COSI 131a: Operating Systems

CPU Scheduling (3)

Chapter 6

Agenda

1. Scheduling Overview
   1. *First Example of Resource (CPU) Management (Sharing)*
   2. *Non-Preemptive (N) vs. Preemptive Scheduling (P)*
   3. *Metrics: Ways to Assess Effectiveness of Scheduling Policies*

2. Scheduling Policies

   1. *First-Come-First-Served (FCFS) (N)*
   2. *Shortest-Job-First (SJF) (N, P)*

   3. *Priority (N, P)*
   4. *Round-Robin (RR) (P)*
   5. *Multilevel Queues (MLQ) (P)*
   6. *Real-time*

3. Examples

Real-Time Scheduling

Idea:

"Real-time" = processes must / should complete by some deadline

Hard Real-Time (Must):

e.g.: adjust landing gear on an aircraft

Soft Real-Time (Should):

e.g.: deliver audio / video packets

Soft Real-Time:

- *can be integrated into Multi-level Feedback Queues*

Hard Real-Time:

- *requires dedicated scheduler*

Real-Time Scheduling

Hard Real-Time Example:

Process	Burst	Deadline
P_1	350	575
P_2	125	550
P_3	475	1000
P_4	250	3000
P_5	75	200

Idea:

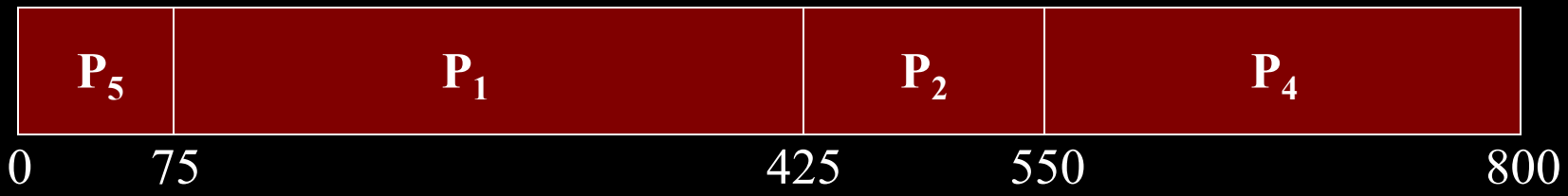
1. *Determine "feasible" processes (deadline – now ≥ burst)*
   - *Re-determine at every context switch*

2. *Greedy heuristic:*
   Choose process with min value of [deadline – (now + burst)]

Real-Time Scheduling

Hard Real-Time Example:

Process	Burst	Deadline
P_1	350	575
P_2	125	550
P_3	475	1000
P_4	250	3000
P_5	75	200



Real-Time Scheduling

Soft Real-Time:

- *Can be integrated into priority-based scheduler (e.g.: MFQ's)*

  → *should not be demoted*

  → *requires <u>strict</u> time-slices*

  (Not always the case: Interrupts only recognized when instructions complete, some instructions are lengthy)

  can make more strict w/ uniform instruction costs or interruptible system calls

Agenda

1. Scheduling Overview
   1. *First Example of Resource (CPU) Management (Sharing)*
   2. *Non-Preemptive (N) vs. Preemptive Scheduling (P)*
   3. *Metrics: Ways to Assess Effectiveness of Scheduling Policies*

2. Scheduling Policies

   1. *First-Come-First-Served (FCFS) (N)*
   2. *Shortest-Job-First (SJF) (N, P)*

   3. *Priority (N, P)*
   4. *Round-Robin (RR) (P)*
   5. *Multilevel Queues (MLQ) (P)*
   6. *Real-time*

3. Examples

Examples of Scheduling

Windows XP:

Priority-based, doubly preemptive. Preemption occurs with both:

1. *End of time quantum*
2. *Arrival of higher-priority thread*

Thread priority is dynamic. Determined by:

Class: *statically assigned*

Relative Priority: *dynamically*

→ *diminishes with expiration of time quantum*

→ *boosts with satisfaction of I/O request*

(Foreground (FG) vs. Background (BG) Process:

Doesn't affect priority, but time quantum of FG = 3x BG

Examples of Scheduling

Windows XP (cont.):

RELATIVE \ *CLASS*	*REAL TIME*	*HIGH*	*ABOVE NORMAL*	*NORMAL*	*BELOW NORMAL*	*IDLE*
TIME-CRITICAL	31	15	15	15	15	15
HIGHEST	26	15	12	10	8	6
ABOVE NORMAL	25	14	11	9	7	5
NORMAL	24	13	10	8	6	4
BELOW NORMAL	23	12	9	7	5	3
LOWEST	22	11	8	6	4	2
IDLE	16	1	1	1	1	1

→ *Many ways for 1 thread to get better treatment than other*

1. *Better* `CLASS`
2. *Better relative priority*
3. *Foreground (FG) process*

Examples of Scheduling

Linux:

Two classes of processes: Real-time, Timeshared

Real-time:

- → *soft*
- → *highest-priority runs first*
- → *within same priority class, FCFS or RR*

Timesharing:

- → *"credits" (similar to aging)*
- → *initial credits based on priority, history*
- → *when thread is time-sliced out, loses credit*
- → *scheduler chooses thread with most credits*

Examples of Scheduling

Java Threads (JVM)

Preemptive, priority-based scheduling

→ *preemption: arrival of higher priority thread*
(no requirement for timer interrupts)

`yield()`:
Running thread yields to another thread of equal priority

Thread priorities

```
Thread.MIN_PRIORITY
Thread.MAX_PRIORITY
Thread.NORM_PRIORITY
```

Set by: `setPriority` *method:*
(e.g.: `setPriority (Thread.NORM_PRIORITY + 2)`*)*