

---

# CIS 721 - Real-Time Systems

## Lecture 10: Arbitrary Deadlines

---

Mitch Neilsen  
**neilsen@cis.ksu.edu**

---

# Outline

- **Priority-Driven Scheduling**
    - **Periodic Tasks (Ch. 6)**
      - **Arbitrary Start Times**
        - **Leung's Feasibility Test**
        - **Audsley's Feasibility Test**
      - **Arbitrary Deadlines**
-

# Arbitrary Deadlines

- J.P. Lehoczky, “Fixed Priority Scheduling of Periodic Task Sets with Arbitrary Deadlines”, In Proceedings of IEEE Real-Time Systems Symposium, pp. 201-209, December, 1990.
- K. Tindell, A. Burns, and A.J. Wellings, “An extensible approach for analysing fixed priority hard real-time tasks”, Real-Time Systems, 6 (2), pp. 133-151, 1994.

# Rate Monotonic Assignment?

- **Q:** What happens if task deadlines are allowed to be greater than their periods?
- **A:** Rate Monotonic and Deadline Monotonic Priority Assignments may no longer be optimal.

# Terms and Concepts

- A **task set**  $\Gamma = \{\tau_1, \tau_2, \dots, \tau_n\}$  is a collection of related tasks.
- Each **periodic task**  $\tau_i$  is characterized by:
  - an **execution time** or **run-time** (  $C_i$  ),
  - a **period** (  $T_i$  ),
  - a **(relative) deadline** (  $D_i$  ), and
  - a **phase** or **offset** (  $\varphi_i$  or  $O_i$  ).

# Release Time

- The **release time** (or **arrival time**) of a job is the time at which the job becomes available for execution (  $r_i$  ).
- The release time of the  $j^{\text{th}}$  instant (job) of task  $\tau_i$  is given by  $r_i = O_i + (j-1) T_i$ .
- **Assumption:**  $O_i = 0$  for all  $i$ ; thus, a critical instant occurs at time 0.

# Level-i Busy Period

- A **level-i busy period** is a time interval  $[a, b]$  in which tasks of priority  $i$  or higher are processed, but no tasks of level- $i$  are processed in  $(a - \varepsilon, a)$  or  $(b, b + \varepsilon)$  for some  $\varepsilon > 0$ .

---

# Deadline Monotonic Scheduling

- If  $D_i \leq T_i$ , then Deadline Monotonic Scheduling Algorithm is optimal.
- If  $D_i > T_i$ , this may not be true.



# Example: task 1 w/ highest priority

```
/* Example */
system
  node node_1
    processor proc_1

      periodic task_1
        period 100 deadline 110 offset 0
        priority 1
        [52,52]
      endper

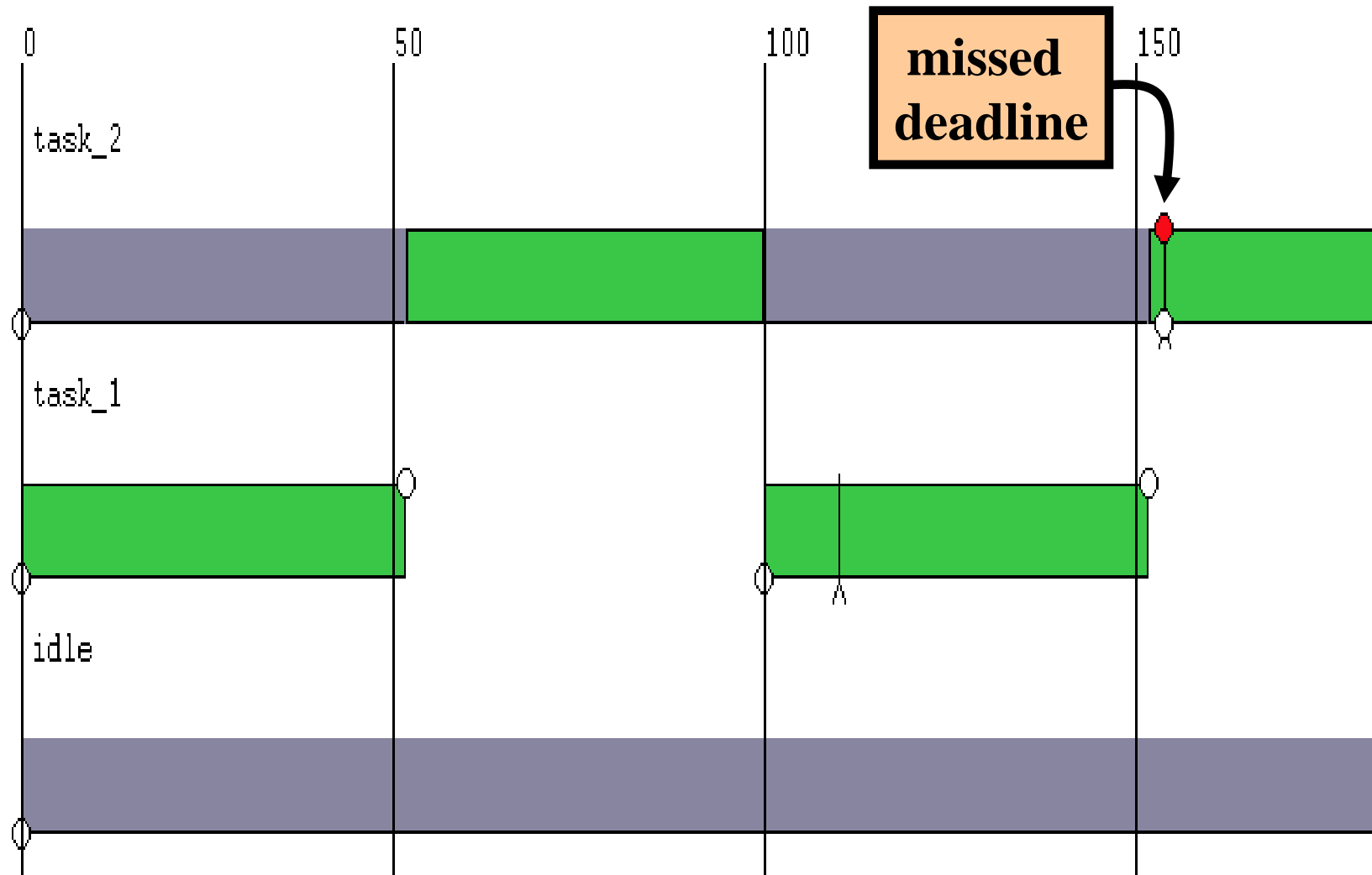
      periodic task_2
        period 140 deadline 154 offset 0
        priority 2
        [52,52]
      endper

    endpro
  endnod
endsys
```

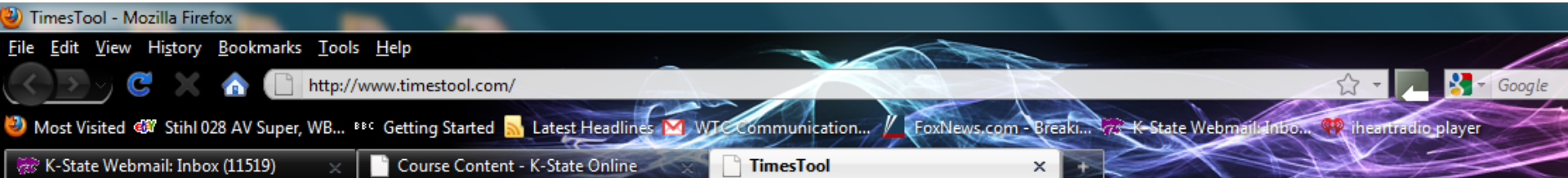


High Priority

## Example: task 1 w/ highest priority



# TimesTool - [www.timestool.com](http://www.timestool.com)



## News

### Beta Version

TIMES 1.3  
Beta now  
available  
online! Download  
[here](#).

### TIMES User Group

Has been  
created on  
YahooGroups!  
[Register now!](#)

### Tool Paper

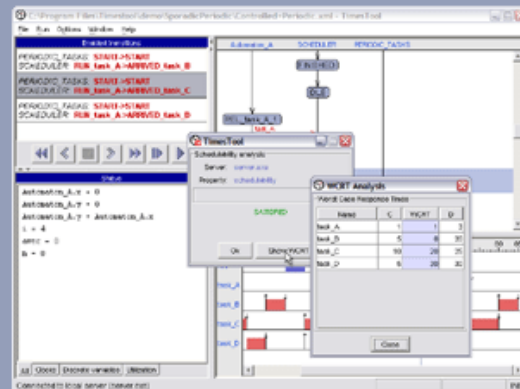
Presented  
at  
TACAS'02,  
received ETAPS'02  
Best Tool Demo  
Award.  
Available [here](#).

### Background Paper

Presented at  
TACAS'02.  
Available [here](#).

## THE TIMES TOOL

**TIMES** - A Tool for Modeling and Implementation of Embedded Systems. It is a tool set for modelling, schedulability analysis, synthesis of (optimal) schedules and executable code. It is appropriate for systems that can be described as a set of tasks which are triggered periodically or sporadically by time or external events.

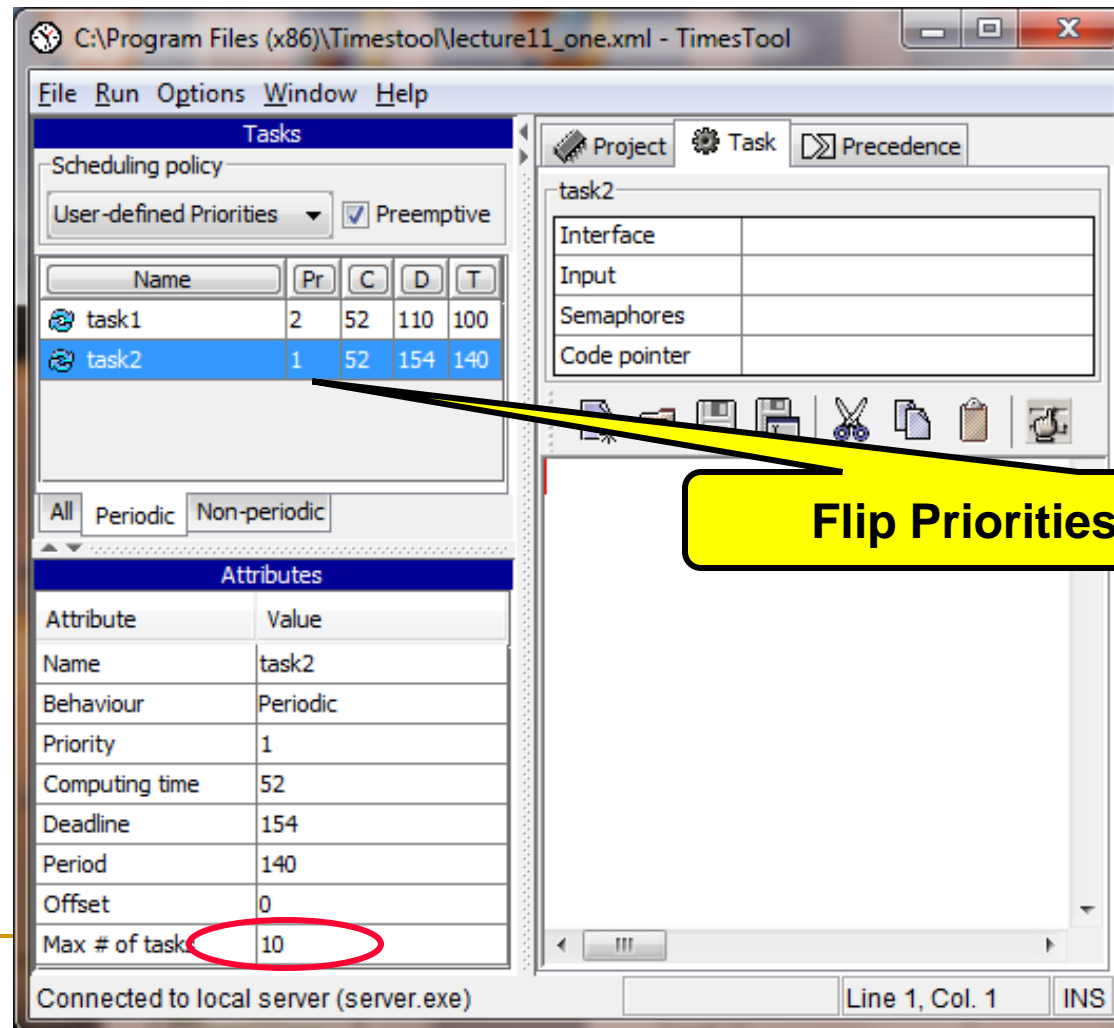


## Main Features

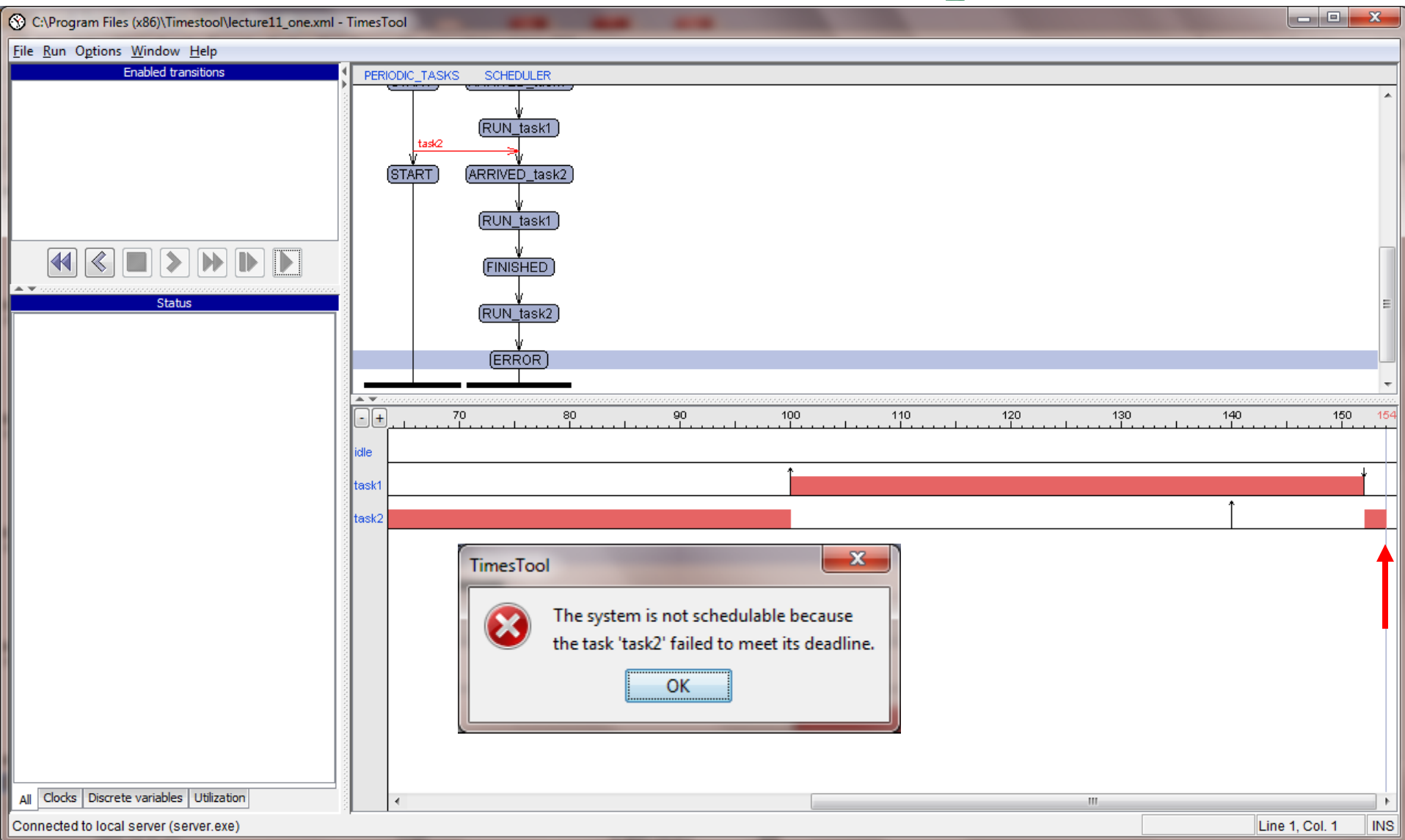
- A **graphical editor** for timed automata extended with tasks, which allows the user to model a system and the abstract behaviour of its environment. In addition the user may specify a set of preemptive or non-preemptive tasks with parameters such as (relative) deadline, execution time, priority, etc.
- A **simulator**, in which the user can validate the dynamic behaviour of the system and see how the tasks execute according to the task parameters and a given scheduling policy. The simulator shows a graphical representation of the

# Using TimesTool — `java -jar timestool.jar`

- <http://www.timestool.com> and online



# TimesTool Simulation Output



# Example: task 2 w/ highest priority

```
/* Example */
system
  node node_1
    processor proc_1

      periodic task_1
        period 100 deadline 110 offset 0
        priority 2
        [52,52]
      endper

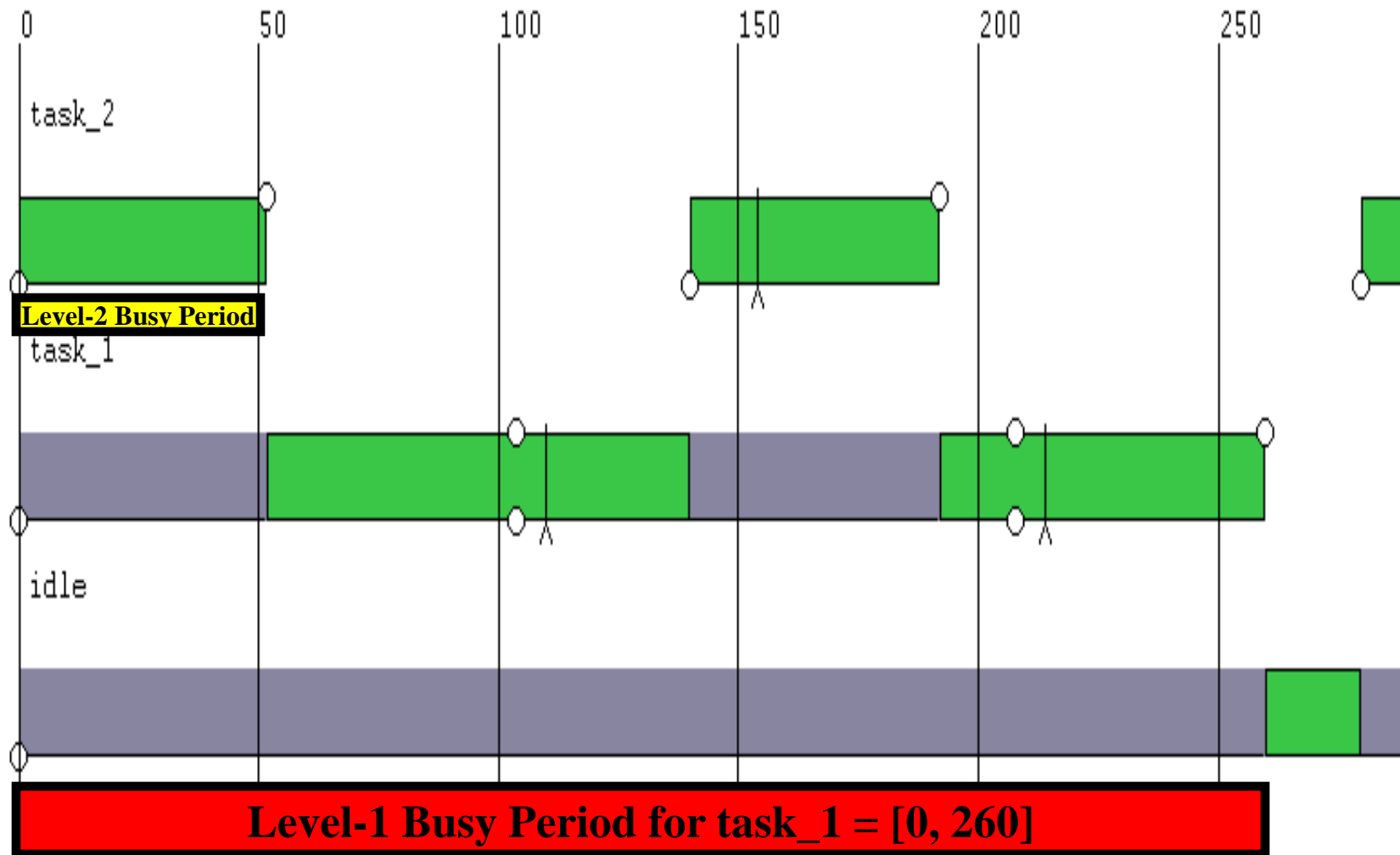
      periodic task_2
        period 140 deadline 154 offset 0
        priority 1
        [52,52]
      endper

    endpro
  endnod
endsys
```

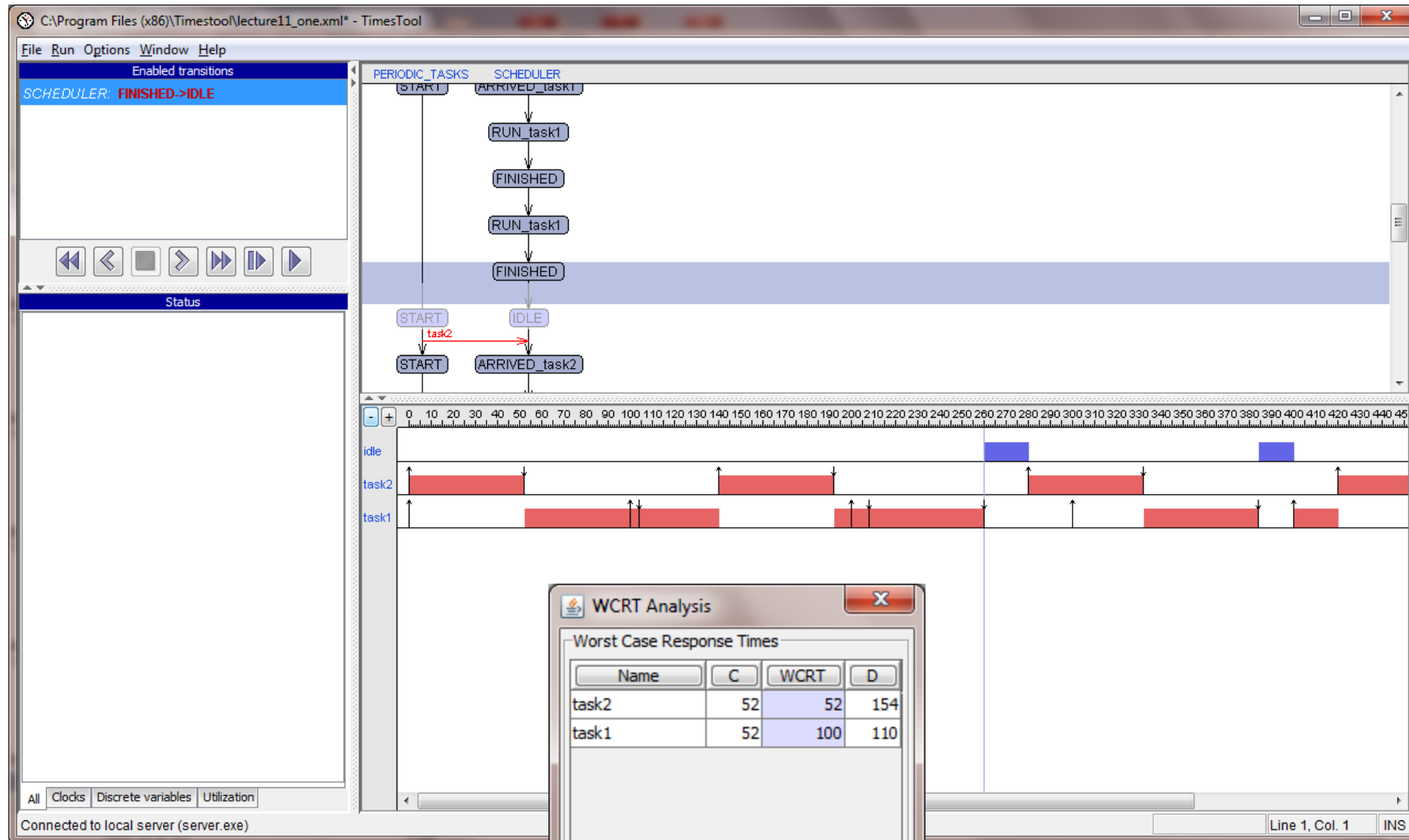


High Priority

## Example: task 2 w/ highest priority



# TimesTool Output





# Response Time Analysis

- For each (potentially overlapping) release, a worst-case completion time  $w_i(q)$  is defined by:

$$w_i^{n+1}(q) = q \cdot C_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^n(q)}{T_j} \right\rceil \cdot C_j$$

$$w_i^0(q) = C_i + (q-1) \cdot T_i$$

where  $q$  is the instance or job number and  $w_i(q)$  is the least fixed point of  $w_i^n(q)$ .

- The response time of the  $q^{th}$  instance,  $R_i(q)$ , is given by  $R_i(q) = w_i(q) - (q-1) T_i$ .

# Response Time Analysis (cont.)

- Set  $q' = \min \{ q \mid R_i(q) \leq T_i \}$ .
- Then, the **level-i busy period** is  $[ 0, w_i(q') ]$ .
- The **worst-case response time**  $R_i$  is given by:

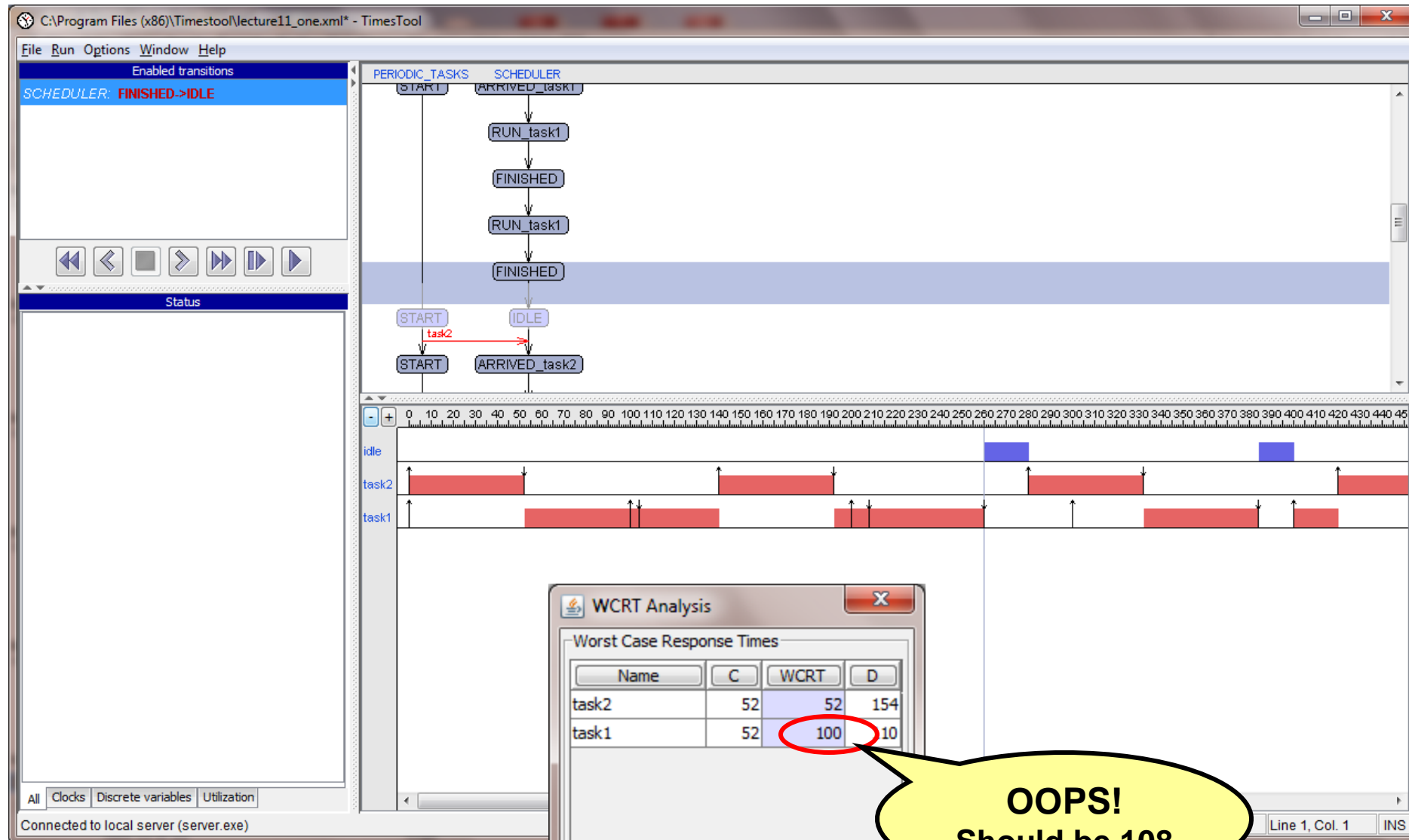
$$R_i = \max_{q=1,2,\dots,q'} \{R_i(q)\}$$

- If  $R_i \leq D_i$  for all  $i$ , the system is **schedulable**.

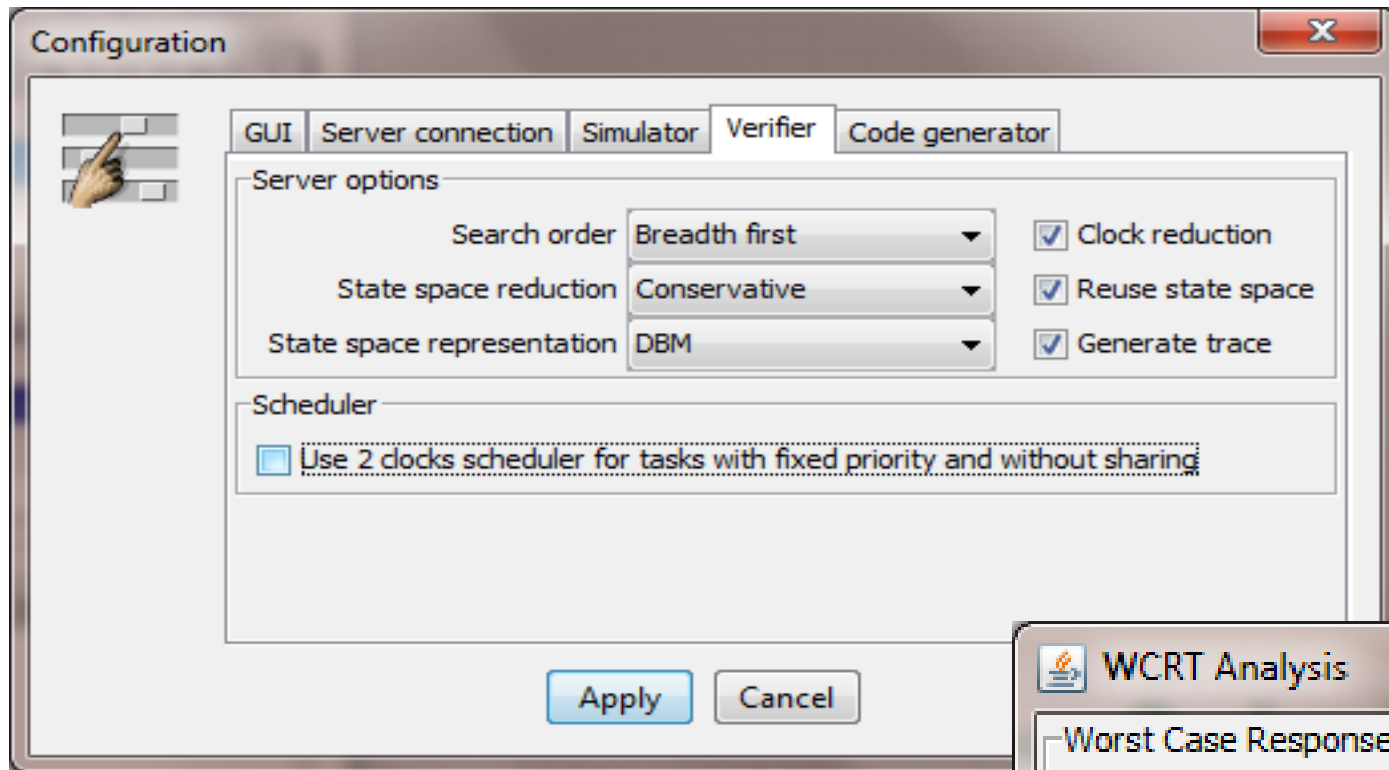
# Example #9

Task	Period	Deadline	Run-Time	Phase
$\tau_i$	$T_i$	$D_i$	$C_i$	$\phi_i$
<hr/>				
$\tau_1$	100	110	52	0
$\tau_2$	140	154	52	0

# TimesTool Output



# Hack: Don't use 2 clocks scheduler...

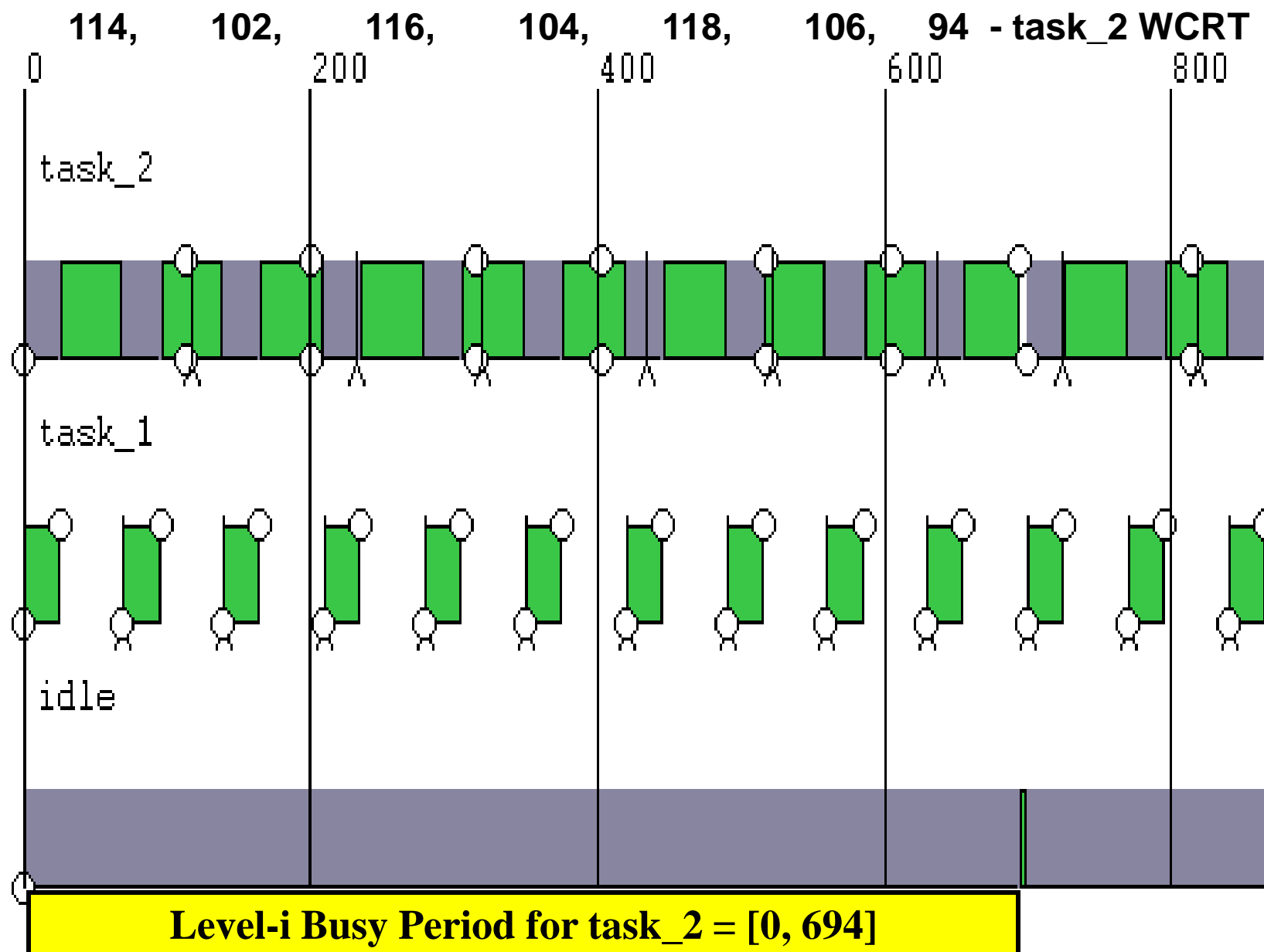


The WCRT Analysis dialog box displays a table titled 'Worst Case Response Times'. The table has four columns: 'Name', 'C', 'WCRT', and 'D'. It contains two rows of data for 'task1' and 'task2'. The 'WCRT' column is highlighted in blue. A 'Close' button is at the bottom.

Name	C	WCRT	D
task1	52	108	110
task2	52	52	154

# Lehoczky's Example

Task	Period	Run-Time	Phase	Deadline
$\tau_i$	$T_i$	$C_i$	$\phi_i$	$D_i$
<hr/>				
$\tau_1$	70	26	0	68
$\tau_2$	100	62	0	118



---

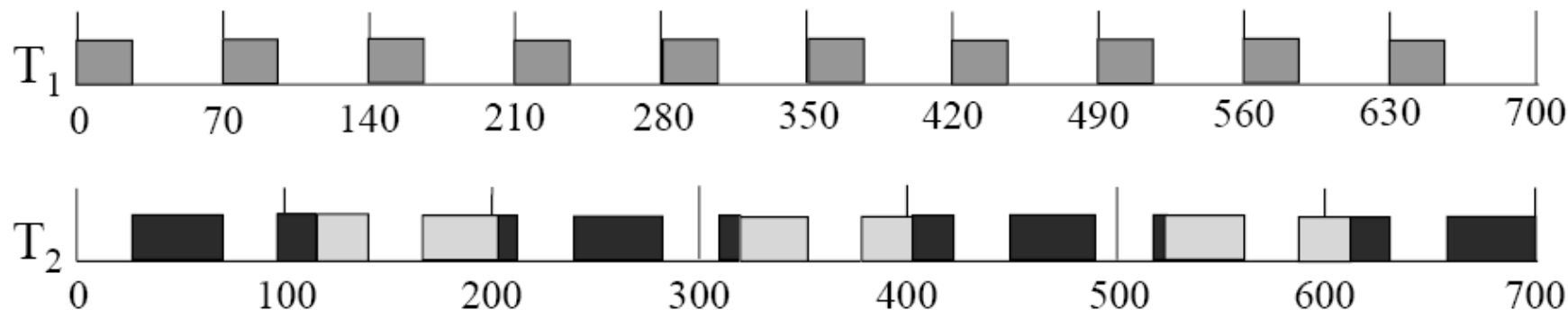
# General Time-Driven Analysis

- Check to see if the first job completes before it's deadline and before the second job in the same task is released.
  - If not, check all jobs over a level-i busy period.
-



# Example

$$T_1 = (70, 26), T_2 = (100, 62)$$



$W_{2,1} = \text{minimum } t \text{ s.t.}$

$$t = w_{2,1}(t)$$

$$= e_2 + \sum_{k=1}^{i-1} \left\lceil \frac{t}{p_k} \right\rceil \cdot e_k$$

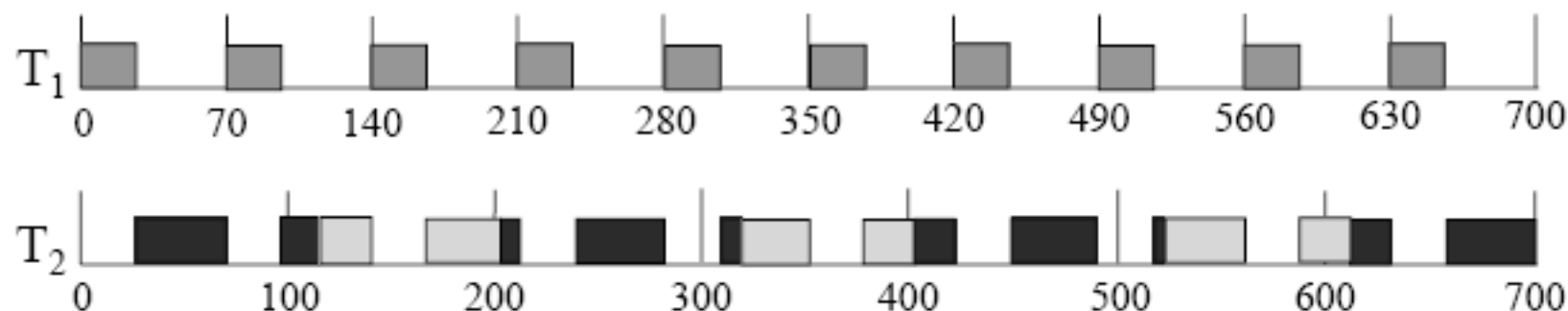
$$= 62 + \left\lceil \frac{t}{70} \right\rceil \cdot 26$$

$$?? 114 = 62 + \left\lceil \frac{114}{70} \right\rceil \cdot 26$$

$$= 62 + 2 \cdot 26$$

$$= 114 \quad \text{Yes!}$$

## Example (cont.)



$W_{2,2} = \text{minimum } t \text{ s.t.}$

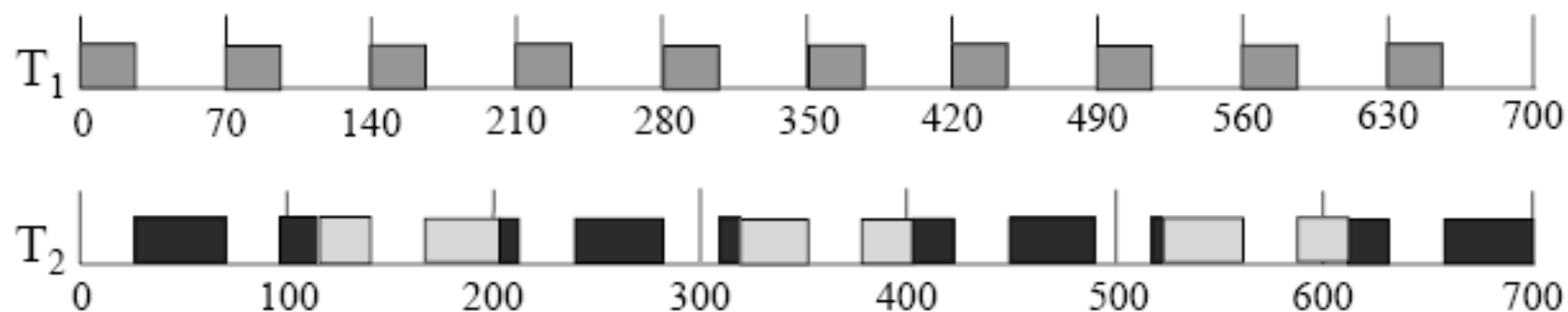
$$t = w_{2,2}(t + p_2) - p_2$$

$$= 2 \cdot e_2 + \sum_{k=1}^{i-1} \left\lceil \frac{t+100}{p_k} \right\rceil \cdot e_k - 100$$

$$= 124 + \left\lceil \frac{t+100}{70} \right\rceil \cdot 26 - 100$$

$$\begin{aligned} ?? 102 &= 124 + \left\lceil \frac{202}{70} \right\rceil \cdot 26 - 100 \\ &= 124 + 3 \cdot 26 - 100 \\ &= 102 \quad \text{Yes!} \end{aligned}$$

## Example (cont.)



$W_{2,3} = \text{minimum } t \text{ s.t.}$

$$t = w_{2,3}(t + 2 \cdot p_2) - 2 \cdot p_2$$

$$= 3 \cdot e_2 + \sum_{k=1}^{i-1} \left\lceil \frac{t + 200}{p_k} \right\rceil \cdot e_k - 200$$

$$= 186 + \left\lceil \frac{t + 200}{70} \right\rceil \cdot 26 - 200$$

$$\begin{aligned} ??116 &= 186 + \left\lceil \frac{316}{70} \right\rceil \cdot 26 - 200 \\ &= 186 + 5 \cdot 26 - 200 \\ &= 116 \quad \text{Yes!} \end{aligned}$$

---

# Summary

- Read Ch. 5-7, next time start on Ch. 8.
  - Read Audsley's paper on scheduling with arbitrary start times.
  - Read Lehoczky's paper on scheduling with arbitrary deadlines.
-