Real time scheduling

Real-Time CPU Scheduling

- Real-time scheduling can be classified as hard real-time systems and soft real-time systems

Soft Real time systems	Hard Real time systems
provide no guarantee as to when a critical real-time process will be scheduled.	Hard real-time systems have stricter requirements.
They guarantee only that the process will be given preference over noncritical processes	A task must be serviced by its deadline; service after the deadline has expired is the same as no service at all.

characteristics of the processes that are to be scheduled in real time

- The processes are considered periodic. That is, they require the CPU at constant intervals (periods)
- Once a periodic process has acquired the CPU, it has a fixed processing time t
- A deadline d by which it must be serviced by the CPU, and a period p
- The relationship of the processing time, the deadline, and the period can be expressed as $0 \leq t \leq d \leq p$. The rate of a periodic task is $1/p$.
- A process has to announce its deadline requirements to the scheduler. Then, using an admission-control algorithm, the scheduler either admits the process, guaranteeing that the process will complete on time, or rejects the request as impossible if it cannot guarantee that the task will be serviced by its deadline.

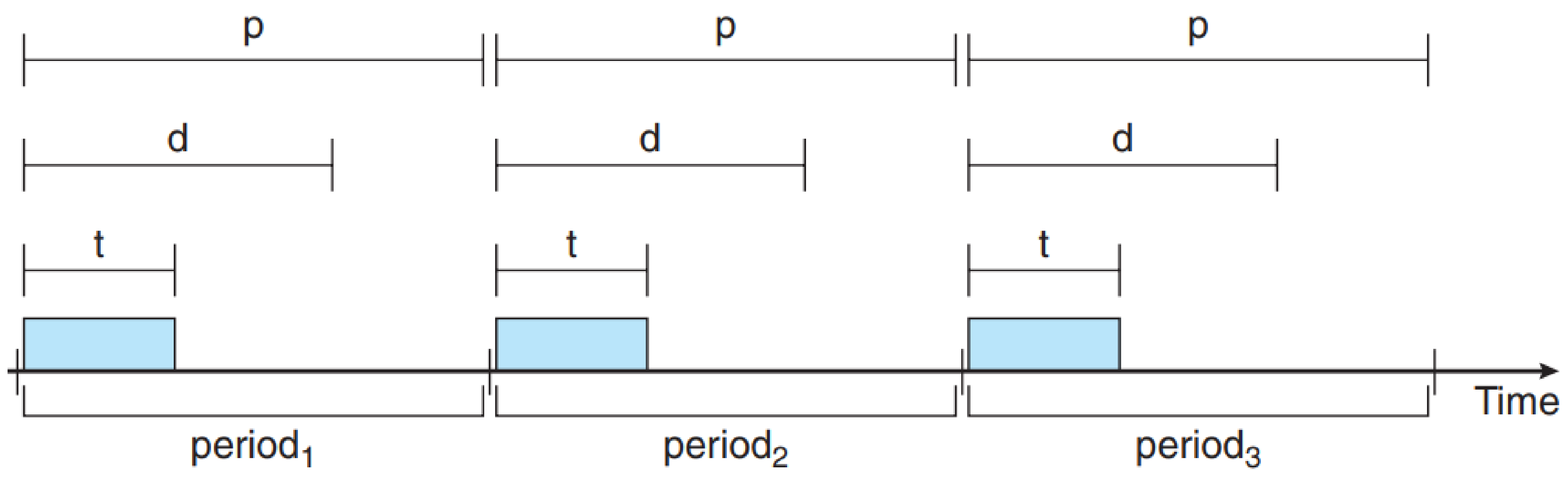

Figure 6.15 Periodic task.

Rate-Monotonic Scheduling

- Periodic tasks are scheduled using a static priority policy with preemption.
- Each periodic task is assigned a priority inversely based on its period. The shorter the period, the higher the priority; the longer the period, the lower the priority.
- Assumes that the processing time of a periodic process is the same for each CPU burst. That is, every time a process acquires the CPU, the duration of its CPU burst is the same.

processes, P1 and P2. The periods for p1 = 50 and p2 = 100. The processing times are t1 = 20 for P1 and t2 = 35 for P2. The deadline for each process requires that it complete its CPU burst by the start of its next period

- CPU utilization of a process —ti/pi
- P1=20/50 = 0.40 and that of P2 is 35/100 = 0.35, for a total CPU utilization of 75 percent. Therefore, it seems we can schedule these tasks in such a way that both meet their deadlines and still leave the CPU with available cycles. P2 a higher priority than P1
- P2 starts execution first and completes at time 35. At this point, P1 starts; it completes its CPU burst at time 55. However, the first deadline for P1 was at time 50, so the scheduler has caused P1 to miss its deadline

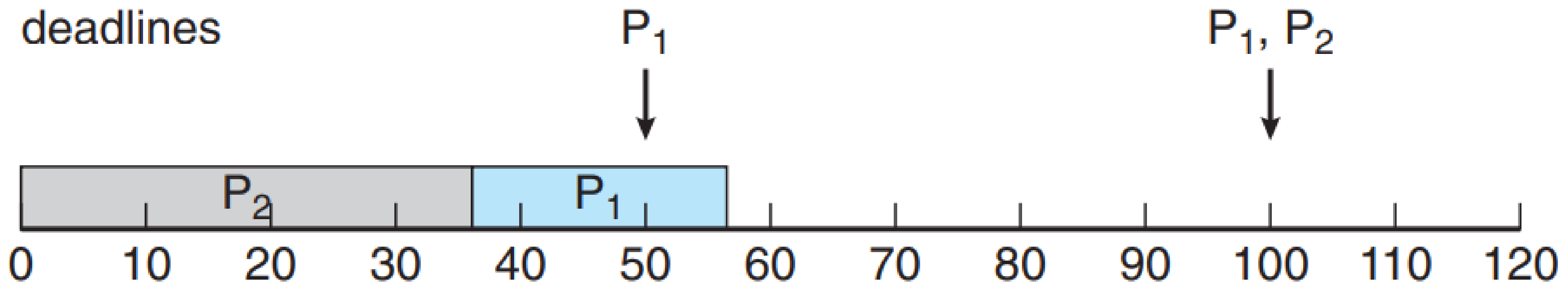

Figure 6.16 Scheduling of tasks when P_2 has a higher priority than P_1.

- Now P1 a higher priority than P2 because the period of P1 is shorter than that of P2
- P1 starts first and completes its CPU burst at time 20, thereby meeting its first deadline. P2 starts running at this point and runs until time 50. At this time, it is preempted by P1, although it still has 5 milliseconds remaining in its CPU burst. P1 completes its CPU burst at time 70, at which point the scheduler resumes P2. P2 completes its CPU burst at time 75, also meeting its first deadline. The system is idle until time 100, when P1 is scheduled again.

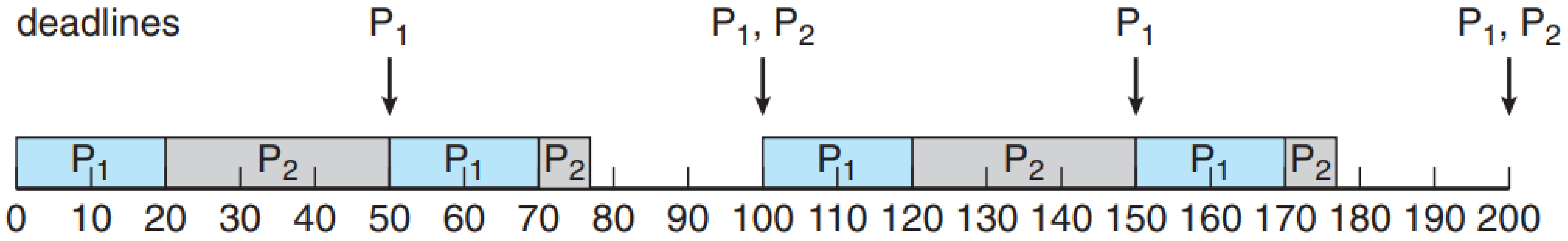

Figure 6.17 Rate-monotonic scheduling.

A set of processes that cannot be scheduled using the rate-monotonic algorithm.

- P1 has a period of p1 = 50 and a CPU burst of t1 = 25. For P2, the corresponding values are p2 = 80 and t2 = 35.
- Rate-monotonic scheduling would assign process P1 a higher priority, as it has the shorter period.
- The total CPU utilization of the two processes is (25/50)+(35/80) = 0.94, and it therefore seems logical that the two processes could be scheduled and still leave the CPU with 6 percent available time.
- Initially, P1 runs until it completes its CPU burst at time 25. Process P2 then begins running and runs until time 50, when it is preempted by P1. At this point, P2 still has 10 milliseconds remaining in its CPU burst. Process P1 runs until time 75; consequently, P2 misses the deadline for completion of its CPU burst at time 80.

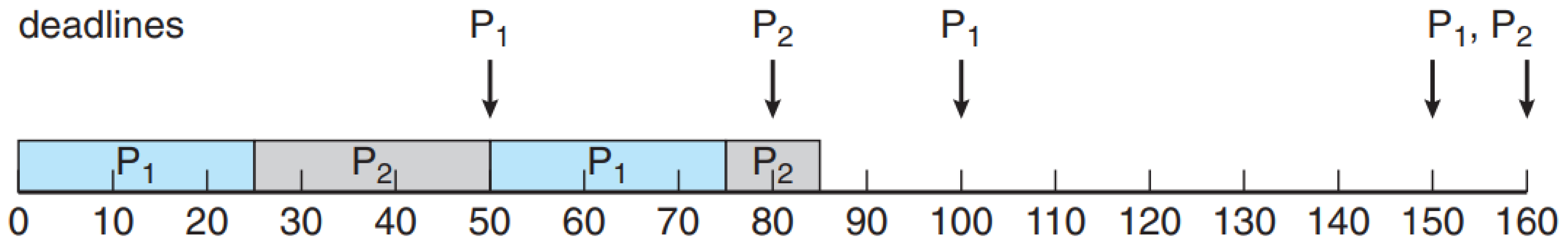

Figure 6.18 Missing deadlines with rate-monotonic scheduling.

Rate-monotonic scheduling limitation

- CPU utilization is bounded, and it is not always possible fully to maximize CPU resources.
- The worst-case CPU utilization for scheduling N processes is N(21/N − 1). With one process in the system, CPU utilization is 100 percent, approximately 69 percent as the number of processes approaches infinity. With two processes, CPU utilization is 83 percent.
- CPU utilization for the two processes scheduled in Figure 6.16 and Figure 6.17 is 75 percent; therefore, the rate-monotonic scheduling algorithm is guaranteed to schedule them so that they can meet their deadlines.
- For the two processes scheduled in Figure 6.18, combined CPU utilization is approximately 94 percent; therefore, rate-monotonic scheduling cannot guarantee that they can be scheduled so that they meet their deadlines.

Earliest-Deadline-First (EDF) Scheduling

- EDF scheduling dynamically assigns priorities according to deadline. The earlier the deadline, the higher the priority; the later the deadline, the lower the priority.
- Under the EDF policy, when a process becomes runnable, it must announce its deadline requirements to the system.
- Priorities may have to be adjusted to reflect the deadline of the newly runnable process where as in rate monotonic scheduling priorities were fixed.

EDF

- P1 has values of p1 = 50 and t1 = 25 and that P2 has values of p2 = 80 and t2 = 35

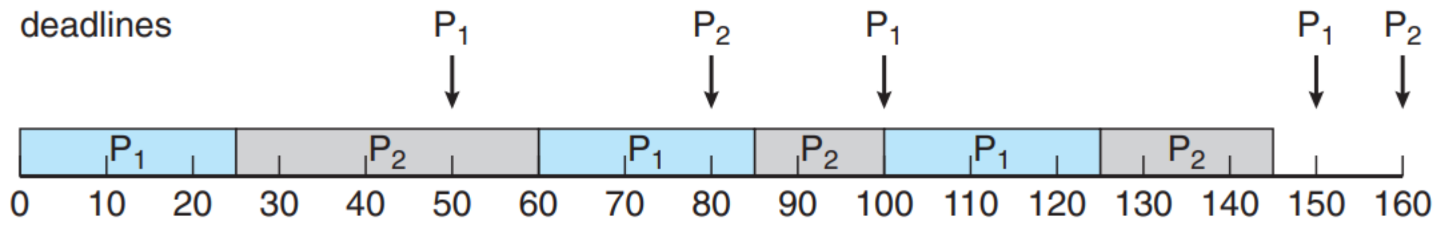

Figure 6.19 Earliest-deadline-first scheduling.

EDF

- Process P1 has the earliest deadline, so its initial priority is higher than that of process P2.
- Process P2 begins running at the end of the CPU burst for P1. However, whereas rate-monotonic scheduling allows P1 to preempt P2 at the beginning of its next period at time 50, EDF scheduling allows process P2 to continue running. P2 now has a higher priority than P1 because its next deadline (at time 80) is earlier than that of P1 (at time 100). Thus, both P1 and P2 meet their first deadlines.
- Process P1 again begins running at time 60 and completes its second CPU burst at time 85, also meeting its second deadline at time 100. P2 begins running at this point, only to be preempted by P1 at the start of its next period at time 100. P2 is preempted because P1 has an earlier deadline (time 150) than P2 (time 160). At time 125, P1 completes its CPU burst and P2 resumes execution, finishing at time 145 and meeting its deadline as well. The system is idle until time 150, when P1 is scheduled to run once again.
- EDF scheduling does not require that processes be periodic, nor must a process require a constant amount of CPU time per burst. The only requirement is that a process announce its deadline to the scheduler when it becomes runnable.