Priority-Driven Scheduling
Advantages of Priority-Driven Scheduling

• Priority-driven scheduling is easy to implement. It does not require the information on the release times and execution times of the jobs a priori.

• The run-time overhead due to maintaining a priority queue of ready jobs can be made small.
On-Line Scheduling

- The priority-driven algorithms are on-line scheduling algorithms.

- The scheduler of on-line scheduling makes each decision without knowledge about the jobs that are released in the future.

- The parameters of each job become known to the on-line scheduler only after the job is released.

- On-line scheduling is the only option in a system whose future workload is unpredictable.
Disadvantages of Priority-Driven Scheduling

• However, the timing behavior of a priority-driven system is nondeterministic.

• It is difficult to validate that all jobs scheduled in a priority-driven manner meet their deadlines when the job parameters vary.
Assumptions of Priority-Driven Scheduling

• Every job is ready for execution as soon as it is released, and can be preempted at any time.

• Scheduling decisions are made immediately upon job releases and completions.

• The context switch overhead is negligibly small compared with execution times.

• The number of priority levels is unlimited.
Scheduling on Uniprocessor Systems

• In a static system, all the tasks are partitioned into subsystems. Each subsystem is assigned to a processor, and tasks on each processor are scheduled by themselves.

• In a dynamic system, jobs ready for execution are placed in one common priority queue and dispatched to processors for execution as the processors become available.

• Most hard real-time systems built and in use to date are static. In the case when tasks in a static system are independent, we consider scheduling jobs to a single processor.
Fixed versus Dynamic Priority Algorithms

• A fixed-priority algorithm assigns the same priority to all the jobs in each task.

• A dynamic-priority algorithm assigns different priorities to the individual jobs in each task. By dynamic, we mean task-level dynamic and job level fixed.

• Most real-time scheduling algorithms of practical interests assign job-level fixed priorities.
Fixed-Priority Algorithms

• Rate-monotonic (RM) algorithm: It assigns priorities to tasks based on their periods: the shorter the period, the higher the priority. Hence, the higher the rate, the higher the priority.

• Deadline-monotonic (DM) algorithm: It assigns priorities to jobs according to their relative deadlines: the shorter the relative deadline, the higher the priority.
Dynamic-Priority Algorithms

- Jobs are assigned different priority-level in different tasks.

- Once a job is placed in the ready job queue according to the priority assigned to it, its order with respect to other jobs in the queue remains fixed.
EDF Algorithm

• Earliest-Deadline-First (EDF) algorithm assigns priorities to jobs according to their deadlines. The earlier the deadline, the higher the priority.

• This algorithm is optimal when used to schedule jobs on a processor as long as preemption is allowed and jobs do not contend for resources.

• Definition of “optimal”: can produce a feasible schedule of a set of jobs with arbitrary release times and deadlines on a processor if a feasible schedule exists.
LST Algorithm

- Least-Slack-Time-First (LST) algorithm, a.k.a. Minimum-Laxity-First (MLF) algorithm, assigns priorities to jobs based on their slacks: the smaller the slack, the higher the priority.

- At any time $t$, the slack (or laxity) of a job with deadline at $d$ is equal to $d - t$ minus the time required to complete the remaining portion of the job.

- EDF and LST algorithms are optimal only when preemption is allowed.
Overloaded Systems

• The system is overloaded if the jobs offered to the scheduler cannot be feasibly scheduled by any scheduler.

• During an overload, some jobs must be discarded in order to allow other jobs to complete in time.

• EDF and LST algorithms are optimal under the condition that the jobs are preemptive, there is only one processor, and the processor is not overloaded.

• EDF and LST algorithms performance poorly when the system is overloaded.
Maximum Schedulable Utilization

• A system of independent, preemptive tasks with relative deadlines equal to their respective periods can be feasibly scheduled on one processor if and only if its total utilization is equal to or less than 1.

• A system of simply periodic, independent, preemptive tasks whose relative deadlines are equal to or larger than their periods is schedulable on one processor according to the RM algorithm if and only if its total utilization is equal to or less than 1.
Maximum Schedulable Utilization

- A system of independent, preemptive periodic tasks that are in phase and have relative deadlines equal to or less than their respective periods can be feasibly scheduled on one processor according to the DM algorithm whenever it can be feasibly scheduled according to any fixed-priority algorithm.