Embedded systems engineering
Distributed real-time systems

David Kendall

Introduction

- The task model and assumptions
- Fixed priority scheduling
- Rate-monotonic scheduling
- Simple schedulability tests
- Utilisation bound theorem
- Response time analysis

Task model

We have looked at *static*, *non-preemptive* scheduling of *periodic* tasks

Now we move on to *dynamic*, *preemptive* scheduling of *fixed-priority* periodic tasks

A reminder of the task model. Each task has

- a period (T)
- an offset (ϕ)
- a worst case execution time (C), and
- a deadline (D)

Assumptions (for now)

- An application consists of a fixed set of *n* tasks
- All tasks are periodic, periods are known
- All offsets are 0
- Worst-case execution times of all tasks are known
- Task deadlines are equal to task periods
- Tasks are independent (no precedence constraints, shared resources, etc.)
- Overheads, such as context switch times, are assumed to be 0

Fixed Priority Scheduling (FPS)

- Each task has a static priority
- Priorities are assigned before runtime
- Priorities of ready tasks determine the execution order
- Priorities are derived from temporal requirements

Rate-Monotonic Scheduling (RMS)

Fixed priority scheduling, preemptive

Rate-montonic priority assignment

- The shorter the period (same as the higher the rate) of a task, the higher its priority, P_i
- Rate (frequency) of task i is $\frac{1}{T_i}$, e.g. a task with a period of $10ms$ has a rate of $\frac{1}{0.010} = 100$ Hz
- For all tasks i, j : $P_i > P_j \Leftrightarrow T_i < T_j$

The ready task with the highest priority is selected for execution

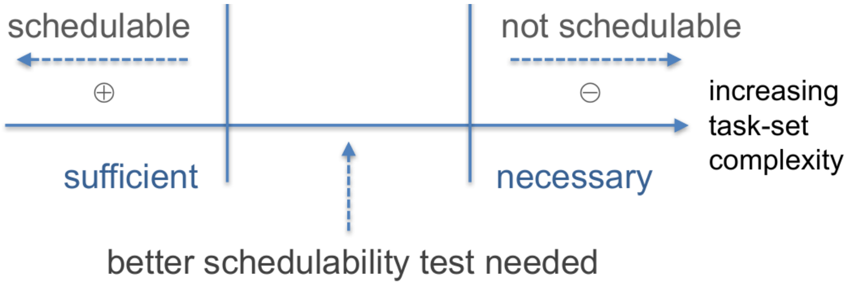
Rate-monotonic priority assignment is *optimal* for FPS

- If a task set can be scheduled with a fixed priority preemptive scheduler, it can be scheduled using RMS

Schedulability Tests

If a sufficient schedulability test is positive, the tasks are definitely schedulable

If a necessary schedulability test is negative, the tasks are definitely not schedulable


schedulable
⊕
sufficient
not schedulable
⊖
necessary
increasing task-set complexity
better schedulability test needed

[Peter Puschner, TU Wien]

Utilisation-Based Schedulability Test

Utilisation, U, for a task set of size n, is defined by

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i}$$

Necessary schedulability test for RMS

$$U \leq 1$$

If test negative, definitely not schedulable

Sufficient schedulability test for RMS

$$U \leq n(2^{1/n} - 1)$$

If test positive, definitely schedulable

If $n(2^{1/n} - 1) \leq U \leq 1$, then we can't tell from these schedulability tests whether the task set is schedulable or not

Utilisation bound theorem

Liu and Layland, *Scheduling algorithms for multiprogramming in a hard real-time environment*, J.ACM 20(1), 1973

Utilisation bound theorem

For a set of *n* tasks, with relative deadlines equal to periods and priorities assigned in RM order, the least upper bound to processor utilisation is $U_{lub} = n(2^{1/n} - 1)$

- U_{lub} is a decreasing function of n
- For large n, $U_{lub} \approx 0.69$

n	$\mathbf{U}_{lub}$	**n**	$\mathbf{U}_{lub}$
2	0.828	6	0.734
3	0.779	7	0.728
4	0.756	8	0.724
5	0.743	9	0.720

Response Time, Critical Instant

The response time of a task instance is the length of time between the arrival of the instance and the completion of its execution

The *worst case response time (WCRT)* of a task is the maximum response time of all its instances

A *critical instant* for a task is defined to be an instant at which the arrival of the task at that instant will give rise to the worst case response time for the task.

According to (Liu and Layland, 1973), the *critical instant* for a task in a FPS task set *occurs whenever the task arrives at the same time as all higher priority tasks*

Response Time Analysis

Response time analysis gives a necessary and sufficient (exact) test for the schedulability of any task set

The test has two parts:

1. Calculate the WCRT, R_i, of each task in the task set
2. If $R_i \leq T_i = D_i$ for all tasks i, then the task set is schedulable, otherwise it is not schedulable

The WCRT of the highest priority task is equal to its computation time. Other processes will suffer *interference* from higher priority tasks, i.e. although ready to run, the task will have to wait while one or more higher priority tasks execute

So, in general, the response time of task i is given by

$$R_i = C_i + I_i$$

where I_i is the maximum interference that task i can suffer in any interval $[t, t + R_i)$

Response Time Analysis (ctd)

Assume all tasks arrive at time 0 (a critical instant for all tasks)

For tasks *i* and *j*, with $P_j > P_i$, task *j* will arrive a number of times (at least once) within the interval $[0, R_i)$. The number of arrival times is given by the expression

$$\left\lceil \frac{R_i}{T_j} \right\rceil$$

where $\lceil \cdot \rceil$ is the *ceiling* function, which returns the smallest integer greater than the fraction inside the ceiling brackets

Each arrival of task *j* will impose an interference of C_j, so the maximum interference suffered by task *i* due to task *j* is

$$\left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

Response Time Analysis (ctd)

Each task of higher priority interferes with task *i*, so the maximum interference, I_i, suffered by task *i* is

$$I_i = \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

Substituting this back into the equation for R_i gives

$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \tag{1}$$

Problem: Although this equation gives an accurate representation of R_i, it is difficult to solve, since R_i appears on both sides and on one side appears within an expression involving the ceiling function

Response Time Analysis (ctd)

The simplest way to solve the equation is to form a *recurrence relation*

$$w_i^{n+1} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^n}{T_j} \right\rceil C_j \qquad (2)$$

This allows us to solve the equation *iteratively*

For each task i, set w_i^0 to C_i and then use the relation (2) to compute $w_i^1, w_i^2, \ldots$

When the iteration converges, i.e. when $w_i^k = w_i^{k+1}$, for some k, a solution to the equation has been found, namely w_i^k, and gives the value of R_i

It is possible that the iteration may not converge, in which case it can be terminated when w_i^k is greater than T_i, where we can see that the task set is not schedulable

Example of response time analysis

Determine whether or not the following task set is schedulable

Task	Period, T	WCET, C	Priority, P
a	7	3	3
b	12	3	2
c	20	5	1