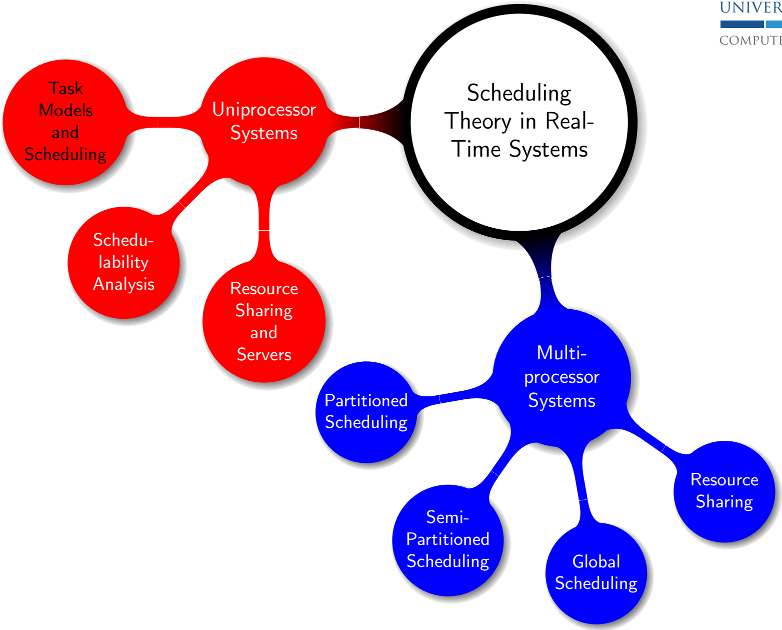
Task Models and Scheduling

Jan Reineke

Saarland University

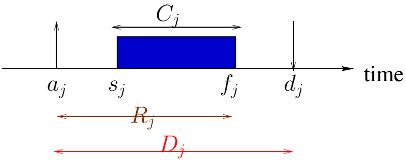
June 27^{th}, 2013

With thanks to Jian-Jia Chen at KIT!


Scheduling Theory in Real-Time Systems
Uniprocessor Systems
Task Models and Scheduling
Schedu-lability Analysis
Resource Sharing and Servers
Multi-processor Systems
Partitioned Scheduling
Semi-Partitioned Scheduling
Global Scheduling
Resource Sharing

Timing parameters of a job J_j

- Arrival time (a_j) or release time (r_j) is the time at which the job becomes ready for execution
- Computation (execution) time (C_j) is the time necessary to the processor for executing the job without interruption (= WCET).
- Absolute deadline (d_j) is the time at which the job should be completed.
- Relative deadline (D_j) is the time length between the arrival time and the absolute deadline.
- Start time (s_j) is the time at which the job starts its execution.
- Finishing time (f_j) is the time at which the job finishes its execution.
- Response time (R_j) is the time length at which the job finishes its execution after its arrival, which is $f_j - a_j$.



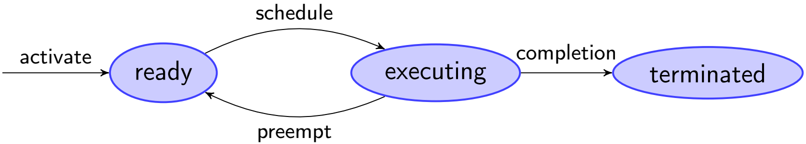
Multi-Tasking (Recap)

- The execution entities (tasks, processes, threads, etc.) are competing with each other for shared resources
- Scheduling policy is needed
  - When to schedule an entity?
  - Which entity to schedule?

Scheduling Concepts

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- **Scheduling Algorithm**: determines the order that jobs execute on the processor
- Jobs (a simplified version) may be in one of three states:

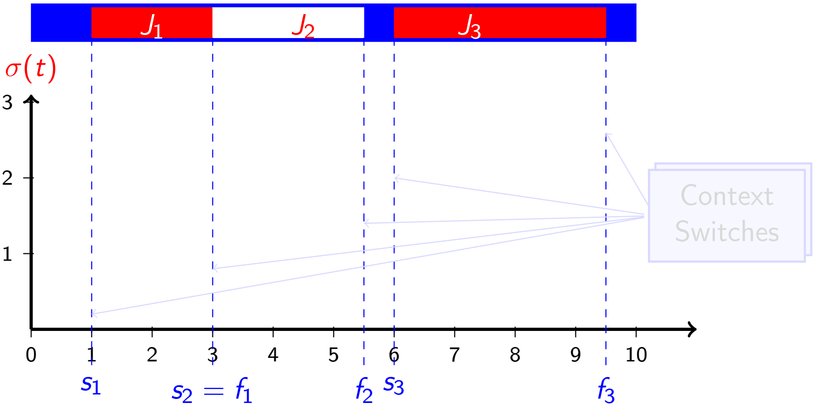


Schedules for a set of jobs $\{J_1, J_2, \ldots, J_N\}$

- A schedule is an assignment of jobs to the processor, such that each job is executed until completion.
- A schedule can be defined as an integer step function $\sigma : \mathbb{R} \to \mathbb{N}$, where $\sigma(t) = j$ denotes job J_j is executed at time t, and $\sigma(t) = 0$ denotes the system is idle at time t.
- If $\sigma(t)$ changes its value at some time t, then the processor performs a context switch at time t.
- Non-preemptive scheduling: there is only one interval with $\sigma(t) = j$ for every J_j.
- Preemptive scheduling: there can be more than one interval with $\sigma(t) = j$.

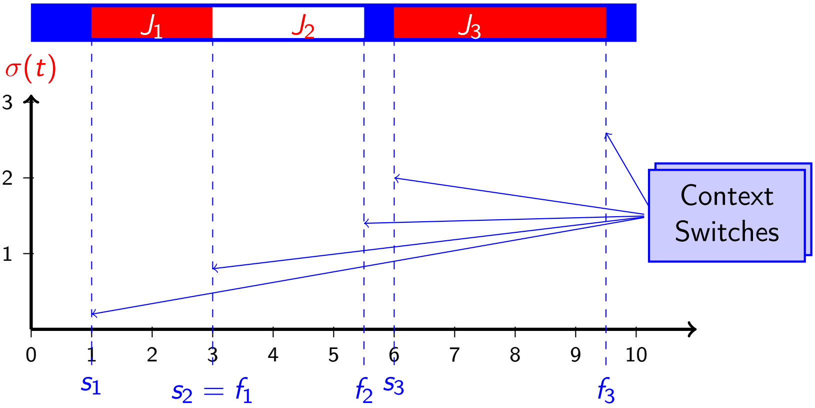
Scheduling Concept: Non-preemptive

Schedule: $\sigma : \mathbb{R} \to \mathbb{N}$ function of processor time to jobs



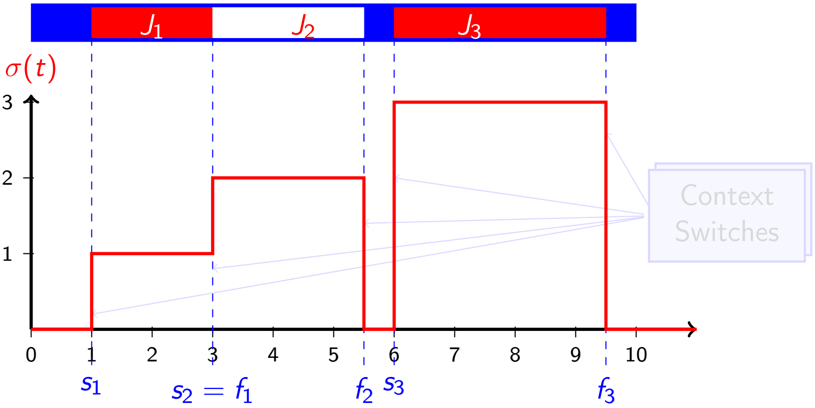
Scheduling Concept: Non-preemptive

Schedule: $\sigma : \mathbb{R} \rightarrow \mathbb{N}$ function of processor time to jobs



Scheduling Concept: Non-preemptive

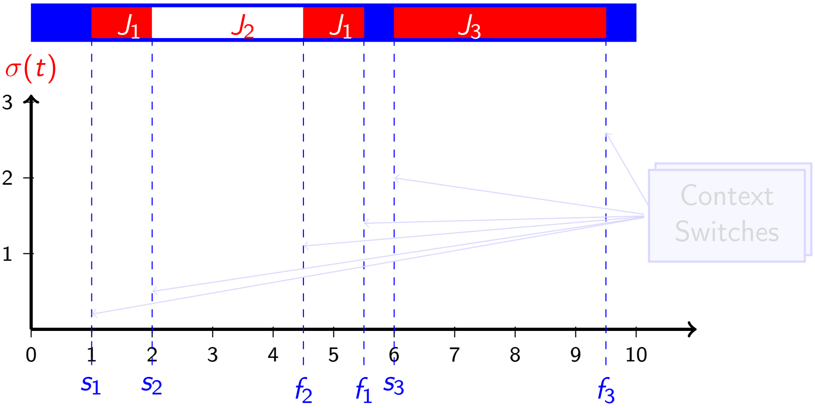
Schedule: $\sigma : \mathbb{R} \rightarrow \mathbb{N}$ function of processor time to jobs



Scheduling Concept: Preemptive

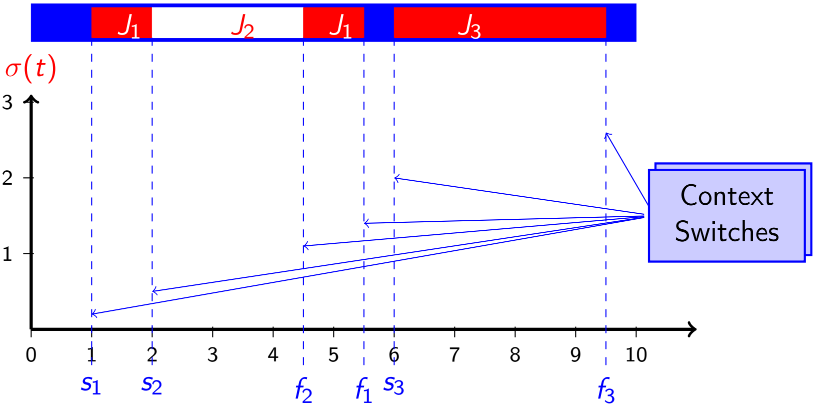
SAARLAND UNIVERSITY
COMPUTER SCIENCE

Schedule: $\sigma : \mathbb{R} \to \mathbb{N}$ function of processor time to jobs



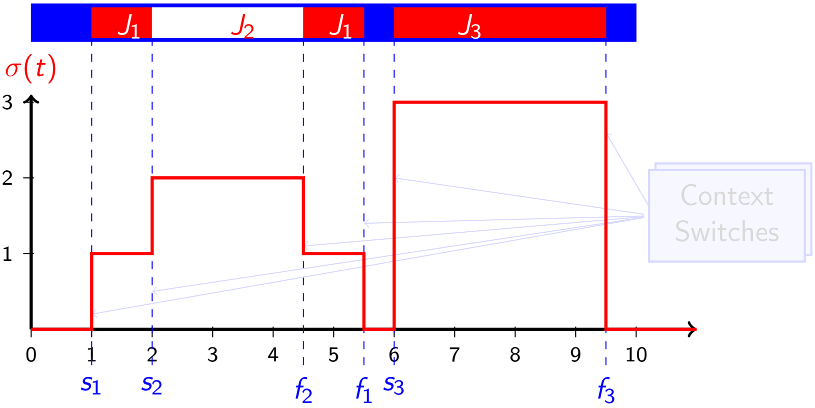
Scheduling Concept: Preemptive

Schedule: $\sigma : \mathbb{R} \rightarrow \mathbb{N}$ function of processor time to jobs



Scheduling Concept: Preemptive

Schedule: $\sigma : \mathbb{R} \rightarrow \mathbb{N}$ function of processor time to jobs



Feasibility of Schedules and Schedulability

- A schedule is feasible if all jobs can be completed according to a set of specified constraints.
- A set of jobs is schedulable if there exists a feasible schedule for the set of jobs.
- A scheduling algorithm is optimal if it always produces a feasible schedule if the given set of jobs is schedulable.

Scheduling Algorithms

- Scheduling Algorithms
  - Static Scheduling (offline, or clock-driven)
  - Dynamic Scheduling (online, or priority-driven)
    - Static-Priority Scheduling
    - Dynamic-Priority Scheduling

- Preemptive vs. Non-preemptive
- Optimal vs. Non-optimal

Scheduling Algorithms

Scheduling Algorithms

Static Scheduling
(offline, or clock-driven)

Dynamic Scheduling
(online, or priority-driven)

Static-Priority Scheduling

Dynamic-Priority Scheduling

- Preemptive vs. Non-preemptive
- Optimal vs. Non-optimal

Evaluating a Schedule

For a job J_j:

- Lateness L_j: delay of job completion with respect to its deadline.

$$L_j = f_j - d_j$$

- Tardiness E_j: the time that a job stays active after its deadline.

$$E_j = \max\{0, L_j\}$$

- Laxity (or Slack Time)(X_j): The maximum time that a job can be delayed and still meet its deadline.

$$X_j = d_j - a_j - C_j$$

Metrics of Scheduling Algorithms (for Jobs)

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Given a set $\mathbb{J}$ of n jobs, common metrics to minimize are

- Average response time:

$$\sum_{J_j \in \mathbb{J}} \frac{f_j - a_j}{|\mathbb{J}|}$$

- Makespan (total completion time):

$$\max_{J_j \in \mathbb{J}} f_j - \min_{J_j \in \mathbb{J}} a_j$$

- Total weighted response time:

$$\sum_{J_j \in \mathbb{J}} w_j(f_j - a_j)$$

- Maximum latency:

$$L_{\max} = \max_{J_j \in \mathbb{J}} (f_j - d_j)$$

- Number of late jobs:

$$N_{late} = \sum_{J_j \in \mathbb{J}} miss(J_j),$$

where $miss(J_j) = 0$ if $f_j \leq d_j$, and $miss(J_j) = 1$ otherwise.

Hard/Soft Real-Time Systems

- Hard Real-Time Systems
  - If any hard deadline is ever missed, then the system is incorrect
  - The tardiness for any job must be 0
  - **Examples**: Nuclear power plant control, flight control
- Soft Real-Time Systems
  - Deadline misses are undesired but do not have catastrophic consequences
  - Possible goals:
    - minimize the number of tardy jobs, minimize the maximum lateness, etc.
  - **Examples**: Telephone switches, multimedia applications

An Example: Shortest-Job-First (SJF)

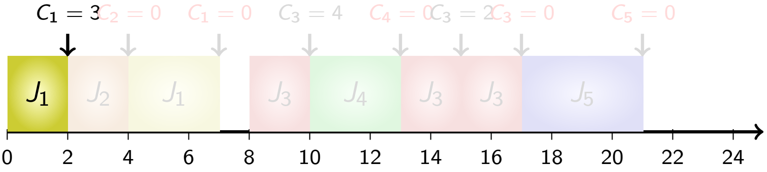
SAARLAND UNIVERSITY COMPUTER SCIENCE

- At any moment, the system executes the job with the *shortest* remaining time among the jobs in the ready queue.

	J_1	J_2	J_3	J_4	J_5
a_j	0	2	8	10	15
C_j	5	2	6	3	4
d_j	6	8	20	14	22

Exercise

What is the average response time of the above schedule?



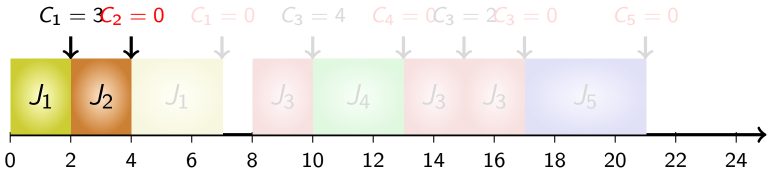
An Example: Shortest-Job-First (SJF)

- At any moment, the system executes the job with the *shortest* remaining time among the jobs in the ready queue.

	J_1	J_2	J_3	J_4	J_5
a_j	0	2	8	10	15
C_j	5	2	6	3	4
d_j	6	8	20	14	22

Exercise

What is the average response time of the above schedule?



An Example: Shortest-Job-First (SJF)

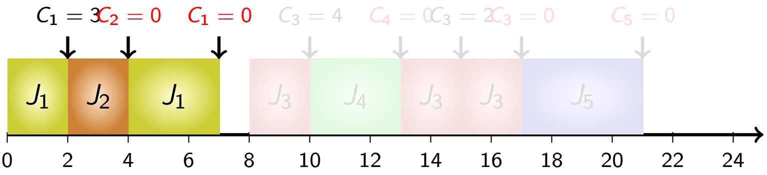
SAARLAND UNIVERSITY
COMPUTER SCIENCE

- At any moment, the system executes the job with the *shortest* remaining time among the jobs in the ready queue.

	J_1	J_2	J_3	J_4	J_5
a_j	0	2	8	10	15
C_j	5	2	6	3	4
d_j	6	8	20	14	22

Exercise

What is the average response time of the above schedule?



An Example: Shortest-Job-First (SJF)

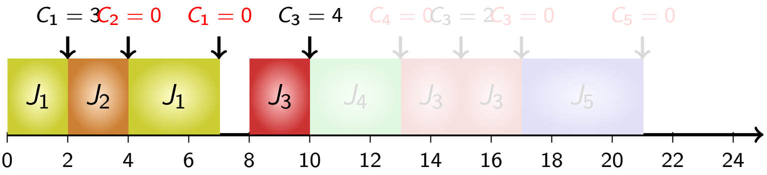
SAARLAND UNIVERSITY
COMPUTER SCIENCE

- At any moment, the system executes the job with the *shortest* remaining time among the jobs in the ready queue.

	J_1	J_2	J_3	J_4	J_5
a_j	0	2	8	10	15
C_j	5	2	6	3	4
d_j	6	8	20	14	22

Exercise

What is the average response time of the above schedule?



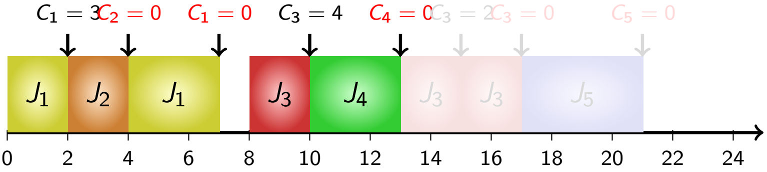
An Example: Shortest-Job-First (SJF)

- At any moment, the system executes the job with the *shortest* remaining time among the jobs in the ready queue.

	J_1	J_2	J_3	J_4	J_5
a_j	0	2	8	10	15
C_j	5	2	6	3	4
d_j	6	8	20	14	22

Exercise

What is the average response time of the above schedule?



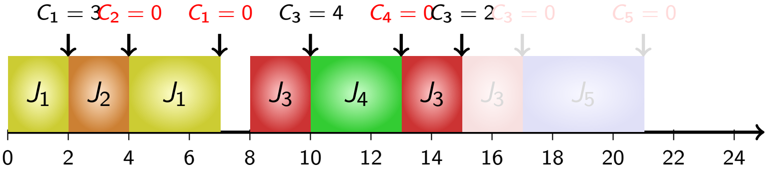
An Example: Shortest-Job-First (SJF)

- At any moment, the system executes the job with the *shortest* remaining time among the jobs in the ready queue.

	J_1	J_2	J_3	J_4	J_5
a_j	0	2	8	10	15
C_j	5	2	6	3	4
d_j	6	8	20	14	22

Exercise

What is the average response time of the above schedule?



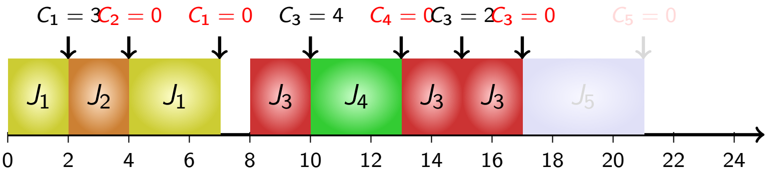
An Example: Shortest-Job-First (SJF)

- At any moment, the system executes the job with the *shortest* remaining time among the jobs in the ready queue.

	J_1	J_2	J_3	J_4	J_5
a_j	0	2	8	10	15
C_j	5	2	6	3	4
d_j	6	8	20	14	22

Exercise

What is the average response time of the above schedule?



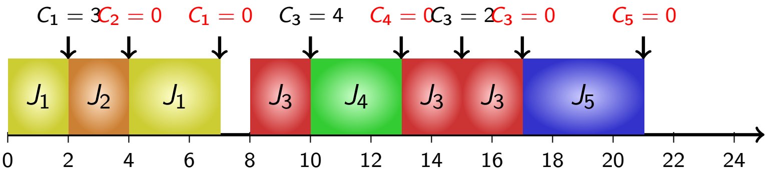
An Example: Shortest-Job-First (SJF)

- At any moment, the system executes the job with the *shortest* remaining time among the jobs in the ready queue.

	J_1	J_2	J_3	J_4	J_5
a_j	0	2	8	10	15
C_j	5	2	6	3	4
d_j	6	8	20	14	22

Exercise

What is the average response time of the above schedule?



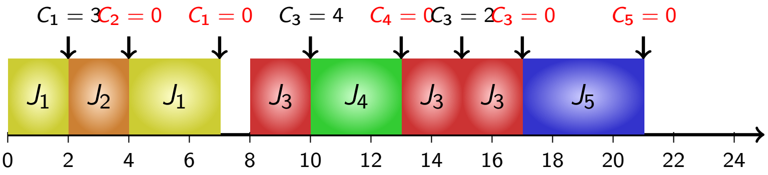
An Example: Shortest-Job-First (SJF)

- At any moment, the system executes the job with the *shortest* remaining time among the jobs in the ready queue.

	J_1	J_2	J_3	J_4	J_5
a_j	0	2	8	10	15
C_j	5	2	6	3	4
d_j	6	8	20	14	22

Exercise

What is the average response time of the above schedule?



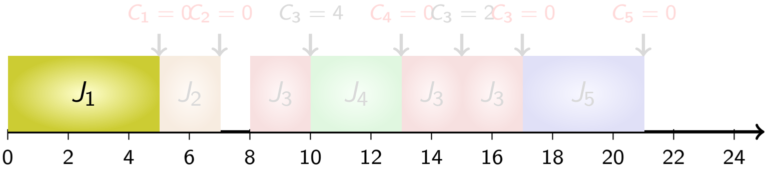
An Example: Earliest-Deadline-First (EDF)

- At any moment, the system executes the job with the *earliest absolute deadline* among the jobs in the ready queue.

	J_1	J_2	J_3	J_4	J_5
a_j	0	2	8	10	15
C_j	5	2	6	3	4
d_j	6	8	20	14	22

Exercise

What is the average response time of the above schedule?



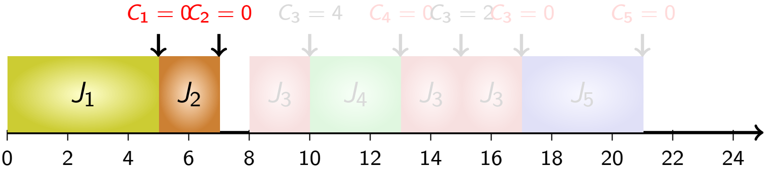
An Example: Earliest-Deadline-First (EDF)

- At any moment, the system executes the job with the *earliest absolute deadline* among the jobs in the ready queue.

	J_1	J_2	J_3	J_4	J_5
a_j	0	2	8	10	15
C_j	5	2	6	3	4
d_j	6	8	20	14	22

Exercise

What is the average response time of the above schedule?



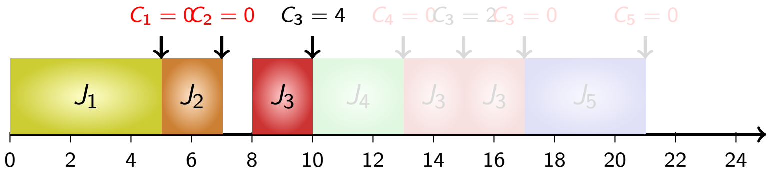
An Example: Earliest-Deadline-First (EDF)

- At any moment, the system executes the job with the *earliest absolute deadline* among the jobs in the ready queue.

	J_1	J_2	J_3	J_4	J_5
a_j	0	2	8	10	15
C_j	5	2	6	3	4
d_j	6	8	20	14	22

Exercise

What is the average response time of the above schedule?



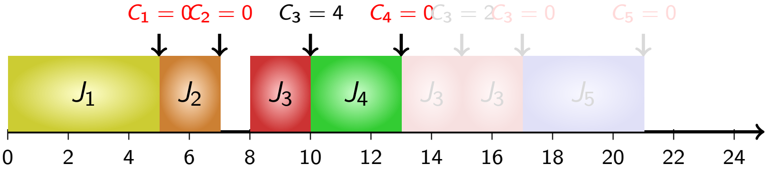
An Example: Earliest-Deadline-First (EDF)

- At any moment, the system executes the job with the *earliest absolute deadline* among the jobs in the ready queue.

	J_1	J_2	J_3	J_4	J_5
a_j	0	2	8	10	15
C_j	5	2	6	3	4
d_j	6	8	20	14	22

Exercise

What is the average response time of the above schedule?



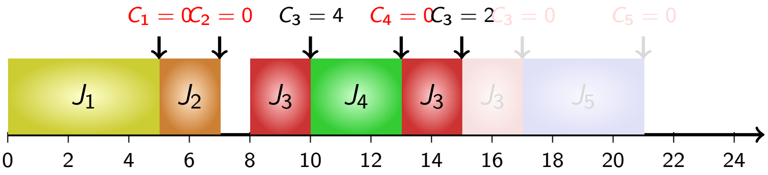
An Example: Earliest-Deadline-First (EDF)

- At any moment, the system executes the job with the *earliest absolute deadline* among the jobs in the ready queue.

	J_1	J_2	J_3	J_4	J_5
a_j	0	2	8	10	15
C_j	5	2	6	3	4
d_j	6	8	20	14	22

Exercise

What is the average response time of the above schedule?



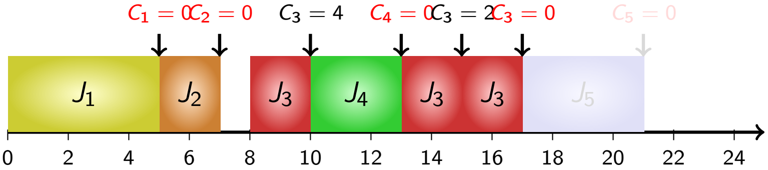
An Example: Earliest-Deadline-First (EDF)

- At any moment, the system executes the job with the *earliest absolute deadline* among the jobs in the ready queue.

	J_1	J_2	J_3	J_4	J_5
a_j	0	2	8	10	15
C_j	5	2	6	3	4
d_j	6	8	20	14	22

Exercise

What is the average response time of the above schedule?



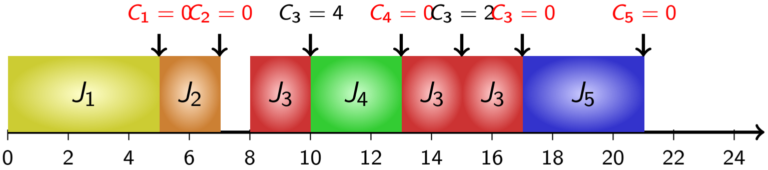
An Example: Earliest-Deadline-First (EDF)

- At any moment, the system executes the job with the *earliest absolute deadline* among the jobs in the ready queue.

	J_1	J_2	J_3	J_4	J_5
a_j	0	2	8	10	15
C_j	5	2	6	3	4
d_j	6	8	20	14	22

Exercise

What is the average response time of the above schedule?



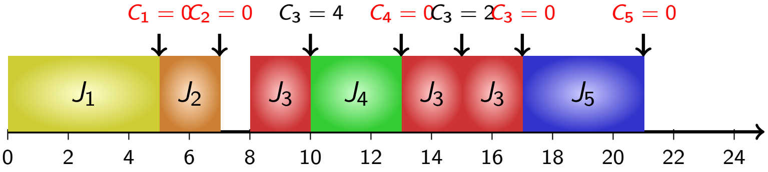
An Example: Earliest-Deadline-First (EDF)

- At any moment, the system executes the job with the *earliest absolute deadline* among the jobs in the ready queue.

	J_1	J_2	J_3	J_4	J_5
a_j	0	2	8	10	15
C_j	5	2	6	3	4
d_j	6	8	20	14	22

Exercise

What is the average response time of the above schedule?



Recurrent Task Models

- When jobs (usually with the same computation requirement) are released recurrently, these jobs can be modeled by a recurrent task
- Periodic Task τ_i:
  - A job is released exactly and periodically by a period T_i
  - A phase ϕ_i indicates when the first job is released
  - A relative deadline D_i for each job from task τ_i
  - (ϕ_i, C_i, T_i, D_i) is the specification of periodic task τ_i, where C_i is the worst-case execution time.
- Sporadic Task τ_i:
  - T_i is the minimal time between any two consecutive job releases
  - A relative deadline D_i for each job from task τ_i
  - (C_i, T_i, D_i) is the specification of sporadic task τ_i, where C_i is the worst-case execution time.
- Aperiodic Task: Identical jobs released arbitrarily.

Recurrent Task Models

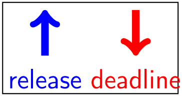
- When jobs (usually with the same computation requirement) are released recurrently, these jobs can be modeled by a recurrent task
- Periodic Task τ_i:
  - A job is released exactly and periodically by a period T_i
  - A phase ϕ_i indicates when the first job is released
  - A relative deadline D_i for each job from task τ_i
  - (ϕ_i, C_i, T_i, D_i) is the specification of periodic task τ_i, where C_i is the worst-case execution time.
- Sporadic Task τ_i:
  - T_i is the minimal time between any two consecutive job releases
  - A relative deadline D_i for each job from task τ_i
  - (C_i, T_i, D_i) is the specification of sporadic task τ_i, where C_i is the worst-case execution time.
- Aperiodic Task: Identical jobs released arbitrarily.

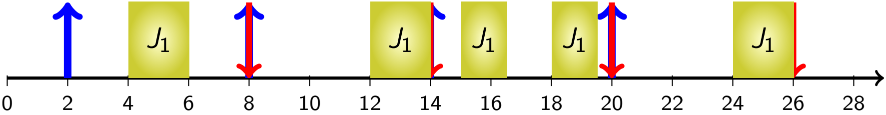
Recurrent Task Models

- When jobs (usually with the same computation requirement) are released recurrently, these jobs can be modeled by a recurrent task
- Periodic Task τ_i:
  - A job is released exactly and periodically by a period T_i
  - A phase ϕ_i indicates when the first job is released
  - A relative deadline D_i for each job from task τ_i
  - (ϕ_i, C_i, T_i, D_i) is the specification of periodic task τ_i, where C_i is the worst-case execution time.
- Sporadic Task τ_i:
  - T_i is the minimal time between any two consecutive job releases
  - A relative deadline D_i for each job from task τ_i
  - (C_i, T_i, D_i) is the specification of sporadic task τ_i, where C_i is the worst-case execution time.
- Aperiodic Task: Identical jobs released arbitrarily.

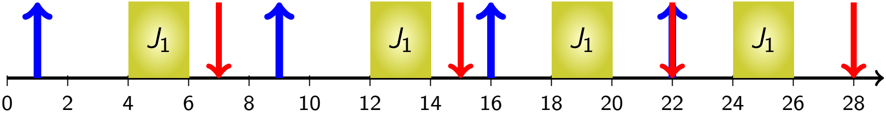
Examples of Recurrent Task Models

Periodic task: $(\phi_i, C_i, T_i, D_i) = (2, 2, 6, 6)$





Sporadic task: $(C_i, T_i, D_i) = (2, 6, 6)$



Example: Sporadic Control System

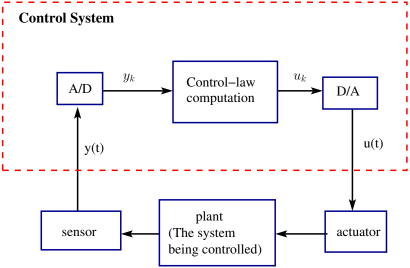
SAARLAND UNIVERSITY
COMPUTER SCIENCE

Pseudo-code for this system

while (true)

- start := get the system tick;
- perform analog-to-digital conversion to get y;
- compute control output u;
- output u and do digital-to-analog conversion;
- end := get the system tick;
- $timeToSleep := T - (end - start)$;
- sleep $timeToSleep$;

end while



Example: Periodic Control System

SAARLAND UNIVERSITY
COMPUTER SCIENCE

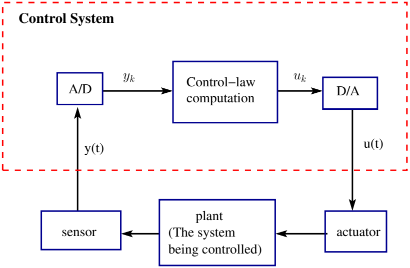
Pseudo-code for this system

set timer to interrupt periodically with period T;

at each timer interrupt
do

- perform analog-to-digital conversion to get y;
- compute control output u;
- output u and do digital-to-analog conversion;

od



Evaluating a Schedule for Tasks

For a job J_j:

- Lateness L_j: delay of job completion with respect to its deadline.

$$L_j = f_j - d_j$$

- Tardiness E_j: the time that a job stays active after its deadline.

$$E_j = \max\{0, L_j\}$$

- Laxity (or Slack Time)(X_j): The maximum time that a job can be delayed and still meet its deadline.

$$X_j = d_j - a_j - C_j$$

For a task τ_i:

- Lateness L_i: maximum latency of jobs released by task τ_i
- Tardiness E_i: maximum tardiness of jobs released by task τ_i
- Laxity X_i: $D_i - C_i$;

Relative Deadline vs Period

For a task set, we say that the task set is with

- implicit deadline when the relative deadline D_i is equal to the period T_i, i.e., $D_i = T_i$, for every task τ_i,
- constrained deadline when the relative deadline D_i is no more than the period T_i, i.e., $D_i \leq T_i$, for every task τ_i, or
- arbitrary deadline when the relative deadline D_i could be larger than the period T_i for some task τ_i.

Some Definitions for Periodic Tasks

- The jobs of task τ_i are denoted $J_{i,1}, J_{i,2}, \ldots\ldots$
- Synchronous system: Each task has a phase of 0.
- Asynchronous system: Phases are arbitrary.
- Hyperperiod: Least common multiple (LCM) of T_i.
- Task utilization of task τ_i: $u_i = \frac{C_i}{T_i}$.
- System utilization: $\sum_{\tau_i} u_i$.

Feasibility and Schedulability for Recurrent Tasks

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- A schedule is feasible if all the jobs of all tasks can be completed according to a set of specified constraints.
- A set of tasks is schedulable if there exists a feasible schedule for the set of tasks.
- A scheduling algorithm is optimal if it always produces a feasible schedule if the set of tasks is schedulable.

Graham's Scheduling Algorithm Classification

- Classification: $a|b|c$
  - a: machine environment
    (e.g., uniprocessor, multiprocessor, distributed, ...)
  - b: task and resource characteristics
    (e.g., preemptive, independent, synchronous, ...)
  - c: performance metric and objectives
    (e.g., $L_{\max}$, sum of finish times, ...)
- Examples:
  - $1|\text{non-prem}|L_{\max}$
  - $M||E_{\max}$

Earliest Due Date Algorithm

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Theorem

$1|\text{sync}|L_{\max}$: Given a set of n independent aperiodic jobs that arrive synchronously (release time is 0), any algorithm that executes tasks in order of non-decreasing deadlines is optimal with respect to minimizing the maximum lateness.
Denoted as Earliest Due Date (EDD) Algorithm [Jackson, 1955]

Proof

Let σ be the schedule for J produced by scheduling algorithm A. We can transform A to EDD schedule A' without increasing $L_{\max}$. Details are in the textbook by Buttazzo [Theorem 3.1].

Optimality of EDF

Theorem

Given a set of n independent aperiodic jobs with arbitrary arrival times, if the aperiodic task set is schedulable on a single processor then any algorithm that executes jobs with earliest deadline (among the set of active jobs) is guaranteed to meet all jobs' deadlines.

- What is the difference between EDD and EDF?
- Several proofs of optimality exist: Liu and Layland (1973), Horn (1974), and Dertouzos (1974).
- Similar to Jackson Algorithm proof of optimality, but need to account for preemption.

Monotonicity of Scheduling Algorithms

SAARLAND UNIVERSITY
COMPUTER SCIENCE

A good scheduling algorithm should be monotonic

- If a scheduling algorithm derives a feasible schedule, it should also guarantee the feasibility with
  - less execution time of a task/job,
  - less number of tasks/jobs, or
  - more number of processors/machines.

Just as a processor should not exhibit *timing anomalies*.

Why is Real-Time Scheduling Hard?

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Multiprocessor (Graham 1976)

Changing the priority order, increasing the number of processors, reducing execution times, or weakening precedence constraints can result in a deadline miss.

Many Cases

Scheduling problems in multiprocessor systems are usually $\mathcal{NP}$-Hard.

Fundamentals: Computational Complexity

- $\mathcal{N}P$-completeness of a problem Π:
  - If Π can be solved in polynomial time by a non-deterministic Turing machine, the problem is in the computational complexity class $\mathcal{N}P$.
  - Π is $\mathcal{N}P$-hard if any problem in the $\mathcal{N}P$ class can be reduced to Π in polynomial time.
  - Π is $\mathcal{N}P$-complete if Π is in $\mathcal{N}P$ and it is $\mathcal{N}P$-hard.
- The computational complexity class $\mathcal{P}$
  - The computing machines we have developed so far are deterministic Turing machines.
  - If Π can be solved in polynomial time by using a deterministic Turing machine, the problem is in the computational complexity class $\mathcal{P}$.
  - If a problem is $\mathcal{N}P$-hard, there is no efficient (polynomial-time) algorithm to derive optimal/feasible solutions unless $\mathcal{P} = \mathcal{N}P$.
    - ⋆ The question about $\mathcal{P} = \mathcal{N}P$ or $\mathcal{P} \neq \mathcal{N}P$ is an essential problem in Computer Science.

Fundamentals: Computational Complexity

- $\mathcal{N}P$-completeness of a problem Π:
  - If Π can be solved in polynomial time by a non-deterministic Turing machine, the problem is in the computational complexity class $\mathcal{N}P$.
  - Π is $\mathcal{N}P$-hard if any problem in the $\mathcal{N}P$ class can be reduced to Π in polynomial time.
  - Π is $\mathcal{N}P$-complete if Π is in $\mathcal{N}P$ and it is $\mathcal{N}P$-hard.
- The computational complexity class $\mathcal{P}$
  - The computing machines we have developed so far are deterministic Turing machines.
  - If Π can be solved in polynomial time by using a deterministic Turing machine, the problem is in the computational complexity class $\mathcal{P}$.
  - If a problem is $\mathcal{N}P$-hard, there is no efficient (polynomial-time) algorithm to derive optimal/feasible solutions unless $\mathcal{P} = \mathcal{N}P$.
    - ⋆ The question about $\mathcal{P} = \mathcal{N}P$ or $\mathcal{P} \neq \mathcal{N}P$ is an essential problem in Computer Science.

Summary

- How to characterize jobs:
  arrival time a_j, computation time C_j, absolute/relative deadline d_j/D_j
- How to characterize schedules:
  start time s_j, finishing time f_j, response time R_j
- Performance metrics for schedules:
  lateness L_j, tardiness E_j, laxity X_j
- Properties of schedules, sets of jobs, and scheduling algorithms:
  - feasibility of schedules
  - schedulability of sets of jobs and tasks
  - optimality of scheduling algorithms
- Recurrent task models:
  periodic, sporadic, aperiodic, synchronous vs asynchronous
- Scheduling algorithms:
  - Shortest-Job-First (SJF)
  - Earliest-Due-Date (EDD)
  - Earliest-Deadline-First (EDF)

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Appendix

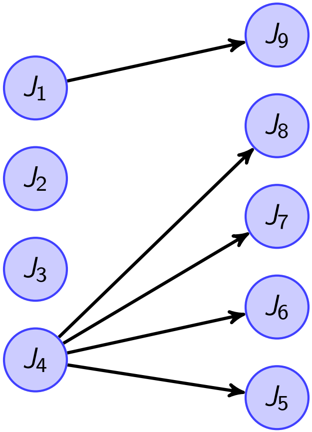
Some Examples for Multiprocessor Scheduling

Why is Real-Time Scheduling Hard? Multiprocessor Anomalies

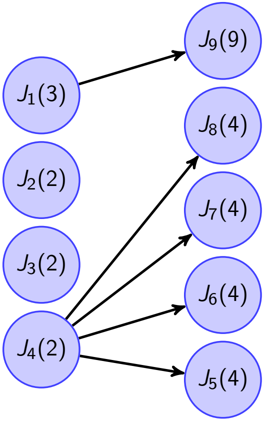
- Partitioned scheduling (Each task/job is on a processor)
  - As most partitioning algorithms are not optimal, a system might become infeasible with
    - Less execution time of a task/job
    - Less number of tasks/jobs
    - More number of processors/machines
- Global scheduling
  - As most priority-assignment algorithms are not optimal, a system might become infeasible with
    - Less execution time of a task/job
    - Less number of tasks/jobs
    - More number of processors/machines

Precedence Constraints

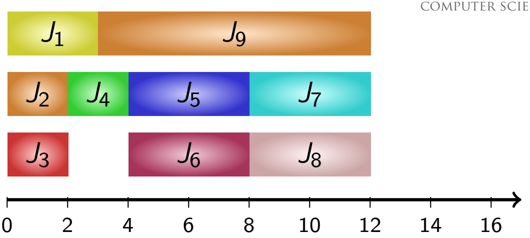
Jobs (and tasks) may have to execute in a pre-specified order.



Multiprocessor Anomaly: Case 1

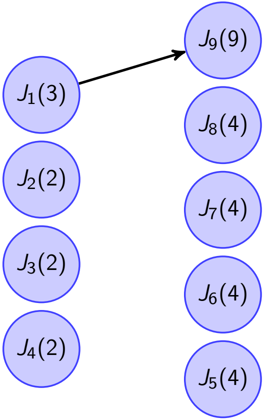


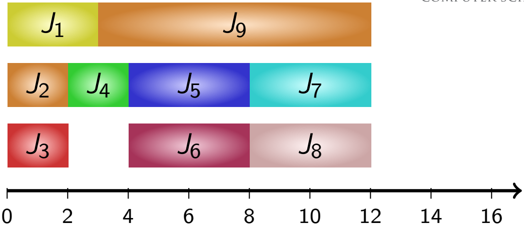
On 3 processors



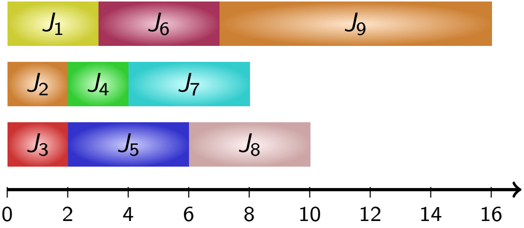
Removing the precedence constraints on J_4...

Multiprocessor Anomaly: Case 1



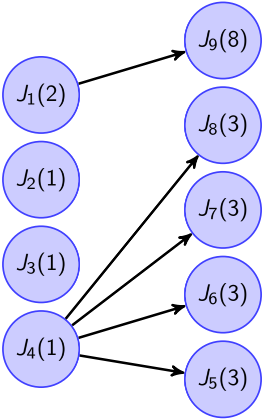


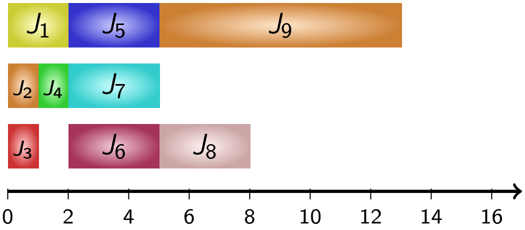
Removing the precedence constraints on J_4...



On 3 processors

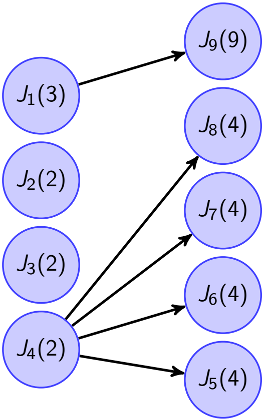
Multiprocessor Anomaly: Case 2



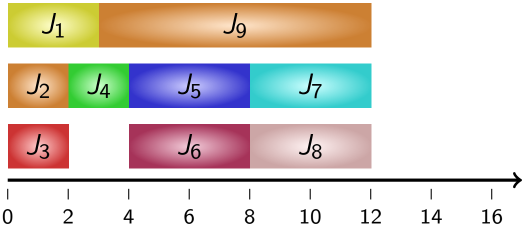


Reduce the execution time by 1, and schedule on 3 processors

Multiprocessor Anomaly: Case 3

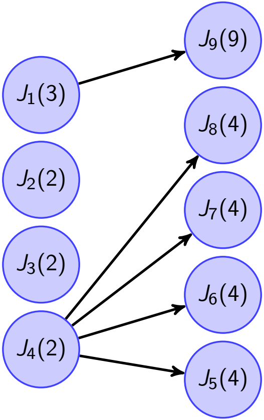


On 4 processors

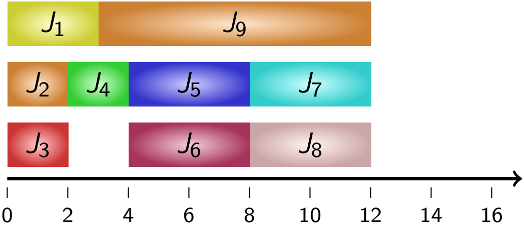


Use 4 processors

Multiprocessor Anomaly: Case 3



On 4 processors



Use 4 processors