p 3 q 7
SMH

Operating Systems Functions (Part IB): Tripos Questions 1999/2000

Paper 3 Question ~~8~~ 7 (20 Marks)

Why are the scheduling algorithms used in general purpose operating systems such as Unix and Windows NT not suitable for real-time systems? [4 marks]

Rate monotonic (RM) and earliest deadling first (EDF) are two popular scheduling algorithms for real-time systems. Describe these algorithms, illustrating your answer by showing how each of them would schedule the following task set.

Task	*Requires Exactly*	*Every*
$\mathcal{A}$	2ms	10ms
$\mathcal{B}$	1ms	4ms
$\mathcal{C}$	1ms	5ms

You may assume that context switches are instantaneous. [8 marks]

Exhibit a task set which is schedulable under EDF but not under RM. You should demonstrate that this is the case, and explain why.
*H*int: consider the relationship between task periods. [8 marks]

Answers to Paper 3 Question ~~8~~ 7

Why are "standard" scheduling algorithms inappropriate?

In real-time systems we have stringent requirements on when a task executes (or more precisley, when it completes). General purpose scheduling algorithms do not give us the predictability we require for this. This is mainly because in general priority specifies *what* (or *who*) to schedule but not *when* or for *how long*; i.e. CPU allocation for thread t_i, priority p_i depends on the set of all other threads at t_j s.t. $p_j \geq p_i$.

If, as in both Unix and NT, we also have dynamic priority adjustment, the problem becomes even more difficult since the priorities of every thread vary over relatively short time-scales.

Scheduling Tasks

Four marks each for RM and EDF.

In RM, priorities are assigned inversely proportional to task periods; that is, the highest priority is assigned to the task with the smallest period. At any point in time, the highest priority runnable process is scheduled. In EDF scheduling, the task chosen to be scheduled at any instant is simply that with the earliest deadline.

For the given task set, the following schedule with ensue in both the RM and EDF cases:

Time(ms)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$\mathcal{A}$	-	-	A	A	-	-	-	-	-	|	-	A	-	A	-	-	-	-	-	|
$\mathcal{B}$	B	-	-	|	B	-	-	|	B	-	-	|	B	-	-	|	B	-	-	|
$\mathcal{C}$	-	C	-	-	|	C	-	-	-	|	C	-	-	-	|	C	-	-	-	|

The same schedule will repeat every 20ms — in general both RM & EDF repeat every $\text{lcm}(p_i)$ time units, where p_i are the periods of the tasks t_i in the task set. The answer should reflect this.

Unschedulable Task Set

4 marks for the reasoning; 4 marks for an example task set.

This question is not that difficult once one reaslises that the reason RM cannot schedule some task sets is because it can be "overeager" — choosing to schedule a high frequency (and hence high priority) task even when another task is about to miss its deadline. This requires (at least a pair of) relatively prime task periods, hence the hint.

We know that (a) RM can schedule anything when the total utilization is $\leq \ln 2$ and (b) EDF can schedule anything when the total utilization is ≤ 1 [assuming zero context switch times in both cases]. Hence we need to find a task set with utilization between these two bounds which RM cannot schedule.

We also know that RM can schedule anything that EDF can when the periods are harmonic. Hence we want a task set with relatively prime periods and utilisation $\geq \ln 2$, and which we can show that RM cannot schedule.

There are any number of these; for example:

Task	*Requires Exactly*	*Every*
$\mathcal{A}$	2ms	5ms
$\mathcal{B}$	3ms	8ms
$\mathcal{C}$	2ms	9ms

This will be scheduled by RM as follows:

Time(ms)	1	2	3	4	5	6	7	8	9	10
$\mathcal{A}$	A	A	-	-	|	A	A	-	-	|
$\mathcal{B}$	-	-	B	B	B	-	-	|	B	-
$\mathcal{C}$	-	-	-	-	-	-	-	C	|	-

At 9ms, task B is scheduled ahead of task C, and hence task C misses its deadline. EDF would schedule C here ahead of B since although B is more frequent, C is closer to its deadline and B can afford to wait.