Real-Time Systems

Modeling Real-Time Systems

Hermann Härtig

WS 2017/18

TECHNISCHE UNIVERSITÄT DRESDEN

Exercise

- Next week: exercise …

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

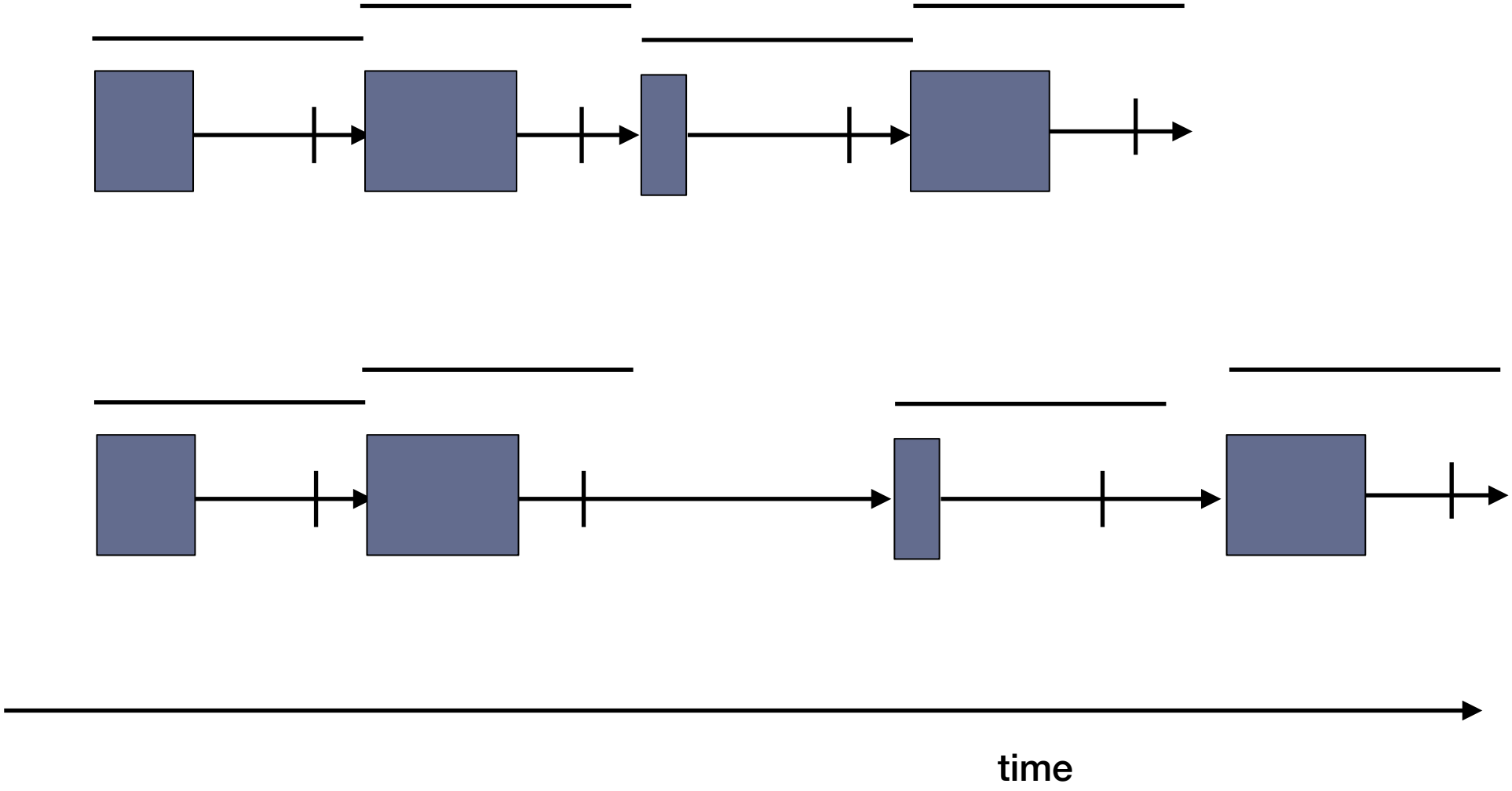
E.g. for building bridges:

materials; weight of cars, velocity of wind; structures; underground

Resources in Real-Time Systems

- Active Resources ... “Processors”
  - cpus, networks, disks, …
  - Property: Speed, Bandwidth, …
  - Need certain time to achieve objective
- Passive Resources
  - memory, (data base) locks, …
- Active vs. passive:
  - Speed, execution, … vs passively holding
  - Passive resources often modeled in term of additional time for active resources
  - Both may or may not be reusable and “preemptable”
  - Distinction not always clear

Load in RTS: Set of Recurrent Tasks



Set of Periodic Tasks

Set of (Strictly) Periodic Tasks

T task, consisting of a sequence of

Ji jobs

P period, inter-release times of jobs (constant)

ϕ release time of first job (phase)

D deadlines of jobs (relative or absolute)

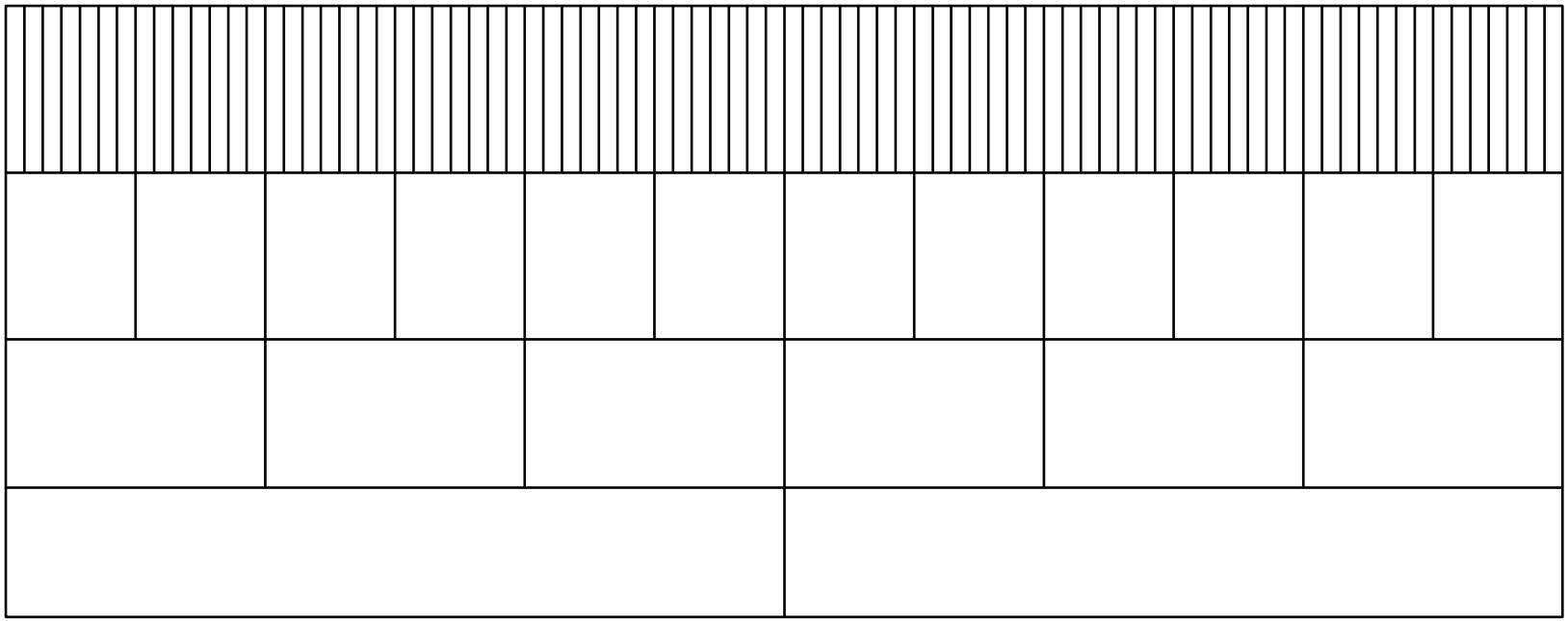
~~ei execution times of jobs, not known in advance~~

wcet 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)

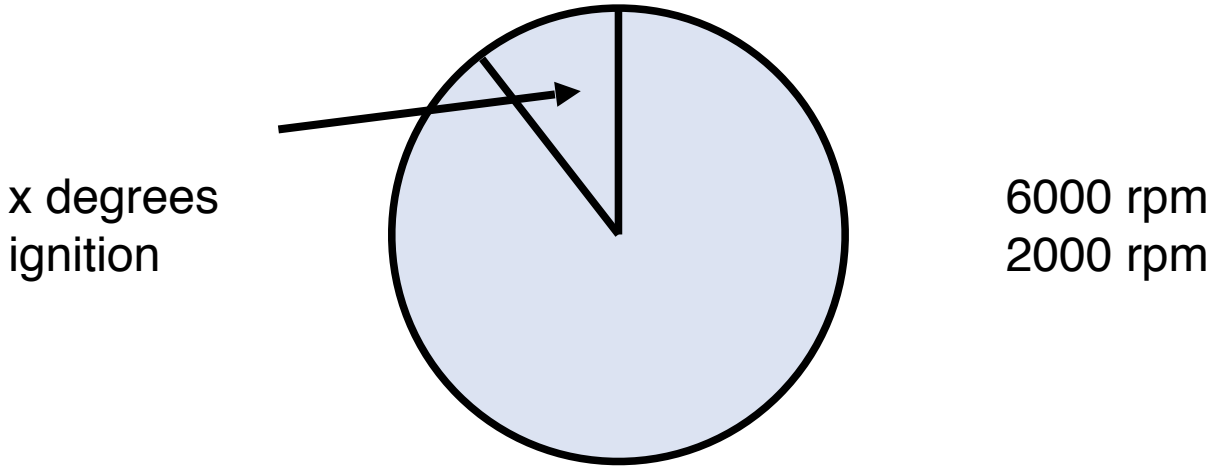
Harmonic Periods (Simply Periodic)



Periods pair-wise multiples ...

A non-periodic example

engine control: “period” depends on degree:



Remedies (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
- Sampling
  - depending on system status:
  - higher or lower frequency
  - but maximum frequency known in advance

Set of Sporadic/Aperiodic Task Models

Sporadic task model:

T task, consisting of a sequence of

Ji jobs

→ P MINIMUM inter-release time, P > 0

→ ~~ϕ release time of first job (phase)~~

D deadlines of jobs, relative to release time of job

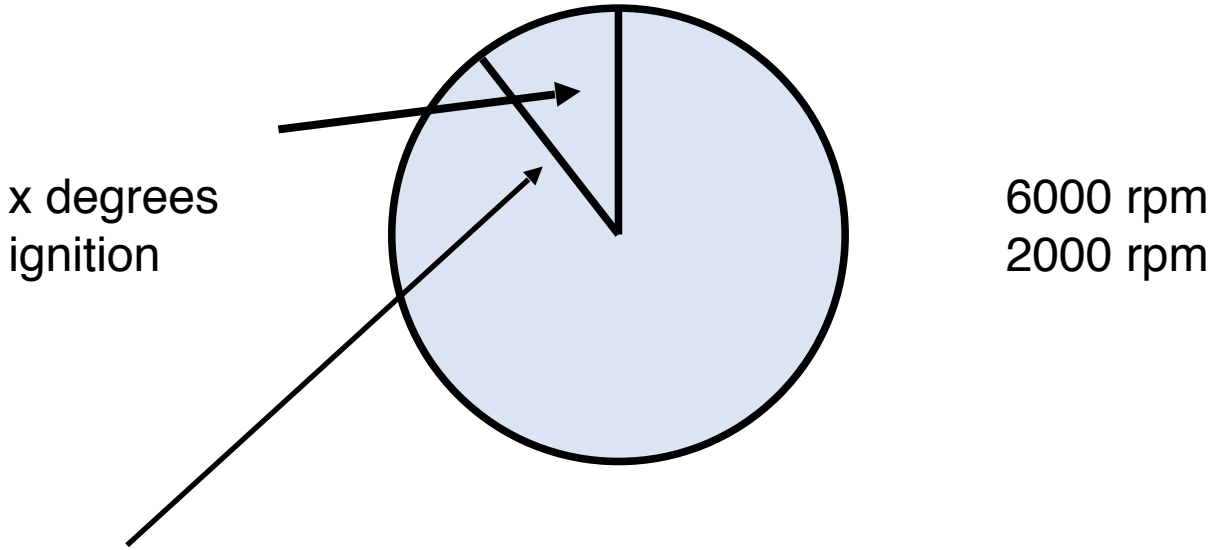
~~ei execution times of jobs, not known in advance~~

wcet worst case execution time

Aperiodic tasks: P = 0

A-priori “admission” (→ later slide) can be determined for sporadic task sets (not for aperiodic)

A non-periodic example



An event triggers the “release of next job”

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

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
- Smart phone: quiet, speaking, video, navigation

Mode change:

- Transition from one mode to another
- May have RT requirements

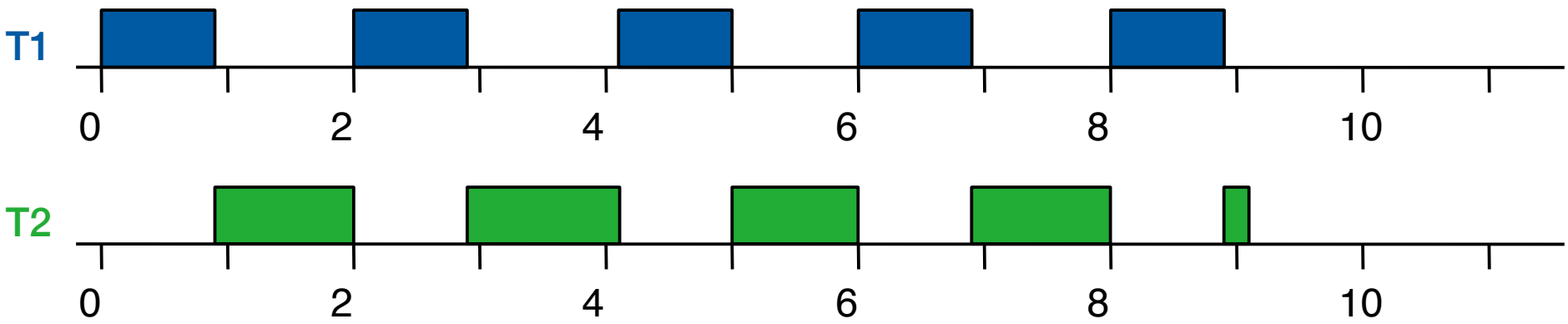
Schedule: Mapping of load to resources

schedule:
assignment of a set of tasks onto the available resources.

- a schedule is called valid if
  - every T is assigned to at most one resource at any time
    more precise: every J is assigned ...
  - 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 schedule is called sustainable if
  - all deadlines are met even under some changing parameters

Example schedule

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



Schedulers and Admission

- Scheduler (sometimes called dispatcher)
  enforces/interprets …
  a schedule
- Admission
  can a new task be allowed into the current set of tasks
  and the current resources
  such that still a feasible schedule exists ?
- Schedulers can be based on:
  - Time tables
  - (static, dynamic) priorities
- Admission heavily depends on the used scheduler

Priority-Based Schedulers

Jobs may have priorities (for example assigned during admission)

A scheduler implements:

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

Preemptability of Jobs

- preemptability of jobs
  - Preempt: stop and resume later
  - can jobs be preempted at any time
    to allow the execution of other jobs?
  - a non preemptable job must not be preempted before completion
    (i.e. cannot release cpu or other resource)
- cost of preemption (context switch)
  - the preemption operation
  - wcet of jobs may depend
    - on the number and position of preemptions within the preempted tasks
    - the operating system
    - on other tasks

Preemptability of Resources

- can jobs be preempted while using a particular resource such that another job can use that resource
- examples:
  - cpu: preemptable
  - lock: non preemptable (at least not easily)
  - Common assumption: passive resources are not preemptable

Dependencies between Tasks

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

- precedence graphs:
  - partial order relation on jobs

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

Temporal Dependency

- Jobs that need to complete within a certain time from each other

- Lip synchronization
- Events must generate response within given time window (I/O)

- “tactile internet” (new buzzword)

Multiple Processors

Task: (Period, WCET)

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

- Migration
  - Jobs
  - Tasks
- Local/global run queues

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:
  - minimum/maximum execution times
    worst case execution time (wcet)
  - probabilistic distributions
- wcet based scheduling:
  - hard to find “good” WCET
  - high underutilization of resources
- more models like hard, firm, soft real-time, mixed criticality

Criticality of Tasks

Variants:

- It may not be necessary to meet all deadlines:
  Hard Real-Time ./. Soft Real-Time
- Some jobs (their deadlines) may be more important (to meet) than others (mixed criticality)

(no clear distinction)

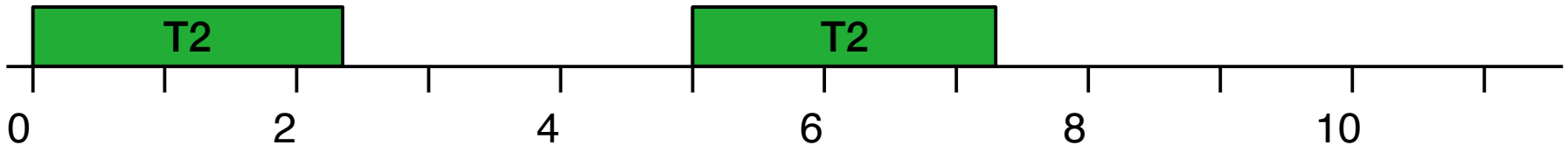
Criticality vs Priorities

Priority Assignment Following “Criticality”

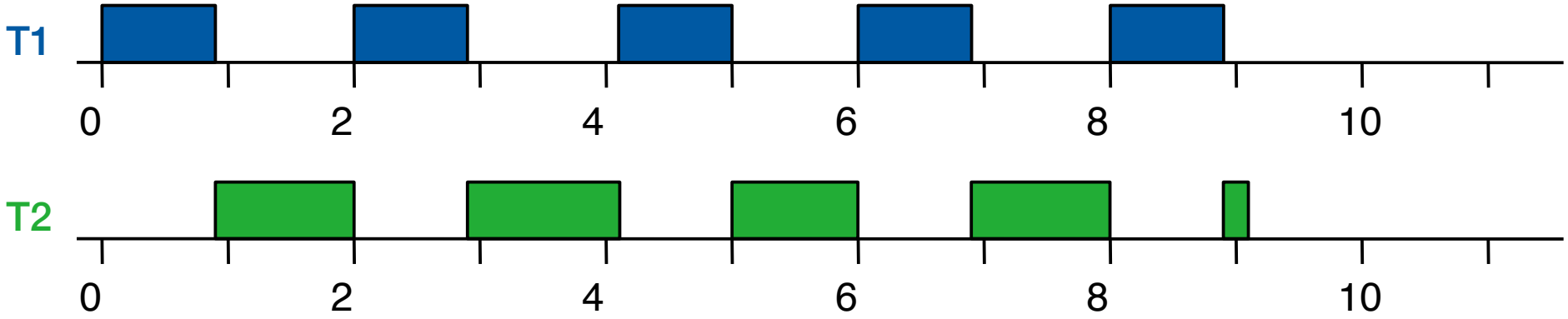
The more critical a task the higher the priority (period, wcet):

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

T2 more critical than **T1**



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



Hard-RT Core Plus Soft-RT

- Combination:
  - 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 serial deadlines are missed
- Maximum Tardiness T:
  deadlines are never missed by more than T
- Probabilistic → next slide

A probabilistic task model

Execution times follow a probability distribution

T task, consisting of a sequence of

Ji jobs

P inter-release times of jobs

D deadlines of jobs (relative or absolute)

→ e execution times as probability distribution (of tasks, jobs)

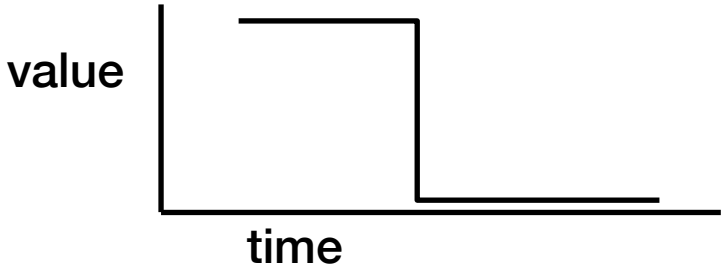
→ Q probability that deadline is met

a schedule is called feasible if Q % of the deadlines are met.

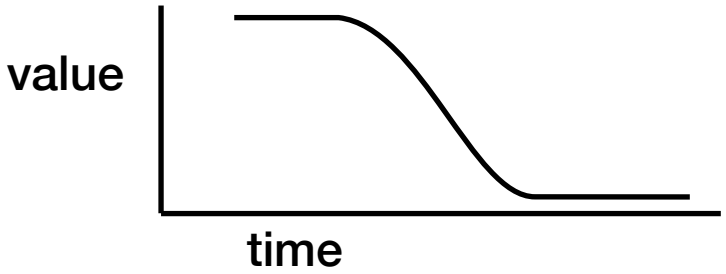
Admission: ... (SRMS, QRMS)

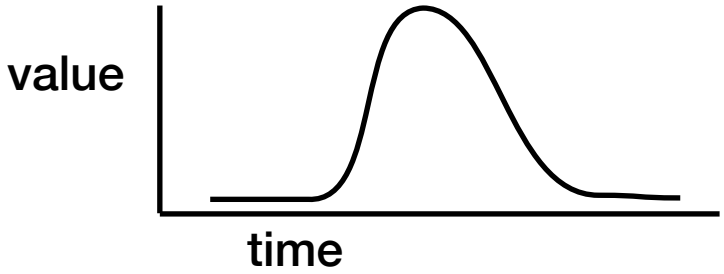
Time value functions

Hard real-time:



Others:





Imprecise Computations

- Imprecise computations:
  - tasks: mandatory jobs followed by optional jobs
  - the longer the optional jobs can run the better the result
- two versions:
  - optimize the average of the optional jobs („class a“)
  - make sure some can be completed („class b“)

Examples for Imprecise Computations

- Video presentation ... (class a)
- Tracking objects (class b)
  sometimes an object must be optimally tracked to avoid drift that may become too large for secure identification
- Some control algorithms
  (stabilization of canons, class a)

Task Pairs

T	tasks, consisting of
J	and
$J_{fallback}$	fall back
D	deadline
$wcet(J_{fallback})$	wcet of $J_{fallback}$

try J otherwise $J_{fallback}$ end

If J is not completed before $D - wcet(J_{fallback})$, abort and do $J_{fallback}$.

Approximate Computing

- Generalized terminology ???

Mixed Criticality Model (Sanjoy Baruah)

- Two or more “criticality levels”
- WCET for jobs with higher criticality are evaluated much more rigorously (pessimistic), required by higher “assurance” level
  -> several WCETs dependent on criticality level
  $WCET_{High}$ (X) > $WCET_{Low}$ (X)
- scheduling challenge:
  All Jobs of a certain criticality level should meet the deadlines if all jobs of the same or higher criticality level execute at most the WCET of that level
- More:
  http://www.artist-embedded.org/docs/Events/2010/Autrans/talks/PDF/Baruah/SkB-ARTIST2010.ppt.pdf

Example Mixed Criticality (P,WCET)

- T1
  low: (12, 2)
  medium: (12, 4)
  high: (12,11))

- T2
  low: (6, 2)
  medium: (6, 4)
- T3
  low: (24, 8)

Modeling Events: Periodic Streams of Events

T_e stream of

J_e events at every

P_e period (inter-release times)

Jitter: deviation of (truly periodic streams) events ...

B_e minimum distance

τ_e maximal deviation from inter-release time

may be larger than P_e

to compute required buffers:

S(J) associated data size of an event

Time Driven vs. Event Driven Scheduling

Time driven

- at design time, a feasible schedule is computed
- the schedule is stored in 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 and dispatches tasks

Conclusion

Before designing a (real-time) system, make sure you understand all aspects of it.