Real-Time Synchronization

Raj Rajkumar

Lecture #5

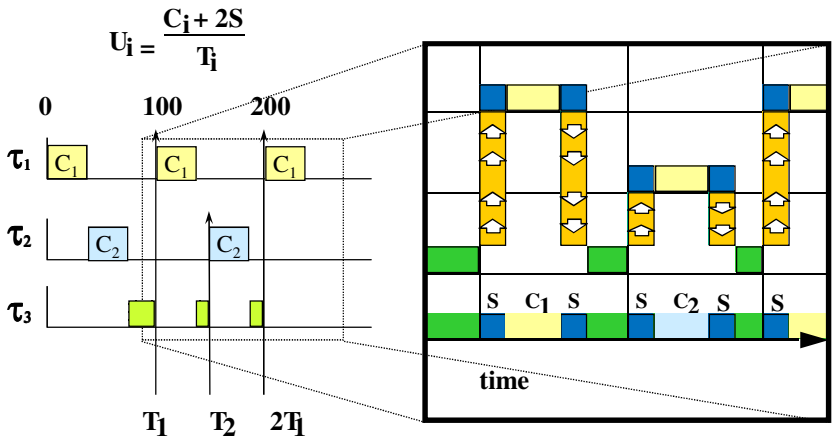
Administrivia

- Lab #1 handout on piazza

Outline

- Real-Time Synchronization Protocols
- Dealing with context switching
- Unbounded priority inversion
- Basic Priority Inheritance Protocol (BIP or PIP)
- Priority Ceiling Protocol (PCP)
- Highest Locker's Priority Protocol (HLP)
- Non-preemption Protocol (NPP)
- Example

Modeling Task Switching as Execution Time

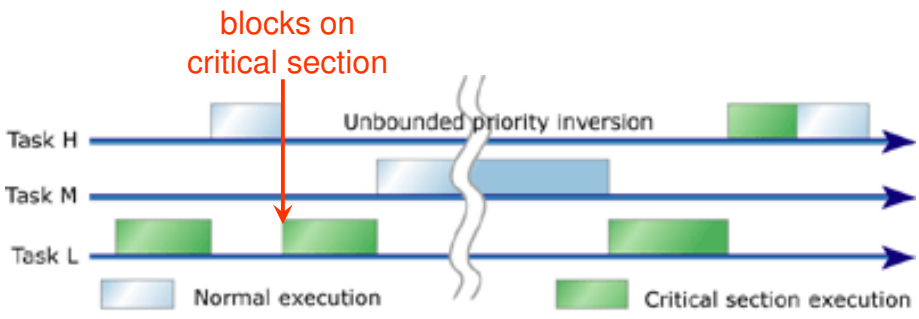


Two scheduling actions per task
(start of period and end of period)

Priority Inversion

- Ideally, under prioritized preemptive scheduling, higher priority tasks should *immediately* preempt lower priority tasks.
- When lower priority tasks cause higher priority tasks to wait (e.g. the locking of shared data), **priority inversion** is said to occur.
- It seems reasonable to expected that the duration of priority inversion (also called **blocking time**) should be a function of the duration of the critical sections.
- **Critical section**:
  - the duration of a task using a shared resource.

Unbounded Priority Inversion



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-preemptable 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 only 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.

Synchronization Protocols

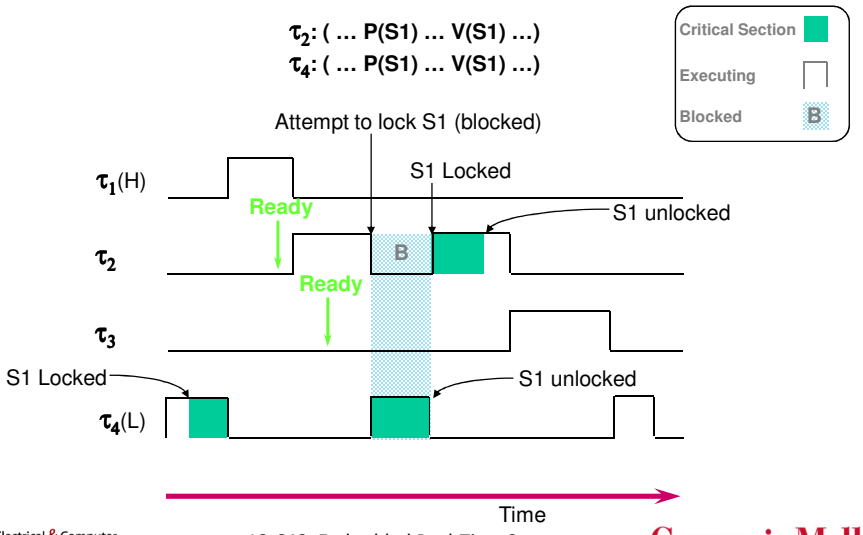
- No preemption
- Basic priority inheritance
- Highest locker's priority
- Priority ceiling

- Each protocol prevents unbounded priority inversion.

Basic Priority Inheritance Protocol

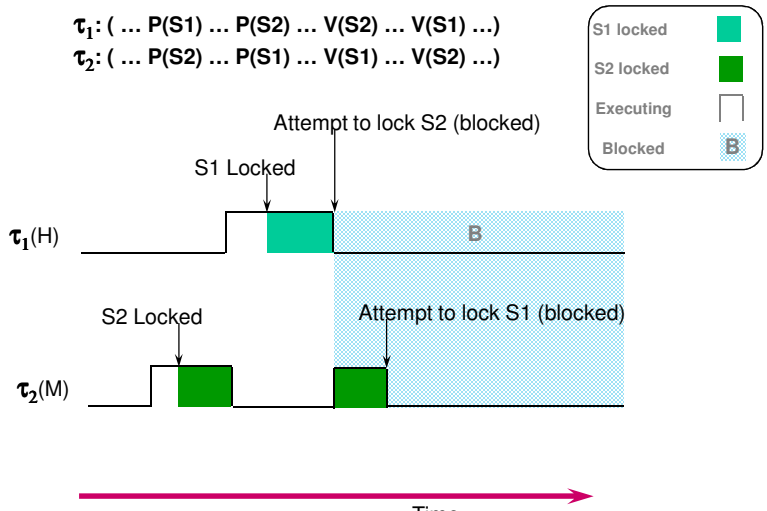
- Let the lower priority task τ_3 use the highest priority of the higher priority tasks it blocks. In this way, the medium priority tasks can no longer preempt low priority task τ_3, which has blocked the higher priority tasks.
- **Priority inheritance is transitive**.
  - If A blocks B and B blocks C, A should execute at the priority of *max*(B,C).

Basic Priority Inheritance Protocol (PIP)



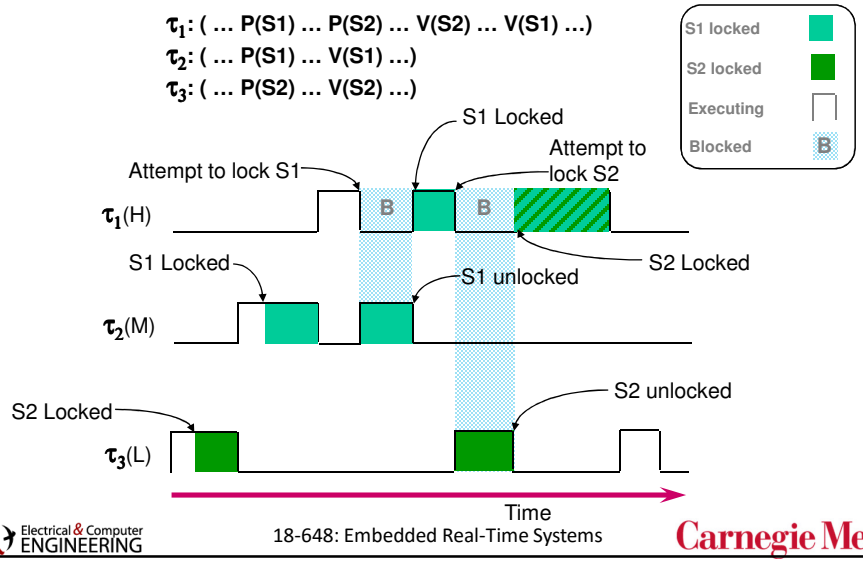
Deadlock: Using PIP

τ_1: (… P(S1) … P(S2) … V(S2) … V(S1) …)
τ_2: (… P(S2) … P(S1) … V(S1) … V(S2) …)



Example Of Chained Blocking (PIP)

τ_1: (… P(S1) … P(S2) … V(S2) … V(S1) …)
τ_2: (… P(S1) … V(S1) …)
τ_3: (… P(S2) … V(S2) …)



Properties of Basic Priority Inheritance

- There will be no deadlock if there is no nested locks, or application level deadlock avoidance scheme such the ordering of resource is used.
- Chained priority is fact of life. But a task is blocked at most by *n* lower priority tasks sharing resources with it, when there is no deadlock.
- The priority inheritance protocol is supported in POSIX real time extensions.
  - It is easy to implement
  - it is supported by not only most RT OS vendors but also Windows, AIX, HP/UX, and Solaris.

Blocking Term under PIP

- *Theorem:* A higher-priority task can be blocked for at most one lower-priority critical section by each mutex
  - *Proof Sketch*: Once a mutex has been released, it cannot. be locked by any other lower-priority tasks
- *Theorem:* A higher-priority task can be blocked for at most one (outermost) critical section of a lower-priority task
  - *Proof Sketch*: Once a lower-priority task has exited a critical section, it can no longer block a higher-priority task until it completes.

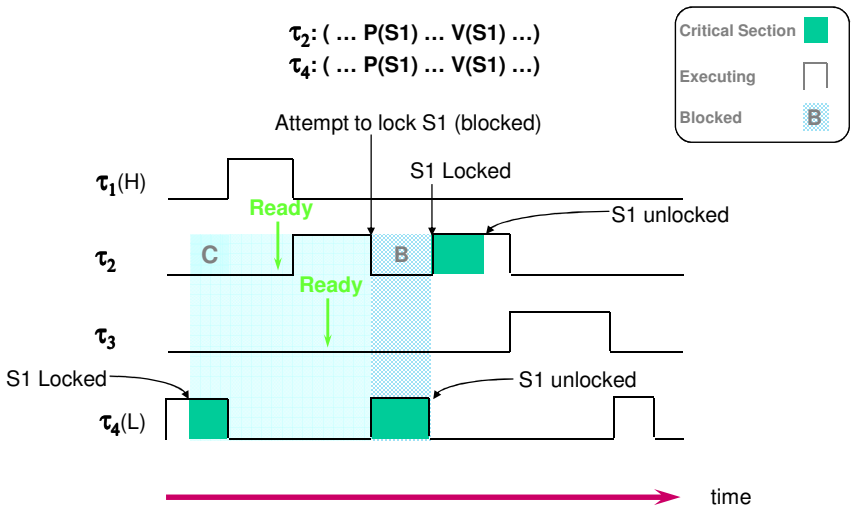
➔ If there are *n* lower-priority tasks and *m* distinct mutexes, a task can be blocked for no more than *min(m, n)* lower priority critical sections

→ Assuming that deadlocks are avoided using other mechanisms

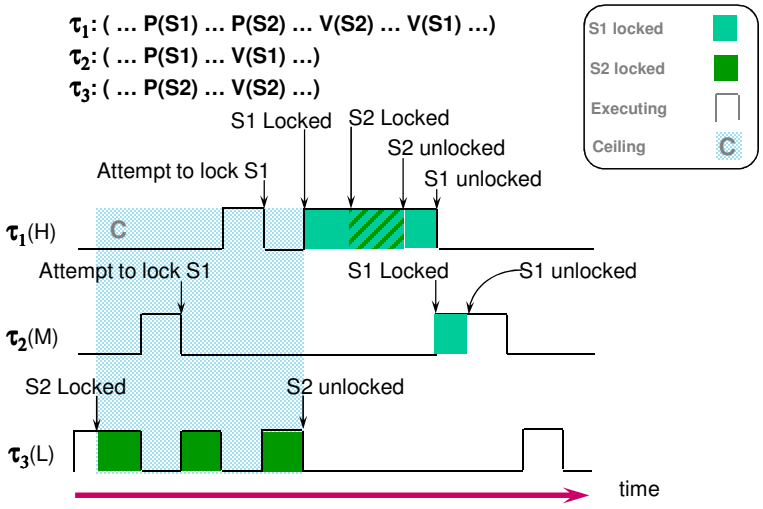
Priority Ceiling Protocol

- A **priority ceiling** is assigned to each mutex, which is equal to the highest priority task that may use this mutex.
- A task can lock a mutex if and only if its priority is higher than the priority ceilings of all mutexes locked by other tasks.
- If a task is blocked by a lower priority task, the lower priority task inherits its priority.

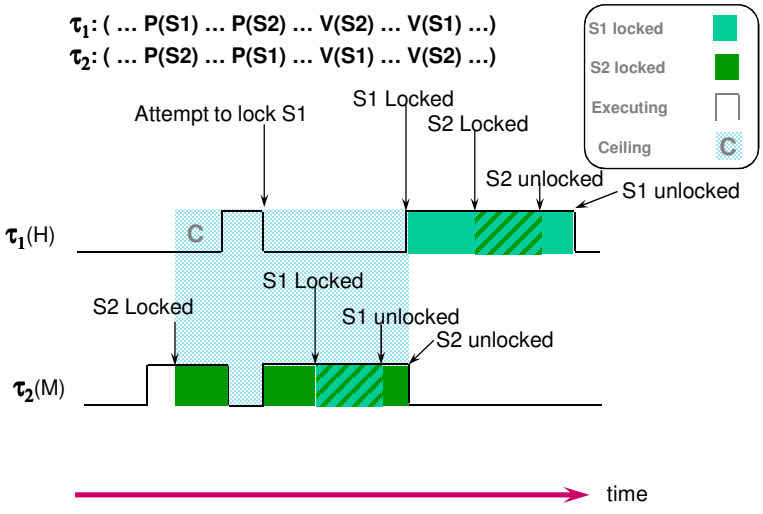
Priority Ceiling Protocol (PCP)



Blocked At Most Once (PCP)

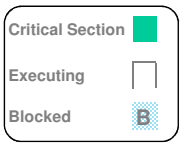


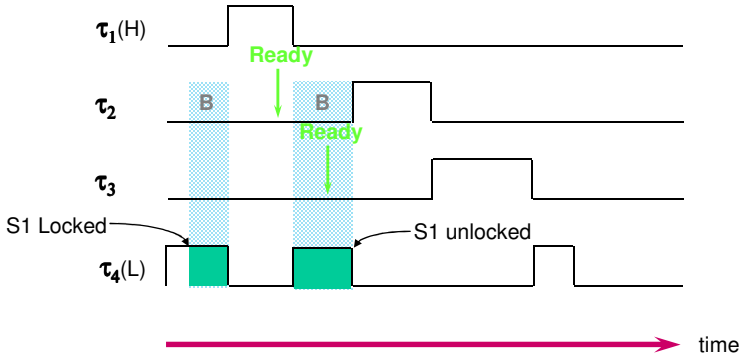
Deadlock Avoidance: Using PCP



Highest Locker's Priority Protocol

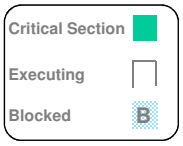
τ_2: (... P(S1) ... V(S1) ...)
τ_4: (... P(S1) ... V(S1) ...)

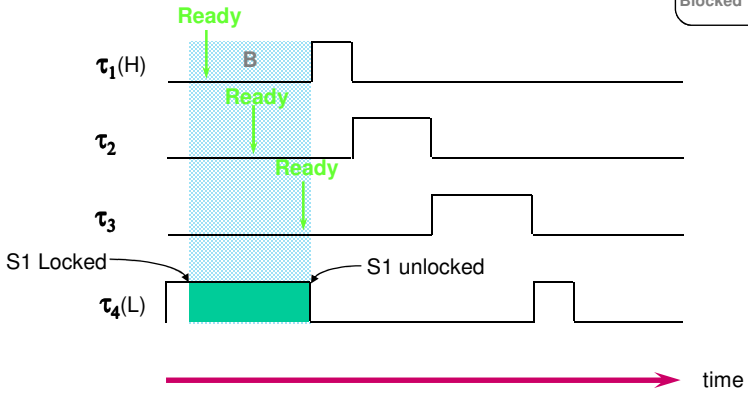




Non-Preemption Protocol

τ_2: (... P(S1) ... V(S1) ...)
τ_4: (... P(S1) ... V(S1) ...)



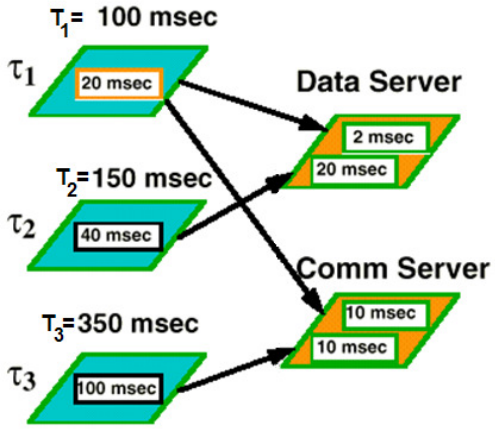


Summary of Synchronization Protocols

Protocol	Bounded Priority Inversion	Blocked for at most one critical section	Deadlock avoidance
Basic priority inheritance protocol	Yes*	No	No
Priority Ceiling protocol	Yes	Yes[2]	Yes
Highest locker's priority	Yes	Yes[1]	Yes[1]
Nonpreemptable critical sections	Yes	Yes[1]	Yes[1]

[1] Only if tasks do not suspend within critical sections
[2] PCP is not affected if tasks suspend within critical sections
* Deadlocks must be avoided using total (or) partial ordering of resources

Example



(Conservative) Analysis for PIP

	C	T	B
τ_1	20	100	
τ_2	40	150	
τ_3	100	350	

preemption execution blocking

$$\frac{C_1}{T_1}+\ldots+\frac{C_{i-1}}{T_{i-1}}+\frac{C_i}{T_i}+\frac{B_i}{T_i}\le U \quad (i)$$

Plugging in the numbers

$$\frac{C_1}{T_1}+\frac{B_1}{T_1}\le U \quad (1) \qquad \frac{20}{100}+\frac{30}{100}=0.50<1.0$$

$$\frac{C_1}{T_1}+\frac{C_2}{T_2}+\frac{B_2}{T_2}\le U \quad (2) \qquad \frac{20}{100}+\frac{40}{150}+\frac{10}{150}=0.533<0.828$$

$$\frac{C_1}{T_1}+\frac{C_2}{T_2}+\frac{C_3}{T_3}\le U \quad (3) \qquad \frac{20}{100}+\frac{40}{150}+\frac{100}{350}=0.753<0.779$$

(Conservative) Analysis for PCP

	C	T	B
τ_1	20	100	
τ_2	40	150	
τ_3	100	350	

preemption execution blocking

$$\frac{C_1}{T_1}+\ldots+\frac{C_{i-1}}{T_{i-1}}+\frac{C_i}{T_i}+\frac{B_i}{T_i}\leq U\ (i)$$

Summary

- Context switching can be handled by schedulability analysis techniques
- Real-Time Synchronization Protocols are required to prevent potentially unbounded priority inversion
- Protocols available:
  - **Priority Inheritance Protocol**
  - Priority Ceiling Protocol
  - **Highest Locker's Priority**
  - **Non-preemption Protocol**
- Developed at Carnegie Mellon and widely supported by standards and products