

Modeling Real-Time Systems

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:

- Resources
- Load (or Objective)
- Mapping

E.g. for building bridges:

materials; weight of cars, velocity of wind; structures

Resources

Active Resources ... Processors

- cpus, networks, disks, ...
- reusable

Passive Resources

- memory, data base locks, sequence numbers, ...
- reusable

Active vs. passive:

- Speed, execution, ... vs passively holding
- Not always clear

Load: Set of 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

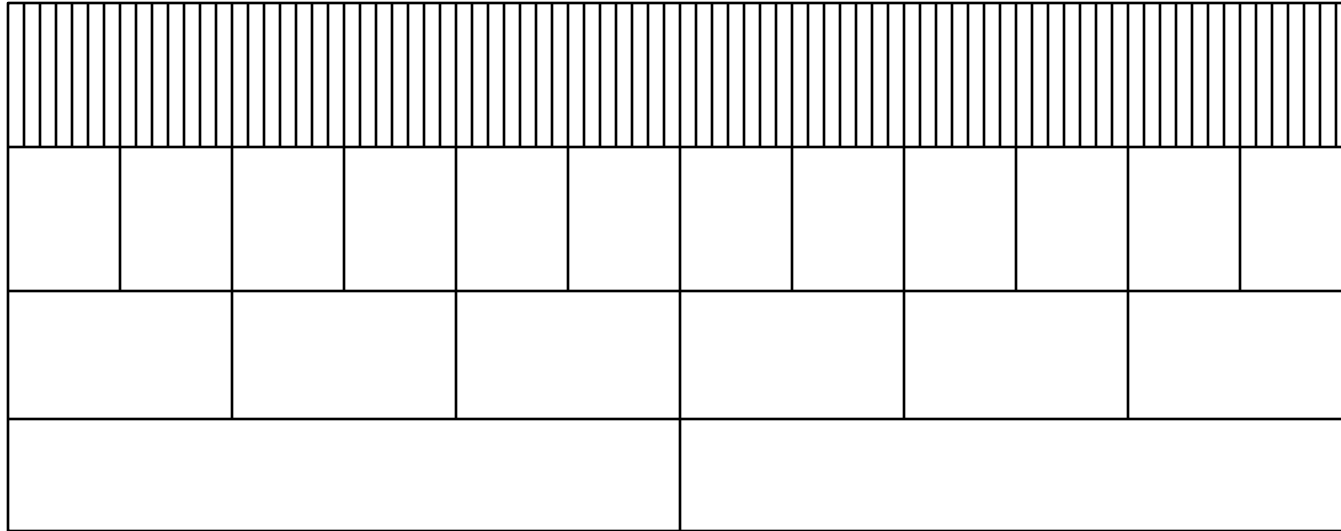
hyperperiod: least common multiple of p_i

-> NOTE

D can also be greater than P:

- **example: buffered audio output ...**

Simply Periodic (Harmonic)



Periods pair-wise multiples ...

Mapping: 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 jobs is scheduled before its release time
- all constraints are satisfied (precedence, resource usage)

a valid schedule is called *feasible* if all deadlines are met.

Schedulers and Admission

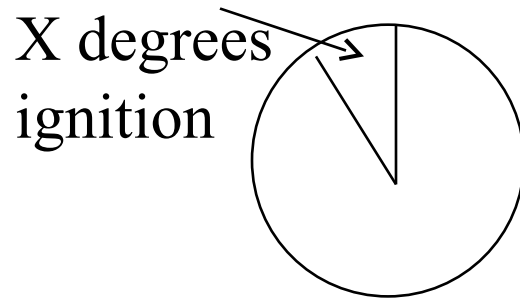
Scheduler
enforces/interprets ...
a schedule

Admission
can a new task be allowed into the current set
and the current resources
such that still a feasible schedule exists ?

Schedulers: (static, dynamic) priority, tabledriven, ...

An interesting example

engine control: period depends on degree:



6000 RPM

2000 RPM

Remedies (alternatives):

- Use “degree” instead of time
- Transform into “real” time
- sporadic/aperiodic tasks

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 Tasks (2)

Sporadic:

P MINIMUM inter-release time

Aperiodic:

No knowledge

A-priori admission only possible for sporadic task sets

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

Preemptability of tasks ...

preemptability of task

- can tasks 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
 - on other tasks and the operating system

And Preemptability Resources

preemptability of resources

- can tasks be preempted while using a particular resource
- examples:
 - cpu: preemptable
 - lock: non preemptable)

Dependence ./ Independence of Tasks

precedence:

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

shared data:

- tasks use some common data base

precedence graphs:

partial order relation on jobs ...

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)

Criticality of Tasks

Some deadlines may be more important to meet than others

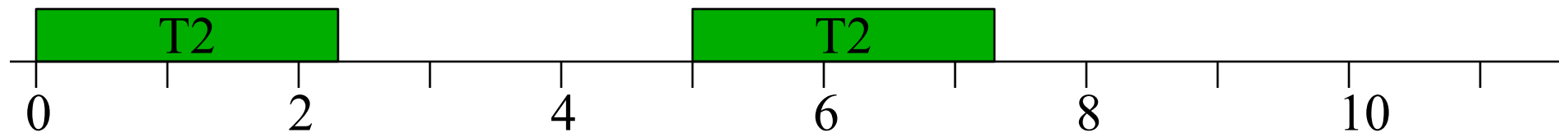
Criticality vs Priorities ...

Priority Assignment Following “Criticality”

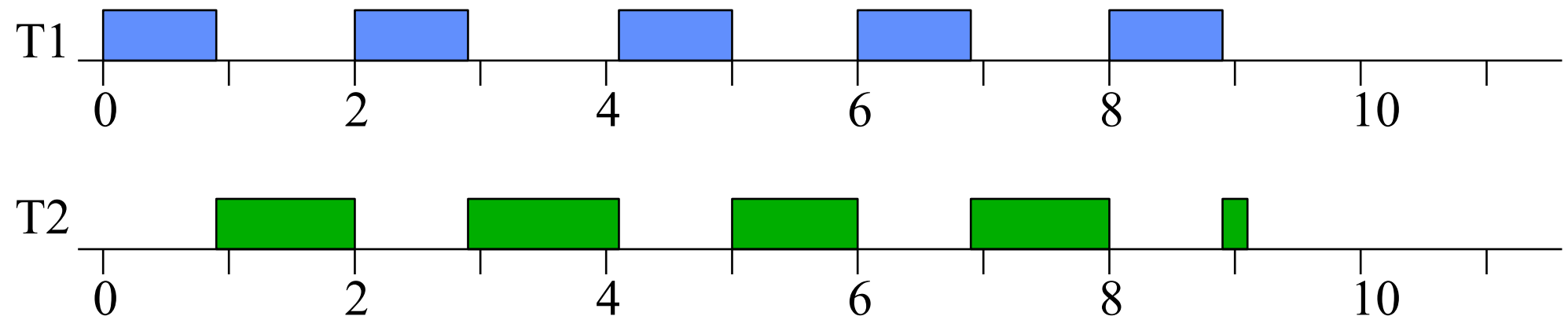
The more critical a task the higher the priority

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:

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

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
- **More:**
 - Imprecise computations
 - Time-value functions
 - Task pairs
 - probabilistic

Value of Data/Computations/Deadlines

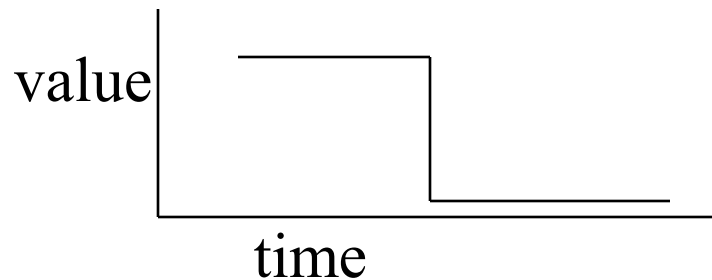
imprecise computations:

- 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)

cost value functions:



hard real-time
some others

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
try J otherwise J_{fallback} end
(if J is not completed before $D\text{-wcet}(J_{\text{fallback}})$
abort and do J_{fallback})

Execution Times

actual execution times, depending on

- data dependencies (if then else, compression, loops)
- hardware (see specific lecture on real-time & hardware)

usually not known, but

- minimum/maximum execution times (wcet)
- distribution

wcet based scheduling: high underutilization of resources
but:

wcet must be known+used for hard real-time systems

Remedies for Soft Real-Time:

- use hardware and algorithms with good wcet/avrg. Ratio
- Use probabilistic modeling

Execution Times, cont.

execution times follow a probability distribution

T task, consisting of a sequence of

J_i jobs

R 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)

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 $\gg P^e$

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.

Modeling helps a lot!