ECE 555
Real-Time Embedded Systems

Real-Time Scheduling - II

Prof. Xiaorui Wang

Recap of Last Class
• Process scheduling problem
• Terminologies and timing parameters
  • Task, job
• Metrics to evaluate scheduling algorithms
  • Schedulability, overhead
• Optimal scheduling algorithms
  • RMS, EDF
  • Assumptions
• CPU utilization analysis
  • Utilization bound
  • RMS utilization bound

Optimal Scheduling Algorithms
• Rate Monotonic Scheduling (RMS)
  • Higher rate (=1/period) → Higher priority
  • Optimal preemptive static priority scheduling algorithm
• Earliest Deadline First (EDF)
  • Earlier absolute deadline → Higher priority
  • Optimal preemptive dynamic priority scheduling algorithm

RMS Missing a Deadline
• T1 = (10,20), T2 = (15,30), utilization is 100%

EDF Meeting a Deadline
• T1 = (10,20), T2 = (15,30), utilization is 100%

EDF Utilization Bound
• \( U_0 = 1 \)
• \( U \leq 1 \) is a sufficient and necessary condition for schedulability.
**EDF Implementation**

- Whenever a job is released / finished:
  - Compute absolute deadline for each job;
  - Sort jobs based on absolute deadlines;
  - Change process priorities on the fly.
- Generally considered too expensive to use in practice.
- Harder to analyze than RMS

**EDF Evaluation**

- Schedulability
  - EDF can guarantee schedulability as long as CPU is not fully utilized
  - Higher overhead than RMS
  - Task priorities may need to be changed online
  - Optimal dynamic priority scheduling algorithm

**Assumptions**

- Single processor.
- All tasks are periodic.
- Zero context switch time.
- Relative deadline = period.
- No priority inversion.
- What if relative deadline < period?

**Optimal Scheduling Algorithms**

Relative Deadline < Period

- Deadline Monotonic Scheduling (DMS)
  - Shorter relative deadline $\Rightarrow$ Higher priority
  - When relative deadline = period, DMS becomes RMS
  - Optimal preemptive static priority scheduling
- Earliest Deadline First (EDF)
  - Earlier absolute deadline $\Rightarrow$ Higher priority
  - Optimal preemptive dynamic priority scheduling algorithm

**DMS Analysis**

- Utilization bound of RMS (Deadline = Period)
  $$U = \sum_{j=1}^{n} \frac{C_j}{P_j} \leq n(2^{1/n} - 1)$$
  - $n$: number of tasks on the processor.
- Sufficient but pessimistic test for DMS (Deadline < Period)
  $$\sum_{j=1}^{n} \frac{C_j}{D_j} \leq n(2^{1/n} - 1)$$
  - Sufficient and necessary test: response time analysis

**Response Time Analysis**

- Idea
  - Longest possible response time < relative deadline
  - Assume fixed-priority scheduling
- Critical instant
  - Results in a task’s longest response time.
  - Occurs when all higher-priority tasks are released at the same time as the task.

interfering tasks

Critical instant
EDF: CPU Time Demand Analysis

- To start, assume \( D_i = P_i \)
- CPU time demand in interval \([0, L]\)
  - Total execution time needed for completing all jobs with deadlines no later than \( L \)
    \[
    C_i(0, L) = \sum_{j=1}^{n} \frac{L}{P_j} C_j
    \]
  - For one task: # of released jobs multiplying execution time
  - For all tasks: summation

Schedulable Condition

- A set of periodic tasks is schedulable by EDF if and only if for all \( L \geq 0 \):
  \[
  L \geq \sum_{i=1}^{n} \frac{L}{P_i} + 1 \cdot C_i
  \]
- Real CPU time \( \geq \) demanded CPU time
- There is always enough time to satisfy CPU time demand at every time point.
- When \( D_i < P_i \)
  \[
  \forall L \in D_i, \quad L \geq \sum_{i=1}^{n} \frac{L-D_i}{P_i} + 1 \cdot C_i
  \]

Schedulability Test Revisited

<table>
<thead>
<tr>
<th>Assumptions</th>
<th>Static Priority</th>
<th>Dynamic Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>( D = P )</td>
<td>RMS (CPU utilization bound; Response time analysis)</td>
<td>EDF (CPU utilization bound)</td>
</tr>
<tr>
<td>( D &lt; P )</td>
<td>DMS (Response time analysis)</td>
<td>EDF (CPU time demand analysis)</td>
</tr>
</tbody>
</table>

Priority Inversion

- A lower-priority task blocks a higher-priority task from running.
- Sources of priority inversion
  - Access shared resources guarded by semaphores
    - Lower-priority task gets the resource first
  - Access non-preemptive subsystems
    - Communication subsystems
    - Storage

Semaphores

- OS primitive for controlling access to critical regions.
- Get access to semaphore \( S \) with \( \text{wait}(S) \).
- Execute critical section to access shared resource.
- Release semaphore with \( \text{signal}(S) \).
- Example:
  ```
  event result_t Timer.fired()
  {  
    wait(busy);  
    call ADC.getData();  
    signal(busy);  
    return SUCCESS;  
  }
  ```
What Happened to Mars Pathfinder?

- July 4th, 1997 landing on the Mars
  - But a few days into the mission, not long after Pathfinder started gathering meteorological data, the spacecraft began experiencing total system resets, each resulting in losses of data.

- Priority inversion
  - Bus management (high priority) is blocked by data gathering (low priority) and long-running communications (medium priority)
  - Watchdog timer resets system after bus is blocked for some time

Real-World (Out of This World) Story:
http://research.microsoft.com/~mbj/Mars_Pathfinder/Mars_Pathfinder.html

Solution: Priority Inheritance

- Let the low-priority task inherit the priority of the blocked high-priority task.

Priority Inheritance Protocol

- When task \( T_i \) is blocked on a semaphore:
  - \( \text{prior}(T_j) \rightarrow T_i \) holding the semaphore if \( \text{prior}(T_i) < \text{prior}(T_j) \)

- When \( T_j \) releases a semaphore:
  - If \( T_j \) does not block any other tasks, it returns to its normal priority
  - If \( T_j \) still blocks other tasks, it inherits the highest priority among the blocked tasks.

- Priority Inheritance is transitive
  - \( T_j \) blocks \( T_i \) and inherits \( \text{prior}(T_i) \)
  - \( T_{i+1} \) blocks \( T_j \) and inherits \( \text{prior}(T_j) \)

- Supported by most RTOS (e.g., VxWorks)

Summary

- Earliest Deadline First (EDF)
  - Utilization bound, implementation, evaluation

- Optimal scheduling algos when relative deadline < period
  - Deadline Monotonic Scheduling (DMS)
    - Critical instant analysis
  - Earliest Deadline First (EDF)
    - Processor demand analysis

- Priority inversion
  - Sources
  - Unbounded priority inversion
  - Priority inheritance protocol
Reading

• Textbook: Chapters 6.1 to 6.4
• Priority inversion on Mars
  • http://research.microsoft.com/~mbj/Mars_Pathfinder/
    Mars_Pathfinder.html