Lecture Note #5: Task Scheduling (2)
EECS 571
Principles of Real-Time Embedded Systems

Kang G. Shin
EECS Department
University of Michigan

Priority-Driven Scheduling of Periodic Tasks

- **Why priority-driven scheduling?**
  - use priority to represent urgency/importance
  - easy implementation of scheduler (compare task priorities and dispatch tasks accordingly)
  - tasks can be added or removed easily
  - no direct control of execution instant
- **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 a single processor**
  - Independent periodic tasks
  - Relative deadline = period
  - Preemptable without any limit
  - no overhead for context switch

Priority-Driven Schedules

- **Assign priority when jobs arrive**
  - static -- all jobs of a periodic task have the same fixed priority
  - dynamic -- different priorities to individual jobs of a periodic task
  - relative priorities don’t change while jobs are waiting for execution
- **Static priority schedules**
  - Rate-monotonic (RM) -- the higher the task frequency, the higher its priority
  - Deadline-monotonic (DM) – the shorter relative deadline, the higher priority
- **Dynamic priority schedules**
  - EDF -- earliest deadline first
  - LSTF (MLF)-- least slack time (laxity) first
- **Schedulable utilization:**
  - a scheduling algorithm can feasibly schedule any set of priority tasks if the total utilization is equal to or less than its *schedulable utilization*

EDF Schedule

- Optimal for uniprocessor systems and preemptable tasks
- How do we know if a set of periodic tasks are schedulable under EDF?
- If we know the schedulable utilization S_U of EDF, then any set of tasks is schedulable as long as $U \leq S_U$
- Theorem: A set of n periodic tasks can be scheduled by EDF iff

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

- Proof
  - the only-if part is obvious
  - the if part --- show if there is a job misses its deadline, then $U > 1$

Extension of EDF Schedulable Utilization

- **If $D_i \geq p_{i,}$ *EDF* is schedulable iff $U \leq 1$**
- **What can we do if $D_i < p_i$**
  - density of task *k* : $\delta_k = e_k / min(p_{k,} D_k)$
  - *EDF* is schedulable if the total density is equal to or less than 1
  - proof: if there is a job missing its deadline, then the total density > 1
  - there is no “only-if” part ---- if the total density > 1, *EDF* may or may not be schedulable
- **If $D_i \geq p_{i,}$ *LSTF* is schedulable iff $U \leq 1$**
- **Predictable for uniprocessor preemptive scheduling of independent tasks**
- **Robust**
  - independent of phases
  - *periods are lower bound* $\Rightarrow$ applicable to sporadic tasks with minimum separations

Example of EDF Schedule

- **A digital robot with EDF schedule**
  - control loop: $e_c \leq$ 8ms at 100Hz
  - BIST (Built-In-Self-Testing): $e_b \leq$ 50ms
  - 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 $e_t \leq$ 15ms**
  - if BIST is done every 1000ms
  - given $$u_c + u_b + ut = \frac{8}{10} + \frac{50}{1000} + \frac{15}{D_t} \leq 1$$
  - the telemetry task can have a relative deadline of 100ms

    $\Rightarrow$ sending or receiving must be separated by at least 100ms

Rate-Monotonic Scheduling Algorithm Liu and Layland 1973

- **A base case: no additional overhead, simple periodic tasks with $p_i = D_i$**
- **Assign priorities according to their periods**
  - T_i has a higher priority than T_k if $i < k$ ($p_i < p_k$)
  - Is RM optimal? $\Rightarrow$ if there is a feasible fixed-priority schedule, then so is RM
  - How do we know RM is feasible $\Rightarrow$ schedulability test
- **Results:**
  - RM is optimal if $p_i \geq D_i$
  - sufficient condition $\Rightarrow$ *utilization test*

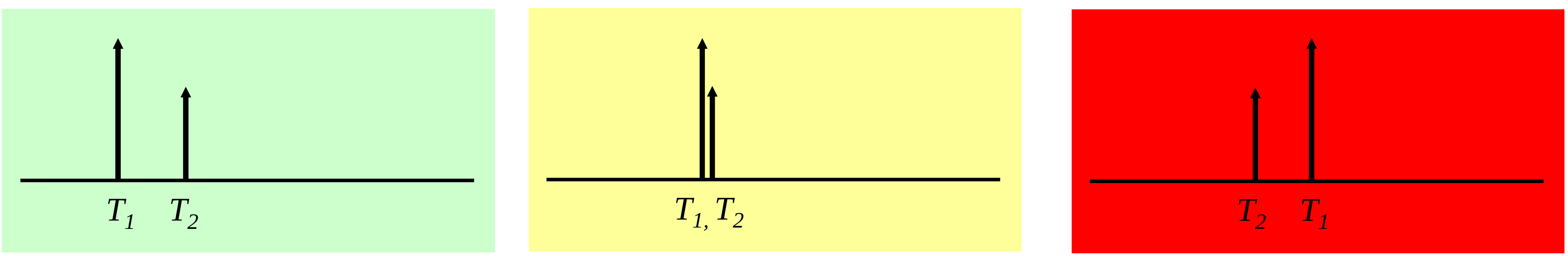
$$U = \sum_{i=1}^{n} \frac{e_i}{p_i} \leq n(2^{1/n} - 1)$$

  - a complete test $\Rightarrow$ *what is the worst-case response time given all possible arrivals and preemptions*

Critical Instant

- **Critical instant** **of T_i: a job of T_i arriving at the critical instant** x^* **has a maximum response time, i.e.,** $R_i(x^*) \geq R_i(x), \forall x$ where $R_i(x)$ is response time of task T_i that arrived at time x
- **If we can find the critical instant of T_i, then**
  - *check whether all jobs of T_i meet their deadlines*
  - *let's increase e_i until the maximum response time = D_i*

    $\Rightarrow$ *schedulable utilization*
- **In-phase instant is critical: all higher priority tasks are released at the same instant of $J_{i,c}$ (assume all jobs are completed before the next job is released.)**
  - *which T_2 has the maximum response time?*



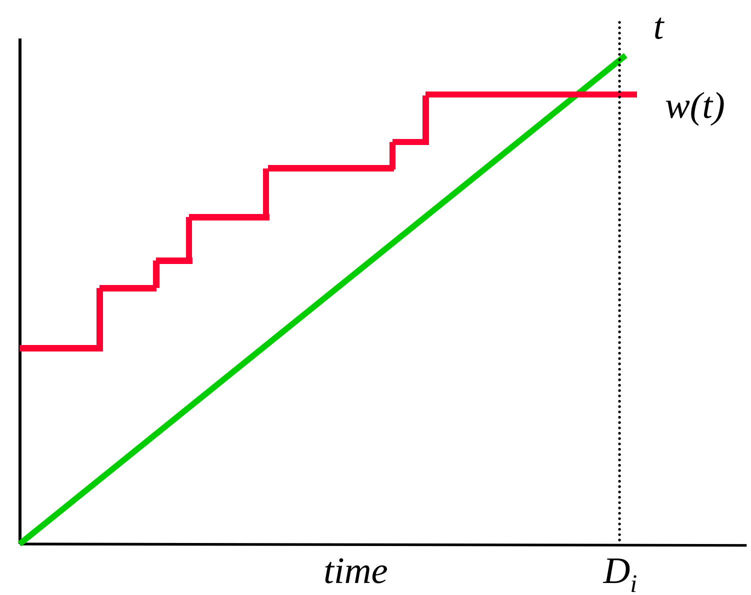
Schedulability Test: Time-Demand Analysis

- Consider in-phase instant only
- **If J_i is done at t, then the total work must be done in $[0,t]$ is (from J_i and all higher priority tasks): time demand function**

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

- **Can we find a $t \le D_i$ such that $w_i(t) \le t$**
  - *cannot check all $t \in [0, D_i]$*
  - *check all arrival instants and D_i*
- **The completion time of J_i satisfies**

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



Schedulability Test

- EDF has a schedulable utilization of 1, how about RMS?
- If $D_i = p_i$, the schedulable utilization exists
  - if $U \leq n(2^{1/n} - 1)$, *done*
  - else do time-demand analysis

- if $D_i < p_i$, do time-demand analysis

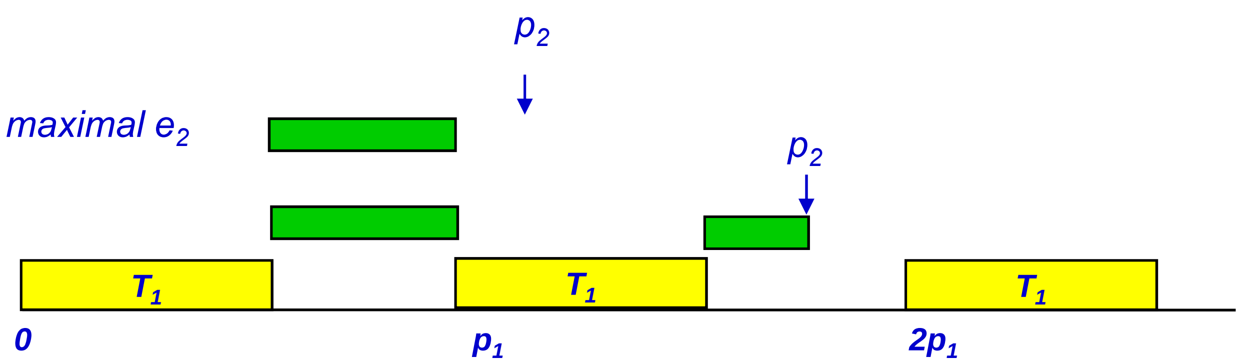
- if $D_i > p_i$, there may be more than one job of task *i* in the system
  - examine *all jobs of task i* in a level-*i* busy interval (in-phase)
  - the following equations represent this case:

$$w_{i,j}(t) = j\,e_i + \sum_{k=1}^{i-1} \left\lceil \frac{t}{p_k} \right\rceil e_k \quad \text{for } (j-1)p_i < t \leq w_{i,j}(t)$$

$$t = w_{i,j}(t - (j-1)p_i) - (j-1)p_i$$

Schedulable Utilization of RMS

- **Must be less than 1**
- **Let's consider two tasks and relative deadline=period**
  - T_2 can only be executed when T_1 is not in the system
- **Let $p_2 < 2p_1$. *Determine the maximum schedulable e_2***
  - *If $p_2 < p_1+e_1$, $max(e_2)=p_1-e_1$* $\Rightarrow U=e_1/p_1+(p_1-e_1)/p_2$
  - *Else, $max(e_2)=p_2-2e_1$* $\Rightarrow U=e_1/p_1+(p_2-2e_1)/p_2$



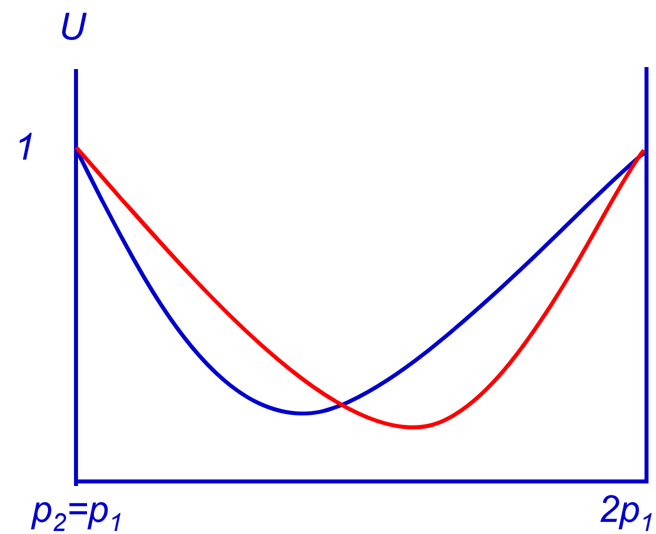
Schedulable Utilization of RMS

- **Given e_1, p_1, and p_2, plot U**
- **The minimum U occurs when**

  $$p_2=p_1+e_1$$

  where

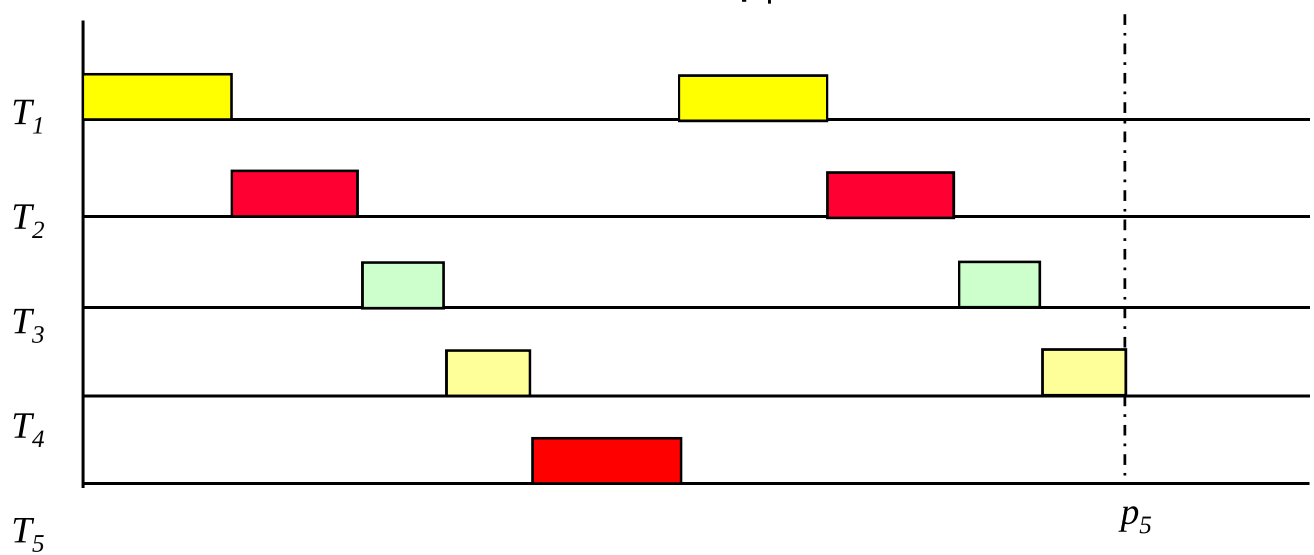
  $$U= e_1/p_1+(p_1-e_1)/(p_1+e_1)$$



- ***What is the minimum U ?***
  - *take the derivative wrt to p_1 and set $dU/dp_1=0$*
  - *we will get $e_1=(2^{1/2}-1)p_1$ and $U=0.828\ldots$*

Schedulable Utilization of RMA

- $U \le n\,(2^{1/n} - 1)$
- **Is there a case that is feasible and gives the minimum schedulable utilization**
- **When $p_n \le 2\,p_1$**
  - processor must be busy in $[0, p_n]$
  - become unscheduable if we increase any e_i
  - processor will be idle if we increase p_i

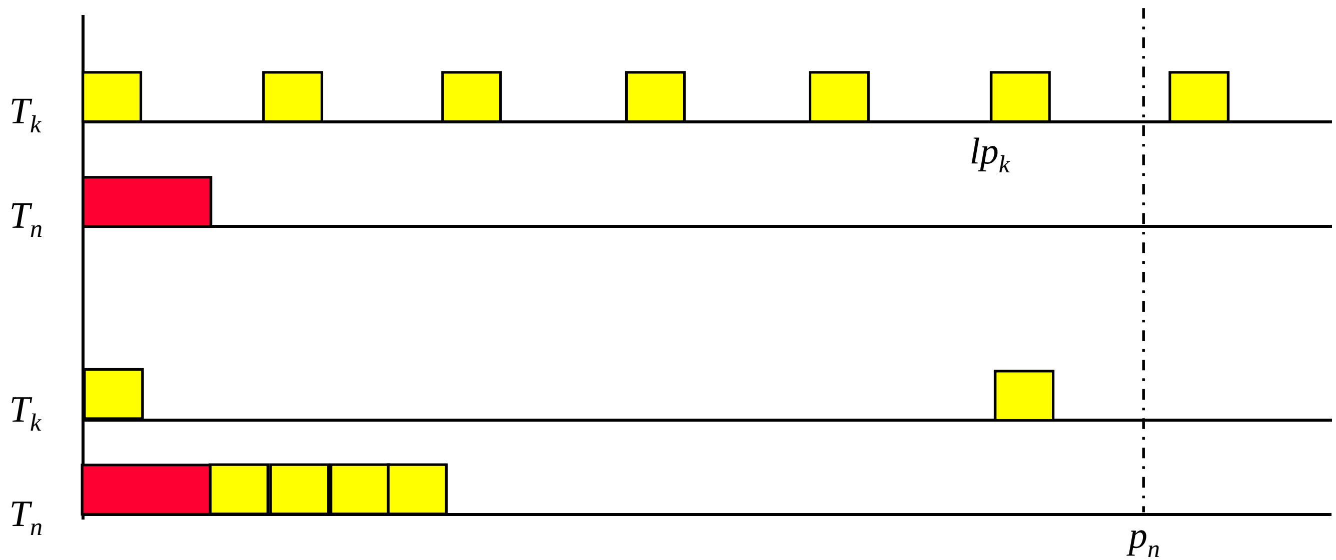


Schedulable Utilization of RMA

- **What do we have from the timeline diagram?**
  - $e_k = p_{k+1} - p_k$ *for k=1,2,..,n-1*
  - $e_n = p_1 - 2(e_1 + e_2 + ... + e_{n-1}) = 2p_1 - p_n$
  - $$U = (\frac{p_2}{p_1} - 1) + (\frac{p_3}{p_2} - 1) + ... + (\frac{p_n}{p_{n-1}} - 1) + (\frac{2p_1}{p_n} - 1)$$
    $$= q_{2,1} + q_{3,2} + ... + q_{n,n-1} + \frac{2}{q_{2,1}q_{3,2}.....q_{n,n-1}} - n$$
- **Can we increase e_1 and decrease e_k by the same amount**
  - still schedulable for the 1st arrival of all tasks
  - utilization is higher
- **Can we decrease e_1 and increase e_k**
- **When will U be minimum ? ----- when $q_{2,1}=q_{3,2}= . . . = 2/n$**

Schedulable Utilization of RMA

- **When $p_n > 2p_k$ and schedulable**
  - construct a new task set that is schedulable and $p_n \leq 2\, p_k$
  - the original set has a higher utilization



Schedulability: Response time test

- **Theorem: for a set of independent, periodic tasks, if each task meets its first deadline, with worst-case task phasing, the deadline will always be met.**
- **Response Time (RT) test: let a_n = response time of task *i*. a_n may be computed by the following iterative formula:**

$$a_{n+1} = e_i + \sum_{j=1}^{i-1} \left\lceil \frac{a_n}{p_j} \right\rceil e_j \qquad \text{where} \quad a_0 = \sum_{j=1}^{i} e_j$$

  - Test terminates when $a_{n+1} = a_n$
  - Task *i* is schedulable if its response time is before its deadline:

    $a_n \leq p_i$

Necessary and Sufficent RM-Schedulability

Theorem 3.4. **Given a set of n periodic tasks with $P_1 \leq P_2 \leq \cdots \leq P_n$, task T_i can be feasibly scheduled using RM iff $L_i = \min_{0<t \leq P_i} W_i(t)/t \leq 1$, where $W_i(t) = \sum_{j=1}^{i} e_j \lceil t/P_j \rceil$.**

Practical Question: How to check for $W_i(t) \leq t$ easily?

- Only need to compute $W_i(t)$ at $\tau_i = kP_j$, $j=1,2,\cdots, i$; $k= 1,\cdots, \lfloor P_i/P_j \rfloor$
- Two RM-schedulability conditions:
  - If $\min_{t \in \tau_i} W_i(t) \leq t$, then T_i is RM-schedulable.
  - If $\max_{i \in \{1,\cdots,n\}} \min_{t \in \tau_i} W_i(t)/t \leq 1$, then the entire task set is RM-schedulable.

Example: UB Test

	e	p	U
Task τ_1	*20*	*100*	*0.200*
Task τ_2	*40*	*150*	*0.267*
Task τ_3	*100*	*350*	*0.286*

- **Total utilization is**

  $$.200 + .267 + .286 = .753 < U(3) = .779$$

- **The periodic tasks in the example are schedulable according to the UB test.**

Example: Applying RT Test (1)

- **If we increase the compute time of τ_1 from 20 to 40; is the task set still schedulable?**
- **Utilization for the first task : 40/100=0.4 < U(1)**
- **Utilization of first two tasks: 0.667 < U(2) = 0.828**
  - First two tasks are schedulable by UB test
- **Utilization of all three tasks: 0.953 > U(3) = 0.779**
  - UB test is inconclusive
  - Need to apply RT test

Example: Applying RT Test (2)

- Use RT test to determine if τ_3 meets its first deadline:

  $i = 3$

$$a_0 = \sum_{j=1}^{3} e_j = e_1 + e_2 + e_3 = 40 + 40 + 100 = 180$$

$$a_1 = e_i + \sum_{j=1}^{i-1} \left\lceil \frac{a_0}{p_j} \right\rceil e_j = e_3 + \sum_{j=1}^{2} \left\lceil \frac{a_0}{p_j} \right\rceil e_j$$

$$= 100 + \left\lceil \frac{180}{100} \right\rceil (40) + \left\lceil \frac{180}{150} \right\rceil (40) = 100 + 80 + 80 = 260$$

Example: Applying RT Test (3)

$$a_2 = e_i + \sum_{j=1}^{i-1} \left\lceil \frac{a_1}{p_j} \right\rceil e_j = 100 + \left\lceil \frac{260}{100} \right\rceil (40) + \left\lceil \frac{260}{150} \right\rceil (40) = 300$$

$$a_3 = e_i + \sum_{j=1}^{i-1} \left\lceil \frac{a_2}{p_j} \right\rceil e_j = 100 + \left\lceil \frac{300}{100} \right\rceil (40) + \left\lceil \frac{300}{150} \right\rceil (40) = 300$$

$a_3 = a_2 = 300$ Done!

- **Task is schedulable using RT test.**
  - $a_3 = 300 < p_3 = 350$

(Another) Example 3.6

- **Task set: $\{T_1, T_2, T_3, T_4\} = \{ (20, 100), (30, 150), (80, 210), (100,400)\}$.**
- **Sets of time points of interest:**

  $\tau_1 = \{100\}$

  $\tau_2 = \{100, 150\}$

  $\tau_3 = \{100, 150, 200, 210\}$

  $\tau_4 = \{100, 150, 200, 210, 300, 400\}$

- **Schedulability conditions:**
  - T_1 is RM-schedulable iff $e_1 \leq 100$
  - T_2 is RM-schedulable iff

    $e_1 + e_2 \leq 100$ **OR**

    $2e_1 + e_2 \leq 150$
  - T_3 is RM-schedulable iff

    $e_1 + e_2 + e_3 \leq 100$ **OR**

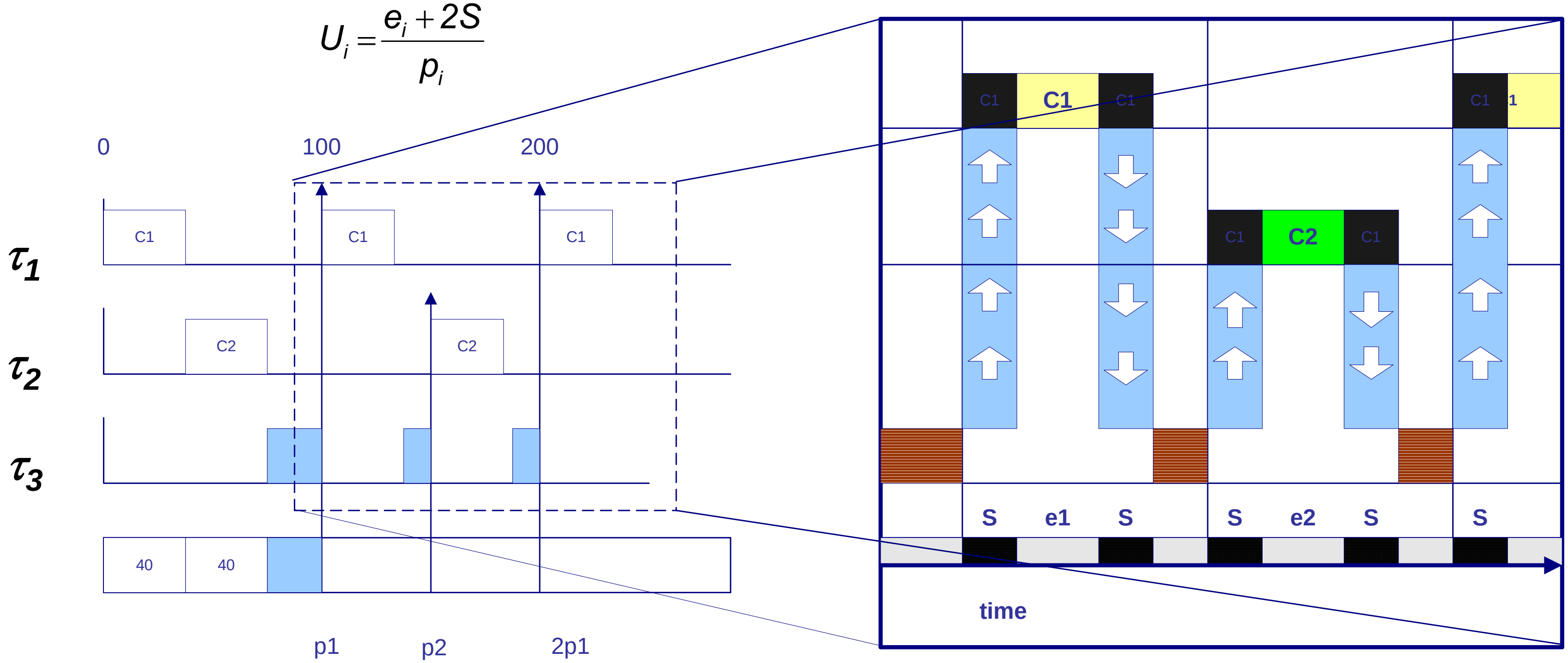
    $2e_1 + e_2 + e_3 \leq 150$ **OR**

    $2e_1 + 2e_2 + e_3 \leq 200$ **OR**

    $3e_1 + 2e_2 + e_3 \leq 210$
  - T_4 is RM-schedulable iff $\cdots$

Modeling Task-Switching

$$U_i = \frac{e_i + 2S}{p_i}$$



Two scheduling actions per task
(start of period and end of period)

Sporadic Tasks

- **Sporadic tasks have a min interarrival interval**
- **For purpose of schedulability analysis:**
  - Consider them periodic, or
  - Use a *periodic polling server* (PPS) to ``serve'' sporadic tasks, or
  - Use a *deferred server* (DS):

    Schedulability condition

    $$\mathbf{U \leq 1\text{-}U_s \text{ if } U_s \leq 0.5}$$

    $$\mathbf{U \leq U_s \text{ if } U_s > 0.5}$$

  - What shall we do if there are no sporadics to execute?
    - PPS: Keep CPU idle
    - DS: Execute other tasks.

Transient Overload

- **Question**: **What if the task with a smaller period is not important to the underlying application?**
- **Answer**: **Consider *period transformation,* period aggregation or period splitting**
- **Example**: **Consider the following *unschedulable* task set:**

T_i	e_i	a_i (avg exec time)	P_i	comments
T_1	20	10	100	
T_2	30	25	150	
T_3	80	40	210	non-critical
T_4	100	20	400	

Transient Load, cont'd

- Solution 1: reduce T_3's priority by lengthening its period, possible only if T_3's relative deadline can be greater than its original period. In such a case, replace T_3 by two tasks T_3' and T_3'', each with period 420, WC exec times $e_3' = e_3'' = 80$, avg exec times $a_3' = a_3'' = 40$. T_3' and T_3'' must be phased to be released 210 time units apart. If the set $\{T_1, T_2, T_3', T_3'', T_4\}$ is RM-schedulable, done.

- Solution 2: increase T_4's priority by splitting each invocation into two: T_4': $e_4' = e_4/2$, $a_4' = a_4/2$, $P_4' = P_4/2$
- $\{T_1, T_2, T_4\}$ or $\{T_1, T_2, T_4'\}$ are schedulable

Period Transformation

- When the task set T is RM-unscheduable, PT decomposes $T = C \cup NC$, where

  C = {all critical tasks} $\cup$ {some non-criticals}

  NC = {remaining non-criticals}

- $P_{c,max} \leq P_{n,min}$

- C is RM-schedulable under worst-case execution times.

Summary of discussion so far

- ***System model parameters***: **task and processor sets, task precedence constraints, task release and execution times, deadlines, periods, ⋯**
- **Problem: Find a feasible schedule**
- **Rate monotonic scheduling of periodic tasks without precedence constraints or resource requirements**
- **Sufficient RM-schedulability condition:**
  $U = \sum_{i=1}^{n} e_i/P_i \leq n\,(2^{1/n} - 1) \rightarrow 0.69$ **as** $n \rightarrow \infty$.
- **Necessary and sufficient RM-schedulability condition:** T_i **is schedulable iff the equation** $t = \sum_{j=1}^{i} e_j \lceil t/P_j \rceil$ **has a solution for** $t < P_i$.
- **Handling sporadic tasks: periodic polling server, deferred server**
- **Transient overload and period transformation.**

Schedulability with Interrupts

- **Interrupt processing can be inconsistent with RM priority assignment.**
  - interrupt handler executes with higher priority irrespective of its period
  - interrupt processing may delay execution of tasks with shorter periods
- **Effects of interrupt processing must be accounted for in schedulability model.**

Task(i)	*Period(p)*	*WCET(e)*	*Priority*	*Deadline(D)*
τ_3	*200*	*60*	*HW*	*200*
τ_1	*100*	*20*	*High*	*100*
τ_2	*150*	*40*	*Medium*	*150*
τ_4	*350*	*40*	*Low*	*350*

UB Test with Interrupt Priority

- Test is applied to **each** task
- Determine effective utilization (f_i) of each task *i* using

$$f_i = \sum_{j \in H_n} \frac{e_j}{p_j} + \frac{e_i}{p_i} + \frac{1}{p_i} \sum_{k \in H_1} e_k$$

Preemption from the tasks that can hit more than once (with period less than D_i)

Execution of a task under test

Preemption from tasks that can hit only once (with period greater than D_i)

- Compare effective utilization against bound, ***U(n).***

$$n = num(H_n) + 1$$

$num(H_n)$ = the number of tasks in the set H_n

UB Test with Interrupt Priority: τ_3

- **For τ_3, no tasks have higher priority:**
  $H = H_n = H_1 = \{\ \}$.

$$f_3 = \quad 0 + \frac{e_3}{p_3} + 0 \leq U(1)$$

- **Note that utilization bound is *U(1):* num(H_n) = 0.**

$$f_3 = \frac{e_3}{p_3} = \frac{60}{200} = 0.3 < 1.0$$

UB Test with Interrupt Priority: τ_1

- **For τ_1, τ_3 has priority over τ_1 :** $H = \{\tau_3\}$; $H_n = \{\}$; $H_1 = \{\tau_3\}$.

$$f_1 = 0 + \frac{e_1}{p_1} + \frac{1}{p_1}\sum_{k=3} e_k \le U(1)$$

- **Utilization bound is** *U(1)* since **num(*H*$_n$) = 0.**

$$f_1 = \frac{e_1}{p_1} + \frac{e_3}{p_1} = \frac{20}{100} + \frac{60}{100} = 0.800 < 1.0$$

UB Test with Interrupt Priority: τ_2

- For τ_2 : $H=\{\tau_1, \tau_3\}; H_n=\{\tau_1\}; H_1=\{\tau_3\};$

$$f_2 = \sum_{j=1} \frac{e_j}{p_j} + \frac{e_2}{p_2} + \frac{1}{p_2} \sum_{k=3} e_k \le U(2)$$

- **Note that utilization bound is** $U(2)$: **num(*H*$_n$) = 1.**

$$\mathrm{f}_2 = \frac{e_1}{p_1} + \frac{e_2}{p_2} + \frac{e_3}{p_3} = \frac{20}{100} + \frac{40}{150} + \frac{60}{200} = 0.867 > 0.828$$

UB Test with Interrupt Priority: τ_4

- $H=\{\tau_1, \tau_2, \tau_3\}$; $H_n=\{\tau_1, \tau_2, \tau_3\}$; $H_1=\{\}$;

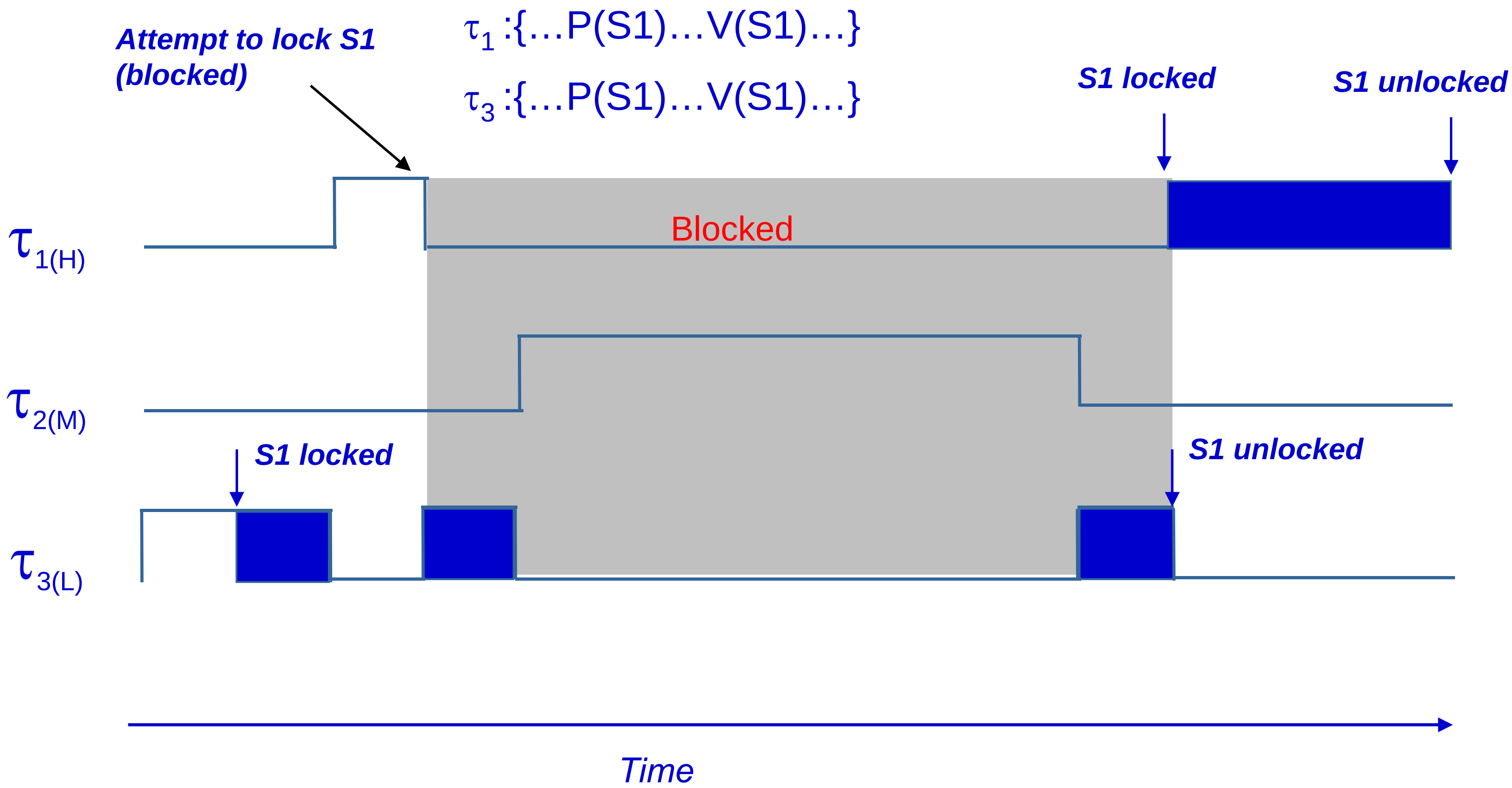
$$f_4 = \sum_{j=1,2,3} \frac{e_j}{p_j} + \frac{e_4}{p_4} + 0 \le U(4)$$

- **Note that utilization bound is *U(4):* num(*H_n*) = 3.**

$$f_4 = \frac{e_1}{p_1} + \frac{e_2}{p_2} + \frac{e_3}{p_3} + \frac{e_4}{p_4}$$

$$= \frac{20}{100} + \frac{40}{150} + \frac{60}{200} + \frac{60}{350} = 0.882 > 0.756$$

Priority Inversion in Synchronization



Priority Inversion

- **Delay to a task's execution caused by interference from, or blocking by, lower priority tasks is known as *priority inversion***
- **Priority inversion is modeled by blocking time**
- **Identifying and evaluating the effect of sources of priority inversion is important in schedulability analysis**
- **Sources of priority Inversion**
  - Synchronization and mutual exclusion
  - Non-preemtable regions of code
  - FIFO (first-in-first-out) queues, e.g., Windows DPC

Accounting for Priority Inversion

- **Recall that task schedulability is affected by**
  - preemption: two types of preemption
    - can occur several times per task period
    - can occur once per period
  - execution: once per period
  - blocking: at most once per period for each source

- **The schedulability formulas are modified to add a "blocking" or "priority inversion" term to account for inversion effects**

UB Test with Blocking

- Include blocking time in calculation of effective utilization for each task:

$$f_i = \underbrace{\boxed{\sum_{j \in H_n} \frac{e_j}{p_j}}}_{H_n \text{ Preemption(can hit n times)}} + \underbrace{\boxed{\frac{e_i}{p_i}}}_{\text{Execution}} + \underbrace{\boxed{\frac{B_i}{p_i}}}_{\text{Blocking}} + \underbrace{\boxed{\frac{1}{p_i} \sum_{k \in H_1} e_k}}_{H_1 \text{ Preemption(can hit once)}}$$

Response Time Test with Blocking

- Blocking is also included in the RT test

$$a_{n+1} = B_i + e_i + \sum_{j=1}^{i-1} \left\lceil \frac{a_n}{p_j} \right\rceil e_j$$

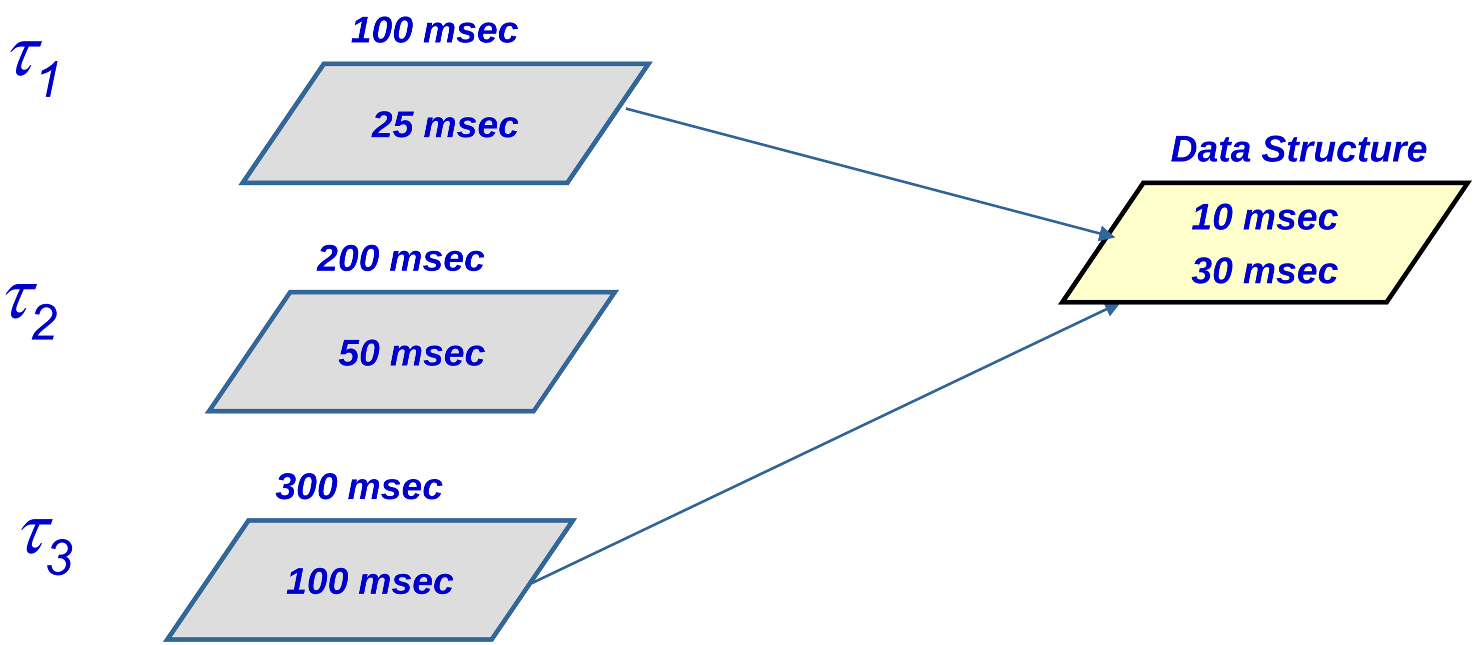
$$where \quad a_0 = B_i + \sum_{j=1}^{i} e_j$$

- Perform test as before, including blocking effect

Example: Considering Blocking

- Consider the following example

Periodic tasks



What is the worst case blocking effect (priority inversion) experienced by each task ?

Example: Adding Blocking

- Task τ_2 does not use the data structure. Task τ_2 experiences no priority inversion
- Task τ_1 shares the data structure with τ_3 . Task τ_1 may have to wait for τ_3 to complete its critical section. But worse, if τ_2 preempts while τ_1 is waiting for the data structure, τ_1 may have to wait for τ_2's entire computation.
- This is the resulting table

task	Period	Execution Time	Priority	Blocking delay	Deadline
τ_1	100	25	High	30+50	100
τ_2	200	50	Medium	0	200
τ_3	300	100	Low	0	300

UB Test for Example

- **UB test with blocking:**

$$f_i = \sum_{j \in H_n} \frac{e_j}{p_j} + \frac{e_i}{p_i} + \frac{B_i}{p_i} + \frac{1}{p_i} \sum_{k \in H_1} e_k$$

$$f_1 = \frac{e_1}{p_1} + \frac{B_1}{p_1} = \frac{25}{100} + \frac{80}{100} = 1.05 > 1.00$$ Not schedulable

$$f_2 = \frac{e_1}{p_1} + \frac{e_2}{p_2} = \frac{25}{100} + \frac{50}{200} = 0.5 < U(2)$$

$$f_3 = \frac{e_1}{p_1} + \frac{e_2}{p_2} + \frac{e_3}{p_3} = \frac{25}{100} + \frac{50}{200} + \frac{100}{300} = 0.84 > U(3)$$

with additional RT test, τ_3 is shown to be schedulable