TDDD07 – Real-time Systems

Course overview & Scheduling I

Simin Nadjm-Tehrani

Real-time Systems Laboratory

Department of Computer and information Science

LINKÖPING UNIVERSITY

Welcome!

- … to this course for computer science and engineering related programs and Master profiles taken by other programs

  – Who are this year's participants?

Teaching team

- Simin Nadjm-Tehrani – Examiner and lecturer
- Reyhane Falanji – Course/lab assistant

Course overview

Course information

- www.ida.liu.se/~TDDD07
  - Lecture slides from last year - will be renewed the day before the actual lecture
- Check your LiU-email, and web timetable/messages for potential changes in schedule!
- In case you need to contact us:
  - Lab-related: your lab assistant
  - Ladok-related questions: adm-gu@ida.liu.se (Hanan Mohsen)
  - Any other questions: ask your examiner!

Literature

- Two recommended text books
  - Jan Carlson et al., Real-time Systems Compendium from MRTC
  - A. Burns and A. Wellings: *Real-Time Systems and Programming Languages (Fourth Edition)*, Addison-Wesley
- Articles and book chapters
  - List and links are on the course website

Lab preparation

- Follow the instructions on the web for registration in **webreg** and forming the lab groups
  - Deadline: **5th November!**
- Please read the lab material and prepare **before** Lesson 1 first hour that introduces the lab part!
  - You will get access to the lab environment once you have answered the preparatory questions for the lab

Lab groups

- Note that we have currently a reserve lab group B on timeEdit as we did not know how many will turn up.
- If the number of registered students (in webreg) is what we have been given in October, then there will **only be a group A.**
- We will then remove the B occasions, and keep a few reserve slots (no teacher attendance).

Finishing on time ☺

- Check the deadline for lab *examination* on the course web page!

- Further assistance in 2025 will have to be individually organised (subject to assistant availability)

- Those not finished before the deadline: two new dates will be available during retakes in March/August

Examination

- Written exam (4hp) in January
  - Bonus points at the exam, given in advance for doing deeper exercises – Look out for $B sign in lectures!
- Laboratory exercises (2hp)
  - demonstration & *individual* oral discussion with lab assistant
  - Nominal amount of work: ~55h of which ~30h on preparation & programming (12h assistant help is available)
  - (pairwise) lab report forms part of examination

Student evaluation HT23

- The course had a mean evaluation score of 3.75 in web evaluation
  - Few text comments
- We usually do a muddy card (anonymous) evaluation half-way through the course
  - Comments from last year are still shown on the course web this year, which will be replaced with a new one (planned 15/11)

Response to evaluations: exam preparation

- Comment:
  - Solutions to things like exam questions, as just going through old exams might make one answer the wrong thing thinking it was correct
  - Response:
  - The adopted philosophy in this course is that the students prepare answers and the teachers comment it.
  - As soon as you have a solution to a question and want it checked ask your examiner to give you comments on it.
  - Any unclear questions can be taken up in the extra resource session planned before the first exam.

Response to evaluations: the labs

- Comment:
  - I also liked the labs - it was a little bit difficult to get started but it was a good experience to just have requirements to meet and not specific instructions.
  - It might also be a good idea to make it a bit clearer what is expected of you during the first lab sessions as it felt sort of unclear if what we were doing would be enough.
- Response:
  - That is what the lab assistant is there for! Do ask questions as soon as you are unsure. Idea is to guide you up to a limit, without stating the exact solution.

Response to evaluations: Literature

- Comment:
  - I do think the articles as reading material worked well when they were all listed with links on one page and they felt relevant and useful even if I did not always read every section of every one exceptionally deeply.
- Response:
  - Happy to hear this!

- Our intention is that the important topics of this course are digested week-by-week
  - The large set of reading materials are not readable/understandable in the last three days before the exam!
  - Spreading the workload over the weeks and interaction with lecturer is highly recommended
  - Examiner is at your disposal for as many questions to clear up your doubts during the six weeks of the autumn period

Questions?

Course Overview

Overall division of content is along the time spent on lectures

- Three lectures on CPU scheduling
- Three lectures on distributed (networked) systems and associated resource/timing aspects
- Three lectures on dependability concepts and predictability in applications
- Final lecture: tying it all together

The first three lectures

- CPU as a resource: Scheduling

This lecture

- Overview on real-time systems and why dependability matters
- We start with cyclic scheduling

What is a real-time system?

Why dependability matters?

What is a real-time system?

- A system that has *explicit* timing requirements

- Typical examples
  - Traffic signal in a railway crossing
  - Alarm monitoring in chemical process plants: deviations in liquid levels, etc

Open Radio Access Network (6G)

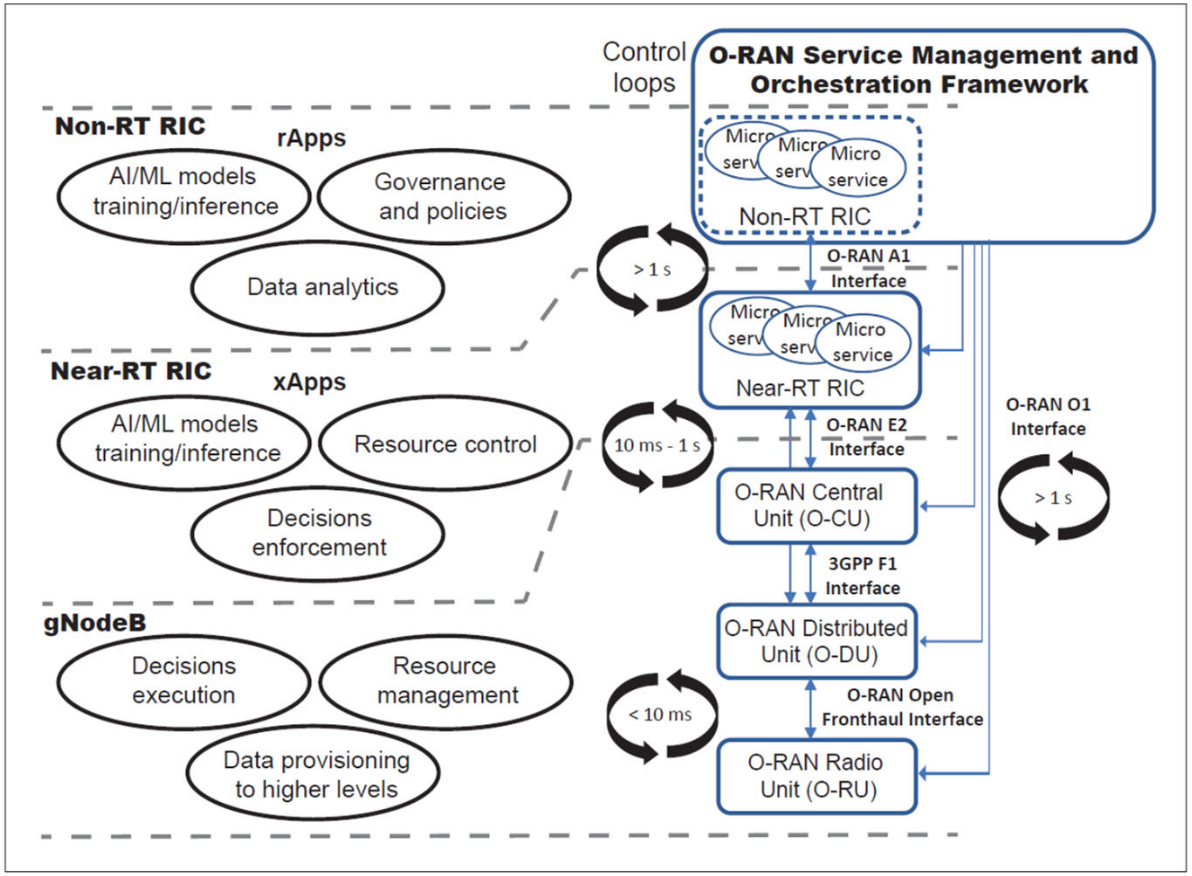


FIGURE 3. Open RAN use case control loops hierarchy.

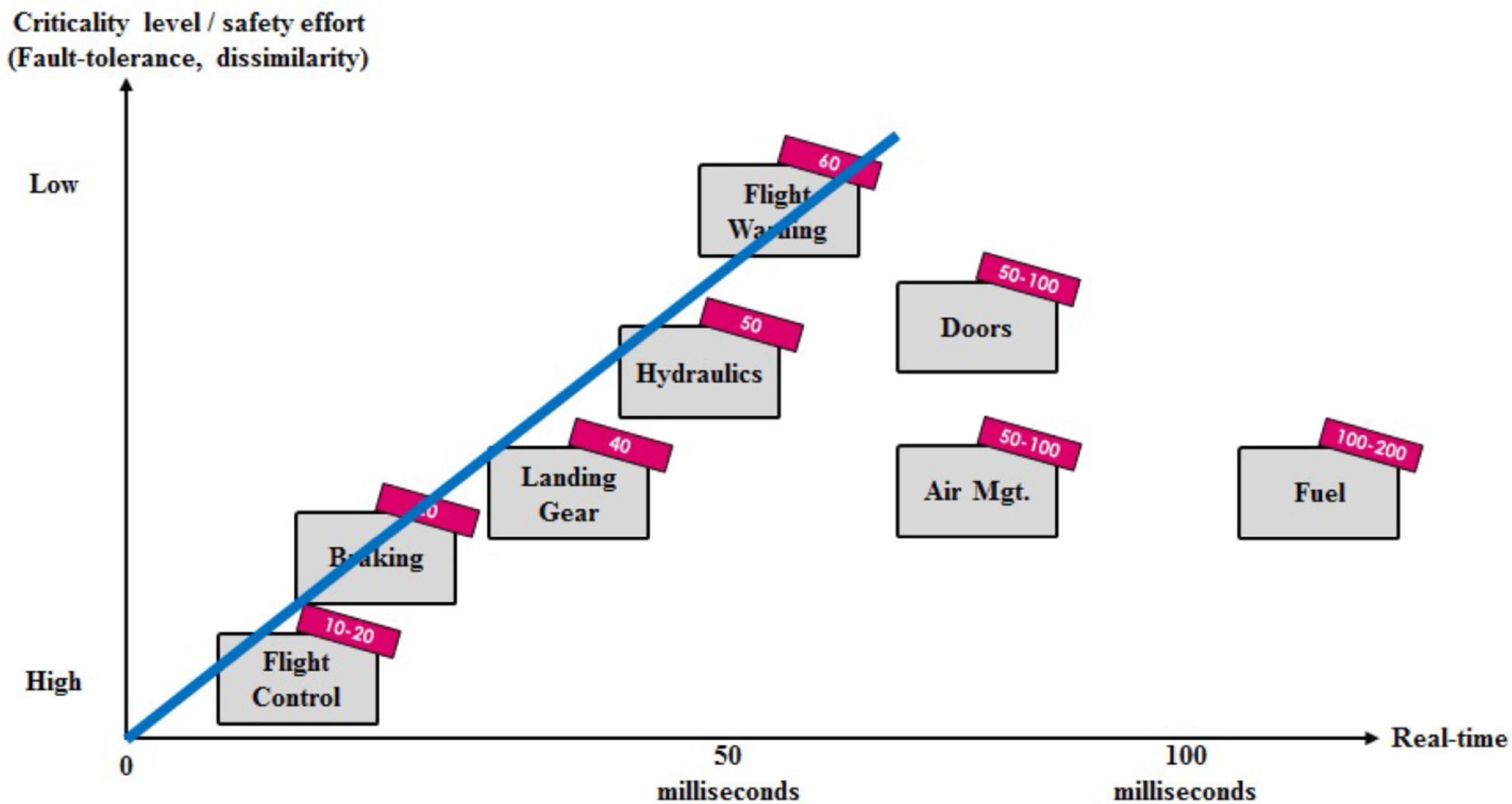
Chaoub et al. 2023
DOI: https://ieeexplore.ieee.org/document/10078076

The cost of latency

- The increased level of data-driven and digital services makes performance a major attribute of almost every application

https://www.gigaspaces.com/blog/amazon-found-every-100ms-of-latency-cost-them-1-in-sales

Fast is not equal to critical



Source: Assurance of multicore processors in airborne systems,

DOT/FAA/TC-16/51, July 2017

Business example – 2 August 2012

- ”An algorithmic trading software bug is being blamed for a day of wild swings at the NYSE – and has resulted in the trader (Knight Capital) placing the dodgy orders that resulted in a $US440 million pre-tax loss for the company.”
- *Reuters* suggests that timing may have been the problem: the orders may have been intended to be filled during the trading day, but instead were filed in the opening minutes of trading
- Similar issue in Stockholm: 2 May 2022!

https://www.dn.se/ekonomi/borsen-faller-spelbolag-mot-strommen/

Early uses of ”real-time” in social media

- Georgia Tech: Crowdsourcing Democracy
  - A social media aggregator that pulls content from 20 sources and analyzes the data in real-time using keywords
  - Used the tool during Nigeria's presidential election April 2011, tracking the election process
  - Identifying problems by monitoring citizen's comments on social media platforms
  - Tracked about 50 reports a second, analyzed based on keywords and location data

Early crowdsourcing

- Cambridge University TIME project:

  create a minute-by-minute image of congestion in cities

- Now in Singapore

DOI: 10.1109/ICDCS.2015.11

Detecting outbreaks

- Using twitter feeds and machine learning to provide early warning of outbreaks of contagious diseases

Serban et al. May 2019:
https://doi.org/10.1016/j.ipm.2018.04.011

Difference

- … with cars and airplanes?

- There is no physical system to control, but the system output is intended to be produced in some temporal relation to system inputs
- System components are distributed over (large) geographic areas

Inputs can be unpredictable

- … and requirements are typically fuzzy

- Customer requirements may say: System is expected to generate the output "in time" even if we can not restrict the rate of arrival of inputs

Misconception!

- Other criteria:
  - Volume: System must cope with high loads
  - Freshness: Only recent data useful, Collected data needs to be time-stamped

Event sequence analysis

Affected 50 million people

“A valuable lesson from the ... blackout is the importance of having time-synchronized system data recorders. The Task Force’s investigators labored over thousands of data items to determine the sequence of events much like putting together small pieces of a very large puzzle. That process would have been significantly faster and easier if there had been wider use of synchronized data recording services.”

https://www.nerc.com/docs/docs/blackout/NERC_Final_Blackout_Report_07_13_04.pdf

Sending important alarms

- 2011 when the connection between the fire alarm systems at South Sweden hospitals and the SOS Alarm did not work for 4 hours...
- This was a sensation!

http://www.dn.se/nyheter/sverige/omfattande-larmstrul-i-sodra-sverige/

- Today we have patients at risk of dying

due to broken gadgets ☹

https://www.svt.se/nyheter/inrikes/svt-avslojar-allvarliga-brister-i-trygghetslarm-kommuner-informerades-inte

Growing dependence on IoT

- Even affecting pets' lives

https://www.theguardian.com/technology/2016/jul/27/petnet-auto-feeder-glitch-google

- Where else can things go wrong?
  - Find your pet example!

Is timeliness reasoning (in one CPU) enough?

- We need to reason about how things can go wrong
  - Without that real-time guarantees are meaningless!

- We need performance concepts in distributed systems

- Let’s start with processes in one CPU…

Real-time processes

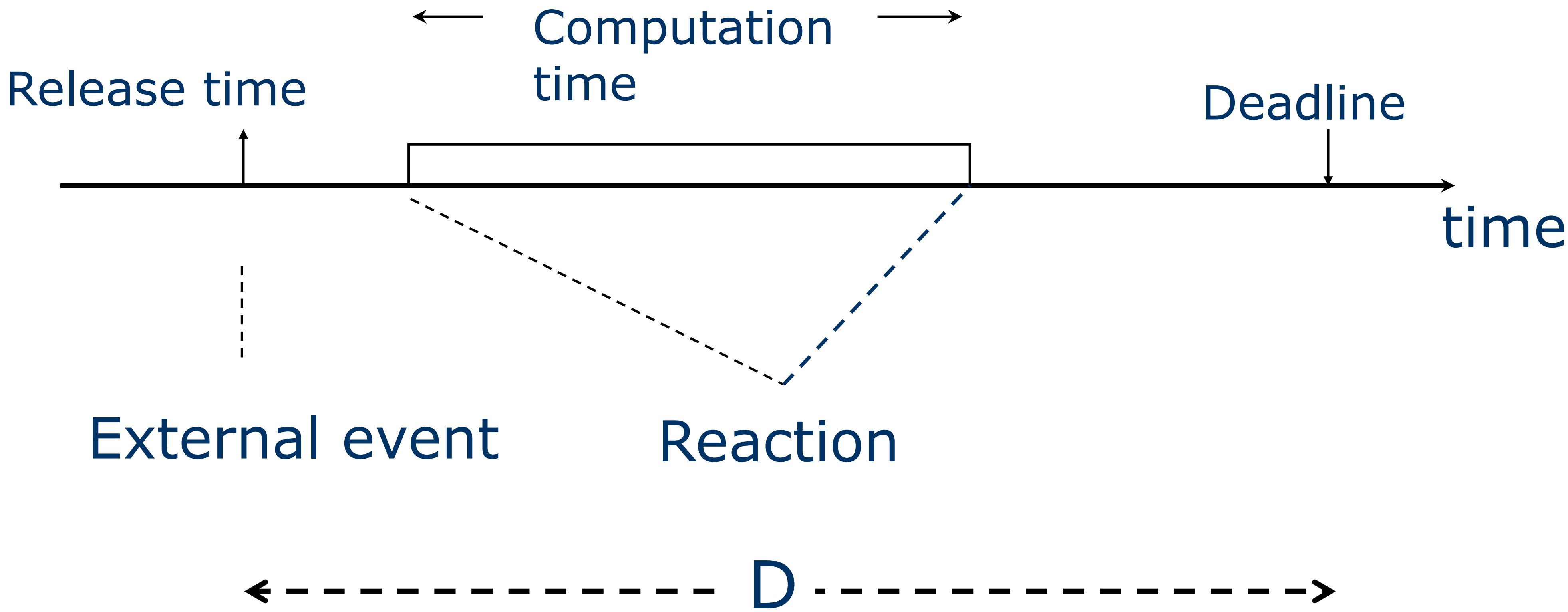
- In your operating systems course: scheduler's role is to ensure that *each process gets a share* of the CPU

- With processes that have **hard** deadlines it is not enough that processes get a share *some time*

  Hard real-time systems: The time that the result of the computation is delivered is as important as the result itself

- Predictability!

What is meant by predictable?



Hard real-time systems: Do **all** processes meet **all** their deadlines?

Deadlines

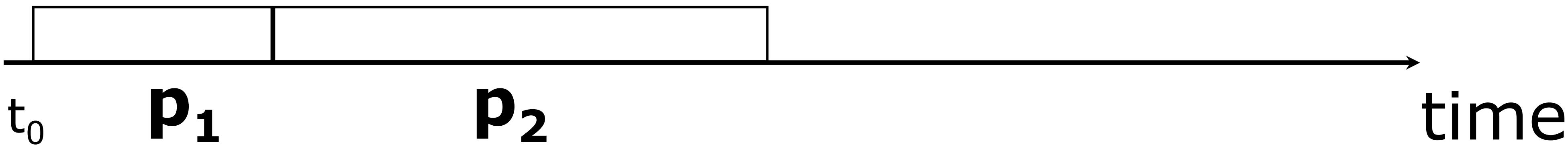
- Hard: Not meeting any deadline is a failure of the system
- Soft: It is desirable that deadlines are met, but OK if they are missed every now and then

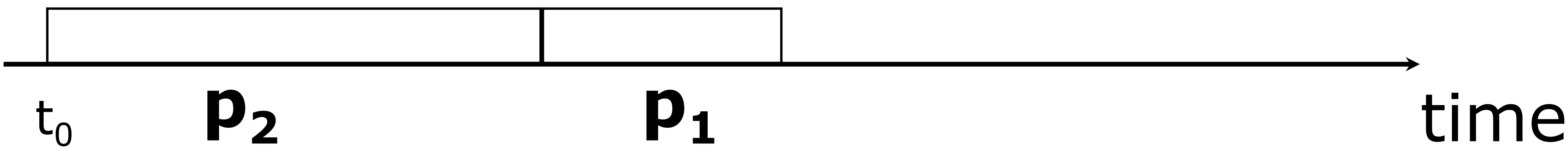
How often?

- Firm: It is OK that they are missed now and again, but after the deadline the result is of no use

The order matters!

Consider the following processes:	p_1	p_2
Computation time (C_i)	5 ms	10 ms
Deadline (D_i)	20 ms	12 ms





Scheduling

… is about allocating resources, specially the CPU time, among all computational processes such that the timeliness requirements are met.

If **all** processes meet their deadlines then the process set is **schedulable**.

CPU as a resource: Scheduling

Scheduling

- Performed off-line or on-line
- With information available statically or dynamically
- Preemptive or non-preemptive

Which parameters?

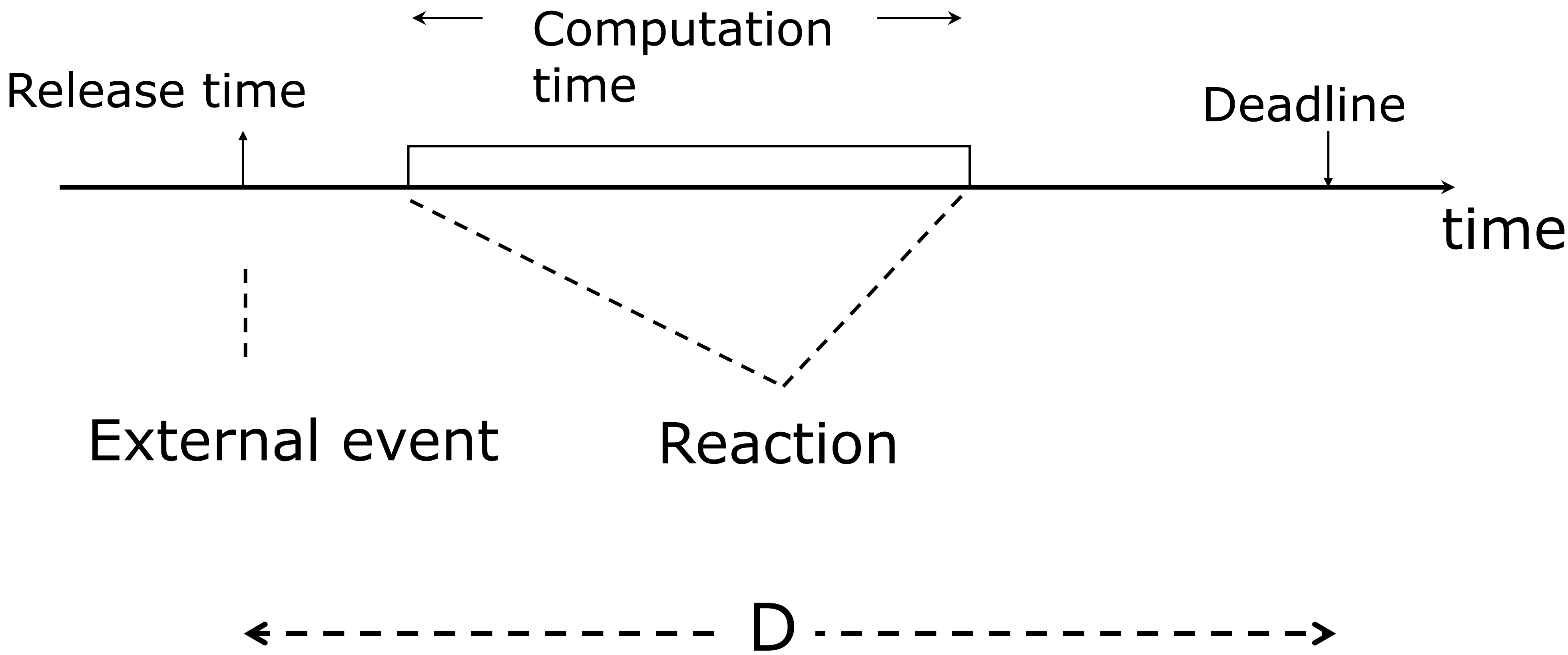
Scheduling policy induces an order on executions given an algorithm and a set of parameters for the task set:

- Deadline
- Release time
- Worst case execution time (WCET)
- …

Process parameters

- How to determine deadlines?
- When (how often) is a process released?
- How to find the maximum computation time (WCET) for each process?

Recall: Parameters



Deadlines are part of application requirements

Release times

- Reading and reacting to continuous signals
  - Periodicity

- Recognising/reacting to some *aperiodic* events
  - Minimum inter-arrival time
  - ➢ Sporadic processes

Computation time (WCET)

- Depends on
  - Hardware
  - Application code (algorithm)
  - Compiler
  - Data

Cyclic scheduling

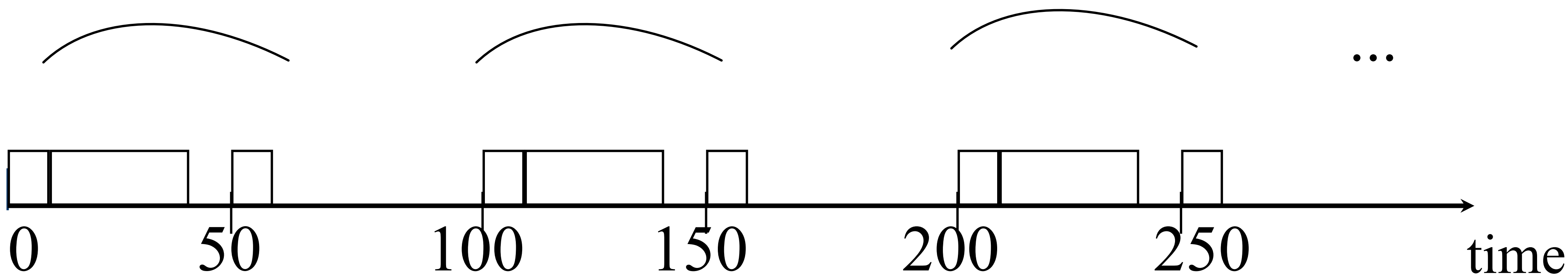
- A schedule is created based on statically known and fixed parameters (period, WCET)
- Off-line decision on which task runs & when it runs
  - When executing: Run the processes in pre-determined order using a table look-up
- To run processes in the “right” frequency find
  - Minor cycle
  - Major cycle

Example (1)

Consider following processes: P_1 P_2

	P_1	P_2
Period(Ti)/Deadline	50	100
Worst case execution time (Ci)	10	30

Note: repetition!



A cyclic executive

```
every_major_cycle do{
   read all in_signals;
   run_minor_cycle_1_processes;
   wait_for_interrupt;
   write all out_signals;
   ...
   read all in_signals;
   run_minor_cycle_n_processes;
   wait_for_interrupt;
   write all out_signals;
   }
```

End of minor cycle

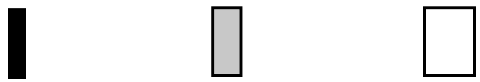
End of minor cycle

Finding Minor/Major Cycle

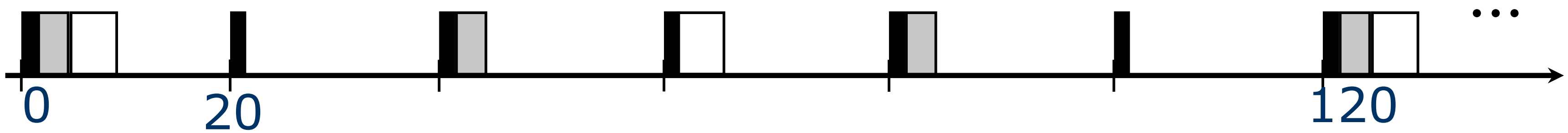
First try:
Minor cycle: greatest common divisor
(sv. sgd)

Major cycle: least common multiplier
(sv. mgm)

Example (2):



process	A	B	C
period	20	40	60



Construction of a cyclic schedule

Off-line analysis in order to fix the schedule might be iterative

- Each process P_i is run periodically every T_i (i.e. should be completed once every T_i)
- Processes are placed in *minor cycle* and *major cycle* until repetition appears
- Check: Are all process instances runnable with the given periods and estimated WCET?
- If not, reconsider the minor/major cycle and/or some process parameters

Next lecture

- We will continue with a more detailed look at construction of cyclic schedules.

Questions?

www.ida.liu.se/~TDDD07

LINKÖPING
UNIVERSITY