Response Time Analysis & Dynamic Pre-emptive Scheduling

KF6010 – Distributed Real Time Systems

Dr Alun Moon
alun.moon@northumbria.ac.uk

Lecture 5

Task Model

Each task has

- **period** T how often the task runs
- **offset** ϕ the initial delay, phase-difference between two tasks of same period
- **WCET** C Worst-Case-Execution-Time of the task
- **deadline** D the time by which the task must complete

Some Assumptions

for now

1. An application consists of a fixed set of N tasks
2. All tasks are periodic and the periods are known
3. All offsets are 0
4. Worst-case execution times are known
5. Task deadlines are equal to task periods
6. Tasks are independent, no precedence constraints, no shared resources, etc.
7. Overheads, such as context switch times, are assumed to be 0

Fixed Priority Scheduling — FPS

- Each task has a static (unchanging) priority
- Priorities are assigned before runtime (cf. TTS)
- Priorities of ready tasks determine the execution order
- Priorities are derived from temporal resuirements

Rate Monotonic Scheduling — RMS

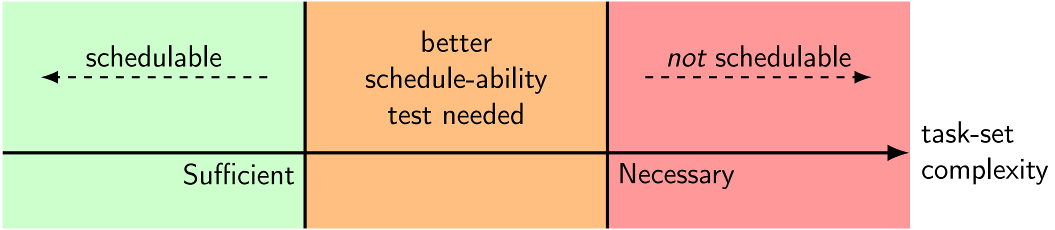
- Fixed priority scheduling
- Pre-emptive
- Rate-monotonic priority assignment

1. The shorter the period (the higher the rate) of a task, the higher its priority P_i
2. Rate (frequency) of task is
$$f_i = \frac{1}{T_i}$$
e.g. a task with a period of 10 ms has a rate of 100 Hz
3. $\forall i, j : P_i > P_j \Leftrightarrow T_i < T_j$

- The ready task with the highest priority is selected for execution
- Rate-monotonic priority assignment is *optimal* for FPS
  If a task can be scheduled with a fixed priority pre-emptive scheduler,
  it can be scheduled using RMS

Schedule-ability Tasks

- If a sufficient schedule-ability test is positive, the tasks are definitely schedule-able
- If a necessary schedule-ability test is negative, the tasks are definitely not-schedule-able



Utilisation-Based Schedule-ability Test

Utilisation U for a task set of size N is defiled by

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

Necessary schedule-ability test for RMS

$$U \leq 1$$

if test negative $U > 1$ definitely not-schedule-able

Sufficient schedule-ability test for RMS

$$U \leq N(2^{\frac{1}{N}} - 1)$$

if test positive (true), definitely schedule-able

If $N(2^{\frac{1}{N}} - 1) \leq U \leq 1$ then we cannot tell from these tests whether the task-set is schedule-able

Utilisation bound theorem

Liu and Layland proposed the *Utilisation Bound theorem*

Utilisation Bound Theorem

For a set of N tasks, with relative deadlines equal to periods and priorities assigned in rate-monotonic order, the least upper bound to processor utilisation is

$$U_{lub} = N(s^{\frac{1}{N}} - 1)$$

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

N	U_{lub}	N	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

Chung Laung Liu and James W Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)*, 20(1):46–61, 1973.

Response Time and Critical Instant

- The response time of a task instance is the length of time between the arrival of the instance and the completion of the execution.
- The *worst-case-response-time* WCRT of a task is the maximum response over 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, the *critical-instant* for a task in a fixed-priority-schedule task set occurs whenever the task arrives at the same time as all the higher-priority tasks.

Response Time Analysis I

- Response time analysis gives a necessary and sufficient (exact) test for the schedule-ability 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 schedule-able, otherwise it is not!

- The WCRT of the highest priority task is equal to its computation time.
- Other processes will suffer *interference* from higher-priority tasks. *ie.* 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 a 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 II

- 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

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

- 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

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

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

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

Response Time Analysis III

- Substituting this back into the equation for R_i gives

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

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

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

  This allows us to solve the equation *iteratively*

Response Time Analysis IV

- For each task i:
  1. set $r_i^0 = C_i$
  2. use the recurrence relation to compute $r_i^1, r_i^2, \ldots$
  3. when the sequence *converges* (*ie* when $r_i^k = r_i^{k+1}$ for some k) the solution has been found
  4. $R_i = r_i^k$
- It is possible that the iteration may not converge, in which case it can be terminated when $w_i^k > T_i$, where the task becomes *un-schedule-able*

Example I

Determine the Response-Time-Analysis, and hence whether the task set is schedule-able for:

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

Task a $R_a = r_a^0 = C_a = 3$

Task b
1. $r_b^0 = C_b = 3$
2. $r_b^1 = C_b + \sum_{j \in \{a\}} \left\lceil \frac{r_b^0}{T_j} \right\rceil C_j = C_b + \left\lceil \frac{r_b^0}{T_a} \right\rceil C_a = 3 + \left\lceil \frac{3}{7} \right\rceil 3 = 3 + 3 = 6$

   $r_b^2 = C_b + \left\lceil \frac{r_b^1}{T_a} \right\rceil C_a = 3 + \left\lceil \frac{6}{7} \right\rceil 3 = 3 + 3 = 6$
3. $R_b = r_b^2 = 6$

Example II

Task c

1. $r_c^0 = C_c = 5$
2. $r_c^1 = C_c + \sum_{j \in \{a,b\}} \left\lceil \frac{r_c^0}{T_j} \right\rceil C_j = C_c + \left\lceil \frac{r_c^0}{T_a} \right\rceil C_a + \left\lceil \frac{r_c^0}{T_b} \right\rceil C_b$
   $r_c^1 = 5 + \left\lceil \frac{5}{7} \right\rceil 3 + \left\lceil \frac{5}{12} \right\rceil 3 = 5 + 3 + 3 = 11$
3. $r_c^2 = C_c + \left\lceil \frac{r_c^1}{T_a} \right\rceil C_a + \left\lceil \frac{r_c^1}{T_b} \right\rceil C_b$
   $= 5 + \left\lceil \frac{11}{7} \right\rceil 3 + \left\lceil \frac{11}{12} \right\rceil 3 = 5 + 2 \times 3 + 3 = 14$
4. $r_c^3 = 5 + \left\lceil \frac{14}{7} \right\rceil 3 + \left\lceil \frac{14}{12} \right\rceil 3 = 5 + 2 \times 3 + 2 \times 3 = 17$
5. $r_c^4 = 5 + \left\lceil \frac{17}{7} \right\rceil 3 + \left\lceil \frac{17}{12} \right\rceil 3 = 5 + 3 \times 3 + 2 \times 3 = 20$
6. $r_c^5 = 5 + \left\lceil \frac{20}{7} \right\rceil 3 + \left\lceil \frac{20}{12} \right\rceil 3 = 5 + 3 \times 3 + 2 \times 3 = 20$
7. $R_c = 20$

Example III

The task set can be annotated with the response times.

Task	Period T	WCET C	Priority P	WCRT R
a	7	3	3	3
b	12	3	2	6
c	20	5	1	20

Schedule-ability

Since we can say

$$\forall p \in \{a, b, c\} : R_p \leq T_p$$

holds.

The Task-set is schedule-able