Announcement
• Mid-term course evaluation will be open till March 26th (Saturday)
• Feedbacks are welcome!

Real-Time Scheduling
• What're the optimal scheduling algorithms?
• Can we meet all deadlines?

Benefit of Scheduling Analysis
<table>
<thead>
<tr>
<th>VEST (UVA)</th>
<th>Baseline (Boeing)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design - one processor 40</td>
<td>Design - one processor 25</td>
</tr>
<tr>
<td>Implementation - one processor 75</td>
<td></td>
</tr>
<tr>
<td>Scheduling analysis - MUF 1</td>
<td>Timing test 30</td>
</tr>
<tr>
<td>Design - two processors 20</td>
<td>Design - two processors 90</td>
</tr>
<tr>
<td>Implementation - two processors 105</td>
<td></td>
</tr>
<tr>
<td>Scheduling analysis - DM/ Offset 1</td>
<td>Timing test 20</td>
</tr>
<tr>
<td>&quot;Implementation&quot; 105</td>
<td>Total composition time 172</td>
</tr>
<tr>
<td>Total composition time 345</td>
<td></td>
</tr>
</tbody>
</table>

• Schedulability analysis reduces composition time by 50%!
  - Reduce wasted implementation/testing rounds
  - Analysis time << testing
• More reduction expected for more complex systems
  => Quick exploration of design space!

Consequence of Deadline Miss
• Hard deadline
  - System fails if missed.
  - Goal: Guarantee no deadline miss.
• Soft deadline
  - User may notice, but system doesn’t necessarily fail.
  - Goal: Meet most deadlines most of the time.

Comparison
• General-purpose systems
  - e.g., PCs, database servers
  - Fairness to all tasks (no starvation)
  - Optimize throughput
  - Optimize average performance
• Embedded systems
  - Meet all deadlines.
  - Fairness or throughput is not important
  - Hard real-time: worry about worst case performance

Terminology
• Task
  - May corresponds to a process or thread
  - May be released multiple times
• Periodic task
  - Ideal: inter-arrival time = period
  - General: inter-arrival time >= period
• Aperiodic task
  - Inter-arrival time does not have a lower bound
• Job: an instance of a task
Timing Parameters

- Task \( T_i \)
- Period \( P_i \)
- Worst-case execution time \( C_i \)
- Relative deadline \( D_i \)
- Job \( J_{ik} \)
  - Release time: time when a job is ready
  - Response time \( R_i = \text{finish time} - \text{release time} \)
  - Absolute deadline = release time + \( D_i \)
  - A job misses its deadline if \( \text{Response time } R_i > D_i \)

Metrics

- Schedulability
  - A task set is schedulable under a scheduling algorithm if all jobs can meet their deadlines
- Overhead
  - Time required for scheduling decision and context switches.

Optimality

- A scheduling algorithm \( S \) is optimal if
  - a task set is not schedulable under \( S \) \( \rightarrow \) it is not schedulable under any other algorithms

Optimal Scheduling Algorithms

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

Assumptions

- Single processor.
- All tasks are periodic.
- Zero context switch time.
- Relative deadline = period.
- No blocking.
- RMS and EDF have been extended to cases with relaxed assumptions

Utilization Bound

- Utilization of a processor:
  \[
  \hat{U} = \sum_{j \in S} \frac{C_j}{P_j}
  \]
  - where \( S \) is the set of tasks on the processor.
- Utilization bound \( U_b \): All tasks are guaranteed to be schedulable if \( U \leq U_b \)
**Necessary Condition**
- No scheduling algorithm can schedule a task set if $U > 1$
- $U_b \leq 1$
- An algorithm is optimal if its $U_b = 1$

**RMS Utilization Bound**
- $U_b(n) = n \left( \frac{2}{n-1} \right)$
- $n$: number of tasks
- $U_b(2) = 0.828$
- $U_b(n) \geq U_b(\infty) = \ln 2 = 0.693$
- $U \leq U_b(n)$ is a sufficient condition, but not necessary in general cases.
- $U_b = 1$ if all process periods are harmonic, i.e., periods are multiples of each other
  - e.g., 1, 10, 100

**RMS**
- RMS may not guarantee schedulability even when CPU is not fully utilized
- Low overhead: When tasks are fixed, priorities are never changed

**EDF Utilization Bound**
- $U_b = 1$
- $U \leq 1$ is a sufficient and necessary condition for schedulability.

**EDF**
- EDF can guarantee schedulability as long as CPU is not fully utilized
- Higher overhead than RMS: Task priorities may need to be changed online

**Assumptions**
- Single processor.
- All tasks are periodic.
- Zero context switch time.
- Relative deadline = period.
- No blocking.
- What if relative deadline < period?
Deadline Monotonic Scheduling (DMS)

- Shorter relative deadline → Higher priority
- Optimal preemptive static priority scheduling when relative deadline < period

Response Time Analysis

- 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.

/* Tasks are ordered by priority; T1 has the highest priority */
for (each task Tj)
{
    I = 0; R = 0;
    while (I + Cj > R)
    {
        R = I + Cj;
        if (R > Dj) return UNSCHEDULABLE;
    }
    I = \sum_{k=1}^{j-1} \left\lfloor \frac{R}{P_k} \right\rfloor C_k;
}
return SCHEDULABLE;