Real-Time Systems

Modeling Real-Time Systems

Hermann Härtig

WS 2018/19

TECHNISCHE UNIVERSITÄT DRESDEN

Models

purpose of models

- describe: certain properties
- derive: knowledge about (same or other) properties (using tools)
- neglect: details not important for property

real-time systems

- properties: timing behavior + system structure
- derive: can timing constraints be met?
- neglect: …

Models in Engineering Disciplines …

- … look at quantitative properties
- very common in (real) engineering disciplines
- yet rare in computer science

Ingredients:

- Load (or Objective)
- Resources
- Mapping

e.g. for building bridges:

materials, weight of cars, velocity of wind, structures

Resources in Real-Time Systems

- Active Resources: “Processors”
  - CPUs, networks, disks, …
  - Property: speed, bandwidth, …
  - Need certain time to achieve objective
- Passive Resources
  - Memory, locks (e.g. in database), sequence numbers, …
- Active vs. passive:
  - Speed/execution vs passive occupation
  - Passive resources often modeled in terms of additional time for active resources
  - Distinction not always clear
  - Both may or may not be reusable and preemptible

Set of Periodic Tasks

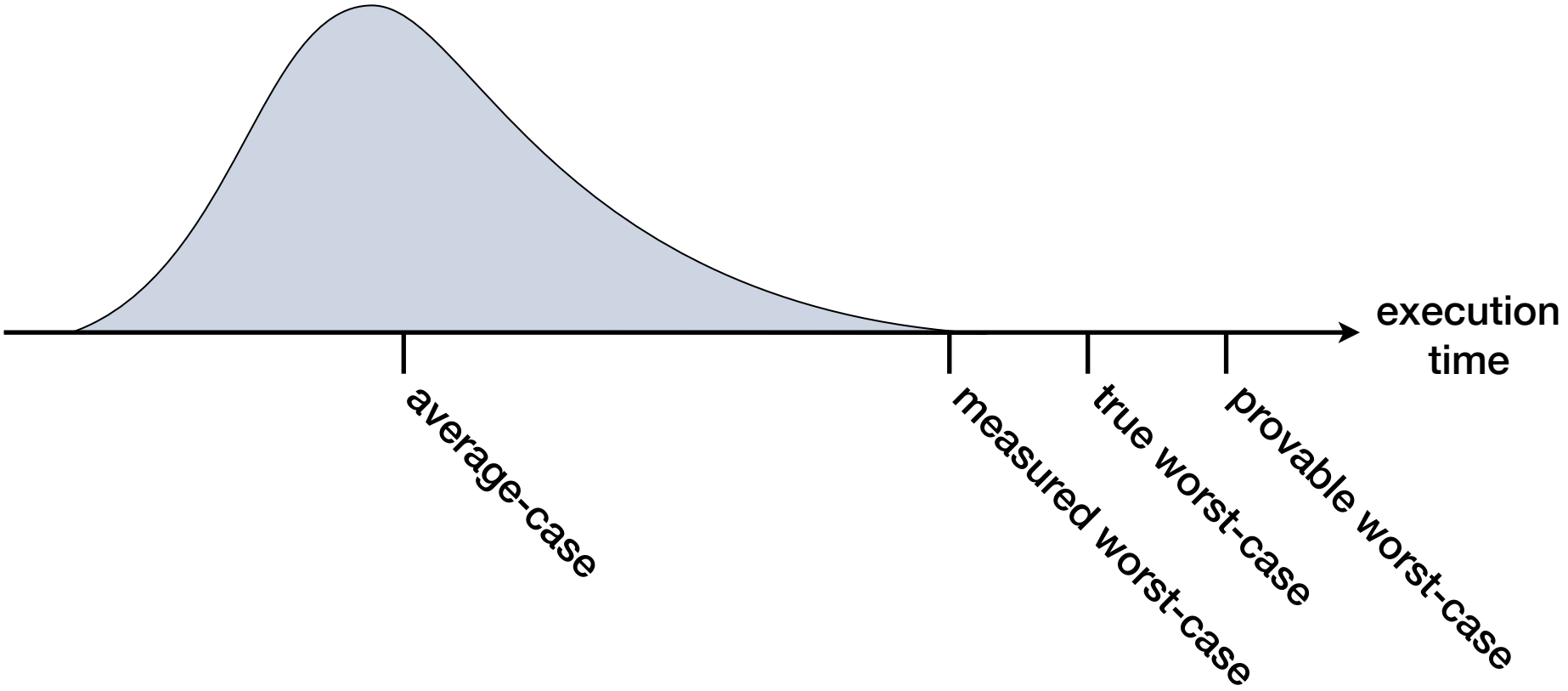
(Strictly) Periodic Tasks

T	task, consisting of a sequence of
J_i	jobs
p	period, inter-release time between consecutive jobs (const)
ϕ	release time of first job (phase)
d	deadlines of jobs (relative or absolute)
~~e_i~~	~~execution times of individual jobs~~
wcet / c	worst case execution time

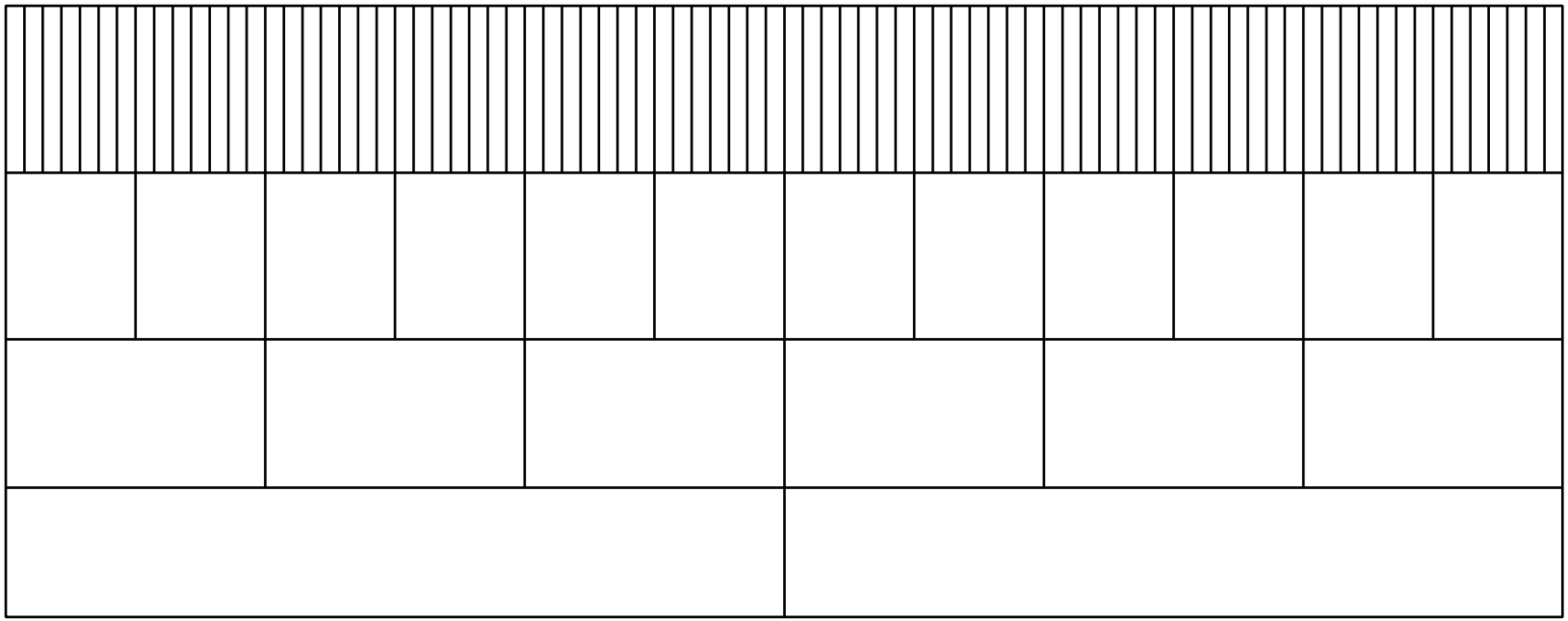
Examples: (p, wcet), d=p, ϕ=0

- T1: (2,0.9) T2: (5,2.3)
- T1: (2,1) T2: (5,4)

Worst-Case Execution Time



Period



Harmonic Periods (Simply Periodic)

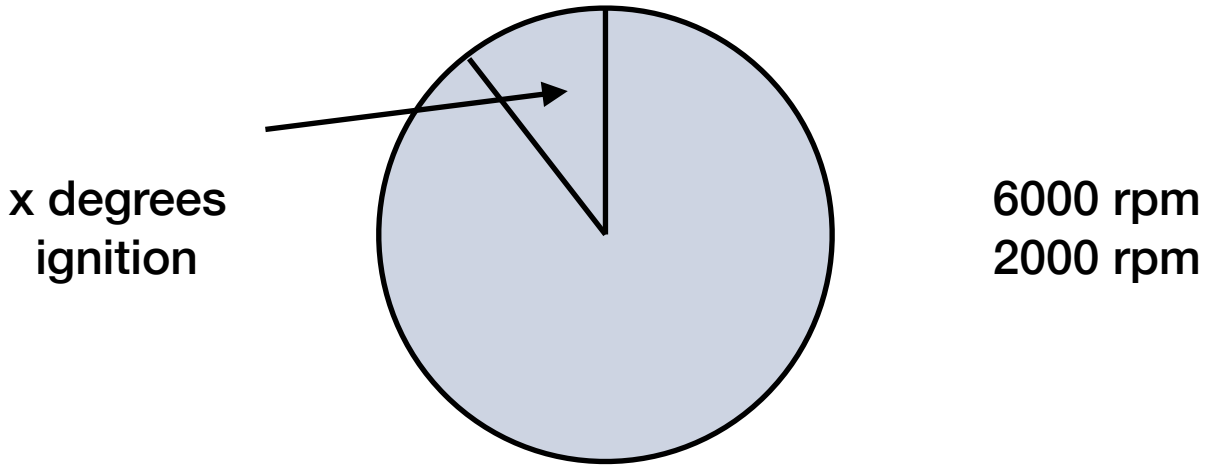
Longer periods are integer multiples of shorter periods.

Hyperperiod

Least common multiple of all periods in a task set.

A Non-Periodic Example

engine control: “period” depends on degree:



Alternatives

- Use “degree” instead of time
- Transform into “real” time
- Use sporadic task model, not periodic → next slide

Set of Sporadic/Aperiodic Tasks

RT-systems often respond to events at random points in time.

Examples:

- Degree driven systems
- Messages or alarms
  - bounded response time required
  - minimum inter-release time known
- Input sampling, e.g. from a sensor
  - higher or lower frequency, depending on system status
  - maximum frequency known in advance

Sporadic/Aperiodic Task Models

Sporadic Task Model

T	task, consisting of a sequence of
J_i	jobs
→ p	**minimum** inter-release time between consecutive jobs
→ ~~ϕ~~	~~release time of first job (phase)~~
→ d	deadlines of jobs, relative to release time of job
~~e_i~~	~~execution times of individual jobs~~
wcet / c	worst case execution time

Aperiodic tasks: p = 0

Schedule feasibility can be determined a priori for sporadic task sets, but not for aperiodic.

A warning about terminology

Jane Liu's text book:

- "Periodic" tasks: periodic or sporadic
- "Sporadic/Aperiodic": p can become arbitrarily small
  - Sporadic: hard deadlines
  - Aperiodic: soft or no deadlines

Dependencies between Tasks

Precedence

- jobs may depend on results of other jobs
- e.g. producer / consumer scenarios
- precedence graph: partial order relation on jobs

Shared Data

- jobs use some resource (e.g. critical section) that can only be used by a single job at a time

Modes and Mode Changes

A system may operate in different modes

i.e., characterized by a different set of parameter values

Examples

- Aircraft: flying, taxiing, start/land
- Smartphone: quiet, speaking, video, navigation

Mode Change

- Transition from one mode to another
- Transition itself may have real-time requirements

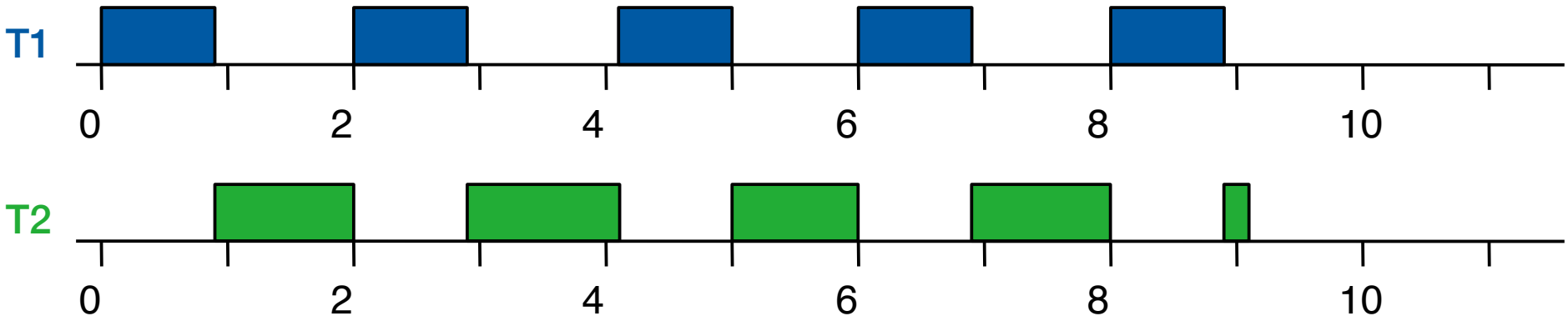
Schedule: Mapping of Load to Resources

Schedule: assignment of a set of tasks onto the available resources

- a schedule is called *valid* if
  - every job is assigned to at most one resource at any time
  - no job is scheduled before its release time
  - all constraints are satisfied:
    precedence, usage of (passive) resources
- a valid schedule is called *feasible* if
  - all deadlines are met
- a feasible schedule is called *sustainable* if
  - all deadlines are met even under some changing parameters
  - intuition: schedule must be stable against small disturbances

Example schedule

T1: (2,0.9) **T2:** (5,2.3)



Feasibility argument extends to infinity, because schedule can be repeated for each hyperperiod.

Schedulers and Admission

Scheduler (sometimes called dispatcher)

enacts/enforces/interprets a schedule

Admission

Can a new task be allowed into the current set of tasks and the current resources such that a feasible schedule exists?

- Admission heavily depends on the used scheduler

Schedulers can be based on:

- Time tables
- (static, dynamic) priorities

Priority-Based Schedulers

Jobs may have priorities, for example assigned during admission.

A priority-based scheduler implements:

If a high-priority job competes with a low-priority job,
then the high-priority job wins.

Preemption

Preemptibility of Jobs

- Preempt: stop and resume later
- Can jobs be preempted at any time to allow the execution of other jobs?
- non-preemptible jobs must not be preempted before completion (i.e. cannot release cpu or other resource)

Cost of Preemption

- Context switch operation
- WCET of jobs may depend on:
  - the number and position of preemptions within the preempted jobs
  - resource usage of other jobs or the operating system

Preemption of Resources

Can jobs be preempted while using a particular resource such that another job can use that resource.

Examples:

- CPU: preemptible
- lock: non preemptible (at least not easily)
- Common assumption: passive resources are not preemptible

Multiple Processors

Task: (period, WCET)

T1: (2,1) **T2:** (5,3)

- Migration
  - Jobs
  - Tasks
- Local/global run queues
- More details in dedicated lecture

Execution Times

- Actual execution times, depend on
  - data dependencies: if then else, compression, loops
  - hardware: caches, predictors, ...
    (see specific lecture on real-time & hardware)
- Actual execution times not known in advance, instead we use:
  - pessimistic bounds like worst case execution time
  - probabilistic distributions
- WCET-based scheduling:
  - extremely hard to find “good” WCET
  - underutilization of resources
- more models like hard, firm, soft real-time, mixed criticality

Criticality of Tasks

Some jobs may be more important than others. We want to meet the deadlines of those important jobs, possibly at the expense of less-important jobs.

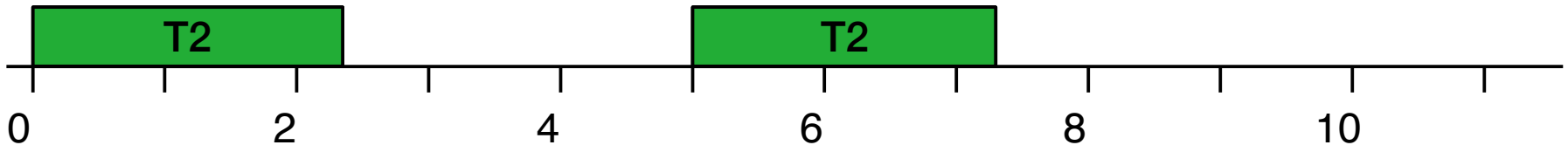
- criticality as a tunable parameter
- priority is often a result of admission, not tunable

Priority Assignment Following Criticality

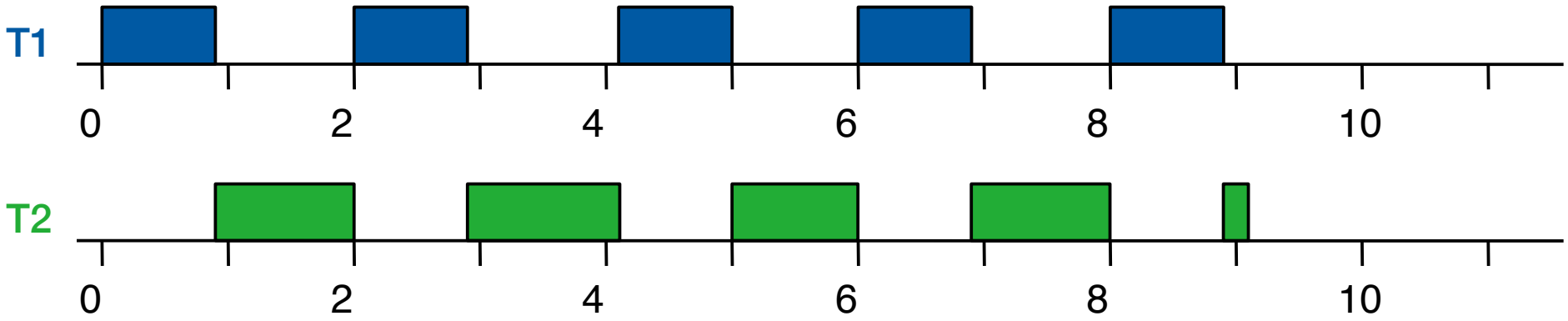
The more critical a task the higher the priority

T1: (2,0.9) **T2:** (5,2.3)

T2 more critical than **T1**



T1 misses deadline in Job 1 and 3, unnecessarily ...



Combining Hard and Soft Real-Time

- core of hard real time tasks
- shell of system with soft RT requirements

Examples

- traffic control:
  avoid crashes (hard), optimize flow (soft)
- measurement system:
  value of timestamps (hard), deliver timestamps(soft)
- quality control using robotics:
  try remove from belt (soft), otherwise shut down belt(hard)

Modeling Soft Real-Time

- m/k systems:
  maximally k of any m consecutive deadlines are missed
- Maximum tardiness t:
  deadlines are never missed by more than t
- Probabilistic

A Probabilistic Task Model

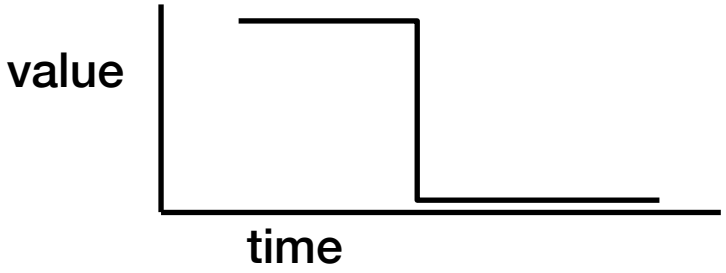
Execution times follow a probability distribution

T	task, consisting of a sequence of
J_i	jobs
p	inter-release times of jobs
d	deadlines of jobs
→ e	execution times as probability distribution
→ q	probability that deadline is met

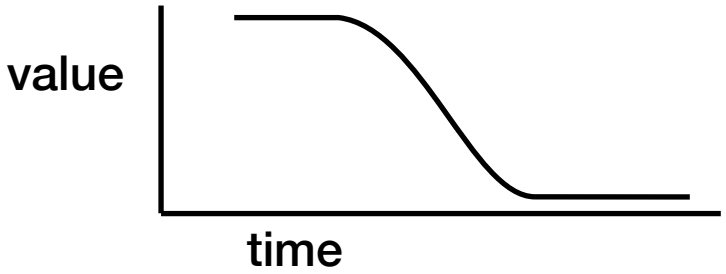
A schedule is called feasible if a q-fraction of the deadlines are met.

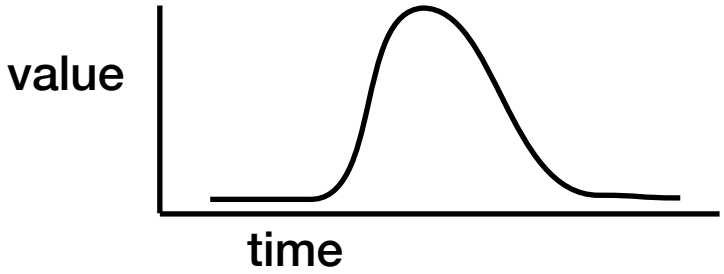
Cost (time) value functions

Hard real-time:



Others:





Imprecise Computations

- tasks: mandatory jobs followed by optional jobs
- optional jobs can be aborted and produce an approximate result
- the longer the optional jobs can run the better the result
- two versions:
  - optimize the average runtime of the optional jobs
  - make sure some can be completed

Task Pairs

T	task, consisting of
J, J_{fb}	job and its fallback
d	deadline
wcet(J_{fb})	wcet of J_{fb}

If J is not completed before d – wcet(J_{fb}), abort and execute J_{fb}.

Modeling Events: Periodic Streams

T_e	stream of
J_e	events at every
p_e	period

Jitter:	deviation from strictly periodic event stream
b_e	minimum distance
τ_e	maximal deviation from inter-release time
	may be larger than p_e

Time Driven vs. Event Driven Scheduling

Time Driven

- at design time, a feasible schedule is computed
- the schedule is stored in a table
- at certain points in time, the scheduler dispatches tasks

Event driven

- at design time, the feasibility of a set of tasks is determined depending on the scheduling algorithm
- at certain events, the scheduler computes a schedule at runtime and dispatches tasks

Conclusion

- modeling maps intended task behaviour to a formal description
- active and passive resources
- periodic, sporadic, aperiodic
- scheduling maps formal description to resources
- admission reasons about feasibility
- practical consideration: preemption
- other task models for soft real-time