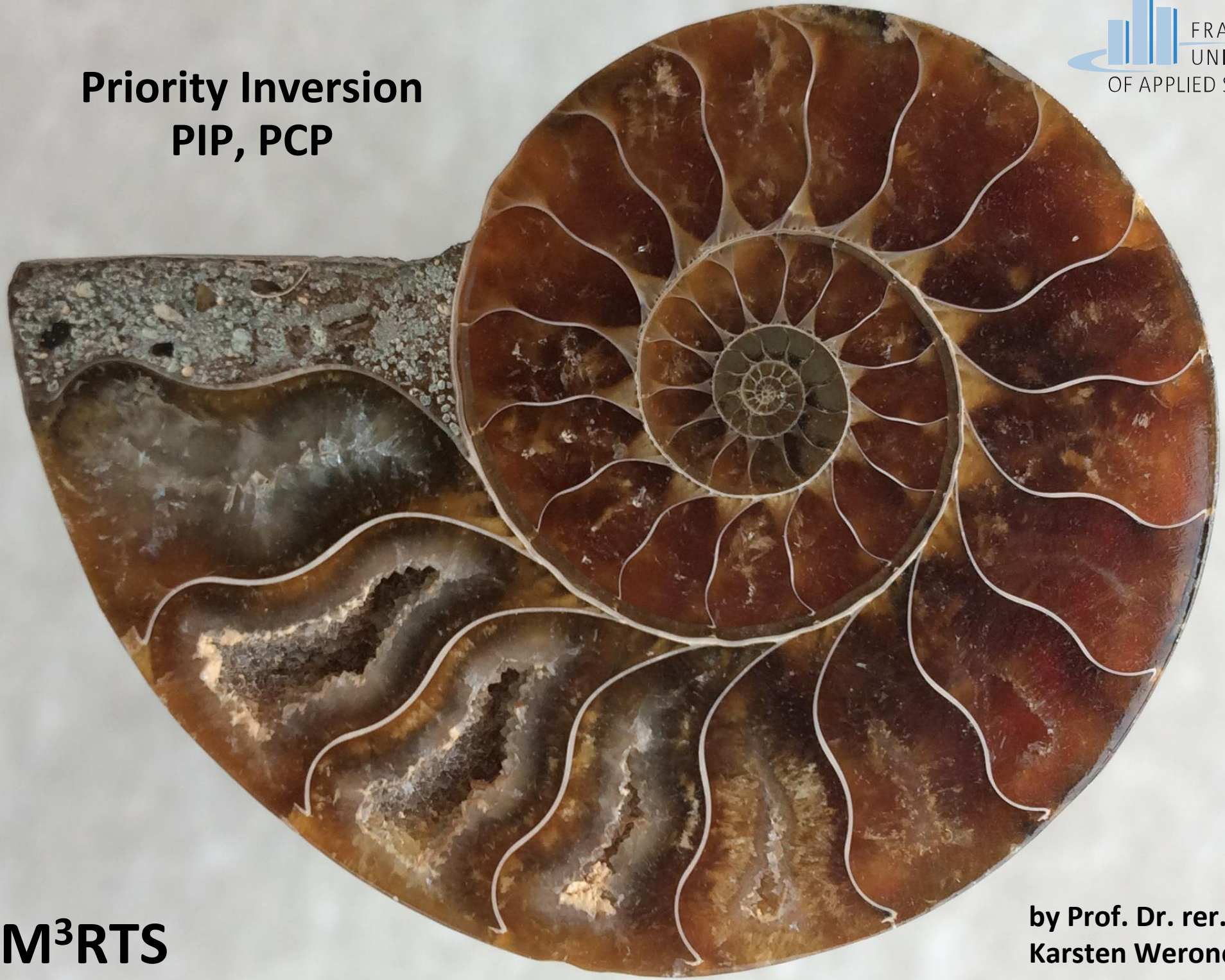
# Priority Inversion
# PIP, PCP



FRANKFURT UNIVERSITY OF APPLIED SCIENCES

**M³RTS**

**by Prof. Dr. rer. nat. Karsten Weronek**

# M³Real-Time-Systems
# SS 2017

**Prof. Dr. rer. nat. Karsten Weronek**
**Faculty 2**
**Computer Science and Engineering**

## Priority Inversion

## Priority Inheritance Protocol (PIP)

## Priority Ceiling Protocol (PCP)

# Priority Inversion

Priority Inversion occurs when

a high priority task requiring a resource is blocked

due to the lock of this resource by a low priority task.

Then the high priority task is blocked

until the low priority task is completed and has released the resource.

This can take a long time,

especially when mid-prioritized tasks delay

the completion of the low prioritized one.

# Priority Inversion Example

There are 3 periodic Tasks $T^k = (t_0; \Delta t_{exec}^k; \Delta t_{per}^k; \Delta t_{allocR}^k; \Delta t_{useR}^k)$; k=1 ... 3
$T^1(0; 3; 36; 1; 2)$; $T^2(3, 16; 32; 0; 0)$; $T^3(2; 3; 18; 1; 2)$

The load of the CPU = 3/36 + 16/33 + 3/18 = 75,0%.

The load of the Resource = 1/36 + 2/18 = 11,3%.

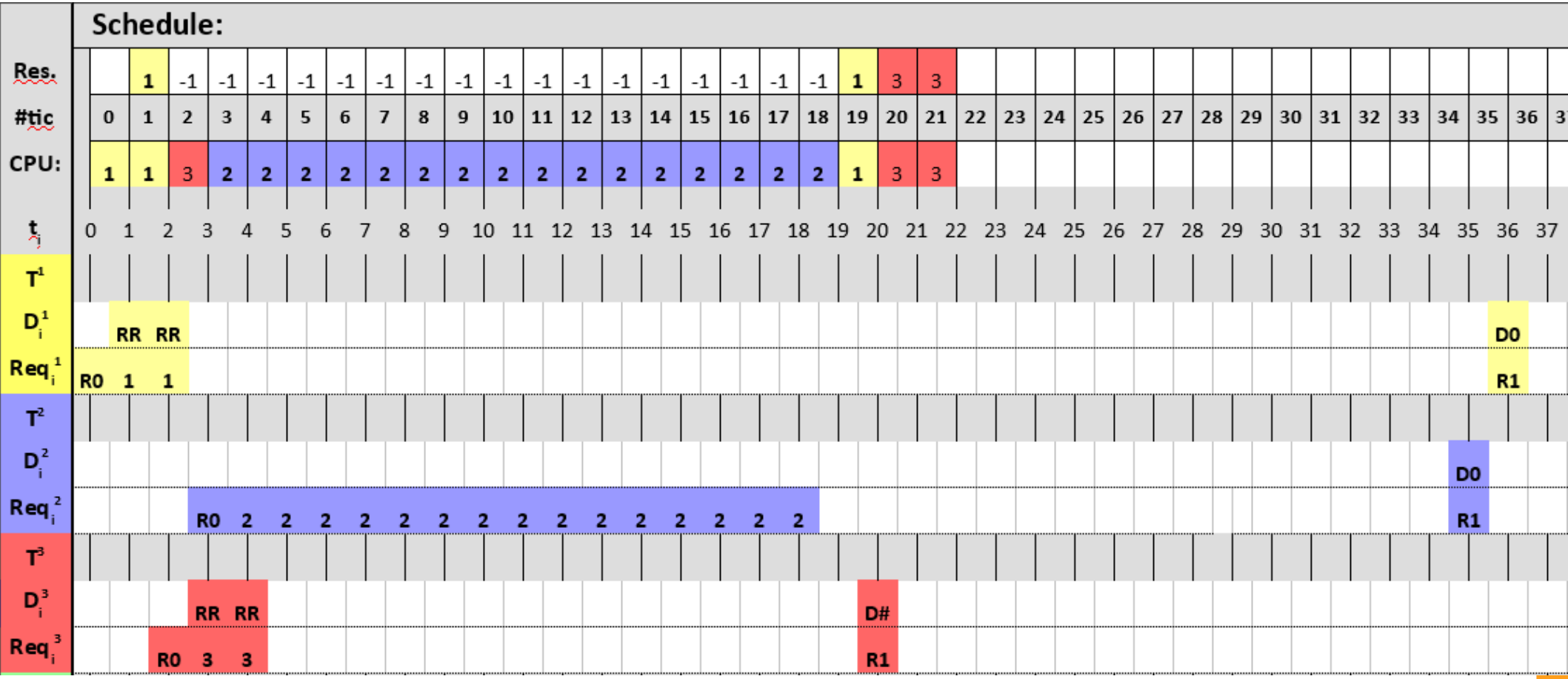
Since the load is below 1 → the **necessary schedulability condition** is fulfilled!

since the average_load = 75,0% ≤ max_load = $n(2^{(1/n)}-1) = 3(2^{(1/3)}-1) \leq 77,97\%$
→ the **sufficient schedulability condition** is fullfilled:

⇒ **feasible schedule exists!**

Using RMS the priorities are $T^1$: lowest; $T^2$ middle; $T^3$ highest

# Priority Inversion (example)




Using RMS. The priorities are $T^1$ : lowest; $T^2$ middle; $T^3$ highest.

T2 with middle priority uses the CPU although T1 with the highest priority is requested!

This leads to that the deadline $D_0$ of $T^3$ is missed!

# To prevent blocking due to Priority Inversion



There are three possibilities to „heal“ Priority Inversion:

1. NPCS (Non Preemptive Critical Section) „Unterbrechungssperre“
2. Priority Inheritance Protocol (PIP)
3. Priority Ceiling Protocol (PCP)

1. Ensures that tasks will be completed without interruption. Implemented for each layer of the Preemption model.
2. Avoids blocking by Priority Inheritance however issues like chaining („cascaded delays“) or deadlocks
3. Enhancement of the PIP without the issues

# Priority Inheritance Protocol (PIP)

A low priority task holding a critical resource that
is requested by a higher prioritized task
inherits the priority of the higher prioritized task.

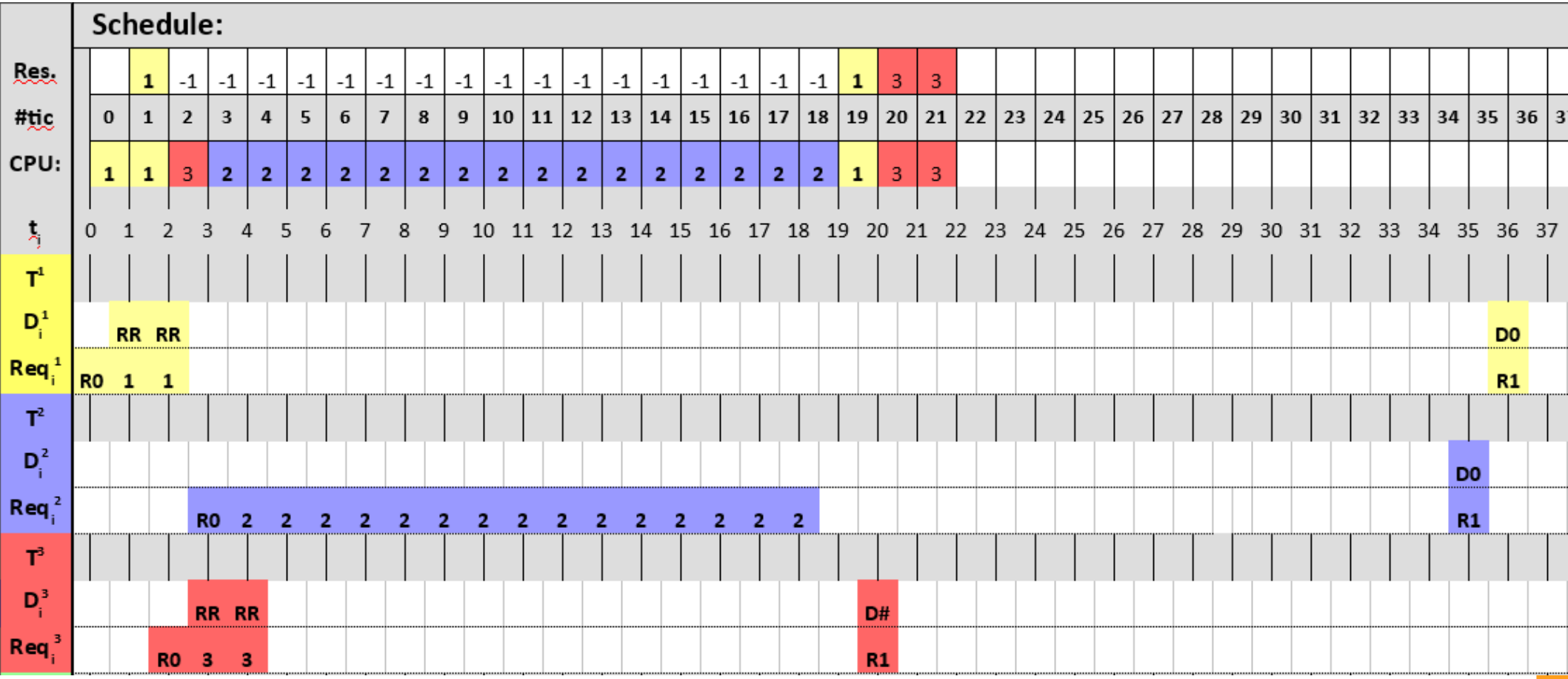
After completion of the access to the critical resource
the priority will be set back to the initial value.

# Priority Ceiling Protocol (PCP)

The Priority Ceiling Protocol (PCP) is
an enhancement of the Priority Inheritance Protocol (PIP).

In case of a priority inversion
the job having allocated a resource
inherits the highest priority of
all tasks that are using the resource.

# Priority Inversion (example)



# Priority Inheritance