

CS418 — Operating Systems

Lecture 13

Real-time Scheduling

Textbook: Operating Systems
by William Stallings

1. Real-time Scheduling Examples

- Real-time scheduling requires not only the correctness of logical computation but also timing
- Examples
 - 1. Process control plants.
 - 2. Robotics.
 - 3. Aircraft control.
 - 4. Cruise missile.
 - 5. etc.

2. Basic concepts

- **Hard real-time task:** one which we must meet its deadline; otherwise, fatal damage or error will occur.
- **Soft real-time task:** one which we should meet its deadline, but not mandatory. We should schedule it even if the deadline is already passed.
- **Aperiodic task:** a somehow 'random' task which may have a constraint on start time or finish time or both.
- **Periodic task:** a sequence of tasks which appear 'once per period T '.

3. Characteristics of real-time OS

- Determinism
 - 1. Multi-process system is in general non-deterministic.
 - 2. Real-time OS should respond by external events/timing, hence should be deterministic.
 - 3. Determinism is determined by the speed the OS responds to interrupts, as well as the capacity of the system.
 - 4. Maximal **delay** is small: microseconds to a millisecond.
- Responsiveness — how long it takes the OS to service the interrupt
 - 1. Time required to start interrupt.
 - 2. Time to finish the interrupt.
 - 3. Is nested interrupt allowed?
- User Control
 - 1. User control should be processed immediately.
 - 2. Should even allow the user to specify hard/soft tasks.
- Reliability
 - 1. Reliability is much more important for real-time systems than regular systems.
 - 2. Error generally not recoverable.

- Fail-soft Operation

- **1.** For some soft tasks, failure is allowed.
- **2.** Ability to preserve as much capacity and data as possible (when failure occurs).
- **3.** Try to either correct the problem or minimize its effects.
- **4. Stability**—when it is impossible to meet all deadlines, system will satisfy the most critical tasks.

4. Features of modern real-time OS

- Fast process/thread switch
 - Small size
 - Responds to external interrupts quickly
 - Preemptive scheduling based on priority
 - Primitives to delay tasks for limited time
 - Special alarms and time-outs
 -
-
- The most important thing in real-time OS is to start hard tasks by their deadline and finish them by their deadlines.

5. Deadline Scheduling

- **Ready time:** Time at which a task becomes ready to run
 - **Starting deadline:** Time by which a task must start
 - **Completion deadline:** Time by which a task must complete
 - **Processing time:** Time to actually serve a task
 - **Resource requirements:** Resources required by a task
 - **Priority:** Importance of a task
-
- On either a uniprocessor or a multiprocessor, scheduling tasks with the earliest deadline gives us an optimal solution.
 - An example on scheduling periodic tasks
 - An example on scheduling aperiodic tasks

6. Rate Monotone Scheduling

- A task's **period**, T , is the time between the arrival of two tasks (within the same sequence).
- A task's **rate** is $1/T$.
- A task's **computation** time, C , is the time to process each occurrence of the task.
- On a uniprocessor system, $C \leq T$.
- If a task can run to completion, the corresponding processor utilization is C/T .

- **RMS** always ranks a task with the shortest period as having the highest priority.

- If we have n tasks, each with a fixed period and execution time, then clearly

$$\frac{C_1}{T_1} + \frac{C_2}{T_2} + \dots + \frac{C_n}{T_n} \leq 1.$$

We can even prove that

$$\frac{C_1}{T_1} + \frac{C_2}{T_2} + \dots + \frac{C_n}{T_n} \leq n(2^{1/n} - 1).$$

- RMS is popular in practice because
 - **1.** The performance difference is small.
 - **2.** It can handle a mixture of hard real-time tasks and soft real-time tasks.
 - **3.** It is stable.