Announcement

• Course website
• Presentation topics
  • By 09/09
  • I will randomly assign a topic to you after 09/09
• Mailing list
  • Used for announcements, assignments, questions & answers, or any changes

Project

• Usually 2 students per team
  • Can use mailing list to help teaming
• Possible topics
  • Develop a real-time application
  • Design a power/thermal control algorithm
  • Design a real-time WSN protocol
• What to do?
  • Develop/integrate software or hardware
  • Perform experiments on real systems or simulators
  • Write a paper
  • Do a demo to the class

Project Topics

• Real-time applications
  • New (e.g., power-aware) scheduling algorithm
  • Real-time information dissemination, database
• Power/thermal control algorithms
  • Power control for multi-core systems
  • Thermal control for computing systems
  • Power-efficient systems
• Real-time WSN protocols
  • Multi-channel, multi-radio real-time routing algorithms
• New ideas
  • Feel free to propose anything interesting

Milestones

1. Team forming: Midnight, Sep. 04, next Tuesday
   • Use mailing list to find a teammate
   • I will randomly assign you to a team after that time
2. Proposal presentation (5%): Midnight, Sept. 16
3. Midterm presentation (10%): Midnight, Oct. 14
4. Final presentation / demo (15%): Midnight, Nov. 18
5. Final Report (20%): before final exam

Project Proposal

• Need to discuss with me before you finalize
• One proposal/team
• Presentation (4 or 5 slides)
  • 8 min presentation + 2 min questions
  • Challenges.
  • Description of project.
  • Goals (i.e. what do you hope to accomplish?).
• 5% of your final score
• Due: Midnight, 09/16

Project Final Report

• Submit electronic versions to me.
• Final Report
  • Similar to research papers in the reading list.
  • 10 pages, double column, 10 pts fonts.
  • Templates are online:
    https://www.ece.utk.edu/~xwang/ece455/format.html
• Source code and documents
  • Documents: README, INSTALL, HOW-to-RUN
  • Separate from final report.
  • Submit with final report.
Advice

• Start early and work on it regularly!
• Discuss with me often for feedbacks and pointers

Recap of Last Class

• Operating systems
• Process states and management
• Context switch
• Process scheduling
• POSIX
• Real-Time OS
  • Proprietary kernels
  • Real-time extensions to general-purpose OS
    • Real-time Linux
    • Linux scheduling

Process Scheduling

Embedded vs. General-Purpose

• 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

The Scheduling Problem

• Can we meet all deadlines?
  • Must be able to meet deadlines in all cases.
• How much CPU horsepower do we need to meet our deadlines?
• Timing violations: What happens if a process doesn’t finish by its deadline?
  • Hard deadline: system fails if missed.
  • Soft deadline: user may notice, but system doesn’t necessarily fail.

Terminologies Used in Scheduling

• Task
  • May correspond 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**
- Task T_i
  - Period P_i
  - Worst-case execution time C_i
  - Relative deadline D_i
    - Usually equal to period

**Timing Parameters - Job**
- Job J_{ik} (denoted as T_{i,k})
  - Release time: time when a job is ready
  - Response time R_i = finish time – release time
  - Absolute deadline = release time + D_i
  - A job misses its deadline if
    - Response time R_i > D_i
    - Finish time > absolute deadline

**Metrics to Evaluate Scheduling Algorithms**
- 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 → it is not schedulable under any other algorithms

**RMS - Static Optimal Algorithm**
- Rate Monotonic Scheduling (RMS)
  - Higher rate (1/period) → Higher priority
  - Optimal preemptive static priority scheduling algorithm

**EDF – Dynamic Optimal Algorithm**
- Earliest Deadline First (EDF)
  - Earlier absolute deadline → Higher priority
  - Optimal preemptive dynamic priority scheduling algorithm
Assumptions
• Single processor.
• All tasks are periodic.
• Zero context switch time.
• Relative deadline = period.
• No priority inversion.
• RMS and EDF have been extended to cases with relaxed assumptions

RMS - Rate Monotonic Scheduling
• Common way to assign priorities
• Result from Liu & Layland, 1973 (JACM)
• Simple to understand and implement:

  Processes with shorter period are given higher priority

  E.g.,
<table>
<thead>
<tr>
<th>Period</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>3</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
</tr>
</tbody>
</table>

CPU Utilization Analysis
• Utilization of a processor:
  \[ U = \sum_{i=1}^{n} \frac{C_i}{P_i} \]
  n: number of tasks on the processor.

  \[ \text{CPU utilization of } T_1 = \frac{3}{5} = 60\% \]

  CPU utilization of T1 = 3/5 = 60%

    -- 0 5 10 time --
    \( C_1 = 3 \text{ sec} \)
    \( T_1 = 5 \text{ sec} \)

Schedulable Utilization Bound
• Utilization bound \( U_b \)
  • All tasks are guaranteed to be schedulable if \( U \leq U_b \)
  • \( U \) is the requested utilization of a task set
  • Necessary condition for scheduling
    • 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 is the optimal static priority scheduling algorithm but not overall optimal

RMS Utilization Bound
• \( U_b(n) = n(2^{\frac{n}{2}}-1) \)
  • \( n \): number of tasks
  • \( U_b(1) = 1 \)
  • \( U_b(2) = 0.828 \)
  • \( U_b(n) \geq U_b(\infty) = \ln 2 = 0.693 \)

  • \( U \leq U_b(n) \) 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 Missing a Deadline
• \( T_1 = (10,20), T_2 = (15,30), \) utilization is 100% > RMS bound

  Job 1 of T2 misses its deadline

  Would have met the deadline if \( T_2 = (10,30), \) utilization reduced 83%
RMS Meeting the Deadline

• T1 = (10,20), T2 = (10,30), utilization is 83%

RMS Evaluation

• Schedulability
  • RMS may not guarantee schedulability even when CPU is not fully utilized
• Low overhead
  • When tasks are fixed, priorities are never changed
• Optimal static priority scheduling algorithm

Summary

• 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

Reading

• Textbook: chapters 3.1 to 3.4