Priority Inheritance Protocols: An Approach to Real-Time Synchronization

김세화

Real-Time Operating Systems Laboratory

SoEE&CS, SNU

Outline

- Introduction
  - Synchronization problem
  - Priority inversion problem
- Assumptions and notations
- Basic priority inheritance protocol
- Priority ceiling protocol
- Schedulability analysis
- Conclusion

Synchronization Problem

- Resource sharing
  - Requires mutual exclusion
  - Critical section
    - A code section that should be executed mutually exclusively by tasks
  - Semaphore
    - One of synchronization primitives

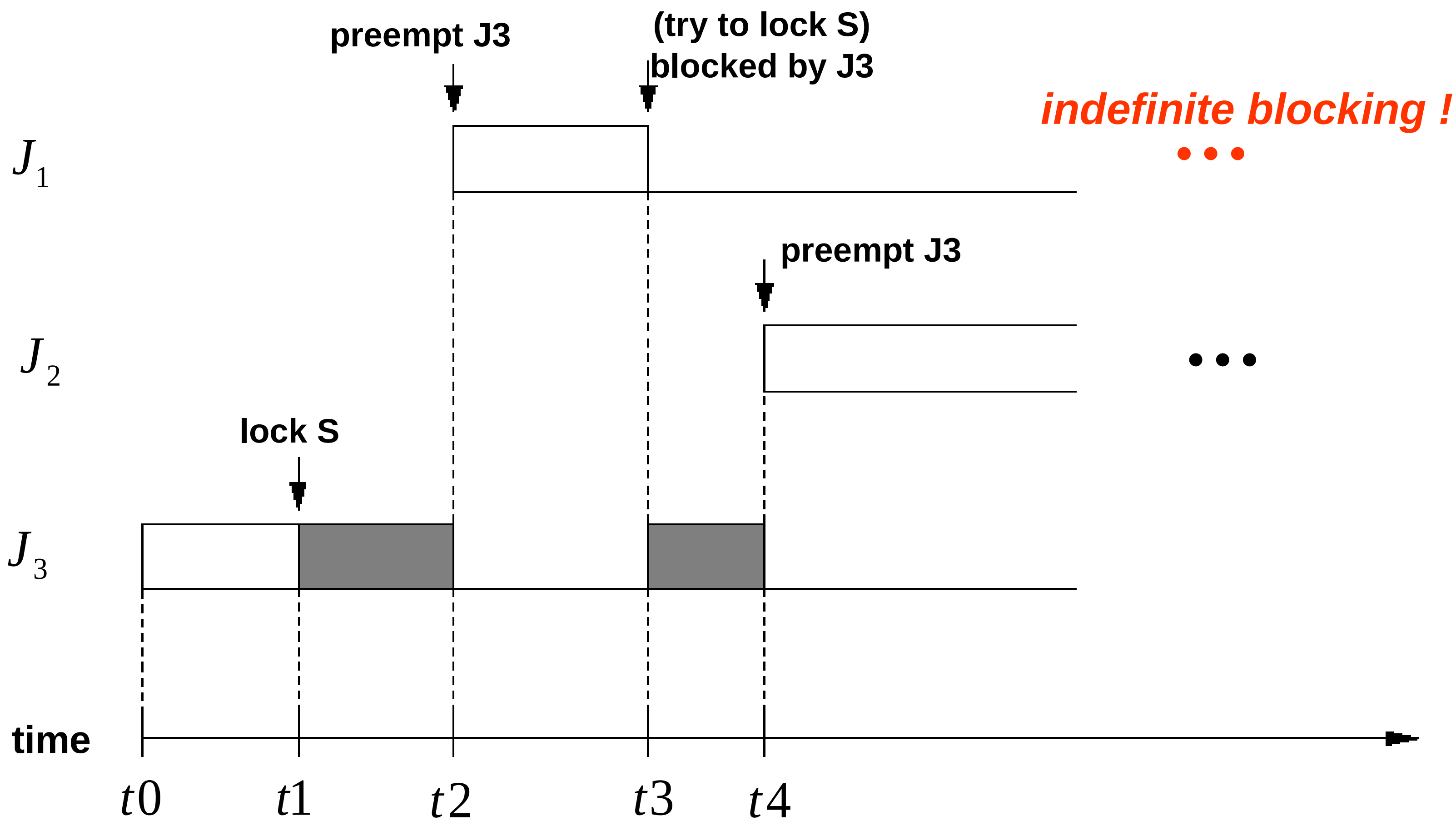
- Assumptions of this paper
  - Fixed-priority preemptive scheduling
  - A uniprocessor environment
  - Binary semaphore (Mutex)

Priority Inversion Problem

- Priority inversion
  - Phenomenon where a higher priority job is *block*ed by lower priority jobs

- Indefinite priority inversion
  - Occurs when a task of medium priority preempts a task of lower priority which is blocking a task of higher priority.

Indefinite Priority Inversion



Outline

- Introduction
  - Synchronization problem
  - Priority inversion problem
- **<u>Assumptions and notations</u>**
- Basic priority inheritance protocol
- Priority ceiling protocol
- Schedulability analysis
- Conclusion

Notations (1/2)

- J_i: A job, i.e., an instance of a task τ_i
  - T_i: Period of τ_i
  - P_i: Priority of τ_i
    - Jobs J_1, J_2, …, J_n are listed in descending order of priority.

- S_i: A Binary semaphore
  - $P(S_i)$: Indivisible lock (wait) operation
  - $V(S_i)$: indivisible unlock (signal) operation

- $z_{i,j}$: The j-th critical section in job J_i
  - $S_{i,j}$: The semaphore locked and released by critical section $z_{i,j}$
  - $z_{i,j} \subset z_{i,k}$: The critical section $z_{i,j}$ is entirely contained in $z_{i,k}$.

Notations (2/2)

- *Definition*: A job J is said to be blocked by the critical section $z_{i,j}$ of job J_i:
  - If J_i has a lower priority than J but J has to wait for J_i to exit $z_{i,j}$ in order to cont inue execution.
- *Definition*: A job J is said to be blocked by job J_i through semaphore S:
  - If the critical section $z_{i,j}$ blocks J and $S_{i,j} = S$.

- Blocking set
  - $\beta_{i,j}$: The set of all critical sections of J_j which can block J_i
    - ∀ $\beta_{i,j} = \{ z_{j,k} \mid j > i$ and $z_{j,k}$ can block $J_j \}$
  - $\beta^*_{i,j}$: The set of all longest (outermost) critical sections of J_j which can block J_i.
    - ∀ $\beta^*_{i,j} = \{ z_{j,k} \mid (z_{j,k} \in \beta_{i,j}) \land (\sim \exists\, z_{j,m} \in \beta_{i,j}$ such that $z_{j,k} \subset z_{j,m} \}$
  - β^*_i: The set of all longest (outermost) critical sections that can block J_i
    - ∀ $\beta^*_i = U_{j>i}\, \beta^*_{i,j}$

Assumptions

- No voluntary blocking
  - Jobs do not suspend themselves, say for I/O operations.
- Properly nested critical sections
  - (ex)

$$J_i = \{..., P(S_1), ..., P(S_2), ..., V(S_2), ..., V(S_1), ...\} \rightarrow$$ ***Properly nested semaphores***

$$J_i = \{..., P(S_1), ..., P(S_2), ..., V(S_1), ..., V(S_2), ...\} \rightarrow$$ ***Non-properly nested semaphores***

Outline

- Introduction
- Assumptions and notations
- **Basic priority inheritance protocol (BPI)**
  - **Description**
  - Examples
  - Properties
  - Problems
  - Summary
- Priority ceiling protocol
- Schedulability analysis
- Conclusion

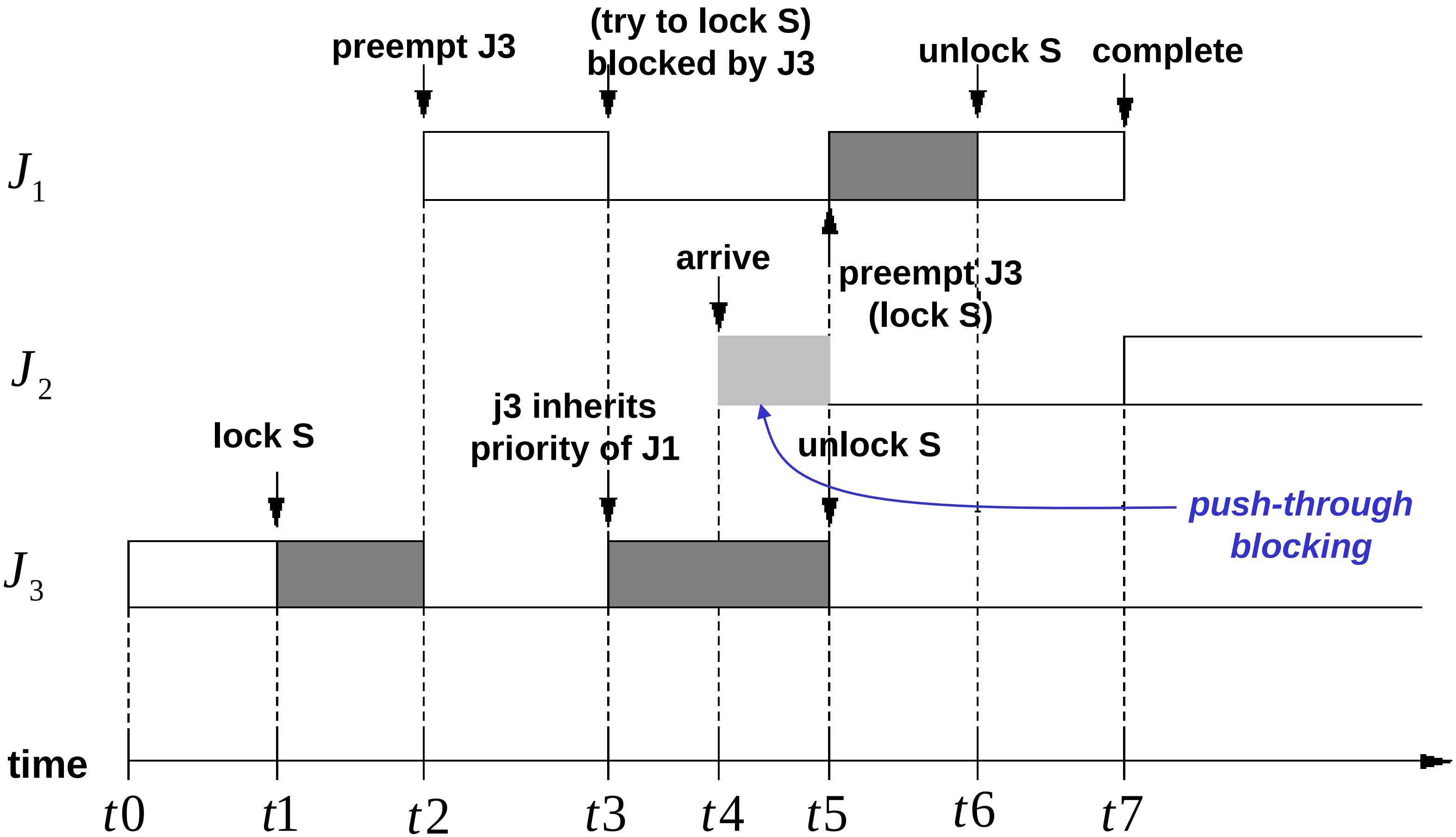
Description of Basic Protocol

- If job J blocks higher priority jobs:
  - J inherits P_H , the highest priority of the jobs blocked by J .
- Priority inheritance is transitive.
  - If J_3 blocks J_2 and J_2 blocks J_1, J_3 would inherit the priority of J_1 via J_2.
- When J exits a critical section:
  - J resumes the priority it had at the point of entry into the critical section.

Outline

- Introduction
- Assumptions and notations
- **Basic priority inheritance protocol (BPI)**
  - Description
  - **Examples**
  - Properties
  - Problems
  - Summary
- Priority ceiling protocol
- Schedulability analysis
- Conclusion

Examples for Basic Protocol (1/2)



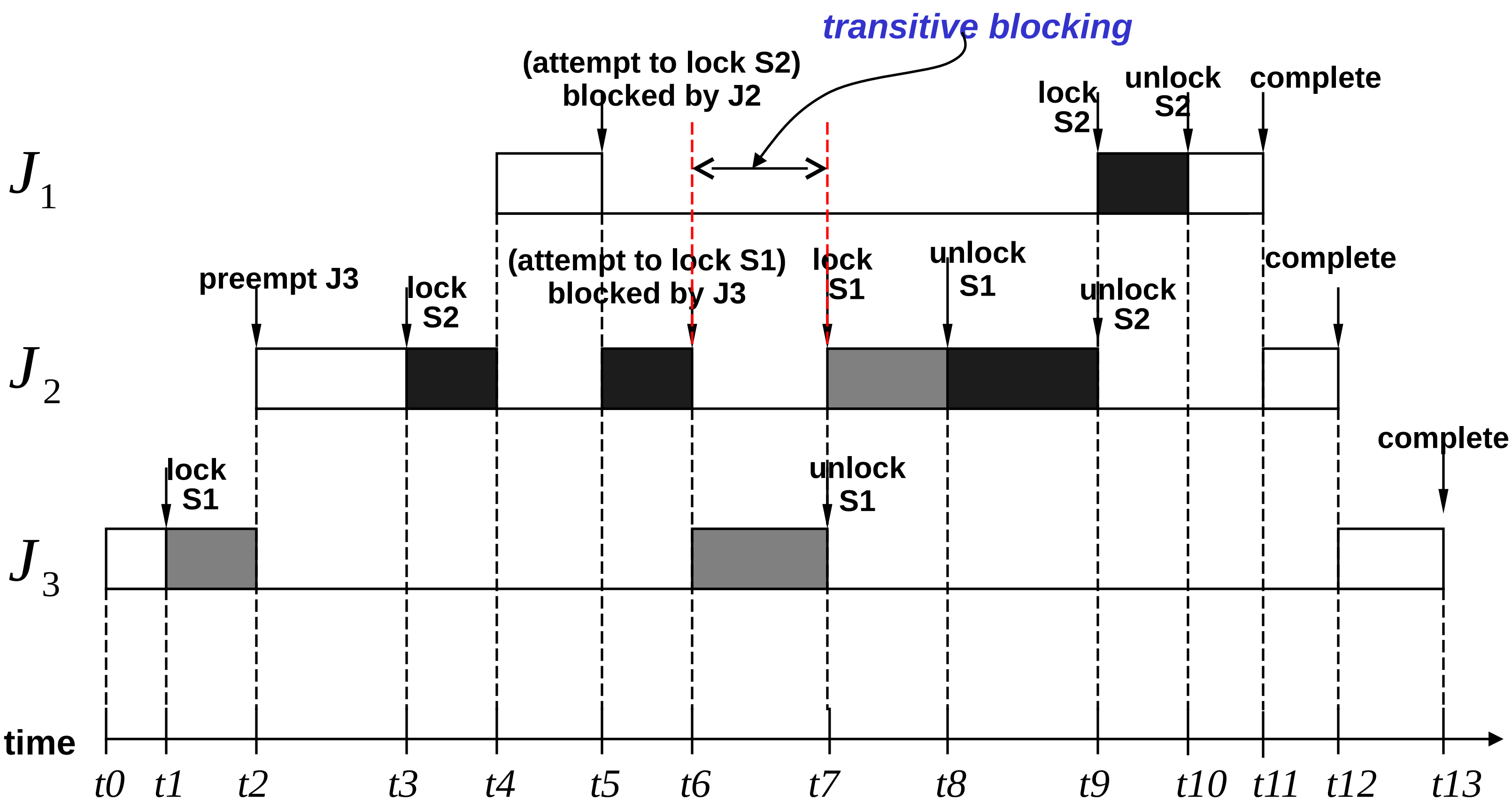
Examples for Basic Protocol (2/2)

$J_1 = \{..., P(S_2), ..., V(S_2), ...\}$

$J_2 = \{..., P(S_2), ..., P(S_1), ..., V(S_1), ..., V(S_2), ...\}$

$J_3 = \{..., P(S_1)\ \ , ..., V(S_1), ...\}$

nested semaphores



Outline

- Introduction
- Assumptions and notations
- **Basic priority inheritance protocol (BPI)**
  - Description
  - Examples
  - **Properties**
  - Problems
  - Summary
- Priority ceiling protocol
- Schedulability analysis
- Conclusion

Blocking in Basic Protocol

- Three Types of blocking
  - Direct blocking
    - Ensures the consistency of shared data.
  - Push-through blocking
    - Prevents indefinite blocking due to priority inversion
  - Transitive blocking
    - By not-directly involved semaphores which are accessed in a nested form by blocking jobs.
    - *Transitive blocking is said to occur if a job J is blocked by J_i whi ch, in turn, is blocked by another job J_j .*

Properties of Basic Protocol (1/3)

- A high priority job J_H can be blocked by a lower priority job J_L:
  - [Lemma1: Blocking condition]: Only if J_L is executing within a critical section, $z_{j,k} \in \beta^*_{H.L}$ when J_H is initiated.
  - [Lemma2: Blocking duration from one lower priority job]: For at mos t the duration of one critical section $\beta^*_{H.L}$, regardless of the number o f semaphores J_H and J_L share.

- [Theorem 3: Blocking duration]: Given a job J_0 for which there ar e *n* lower priority jobs { J_1,..., J_n }:
  - J_0 can be blocked for at most the duration of one critical section in ***e ach*** of $\beta^*_{0,i}$, $1 \le i \le n$.
    - NOTE: Each of *n* lower priority jobs can block job J_0 for at most the dura tion of a single critical section in each of the blocking in sets $\beta^*_{0,i}$.

Properties of Basic Protocol (2/3)

- [Lemma 4: Push-through blocking condition]: Semaphore S can cause push-through blocking to job J:
  - Only if S is accessed both (1) by a job which has priority lower than that of J and (2) by a job which has or can inherit priority equal to or higher than that of J .
    - J is a medium-priority job which may have no relation with S

- Notation
  - $\zeta^*_{i,j,k}$: The set of all longest critical sections of job J_j guarded by sema phore S_k and block job J_i either directly or via push-trough blocking
    $\zeta^*_{i,j,k} = \{ z_{j,p} \mid z_{j,p} \in \beta^*_{i,j}$ and $s_{j,p} = S_k \}$
    $\zeta^*_{i,\bullet,k} = U_{j \geq i}\, \zeta^*_{i,j,k}$

Properties of Basic Protocol (3/3)

- [Lemma 5: Blocking frequency for each semaphore]: A job J_i can encounter blocking:
  - By at most one critical section in $\zeta^*_{i,\bullet,k}$ for each semaphore S_k, $1 \le k \le m$, where m is the number of distinct semaphores.

- [Theorem 6: Blocking frequency]: If there are m semaphores which can block job J:
  - J can be blocked at most m times.

- Upper bound on *the total blocking delay* that a job can encounter
  - Can be determined by studying:
    - The durations of the critical sections in $\beta^*_{i,j}$ and $\zeta^*_{i,\bullet,k}$
  - From [Theorem 3: Blocking duration] and [Theorem 6: Blocking frequency]

Outline

- Introduction
- Assumptions and notations
- **Basic priority inheritance protocol (BPI)**
  - Description
  - Examples
  - Properties
  - **Problems**
  - Summary
- Priority ceiling protocol
- Schedulability analysis
- Conclusion

Problems of Basic Protocol

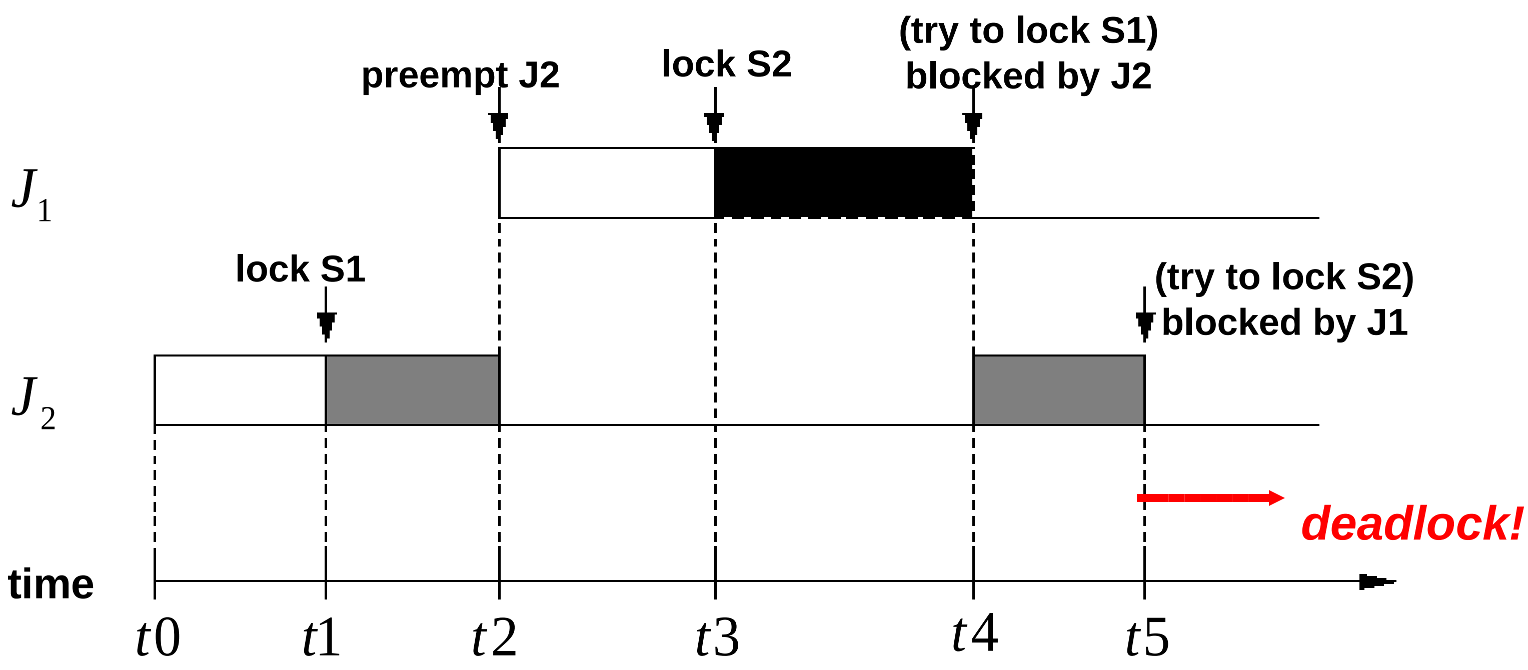
- Deadlocks
  - Due to crossing nested semaphores

- Long blocking delay
  - Due to
    - Transitive blocking
      - Blocking by not-directly involved semaphores which are accessed in nested form by blocking jobs.
    - Blocking chains
      - Blocking can occur sequentially whenever accessing each semaphore.

Deadlocks in Basic Protocol

$$J_1 = \{\ldots, P(S_2) \ldots, P(S_1),\ldots,V(S_1) \ldots,V(S_2),\ldots\}$$
$$J_2 = \{\ldots, P(S_1) \ldots, P(S_2),\ldots,V(S_2) \ldots,V(S_1),\ldots\}$$

crossing nested semaphores

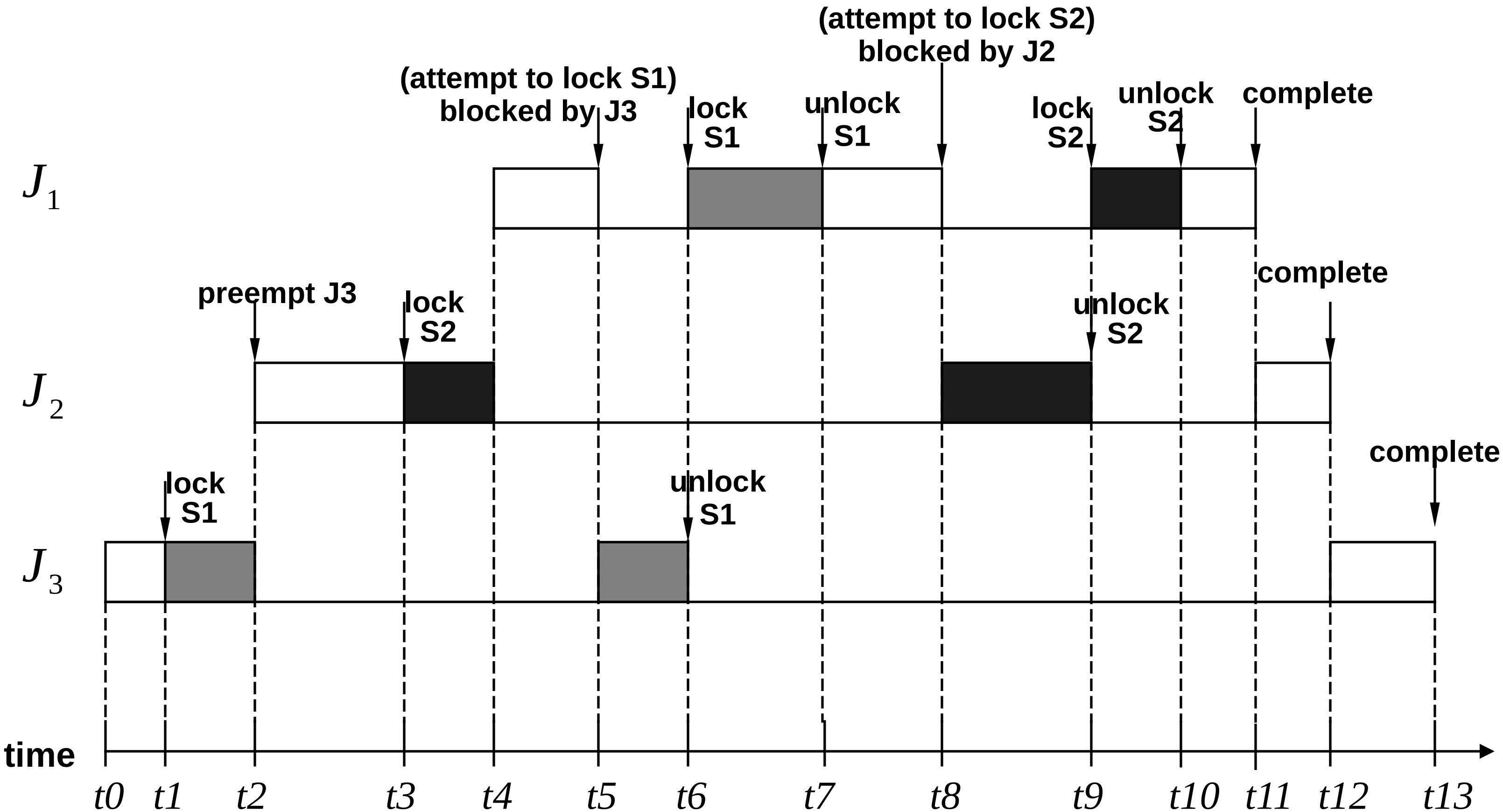


Blocking Chains in Basic Protocol

$$J_1 = \{\ldots, \overbrace{P(S_1), \ldots, V(S_1)}, \ldots, \overbrace{P(S_2), \ldots, V(S_2)}, \ldots\}$$

$$J_2 = \{\ldots, P(S_2)\ \ , \ldots, V(S_2), \ldots\}$$

$$J_3 = \{\ldots, P(S_1)\ \ , \ldots, V(S_1), \ldots\}$$



Summary of Basic Protocol

- Basic Idea
  - If a job blocks higher priority jobs, it inherits the highest priority of the jobs blocked by it.
- Three types of blocking
  - (1) Direct blocking, (2) push-through blocking, and (3) transitive blocking
- Controlled priority inversion
  - Upper bound on *the total blocking delay* that a job can encounter
    - Can be determined by studying the durations of the critical sections in $\beta^*_{i,j}$ and $\zeta^*_{i,\bullet,k}$.
- Problems
  - Deadlocks
  - Long blocking delay
    - Due to transitive blocking and blocking chains

Outline

- Introduction
- Assumptions and notations
- Basic priority inheritance protocol (BPI)
- **Priority ceiling protocol (PCP)**
  - **Overview**
  - Examples
  - Properties
  - Summary
- Schedulability analysis
- Conclusion

Overview of PCP

- Goals:
  - Solve problems of BPI.
    - Prevent deadlocks, transitive blocking and blocking chains
- Basic idea:
  - Priority ceiling of a semaphore:
    - The priority of the highest priority task that may use the semaphore
  - Additional condition for allowing a job *J* to start a new critical section
    - only if *J*'s priority is higher than all priority ceilings of all the semaphores locked by jobs other than *J* .

Outline

- Introduction
- Assumptions and notations
- Basic priority inheritance protocol (BPI)
- **Priority ceiling protocol (PCP)**
  - Overview
  - **Examples**
  - Properties
  - Summary
- Schedulability analysis
- Conclusion

Examples for PCP (1/3)

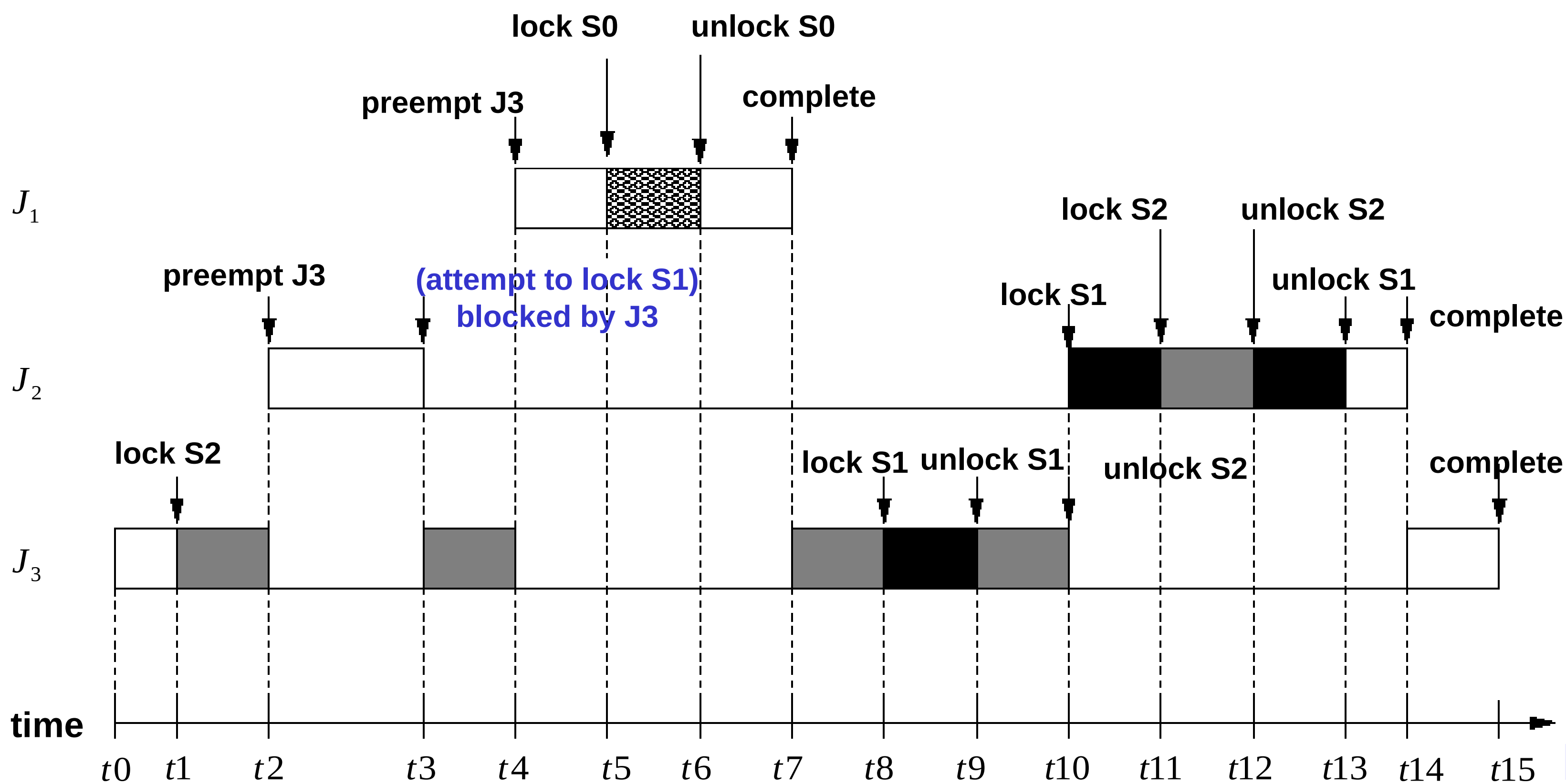
- Prevent deadlocks

$J_1 = \{..., P(S_0) \;..., V(S_0),...\}$

$J_2 = \{..., P(S_1) \;..., P(S_2),...,V(S_2) \;...,V(S_1),...\}$

$J_3 = \{..., P(S_2) \;..., P(S_1),...,V(S_1) \;...,V(S_2),...\}$

nested crossing semaphores



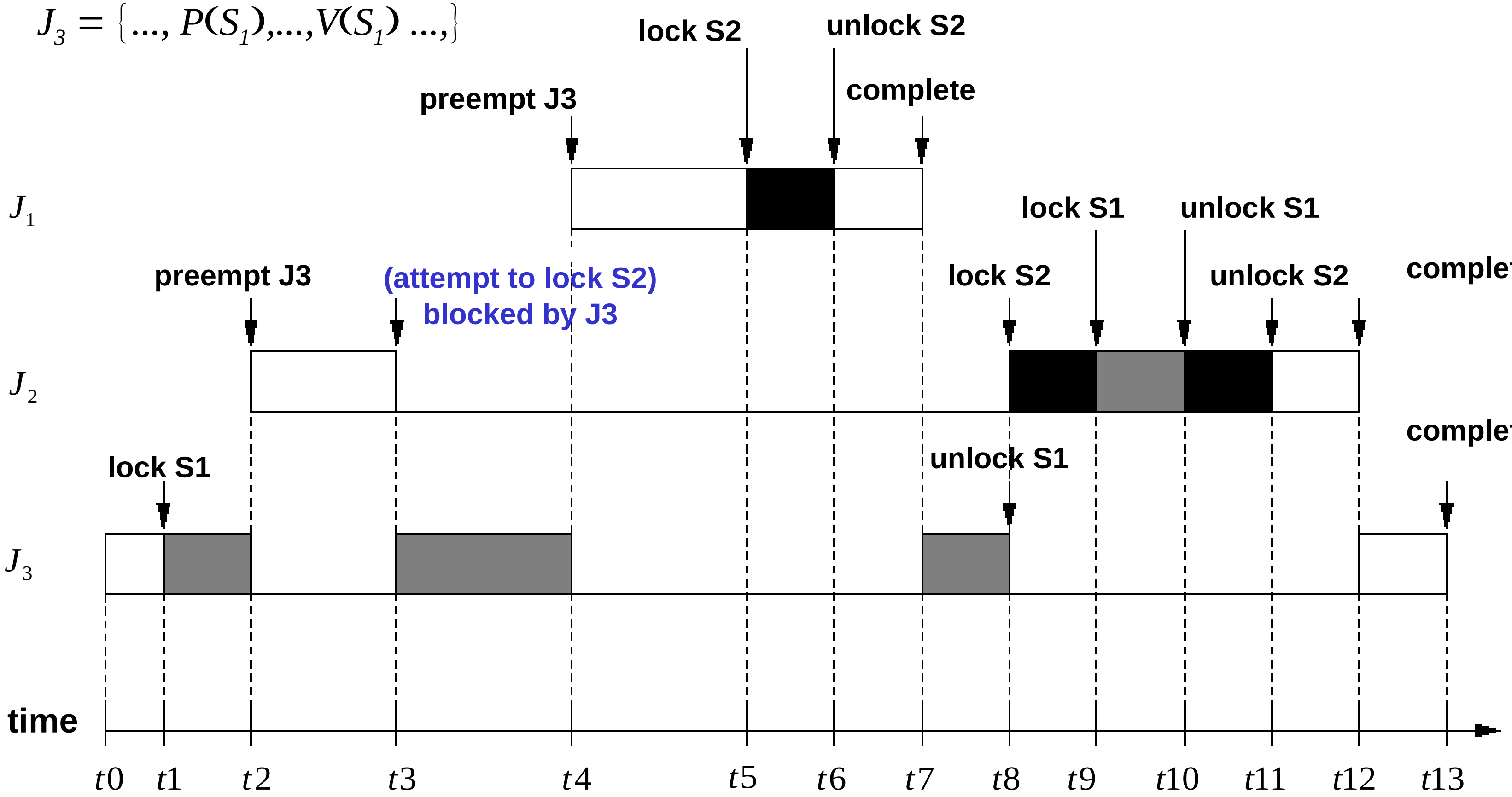
Examples for PCP (2/3)

- Prevent transitive blocking

$$J_1 = \{\ldots, P(S_2) \ldots, V(S_2), \ldots\}$$

$$J_2 = \{\ldots, P(S_2) \ldots, P(S_1), \ldots, V(S_1) \ldots, V(S_2), \ldots\} \longrightarrow \textit{nested semaphores}$$

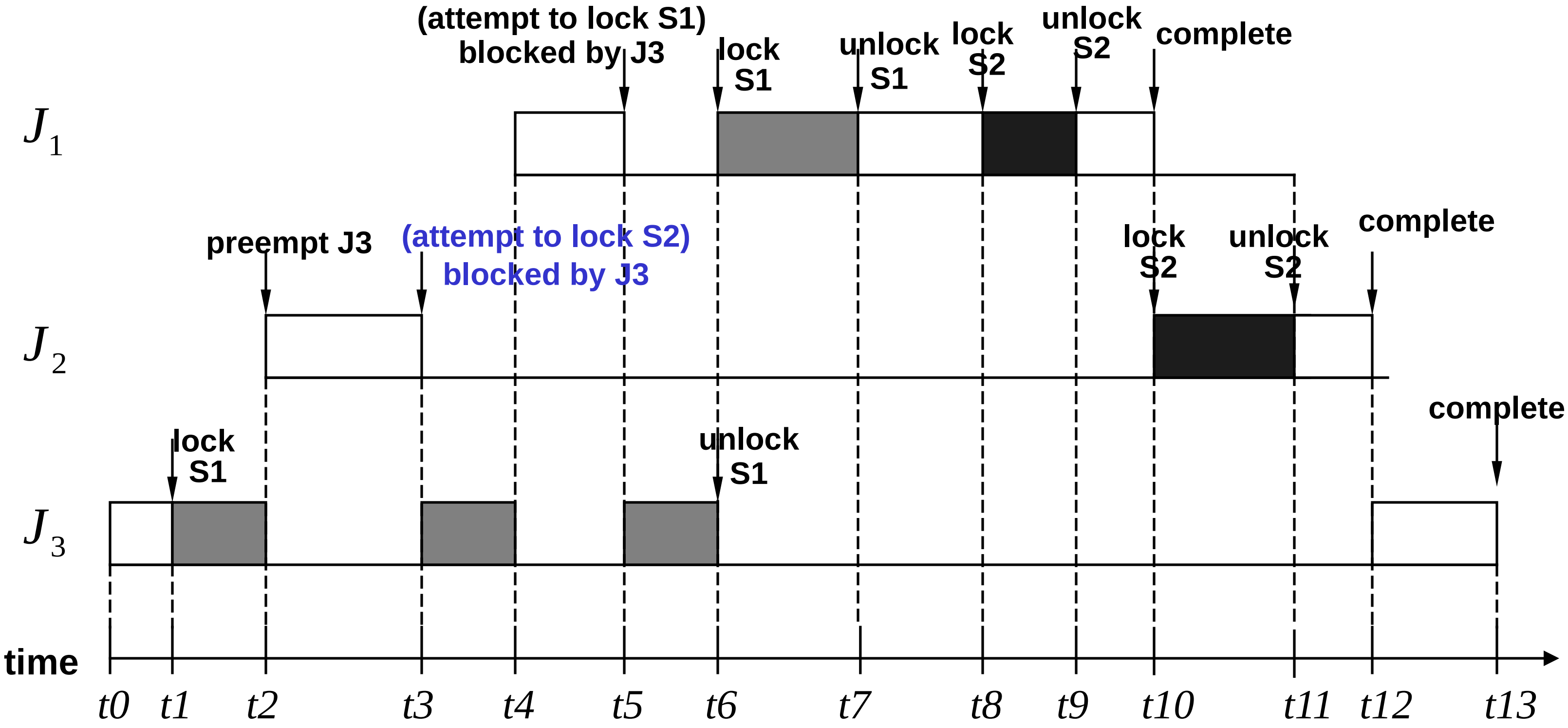
$$J_3 = \{\ldots, P(S_1), \ldots, V(S_1) \ldots,\}$$



Examples for PCP (3/3)

- Prevent blocking chains

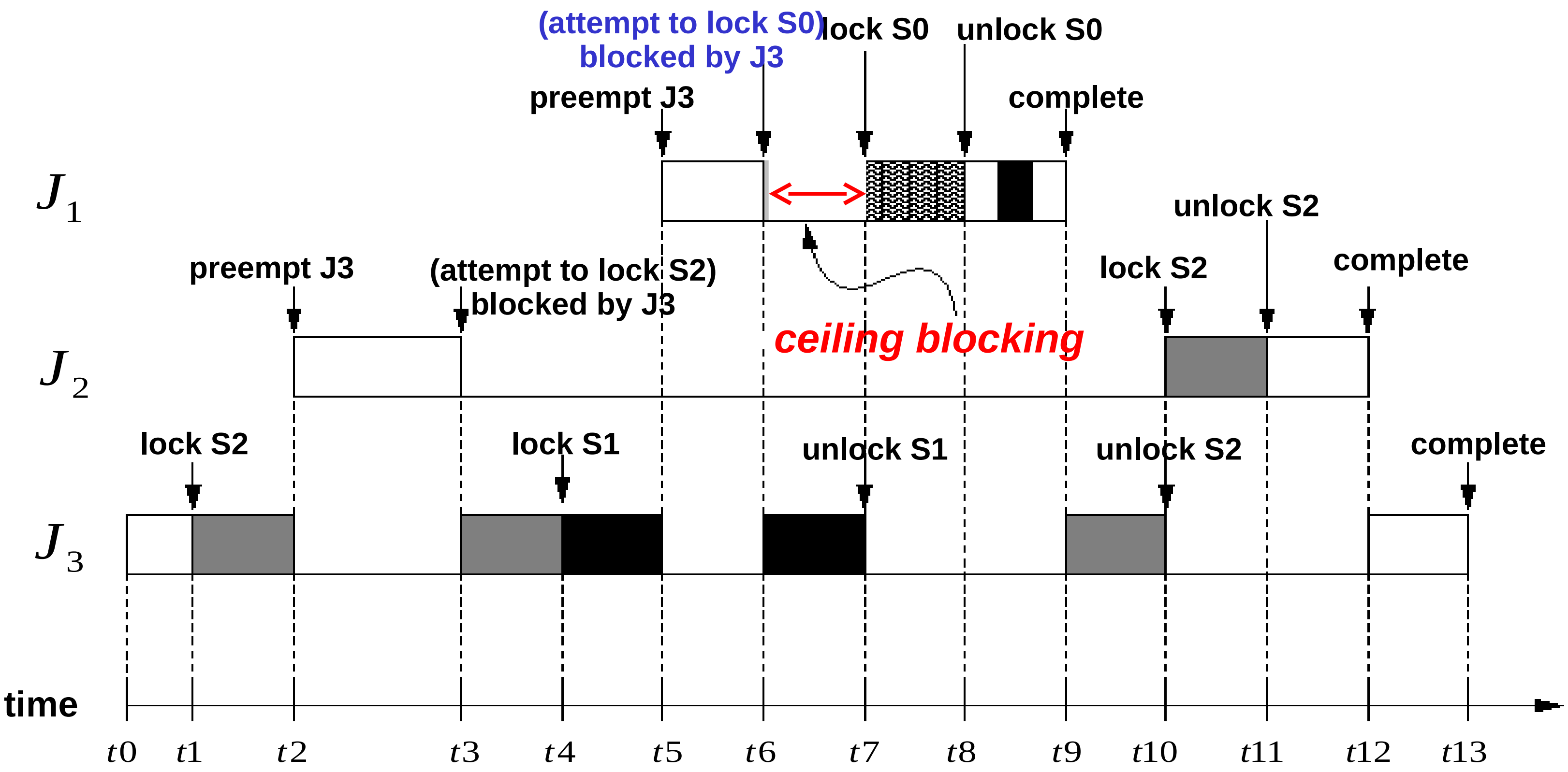
$$J_1 = \{\ldots, P(S_1), \ldots, V(S_1), \ldots, P(S_2), \ldots, V(S_2), \ldots\}$$
$$J_2 = \{\ldots, P(S_2)\;, \ldots, V(S_2), \ldots\}$$
$$J_3 = \{\ldots, P(S_1)\;, \ldots, V(S_1), \ldots\}$$



Examples for PCP (4/4)

- A New type of blocking: Ceiling blocking

$$J_1 = \{\ldots, P(S_0), \ldots, V(S_0), \ldots, P(S_1), \ldots, V(S_1), \ldots\}$$
$$J_2 = \{\ldots, P(S_2), \ldots, V(S_2), \ldots\}$$
$$J_3 = \{\ldots, P(S_2), \ldots, P(S_1), \ldots, V(S_1), \ldots, V(S_2), \ldots\}$$



Outline

- Introduction
- Assumptions and notations
- Basic priority inheritance protocol (BPI)
- **Priority ceiling protocol (PCP)**
  - Overview
  - Examples
  - **Properties**
  - Summary
- Schedulability analysis
- Conclusion

Properties of PCP (1/2)

- [Lemma 7: Blocking condition]: A job J can be blocked by a lowe r priority job J_L,
  - Only if the priority of job J is no higher than the highest priority ceilin g of all the semaphores that are locked by all lower priority jobs whe n J enters its critical section.
- [Lemma 8: Inheritance condition]: If $z_{j,n}$ of J_j is preempted by J_i w hich enters $z_{i,m}$,
  - J_j cannot inherit a priority level which is higher than or equal to that of J_i until J_i completes.
- [Lemma 9: No transitive blocking]: The priority ceiling protocol pr events transitive blocking.
- [Theorem 10: No deadlocks]: The priority ceiling protocol preven ts deadlocks.

Properties of PCP (2/2)

- [Lemma 11: Blocking duration from one lower priority job]: J_i can be blocked by J_L
  - For at most the duration of one critical section in $\beta^*_{i,L}$.
- [Theorem 12: Blocking duration]: J_i can be blocked
  - For at most the duration of at most one element of β^*_i.

→ The maximum blocking delay for a job is bounded by
  - The duration of the longest critical section among those of lower priority jobs

- [Corollary 13: Blocking duration]: If a generalized job J_i suspends itself n times during its execution, it can be blocked
  - By at most $n+1$ not necessarily distinct elements of β^*_i.

Summary of PCP

- Basic Idea
  - Priority ceiling of a semaphore:
    - The priority of the highest priority task that may use the semaphore
  - Additional condition for allowing a job J to start a new critical section
    - only if J's priority is higher than all priority ceilings of all the semaphores locked by jobs other than J .
- PCP solves the problems of BPI
  - No Deadlocks, no transitive blocking, and no blocking chains
- Three types of blocking
  - (1) Direct blocking, (2) push-through blocking, and (3) ceiling blocking
- The maximum blocking delay for a job is bounded by
  - The duration of the longest critical section among those of lower priority jobs

Outline

- Introduction
- Assumptions and notations
- Basic priority inheritance protocol (BPI)
- Priority ceiling protocol
- **Schedulability analysis**
  - Utilization bound based approach
  - Time demand based approach
- Conclusion

Schedulability Analysis

- Goal:
  - Extend the RMS algorithms
    - Considering the effect of blocking under PCP
- [Theorem 14: RMS theory of Liu & Layland]: Utilization bound based approach
- [Theorem 15: RMS theory of Lehoczky et al]: Time demand based approach

$$\forall i, 1 \le i \le n, \min_{(k,l)\in R_i} \left[\frac{1}{lT_k} \sum_{j=1}^{i} C_j \left\lceil \frac{lT_k}{T_j} \right\rceil \right] \le 1$$

, where $R_i = \{(k,l) \mid 1 \le k \le i, l = 1, \cdots, \lfloor T_i / T_k \rfloor \}.$

- Notation
  - B_i: The worst case blocking time of a job in task τ_i

Utilization Bound Based Approach

- [Theorem 16: Extension of Theorem 14]: A set of n periodic tasks using PCP can be scheduled by RMS if

$$\underline{\forall i, 1 \le i \le n,}\quad \frac{C_1}{T_1} + \frac{C_2}{T_2} + \cdots + \frac{(C_i + B_i)}{T_i} \le i(2^{1/i} - 1)$$

  - Proof
    - The inequality holds when there is no blocking.
    - Task τ_i still meets its deadline when its execution is delayed by B_i.
- [Corollary 17]: A set of n periodic tasks using PCP can be schedul ed by RMS if

$$\frac{C_1}{T_1} + \cdots + \frac{C_n}{T_n} + \max\left\{\frac{B_1}{T_1}, \ldots, \frac{B_{n-1}}{T_{n-1}}\right\} \le n(2^{1/n} - 1)$$

  - Proof
    - If this equation holds, then all the equations in [Theorem 16] holds.
      - $n(2^{1/n} - 1) \le i(2^{1/i} - 1)$
      - $\max\{B_1/B_1, \ldots, B_{n-1}/B_{n-1}\} \ge B_i/B_i$

Time Demand Based Approach

- [Theorem 18: Extension of Theorem 15]: A set of n periodic task s using PCP can be scheduled by RMS for all task phasings if

$$\forall i, 1 \le i \le n, \min_{(k,l) \in R_i} \left[\frac{1}{lT_k} \left(\sum_{j=1}^{i-1} C_j \left\lceil \frac{lT_k}{T_j} \right\rceil + (C_i + B_i) \right) \right] \le 1$$

, where $R_i = \{(k,l) \mid 1 \le k \le i, l = 1, \cdots, \lfloor T_i / T_k \rfloor\}.$

  – Proof
    - identical to that of proof for [Theorem 16]

Outline

- Introduction
- Assumptions and notations
- Basic priority inheritance protocol (BPI)
- Priority ceiling protocol
- Schedulability analysis
- **Conclusion**

Conclusion

- Problems in real-time synchronization
  – Uncontrolled priority inversion (→ unbounded blocking delay)
- Two Priority inheritance protocols
  1. BPI (Basic priority inheritance protocol)
     - prevents indefinite blocking → bonded blocking delay
     - 3 types of blocking
       – direct blocking, push-through blocking, and transitive blocking
     - Problems
       – Deadlocks, transitive blocking and blocking chains
  2. PCP (Priority ceiling protocol)
     - Solves problems of BPI
     - 3 types of blocking
       – direct blocking, push-through blocking, and ceiling blocking
     - At most one blocking with no transitive blocking for each job