

TDDD07 – Real-time Systems

Lecture 1:

Course overview & Scheduling I

Simin Nadjm-Tehrani

Real-time Systems Laboratory

Department of Computer and Information Science
Linköping University

- To this course for computer science and engineering related programs and Master profiles taken by other programs
- Who are this year's participants?

Updated information

- www.ida.liu.se/~TDDD07
 - Tell us if you miss something!
- Check timetable/messages for potential changes in schedule!
- Your other contacts:
 - Course assistant: Klervie Toczé
 - Lab assistants: Klervie Toczé, and Chih-Yuan (Sana) Lin

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

Lab organisation

- Follow the instructions on the web for registration in **webreg** and forming the lab groups
 - Deadline: **1st Nov!**
- Please read the lab material and prepare **before** the labs!
 - You will get access to the lab environment once you have answered the preparatory questions for the lab
- Students from earlier years?
 - Please do not register for labs!

Finishing on time😊

- Check the deadline for lab examination on the course web page!
- Further assistance in 2018 will have to be individually organised (subject to assistant availability)
- For labs not demonstrated before the course deadline two new dates will be announced in April/August

Web evaluations earlier years

- Labs are too easy
- Labs are difficult to get into, no exact instructions on what to do
 - We have had both types of comments before
 - Which is why we have a minimum requirement (two first exercises) and an additional one for those who aim for a deeper learning (lab 3, \$B)

Web evaluations earlier years

- “Difficult to read so much”
 - Part of the course goals is to be able to extract information from various sources and read research articles
 - For Swedish students this is actually part of the educational goals!

HSV: För civilingenjörsexamen skall studenten visa kunskap om det valda teknikområdets vetenskapliga grund och beprövade erfarenhet samt insikt i aktuellt forsknings- och utvecklingsarbete

Questions?

Course overview

- 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

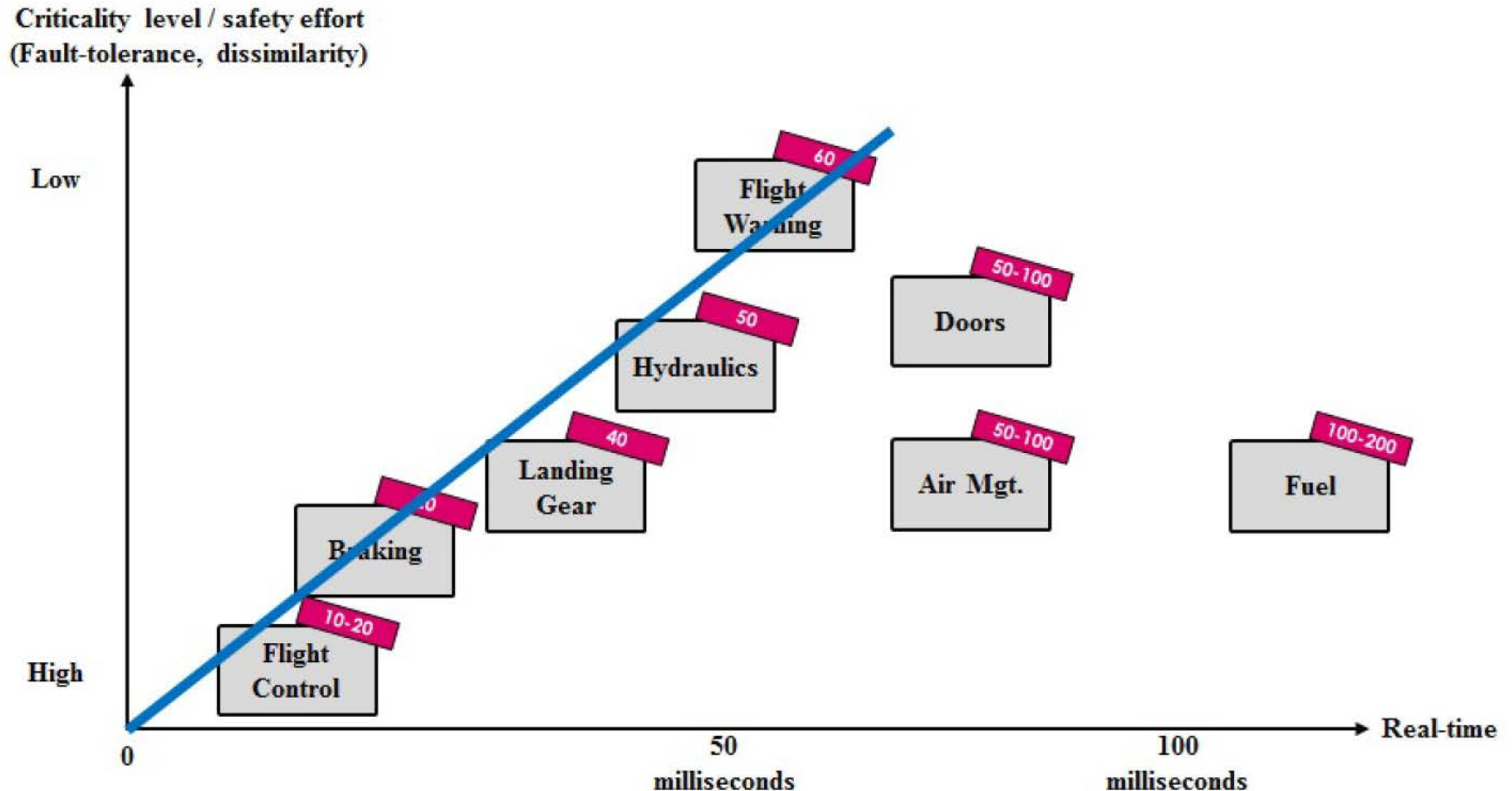
This lecture

- Introduction to Real-time systems
- We start with cyclic scheduling

What is a real-time system?

- Systems that have explicit timing requirements
- Typical examples
 - Traffic signal in a railway crossing: if signal does not change to red (and a gate closed) traffic accidents can happen
 - Alarm monitoring in chemical process plants: deviations in liquid levels, pressure, temperature can be a sign of broken pipes/valves and need to be detected in time

Fast is not = critical



Source: Assurance of multicore processors in airborne systems,

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

Business: 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

Novel uses of “real-time”

- Georgia Tech: Crowdsourcing Democracy

Researchers have developed a social media aggregator tool that can pull content from about 20 sources and analyze the data in real-time using keywords. The researchers, led by Georgia Tech professor Michael Best, used the tool during Nigeria's presidential election April 2011, tracking the election process and identifying problems that arose by monitoring citizen's comments on social media platforms. At the system's highest activity level, the aggregator tracked about 50 reports a second, analyzing them based on keywords and location data.

More on crowdsourcing

- Carnegie Mellon University:
Researchers have developed Tiramisu, an iPhone application that uses crowdsourcing to enable riders to know when the next bus will arrive
- Cambridge University TIME project:
A new project has shown that by using existing sources of information about traffic flow it is possible to create a minute-by-minute image of congestion in cities

- ... 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

Inputs can be unpredictable

- ... and requirements are typically fuzzy
- Customer requirements: System is expected to generate the output “in time” even if we can not restrict the rate of arrival of inputs
- Other criteria:
 - Volume: System must cope with high loads
 - Freshness: Only recent data useful,
Collected data needs to be time-stamped

Misconception!

Event sequence analysis

- From the report on the 2003 electricity blackout in the USA-Canada:

"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."

Sending of fire alarms

- When the connection between the alarm systems at South Sweden hospitals and the SoS Larm does not work for 4 hours...

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

- Need to study systems that are not in one location, need to consider faults

Growing dependence on IoT

- Now affecting pets' lives

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

- Where else can things go wrong?

Real-time processes

- From 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*

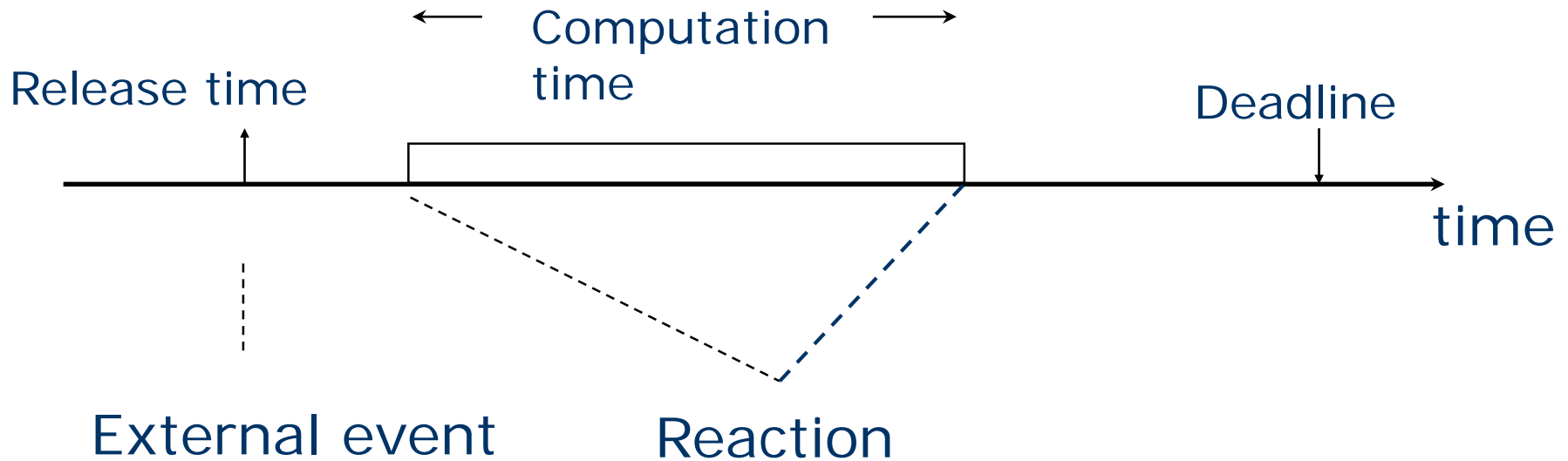
Hard real-time systems: The time that the result of the computation is delivered is as important as the result itself

- Predictability!

Predictable not = “fast”!

The film...

What is meant by predictable?



← ——— D ——— →

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

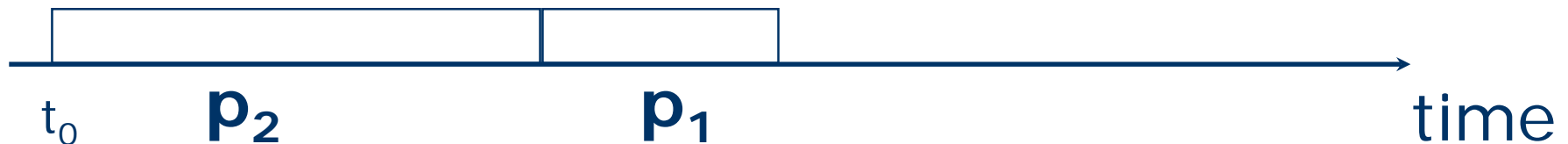
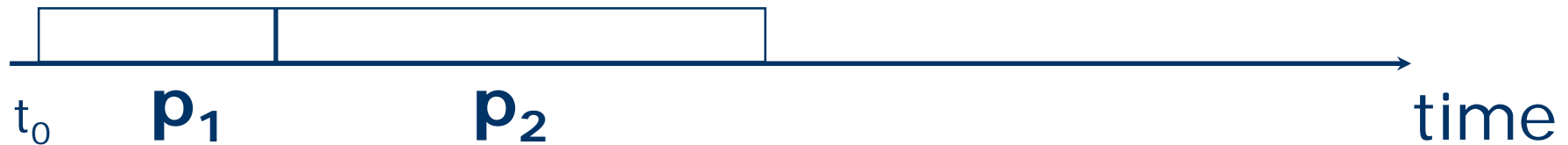
- Hard: Not meeting any deadline is a failure of the system
- Soft: It is desirable that deadlines are met, but OK if they are missed every now and then
- Firm: It is OK that they are missed now and again, but after the deadline the result is of no use



How often?

Order matters!

Consider following processes:	p₁	p₂
Computation time (C_i)	5 ms	10 ms
Deadline (D_i)	20 ms	12 ms



... 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**.

The first three lectures

- Introduction to Real-time systems
- CPU as a resource: Scheduling

This lecture:

- We start with cyclic scheduling

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

Which parameters?

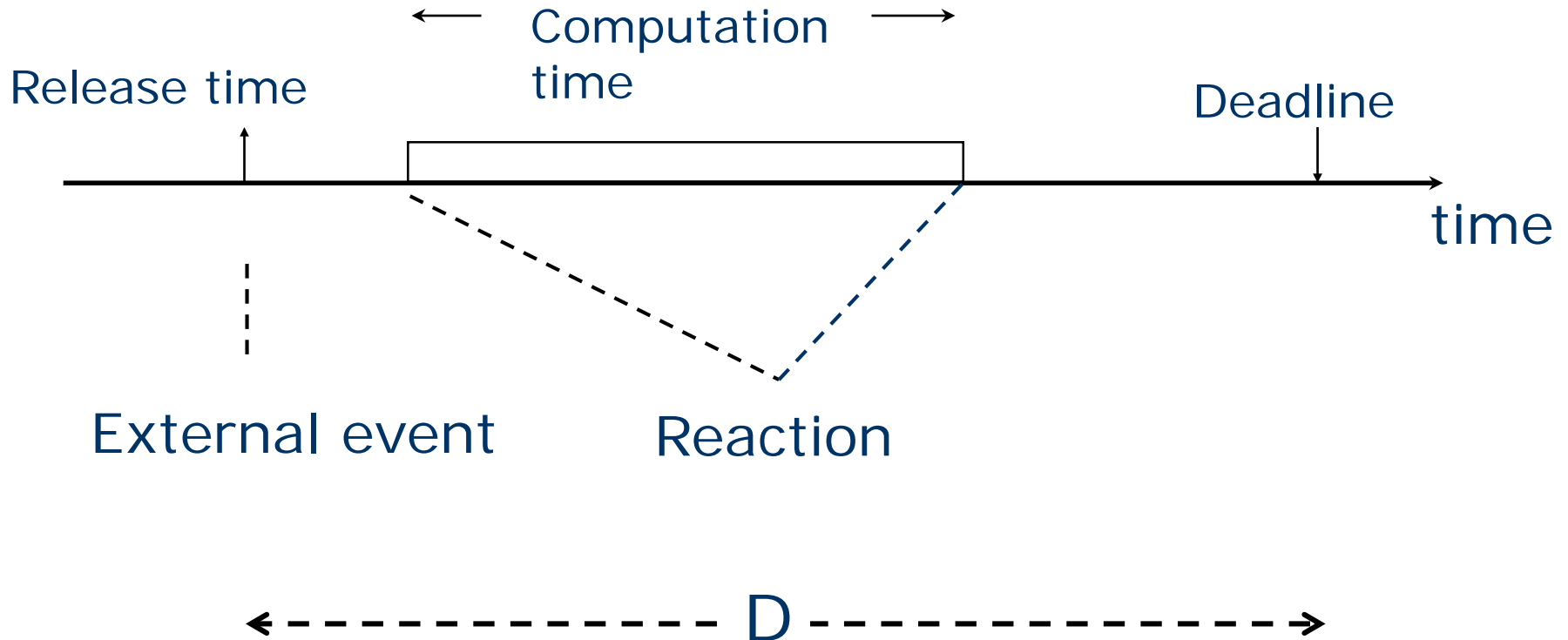
Scheduling policy induces an order on executions given an algorithm and a set of parameters for the task set:

- 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

- 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
- 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

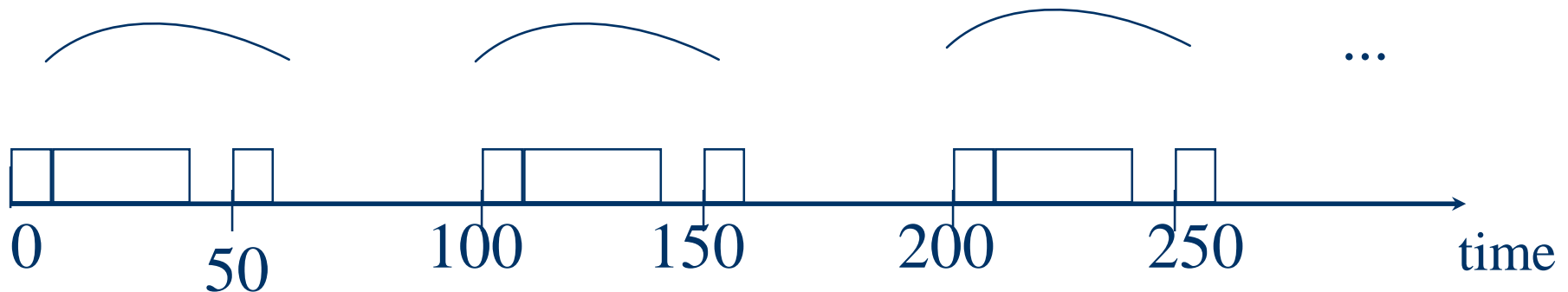
Period(T_i)/Deadline

50 100

Worst case execution time (C_i)

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;  
}
```



Finding Minor/Major Cycle




First try:

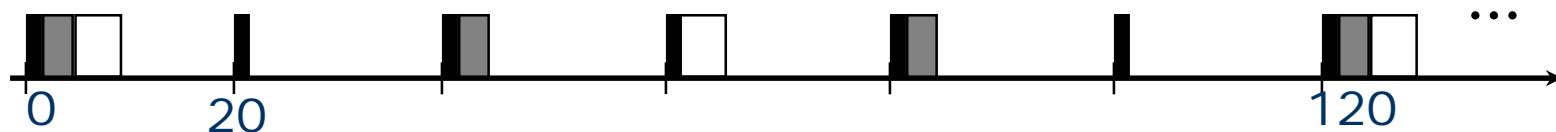
Minor cycle: greatest common divisor
(sv. sgd)

Major cycle: least common multiplier
(sv. mgn)

Example (2):

process
period

		
A	B	C
20	40	60



Construction of 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.