CG2271

Real-Time Operating Systems

Lecture 6

Scheduling Algorithms

colintan@nus.edu.sg

NUS
National University
of Singapore

School of Computing

Learning Objectives

- **By the end of this lecture you will:**
  - Understand the special needs of real-time tasks.
  - Understand how the RTOS can guarantee that these needs are met.
  - Be able to model the timing constraints of tasks and test if tasks are “schedulable”.

Introduction

- **Scheduling Algorithms:**
  - Given a set of *n* tasks (thread or process), a scheduling algorithm assigns priorities to decide which tasks should run next.
- **In this lecture we will look at 4 common real-time scheduling algorithms:**
  - Fixed priority
    - ✓**Tasks are picked and run in a fixed manner.**
  - Round-robin
    - ✓**Current highest priority task becomes lowest priority next time, and all other tasks are promoted by 1 position.**

Introduction

- Rate Monotonic Scheduling
  - ✓**Tasks are picked according to their rates of execution.**
- Earliest Deadline First
  - ✓**Tasks are picked according to who has the closest deadline.**

NUS National University of Singapore
School of Computing

Source

- **All materials are taken from "Embedded Real Time Systems: Introductory Concepts and Tools".**
  - By Graham Leedham and Kim-Tian Seow
  - ISBN 981-244-737-7
- **Materials are copyrighted by the authors and Pearson Education.**

Scheduling Algorithms

BASIC IDEAS AND ASSUMPTIONS

Assumptions

- **We will make the following assumptions:**
  - The system has a fixed number of tasks.
  - All tasks have known temporal scopes.
  - All tasks are periodic.
  - All tasks are independent.
  - Task switching overheads are negligible.
  - All task deadlines are equal to their periods.
  - All task periods are multiples of a unit period.
    - ✓**Defined to be the shortest schedulable period.**

Definitions

- **Co-operative multitasking:**
  - Each task runs for as long as it needs to.
  - When the task completes what it's doing, it surrenders control to the kernel, which then picks another task to run.
- **Pre-emptive multitasking:**
  - A task will run until a higher priority task is READY. When this happens.
    - ✓**The OS actively moves the current task to READY.**
    - ✓**It then picks the higher priority task and moves it to RUN.**

Definitions

- **CPU Time C_i**
  - The amount of CPU time consumed by process i each time it is run.
- **Period P_i**
  - The time between which the task expects to run.
  - Note that the task doesn't necessary run immediately when its period is due.
    - ✓**There may be a higher priority task that is also due at the same time.**

Definitions

- **Deadline D_i**
  - The "deadline" of a process i is defined to be the time by which the task must finish what it is doing.
  - Usually assumed to be the same timing as P_i.
    - ✓**If a task gets pre-empted regularly and takes longer than P_i time to complete, it "misses" its deadline.**
    - ✓**Alternatively, if the task starts too late, it may also miss its deadline.**

NUS National University of Singapore
School of Computing

Definitions

- **For our example, we will use tasks with the following specifications:**

	Ci	Pi
P1	1	4
P2	2	8
P3	3	12

Scheduling Algorithms

- **In the following examples, all tasks are ready to run at t=0. The deadlines for the 3 tasks are therefore t=4 for P1, t=8 for P2 and t=12 for P3.**
- **For the moment we will consider co-operative multitasking.**
  - Although we will only show co-operative forms for fixed and round-robin, they also work for pre-emptive OSes.

Scheduling Algorithms

FIXED PRIORITY

Fixed Priority

- **In fixed priority:**
  - Every task is assigned a unique priority when the task is created (using os_create_task for example).
    - ✓**Standard practice is to have a range of 0 to n, with 0 being the highest priority.**
  - When one task completes, the highest priority READY task is picked to run.

NUS National University of Singapore

School of Computing

Fixed Priority

	Ci	Pi
P1	1	4
P2	2	8
P3	3	12

P1	P2	P3
4	8	12
8	16	24
12	24	36
16	32	48
20	40	60
24	48	72
28	56	84
32	64	96
36	72	108
40	80	120

Time	Process	Deadline	Priority	Comments
0	P1	P1,P2,P3	P1P2P3	
1	P2		P1P2P3	
2	P2		P1P2P3	
3	P3		P1P2P3	
4	P3	P1	P1P2P3	
5	P3		P1P2P3	
6	P1		P1P2P3	
7			P1P2P3	
8	P1	P1, P2	P1P2P3	
9	P2		P1P2P3	
10	P2		P1P2P3	
11			P1P2P3	
12	P1	P1, P3	P1P2P3	
13	P3		P1P2P3	
14	P3		P1P2P3	
15	P3		P1P2P3	
16	P1	P1, P2	P1P2P3	

Fixed Priority

- **Fixed priority depends on the programmer to assign priorities to tasks.**
  - How to assign?
    - ✓**Based on importance of a task?**
    - ✓**Based on periods?**
    - ✓**…??**
  - In addition two different programmers may not agree on the relative importance of tasks.
    - ✓**Can be subjective.**
  - Nonetheless fixed priority is simple to implement and therefore popular.
    - ✓**E.g. uC/OS-II.**

Scheduling Algorithms

ROUND ROBIN PRIORITY

Round Robin Priority

- **In round-robin, the priority of the task that is picked to run is demoted to the lowest priority, while all the remaining tasks are promoted by 1 position each.**
- **The next slide shows an example run of the 3 processes using round-robin.**
  - As before, the deadline is assumed to be equal to the period of the task.
    - ✓**If a task runs every 4 ms, then at 0 ms, the deadline for the task is 4 ms. At 4 ms, the deadline is 8 ms, etc.**
    - ✓**The previous instance must finish running by the end of time 7!**
  - In the example on the next page, notice that P1 misses one deadline.
    - ✓**In a hard real-time system, this is BAD.**

Round-Robin Priority

	Ci	Pi
P1	1	4
P2	2	8
P3	3	12

P1	P2	P3
4	8	12
8	16	24
12	24	36
16	32	48
20	40	60
24	48	72
28	56	84
32	64	96
36	72	108
40	80	120

Time	Process	Deadline	Priority	Comments
0	P1		P2P3P1	
1	P2		P3P1P2	
2	P2			
3	P3		P1P2P3	
4	P3	P1		
5	P3			
6	P1		P2P3P1	
7				
8	P2	P1,P2	P3P1P2	
9	P2			
10	P1		P3P2P1	
11				
12	P3	P1,P3	P2P1P3	
13	P3			
14	P3			
15	P1		P2P3P1	
16	P2	P1,P2	P3P1P2	
17	P2			
18	P1		P3P2P1	
19				
20	P1	P1	P3P2P1	
21				
22				
23				
24	P3	P1,P2,P3	P2P1P3	
25	P3			
26	P3			
27	P2		P1P3P2	
28	P2	P1		P1 misses deadline
29	P1		P3P2P1	
30				

Scheduling Algorithms

RATE MONOTONIC SCHEDULING

Rate Monotonic Scheduling

- **In the previous slide, Task P1 missed its deadline because of it's tight period, and as a result, tight deadline.**
  - Notice that the t=32 deadline was skipped completely.
- **One way around this problem is to prioritize tasks according to their periods P_i. This results in the schedule shown in the next slide.**
  - P1 has the smallest period of 4, and has the highest priority. P2 is next with 8, P3 is last with a period of 12.

Rate Monotonic Scheduling

	Ci	Pi
P1	1	4
P2	2	8
P3	3	12

P1	P2	P3
4	8	12
8	16	24
12	24	36
16	32	48
20	40	60
24	48	72
28	56	84
32	64	96
36	72	108
40	80	120

Time	Process	Priority	Deadlines
0	P1	P1P2P3	
1	P2	P1P2P3	
2	P2	P1P2P3	
3	P3	P1P2P3	
4	P3	P1P2P3	P1
5	P3	P1P2P3	
6	P1	P1P2P3	
7		P1P2P3	
8	P1	P1P2P3	P1,P2
9	P2	P1P2P3	
10	P2	P1P2P3	
11		P1P2P3	
12	P1	P1P2P3	P1, P3
13	P3	P1P2P3	
14	P3	P1P2P3	
15	P3	P1P2P3	
16	P1	P1P2P3	P1,P2
17	P2	P1P2P3	
18	P2	P1P2P3	
19		P1P2P3	
20	P1	P1P2P3	P1
21		P1P2P3	
22		P1P2P3	
23		P1P2P3	
24	P1	P1P2P3	P1,P2,P3
25	P2	P1P2P3	
26	P2	P1P2P3	
27	P3	P1P2P3	
28	P3	P1P2P3	P1
29	P3	P1P2P3	
30	P1	P1P2P3	

Pre-Emptive Scheduling

- **In pre-emptive scheduling, a task higher priority task that becomes due to run can pre-empt a currently running task.**
  - In our previous example, when P1 became ready at time 4, it can pre-empt P3 and P2.

NUS National University of Singapore
School of Computing

Rate Monotonic Scheduling Pre-Emptive Case

Time	Process	Priority	Deadline
0	P1	P1P2P3	P1P2P3
1	P2	P1P2P3	
2	P2	P1P2P3	
3	P3	P1P2P3	
4	P1	P1P2P3	P1
5	P3	P1P2P3	
6	P3	P1P2P3	
7		P1P2P3	
8	P1	P1P2P3	P1P2
9	P2	P1P2P3	
10	P2	P1P2P3	
11		P1P2P3	
12	P1	P1P2P3	P1P3
13	P3	P1P2P3	
14	P3	P1P2P3	
15	P3	P1P2P3	
16	P1	P1P2P3	P1P2
17	P2	P1P2P3	
18	P2	P1P2P3	
19		P1P2P3	
20	P1	P1P2P3	P1
21		P1P2P3	
22		P1P2P3	
23		P1P2P3	
24	P1	P1P2P3	P1P2P3
25	P2	P1P2P3	
26	P3	P1P2P3	
27	P3	P1P2P3	
28	P1	P1P2P3	P1
29	P3	P1P2P3	
30		P1P2P3	
31		P1P2P3	
32	P1	P1P2P3	P1P2
33	P2	P1P2P3	
34	P2	P1P2P3	
35		P1P2P3	
36	P1	P1P2P3	P1P3

Rate Monotonic Scheduling Pre-emptive Case

- **In the previous slide we see RMS used in a pre-emptive OS.**
  - All the tasks meet their deadlines.
    - ✓**Small-period tasks like P1 pre-empt tasks with longer periods, thus meeting their own deadlines.**
- **However, tasks are not always guaranteed to meet their deadlines in RMS.**
  - If all tasks can meet their deadlines, the tasks are said to be "schedulable".

Pre-Emptive RMS Unschedulable Example

- **Consider 3 tasks with the following characteristics:**

	Ci	Pi
P1	1	3
P2	2	6
P3	3	8

NUS National University of Singapore

School of Computing

Unschedulable Pre-emptive RMS

Time	Process	Priority	Deadline
0	P1	P1P2P3	P1P2P3
1	P2	P1P2P3	
2	P2	P1P2P3	
3	P1	P1P2P3	P1
4	P3	P1P2P3	
5	P3	P1P2P3	
6	P1	P1P2P3	P1P2
7	P2	P1P2P3	
8	P2	P1P2P3	P3
9	P1	P1P2P3	P1
10	P3*	P1P2P3	
11	P3	P1P2P3	
12	P1	P1P2P3	P1P2
13	P2	P1P2P3	
14	P2	P1P2P3	
15	P1	P1P2P3	P1
16	P3*	P1P2P3	P3
17	P3*	P1P2P3	
18	P1	P1P2P3	P1P2
19	P2	P1P2P3	
20	P2	P1P2P3	
21	P1	P1P2P3	P1
22	P3	P1P2P3	
23	P3	P1P2P3	
24	P1	P1P2P3	P1P2P3
25	P2	P1P2P3	
26	P2	P1P2P3	
27	P1	P1P2P3	P1
28	P3*	P1P2P3	
29		P1P2P3	
30	P1	P1P2P3	P1P2
31	P2	P1P2P3	
32	P2	P1P2P3	P3
33	P1	P1P2P3	P1
34	P3	P1P2P3	
35	P3	P1P2P3	
36	P1	P1P2P3	P1P2

The runs marked with * (e.g. P3*) are runs that have already exceeded their deadlines.

Unschedulable Pre-Emptive RMS

- **In the previous example, P3 keeps missing its deadline.**
  - This is because P3 keeps getting pre-empted by P1 and P2.
  - Eventually P3 gets delayed so much that it misses the deadline.

NUS National University of Singapore
School of Computing

Liu and Layland Criteria for Schedulability of RMS

- **Liu and Leyland (1973) states that for a set of *n* tasks, the tasks are schedulable provided that:**

$$\sum_{i=1}^{n} \frac{C_i}{P_i} \leq n(2^{\frac{1}{n}} - 1)$$

Liu and Leyland Criteria for RMS

- **The left-hand-side of L&L is the CPU utilization.**
- **The right hand side is a threshold.**
- **Basically L&L states that if the CPU utilization by the tasks does not exceed a threshold, the tasks are guaranteed to be schedulable.**
  - Some thresholds are shown on the following slide.

L&L Thresholds

- **This table shows the L&L thresholds for various values of *n*.**
  - Notice that *n* converges towards 69.3% for large *n*.
    - ✓**Thus, in general, as long as the CPU is used less than 69.3% of the time, a schedule is guaranteed.**

n	Threshold
1	100%
2	82.84%
3	77.79%
4	75.68%
100	69.55%
Large	69.3%

L&L Threshold

- **Analysing our two problems:**
  - L&L Threshold for 3 tasks = 77.79%

	Ci	Pi
P1	1	4
P2	2	8
P3	3	12

$$\sum_{i=1}^{n} \frac{C_i}{P_i} = \frac{1}{4} + \frac{2}{8} + \frac{3}{12} = 75\%$$

	Ci	Pi
P1	1	3
P2	2	6
P3	3	8

$$\sum_{i=1}^{n} \frac{C_i}{P_i} = \frac{1}{3} + \frac{2}{6} + \frac{3}{8} = 104.2\%$$

NUS National University of Singapore
School of Computing

Limits of the L&L Criteria

- **The L&L criteria can only tell you if a set of tasks is guaranteed to be schedulable.**
  - As long as the criteria is met, all tasks will meet their deadlines in a pre-emptive system.
- **However, tasks that don't meet the criteria are not necessarily unschedulable!**
  - Relatively easy to construct a counter-example.

A Stronger Schedulability Test Critical Instant Analysis

- **Leedham and Seow give a better test in pages 168-173.**
- **To use this test, assume:**
  - $T = \{T_1, T_2, T_3, \ldots\}$ is the set of tasks. In the previous slides we used P1, P2, etc. to denote tasks. T_i denote the same tasks, but sorted by periods. T_1 is the task with the shortest period, T_2 has the next shortest, etc.
  - C_i is the running time of T_i assuming it is not pre-empted.
  - P_i is the period of task T_i.

Critical Instant Analysis

1. Sort T by period of each task, if T is not already sorted. We will assume that T_1 has the shortest period, T_2 has the 2nd shortest, etc.
2. For each task $T_i \in T$, recursively compute S_{i0}, S_{i1}, ... where:
   a. $S_{i,0} = \sum_{j=1}^{i} C_j$
   b. $S_{i,(x+1)} = C_i + \sum_{j=1}^{i-1} C_j \times \left\lceil \frac{S_{i,x}}{P_j} \right\rceil$

   Stop when $S_{i,(x+1)} = S_{i,x}$. Call this $S_{i,(x+1)}$ the final value $S_{i,F}$. I.e. let $S_{i,F} = S_{i,(x+1)}$ when the iteration terminates.

3. If $S_{i,F} < D_i$, then the task *i* is schedulable and will not miss its deadlines.

NUS National University of Singapore
School of Computing

Example

- **We will apply the CIA to our two examples to test their schedulabilities.**
- **Our tasks are already sorted by P_i, so for our example T_1 = P1, T_2 = P2, T_3 = P3.**

	Ci	Pi
P1	1	4
P2	2	8
P3	3	12

	Ci	Pi
P1	1	3
P2	2	6
P3	3	8

First Example:

$$S_{1,0} = C_1 = 1$$

$$S_{1,1} = C_1 + \sum_{j=1}^{0} C_j \left\lceil \frac{S_{1,0}}{P_j} \right\rceil = C_1 = 1$$

Since $(S_{1,1} = 1) < (P_1 = 4)$, task T_1 is schedulable.

$$S_{2,0} = C_1 + C_2 = 1 + 2 = 3$$

$$S_{2,1} = C_2 + \sum_{j=1}^{1} C_j \left\lceil \frac{S_{2,0}}{P_j} \right\rceil = C_2 + C_1 \times \left\lceil \frac{S_{2,0}}{P_1} \right\rceil = 2 + 1 \times \left\lceil \frac{3}{4} \right\rceil = 3$$

$$S_{2,2} = C_2 + \sum_{j=1}^{1} C_j \left\lceil \frac{S_{2,1}}{P_j} \right\rceil = C_2 + C_1 \times \left\lceil \frac{S_{2,1}}{P_1} \right\rceil = 2 + 1 \times \left\lceil \frac{3}{4} \right\rceil = 3$$

Since $(S_{2,2} = 3) < (P_2 = 8)$, task T_2 is schedulable.

$$S_{3,0} = C_1 + C_2 + C_3 = 1 + 2 + 3 = 6$$

$$S_{3,1} = C_3 + \sum_{j=1}^{2} C_j \left\lceil \frac{S_{3,0}}{P_j} \right\rceil = C_3 + C_1 \times \left\lceil \frac{S_{3,0}}{P_1} \right\rceil + C_2 \times \left\lceil \frac{S_{3,0}}{P_2} \right\rceil = 3 + 1 \times \left\lceil \frac{6}{4} \right\rceil + 2 \times \left\lceil \frac{6}{8} \right\rceil = 7$$

$$S_{3,2} = C_3 + \sum_{j=1}^{2} C_j \left\lceil \frac{S_{3,1}}{P_j} \right\rceil = C_3 + C_1 \times \left\lceil \frac{S_{3,1}}{P_1} \right\rceil + C_2 \times \left\lceil \frac{S_{3,1}}{P_2} \right\rceil = 3 + 1 \times \left\lceil \frac{7}{4} \right\rceil + 2 \times \left\lceil \frac{7}{8} \right\rceil = 7$$

Since $(S_{3,2} = 7) < (P_3 = 12)$, T_3 is schedulable. Thus T is schedulable.

Second Example:

$$S_{1,0} = C_1 = 1$$

$$S_{1,1} = C_1 + \sum_{j=1}^{0} C_j \left\lceil \frac{S_{1,0}}{P_j} \right\rceil = C_1 = 1$$

Since $(S_{1,1} = 1) < (P_1 = 3)$, task T_1 is schedulable.

$$S_{2,0} = C_1 + C_2 = 1 + 2 = 3$$

$$S_{2,1} = C_2 + \sum_{j=1}^{1} C_j \left\lceil \frac{S_{2,0}}{P_j} \right\rceil = C_2 + C_1 \times \left\lceil \frac{S_{2,0}}{P_1} \right\rceil = 2 + 1 \times \left\lceil \frac{3}{3} \right\rceil = 3$$

$$S_{2,2} = C_2 + \sum_{j=1}^{1} C_j \left\lceil \frac{S_{2,1}}{P_j} \right\rceil = C_2 + C_1 \times \left\lceil \frac{S_{2,1}}{P_1} \right\rceil = 2 + 1 \times \left\lceil \frac{3}{3} \right\rceil = 3$$

Since $(S_{2,2} = 3) < (P_2 = 6)$, task T_2 is schedulable.

$$S_{3,0} = C_1 + C_2 + C_3 = 1 + 2 + 3 = 6$$

$$S_{3,1} = C_3 + \sum_{j=1}^{2} C_j \left\lceil \frac{S_{3,0}}{P_j} \right\rceil = C_3 + C_1 \times \left\lceil \frac{S_{3,0}}{P_1} \right\rceil + C_2 \times \left\lceil \frac{S_{3,0}}{P_2} \right\rceil = 3 + 1 \times \left\lceil \frac{6}{3} \right\rceil + 2 \times \left\lceil \frac{6}{6} \right\rceil = 7$$

$$S_{3,2} = C_3 + \sum_{j=1}^{2} C_j \left\lceil \frac{S_{3,1}}{P_j} \right\rceil = C_3 + C_1 \times \left\lceil \frac{S_{3,1}}{P_1} \right\rceil + C_2 \times \left\lceil \frac{S_{3,1}}{P_2} \right\rceil = 3 + 1 \times \left\lceil \frac{7}{3} \right\rceil + 2 \times \left\lceil \frac{7}{6} \right\rceil = 9$$

$$S_{3,3} = C_3 + \sum_{j=1}^{2} C_j \left\lceil \frac{S_{3,2}}{P_j} \right\rceil = C_3 + C_1 \times \left\lceil \frac{S_{3,2}}{P_1} \right\rceil + C_2 \times \left\lceil \frac{S_{3,2}}{P_2} \right\rceil = 3 + 1 \times \left\lceil \frac{9}{3} \right\rceil + 2 \times \left\lceil \frac{9}{6} \right\rceil = 10$$

$$S_{3,4} = C_3 + \sum_{j=1}^{2} C_j \left\lceil \frac{S_{3,3}}{P_j} \right\rceil = C_3 + C_1 \times \left\lceil \frac{S_{3,3}}{P_1} \right\rceil + C_2 \times \left\lceil \frac{S_{3,3}}{P_2} \right\rceil = 3 + 1 \times \left\lceil \frac{10}{3} \right\rceil + 2 \times \left\lceil \frac{10}{6} \right\rceil = 11$$

$$S_{3,5} = C_3 + \sum_{j=1}^{2} C_j \left\lceil \frac{S_{3,4}}{P_j} \right\rceil = C_3 + C_1 \times \left\lceil \frac{S_{3,4}}{P_1} \right\rceil + C_2 \times \left\lceil \frac{S_{3,4}}{P_2} \right\rceil = 3 + 1 \times \left\lceil \frac{11}{3} \right\rceil + 2 \times \left\lceil \frac{11}{6} \right\rceil = 11$$

Since $(S_{3,5} = 11) > (P_3 = 8)$, T_3 is unschedulable. Thus T is unschedulable.

- **Actually, we could've stopped at $S_{3,2}$, which was already larger than 8.**

Scheduling Algorithms

EARLIEST DEADLINE FIRST SCHEDULING

Earliest Deadline First Scheduling

- **In this scheduling algorithm, tasks are prioritized according to whose deadline is the closest.**
  - Unlike RMS, the priorities can and do change dynamically.
  - This is because a process may be delayed or pre-empted by currently higher priority processes, and as a result end closer to their deadlines.

Earliest Deadline First Scheduling

	Ci	Pi
P1	1	4
P2	2	6
P3	3	8

Time	Process	Deadline	Oustanding
0	P1	P1P2P3	P2P3
1	P2		P3
2	P2		P3
3	P3		-
4	P3	P1	P1
5	P3		P1
6	P1	P2	P2
7	P2		
8	P2	P1P3	P1P3
9	P1		P3
10	P3		-
11	P3		-
12	P3	P1P2	P1P2
13	P1		P2
14	P2		-
15	P2		-
16	P1	P1P3	P3
17	P3		-
18	P3	P2	P2
19	P3		P2
20	P1	P1	P2
21	P2		-
22	P2		-
23			
24	P1	P1P2P3	P2P3
25	P2		
26	P2		
27	P3		
28	P3	P1	P1
29	P3		P1
30	P1	P2	P2
31	P2		-
32	P2	P1P3	P1P3
33	P1		P3
34	P3		-
35	P3		-
36	P3	P1P2	P1P2

EDF Scheduling

- **The example in the previous slide is not schedulable in RMS.**
  - Easily shown by applying CIA.
    - ✓**$S_{1,1} = 1$**
    - ✓**$S_{2,2} = 3$**
    - ✓**$S_{3,5} = 11$, larger than P_3=8.**
- **We are however able to find a schedule for this example.**
  - CPU Utilization = 1/4 + 2/6 + 3/8 = 96%

EDF Scheduling

- **However, just as in the case of RMS, EDF scheduling does not always guarantee that tasks meet their deadlines.**
  - E.g. easy to show that for this set of tasks, the tasks will miss their deadlines.
    - ✓**This is because the CPU Utilization is 1/2 + 2/3 + 3/6 = 1.67**

      Thus the tasks occupy >> 100% of the CPU time!

	Ci	Pi
P1	1	2
P2	2	3
P3	3	6

Unschedulable EDF Tasks

Time	Process	Deadline	Oustanding
0	P1	P1P2P3	P2P3
1	P2		P3
2	P2	P1	P1P3
3	P1	P2	P2P3
4	P3	P1	P1P2
5	P3		P1P2
6	P3	P1P2P3	P1*P1P2*P2P3

Scheduling Algorithms

APERIODIC TASKS

Aperiodic Tasks

- **Not all tasks occur at fixed periods.**
  - E.g. a task to read a key from the keyboard will run only when someone presses a key.
- **We need to make an aperiodic task become periodic in order to apply our periodicity analysis.**

Aperiodic Tasks

- **Two Approaches, both based on fixing the aperiodic tasks to a period.**
  - Conservative
    - ✓**Assume the worst case that the sporadic tasks occur at their maximum rate, i.e., their shortest periods.**
  - Optimistic
    - ✓**Assume that sporadic tasks occur at their average rates.**
    - ✓**This can cause some tasks to miss their deadline, even if the analysis guarantees schedulability.**

Transient Overload

Aperiodic Task

- **Which approach should we use?**
  - Approach 1: We take worse case assumption that tasks run at their maximum rates.
    - ✓**Guarantees no missed deadlines.**
    - ✓**Good for hard real-time system**
    - ✓**BUT may result in significant CPU underutilization.**
  - Approach 2: Take average rates.
    - ✓**Better CPU utilization, BUT**
    - ✓**If task activates at a higher than average rate, some tasks will miss their deadlines.**

      System is temporarily overloaded.

NUS National University of Singapore
School of Computing

Summary

- **In this lecture we covered:**
  - Round-robin scheduling.
  - Rate Monotonic Scheduling.
  - Simple schedulability test for RMS.
  - Critical Instant Analysis for RMS and EDF.
  - Earliest Deadline First scheduling.
  - Aperiodic Analysis.