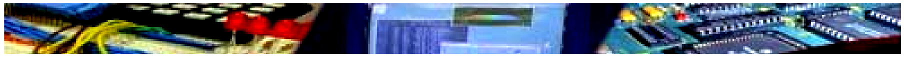
Scheduling Algorithm and Analysis

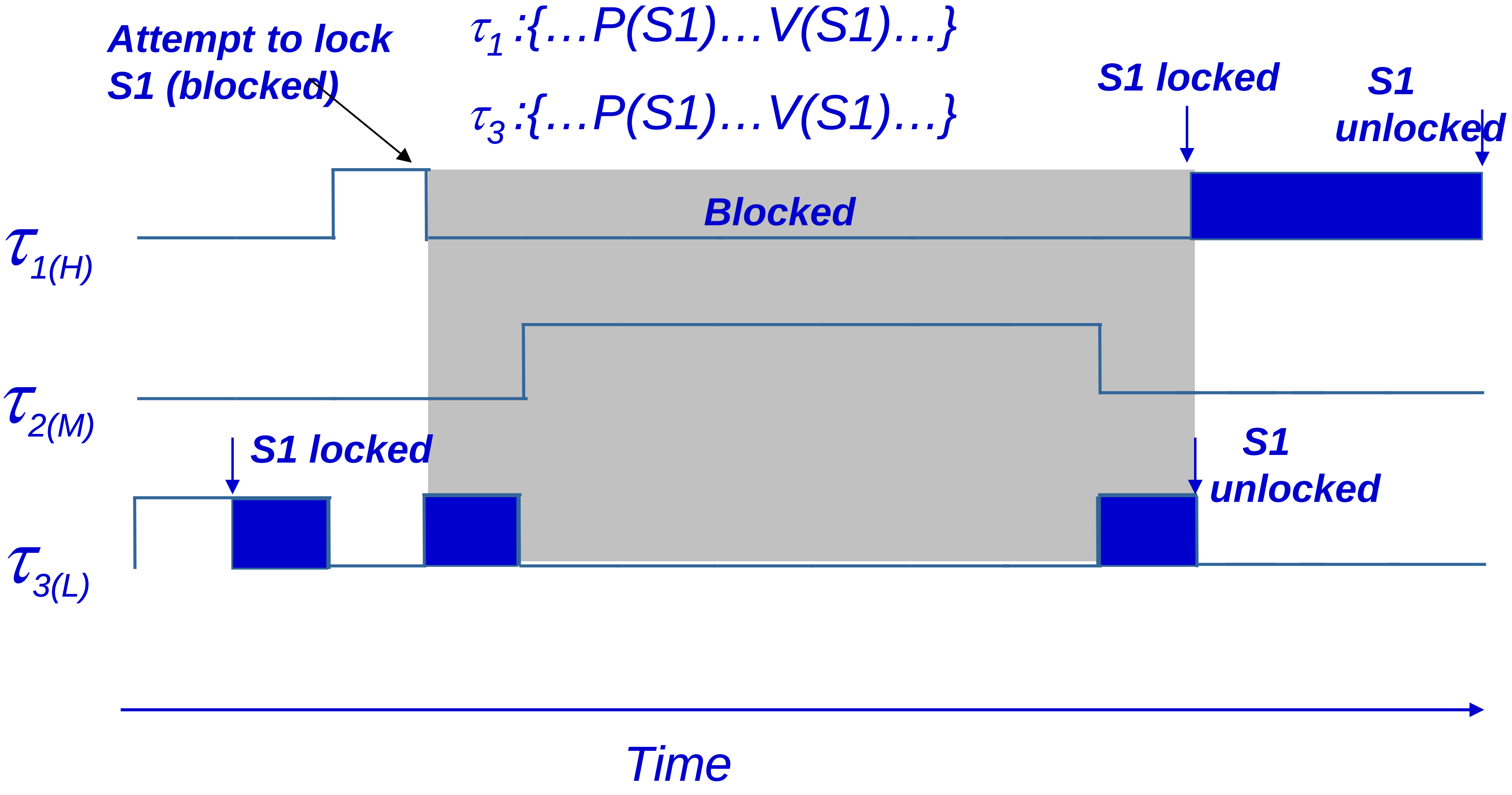
Priority Inversion
(Module 32)

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

Summer 2014



Priority Inversion in Synchronization



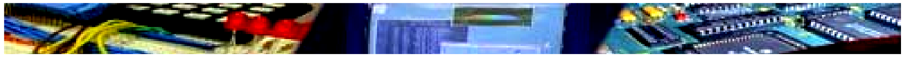
Priority Inversion

- **Delay to a task's execution caused by interference from 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

Accounting for Priority Inversion

- **Recall that task schedulability is affected by**
  - preemption: two types of preemption
    - can occur several times per period
    - can occur once per period
  - execution: once per period
  - blocking: at most once per period for each request to a 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 while calculating effective utilization for each tasks:**

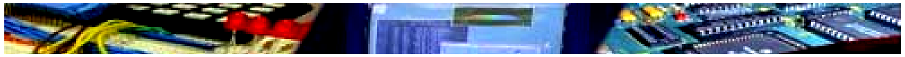
$$f_i = \sum_{j \in H_n} \frac{e_j}{p_j} + \frac{e_i}{p_i} + \frac{B_i}{p_i} + \frac{1}{p_i} \sum_{k \in H_1} e_k$$

H_n Preemption(can hit n times)

Execution

Blocking

H_1 Preemption(can hit once)



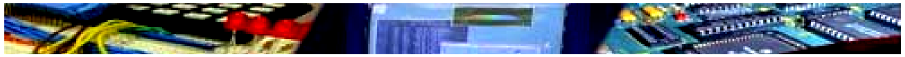
RT Test with Blocking

- **Blocking is also included in the RT test**

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

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

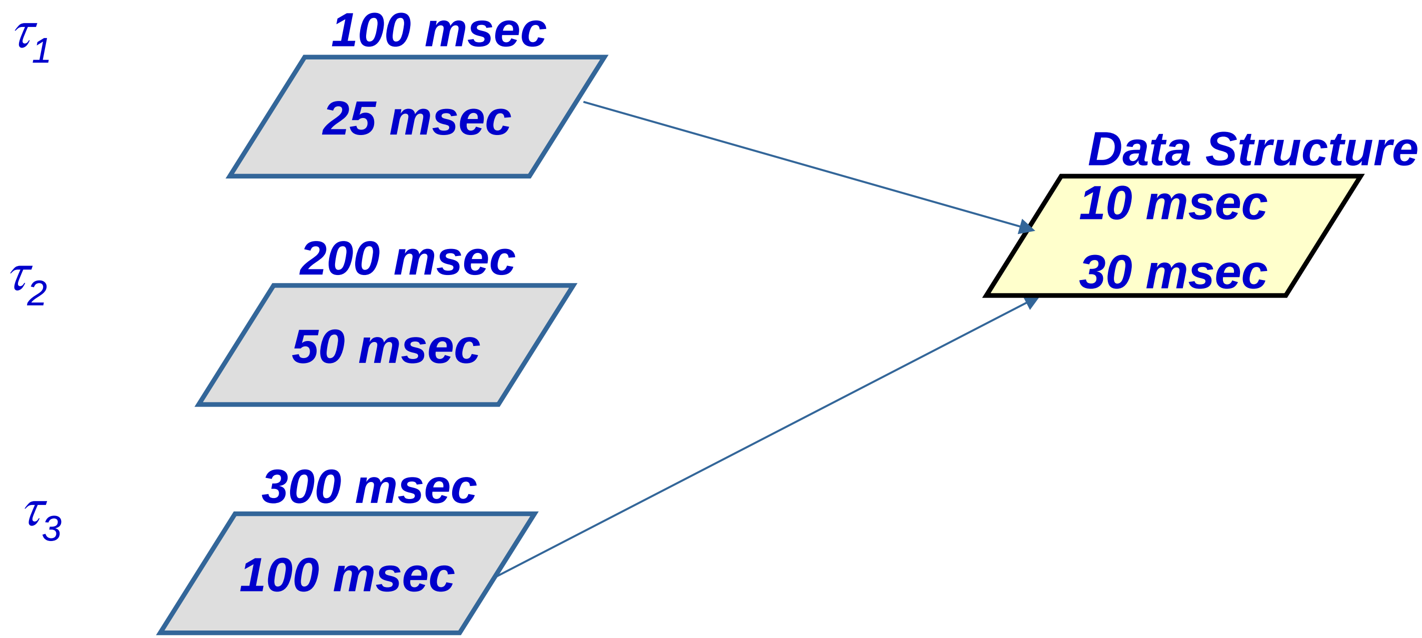
- **Perform test as before, including blocking effect**



Example: Considering Blocking

- Consider the following example

Periodic tasks

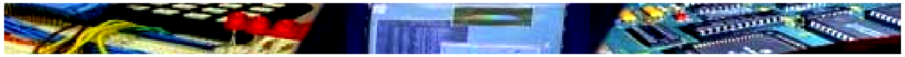


What is the worst case blocking effect (priority inversion) experienced by each task ?

Example: Adding Blocking

- Task τ_2 does not use the data structure. Task τ_2 does experiences no priority inversion
- Task τ_1 shares the data structure with τ_3 . Task τ_1 could 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 could 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:**

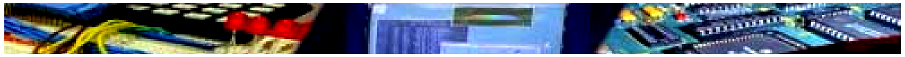
$$f_i = \sum_{j \in H_n} \frac{e_j}{p_j} + \frac{e_i}{p_i} + \frac{B_i}{p_i} + \frac{1}{p_i} \sum_{k \in H_1} e_k$$

$$f_1 = \frac{e_1}{p_1} + \frac{B_1}{p_1} = \frac{25}{100} + \frac{80}{100} = 1.05 > 1.00$$ 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



Supplementary Slides