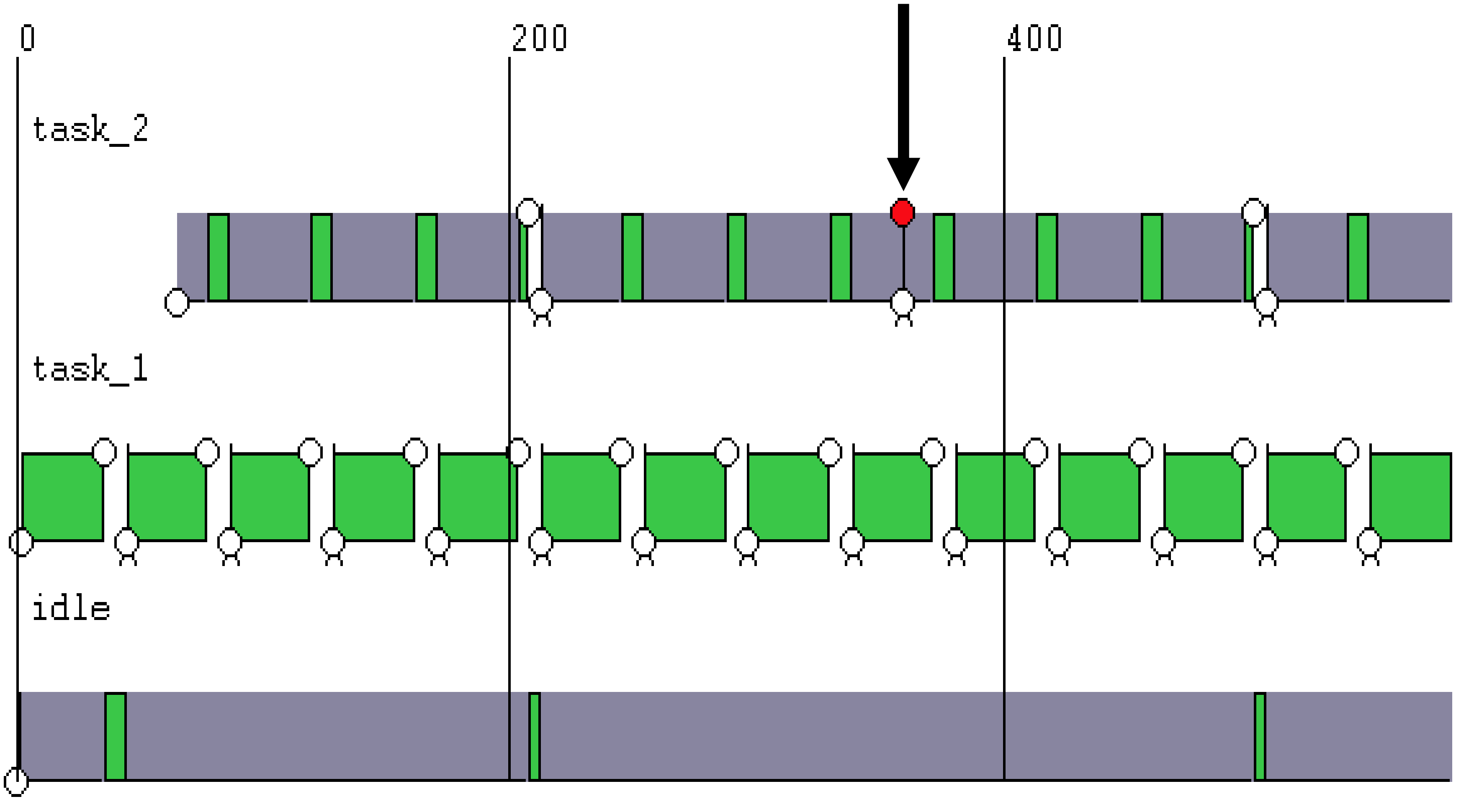
Example #5

Task τ_i	Period T_i	Deadline D_i	Run-Time C_i	Phase φ_i
τ_1	42	42	33	3
τ_2	147	147	31	66

Stress Program Input

```
/* audsley2.str: Example From Audsley's Paper */
system
  node node_1
    processor proc_1
      periodic task_1
        period 42 deadline 42 offset 3
        priority 1
        [33,33]
      endper
      periodic task_2
        period 147 deadline 147 offset 66
        priority 2
        [31,31]
      endper
    endpro
  endnod
endsys
```

Example: Missed Deadline



Summary

- Read Ch. 4-7 + Liu and Layland’s paper.
- Homework #1.
- Homework #2.