In summary, Lemmas 6.7—6.10 tell us that the busy interval we examine according to the general time-demand analysis method contains the most number of jobs in T_i that can possibly execute in any level-π_i busy interval. Moreover, the response time of each job in the examined busy interval is larger than the response time of the corresponding job executed in any level-π_i busy interval. This is why if we find that none of the examined jobs completes late, we know that no job in T_i will.

6.7 SUFFICIENT SCHEDULABILITY CONDITIONS FOR THE RM AND DM ALGORITHMS

When we know the periods and execution times of all the tasks in an application system, we can use the schedulability test described in the last section to determine whether the system is schedulable according to the given fixed-priority algorithm. However, before we have completed the design of the application system, some of these parameters may not be known. In fact, the design process invariably involves the trading of these parameters against each other. We may want to vary the periods and execution times of some tasks within some range of values for which the system remains feasible in order to improve some aspects of the system. For this purpose, it is desirable to have a schedulability condition similar to the ones given by Theorems 6.1 and 6.2 for the EDF and the LST algorithms. These schedulability conditions give us a flexible design guideline for the choices of the periods and execution times of tasks. The schedulable utilizations presented in this section give us similar schedulability conditions for systems scheduled according to the RM or DM algorithms. An acceptance test based on such a schedulable utilization can decide whether to accept or reject a new periodic task in constant time. In contrast, the more accurate time-demand analysis test takes $O(nq_{n,1})$ time; moreover, the accurate test is less robust because its result is sensitive to the values of periods and execution times.

6.7.1 Schedulable Utilization of the RM Algorithm for Tasks with $D_i = p_i$

Specifically, the following theorem from [LiLa] gives us a schedulable utilization of the RM algorithm. We again focus on the case when the relative deadline of every task is equal to its period. For such systems, the RM and DM algorithms are identical.

THEOREM 6.11. A system of n independent, preemptable periodic tasks with relative deadlines equal to their respective periods can be feasibly scheduled on a processor according to the RM algorithm if its total utilization U is less than or equal to

$$U_{RM}(n) = n(2^{1/n} - 1) \tag{6.10}$$

$U_{RM}(n)$ is the schedulable utilization of the RM algorithm when $D_i = p_i$ for all $1 \leq k \leq n$. Figure 6–14 shows its value as a function of the number n of tasks in the set. When n is equal to 2, $U_{RM}(n)$ is equal to 0.828. It approaches ln 2 (0.693), shown by the dashed line, for large n.

Specifically, $U(n) \leq U_{RM}(n)$ is a sufficient schedulability condition for any system of n independent, preemptable tasks that have relative deadlines equal to their respective periods to be schedulable rate-monotonically. (We use the notation $U(n)$ in place of U in our subsequent discussion whenever we want to bring the number of tasks n to our attention.) As long as

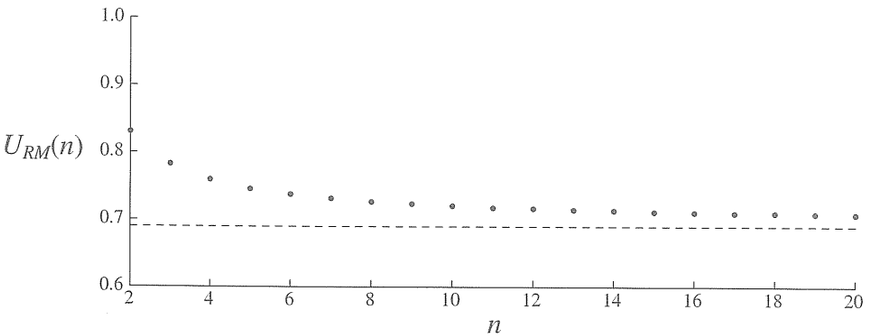

FIGURE 6–14 $U_{RM}(n)$ as a function n.

the total utilization of such a system satisfies this condition, it will never miss any deadline. In particular, we can reach this conclusion without considering the individual values of the phases, periods, and execution times.

As an example, we consider the system $\mathbf{T}$ of 5 tasks: (1.0, 0.25), (1.25, 0.1), (1.5, 0.3), (1.75, 0.07), and (2.0, 0.1). Their utilizations are 0.25, 0.08, 0.2, 0.04, and 0.05. The total utilization is 0.62, which is less than 0.743, the value of $U_{RM}(5)$. Consequently, we can conclude that we can feasibly schedule $\mathbf{T}$ rate-monotonically. Suppose that the system is later enhanced. As a result, the tasks are modified, and the resultant tasks are (0.3, 1.3, 0.1), (1.0, 1.5, 0.3), (1.75, 0.1), (2.0, 0.1), and (7.0, 2.45). Since their total utilization is 0.737, which is still less than 0.743, we know for sure that the system remains schedulable. There is no need for us to do the more complex time-demand analysis to verify this fact. On the other hand, suppose that to make the above five-task system more modular, we divide the task with period 7.0 into three smaller tasks with periods 5, 6, and 7, while keeping the total utilization of the system at 0.737. We can no longer use this condition to assure ourselves that the system is schedulable because $U_{RM}(7)$ is 0.724 and the total utilization of the system exceeds this bound.

Since $U(n) \leq U_{RM}(n)$ is not a necessary condition, a system of tasks may nevertheless be schedulable even when its total utilization exceeds the schedulable bound. For example, the total utilization of the system with the four tasks (3, 1), (5, 1.5), (7, 1.25), and (9, 0.5) is 0.85, which is larger than $U_{RM}(4) = 0.757$. Earlier in Figure 6–9, we have shown by the time-demand analysis method that this system is schedulable according to the RM algorithm.

*6.7.2 Proof of Theorem 6.11

While the schedulable utilization of the EDF algorithm given by Theorem 6.1 is intuitively obvious, the schedulable utilization $U_{RM}(n)$ given by Theorem 6.11 is not. We now present an informal proof of this theorem in order to gain some insight into why it is so.

The proof first shows that the theorem is true for the special case where the longest period p_n is less than or equal to two times the shortest period p_1. After the truth of the theorem is established for this special case, we then show that the theorem remains true when

this restriction is removed. As before, we assume that the priorities of all tasks are distinct. Here, this means that $p_1 < p_2 < \cdots < p_n$.

Proof for the Case of $p_n \leq 2p_1$. The proof for the case where $p_n \leq 2p_1$ consists of the four steps that are described below. Their goal is to find the most difficult-to-schedule system of n tasks among all possible combinations of n tasks that are difficult-to-schedule rate-monotonically. We say that a system is *difficult to schedule* if it is schedulable according to the RM algorithm, but it fully utilizes the processor for some interval of time so that any increase in the execution time or decrease in the period of some task will make the system unschedulable. The system sought here is the *most difficult* in the sense that its total utilization is the smallest among all difficult-to-schedule n-task systems. The total utilization of this system is the schedulable utilization of the RM algorithm, and any system with a total utilization smaller than this value is surely schedulable. Each of the following steps leads us closer to this system and the value of its total utilization.

Step 1: In the first step, we identify the phases of the tasks in the most difficult-to-schedule system. For this we rely on Theorem 6.5. You recall that according to that theorem, a job has its maximum possible response time if it is released at the same time as a job in every higher-priority task. The most difficult-to-schedule system must have one or more in-phase busy intervals. Therefore, in the search for this system, we only need to look for it among in-phase systems.

Step 2: In the second step, we choose a relationship among the periods and execution times and hypothesize that the parameters of the most difficult-to-schedule system of n tasks are thus related. In the next step, we will verify that this hypothesis is true. Again from Theorem 6.5, we know that in making this choice, we can confine our attention to the first period of every task. To ensure that the system is schedulable, we only need to make sure that the first job of every task completes by the end of the first period of the task. Moreover, the parameters are such that the tasks keep the processor busy once some task begins execution, say at time 0, until at least p_n, the end of the first period of the lowest priority task T_n.

The combination of n periods and execution times given by the pattern in Figure 6–15 meets these criteria. By construction, any system of n tasks whose execution times are related to their periods in this way is schedulable. It is easy to see that any increase in execution time of any task makes this system unschedulable. Hence systems whose parameters satisfy this relationship are difficult to schedule. Expressing analytically the dependencies of execution times on the periods of tasks that are given by Figure 6–15, we have

$$e_k = p_{k+1} - p_k \qquad \text{for } k = 1, 2, \ldots, n-1 \tag{6.11a}$$

Since each of the other tasks execute twice from 0 to p_n the execution time of the lowest priority task T_n is

$$e_n = p_n - 2\sum_{k=1}^{n-1} e_k \tag{6.11b}$$

Step 3: We now show that the total utilization of any difficult-to-schedule n-task system whose execution times are not related to their periods according to Eq. (6.11) is larger than or equal to the total utilization of any system whose periods and execution times are thus

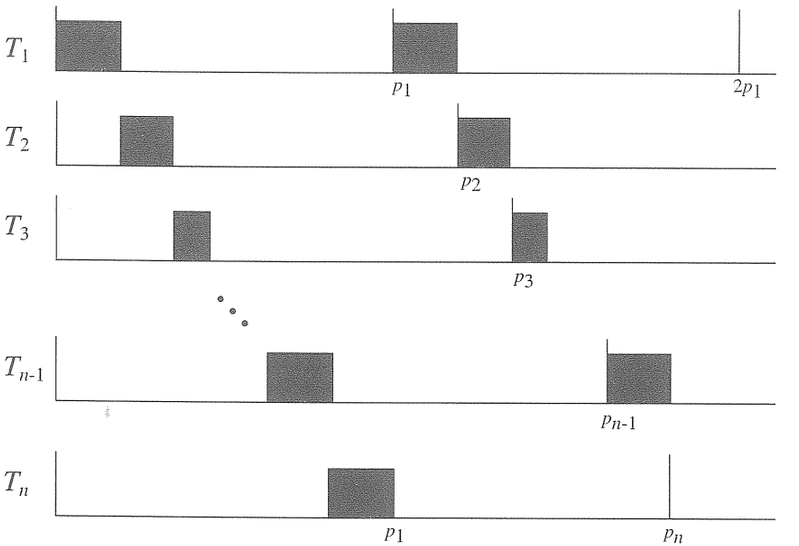

FIGURE 6–15 Relationship among parameters of difficult-to-schedule tasks.

related. Since we are looking for the difficult-to-schedule system with the least total utilization, we need not consider any system whose parameters are not thus related.

To do so, we construct new systems, whose parameters do not satisfy Eq. (6.11), from an original system whose parameters satisfy Eq. (6.11). There are two ways to do this. One way is by increasing the execution time of a higher-priority task from the value given by Eq. (6.11) by a small amount $\varepsilon > 0$. Without loss of generality, let this task be T_1. In other words, in the new system, the execution time e'_1 of T_1 is equal to

$$e'_1 = p_2 - p_1 + \varepsilon = e_1 + \varepsilon$$

For the new system to be schedulable, some other task must have a smaller execution time. From Figure 6–15, we see that the first job in every task can complete in time if we let the execution time of any other task be ε units less than the value given by Eq. (6.11a). Suppose that we choose T_k, for some $k \neq 1$, to be this task and make its new execution time equal to

$$e'_k = e_k - \varepsilon$$

The execution times of the tasks other than T_1 and T_k are still given by Eq. (6.11). The new system still keeps the processor busy in the interval $(0, p_n]$. The difference between the total utilization U' of the new system and the total utilization U of the original system is

$$U' - U = \frac{e'_1}{p_1} + \frac{e'_k}{p_k} - \frac{e_1}{p_1} - \frac{e_k}{p_k} = \frac{\varepsilon}{p_1} - \frac{\varepsilon}{p_k}$$

Since $p_1 < p_k$, this difference is positive, and the total utilization of the new system is larger. (You may want to convince yourself that we would reach the same conclusion if in the construction of the new system, we make the execution time of some task other than T_1 larger by ε units and make the execution times of one or more tasks with priorities lower than this task smaller by a total of ε units.)

Another way to construct a new difficult-to-schedule system from the original one is to let the execution time of a higher-priority task be ε units smaller than the value given by Eq. (6.11). Again, suppose that we choose T_1 to be this task, that is, its new execution time is

$$e_1'' = p_2 - p_1 - \varepsilon$$

From Figure 6–15, we see that if we do not increase the execution time of some other task, the processor will be idle for a total of 2ε units of time in $(0, p_n]$. To keep the processor busy throughout this interval and the system schedulable, we can increase the execution time of any of the other tasks by 2ε units, that is,

$$e_k'' = e_k + 2\varepsilon$$

for some $k \neq 1$. It is easy to see that with this increase accompanying the decrease in the execution time of T_1, the first job of every task in the new system can still complete by its deadline and the processor never idles from 0 to p_n. Comparing the total utilization U'' of this new system with that of the original system, we find that

$$U'' - U = \frac{2\varepsilon}{p_k} - \frac{\varepsilon}{p_1}$$

Since $p_k \leq 2p_1$ for all $k \neq 1$, this difference is never negative. (Again, we could also divide the 2ε units of time arbitrarily among the $n-1$ lower-priority tasks and get a new system with a total utilization larger than or equal to U.)

Step 4: As a result of step 3, we know that the parameters of the most difficult-to-schedule system of tasks must be related according to Eq. (6.11). To express the total utilization of a system whose parameters are given by Eq. (6.11) in terms of periods of the tasks in it, we substitute Eq. (6.11) into the sum $\sum_{k=1}^{n} e_k/p_k$ and thus obtain

$$U(n) = q_{2,1} + q_{3,2} + \cdots + q_{n,(n-1)} + \frac{2}{q_{2,1}q_{3,2}\cdots q_{n,(n-1)}} - n \tag{6.12}$$

where $q_{k,i}$, for $k > i$, is the ratio of the larger period p_k to the smaller period p_i, that is, $q_{k,i} = p_k/p_i$. In particular, the total utilization of any n-task system whose parameters are related according to Eq. (6.11) is a function of the $n-1$ *adjacent period ratios* $q_{k+1,k}$ for $k = 1, 2, \ldots, n-1$.

This equation shows that $U(n)$ is a symmetrical convex function of the adjacent period ratios. It has a unique minimum, and this minimum is the schedulable utilization $U_{RM}(n)$ of the RM algorithm. To find the minimum, we take the partial derivative of $U(n)$ with respect to each adjacent period ratio $q_{k+1,k}$ and set the derivative to 0. This gives us the following $n-1$ equation:

$$1 - \frac{2}{q_{2,1}q_{3,2}\cdots q_{(k+1),k}^{2}, \cdots q_{n,(n-1)}} = 0$$

for all $k = 1, 2, \ldots, n-1$.

Solving these equations for $q_{k+1,k}$, we find that $U(n)$ is at its minimum when all the $n-1$ adjacent period ratios $q_{k+1,k}$ are equal to $2^{1/n}$. Their product $q_{2,1}q_{3,2}\cdots q_{n,(n-1)}$ is the ratio $q_{n,1}$ of the largest period p_n to the smallest period p_1. This ratio, being equal to $2^{(n-1)/n}$, satisfies the constraint that $p_n \leq 2p_1$. Substituting $q_{k+1,k} = 2^{1/n}$ into the right-hand side of Eq. (6.12), we get the expression of $U_{RM}(n)$ given by Theorem 6.11.

For more insight, let us look at the special case where n is equal to 3. The total utilization of any difficult-to-schedule system whose parameters are related according to Eq. (6.11) is given by

$$U(3) = q_{2,1} + q_{3,2} + + \frac{2}{q_{3,2}q_{2,1}} - 3$$

$U(3)$ is a convex function of $q_{2,1}$ and $q_{3,1}$. Its minimum value occurs at the point $q_{2,1} = q_{3,2} \geq 2^{1/3}$, which is equal to 1.26. In other words, the periods of the tasks in the most difficult-to-schedule three-task system are such that $p_3 = 1.26p_2 = 1.59p_1$.

Generalization to Arbitrary Period Ratios. The ratio $q_{n,1} = p_n/p_1$ is the *period ratio* of the system. To complete the proof of Theorem 6.11, we must show that any n-task system whose total utilization is no greater than $U_{RM}(n)$ is schedulable rate-monotonically, not just systems whose period ratios are less than or equal to 2. We do so by showing that the following two facts are true.

1. Corresponding to every difficult-to-schedule n-task system whose period ratio is larger than 2 there is a difficult-to-schedule n-task system whose period ratio is less than or equal to 2.
2. The total utilization of the system with period ratio larger than 2 is larger than the total utilization of the corresponding system whose period ratio is less than or equal to 2.

Therefore, the restriction of period ratio being equal to or less than 2, which we imposed earlier in steps 1–4, leads to no loss of generality.

We show that fact 1 is true by construction. The construction starts with any difficult-to-schedule n-task system $\{T_i = (p_i, e_i)\}$ whose period ratio is larger than 2 and step-by-step transforms it into a system with a period ratio less than or equal to 2. Specifically, in each step, we find a task T_k whose period is such that $lp_k < p_n \leq (l+1)p_k$ where l is an integer equal to or larger than 2; the transformation completes when no such task can be found. In this step, we modify only this task and the task T_n with the largest period p_n. T_k is transformed into a new task whose period is equal to lp_k and whose execution time is equal to e_k. The period of the task with period p_n is unchanged, but its execution time is increased by $(l-1)e_k$. Figure 6–16 shows the original tasks and the transformed tasks. Clearly, the ratio of p_n and the period of the task transformed from T_k is less than or equal to 2, and the system thus obtained is also a difficult-to-schedule system. By repeating this step until $p_n \leq 2p_k$ for all $k \neq n$, we systematically transform the given system into one in which the ratio of p_n and the period of every other task is less than or equal to 2.

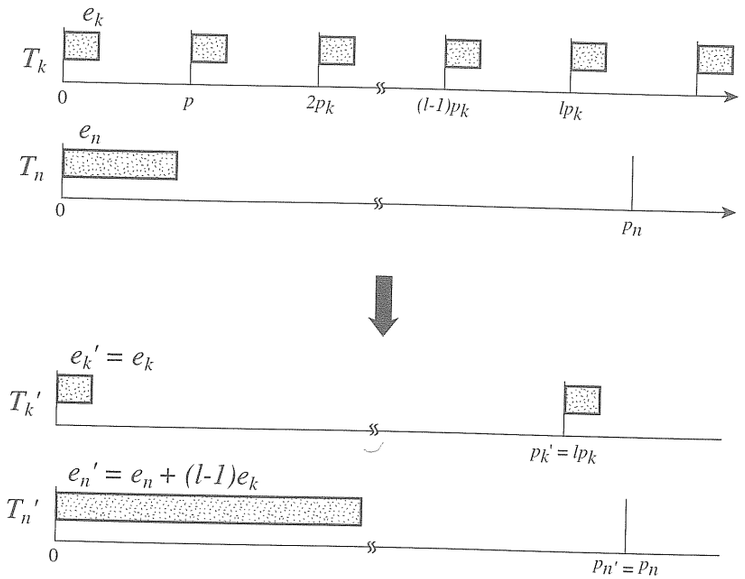

FIGURE 6–16 Transformation of two tasks.

To show fact 2 is true, we compute the difference between the total utilization of the system before the transformation of each task and the total utilization of the system after the transformation. This difference is

$$\frac{e_k}{p_k} - \frac{e_k}{lp_k} - \frac{(l-1)e_k}{p_n} = \left(\frac{1}{lp_k} - \frac{1}{p_n}\right)(l-1)e_k$$

which is larger than 0 because $lp_k < p_n$. This allows us to conclude that the system with a period ratio less than 2 obtained when the transformation completes has a smaller total utilization than the given system.

6.7.3 Schedulable Utilization of RM Algorithm as Functions of Task Parameters

When some of the task parameters are known, this information allows us to improve the schedulable utilization of the RM algorithm. We now give several schedulable utilizations that are larger than $U_{RM}(n)$ for independent, preemptive periodic tasks whose relative deadlines are equal to their respective periods. These schedulable utilizations are expressed in terms of known parameters of the tasks, for example, the utilizations of individual tasks, the number n_h of disjoint subsets each containing simply periodic tasks, and some functions of the periods of the tasks. The general schedulable utilization $U_{RM}(n)$ of the RM algorithm is the minimum value of these specific schedulable utilizations. Because they are larger than $U_{RM}(n)$, when applicable, these schedulable utilizations are more accurate criteria of schedulability. They are particularly suited for on-line acceptance tests. When checking whether a new periodic task can be scheduled with existing tasks, many of the task parameters are already known,

and computing one of these schedulable utilizations takes a constant amount of time, much less than the time required to do a time-demand analysis.

Schedulable Utilization $U_{RM}(u_1, u_2, \ldots u_n)$ as a Function of Task Utilizations. Rather than replacing the individual periods in Eq. (6.11a) by adjacent period ratios as we did earlier, we rewrite the equation as follows:

$$p_{k+1} = p_k(1 + u_k) \qquad \text{for } k = 1, 2, \ldots, n-1$$

Moreover, from Eq. (6.11b) and the fact that $p_n \leq 2p_1$, we can conclude that

$$p_n(1 + u_n) \leq 2p_1$$

Combining these two expressions, we have the following corollary.

COROLLARY 6.12. *n independent, preemptable periodic tasks with relative deadlines equal to their respective periods are schedulable rate-monotonically if their utilizations $u_1, u_2, \ldots, u_n$ satisfy the inequality*

$$(1 + u_1)(1 + u_2) \cdots (1 + u_n) \leq 2 \tag{6.13}$$

We denote the total utilization of the tasks whose utilizations satisfy the constraint Eq. (6.13) by $U_{RM}(u_1, u_2, \ldots, u_n)$.

As an example, we consider a system of two tasks T_1 and T_2. The schedulable utilization $U_{RM}(u_1, u_2)$ of the system is equal to 0.957, 0.899, 0.861, and 0.828, respectively, when the ratio u_1/U of the utilization of T_1 to the total utilization of both tasks is equal to 0.05, 0.1, 0.25, and 0.5. The minimum of $U(u_1, u_2)$ is at the point $u_1 = 0.5U$ (i.e., when $u_1 = u_2$) and is 0.828, the Liu and Layland bound for n equal to 2.

For arbitrary n, the inequality Eq. (6.13) becomes $(1 + U(n)/n)^n \leq 2$ when the utilizations of all the tasks are equal. For this combination of utilizations, the inequality Eq. (6.13) becomes the same as the Liu and Layland bound $U(n) \leq n(2^{1/n} - 1)$.

Schedulable Utilization of Subsets of Simply Periodic Tasks. We now consider a system of periodic tasks that are not simply periodic but can be partitioned into n_h subsets of simply periodic tasks. For example, we can partition the system $\mathbf{T}$ of tasks with periods 4, 7, 8, 14, 16, 28, 32, 56, and 64 into two subsets $\mathbf{Z}_1$ and $\mathbf{Z}_2$. $\mathbf{Z}_1$ contains the tasks with period 4, 8, 16, 32, and 64; and $\mathbf{Z}_2$ contains tasks with periods 7, 14, 28, and 56. Let $U(\mathbf{Z}_1)$ and $U(\mathbf{Z}_2)$ denote the total utilization of the tasks in $\mathbf{Z}_1$ and $\mathbf{Z}_2$, respectively. Kuo, *et al.* [KuMo91] have shown that if $U(\mathbf{Z}_1) + U(\mathbf{Z}_2) \leq 0.828$ [i.e., $U_{RM}(2)$], all these tasks are schedulable rate-monotonically. In contrast, if we were to treat the tasks separately, we would have to use the bound $U_{RM}(9)$, which is only 0.712.

The following theorem by Kuo, *et al.* [KuMo91] states this fact in general.

THEOREM 6.13. *If a system $\mathbf{T}$ of independent, preemptable periodic tasks, whose relative deadlines are equal to their respective periods, can be partitioned into n_h disjoint*