8. Advanced Topics

UNIVERSIDAD DE CANTABRIA

8.1 Post-period deadlines

8.2 Optimum Priority Assignment

8.3 Handling the jitter effect

8.4 Response Time Analysis for EDF

8.1 Post-Period Deadlines

UNIVERSIDAD DE CANTABRIA

For a set of periodic tasks with some deadlines after the end of the period:

- **The worst-case response time is obtained at the critical instant in which all tasks start at the same time**
- **It is not sufficient to check the first deadline. All deadlines in a busy period must be checked**
- **Nor rate monotonic nor deadline monotonic priorities are optimum**

τ_i-Busy period: An interval of time during which the processor is continuously executing tasks of priority P_i or higher

A modified version of the response test can be used to assess schedulability

Notes:

When tasks have their deadlines after the end of a period, the critical-instant concept still applies. However, it is not sufficient to only check the first deadline, because an earlier job of task τ_i can delay the execution of a later job.

How many deadlines do we need to check to determine schedulability?. We must determine the length of the worst-case busy period which, intuitively, is the interval of time from the activation of task τ_i, until all jobs of τ_i have been completed.

In the next slide we show how to obtain the length of the busy period started at a critical instant, the number of jobs of τ_i in that busy period, and the response times of each of these jobs.

RTA Test for Arbitrary Deadlines

Length of the busy period:

$$L_i = [min(t) > 0 \mid W_i(t) = t]$$

This is calculated with the usual iterative equation, but taking into account that C_i can now appear multiple times:

$$a_{k+1} = W_i(a_k) = \left\lceil \frac{a_k}{T_1} \right\rceil C_1 + \ldots + \left\lceil \frac{a_k}{T_{i-1}} \right\rceil C_{i-1} + \left\lceil \frac{a_k}{T_i} \right\rceil C_i + B_i$$

When the iteration completes we can obtain the number of jobs in the busy period, n_i:

$$n_i = \lceil L_i / T_i \rceil$$

Notes:

The length of the worst-case busy period is obtained using our iterative equation, but now taking into account that the task under analysis, τ_i, can influence the calculation several times. The iteration starts with $a_0=\Sigma C_i$, and finishes when the same value is obtained in two successive iterations. As long as the total utilization is less than 1.0, the iteration is guaranteed to be finite.

Once the length of the busy period is obtained, we can easily determine the number of jobs of task τ_i that belong to that busy period. For each of these jobs we will have to calculate the worst-case response time and compare it to the deadline D_i. We show how to do this in the next slide.

RTA Test for Arbitrary Deadlines

The completion time of job j of τ_i, E_{ij}, is obtained with the following iteration:

$$a_{k+1} = W_i(a_k) = \left\lceil \frac{a_k}{T_1} \right\rceil C_1 + \ldots + \left\lceil \frac{a_k}{T_{i-1}} \right\rceil C_{i-1} + jC_i + B_i$$

The response time of each job R_{ij} must be checked against its deadline:

$$R_{ij} = E_{ij} - (j-1)T_i \leq D_i$$

Notes:

The iterative equation for job *j* of task τ_i takes into account that *j* activations of τ_i must have been completed, besides all the higher priority work initiated since the critical instant.

Each response time is checked against the deadline, but taking into account that the deadline is not relative to the start of the busy period, but to the start of each job.

If all deadlines are met then the task is schedulable, because the worst-case condition corresponds to the conditions of the critical instant.

8.3 Optimum Priority Assignment

UNIVERSIDAD DE CANTABRIA

The following algorithm will find a schedulable solution if there is one:

```
for Ordered in reverse 2..N loop
   Failed:=True;
      for j in reverse 1..Ordered loop
         exchange tasks j and Ordered;
         if task Ordered is schedulable then
            Failed:=False;
            exit;
         else
            exchange tasks j & Ordered back at old priorities;
         end if;
      end loop;
   exit when Failed;
end loop;
```

Notes:

UNIVERSIDAD DE CANTABRIA

As we mentioned before, nor deadline nor rate monotonic priority assignments were optimum when deadlines were after the end of the period. An ordering algorithm was proposed by Audsley "*Optimal Priority Assignment and Feasibility of Static Priority Tasks with Arbitrary Start Times*", Dept. of Computer Science, University of York (December 1991).

This algorithm is optimum in the sense that if it is unable to find a priority ordering where all tasks are schedulable, then no priority ordering exists where all tasks are schedulable.

The algorithm assumes a set of *N* tasks, numbered from 1 to *N*, where the number is the priority assigned (1 is the highest priority, and *N* is the lowest). The initial priority assignment can be an arbitrary guess. A deadline monotonic approach can be used as an initial guess, because the better the initial guess, the less time it takes to execute the algorithm.

The algorithm uses the exact schedulability test for arbitrary priorities. The algorithm works as follows: we first try to find a task that is schedulable at the lowest priority (*N*). We try with all the tasks that are unsorted, starting from task *N*, until task 1. When we find one task that is schedulable at priority *N*, we then try to find a task that is schedulable at the next highest priority, *N*-1; for this case we try with all the unsorted tasks, i.e., from task *N*-1 to task 1. The algorithm continues in the same way until all tasks are ordered. The worst-case execution time of this algorithm is $O(N^2)$.

Example with Arbitrary Deadlines

Task set requirements:

	C	T	D
Task τ_1:	30	100	100
Task τ_2:	80	150	250
Task τ_3:	40	250	400

Total utilization is 99.3%

Solution with Arbitrary Deadlines

UNIVERSIDAD DE CANTABRIA

Task 1. Length of the busy period:

$$L_1 = 1 \cdot C_1 = 30ms$$

Task 2. Length of the busy period:

$$L_2 = 2 \cdot C_1 + 1 \cdot C_2 = 140ms$$

Task 3. Length of the busy period:

$$L_3 = 12 \cdot C_1 + 8 \cdot C_2 + 5 \cdot C_3 = 1200ms$$

For task 3 we need to check five jobs in the busy period.

Solution (cont'd)

Task 3. completion time and response time of each job:

$$E_{31} = 3 \cdot C_1 + 2 \cdot C_2 + 1 \cdot C_3 = 290ms \qquad R_{31} = 290ms$$

$$E_{32} = 6 \cdot C_1 + 4 \cdot C_2 + 2 \cdot C_3 = 580ms \qquad R_{32} = 330ms$$

$$E_{33} = 9 \cdot C_1 + 6 \cdot C_2 + 3 \cdot C_3 = 870ms \qquad R_{32} = 370ms$$

$$E_{34} = 11 \cdot C_1 + 7 \cdot C_2 + 4 \cdot C_3 = 1050ms \qquad R_{34} = 300ms$$

$$E_{35} = 12 \cdot C_1 + 8 \cdot C_2 + 5 \cdot C_3 = 1200ms \qquad R_{35} = 290ms$$

Solution (cont'd)

UNIVERSIDAD DE CANTABRIA

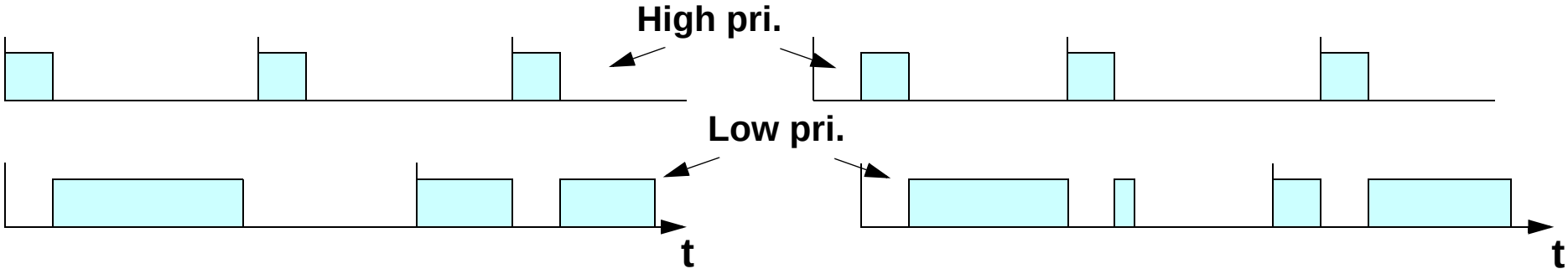
Therefore, the worst-case response time is 370 ms., and the task set is schedulable

8.3 Handling the Release Jitter Effect

Periodic events with release jitter have an arrival time which may be early or late, within a bounded interval:

- events arrive at $t_0 + nT + j, 0 \leq j \leq J$

Release jitter may have a delay effect on lower priority tasks:



Execution sequence for two periodic tasks. Worst case is one preemption

Execution sequence with jitter. Worst case is two preemptions

Notes:

Sometimes, periodic events have a certain degree of jitter. This means that, although the event arrivals are basically periodic, the actual arrival time is indeterminate within a given interval around the start of the period. This effect is also called the deferred execution effect.

When a system has tasks with jitter, some delay effects may occur for that task and for lower priority tasks. The figure above shows an example of such a delay:

- In the execution sequence at the left, there are two periodic tasks, and the worst-case response for the lower priority task is obtained when it is preempted once by the high priority task.
- In the execution sequence at the right, the high priority task shows some jitter. The figure shows the situation in which the first activation of the task is late, while the second activation is early. Under these conditions, the lower priority task is delayed because it may now suffer from two preemptions in the same job, thus increasing its worst-case response time.

The worst-case phasing in a situation with jitter occurs when the first activation occurs the latest time possible, while all the next activations occur the earliest possible. Besides, the lower priority tasks must be phased against the first activation, in the usual way in which we create a critical instant.

Analysis with arbitrary deadlines and jitter (Tindell, 1992):

UNIVERSIDAD DE CANTABRIA

We call the maximum release jitter of task i, J_i

Worst case *response time* of a task: found in a busy period in which all tasks:

- are released at the beginning of the busy period,
- have experienced their maximum jitter on the first job
- and experience the minimum jitter on the following jobs that make them happen inside the busy period

Analysis with arbitrary deadlines and jitter (cont'd):

UNIVERSIDAD DE CANTABRIA

Iterative equation used for analysis of task-*i* in one resource:

$$w_i^{n+1}(p) = pC_i + B_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{J_j + w_i^n(p)}{T_j} \right\rceil C_j$$

This is carried out for p=1,2,3,..., until

$$w_i(p) \leq pT_i$$

The response time is:

$$R_i(p) = w_i(p) - (p-1)T_i + J_i$$

8.4 Response Time Analysis for EDF

UNIVERSIDAD DE CANTABRIA

Worst case *response time* of a task: found in a busy period in which all *other* tasks:

- **are released at the beginning of the busy period,**
- **and have experienced their maximum jitter**

Differences with fixed priorities:

- **The task under analysis does not necessarily start with the busy period**
- **The busy period is longer, because it involves all tasks**

Response Time Analysis for EDF

Worst contribution of task τ_i to the busy period at time t, when the deadline of the analyzed task, τ_a, is D:

$$W_i(t, D) = min\left(\left\lceil \frac{t + J_i}{T_i} \right\rceil, \left\lfloor \frac{J_i + D - d_i}{T_i} \right\rfloor + 1\right)_0 \cdot C_i$$

Worst completion time of activation p, if first activation at A:

$$w_a^A(p) = pC_a + \sum_{\forall i \neq a} W_i(w_a^A(p), D^A(p))$$

Worst response time if first activation is A:

$$R^A(p) = w_a^A(p) - A + J_a - (p-1)T_a$$

Response time analysis in a single resource (cont'd)

UNIVERSIDAD DE CANTABRIA

Set of potential critical instants; L is the longest busy period:

$$\Psi = \cup\{(p-1)T_i - J_i + d_i\} \qquad \forall p = 1\ldots\left\lceil \frac{L - J_a}{T_a} \right\rceil, \forall i \neq a$$

Values of A to check:

$$\Psi^* = \{\Psi_x \in \Psi \,|\, (p-1)T_a - J_a + d_a \leq \Psi_x < pT_a - J_a + d_a\}$$

$$A = \Psi_x - [(p-1)T_a - J_a + d_a]$$

Worst-case response time

$$R_a = max[R^A(p)] \qquad \forall p = 1\ldots\left\lceil \frac{L - J_a}{T_a} \right\rceil, \forall A \in \Psi^*$$