## Priority-driven Scheduling of Periodic Tasks

- Focus on well-known priority-driven algorithms for scheduling periodic tasks on a processor
- Assume a restricted periodic task model
  1. The tasks are independent
  2. There are no aperiodic or sporadic tasks
- Other assumptions made:
  1. Every job is:
     - Ready for execution as soon as it is released
     - Can be preempted at any time
     - Never suspends itself
  2. Scheduling decisions are made immediately upon job releases and completions
  3. Context switch overhead is negligibly small compared with execution times of the jobs
  4. The number of priority levels is unlimited



## Priority-driven Scheduling of Periodic Tasks

- Additional assumptions
  - The period of a task means the minimum interrelease time of jobs in the task
  - A fixed number of periodic tasks
  - The addition of another task to the system requires the scheduler to perform an acceptance test – i.e. the task will be added to the system only if the new task and all other existing tasks can be feasibly scheduled
  - Focus on scheduling on uniprocessor systems
- Recall from lecture 3 …
  - Priority-driven algorithms NEVER intentionally leave any resource idle.
  - Scheduling decisions are made when events such as releases and job completions occur; hence, such algorithms are ***event-driven***
  - Locally optimal scheduling decisions are often NOT globally optimal

## Priority-driven Scheduling of Periodic Tasks

- Dynamic vs Static Systems
  - If jobs are scheduled on multiple processors, and a job can be dispatched to any of the processors, the system is ***dynamic***
  - If jobs are partitioned into subsystems, and each subsystem is bound statically to a processor, we have a ***static*** system.
  - In static systems, the scheduler for a particular processor schedules the jobs in its subsystem independently of the schedulers for other processors
  - Difficult to determine the worst-case and best-case performance of dynamic systems.
  - Most hard RT systems built to date are static
  - Results that we prove with regards to a uni-processor system are directly applicable to each subsystem in the static case



## Priority-driven Scheduling of Periodic Tasks

- Fixed-priority vs. Dynamic-priority Algorithms
  - A priority-driven scheduler is an on-line scheduler
    - It does NOT precompute a schedule of tasks/jobs
    - It assigns priorities to jobs when they are released and places them on a ready job queue in priority order
    - When preemption is allowed, a scheduling decision is made whenever a job is released or completed
    - At each scheduling decision time, the scheduler updates the ready job queue and then schedules and executes the job at the head of the queue
  - A ***fixed-priority*** algorithm assigns the same priority to all the jobs in a task.
  - A ***dynamic-priority*** algorithm assigns different priorities to the individual jobs in a task.
  - The priority of a job is usually assigned upon its release and does not change
  - Three categories of algorithms:
    - Task-level fixed-priority
    - Task-level dynamic-priority and job-level fixed-priority
    - Job-level dynamic-priority

# Priority-driven Scheduling of Periodic Tasks

## Fixed-priority Algorithms

- Rate-monotonic algorithm (RM)
  - Assigns priorities to tasks based on their periods – the shorter the period, the higher the priority
  - Since the rate is $(\text{period})^{-1}$, the higher the rate, the higher the priority
- Deadline-monotonic algorithm (DM)
  - Assigns priorities to tasks according to their relative deadlines – the shorter the relative deadline, the higher the priority
- When the relative deadline of every task is proportional to its period, the RM and DM algorithms give identical results
- When the relative deadlines are arbitrary, the DM algorithm performs better in the sense that it can sometimes produce a feasible schedule when RM fails, while RM always fails when DM fails



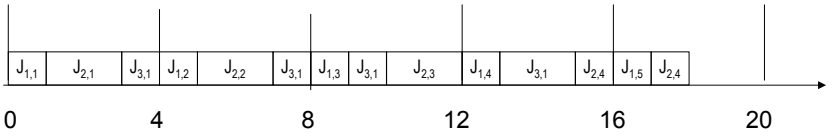
# Priority-driven Scheduling of Periodic Tasks

- Rate-monotonic example
  - $T_1 = (,4, 1)$; $T_2 = (, 5, 2)$; $T_3 = (, 20, 5)$
  - Relative priorities: $T_1 > T_2 > T_3$

| Time | Ready to run | Scheduled |
|---|---|---|
| 0 | $J_{1,1}$; $J_{2,1}$; $J_{3,1}$ | $J_{1,1}$ |
| 1 | $J_{2,1}$; $J_{3,1}$ | $J_{2,1}$ |
| 3 | $J_{3,1}$ | $J_{3,1}$ |
| 4 | $J_{1,2}$; $J_{3,1}$ | $J_{1,2}$ |
| 5 | $J_{2,2}$; $J_{3,1}$ | $J_{2,2}$ |
| 7 | $J_{3,1}$ | $J_{3,1}$ |
| 8 | $J_{1,3}$; $J_{3,1}$ | $J_{1,3}$ |

# Priority-driven Scheduling of Periodic Tasks

| Time | Ready to run | Scheduled |
|---|---|---|
| 9 | $J_{3,1}$ | $J_{3,1}$ |
| 10 | $J_{2,3}$; $J_{3,1}$ | $J_{2,3}$ |
| 12 | $J_{1,4}$; $J_{3,1}$ | $J_{1,4}$ |
| 13 | $J_{3,1}$ | $J_{3,1}$ |
| 15 | $J_{2,4}$ | $J_{2,4}$ |
| 16 | $J_{1,5}$; $J_{2,4}$ | $J_{1,5}$ |
| 17 | $J_{2,4}$ | $J_{2,4}$ |
| 18 | | |






# Priority-driven Scheduling of Periodic Tasks

Dynamic-priority algorithms

- Earliest-deadline-first (EDF)
  - The job queue is ordered by earliest deadline
- Least-slack-time-first (LST)
  - The job queue is ordered by least slack time
  - Nonstrict – scheduling decisions are made only when jobs are released or completed
  - Strict – scheduling decisions are made also whenever a queued job's slack time becomes smaller than the executing job's slack time – huge overheads, not used
- First-in-first-out (FIFO)
  - Job queue is first-in-first-out by release time
- Last-in-first-out (LIFO)
  - Job queue is last-in-first-out by release time

## Priority-driven Scheduling of Periodic Tasks

- Compare RM, EDF, LST, FIFO
- T1 = (, 2, 1); T2 = (, 5, 2.5)
- The total utilization is 1.0
- Expect some of these algorithms to lead to missed deadlines!



## Priority-driven Scheduling of Periodic Tasks

| Rate-monotonic | | |
|---|---|---|
| 0 | $J_{1,1}$; $J_{2,1}$ | $J_{1,1}$ |
| 1 | $J_{2,1}$ | $J_{2,1}$ |
| 2 | $J_{1,2}$; $J_{2,1}$ | $J_{1,2}$ |
| 3 | $J_{2,1}$ | $J_{2,1}$ |
| 4 | $J_{1,3}$; $J_{2,1}$ | $J_{1,3}$ |
| 5 | $J_{2,1}$; $J_{2,2}$ | $J_{2,1}$ |
| 5.5 | $J_{2,2}$ | $J_{2,2}$ |
| 6 | $J_{1,4}$; $J_{2,2}$ | $J_{1,4}$ |
| 7 | $J_{2,2}$ | $J_{2,2}$ |
| 8 | $J_{1,5}$; $J_{2,2}$ | $J_{1,5}$ |
| 9 | $J_{2,2}$ | $J_{2,2}$ |

| Earliest-deadline-first | | |
|---|---|---|
| 0 | $J_{1,1}[2]$; $J_{2,1}[5]$ | $J_{1,1}$ |
| 1 | $J_{2,1}[5]$ | $J_{2,1}$ |
| 2 | $J_{1,2}[4]$; $J_{2,1}[5]$ | $J_{1,2}$ |
| 3 | $J_{2,1}[5]$ | $J_{2,1}$ |
| 4 | $J_{2,1}[5]$; $J_{1,3}[6]$ | $J_{2,1}$ |
| 4.5 | $J_{1,3}[6]$ | $J_{1,3}$ |
| 5 | $J_{1,3}[6]$; $J_{2,2}[10]$ | $J_{1,3}$ |
| 5.5 | $J_{2,2}[10]$ | $J_{2,2}$ |
| 6 | $J_{1,4}[8]$; $J_{2,2}[10]$ | $J_{1,4}$ |
| 7 | $J_{2,2}[10]$ | $J_{2,2}$ |
| 8 | $J_{1,5}[10]$; $J_{2,2}[10]$ | $J_{1,5}$ |
| 9 | $J_{2,2}[10]$ | $J_{2,2}$ |

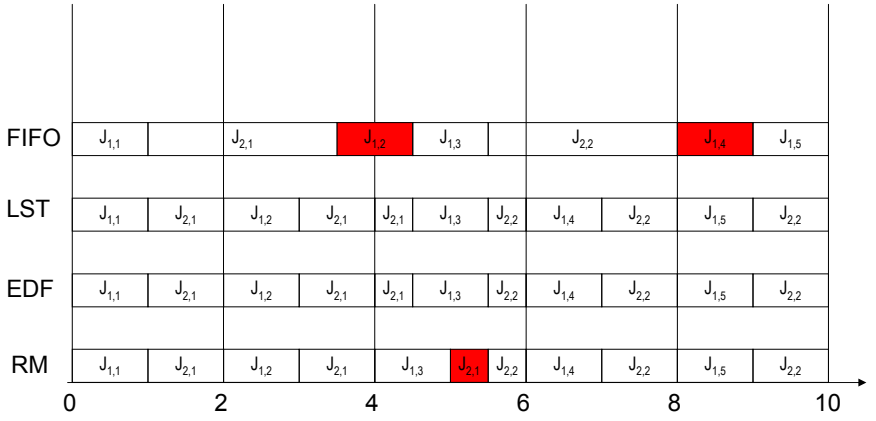
# Priority-driven Scheduling of Periodic Tasks

| Least-slack-time-first | | |
|---|---|---|
| 0 | $J_{1,1}[1]; J_{2,1}[2.5]$ | $J_{1,1}$ |
| 1 | $J_{2,1}[1.5]$ | $J_{2,1}$ |
| 2 | $J_{1,2}[1]; J_{2,1}[1.5]$ | $J_{1,2}$ |
| 3 | $J_{2,1}[0.5]$ | $J_{2,1}$ |
| 4 | $J_{2,1}[0.5]; J_{1,3}[1]$ | $J_{2,1}$ |
| 4.5 | $J_{1,3}[0.5]$ | $J_{1,3}$ |
| 5 | $J_{1,3}[0.5]; J_{2,2}[2.5]$ | $J_{1,3}$ |
| 5.5 | $J_{2,2}[2]$ | $J_{2,2}$ |
| 6 | $J_{1,4}[1]; J_{2,2}[2]$ | $J_{1,4}$ |
| 7 | $J_{2,2}[1]$ | $J_{2,2}$ |
| 8 | $J_{1,5}[1]; J_{2,2}[1]$ | $J_{1,5}$ |
| 9 | $J_{2,2}[0]$ | $J_{2,2}$ |

| First-in-first-out | | |
|---|---|---|
| 0 | $J_{1,1}; J_{2,1}$ | $J_{1,1}$ |
| 1 | $J_{2,1}$ | $J_{2,1}$ |
| 2 | $J_{2,1}; J_{1,2}$ | $J_{2,1}$ |
| 3.5 | $J_{1,2}$ | $J_{1,2}$ |
| 4 | $J_{1,2}; J_{1,3}$ | $J_{1,2}$ |
| 4.5 | $J_{1,3}$ | $J_{1,3}$ |
| 5 | $J_{1,3}; J_{2,2}$ | $J_{1,3}$ |
| 5.5 | $J_{2,2}$ | $J_{2,2}$ |
| 6 | $J_{2,2}; J_{1,4}$ | $J_{2,2}$ |
| 8 | $J_{1,4}; J_{1,5}$ | $J_{1,4}$ |
| 9 | $J_{1,5}$ | $J_{1,5}$ |



# Priority-driven Scheduling of Periodic Tasks



## Priority-driven Scheduling of Periodic Tasks

- Relative merits
  - Algorithms that do not take into account the urgencies of jobs in priority assignment usually perform poorly (FIFO, LIFO)
  - Algorithms are ranked by their ability to maximize the utilization of the system in terms of meeting job deadlines – maximum value of 1 – EDF is optimal in this sense, while RM and DM are not
  - EDF continues to give high priority to jobs that have already missed their deadlines relative to a job whose deadline is in the future; therefore, EDF is not particularly suitable to systems where overload conditions are unavoidable