AARHUS
UNIVERSITY
DEPARTMENT OF ENGINEERING

Architecture & Design of Embedded Real-Time Systems (TI-AREM)

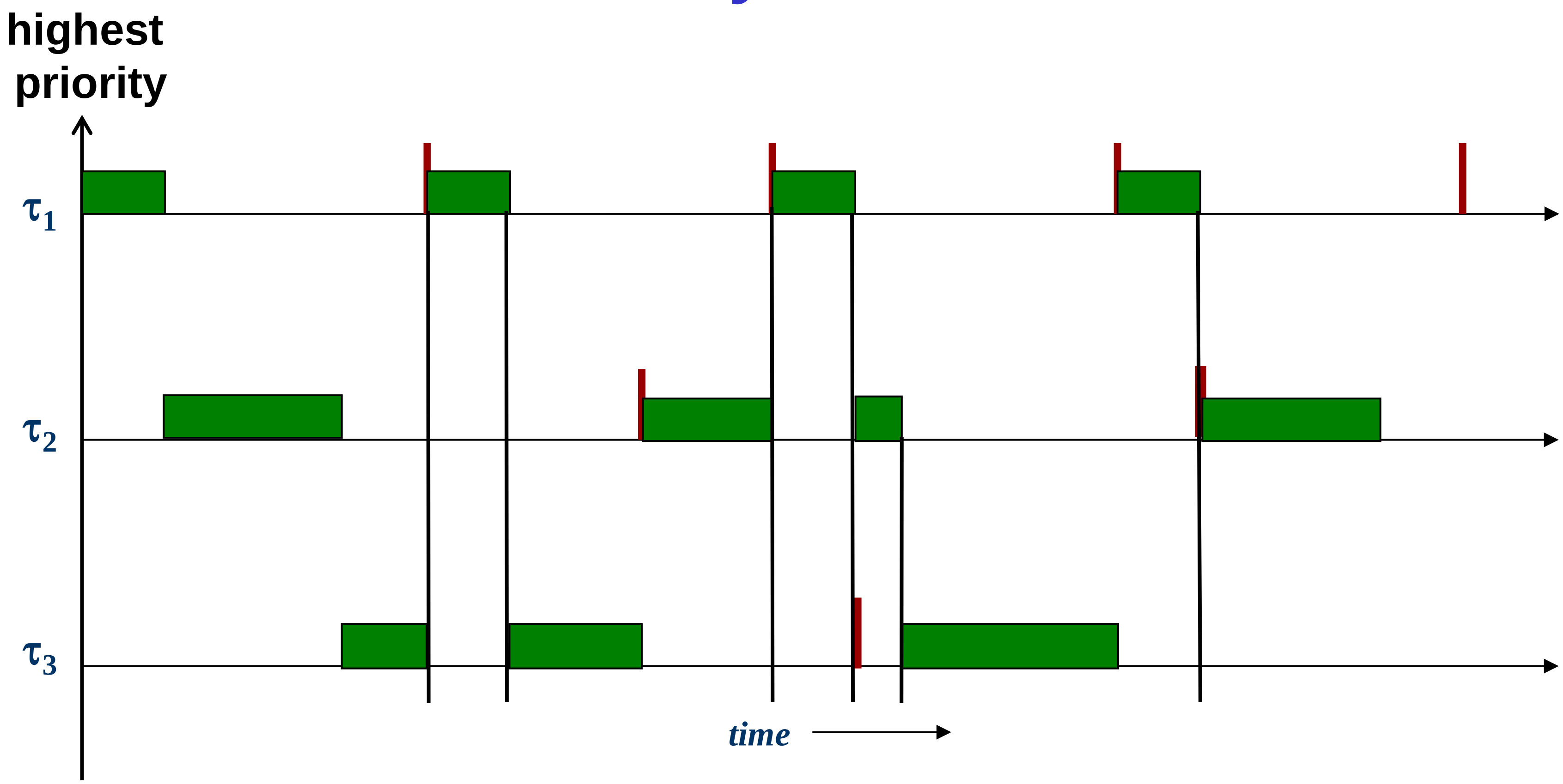
Threads and Schedulability (RMA/RMS)

Version: 3-3-2015

Agenda

- Scheduling policies:
  - Earliest-Deadline-First scheduling (EDF)
  - Rate monotonic scheduling (RMS)
- Rate Monotonic Analysis (RMA)
- Demo of Times tool

Anatomy of a Task



Task Execution Time (C)

End Of Period (T = Period Length)

Evaluation of Scheduling Policy

- A scheduling policy is evaluated by:
  - Its ability to satisfy all **deadlines**
  - **Its CPU utilization**: percentage of time devoted to useful work
  - **Its Scheduling overhead**: time required to make scheduling decision (also called context switching)

Earliest-Deadline-First Scheduling (EDF)

- **EDF: dynamic priority** scheduling scheme
- A task **closest to its deadline** is given **highest priority**
- Requires recalculating processes at every timer interrupt
- **Relates to BPDs Dynamic Priority Pattern**

EDF Analysis and Implementation

- EDF can use 100 % of CPU
- A set of tasks is schedulable if the sum of the task loading is less than 100 %
- On each timer interrupt:
  - compute time to deadline;
  - choose task closest to deadline
- Generally considered too expensive to be used in practice

Rate Monotonic Scheduling (RMS)

- **RMS** (Liu and Layland 1973): widely-used, analyzable scheduling policy
- The **Rate Monotonic Scheduling** algorithm:
  - assigns fixed priorities
  - assumes periodic tasks
  - assigns highest priority to the task with the shortest period
- Analysis is known as **Rate Monotonic Analysis (RMA)**
- **Relates to BPDs Static Priority Pattern**

Rate Monotonic Analysis (RMA) **Assumptions**

1. All tasks run on a single CPU
2. All tasks are **periodic**
3. **Deadline is at end of period**
4. Task switching is instantaneous (Zero context switching time)
5. **No data dependencies between tasks**
6. Highest-priority ready task runs
7. Tasks accounts for all processor execution time

Periodic Tasks and Utilization Bound

Theorem #1

- A set of ***n*** independent periodic tasks scheduled by the ***Rate Monotonic Algorithm*** will always meet its deadlines, for all task phasing's, if

$$\boxed{\frac{C_1}{T_1} + \ldots + \frac{C_n}{T_n} \le n(2^{1/n} - 1) = U(n)}$$

where

C_i = worst-case task execution time of task_i (**WCE**)

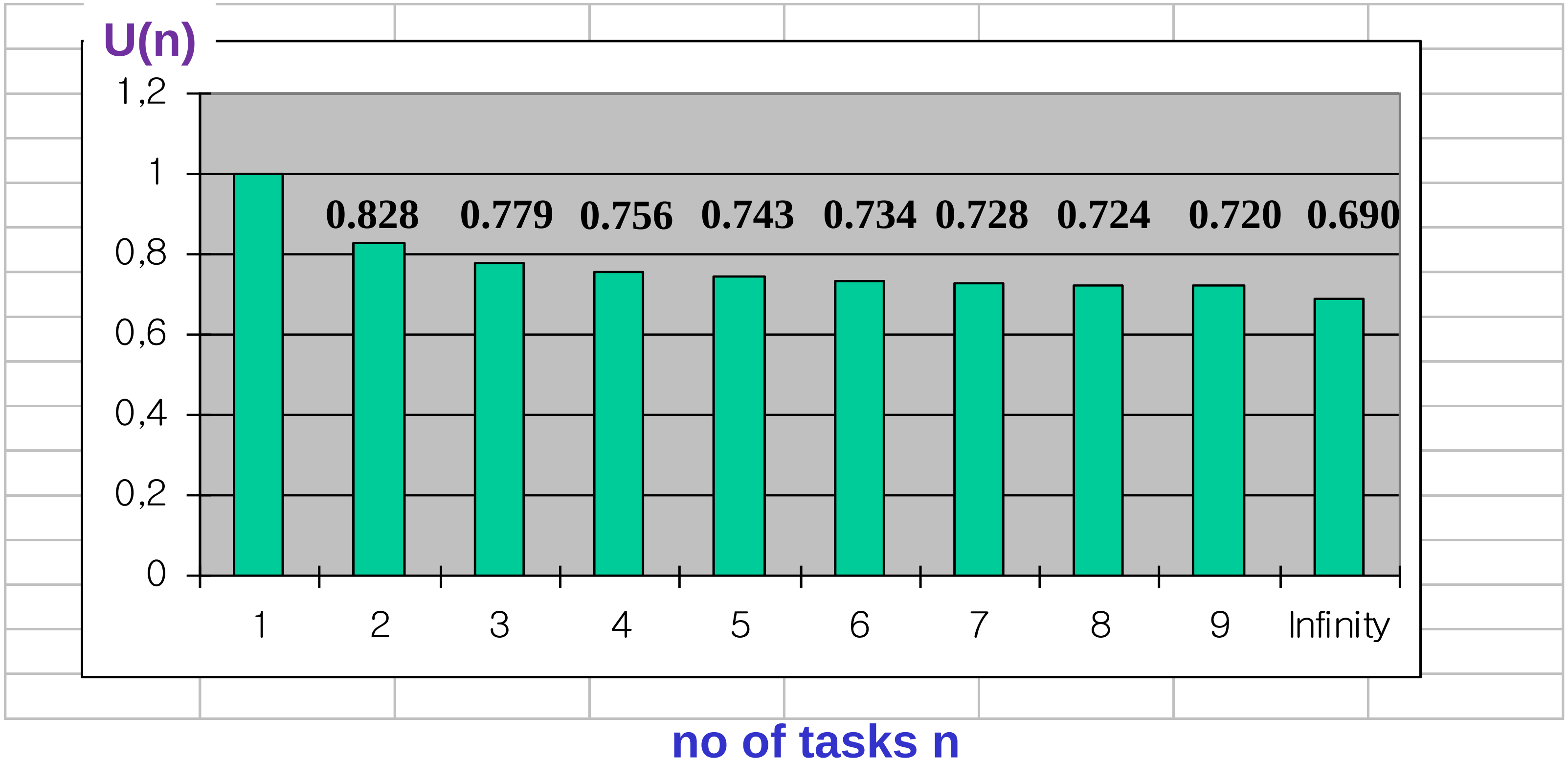
T_i = period of task_i

U(n) = utilization bound for *n* tasks

[Ref. Liu and Layland, 1973]

Utilization Bound Theorem

- RMA assigns a fixed priority such that the shorter the period of a task, the higher its priority



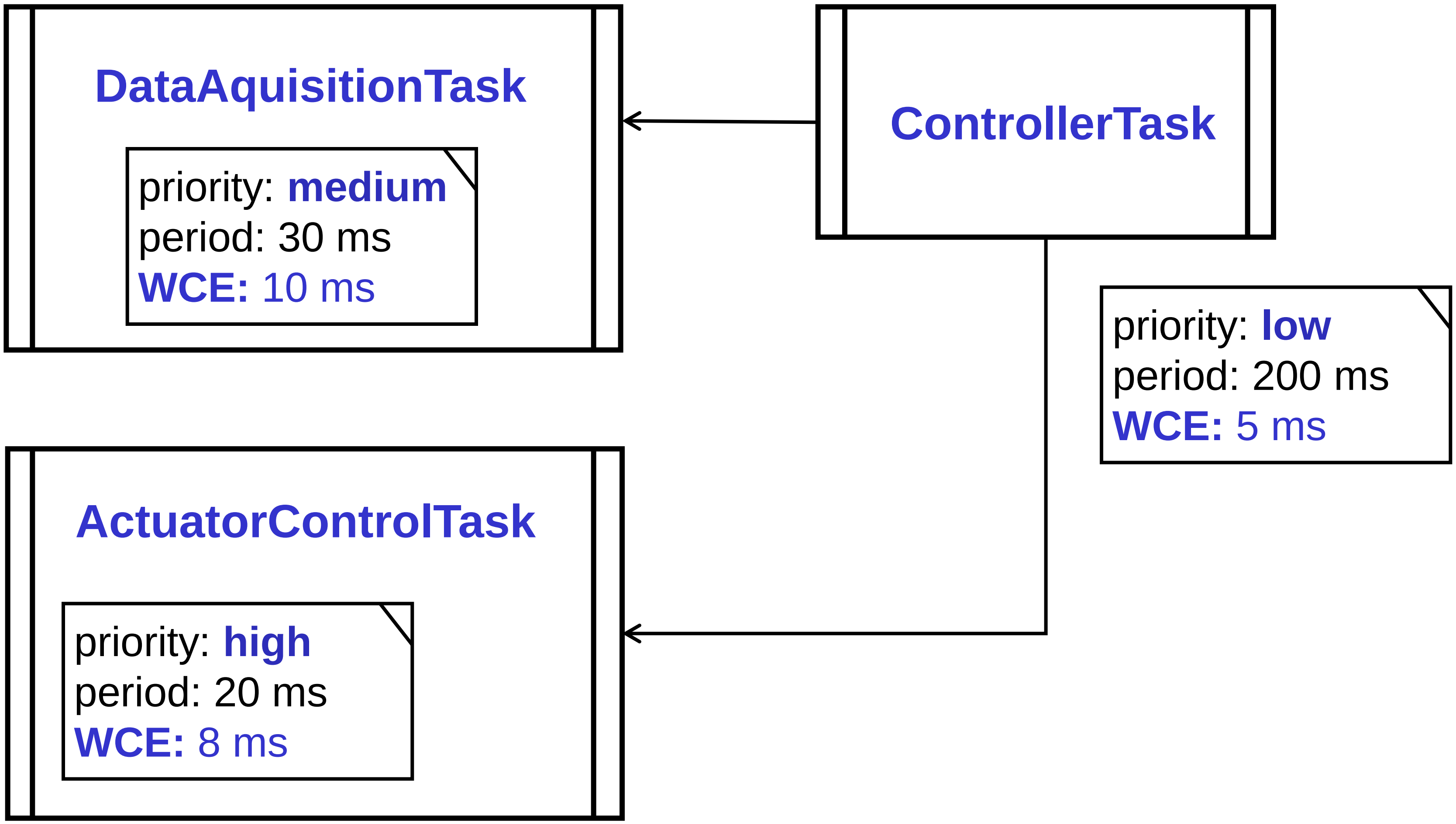
RMS CPU Utilization

- RMS cannot use 100% of CPU, even with zero context switching overhead
- Must keep idle cycles available to handle worst-case scenarios
- However, **RMS guarantees** that all processes will always meet their deadlines **if Theorem #1 is fulfilled**.

Getting a Sense for Schedulability

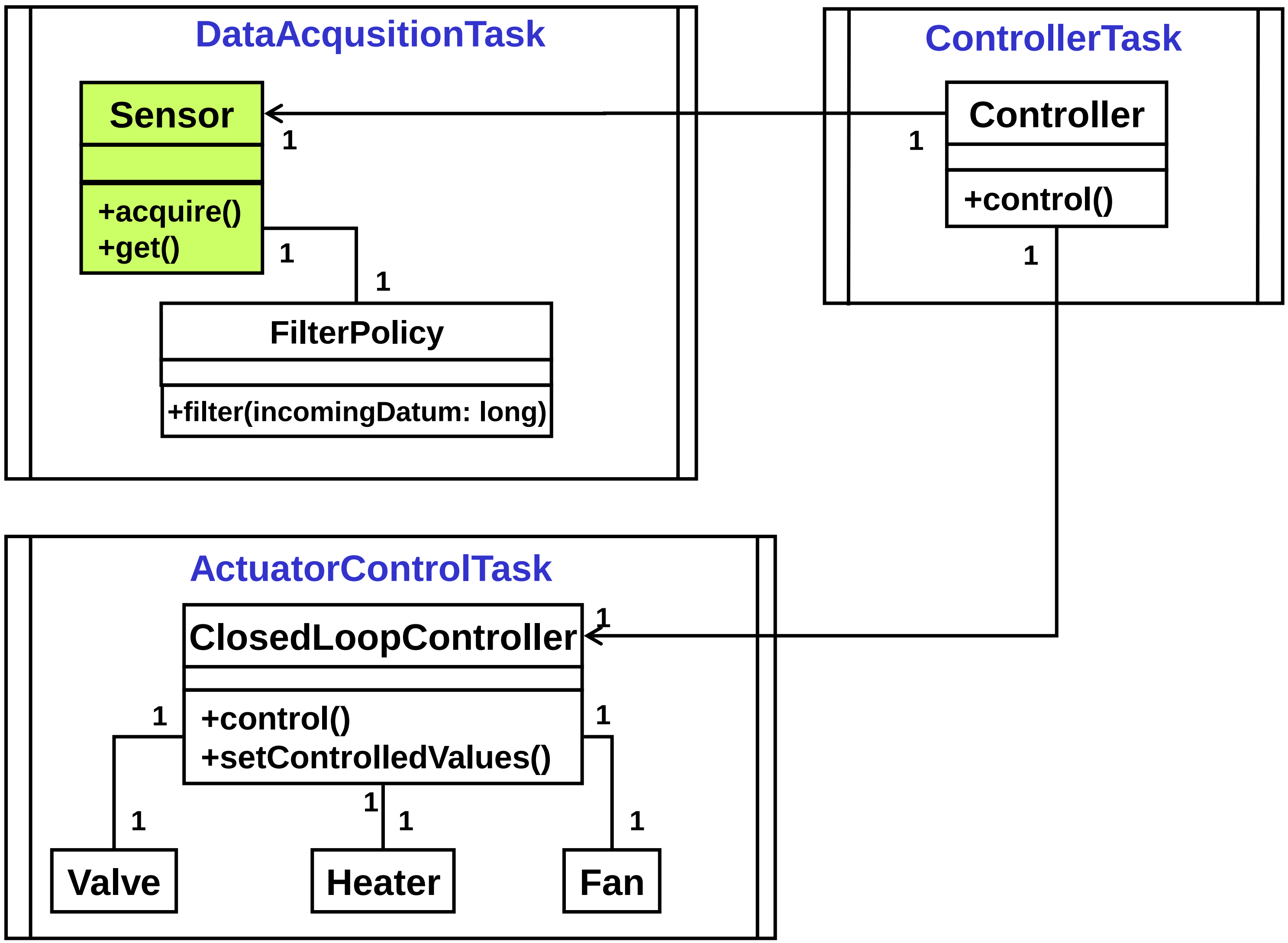
- **Technique 1:** Using one Utilization Bound for the Entire Situation (Theorem 1).
  - A successful test guarantees that timing requirements will be met
  - An unsuccessful test means that a more precise method should be tried
- Assumptions:
  - Deadlines must be equal to the end of the period
  - Responses are assigned rate monotonic priorities (shorter period = higher priority)

Example 1 (1)

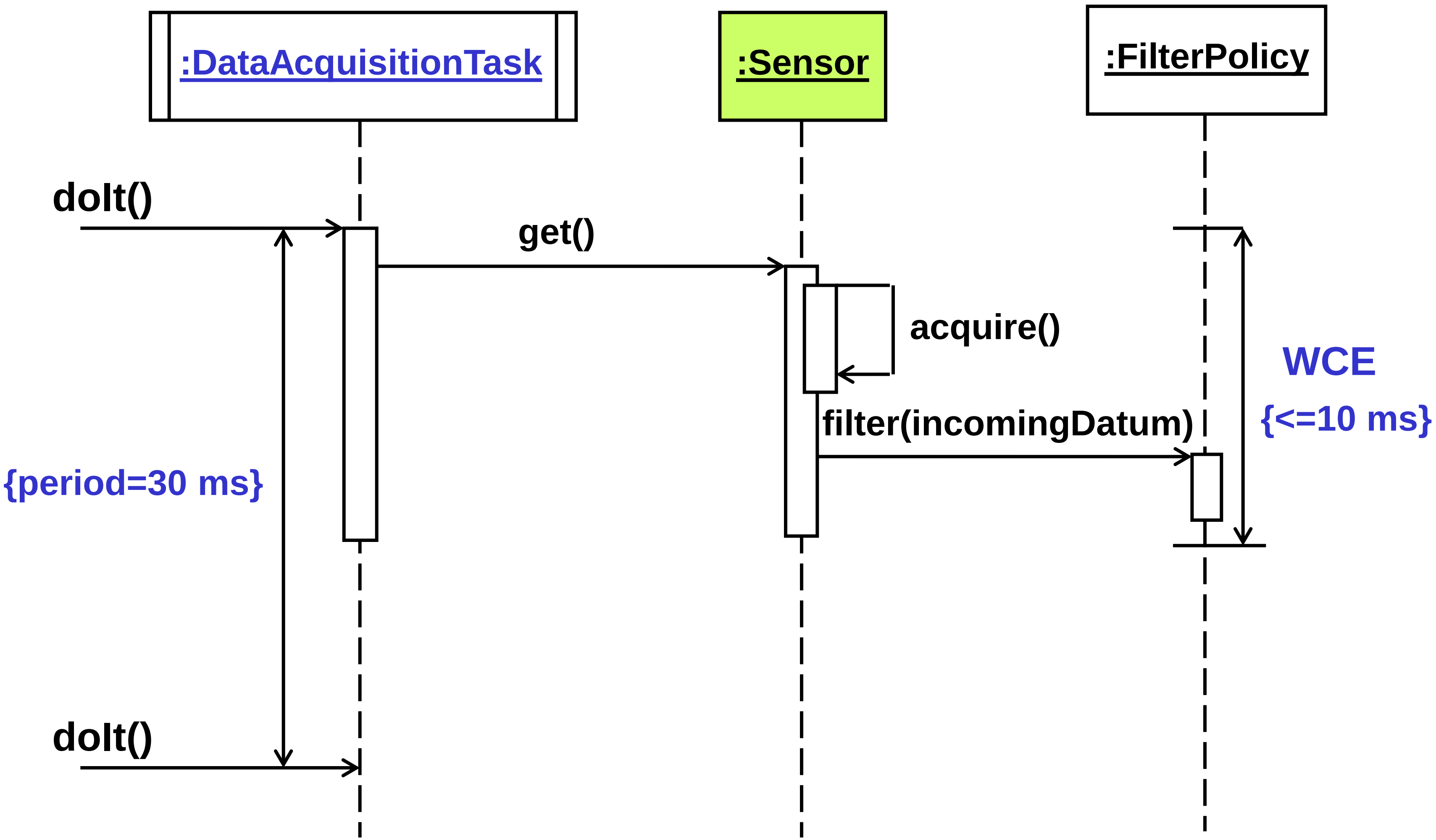


(NB! not quite independent tasks)

Example 1 (2)

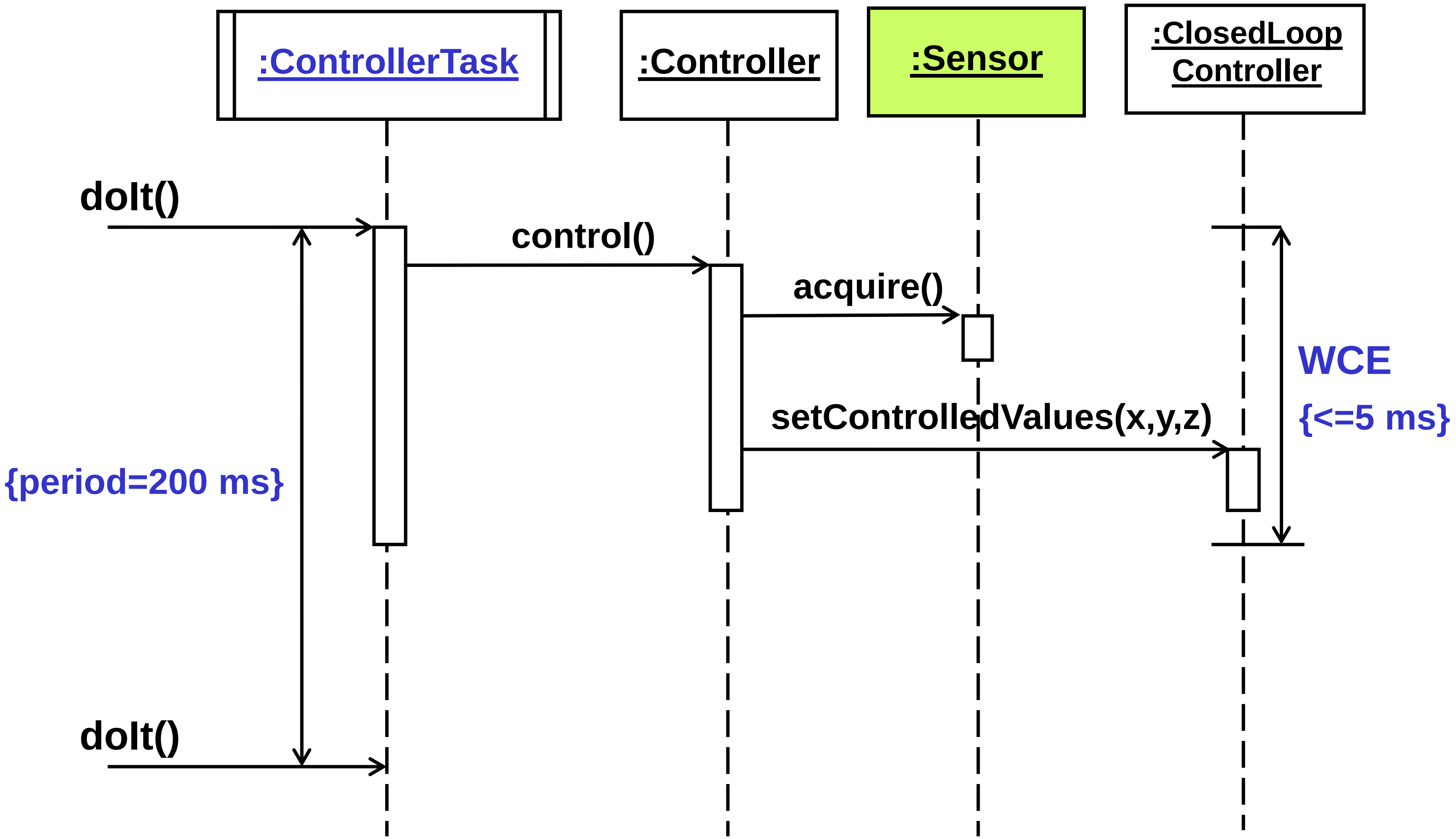

DataAcqusitionTask
Sensor
+acquire()
+get()
1
1
1
FilterPolicy
+filter(incomingDatum: long)
ControllerTask
Controller
+control()
1
1
ActuatorControlTask
ClosedLoopController
+control()
+setControlledValues()
1
1
1
1
1
1
1
1
Valve
Heater
Fan

Example 1 (3)



:DataAcquisitionTask
:Sensor
:FilterPolicy
doIt()
get()
acquire()
WCE
{<=10 ms}
filter(incomingDatum)
{period=30 ms}
doIt()

Example 1 (4)


:ControllerTask
:Controller
:Sensor
:ClosedLoop Controller
doIt()
control()
acquire()
setControlledValues(x,y,z)
WCE
{<=5 ms}
{period=200 ms}
doIt()

Example 1 (5)

Task	**Execution Time (C_i)**	**Period (T_i)**	**C_i /T_i**	**Priority**
Actuator Task	8 ms	20 ms	0.4	High
Data Acq. Task	10 ms	30 ms	0.333	Medium
Control Task	5 ms	200 ms	0.025	Low
Computed utilization			**0.758**	

Schedulable as 0.758 <= $3(2^{1/3}-1)$ = 0.78

AARHUS UNIVERSITY
DEPARTMENT OF ENGINEERING

RMA with Task Blocking
Theorem #2

- Utilization bound, where tasks may be blocked by other lower-priority tasks

$$Ub(n) = \sum_{i=1}^{n} C_i/T_i + \max\left(B_1/T_1, \ldots B_n/T_n\right) \leq n\left(2^{1/n} - 1\right)$$

where

n is the number of periodic tasks
C_i is worst case execution time of Task T_i
B_i is the blocking delay of Task T_i
caused by lower priority tasks

Calculation of Utilization Bound

Step 1. **Calculate the total utilization for all of the events**

$$U_{total} = \sum_{i=1}^{n} C_i/T_i$$

Step 2. **Calculate the blocking term**

$$B_{total} = \max(B_1/T_1, \ldots B_n/T_n)$$

Step 3. **Calculate the utilization bound**

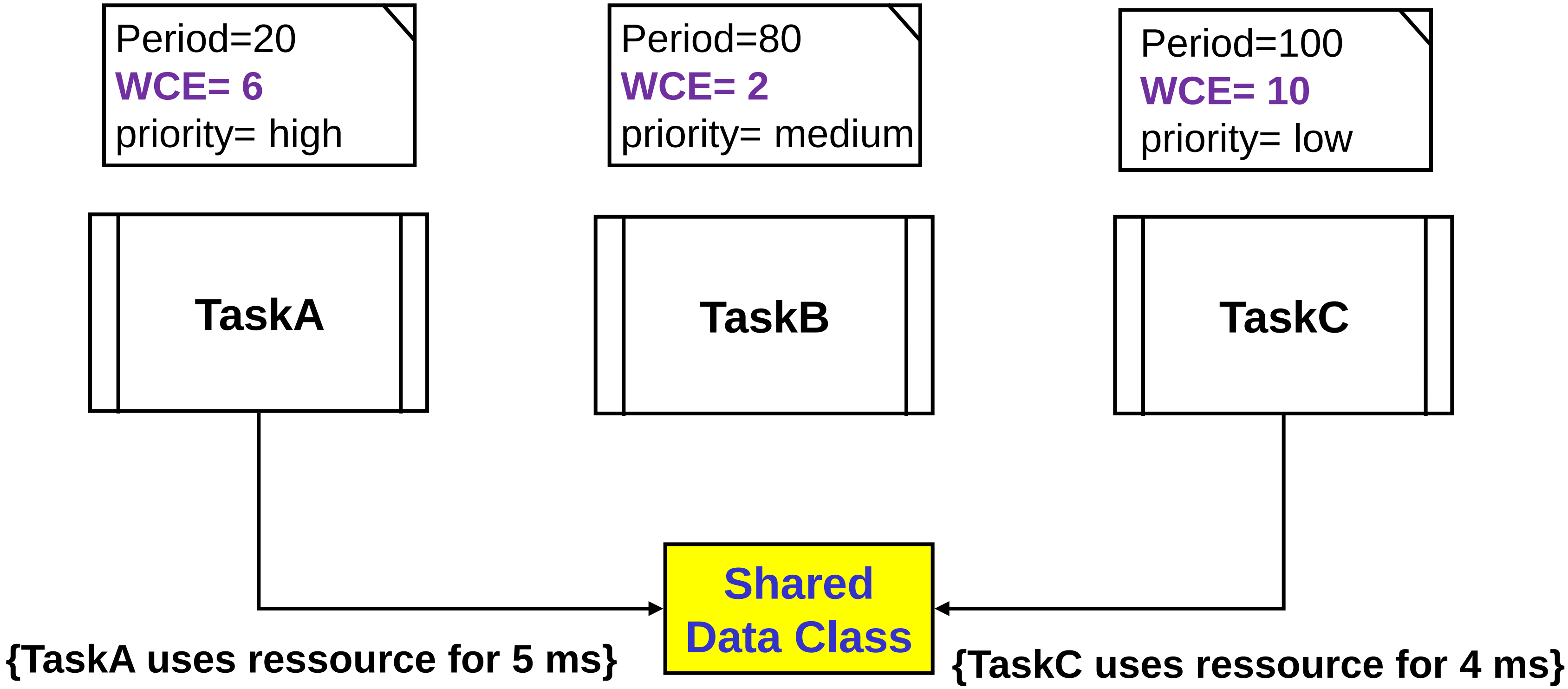
$$Ub(n) = n\,(2^{1/n} - 1)$$

Step 4. **Compare the sum against the utilization bound**

$$U_{total} + B_{total} <= Ub(n)$$

AARHUS UNIVERSITY
DEPARTMENT OF ENGINEERING

Example 2 (with priority inversion)



Blocking:
TaskC can block TaskA for 4 ms
plus 2 ms from TaskB
(caused by priority inversion)

$U_{total}=0.425$

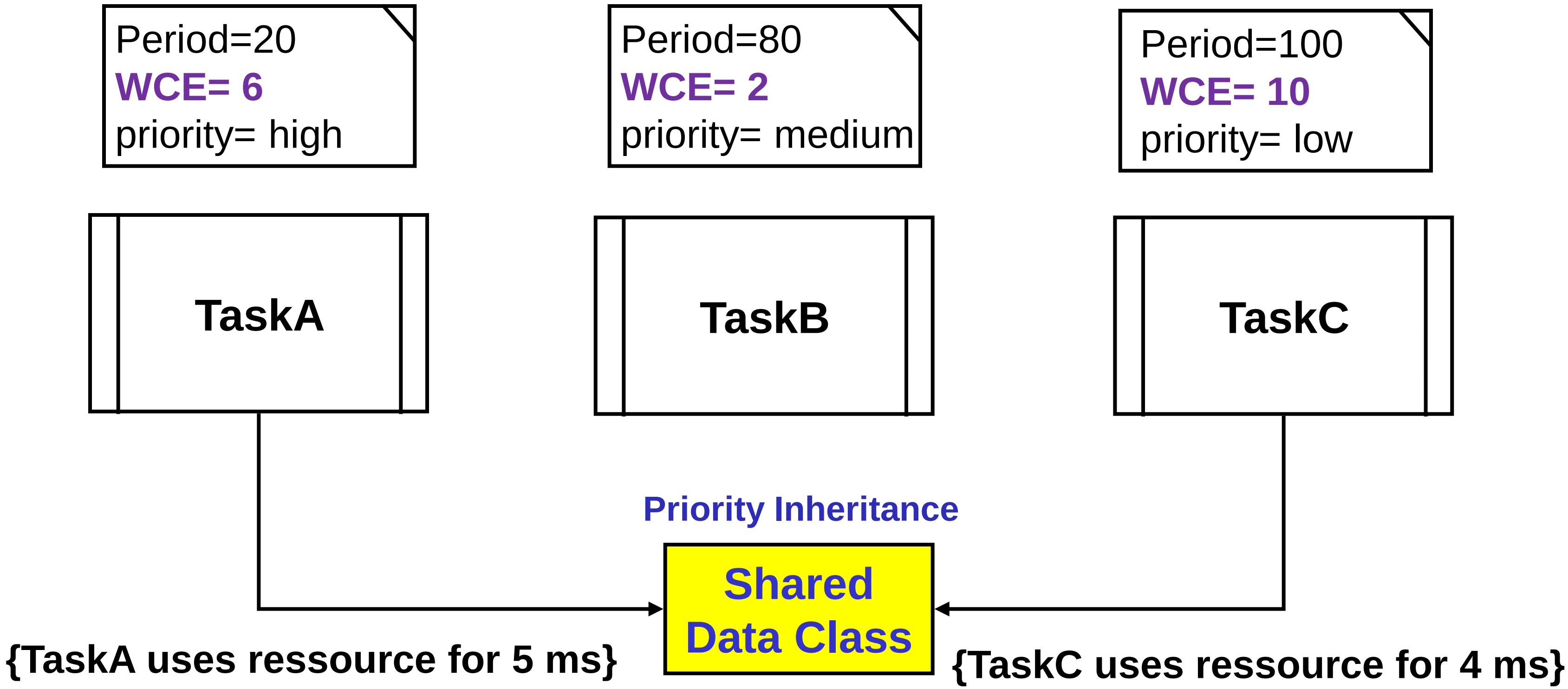
$B_{total}=6/20=0.3$

$U_{total} + B_{total} <= Ub(n)$

$0.725 <= 0.758$

AARHUS UNIVERSITY
DEPARTMENT OF ENGINEERING

Example 2 (without priority inversion)



Blocking:
TaskC can block TaskA for 4 ms
TaskB can be blocked 4 ms from TaskC due to priority inheritance

Utotal=0.425
Btotal= max (4/20,4/80)=0.2
Utotal + Btotal <= Ub(n)
0.625 <= 0.758

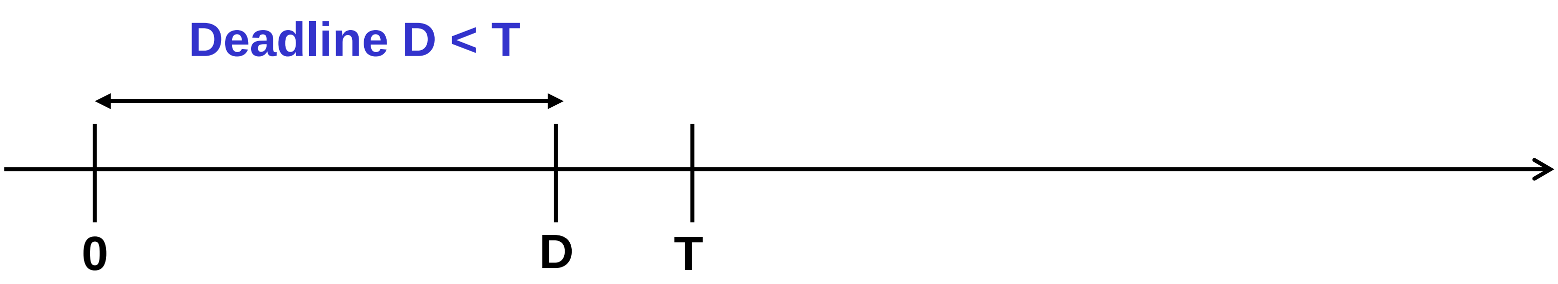
Example 3: Calculation of Utilization Bound

Event ID (e)	Arrival Period (T)	Execution Time (C)	Priority (P)	Blocking Delays (B)	Deadline (D)
e1	**40**	4	Very High	0	**40**
e2	**150**	10	High	15	**150**
e3	**180**	20	Medium	0	**180**
e4	**250**	10	Low	5	**250**
e5	**300**	80	Very Low	0	**300**

1. $U_{total} = 4/40 + 10/150 + 20/180 + 10/250 + 80/300 = 0.59$
2. $B_{total} = \max(15/150, 5/250) = 0.10$
3. $UB(5) = 0.743$
4. $U_{total} + B_{total} = 0.69 <= 0.743$

Calculation of Utilization Bound (UB) for each Event Sequence

- Technique used when **the deadlines are within the period (D < T)**
- The following **4 steps** are applied **to each event sequence e_i** that has a response time requirement (a deadline)



Example 4: test the Schedulability of e4

Event ID (e)	**Arrival Period (T)**	**Execution Time (C)**	**Priority (P)**	**Blocking Delays (B)**	**Deadline (D)**
e1	**40**	4	Very High	0	**10**
e2	**300**	80	High	0	**300**
e3	**180**	20	Medium	0	**140**
e4	**250**	10	Low	5	**150**
e5	**150**	10	V.Low	0	**150**

Notice: Rate Monotonic assignment of priorities is not required

UB Algorithm for Event e_i (1)

- **Step 1: Identify H**
  - Identify **H** as the set of event sequences with priorities (P) higher than P_i the priority of e_i
    - **Example e4: H: (e1, e2, e3)**
  - Partition the events in **H** in two sets $\mathbf{H_1}$ and $\mathbf{H_n}$
  - $\mathbf{H_1}$ = set of events with arrival periods (T) >= $\mathbf{D_i}$ ($\mathbf{D_i}$ =deadline of e_i)
    - **Example with e4: D4= 150 ms, H_1= (e2,e3)**
  - $\mathbf{H_n}$ = set of events with arrival periods (T) < $\mathbf{D_i}$ ($\mathbf{D_i}$ =deadline of e_i)
    - **Example e4: D_4= 150 ms, H_n= e_1**

UB Algorithm for Event e_i (2)

- **Step 2: Calculate f_i**, the total effective utilization for event e_i

$$f_i = \left(\sum_{j \in H_n} \frac{C_j}{T_j} \right) + \frac{1}{T_i} \left(C_i + B_i + \sum_{k \in H_1} C_k \right)$$

The total utilization of events in H_n (i.e. arrival periods < D_i) plus the execution time of C_i, added to the blocking delay B_i, added to preemption from events in set H_1, all divided by the period of e_i (T_i).

Example:

$$f_4 = C_1/T_1 + 1/T_4\,(C_4 + B_4 + C_2 + C_3)$$
$$= 4/40 + 1/250(10+5+80+20) = 0.1+0.46 = 0.56$$

UB Algorithm for Event e_i (3)

- **Step 3: Determine the utilization bound $U(n, \Delta_i)$**
- n= number of elements in H_n plus 1
- $\Delta_i = D_i / T_i <= 1.0$

$$U(n, \Delta_i) = \begin{cases} n\left((2\Delta_i)^{1/n} - 1\right) + 1 - \Delta_i, & 0.5 < \Delta_i \le 1 \\ \Delta_i, & \Delta_i \le 0.5 \end{cases}$$

Example:

$n = 1+1 = 2$, $\Delta_4 = 150/250 = 0.6$
$U(2, 0.6) = 0.59$

UB Algorithm for Event e_i (4)

- **Step 4: Compare the effective utilization f_i with the utilization bound $U(n,\Delta_i)$**
- If $f_i <= U(n, \Delta_i)$ then

  event sequence e_i will meet its deadline

Example:

$f_4 = 0.56 <= U(2, 0.6) = 0,59$

=> Event sequence e_4 will meet its deadline

Class Exercise

For the data on slide 24, where f_4 was proved to be schedulable:

- Calculate f_2 and compare with utilization band **$U(n,\Delta_2)$**
- **Is f_2 schedulable ?**
- Calculate f_3 and compare with utilization band **$U(n,\Delta_3)$**
- **Is f_3 schedulable ?**
- Calculate f_5 and compare with utilization band **$U(n,\Delta_5)$**
- **Is f_5 schedulable ?**

Demo of Times tool

- Times is a research tool for schedulability analysis
- Developed by the DARTS (Design and Analysis of Real-Time Systems) research group at Upsala University
- http://www.timestool.com/
- Download version **Times 1.3 beta** (last updated nov. 2008).
- Run downloaded version from a command prompt by typing: **java –jar timestool.jar**

Summary

- RMS & RMA is a relatively simple technique to use
  - Requires WCET values for operations
  - Assumes periodic tasks
  - One technique: if deadline == period
  - Another technique: if deadline < period
- Times tool demo