Dual Priority Scheduling: Is the Processor Utilisation bound 100%?

Alan Burns
Department of Computer Science
University of York, UK

EXTENDED ABSTRACT

Fixed priority (FP) schemes have the disadvantage that processor utilisations less than 100% must be tolerated if a system is to be guaranteed off-line. By comparison earliest deadline first (EDF) scheduling can theoretically utilise all of a processor's capacity. In this paper the dual priority scheme is revisited: here a task may execute in two phases; each phase has a static priority assigned, the transition from one phase to another is made at a fixed offset in time from the release of the task.

In their seminal paper of 1973 Liu and Layland [2] introduced the fixed priority (FP) scheme known as *rate monotonic priority assignment*, and the dynamic priority scheme known as *earliest deadline first* (EDF). Their theoretical treatment produced the well known utilisation (U) bounds for a set of n periodic tasks with period T equal to deadline D; and computation time, C. For FP the bound converges, approximately, on the value 0.69 (69%) for large n. For n equal to 2 it is close to 0.83. For EDF the bound is 1 for all n.

In 1993 a dual priority assignment scheme was introduced [1]. This paper contained the conjecture:

Conjecture 1: For any task set with total utilisation less than or equal to 100% there exists a dual priority assignment that will meet all deadlines.

This remains an open question. Search techniques have failed to find a counter example. Even if one exists then there remains the question as to what is the utilisation bound? is this is a function of the number of tasks (n)? and if not 'dual' priority then 'triple'?, or 'quad'?, or what? At some level (m), 'm-priority' assignment can be made to emulate EDF and hence there does exist a 'static' scheme that can obtain the 100% bound – but is it the dual priority scheme?

The dual priority scheme is a minimal dynamic scheme that allows a task to change (increase) its priority during its execution. Each task has at most two priority levels: many tasks will continue to have only one priority. At run-time only standard preemptive priority based scheduling is required. A standard RTOS (with a priority change primitive – which is a commonly supported feature) could therefore support task sets with the same utilisation bound as EDF. A dual priority scheme could also be used with priority-based non-preemptive communication protocols such as CAN. Here an EDF-based protocol is not possible, but a dual-priority scheme, in which the priority of a message is increased if it has been in a node's output buffer for a predefined interval of time, is certainly feasible with only a minor change to the CAN protocol.

In addition to the usual notation each task has an intermediate deadline S_i at which time it undergoes a step change (increase) in priority. For all tasks: $0 \leq S_i \leq T_i$. Each task has a *phase 1* priority P_i^1 and if $S_i < T_i$ a *phase 2* priority P_i^2 with $P_i^1 < P_i^2$.

The following assumptions are used. Any task, in either phase, can be preempted by any other task running at a higher priority. Tasks do not suspend themselves other than at the end of their computations. The time required to perform context switching, priority changes etc is ignored (i.e. assumed to be zero). A single processor is assumed.

A simple example will illustrate the benefits of this scheme. Table I gives the details of a three task system, each task has $T = D$. Note that priority 1 is high and 4 is low. The total utilisation is 100%, and the LCM of the task periods is 24.

	T	C	P	P^2	S	U
τ_1	6	3	2			50%
τ_2	8	2	3			25%
τ_3	12	3	4	1	11	25%

TABLE I
EXAMPLE TASK SET

If the dual phasing is ignored (i.e. the tasks are treated as having single priorities) then the task set is not schedulable by rate monotonic priority assignment (or any other static priority assignment scheme as rate monotonic is optimal). The lowest priority task (τ_3) can only execute for two ticks before its first deadline. But if τ_3 has its priority raised at tick 11 to above τ_2 then all deadlines are met.

Another task set that requires all tasks to have a priority change is given in Table II.

	T	C	P	P^2	S	U
τ_1	28	21	4	1	9	75%
τ_2	100	15	5	2	84	15%
τ_3	160	16	6	3	130	10%

TABLE II
EXAMPLE TASK SET

The current state of the dual priority conjecture is:

- For $n = 2$ a proof has been obtained (so $U = 1$ rather than $U = 0.83$ for standard FP) – see Appendix.

- No counter example found with extensive searches for $n = 3$.
- No counter example found for $n > 3$, but search is computationally expensive as:
  - Simulation up to the LCM needed as the scheduling test.
  - No formulae exists for computing the migration points (the Ss).
- The phase 1 priorities are probably Rate Monotonic.
- The phase 2 priorities are possible also Rate Monotonic with all phase 2 priorities higher than all phase 1 priorities.
- For some task sets the highest priority tasks may have $S_i = 0$.
- A range of promotion points may all lead to a schedulable system.
- The promotion points are a function of task computation times (ie. not just task periods).

The open question is therefore: is the utilisation bound for the dual priority scheme 1? And if it is, how are the promotion points (Ss) computed? If it is not, what is the bound for the dual priority scheme, and is there a m-priority scheme that does provide the maximum bound – and is m then a function of n?

One possible method of tackling this question is to consider the behaviour of the EDF scheme. Whilst EDF tasks do not have static priorities, EDF jobs do. There is therefore a fixed number of partial orders for job executions. Does the dual priority scheme have a similar number?

REFERENCES

[1] A. Burns and A. J. Wellings. Dual priority assignment: A practical method of increasing processor utilisation. In *Proceedings of the Fifth Euromicro Workshop on Real-Time Systems, IEEE Computer Society Press*, pages 48–53, Oulu, Finland, 1993.

[2] C.L. Liu and J.W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *JACM*, 20(1):46–61, 1973.

APPENDIX - PROOF FOR TWO TASKS

Consider two tasks, τ_1 and τ_2 with $T_1 < T_2$ and hence priority of τ_1 greater then the priority of τ_2. Assume total utilisation is the worst-case value of 1:

$$C_1/T_1 \ + \ C_2/T_2 \ = \ 1 \tag{1}$$

Assume τ_2 has a promotion point at time g before its deadline (ie. $g = T_2 - S_2$). For a two task system τ_1 does not need to be promoted. Consider an arbitrary deadline of τ_2, as $D_2 = T_2$ then this deadline can be represented by the time pT_2 – for some value of p (assuming the system starts its execution at time 0). Finally let l be the shortest interval from a previous release of τ_1 and the point pT_2 – see figure 1.

At point pT_2, τ_2 should should have executed for time pC_2; τ_1 will have executed for a number of complete invocations:

$$\left((pT_2 - l)/T_1\right) C_1$$

plus a partial computation of maximum size: $l - g$. This implies that:

$$\left((pT_2 - l)/T_1\right) C_1 + pC_2 + l - g \ \leq \ pT_2 \tag{2}$$

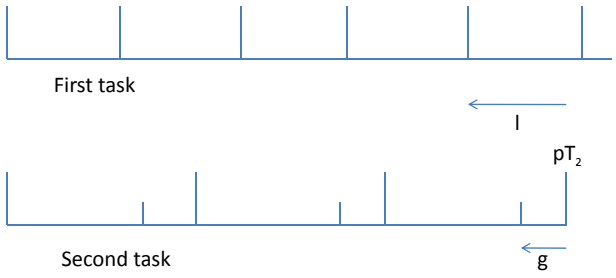

Fig. 1. Two task execution

Substituting for C_1 from Eqn (1) in Eqn (2) produces:

$$(pT_2 - l)(1 - C_2/T_2) + pC_2 + l - g \ \leq \ pT_2$$

Expanding the first terms and canceling a number of balanced terms allow this relation to be simplified to:

$$lC_2/T_2 \leq g$$

Now the value l depends on which deadline of τ_2 is considered, but its maximum value is the maximum distance between a multiple of T_1 a corresponding 'next' release of τ_2. Hence the maximum value of l is equal to $T_1 - H$, where H is the highest common factor of the integers T_1 and T_2. This gives the final bound on g of

$$(T_1 - H)C_2/T_2 \ \leq \ g \tag{3}$$

Note if T_2 is a multiple of T_1 then H is equal to T_1 and g can be zero, ie. no promotion point needed. This result would be expected as task sets with utilisation of 1 are schedulable in this circumstance. Also if T_1 and T_2 are co-primes then Eqn (3) becomes:

$$(T_1 - 1)C_2/T_2 \ \leq \ g$$

The other constraint on g is that τ_1 must remain schedulable at its deadlines. Here the worst case is when two deadlines coincide. Now all of time g can be used by τ_2, and hence

$$g \ \leq \ T_1 - C_1 \tag{4}$$

Now a value of g must exist (and hence the two tasks be schedulable) if Eqns (3) and (4) can both be satisfied for any feasible task set (ie. one satisfying Eqn(1)). If this were *not* the case then

$$T_1 - C_1 \ < \ (T_1 - H)C_2/T_2,$$

which implies

$$T_1 - C_1 \ < \ (T_1)C_2/T_2,$$

giving

$$1 - C_1/T_1 \ < \ C_2/T_2,$$

and hence

$$1 \ < \ C_2/T_2 + C_1/T_1$$

which is clearly false.

This provides the proof that all two task systems have a dual priority scheme that will guarantee schedulability if the utilisation of the tasks is feasible (not more than 1). Moreover, the promotion time is provided by any value of g satisfying Eqns (3) and (4).

As an example consider a simple task set with $T_1 = 8$, $C_1 = 4$, $T_2 = 12$ and $C_2 = 6$; H has the value 4 and so the promotion point for τ_2 must be at least 2 (from its deadline) and no more that 4. A simple simulation shows that values of 2, 3 and 4 will all lead to schedulability.