Scheduling of Real-Time processes

Follows: some pages on

”Response time analysis for FPS” by A. Burns and A. Wellings

13.5 Response time analysis for FPS

The utilization-based tests for FPS have two significant drawbacks: they are not exact, and they are not really applicable to a more general process model. This section provides a different form of test. The test is in two stages. First, an analytical approach is used to predict the worst-case response time of each process. These values are then compared, trivially, with the process deadlines. This requires each process to be analyzed individually.

For the highest-priority process, its worst-case response time will equal its own computation time (that is, $R = C$). Other processes will suffer **interference** from higher-priority processes; this is the time spent executing higher-priority processes when a low-priority process is runnable. So for a general process i:

$$R_i = C_i + I_i \tag{13.3}$$

where I_i is the maximum interference that process i can experience in any time interval $[t, t + R_i)$.[1] The maximum interference obviously occurs when all higher-priority processes are released at the same time as process i (that is, at a critical instant). Without loss of generality, it can be assumed that all processes are released at time 0. Consider one process (j) of higher priority than i. Within the interval $[0, R_i)$, it will be released a number of times (at least one). A simple expression for this number of releases is obtained using a ceiling function:

$$Number\_Of\_Releases = \left\lceil \frac{R_i}{T_j} \right\rceil$$

The ceiling function ($\lceil\ \rceil$) gives the smallest integer greater than the fractional number on which it acts. So the ceiling of 1/3 is 1, of 6/5 is 2, and of 6/3 is 2. The definitions of the ceilings of negative values need not be considered.

[1] Note that as a discrete time model is used in this analysis, all time intervals must be closed at the beginning (denoted by '[') and open at the end (denoted by a ')'). Thus a process can complete executing on the same tick as a higher-priority process is released.

So, if R_i is 15 and T_j is 6 then there are 3 releases of process j (at times 0, 6 and 12). Each release of process j will impose an interference of C_j. Hence

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

If $C_j = 2$ then in the interval [0, 15) there are 6 units of interference. Each process of higher priority is interfering with process i, and hence:

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

where $hp(i)$ is the set of higher-priority processes (than i). Substituting this value back into Equation (13.3) gives (Joseph and Pandya, 1986):

$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \qquad \textbf{(13.4)}$$

Although the formulation of the interference equation is exact, the actual amounts of interference is unknown as R_i is unknown (it is the value being calculated). Equation (13.4) has R_i on both sides, but is difficult to solve due to the ceiling functions. It is actually an example of a fixed-point equation. In general, there will be many values of R_i that form solutions to Equation (13.4). The smallest such value of R_i represents the worst-case response time for the process. The simplest way of solving Equation (13.4) is to form a recurrence relationship (Audsley et al., 1993a):

$$w_i^{n+1} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^n}{T_j} \right\rceil C_j \qquad \textbf{(13.5)}$$

The set of values $\{w_i^0, w_i^1, w_i^2, ..., w_i^n, ...\}$ is, clearly, monotonically non-decreasing. When $w_i^n = w_i^{n+1}$, the solution to the equation has been found. If $w_i^0 < R_i$ then w_i^n is the smallest solution and hence is the value required. If the equation does not have a solution then the w values will continue to rise (this will occur for a low-priority process if the full set has a utilization greater than 100%). Once they get bigger than the process's period, T, it can be assumed that the process will not meet its deadline. The above analysis gives rise to the following algorithm for calculation response times:

```
for i in 1..N loop -- for each process in turn
  n := 0
  $w_i^n$ := $C_i$
  loop
    calculate new $w_i^{n-1}$ from Equation (13.5)
    if $w_i^{n+1}$ = $w_i^n$ then
      $R_i$ := $w_i^n$
      exit {value found}
    end if
    if $w_i^{n+1} > T_i$ then
      exit {value not found}
    end if
    n := n + 1
  end loop
end loop
```

By implication, if a response time is found it will be less than T_i, and hence less than D_i, its deadline (remember with the simple process model $D_i = T_i$).

In the above discussion, w_i has been used merely as a mathematical entity for solving a fixed-point equation. It is, however, possible to get an intuition for w_i from the problem domain. Consider the point of release of process i. From that point, until the process completes, the processor will be executing processes with priority P_i or higher. The processor is said to be executing a P_i-**busy period**. Consider w_i to be a time window that is moving down the busy period. At time 0 (the notional release time of process i), all higher priority processes are assumed to have also been released, and hence

$$w_i^1 = C_i + \sum_{j \in hp(i)} C_j$$

This will be the end of the busy period unless some higher-priority process is released a second time. If it is, then the window will need to be pushed out further. This continues with the window expanding and, as a result, more computation time falling into the window. If this continues indefinitely then the busy period is unbounded (that is, there is no solution). However, if at any point, an expanding window does not suffer an extra 'hit' from a higher-priority process then the busy period has been completed, and the size of the busy period is the response time of the process.

To illustrate how the response time analysis is used, consider process set D given in Table 13.8.

The highest-priority process, a, will have a response time equal to its computation time (for example, $R_a = 3$). The next process will need to have its response time calculated. Let w_b^0 equal the computation time of process a, which is 3. Equation (13.5)

Process	Period, T	Computation time, C	Priority, P
a	7	3	3
b	12	3	2
c	20	5	1

Table 13.8 Process set D

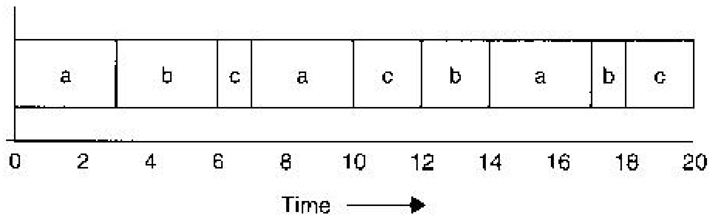

Figure 13.5 Gantt chart for process set D.

is used to derive the next value of w:

$$w_b^1 = 3 + \left\lceil \frac{3}{7} \right\rceil 3$$

that is, $w_b^1 = 6$. This value now balances the Equation ($w_b^2 = w_b^1 = 6$) and the response time of process b has been found (that is, $R_b = 6$).

The final process will give rise to the following calculations:

$$w_c^0 = 5$$

$$w_c^1 = 5 + \left\lceil \frac{5}{7} \right\rceil 3 + \left\lceil \frac{5}{12} \right\rceil 3 = 11$$

$$w_c^2 = 5 + \left\lceil \frac{11}{7} \right\rceil 3 + \left\lceil \frac{11}{12} \right\rceil 3 = 14$$

$$w_c^3 = 5 + \left\lceil \frac{14}{7} \right\rceil 3 + \left\lceil \frac{14}{12} \right\rceil 3 = 17$$

$$w_c^4 = 5 + \left\lceil \frac{17}{7} \right\rceil 3 + \left\lceil \frac{17}{12} \right\rceil 3 = 20$$

$$w_c^5 = 5 + \left\lceil \frac{20}{7} \right\rceil 3 + \left\lceil \frac{20}{12} \right\rceil 3 = 20$$

Hence R_c has a worst-case response time of 20, which means that it will just meet its deadline. This behaviour is illustrated in the Gantt chart shown in Figure 13.5.

Consider again the process set C. This set failed the utilization-based test but was observed to meet all its deadlines up to time 80. Table 13.9 shows the response times calculated by the above method for this collection. Note that all processes are now predicted to complete before their deadlines.

The response time calculations have the advantage that they are sufficient and necessary – if the process set passes the test they will meet all their deadlines; if they

Process	Period, T	Computation time, C	Priority, P	Response time, R
a	80	40	1	80
b	40	10	2	15
c	20	5	3	5

Table 13.9 Response time for process set C.

Process	T(=D)	C
a	4	1
b	12	3
c	16	8

Table 13.10 A process set for EDF.

fail the test, then, at run-time, a process will miss its deadline (unless the computation time estimations, C, themselves turn out to be pessimistic). As these tests are superior to the utilization-based ones, this chapter will concentrate on extending the applicability of the response time method.

13.6 Response time analysis for EDF

One of the disadvantages of the EDF scheme is that the worst-case response time for each process does not occur when all processes are released at a critical instant. In that situation only processes with a shorter relative deadline will interfere. But later there may exist a position in which all (or at least more) processes have a shorter absolute deadline. For example, consider a three process system as depicted in Table 13.10. The behaviour of process b illustrates the problem. At time 0, a critical instant, b only gets interference from process a (once) and has a response time of 4. But at its next release (at time 12) process c is still active and has a shorted deadline (16 versus 24) and hence c takes precedence; the response time for this second release of b is 8, twice the value obtained at the critical instant. Later releases may give an even larger value, although it is bounded at 12 as the system is schedulable by EDF (utilization is 1). Hence to find the worst case is much more complex. It is necessary to consider all process releases to see which one suffers the maximum interference from other processes with shorter deadlines.

In the simple model with all periodic processes, the full process set will repeat its execution every *hyper-period*; that is, the least common multiple (LCM) of the process periods. For example, in a small system with only four processes but periods of 24, 50, 73 and 101 time units, the LCM is 4 423 800. To find the worst-case response time for