TCSS 465A

Embedded Real-Time System Programming

Midterm - Spring 2007

Name **Key**

1) Name five properties that real-time systems must have to support critical applications:

A) **Timeliness – must meet deadlines appropriately**

B) **Design for peak-load – Can't crash under peak loads**

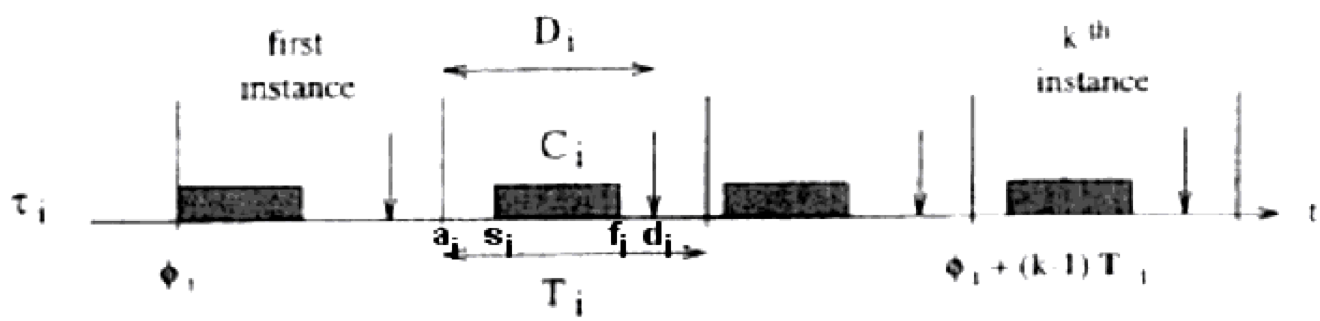
C) **Predictability – Must not have unanticipated surprises**

D) **Fault Tolerance – The system must be robust and fail-safe**

E) **Maintainability – Must be modifiable as requirements evolve**

2) A) CLEARLY show on a timing diagram the parameters for a real time periodic task:

- Compute time C_i
- Finish time f_i
- Arrival time a_i
- Phase Φ_i
- Start time s_i



B) Define in terms of the parameters above:

- Response time:

$$\mathbf{R_i = f_i - a_i}$$

- Average response time:

$$\mathbf{R_{ave} = 1/n \sum_{i=1}^{n} (f_i - a_i)}$$

- Lateness:

$$\mathbf{L_i = f_i - d_i}$$

- Maximum lateness:

$$\mathbf{L_{max} = \max_i (f_i - d_i)}$$

3) Clearly describe and distinguish between each of the following scheduling algorithms:

A) Earliest Due Date

Used for scheduling aperiodic, independent, non-preemptive tasks with synchronous arrival times
Provides an optimal schedule with respect to minimizing L_{max}
Tasks are scheduled in order of non-decreasing deadlines

B) Earliest Deadline First

Used for scheduling preemptive, independent, preemptive, dynamic tasks (arbitrary arrival times)
Provides optimal schedule with respect to minimizing L_{max}
Task which, at any instant in time, has the earliest absolute deadline among ready tasks has priority

C) Latest Deadline First

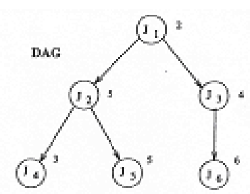
Used for scheduling aperiodic, non-preemptive tasks with synchronous arrival times and precedence constraints
Provides optimal schedule with respect to minimizing L_{max}
Tasks are scheduled backwards with the task having the latest deadline scheduled "first"

D) Spring Algorithm

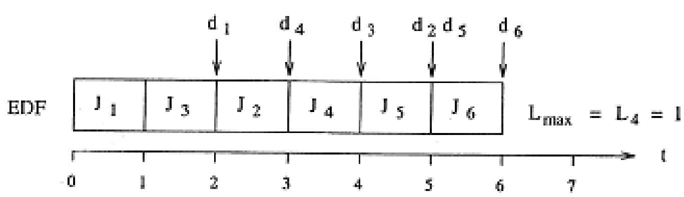
Used to schedule tasks with arbitrary arrivals, non-preemptive, and with various constraints:
priorities, precedence relations, resource constraints, etc.
Seeks to find feasible schedule
Tasks are scheduled based upon an optimization driven by successive application of a heuristic function

4) Consider the following set at tasks with precedence constraints:

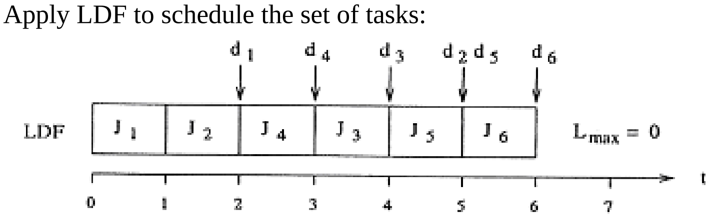
	J_1	J_2	J_3	J_4	J_5	J_6
C_i	1	1	1	1	1	1
d_i	2	5	4	3	5	6



A) Apply EDF to schedule the following set of tasks



B) Apply LDF to schedule the set of tasks:



5) Consider the following set of periodic tasks:

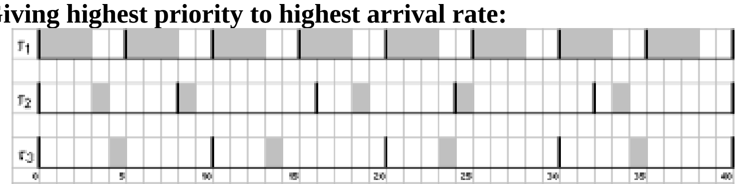
	C_i	T_i
τ_1	3	5
τ_2	1	8
τ_3	1	10

A) Verify the schedulability according to the Rate Monotonic algorithm

$$U = \frac{3}{5} + \frac{1}{8} + \frac{1}{10} = 0.825$$ **< = 1 , so it is potentially schedulable**.

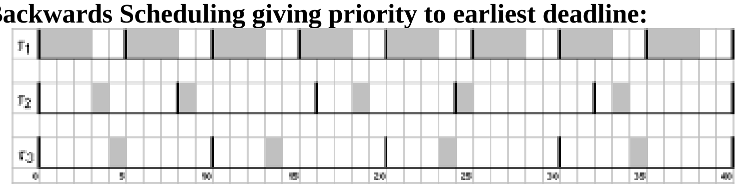
B) Construct the schedule using Rate Monotonic.

Giving highest priority to highest arrival rate:



C) Construct the schedule using Earliest Deadline First.

Backwards Scheduling giving priority to earliest deadline:



6) A) Describe Priority Inversion and clearly show an example of it:

To reduce worst case blocking time and to prevent deadlocks, a task is not allowed to enter a critical section if there are locked semaphores that could block it.

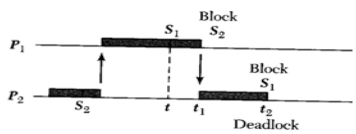


Figure 6.8 Deadlock example.

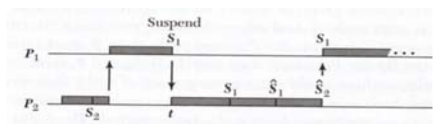


Figure 6.10 Priority ceiling: No deadlock.

B) Define the Priority Ceiling Protocol.

1) **Each semaphore is assigned a *priority ceiling* equal to the highest priority task that can lock it.**

2) **A task J is allowed to enter a critical section only if its priority is higher than all priority ceilings of the semaphores locked by tasks other than task J. While holding a semaphore, J inherits the highest priority of any job that is blocked by it.**

3) **When a task J exits a critical section, it unlocks the semaphore and the highest-priority task waiting on that semaphore is awakened. The priority of J is set to the highest priority of all tasks that are blocked by it, or otherwise returned to its nominal priority.**

C) Given the Multilevel Blocking Mutual Exclusion problem:

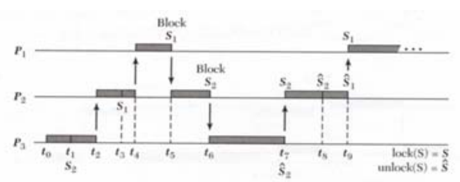


Figure 6.7 Multilevel blocking.

Show how the Priority Ceiling Protocol will improve the scheduling.

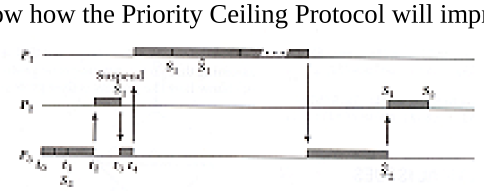


Figure 6.9 Priority ceiling: blocking.