13 SOLUTIONS TO THE EXERCISES

SOLUTIONS FOR CHAPTER 1

1.1 Fast computing tends to minimize the average response time of computation activities, whereas real-time computing is required to guarantee the timing constraints of each task.

1.2 The main limitations of the current real-time kernels are mainly due to the fact that they are developed to minimize runtime overhead (hence functionality) rather than offering support for a predictable execution. For example, short interrupt latency is good for servicing I/O devices, but introduces unpredictable delays in task execution for the high priority given to the interrupt handlers. Scheduling is mostly based on fixed priority, and explicit timing constraints cannot be specified on tasks. No specific support is usually provided for periodic tasks and no aperiodic service mechanism is available for handling event-driven activities. Access to shared resources is often realized through classical semaphores, which are efficient, but prone to priority inversion if no protocol is implemented for entering critical sections. Finally, no temporal protection or resource reservation mechanism is usually available in current real-time kernels for coping with transient overload conditions, so a task executing too much may introduce unbounded delays on the other tasks.

1.3 A real-time kernel should allow the user to specify explicit timing constraints on application tasks and support a predictable execution of real-time activities with specific real-time mechanisms, including scheduling, resource management, synchronization, communication, and interrupt handling. In critical real-time systems, predictability is more important than high performance, and often an increased functionality can only be reached at the expense of a higher runtime overhead. Other important features that a real-time system should have include maintainability, fault-tolerance, and overload management.

1.4 Three approaches can be used. The first one is to disable all external interrupts, letting application tasks access peripheral devices through polling. This solution gives great programming flexibility and reduces unbounded delays caused by the driver execution, but it characterized by a low processor efficiency on I/O operations, due to the busy wait.

A second solution is to disable interrupts and handle I/O devices by polling through a dedicated periodic kernel routine, whose load can be taken into account through a specific utilization factor. As in the previous solution, the major problem of this approach is due to the busy wait, but the advantage is that all hardware details can be encapsulated into a kernel routine and do not need to be known to the application tasks. An additional overhead is due to the extra communication required among application tasks and the kernel routine for exchanging I/O data.

A third approach is to enable interrupts but limit the execution of interrupt handlers as much as possible. In this solution, the interrupt handler activates a device handler, which is a dedicated task that is scheduled (and guaranteed) by the kernel as any other application task. This solution is efficient and minimizes the interference caused by interrupts.

1.5 The restrictions that should be used in a programming language to permit the analysis of real-time applications should limit the variability of execution times. Hence, a programmer should avoid using dynamic data structures, recursion, and all high level constructs that make execution time unpredictable. Possible language extensions should be aimed at facilitating the estimation of worst-case execution times. For example, a language could allow the programmer to specify the maximum number of iterations in each loop construct, and the probability of taking a branch in conditional statements.

SOLUTIONS FOR CHAPTER 2

2.1 A schedule is formally defined as a step function $\sigma : \mathbf{R}^+ \rightarrow \mathbf{N}$ such that $\forall t \in \mathbf{R}^+$, $\exists t_1, t_2$ such that $t \in [t_1, t_2)$ and $\forall t' \in [t_1, t_2)$ $\sigma(t) = \sigma(t')$. For any $k > 0$, $\sigma(t) = k$, means that task J_k is executing at time t, while $\sigma(t) = 0$

means that the CPU is idle. A schedule is said to be *preemptive* if the running task can be arbitrarily suspended at any time to assign the CPU to another task according to a predefined scheduling policy. In a preemptive schedule, tasks may be executed in disjointed interval of times. In a non-preemptive schedule, a running task cannot be interrupted and therefore it proceeds until completion.

2.2 A periodic task consists of an infinite sequence of identical jobs that are regularly activated at a constant rate. If ϕ_i is the activation time of the first job of task τ_i, the activation time of the kth job is given by $\phi_i + (k-1)T_i$, where T_i is the task period. Aperiodic tasks also consist of an infinite sequence of identical jobs; however, their activations are not regular. An aperiodic task where consecutive jobs are separated by a minimum interarrival time is called a sporadic task. The most important timing parameters defined for a real-time task are

- the *arrival time* (or *release time*); that is, the time at which a task becomes ready for execution;
- the *computation time*; that is, the time needed by the processor for executing the task without interruption;
- the *absolute deadline*; that is, the time before which a task should be completed to avoid damage to the system;
- the *finishing time*; that is, the time at which a task finishes its execution;
- the *response time*; that is, the difference between the finishing time and the release time: $R_i = f_i - r_i$;

2.3 A real-time application consisting of tasks with precedence relations is shown in Section 2.2.2.

2.4 A static scheduler is one in which scheduling decisions are based on fixed parameters, assigned to tasks before their activation. In a dynamic scheduler, scheduling decisions are based on dynamic parameters that may change during system evolution. A scheduler is said to be *off-line* if it is pre-computed (before task activation) and stored in a table. In an online scheduler, scheduling decisions are taken at runtime when a new task enters the system or when a running task terminates. An algorithm is said to be *optimal* if it minimizes some given cost function defined over the task set. A common optimality criterion for real-time system is related to feasibility. Then, a scheduler is optimal whenever it can find a feasible schedule, if one exists. Heuristic schedulers use a heuristic function to search for a feasible schedule; hence it is not guaranteed that a feasible solution is found.

2.5 An example of domino effect is shown in Figure 2.15.

SOLUTIONS FOR CHAPTER 3

3.1 To check whether the EDD algorithm produces a feasible schedule, tasks must be ordered with increasing deadlines, as shown in Table 13.1:

	J_1'	J_2'	J_3'	J_4'
C_i'	2	4	3	5
D_i'	5	9	10	16

Table 13.1 Task set ordered by deadline.

Then applying Equation (3.1) we have

$$\begin{aligned} f_1' &= C_1' = 2 \\ f_2' &= f_1' + C_2' = 6 \\ f_3' &= f_2' + C_3' = 9 \\ f_4' &= f_3' + C_4' = 14 \end{aligned}$$

Since each finishing time is less than the corresponding deadline, the task set is schedulable by EDD.

3.2 The algorithm for finding the maximum lateness of a task set scheduled by the EDD algorithm is shown in Figure 13.1.

3.3 The scheduling tree constructed by the Bratley's algorithm for the following set of non-preemptive tasks is illustrated in Figure 13.2.

	J_1	J_2	J_3	J_4
a_i	0	4	2	6
C_i	6	2	4	2
D_i	18	8	9	10

Table 13.2 Task set parameters for the Bratley's algorithm.

3.4 The schedule found by the Spring algorithm on the scheduling tree developed in the previous exercise with the heuristic function $H = a + C + D$ is $\{J_2, J_4, J_3, J_1\}$, which is unfeasible, since J_3 and J_4 miss their deadlines. Noted that the feasible solution is found with $H = a + d$.

```
**Algorithm: EDD_$L_{max}$($\mathcal{J}$)**
{
    $L_{max} = -D_n$;
    $f_0 = 0$;
    **for** (each $J_i \in \mathcal{J}$) {
        $f_i \ = \ f_{i-1} + C_i$;
        $L_i \ = \ f_i + D_i$;
        **if** $(L_i > L_{max})$ $L_{max} = L_i$;
    }
    return($L_{max}$);
}
```

Figure 13.1 Algorithm for finding the maximum lateness of a task set scheduled by EDD.

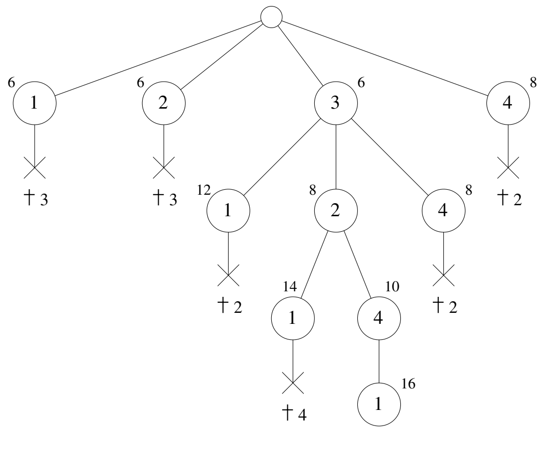

Figure 13.2 Scheduling tree constructed by the Bratley's algorithm for the task set shown in Table 13.2.

3.5 The precedence graph is shown in Figure 13.3.

By applying the transformation algorithm by Chetto and Chetto, we get the parameters shown in Table 13.3.

So the schedule produced by EDF will be $\{B, A, D, C, E, F, G\}$.

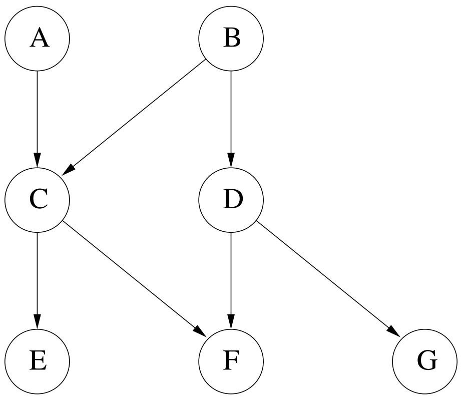

Figure 13.3 Precedence graph for Exercise 3.5.

	C_i	r_i	$r*_i$	d_i	$d*_i$
A	2	0	0	25	20
B	3	0	0	25	15
C	3	0	3	25	23
D	5	0	3	25	20
E	1	0	6	25	25
F	2	0	8	25	25
G	5	0	8	25	25

Table 13.3 Task set parameters modified by the Chetto and Chetto's algorithm.

SOLUTIONS FOR CHAPTER 4

4.1 The processor utilization factor of the task set is

$$U = \frac{2}{6} + \frac{2}{8} + \frac{2}{12} = 0.75$$

and considering that for three tasks the utilization least upper bound is

$$U_{lub}(3) = 3(2^{1/3} - 1) \simeq 0.78$$

from the Liu and Layland test, since $U \leq U_{lub}$, we can conclude that the task set is schedulable by RM, as shown in Figure 13.4.

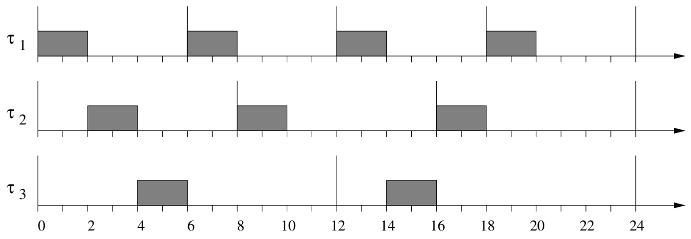

Figure 13.4 Schedule produced by Rate Monotonic for the task set of Exercise 4.1.

4.2 The processor utilization factor of the task set is

$$U = \frac{3}{5} + \frac{1}{8} + \frac{1}{10} = 0.825,$$

which is greater than $U_{lub}(3)$. Hence, we cannot verify the feasibility with the Liu and Layland test. Using the Hyperbolic Bound, we have the following:

$$\prod_{i=1}^{n}(U_i + 1) = 1.98,$$

which is less than 2. Hence, we can conclude that the task set is schedulable by RM, as shown in Figure 13.5.

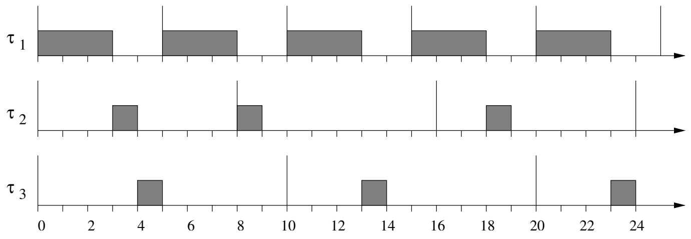

Figure 13.5 Schedule produced by Rate Monotonic for the task set of Exercise 4.2.

4.3 Applying the Liu and Layland test we have

$$U = \frac{1}{4} + \frac{2}{6} + \frac{3}{10} = 0.88 > 0.78,$$

so we cannot conclude anything. With the Hyperbolic Bound we have

$$\prod_{i=1}^{n}(U_i + 1) = 2.16 > 2$$

so we cannot conclude anything. Applying the Response Time Analysis we have to compute the response times and verify that they are less than or equal to the relative deadlines (which in this case are equal to periods). Hence, we have the following:

$$R_1 = C_1 = 1$$

So τ_1 does not miss its deadline. For τ_2 we have

$$\begin{aligned} R_2^{(0)} &= \sum_{j=1}^{2} C_j = C_1 + C_2 = 3 \\ R_2^{(1)} &= C_2 + \left\lceil \frac{R_2^{(0)}}{T_1} \right\rceil C_1 = 2 + \left\lceil \frac{3}{4} \right\rceil 1 = 3. \end{aligned}$$

So $R_2 = 3$, meaning that τ_2 does not miss its deadline. For τ_3 we have

$$\begin{aligned} R_3^{(0)} &= \sum_{j=1}^{3} C_j = C_1 + C_2 + C_3 = 6 \\ R_3^{(1)} &= C_3 + \left\lceil \frac{R_3^{(0)}}{T_1} \right\rceil C_1 + \left\lceil \frac{R_3^{(0)}}{T_2} \right\rceil C_2 = 2 + \left\lceil \frac{6}{4} \right\rceil 1 + \left\lceil \frac{6}{6} \right\rceil 2 = 7 \\ R_3^{(2)} &= 2 + \left\lceil \frac{7}{4} \right\rceil 1 + \left\lceil \frac{7}{6} \right\rceil 2 = 9 \\ R_3^{(3)} &= 2 + \left\lceil \frac{9}{4} \right\rceil 1 + \left\lceil \frac{9}{6} \right\rceil 2 = 10 \\ R_3^{(4)} &= 2 + \left\lceil \frac{10}{4} \right\rceil 1 + \left\lceil \frac{10}{6} \right\rceil 2 = 10. \end{aligned}$$

So $R_3 = 10$, meaning that τ_3 does not miss its deadline. Hence, we can conclude that the task set is schedulable by RM, as shown in Figure 13.6.

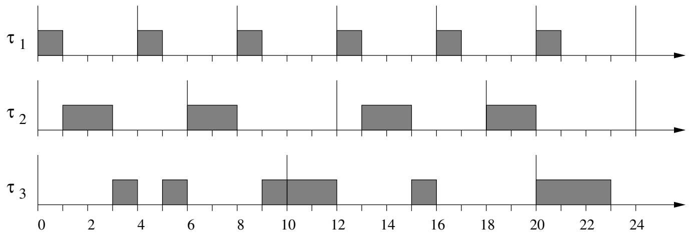

Figure 13.6 Schedule produced by Rate Monotonic for the task set of Exercise 4.3.

4.4 Applying the Response Time Analysis, we can easily verify that $R_3 = 10$ (see the solution of the previous exercise); hence the task set is not schedulable by RM.

4.5 Since

$$U = \frac{1}{4} + \frac{2}{6} + \frac{3}{8} = 0.96 < 1$$

the task set is schedulable by EDF, as shown in Figure 13.7.

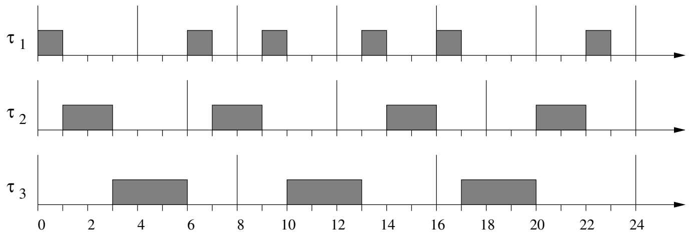

Figure 13.7 Schedule produced by EDF for the task set of Exercise 4.5.

4.6 Applying the processor demand criterion, we have to verify that

$$\forall L \in \mathcal{D} \quad \sum_{i=1}^{n} \left\lfloor \frac{L + T_i - D_i}{T_i} \right\rfloor C_i \;\leq\; L.$$

where

$$\mathcal{D} = \{d_k \mid d_k \leq \min(L^*, H)\}.$$

For the specific example, we have

$$\begin{aligned} U &= \frac{2}{6} + \frac{2}{8} + \frac{4}{12} = \frac{11}{12} \\ L^* &= \frac{\sum_{i=1}^{n}(T_i - D_i)U_i}{1 - U} = 32 \\ H &= \text{lcm}(6, 8, 12) = 24. \end{aligned}$$

Hence, the set of checking points is given by $\mathcal{D} = \{4, 5, 8, 11, 12, 17, 20, 23\}$. Since the demand in these intervals is $\{2, 4, 8, 10, 12, 14, 20, 22\}$ we can conclude that the task set is schedulable by EDF. The resulting schedule is shown in Figure 13.8.

4.7 Applying the Response Time Analysis, we have to start by computing the response time of task τ_2, which is the one with the shortest relative deadline, and hence the highest priority:

$$R_2 \;=\; C_2 = 2.$$

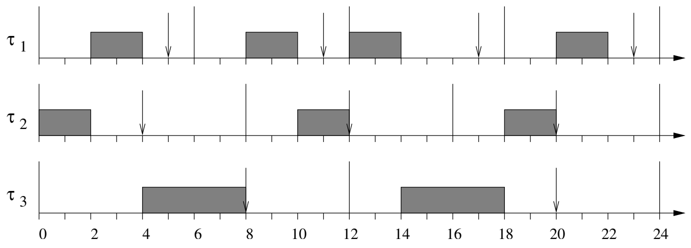

Figure 13.8 Schedule produced by EDF for the task set of Exercise 4.6.

So τ_2 does not miss its deadline. For τ_1 we have

$$\begin{aligned} R_1^{(0)} &= \sum_{j=1}^{2} C_j = C_1 + C_2 = 4 \\ R_1^{(1)} &= C_1 + \left\lceil \frac{R_1^{(0)}}{T_2} \right\rceil C_2 = 2 + \left\lceil \frac{4}{8} \right\rceil 2 = 4 \end{aligned}$$

So $R_1 = 4$, meaning that τ_1 does not miss its deadline. For τ_3 we have

$$\begin{aligned} R_3^{(0)} &= \sum_{j=1}^{3} C_j = C_1 + C_2 + C_3 = 8 \\ R_3^{(1)} &= C_3 + \left\lceil \frac{R_3^{(0)}}{T_2} \right\rceil C_2 + \left\lceil \frac{R_3^{(0)}}{T_1} \right\rceil C_1 = 4 + \left\lceil \frac{8}{8} \right\rceil 2 + \left\lceil \frac{8}{6} \right\rceil 2 = 10 \end{aligned}$$

And since $R_3^{(1)} > D_3$, we can conclude that the task set is not schedulable by DM. The resulting schedule is shown in Figure 13.9.

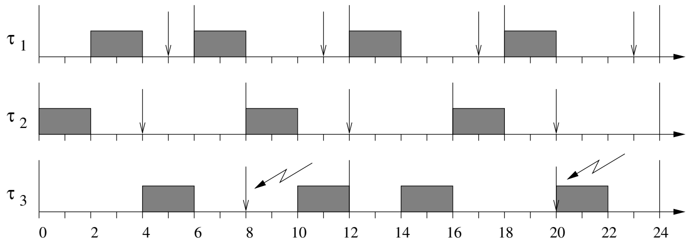

Figure 13.9 Schedule produced by Deadline Monotonic for the task set of Exercise 4.7.

SOLUTIONS FOR CHAPTER 5

5.1 The maximum Sporadic Server size can be computed by Equation (5.24):

$$U_{SS}^{max} = \frac{2-P}{P}$$

where

$$P = \prod_{i=1}^{n}(U_i + 1) = \frac{7}{6} \cdot \frac{9}{7} = \frac{3}{2}.$$

Hence substituting the value of P into Equation (5.24) we have

$$U_{SS}^{max} = \frac{1}{3}.$$

To enhance aperiodic responsiveness, the server must run at the highest priority, and this can be achieved by setting its period to $T_s = T_1 = 6$. Then, assuming $U_s = U_{SS}^{max}$, its capacity will be $C_s = U_s T_s = 2$.

5.2 The maximum Deferrable Server size can be computed by Equation (5.15). Hence:

$$U_{DS}^{max} = \frac{2-P}{2P-1}.$$

And substituting the value of $P = 3/2$ into Equation (5.15) we have

$$U_{DS}^{max} = \frac{1}{4}.$$

Hence, by setting $U_s = U_{DS}^{max}$ and $T_s = T_1 = 6$, the capacity will be $C_s = U_s T_s = 6/4 = 1.5$.

5.3 Following the same steps reported in Exercise 5.1, we know that the maximum utilization that can be assigned to a Polling Server to guarantee the periodic task set is

$$U_{PS}^{max} = \frac{2-P}{P} = \frac{1}{3}.$$

So, by setting $T_s = 6$ (intermediate priority) and $C_s = 2$, we satisfy the constraints. The resulting schedule is illustrated in Figure 13.10.

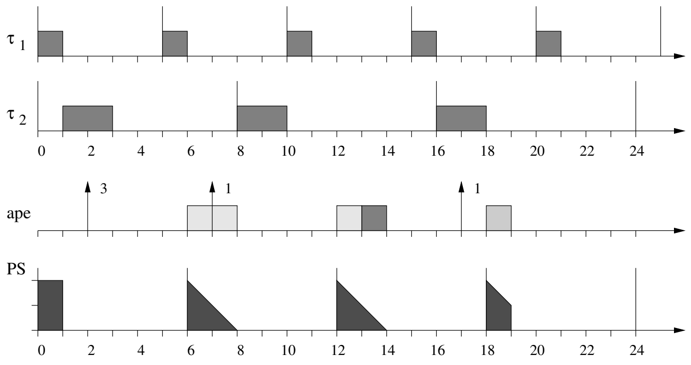

Figure 13.10 Schedule produced by Rate Monotonic and Polling Server for the task set of Exercise 5.3.

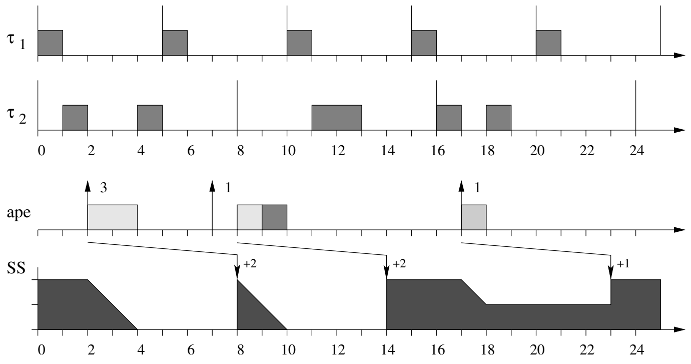

Figure 13.11 Schedule produced by Rate Monotonic and Sporadic Server for the task set of Exercise 5.4.

5.4 A Sporadic Server can be guaranteed with the same method used for the Polling Server. So, using the same parameters computed before ($C_s = 2$ and $T_s = 6$) we have the schedule shown in Figure 13.11.

5.5 Applying Equation (5.15) to the considered task set, we see that the maximum utilization that can be assigned to a Deferrable Server to guarantee the periodic task set is $U_{s_{max}} = 1/4$. So, by setting $T_s = 4$ (maximum priority) and $C_s = 1$, we satisfy the constraints. The resulting schedule is illustrated in Figure 13.12.

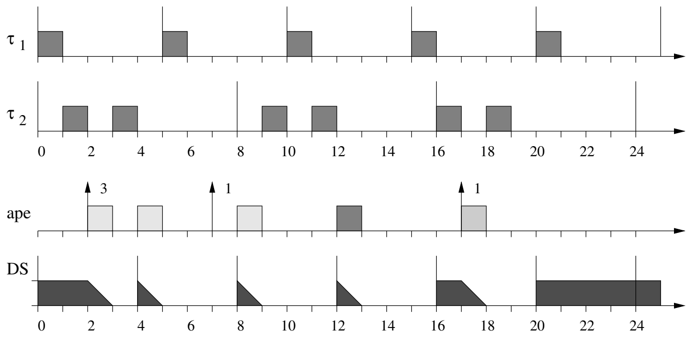

Figure 13.12 Schedule produced by Rate Monotonic and Deferrable Server for the task set of Exercise 5.5.

5.6 The resulting schedule is illustrated in Figure 13.13.

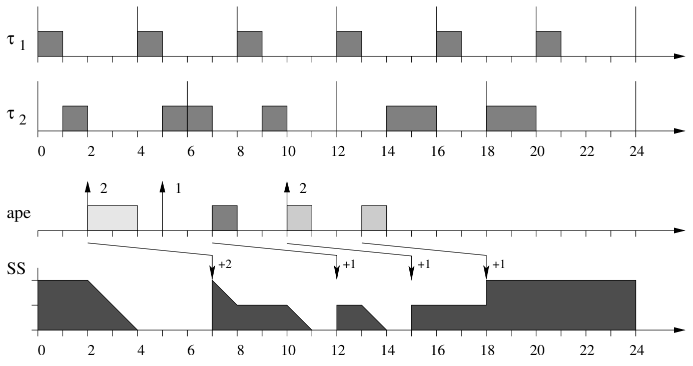

Figure 13.13 Schedule produced by Rate Monotonic and Sporadic Server for the task set of Exercise 5.6.

SOLUTIONS FOR CHAPTER 6

6.1 For any dynamic server we must have $U_p + U_s \leq 1$; hence, considering that $U_p = 2/3$, the maximum server utilization that can be assigned to a Dynamic Sporadic Server is

$$U_s = 1 - U_p = 1/3.$$

6.2 The deadlines computed by the server for the aperiodic jobs result: $d_1 = a_1 + T_s = 7$, $d_2 = d_1 + T_s = 13$, and $d_3 = a_3 + T_s = 21$. The resulting schedule produced by EDF + DSS is illustrated in Figure 13.14.

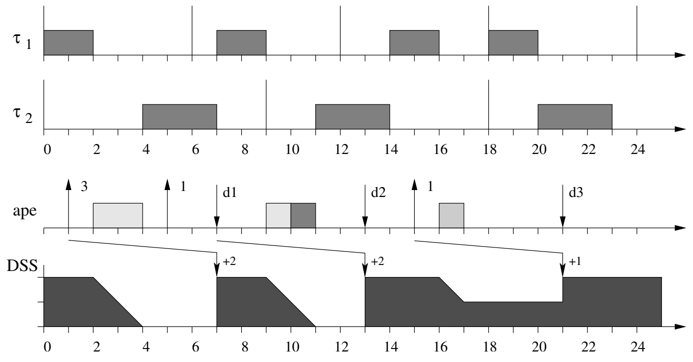

Figure 13.14 Schedule produced by EDF + DDS for the task set of Exercise 6.2.

6.3 The deadlines computed by the server for the aperiodic jobs are $d_1 = a_1 + C_1/U_s = 10$, $d_2 = d_1 + C_2/U_s = 13$, and $d_3 = a_3 + C_3/U_s = 18$. The resulting schedule produced by EDF + TBS is illustrated in Figure 13.15.

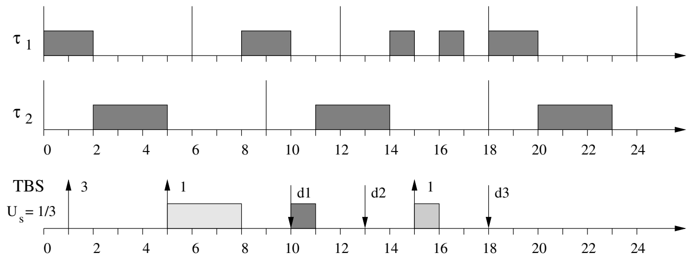

Figure 13.15 Schedule produced by EDF + TBS for the task set of Exercise 6.3.

6.4 The events handled by the CBS are

time	event	action
$t = 1$	arrival	$c_s = Q_s, d_s = a_1 + T_s = 7$
$t = 4$	$c_s = 0$	$c_s = Q_s, d_s = d_s + T_s = 13$
$t = 5$	arrival	enqueue request
$t = 11$	$c_s = 0$	$c_s = Q_s, d_s = d_s + T_s = 19$
$t = 15$	arrival	$c_s = Q_s, d_s = a_3 + T_s = 21$

The resulting schedule produced by EDF + CBS is illustrated in Figure 13.16.

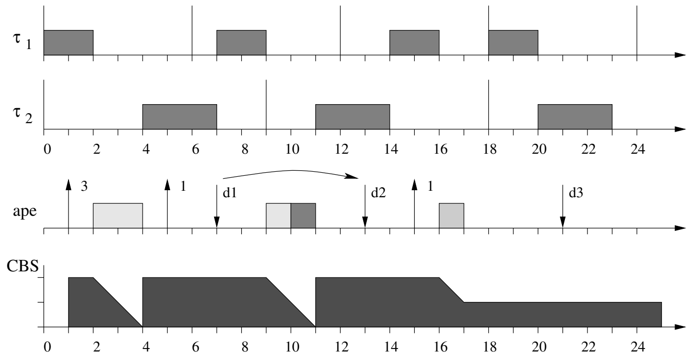

Figure 13.16 Schedule produced by EDF + CBS for the task set of Exercise 6.4.

6.5 The deadlines computed by the server are

$$\begin{aligned} d_1^{(0)} &= a_1 + C_1/U_s = 10 \\ d_1^{(1)} &= f_1^{(0)} = 8 \\ \\ d_2^{(0)} &= d_2^{(0)} + C_2/U_s = 13 \\ d_2^{(1)} &= f_2^{(0)} = 11 \\ \\ d_3^{(0)} &= a_3 + C_3/U_s = 18 \\ d_3^{(1)} &= f_3^{(0)} = 16. \end{aligned}$$

The resulting schedule produced by EDF + TB(1) is illustrated in Figure 13.17.

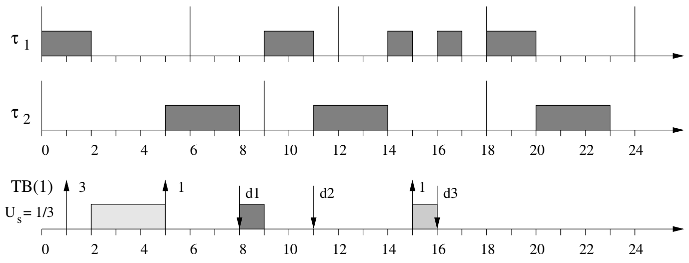

Figure 13.17 Schedule produced by EDF + TB(1) for the task set of Exercise 6.5.

6.6 The deadlines computed by the server are

$$\begin{aligned}
d_1^{(0)} &= a_1 + C_1/U_s = 10\\
d_1^{(1)} &= f_1^{(0)} = 8\\
d_1^{(2)} &= f_1^{(1)} = 5\\
d_1^{(3)} &= f_1^{(2)} = 4\\
\\
d_2^{(0)} &= d_2^{(0)} + C_2/U_s = 13\\
d_2^{(1)} &= f_2^{(0)} = 11\\
d_2^{(2)} &= f_2^{(1)} = 9\\
d_2^{(3)} &= f_2^{(2)} = 6\\
\\
d_3^{(0)} &= a_3 + C_3/U_s = 18\\
d_3^{(1)} &= f_3^{(0)} = 16.
\end{aligned}$$

The resulting schedule produced by EDF + TB* is illustrated in Figure 13.18.

6.7 The resulting schedule is illustrated in Figure 13.19.

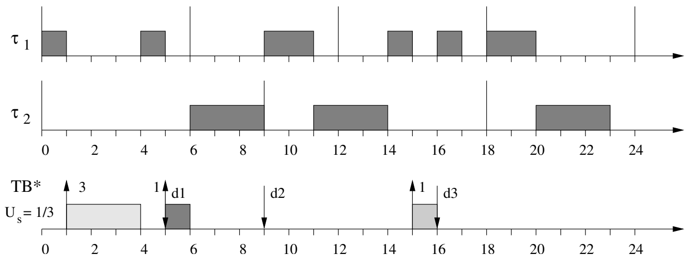

Figure 13.18 Schedule produced by EDF + TB* for the task set of Exercise 6.6.

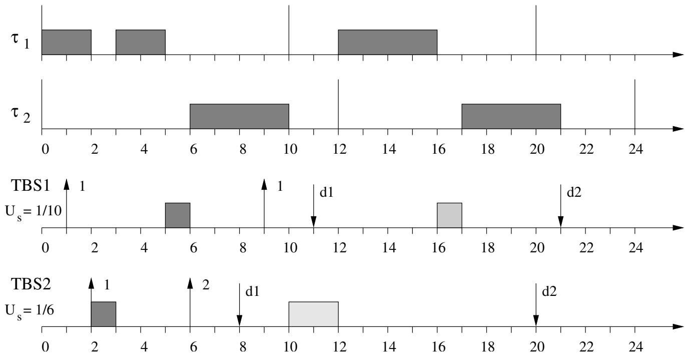

Figure 13.19 Schedule produced by EDF+TB_1+TB_2 for the task set of Exercise 6.7.

6.8 First of all, the utilization of the periodic task set is

$$U_p = \frac{8}{20} + \frac{6}{30} = \frac{3}{5} = 0.6;$$

hence, the largest utilization that can be assigned to a CBS is $U_s = 1 - U_p = 0.4$. Then, converting all times in microseconds and substituting the values in Equation (6.15) we obtain

$$\begin{aligned} C^{\mathsf{avg}} &= 1200\,\mu s \\ T_s &= \frac{10}{4}\left(20 + \sqrt{\frac{20 \cdot 1200}{0.6}}\right) = 550\mu s \\ Q_s &= T_s U_s = 220\mu s. \end{aligned}$$

SOLUTIONS FOR CHAPTER 7

7.1 Applying Equation (7.19), we can verify that

$$\forall i, \;\; 1 \leq i \leq n, \quad \sum_{k=1}^{i} \frac{C_k}{T_k} + \frac{B_i}{T_i} \;\leq\; i(2^{1/i} - 1).$$

So we have

$$\begin{aligned}
\frac{C_1 + B_1}{T_1} &= \frac{9}{10} < 1 \\
\frac{C_1}{T_1} + \frac{C_2 + B_2}{T_2} &= \frac{4}{10} + \frac{6}{15} = 0.8 < 0.83 \\
\frac{C_1}{T_1} + \frac{C_2}{T_2} + \frac{C_3}{T_3} &= \frac{4}{10} + \frac{3}{15} + \frac{4}{20} = 0.8 > 0.78.
\end{aligned}$$

Being condition (7.19) only sufficient, we cannot conclude anything about feasibility. By applying the response time analysis we have to verify that

$$\forall i, \;\; 1 \leq i \leq n, \quad R_i \;\leq\; D_i$$

where

$$R_i \;=\; C_i + B_i + \sum_{k=1}^{i-1} \left\lceil \frac{R_i}{T_k} \right\rceil C_k.$$

So we have

$$\begin{aligned}
R_1 &= C_1 + B_1 \;=\; 9 < 10 \\
\\
R_2^{(0)} &= C_1 + C_2 + B_2 = 10 \\
R_2^{(1)} &= C_2 + B_2 + \left\lceil \frac{10}{10} \right\rceil 4 = 10 \;<\; 15 \\
\\
R_3^{(0)} &= C_1 + C_2 + C_3 = 11 \\
R_3^{(1)} &= C_3 + \left\lceil \frac{11}{10} \right\rceil 4 + \left\lceil \frac{11}{15} \right\rceil 3 = 15 \\
R_3^{(2)} &= C_3 + \left\lceil \frac{15}{10} \right\rceil 4 + \left\lceil \frac{15}{15} \right\rceil 3 = 15 \;<\; 20.
\end{aligned}$$

Hence, we can conclude that the task set is schedulable by RM.

7.2 Using the Priority Inheritance Protocol, a task τ_i can be blocked at most for one critical section by each lower priority task. Moreover, a critical section can block τ_i only if it belongs to a task with lower priority and it is shared with τ_i (direct blocking) or with higher priority tasks (push-through blocking). Finally, we have to consider that two critical sections cannot block a task if they are protected by the same semaphore or they belong to the same task. Hence, if $Z_{i,k}$ denotes the longest critical section of τ_i guarded by semaphore S_k, we have the following:

- Task τ_1 can only experience direct blocking (since there are no tasks with higher priority), and the set of critical sections that can potentially block it is $\{Z_{2A}, Z_{3A}, Z_{3C}\}$. It can be blocked at most for the duration of two critical sections in this set. Thus, the maximum blocking time is given by the sum of the two longest critical sections in this set. In this case, however, note that the longest critical sections are Z_{3A} and Z_{3C}, which belong to the same task; hence they cannot be selected together. Hence, the maximum blocking time for τ_1 is $B_1 = (\delta_{2A} - 1) + (\delta_{3C} - 1) = 2 + 5 = 7$.
- Task τ_2 can experience direct blocking on Z_{3A} and Z_{3B} and push-through blocking on Z_{3A} and Z_{3C}. Hence, the set of critical sections that can potentially block τ_2 is $\{Z_{3A}, Z_{3B}, Z_{3C}\}$. It can be blocked at most for the duration of one critical section in this set. Thus, the maximum blocking time is given by the longest critical section in this set, that is Z_{3C}. Hence, we have $B_2 = \delta_{3C} - 1 = 5$.
- Task τ_3 cannot be blocked, because it is the task with the lowest priority (it can only be preempted by higher priority tasks). Hence, we have $B_3 = 0$.

7.3 Using the Priority Ceiling Protocol, a task τ_i can be blocked at most for one critical section during its execution. The set of critical sections that can potentially block τ_i is the same as that computed for the Priority Inheritance Protocol. Hence, if $Z_{i,k}$ denotes the longest critical section of τ_i guarded by semaphore S_k, we have the following:

- The set of critical sections that can block τ_1 is $\{Z_{2A}, Z_{3A}, Z_{3C}\}$. Hence, the maximum blocking time for τ_1 is $B_1 = \delta_{3C} - 1 = 5$.
- The set of critical sections that can block τ_2 is $\{Z_{3A}, Z_{3B}, Z_{3C}\}$. Hence, the maximum blocking time for τ_2 is $B_2 = \delta_{3C} - 1 = 5$.
- Task τ_3 cannot be blocked, because it is the task with the lowest priority (it can only be preempted by higher priority tasks). Hence, $B_3 = 0$.

7.4 The maximum blocking time for τ_2 is given by a push-through blocking on Z_{3C}. For this to happen, τ_3 must start first and must enter its critical section Z_{3C}. Then, τ_1 must preempt τ_3, so that τ_3 can inherit the highest priority to prevent τ_2 from executing. The situation is illustrated in Figure 13.20.

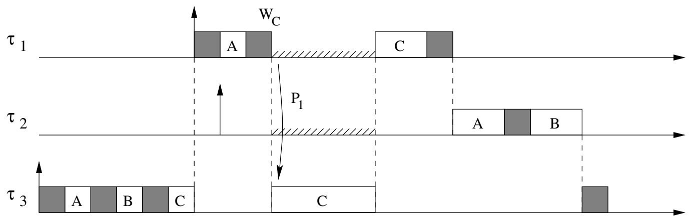

Figure 13.20 Schedule produced by RM + PIP for the task set of Exercise 7.2.

7.5 To compute the maximum blocking time under the Priority Inheritance Protocol we reason as follows.

- The set of critical sections that can potentially block τ_1 is $\{Z_{2C}, Z_{3B}, Z_{3E}, Z_{4A}, Z_{4C}, Z_{4E}\}$. Among these, we have to select the three longest ones, one for each lower priority task. Note that, if we select Z_{2C} and Z_{3E}, we cannot select Z_{4E} (which is the longest of τ_4) because resource E has already been selected for τ_3, and we cannot select Z_{4C} for the same reason. So, we have to select Z_{4A}. Hence, the maximum blocking time for τ_1 is $B_1 = (\delta_{2C} - 1) + (\delta_{3E} - 1) + (\delta_{4A} - 1) = 26$.
- Task τ_2 can experience direct blocking on Z_{4C} and push-through blocking on Z_{3B}, Z_{3E}, Z_{4A}, Z_{4C}, and Z_{4E}. Hence, the set of critical sections that can potentially block τ_2 is $\{Z_{3B}, Z_{3E}, Z_{4A}, Z_{4C}, Z_{4E}\}$. It can be blocked at most for the duration of two critical sections in this set. Thus, we have $B_2 = (\delta_{3E} - 1) + (\delta_{4C} - 1) = 21$. Note that Z_{3E} and Z_{4E} cannot block τ_2 together.
- Task τ_3 can experience direct blocking on Z_{4E} and push-through blocking on Z_{4A}, Z_{4C}, and Z_{4E}. Hence, the set of critical sections that can block τ_3 is $\{Z_{4A}, Z_{4C}, Z_{4E}\}$. It can be blocked at most for the duration of one critical section in this set. Thus, we have $B_3 = \delta_{4E} - 1 = 10$.
- Task τ_4 cannot be blocked, because it is the task with the lowest priority (it can only be preempted by higher priority tasks). Hence, we have $B_4 = 0$.

7.6 The sets of critical sections that can cause blocking under the Priority Ceiling Protocol are the same as those derived in the previous exercise for the Priority Inheritance Protocol. The only difference is that under the Priority Ceiling Protocol each task can only be blocked for the duration of a single critical section. Hence, we have the following:

- The set of critical sections that can potentially block τ_1 is $\{Z_{2C}, Z_{3B}, Z_{3E}, Z_{4A}, Z_{4C}, Z_{4E}\}$. Hence, the maximum blocking time for τ_1 is $B_1 = \delta_{3E} - 1 = 13$
- The set of critical sections that can potentially block τ_2 is $\{Z_{3B}, Z_{3E}, Z_{4A}, Z_{4C}, Z_{4E}\}$. Hence, the maximum blocking time for τ_2 is $B_2 = \delta_{3E} - 1 = 13$.
- The set of critical sections that can potentially block τ_3 is $\{Z_{4A}, Z_{4C}, Z_{4E}\}$. Hence, the maximum blocking time for τ_3 is $B_3 = \delta_{4E} - 1 = 10$.
- Task τ_4 cannot be blocked, because it is the task with the lowest priority (it can only be preempted by higher priority tasks). Hence, we have $B_4 = 0$.

7.7 The maximum blocking time for τ_2 is given by a push-through blocking on C_4 and E_3. This means that for this to happen, τ_4 must start first and must enter its critical section C_4. Then, τ_3 must preempt τ_4, entering E_3. Now, when τ_1 arrives, it experiences a chained blocking when entering C_1 and E_1, which are both locked. The situation is illustrated in Figure 13.21.

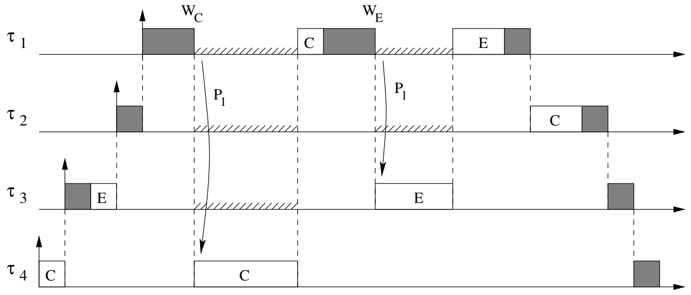

Figure 13.21 Schedule produced by RM + PIP for the task set of Exercise 7.7.

7.8 If tasks are assigned decreasing preemption levels as $\pi_1 = 3$, $\pi_2 = 2$, and $\pi_3 = 1$, the resource ceilings have the values shown in Table 13.4.

	$C_R(3)$	$C_R(2)$	$C_R(1)$	$C_R(0)$
A	0	1	2	3
B	0	0	0	2
C	-	0	2	3

Table 13.4 SRP resource ceilings resulting for Exercise 7.8.

SOLUTIONS FOR CHAPTER 8

8.1 We first note that the task set is feasible in fully preemptive mode, in fact

$$\begin{aligned}
R_1 &= C_1 = 2 \ \leq \ D_1 \\
\\
R_2^{(0)} &= C_1 + C_2 = 4 \\
R_2^{(1)} &= C_2 + \left\lceil \frac{4}{T_1} \right\rceil C_1 = 4 \ \leq \ D_2 \\
\\
R_3^{(0)} &= C_1 + C_2 + C_3 = 8 \\
R_3^{(1)} &= C_3 + \left\lceil \frac{8}{T_1} \right\rceil C_1 + \left\lceil \frac{8}{T_2} \right\rceil C_2 = 10 \\
R_3^{(2)} &= C_3 + \left\lceil \frac{10}{T_1} \right\rceil C_1 + \left\lceil \frac{10}{T_2} \right\rceil C_2 = 12 \\
R_3^{(3)} &= C_3 + \left\lceil \frac{12}{T_1} \right\rceil C_1 + \left\lceil \frac{12}{T_2} \right\rceil C_2 = 12 \ \leq \ D_2
\end{aligned}$$

Hence, by the result of Theorem 8.1, the feasibility of the task set in non-preemptive mode can be verified by just checking the first job of each task, when activated at its critical instant. The critical instant for task τ_i occurs when τ_i is activated together with all higher priority tasks, and one unit after the longest lower priority task.

Using Equation (8.1), the blocking times result to be $B_1 = 3$, $B_2 = 3$, $B_3 = 0$, and tasks response times can be computed as $R_i = S_i + C_i$, where S_i is given by Equation (8.8). So we have

$$\begin{aligned}
S_1 &= B_1 = 3 \\
R_1 &= S_1 + C_1 = 3 + 2 = 5 \ \leq \ D_1
\end{aligned}$$

$$\begin{aligned}
S_2^{(0)} &= B_2 + C_1 = 5 \\
S_2^{(1)} &= B_2 + \left(\left\lfloor \frac{5}{T_1} \right\rfloor + 1 \right) C_1 = 5 \\
R_2 &= S_2 + C_2 = 7 \ > \ D_2.
\end{aligned}$$

Hence, the task set is not schedulable by non-preemptive RM, since τ_2 misses its deadline.

8.2 We first note that the task set is not feasible in fully preemptive mode, since

$$\begin{aligned}
R_3^{(0)} &= C_1 + C_2 + C_3 = 9 \\
R_3^{(1)} &= C_3 + \left\lceil \frac{9}{T_1} \right\rceil C_1 + \left\lceil \frac{9}{T_2} \right\rceil C_2 = 12 \\
R_3^{(2)} &= C_3 + \left\lceil \frac{12}{T_1} \right\rceil C_1 + \left\lceil \frac{12}{T_2} \right\rceil C_2 = 15 \ > \ D_3.
\end{aligned}$$

Therefore, the response time of a task τ_i cannot be restricted to its first job, but has to be extended up to job $K_i = \lceil \frac{L_i}{T_i} \rceil$, where L_i is the longest Level-i Active Period. Using Equations (8.1), (8.2), and (8.3), we get the following results:

	B_i	L_i	K_i
τ_1	2	5	1
τ_2	2	8	1
τ_3	1	37	3
τ_4	0	38	1

For task τ_1 we have

$$\begin{aligned}
s_{1,1} &= B_1 = 2 \\
f_{1,1} &= s_{1,1} + C_1 = 2 + 3 = 5 \\
R_1 &= f_{1,1} = 5 \ \leq \ D_1.
\end{aligned}$$

For task τ_2 we have

$$\begin{aligned}
s_{2,1}^{(0)} &= B_2 + C_1 = 5 \\
s_{2,1}^{(1)} &= B_2 + \left(\left\lfloor \frac{5}{T_1} \right\rfloor + 1\right) C_1 = 5 \\
f_{2,1} &= s_{2,1} + C_2 = 5 + 2 = 7 \\
R_2 &= f_{2,1} = 7 \ \leq \ D_2.
\end{aligned}$$

For task τ_3, the response time must be checked in the first three jobs.
For $k = 1$:

$$\begin{aligned}
s_{3,1}^{(0)} &= B_3 + C_2 + C_1 = 7 \\
s_{3,1}^{(1)} &= B_3 + \left(\left\lfloor \frac{7}{T_1} \right\rfloor + 1\right) C_1 + \left(\left\lfloor \frac{7}{T_2} \right\rfloor + 1\right) C_2 = 7 \\
f_{3,1} &= s_{3,1} + C_3 = 7 + 3 = 10 \\
R_{3,1} &= f_{3,1} = 10.
\end{aligned}$$

For $k = 2$:

$$\begin{aligned}
s_{3,2}^{(0)} &= B_3 + C_3 + C_1 + C_2 = 10 \\
s_{3,2}^{(1)} &= B_3 + C_3 + \left(\left\lfloor \frac{10}{T_1} \right\rfloor + 1\right) C_1 + \left(\left\lfloor \frac{10}{T_2} \right\rfloor + 1\right) C_2 = 16 \\
s_{3,2}^{(2)} &= B_3 + C_3 + \left(\left\lfloor \frac{16}{T_1} \right\rfloor + 1\right) C_1 + \left(\left\lfloor \frac{16}{T_2} \right\rfloor + 1\right) C_2 = 19 \\
s_{3,2}^{(3)} &= B_3 + C_3 + \left(\left\lfloor \frac{19}{T_1} \right\rfloor + 1\right) C_1 + \left(\left\lfloor \frac{19}{T_2} \right\rfloor + 1\right) C_2 = 22 \\
s_{3,2}^{(4)} &= B_3 + C_3 + \left(\left\lfloor \frac{22}{T_1} \right\rfloor + 1\right) C_1 + \left(\left\lfloor \frac{22}{T_2} \right\rfloor + 1\right) C_2 = 22 \\
f_{3,2} &= s_{3,2} + C_3 = 22 + 3 = 25 \\
R_{3,2} &= f_{3,2} - T_3 = 25 - 14 = 11.
\end{aligned}$$

For $k = 3$:

$$\begin{array}{rcl}
s_{3,3}^{(0)} & = & B_3 + 2C_3 + C_1 + C_2 = 13 \\
s_{3,3}^{(1)} & = & B_3 + 2C_3 + \left(\left\lfloor \frac{13}{T_1} \right\rfloor + 1\right) C_1 + \left(\left\lfloor \frac{13}{T_2} \right\rfloor + 1\right) C_2 = 19 \\
s_{3,3}^{(2)} & = & B_3 + 2C_3 + \left(\left\lfloor \frac{19}{T_1} \right\rfloor + 1\right) C_1 + \left(\left\lfloor \frac{19}{T_2} \right\rfloor + 1\right) C_2 = 25 \\
s_{3,3}^{(3)} & = & B_3 + 2C_3 + \left(\left\lfloor \frac{25}{T_1} \right\rfloor + 1\right) C_1 + \left(\left\lfloor \frac{25}{T_2} \right\rfloor + 1\right) C_2 = 28 \\
s_{3,3}^{(4)} & = & B_3 + 2C_3 + \left(\left\lfloor \frac{28}{T_1} \right\rfloor + 1\right) C_1 + \left(\left\lfloor \frac{28}{T_2} \right\rfloor + 1\right) C_2 = 31 \\
s_{3,3}^{(5)} & = & B_3 + 2C_3 + \left(\left\lfloor \frac{31}{T_1} \right\rfloor + 1\right) C_1 + \left(\left\lfloor \frac{31}{T_2} \right\rfloor + 1\right) C_2 = 31 \\
f_{3,3} & = & s_{3,3} + C_3 = 31 + 3 = 34 \\
R_{3,3} & = & f_{3,3} - 2T_3 = 34 - 28 = 6.
\end{array}$$

Hence for τ_3 we have that $R_3 = \max\{R_{3,1}, R_{3,2}, R_{3,3}\} = 11 \leq D_3$. For τ_4, it can be easily verified that $R_4 = L_4 = 38 \leq D_4$. Hence, we conclude that the task set is schedulable by non-preemptive RM.

8.3 Using the Liu and Layland test, the blocking tolerance of each task can be computed by Equation (8.20), where $U_{lub} = 1$, since tasks are scheduled by EDF:

$$\beta_i = \left\lfloor T_i \left(1 - \sum_{h:P_h \geq P_i} \frac{C_h}{T_h} \right) \right\rfloor .$$

and, according to Theorem (8.2), Q_i results to be:

$$Q_i = \min\{Q_{i-1}, \beta_{i-1} + 1\}$$

where $Q_1 = \infty$ and $\beta_1 = D_1 - C_1$. Hence, we have

	U_i	$\sum_{h=1}^{i} U_h$	β_i	Q_i
τ_1	1/4	1/4	6	∞
τ_2	1/5	9/20	5	7
τ_3	1/6	37/60	11	6
τ_4	1/12	42/60	18	6
τ_5	1/30	44/60	24	6

Note that, being $Q_i \geq C_i$ for all i's, all tasks can execute non preemptively.

8.4 Under Rate Monotonic, using the Liu and Layland test, the blocking tolerance of each task can be computed by Equation (8.20):

$$\beta_i = T_i \left(U_{lub}(i) - \sum_{h=1}^{i} \frac{C_h}{T_h} \right).$$

And, according to Theorem (8.2), Q_i results to be:

$$Q_i = \min\{Q_{i-1}, \beta_{i-1} + 1\}$$

where $Q_1 = \infty$ and $\beta_1 = D_1 - C_1$. Hence, we have

	$U_{lub}(i)$	$\sum_{h=1}^{i} U_h$	β_i	Q_i
τ_1	1.0	1/4	6	∞
τ_2	0.828	9/20	3	7
τ_3	0.780	37/60	4	4
τ_4	0.757	42/60	3	4
τ_5	0.743	44/60	0	4

Hence, to make the task set schedulable under RM, one preemption point must be inserted in τ_3 and τ_4.

8.5 First of all, from the task structures, the following parameters can be derived:

	C_i	T_i	U_i	q_i^{max}	q_i^{last}	B_i
τ_1	6	24	1/4	3	0	7
τ_2	10	40	1/4	4	4	7
τ_3	18	120	3/20	8	5	5
τ_4	15	150	1/10	6	6	0

We note that since the total utilization is $U = 0.75$, the task set is schedulable under Rate Monotonic in fully preemptive mode (in fact $U_{lub}(4) = 0.757$). Hence, the worst-case response time of each task can be computed considering the first job under the critical instant, using Equations (8.32) and (8.33).

For task τ_1 we have

$$R_1 \;=\; B_1 + C_1 = 7 + 6 = 13.$$

For task τ_2 we have

$$\begin{aligned}
S_2^{(0)} &= B_2 + C_1 + C_2 - q_2^{last} = 7 + 6 + 10 - 4 = 19 \\
S_2^{(1)} &= B_2 + C_2 - q_2^{last} + \left(\left\lfloor \frac{19}{T_1} \right\rfloor + 1 \right) C_1 = 19 \\
R_2 &= S_2 + q_2^{last} = 19 + 4 = 23.
\end{aligned}$$

For task τ_3 we have:

$$\begin{aligned}
S_3^{(0)} &= B_3 + C_1 + C_2 + C_3 - q_3^{last} = 5 + 6 + 10 + 18 - 5 = 34 \\
S_3^{(1)} &= B_3 + C_3 - q_3^{last} + \left(\left\lfloor \frac{34}{T_1} \right\rfloor + 1 \right) C_1 + \left(\left\lfloor \frac{34}{T_2} \right\rfloor + 1 \right) C_2 = 40 \\
S_3^{(2)} &= B_3 + C_3 - q_3^{last} + \left(\left\lfloor \frac{40}{T_1} \right\rfloor + 1 \right) C_1 + \left(\left\lfloor \frac{40}{T_2} \right\rfloor + 1 \right) C_2 = 50 \\
S_3^{(3)} &= B_3 + C_3 - q_3^{last} + \left(\left\lfloor \frac{50}{T_1} \right\rfloor + 1 \right) C_1 + \left(\left\lfloor \frac{50}{T_2} \right\rfloor + 1 \right) C_2 = 56 \\
S_3^{(4)} &= B_3 + C_3 - q_3^{last} + \left(\left\lfloor \frac{56}{T_1} \right\rfloor + 1 \right) C_1 + \left(\left\lfloor \frac{56}{T_2} \right\rfloor + 1 \right) C_2 = 56 \\
R_3 &= S_3 + q_3^{last} = 56 + 5 = 61
\end{aligned}$$

For task τ_4 we have

$$\begin{aligned}
S_4^{(0)} &= B_4 + C_1 + C_2 + C_3 + C_4 - q_4^{last} = 6 + 10 + 18 + 15 - 6 = 43 \\
S_4^{(1)} &= B_4 + C_4 - q_4^{last} + \left(\left\lfloor \frac{43}{T_1} \right\rfloor + 1 \right) C_1 + \left(\left\lfloor \frac{43}{T_2} \right\rfloor + 1 \right) C_2 + \left(\left\lfloor \frac{43}{T_3} \right\rfloor + 1 \right) C_3 = 59 \\
S_4^{(2)} &= B_4 + C_4 - q_4^{last} + \left(\left\lfloor \frac{59}{T_1} \right\rfloor + 1 \right) C_1 + \left(\left\lfloor \frac{59}{T_2} \right\rfloor + 1 \right) C_2 + \left(\left\lfloor \frac{59}{T_3} \right\rfloor + 1 \right) C_3 = 65 \\
S_4^{(3)} &= B_4 + C_4 - q_4^{last} + \left(\left\lfloor \frac{65}{T_1} \right\rfloor + 1 \right) C_1 + \left(\left\lfloor \frac{65}{T_2} \right\rfloor + 1 \right) C_2 + \left(\left\lfloor \frac{65}{T_3} \right\rfloor + 1 \right) C_3 = 65 \\
R_4 &= S_4 + q_4^{last} = 65 + 6 = 71.
\end{aligned}$$

SOLUTIONS FOR CHAPTER 9

9.1 Applying the definition of instantaneous load we have

time	$\rho_1(t)$	$\rho_2(t)$	$\rho(t)$
$t = 0$	0	5/10 = 0.5	0.5
$t = 1$	0	4/9 = 0.444	0.444
$t = 2$	0	3/8 = 0.375	0.375
$t = 3$	3/5 = 0.6	(3+2)/7 = 0.714	0.714
$t = 4$	2/4 = 0.5	(2+2)/6 = 0.667	0.667
$t = 5$	1/3 = 0.333	(1+2)/5 = 0.6	0.6
$t = 6$	0	2/4 = 0.5	0.5
$t = 7$	0	1/3 = 0.333	0.333
$t = 8$	0	0	0

9.2 Checking condition (9.24), necessary for the schedulability of the task set, we have

$$\sum_{i=1}^{n} \frac{C_i(S_i - 1)}{T_i S_i} = \frac{2}{5} + \frac{2 \cdot 3}{6 \cdot 4} + \frac{4 \cdot 4}{8 \cdot 5} = \frac{21}{20} > 1.$$

Hence, we conclude that the task set is not schedulable by EDF.

9.3 From the service intervals provided by the server, it is clear that the longest service delay occurs when a task is ready at time $t = 2$, since it has to wait for 3 units of time. Then, the service will be provided according to the supply function illustrated in Figure 13.22.

From the graph, it is easy to see that the associated bounded delay function has parameters $\alpha = 0.4$ and $\Delta = 3.5$.

9.4 By applying Equation (9.33) to the tasks we have

$$\begin{aligned} U_1 &= U_{1_0} - (U_0 - U_d)\frac{E_1}{E_0} = 0.6 - (1.4 - 1.0)\frac{1}{4} = 0.5 \\ U_2 &= U_{2_0} - (U_0 - U_d)\frac{E_2}{E_0} = 0.8 - (1.4 - 1.0)\frac{3}{4} = 0.5. \end{aligned}$$

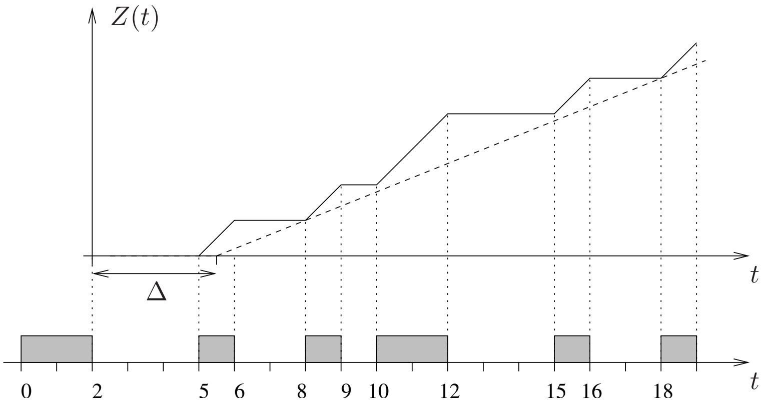

Figure 13.22 Supply function and bounded delay function of the server.

Hence,

$$T_1' = \frac{C_1}{U_1} = \frac{9}{0.5} = 18$$
$$T_2' = \frac{C_2}{U_2} = \frac{16}{0.5} = 32.$$

9.5 By applying Equation (9.42) to the tasks we have

$$T_1' = T_{1_0}\frac{U_0}{U_d} = 15 \cdot 1.4 = 21$$
$$T_2' = T_{2_0}\frac{U_0}{U_d} = 20 \cdot 1.4 = 28.$$