Scheduling Algorithm and Analysis

RT Synchronization Protocol
(Module 34)

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

Summer 2014

Properties of Priority Inheritance

- For each resource (semaphore), a list of blocked tasks must be stored in a priority queue.
- A task (job) τ_i uses its assigned priority, and uses (inherits) the highest dynamic priority of all the tasks it blocks when it is in its critical section and blocks some higher priority tasks.
- Priority inheritance is *transitive*; that is, if task τ_i blocks τ_j and τ_j blocks τ_k , then τ_i can inherit the priority of τ_k.
- When task τ_i releases a resource, which priority it should use?
- Chained blocking if requesting multiple resources (nested mutex requests)
- Direct blocking and indirect (inheritance) blocking (when the lower priority task inherits the higher priority task's priority).

Implementation Issues of BIP

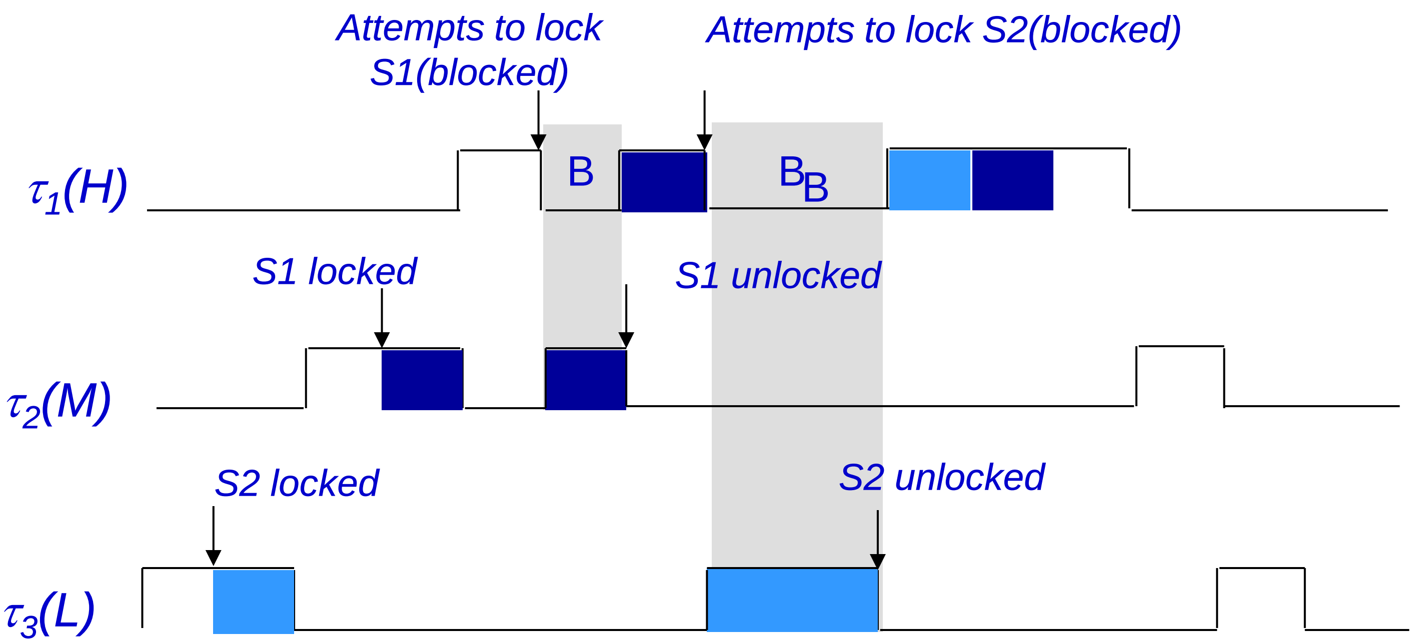
- **PI semaphore and basic semaphore**
- **Priority is changed when acquiring and releasing a lock**
- **When release a lock, can the priority be restore to the one before acquiring?**
- **Need to maintain a list of PI semaphores locked by a task**
  - when holding multiple locks and release one
  - when a PI semaphore is deleted
  - when a task waiting for a PI semaphore quits the waiting (due to timeout)
  - when the priority of a task waiting for a PI semaphore is changed
- **Note that for each semaphore, a queue for all waiting tasks**

Example Of Chained Blocking (BIP)

τ_1:{…P(S1)…P(S2)…V(S2)…V(S1)…}

τ_2:{…P(S1)…V(S1)…}

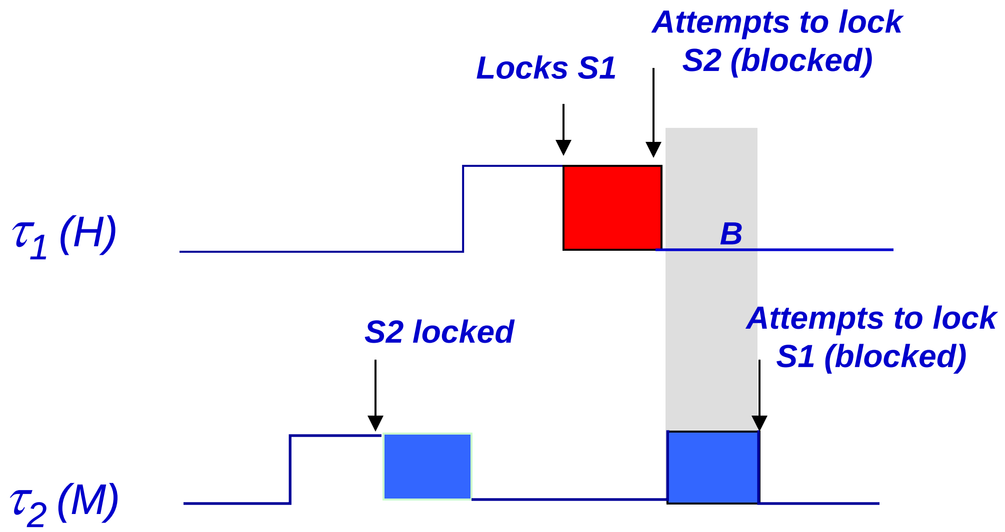
τ_3:{…P(S2)…V(S2)…}



Deadlock: Using BIP

τ_1 :{…P(S1)…P(S2)…V(S2)…V(S1)..}

τ_2 :{…P(S2)…P(S1)…V(S1)…V(S2)..}



Blocking Time Under BIP

❑ Example

T1 = {.. P(A) .3. P(B) .2. V(B) .1. V(A) ..}

T2 = {.. P(C) .2. V(C) ..}

T3 = {.. P(A) .1. P(B) .2. V(B) .2. V(A) .. }

T4 = {.. P(A) .1. P(C) .1. P(B) .3. V(B) .1. V(C) .1. V(A).. }

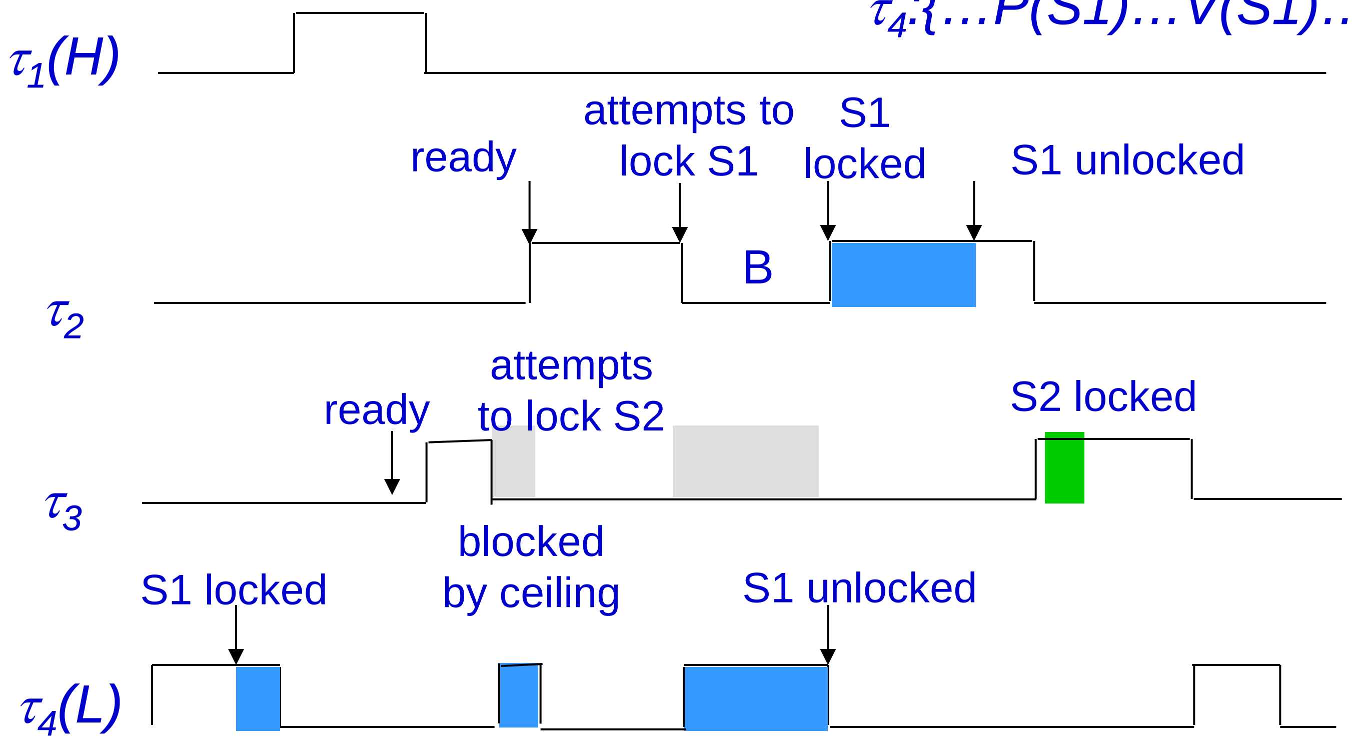
	direct blocking by			indirect blocking by			blocking time		
	T2	T3	T4	T2	T3	T4	T2	T3	T4
T1	N	Y	Y					5	7
T2		N	Y		Y	Y		5	7
T3			Y			Y			7

Priority Ceiling Protocol (PCP)

τ_2:{…P(S1)…V(S1)…}

τ_3:{…P(S2)…V(S2)…}

τ_4:{…P(S1)…V(S1)…}



Basic Priority Ceiling Rules (1)

- $\Pi(R)$ = priority ceiling of resource *R* – the highest priority of the tasks that request *R*
- $\Pi_S(t)$ = system priority ceiling -- the highest priority ceiling of the resources that are in use at time *t*
- *Scheduling Rule:* same as the assumptions
- ***Allocation Rule:***
  - if $J_i \rightarrow R_k \rightarrow J_l$ at $t = t_1$ then block J_i *(no change)*
  - R_k free at t_1,
    - if $\pi_i(t_1) > \Pi_S(t_1)$, then $R_k \rightarrow J_i$
    - else (i.e. $\pi_i(t_1) \leq \Pi_S(t_1)$)
      if for some $R_x \rightarrow J_i$ and $\Pi(R_x) = \Pi_S(t_1)$, then $R_k \rightarrow J_i$
      [J_i holds a resource R_x whose priority ceiling is $\Pi_S(t_1)$]
    - else deny and block ($J_i \rightarrow R_k$)

Basic Priority Ceiling Rules (2)

- *Priority-Inheritance Rule:*
  - if $J_i \rightarrow R_k$ at $t = t_1$ and is blocked by J_l (and $\pi_l(t_1^-)$ = priority of J_l)
    - either $R_k \rightarrow J_l$, (J_l holds the resource R_k)

      or $J_l \rightarrow R_x$ and $\Pi(R_x) = \Pi_S(t_1) \geq \pi_i(t_1)$
  - then $\pi_l(t_1^+) = \pi_i(t_1)$ (*inherited priority*)
  - until J_l releases all R_x with $\Pi(R_x) \geq \pi_i(t_1)$, $\pi_l(t_2^+) = \pi_l(t_1^-)$ at $t = t_2$.

Supplementary Slides