Real-Time Systems

Hermann Härtig

Modeling Real-Time Systems

Hermann Härtig

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, (data base) locks, sequence numbers, ...

Active vs. passive:

- Speed, execution, ... vs passively holding
- Distinction not always clear
- Passive resources often modeled in term of additional time for active resources
- Both may or may not be reusable and „preemptable“

Load in Real-Time System: Periodic Tasks

Set of (Strictly) Periodic Tasks

T	task, consisting of a sequence of
J_i	jobs
P	period, inter-release times of jobs (constant)
ϕ	release time of first job (phase)
D	deadlines of jobs (relative or absolute)
e_i	execution times of jobs, not known in advance
wcet	worst case execution time

→ NOTES

Load and resources:

- (worst case) execution time refers to active resources
- active resources then can be specified by “number of …“

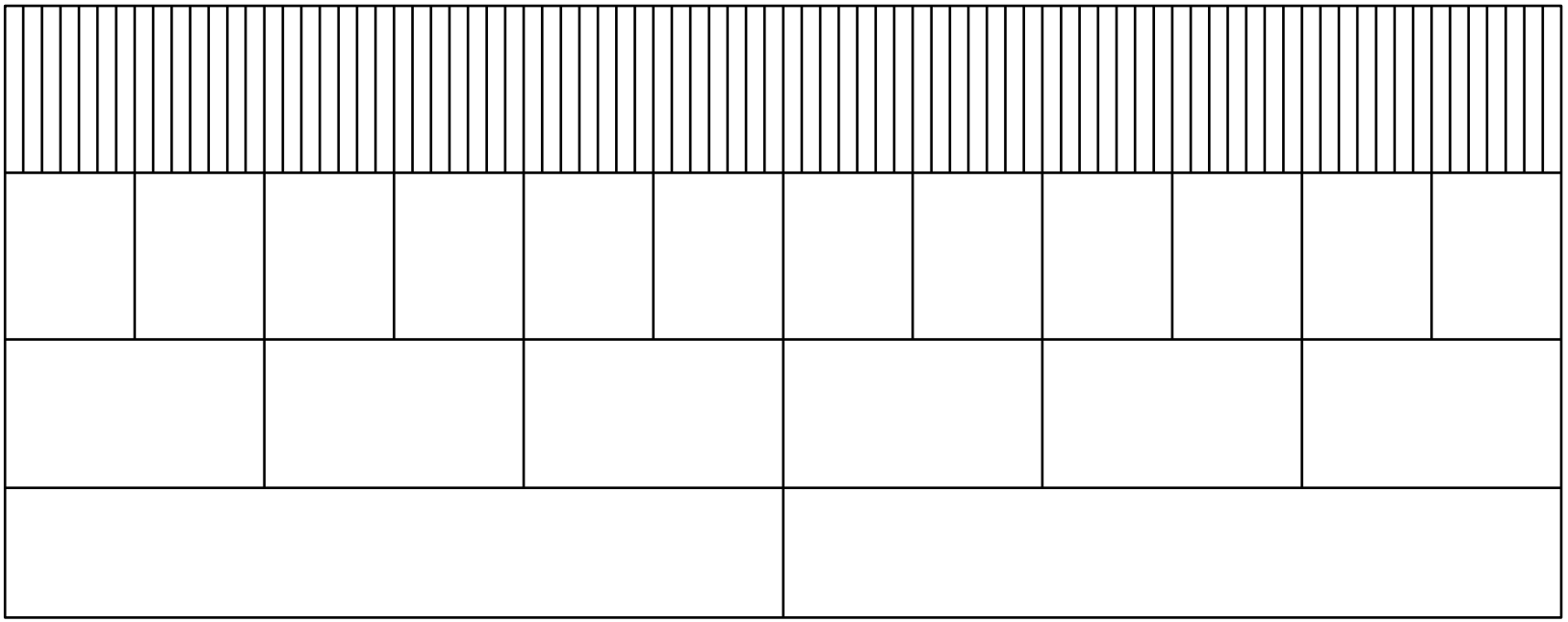
D (relative deadline) can also be greater than P:

- example: buffered audio output …

Hyperperiod:

- Least common multiple of periods

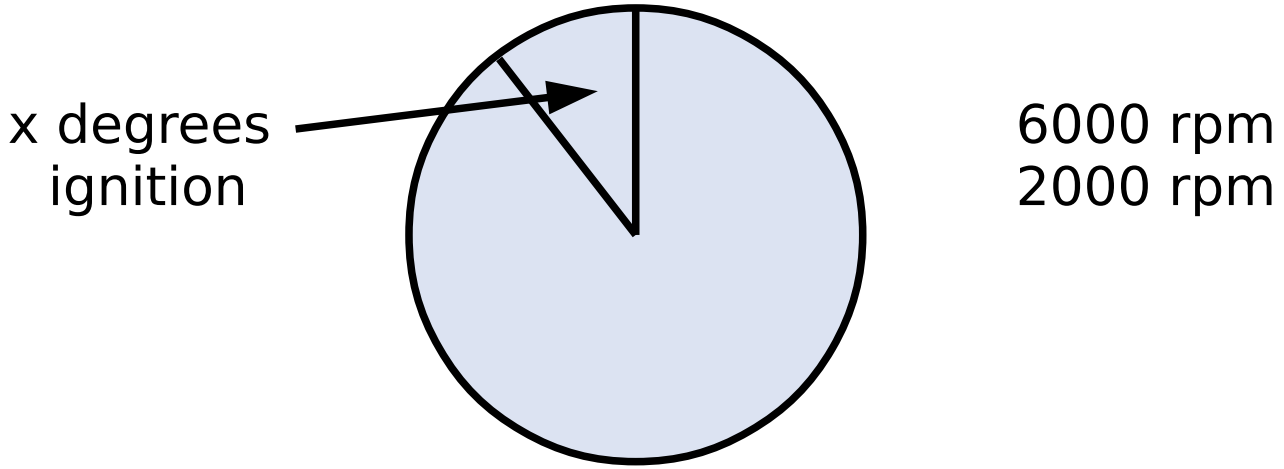
Harmonic Periods (Simply Periodic)



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

Load: Sporadic/Aperiodic Tasks

RT-systems often have to 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

Sporadic/Aperiodic Task Models

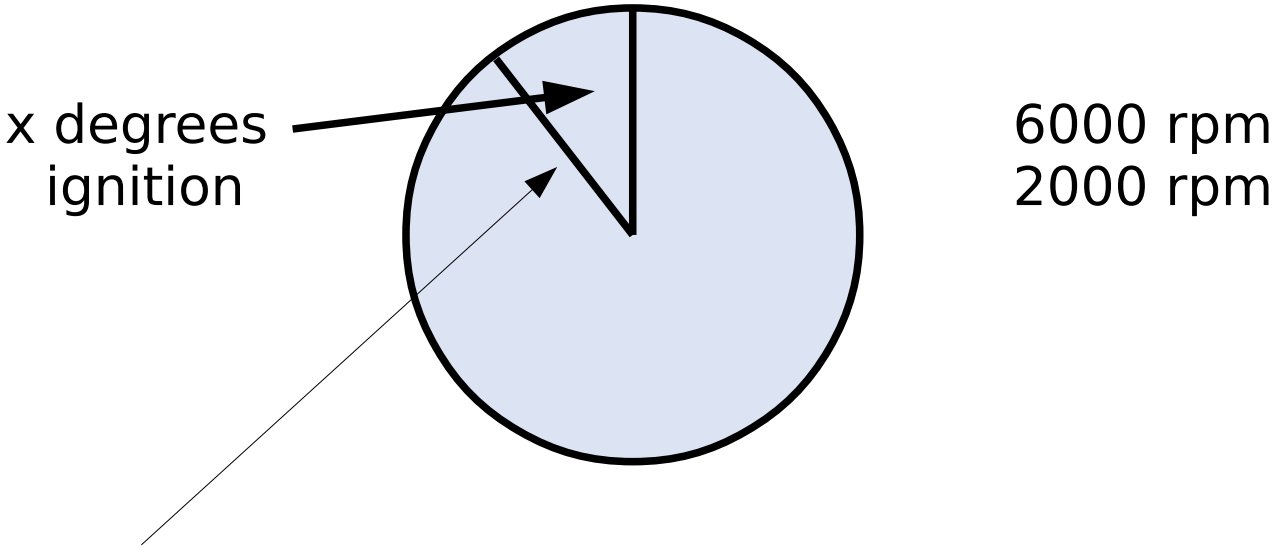
Sporadic task model ! :

	T	task, consisting of a sequence of
	J_i	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
→	~~e_i~~	~~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., characterised 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

Mapping of load to resources: Schedule

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

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

Priorities of Jobs

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

And Preemptability Resources

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 preemtable

Dependence ./. Independence of Tasks

precedence:

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

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

Precedence vs. Shared Data example: Mars Path Finder

Three tasks:

- craft control
- earth communication
- survey task

Shared data base:

- contains data
- access via very short organization operations
  (to reduce shared data access frames)

Precedence: availability of data for communication...

Shared data: access to data base

Temporal Dependency

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

- Lip synchronization (< 80 ms)
- Events must generate response within given time (io)

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:
  - extremely hard to find “good” WCET
  - high underutilization of resources

→ more models: hard ./. soft, 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)

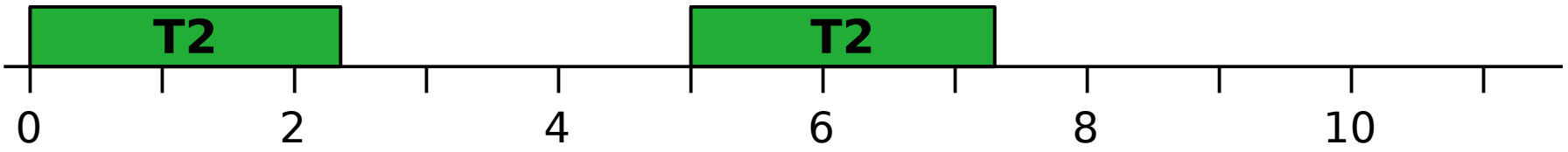
Criticality vs Priorities ...

Priority Assignment Following “Criticality”

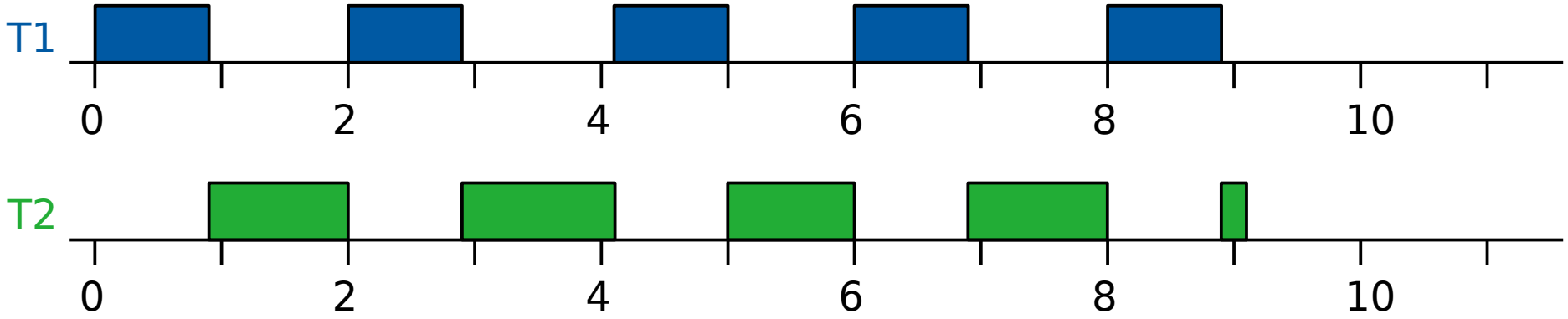
The more critical a task the higher the priority

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

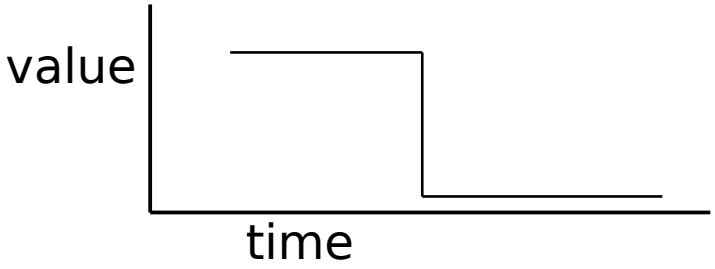
	T	task, consisting of a sequence of
	J_i	jobs
	P	inter-release times of jobs
	D	deadlines of jobs (relative or absolute)
→	e	execution times as probability distribution (of tasks, jobs)
→	Q	probability that deadline is met

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

Admission: ... (SRMS, QRMS)

Cost (time) value functions

- Hard real-time:



- Others (watch black board) :-(

- We may have an external speaker ...

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 $J_{fallback}$.)

Mixed Criticality Model (Sanjoy Baruah)

- Two or more “criticality levels”
- WCET for jobs with higher criticality are evaluated much more rigorously(pessimistic)

  higher “assurance” level requires

  → next slide
- More:

  http://www.artist-embedded.org/docs/Events/2010/Autrans/talks/PDF/Baruah/SkB-ARTIST2010.ppt.pdf
- May be Baruah as speaker in this course

Mixed Criticality Model (Sanjoy Baruah)

The mixed-criticality job model

Job J_i

- arrival time A_i
- deadline D_i
- criticality level L_i
- WCET function $C_i(1)$, $C_i(2)$, ...

The MIXED-CRIT SCHEDULING PROBLEM: Given an instance $\{J_1, J_2, ..., J_n\}$ of mixed-criticality jobs, determine an appropriate scheduling strategy

CERTIFICATION CRITERION: Job J_i should meet its deadline when each job J_k executes for at most $C_k(L_i)$, for all J_i.

The WCET of J_k, computed at J_i's criticality level

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 >> Pe

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.

Modeling helps a lot!