SFWR ENG 4AA4

Kemal Ahmed
Fall 2015
Dr. Down

Note: information from the pre-requisite, SFWR ENG 3DX4 will not be included in this summary (although corrections will be).

Contents

Real-Time Systems

Classifications

What happens upon failure to meet deadlines:

- **Soft**: performance is degraded but not destroyed
- **Firm**: a few times will simply degrade performance, but after may lead to system failure
- **Hard**: complete and catastrophic system failure
    - **Safety Critical**: may cause injury / death (a type of hard)

Forward difference method: derivatives using $f'(x) = \frac{f(x+h) - f(x)}{h}$

Backwards Difference method: derivatives using $f'(x) = \frac{f(x) - f(x-h)}{h}$

Controller [C(s)]:
Input [E(s)]:
Output [U(s)]:
$U(s) = C(s)E(s)$

Task optimization

Task [T]: $T_i = (p_i, r_i, e_i, d_i)$

Period [p]: time between tasks are repeatedly released
Release time [r]: time it takes to release task
Execution time [e]: slowest time task could take to be completed (but assume the tasks will take this long no matter what)
Deadline [d]: when task <u>needs</u> to be completed

Number of tasks [n]:

Processor Utilization [U]: used as a priority level $U = \sum_{i=1}^{n} \frac{e_i}{p_i}$

If $r_i = 0$ and $p_i = d_i$, then write $T_i = (p_i , e_i)$

Types of Scheduling

Static

Static Scheduling:

- task's priority is assigned before execution and does not change

FIFO

First In First Out (FIFO):

- Could cause problems for tasks whose execution time is significantly shorter than the rest when there are deadlines
    - E.g. $T_1 = (100, 3)$; $T_2 = (2, 1)$
- A.K.A. **First Come, First Served (FCFS)**

Schedule: the order in which tasks will be executed
Hyperperiod [H]: the entire length of a cycle, least common multiple

Frame Size [f]:

- The best way for computers to segment the schedule in a way that it verify that the appropriate tasks have been executed
- Constraints:
    1. $f \geq \max_{1 \leq i \leq n} (e_i)$
    2. H % f = 0
    3. $2f - \gcd(p_i , f) \leq d_i$

Least Compute Time (LCT): tasks with smallest execution times executed first

- Think *greedy*
- Works poorly; worse than RR

Rate Monotonic (RM): shorter period, higher priority

- Think: tasks requiring frequent attention should have higher priority

- *Static scheduling* can be guaranteed to be *feasible* using Rate Monotonic scheduling if: $U \leq n\left(2^{\frac{1}{n}} - 1\right)$
    - If the equation fails, it still might be, so draw the whole thing to be safe.

Period Attributes

Harmonic: every task period evenly divides every longer period

- Always feasible with RM schedule

Dynamic

Preempting: splitting tasks up into mini tasks

The only two optimal dynamic priorities are:

- **Earliest Deadline First (EDF)**:
    - more flexible, better U
    - If deadlines < periods, still optimal, but determining feasibility is NP-hard
- **Least Slack Theorem (LST)**: not as popular as EDF

Multiprocessor

Once you have multiple processors, neither EDF nor RM are guaranteed to work.

Look into first-fit algorithms

Priority Inversion: