Fixed-Priority Multiprocessor Scheduling with Liu & Layland's Utilization Bound

Nan Guan, Martin Stigge, Wang Yi

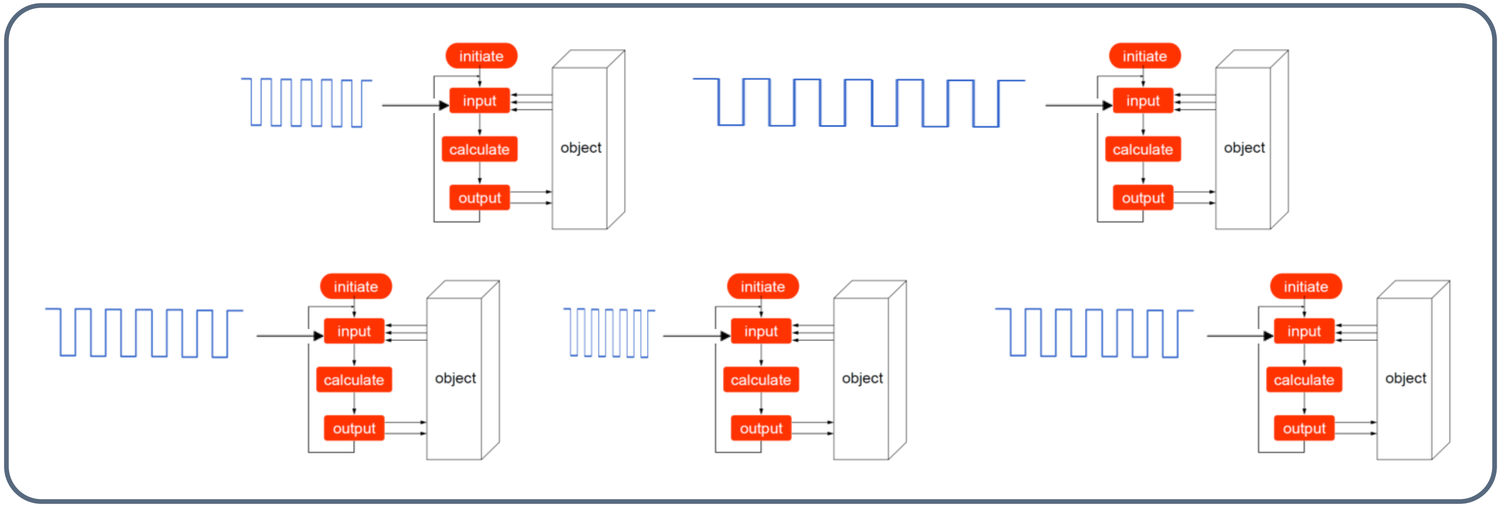
Uppsala University, Sweden

Outline

- Problem
- Previous Results
- Our New Result

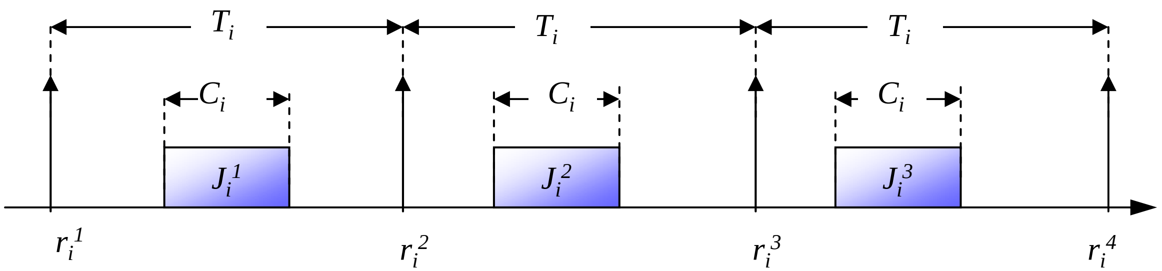
Scheduling of Multi-task System

- multi-rate real-time task system



- each task

Utilization: C_i/T_i



Liu and Layland's Utilization Bound

- Liu and Layland's utilization bound for **single-processor scheduling** [Liu1973]
  (the 19th most cited paper in computer science)

$$\Theta(N) = N(2^{\frac{1}{N}} - 1)$$

- N: the number of tasks, $N \to \infty$, $\Theta(N) \doteq 69.3\%$
- optimal

$$\sum C_i/T_i \leq N(2^{1/N} - 1)$$

$$\Rightarrow \quad \text{the task set is schedulable}$$

Multiprocessor Scheduling

Significantly more difficult

- Bin-packing problem
- Hard to identify the worst-case scenario
- Suffer from timing anomalies
- May lead to arbitrarily low utilization

Open Problem

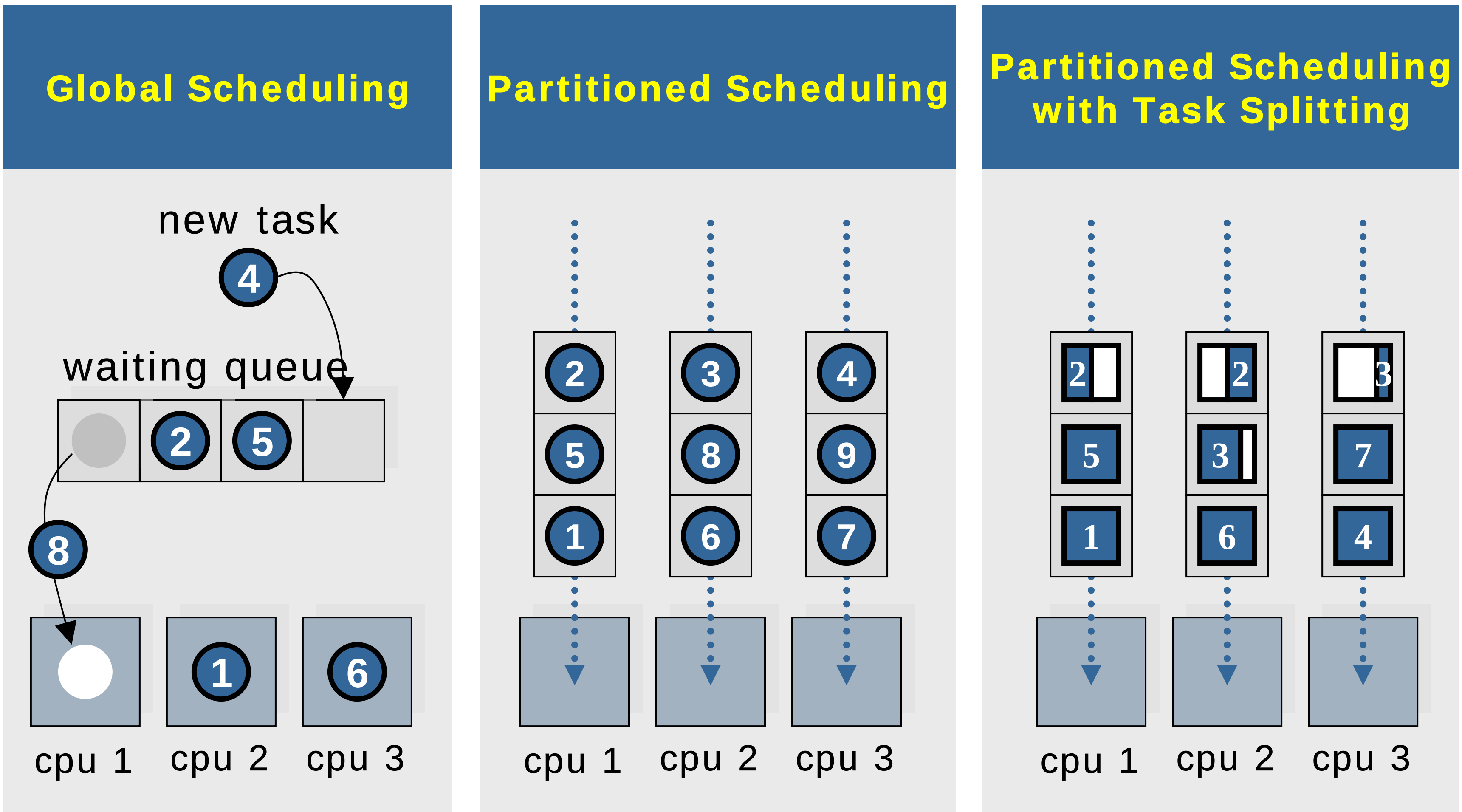
- find a multiprocessor scheduling algorithm that can achieve Liu and Layland's utilization bound

$$\frac{\sum C_i/T_i}{M} \leq N(2^{1/N} - 1)$$

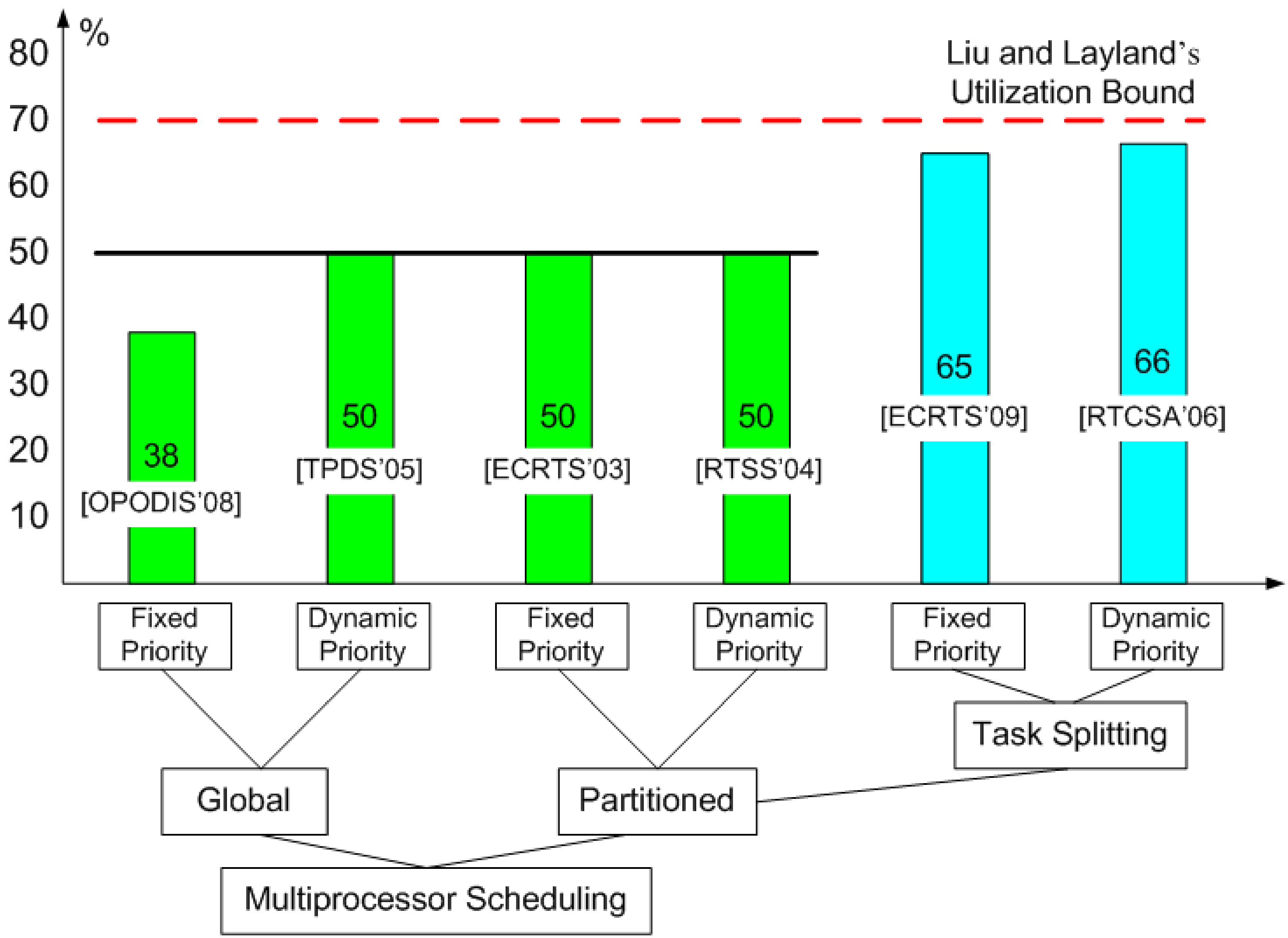
$$\Rightarrow \quad \text{the task set is schedulable}$$

number of processors

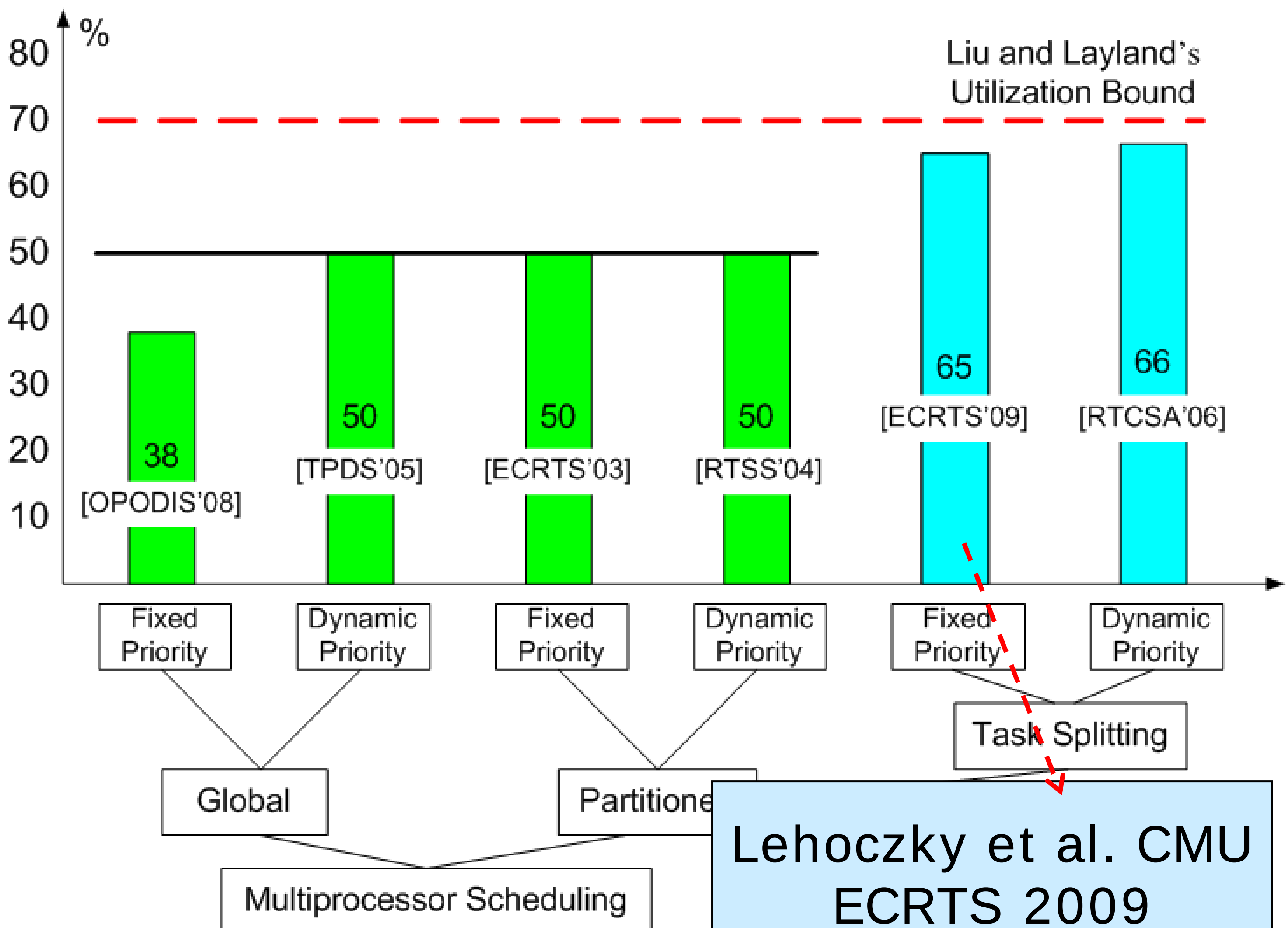
Multiprocessor Scheduling



Best Known Results

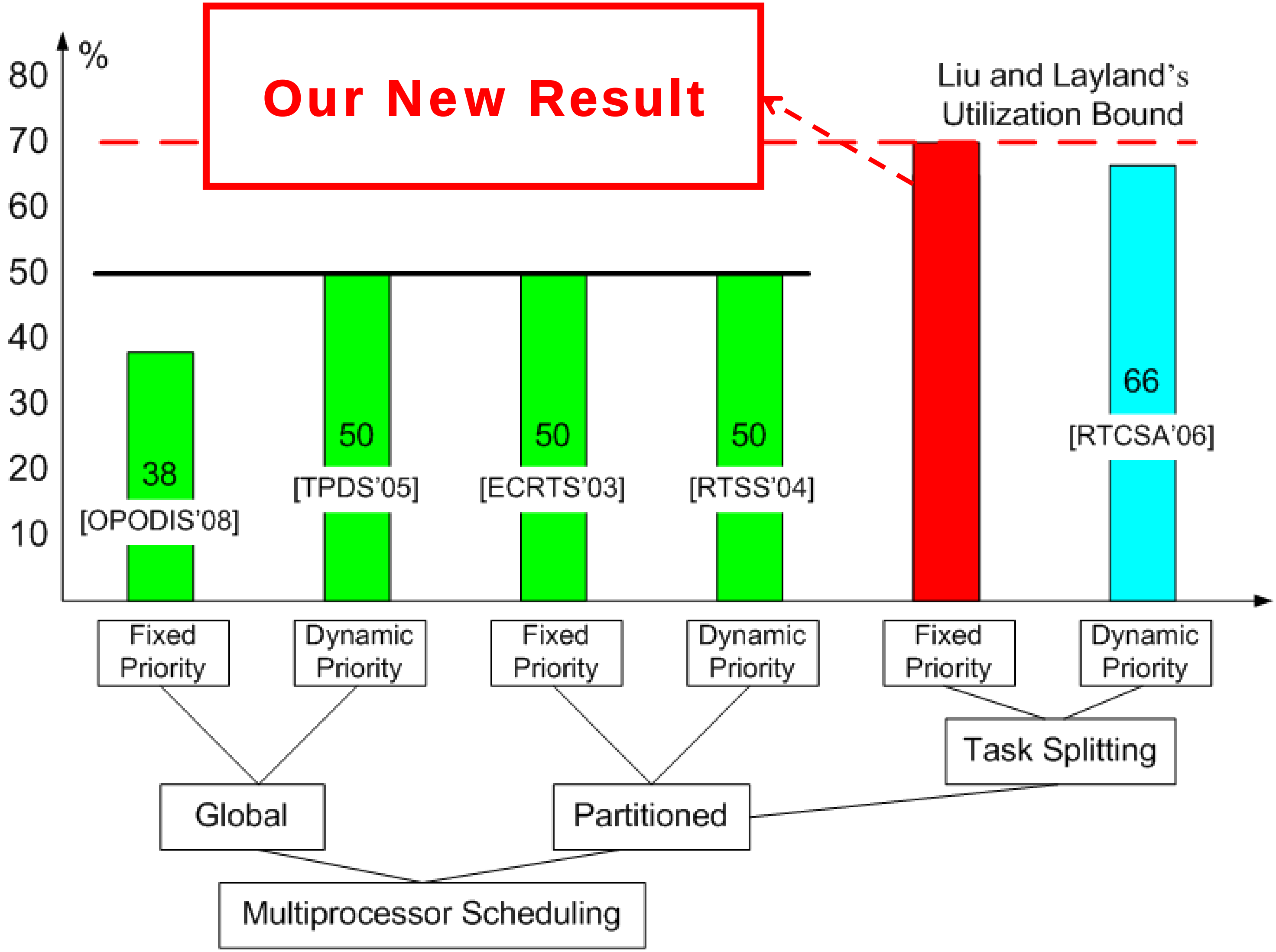


Best Known Results



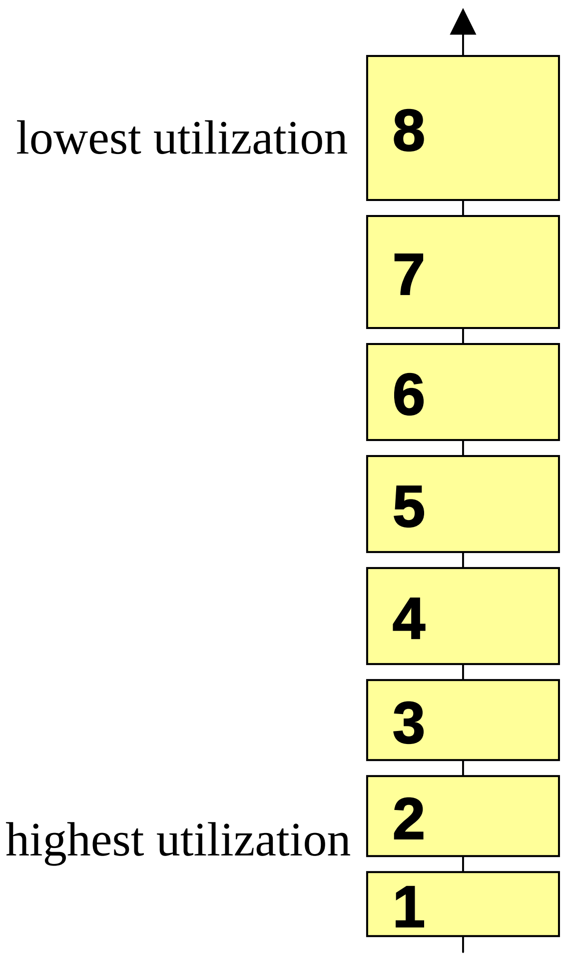
Lehoczky et al. CMU
ECRTS 2009

Best Known Results



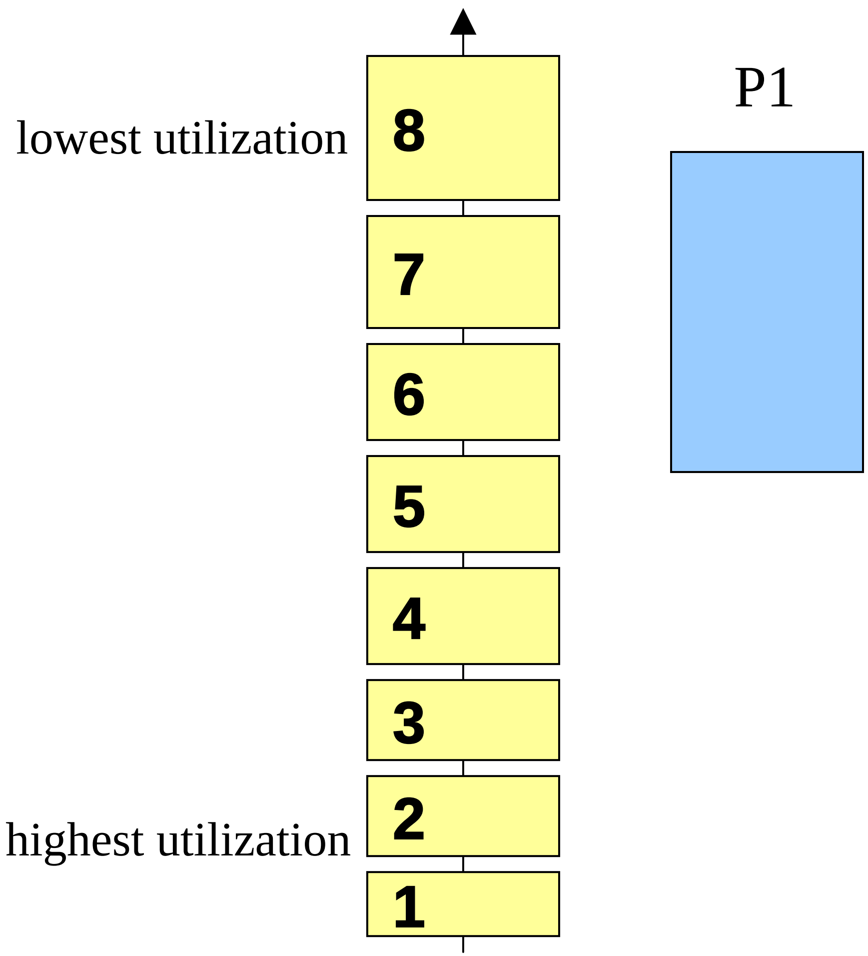
Lehoczky's Algorithm [ECRTS'09]

- sort all tasks in decreasing order of utilization



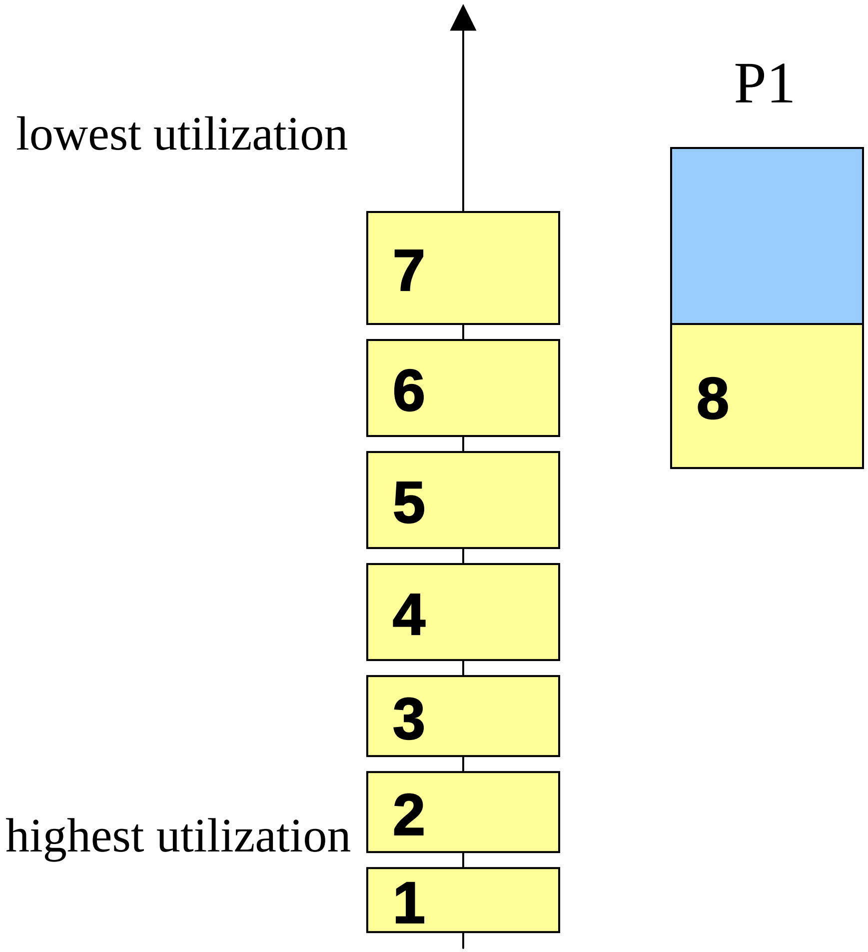
Lehoczky's Algorithm [ECRTS'09]

- pick up one processor, and assign as many tasks as possible



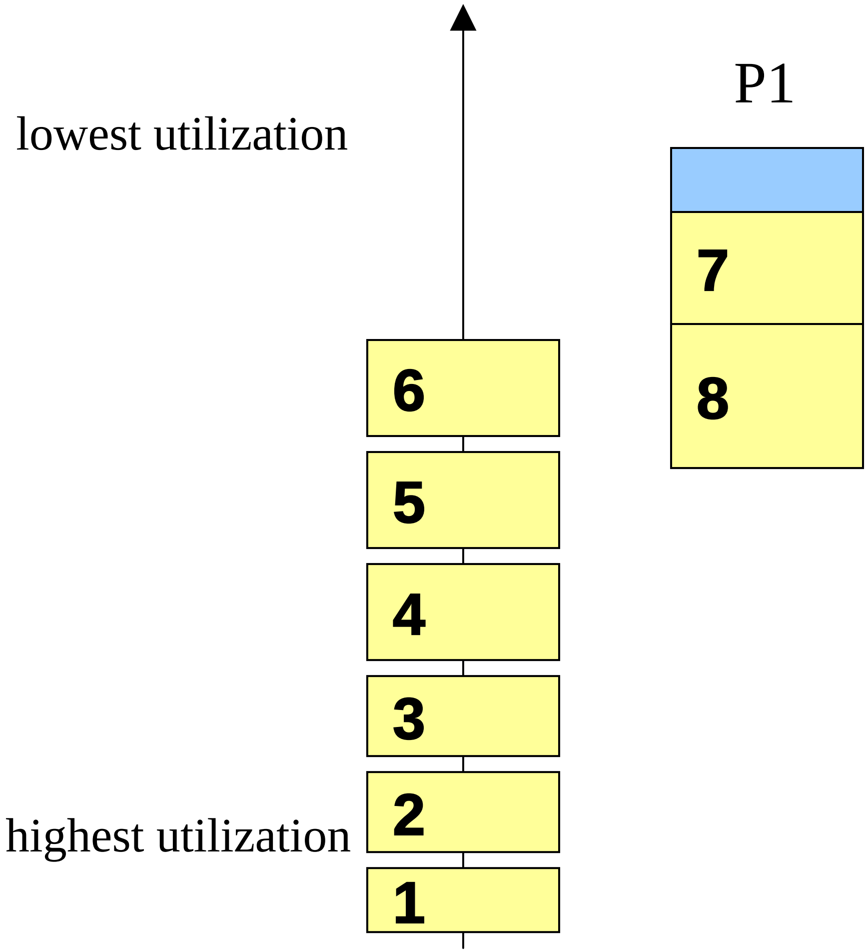
Lehoczky's Algorithm [ECRTS'09]

- pick up one processor, and assign as many tasks as possible



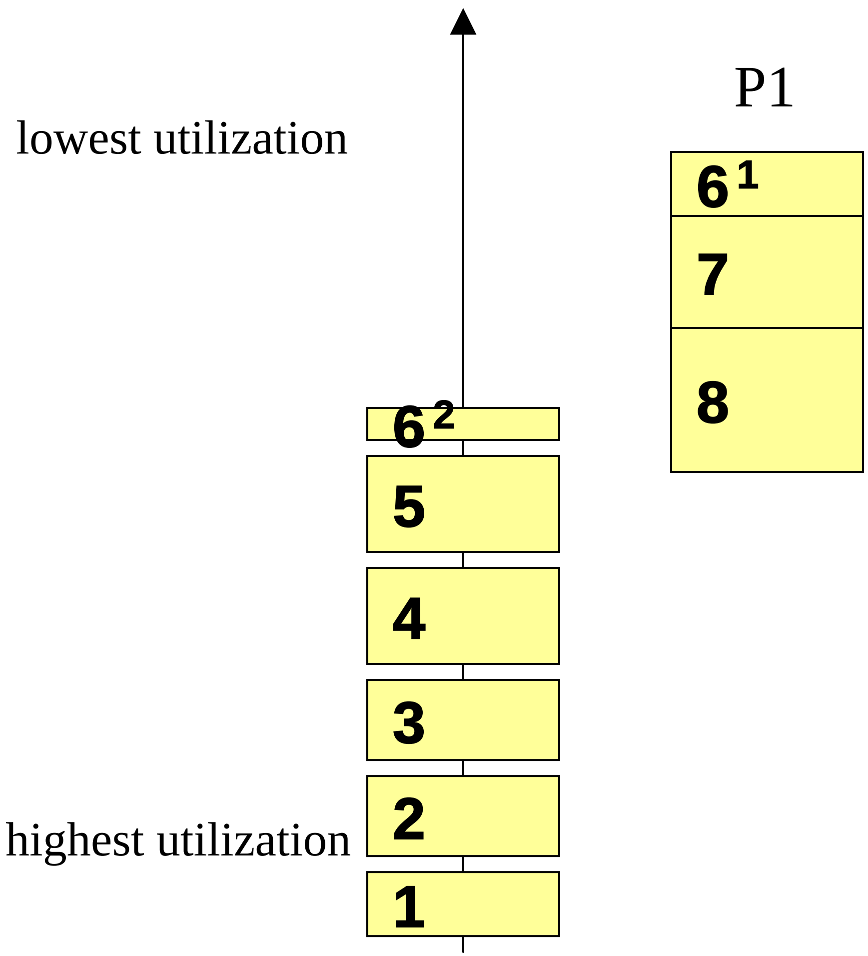
Lehoczky's Algorithm [ECRTS'09]

- pick up one processor, and assign as many tasks as possible



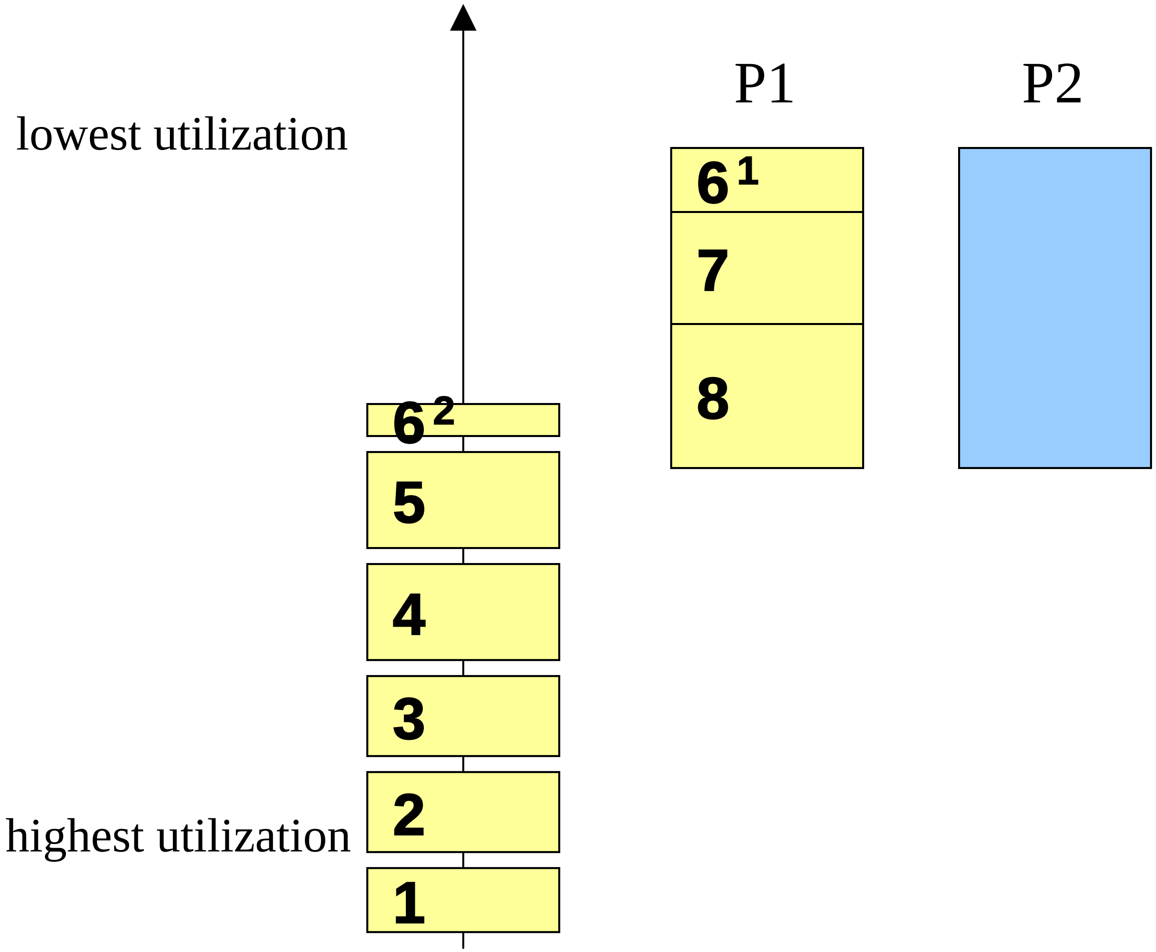
Lehoczky's Algorithm [ECRTS'09]

- pick up one processor, and assign as many tasks as possible



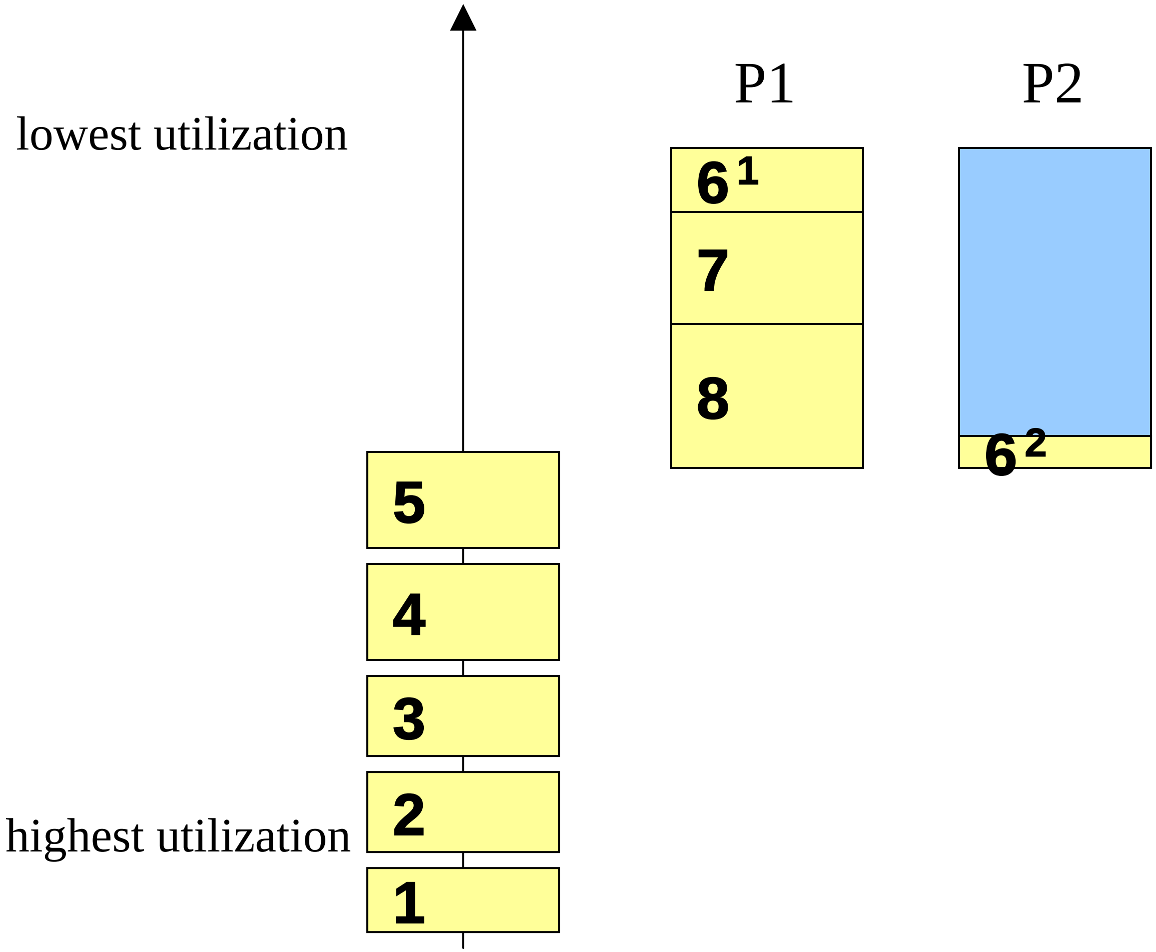
Lehoczky's Algorithm [ECRTS'09]

- pick up one processor, and assign as many tasks as possible



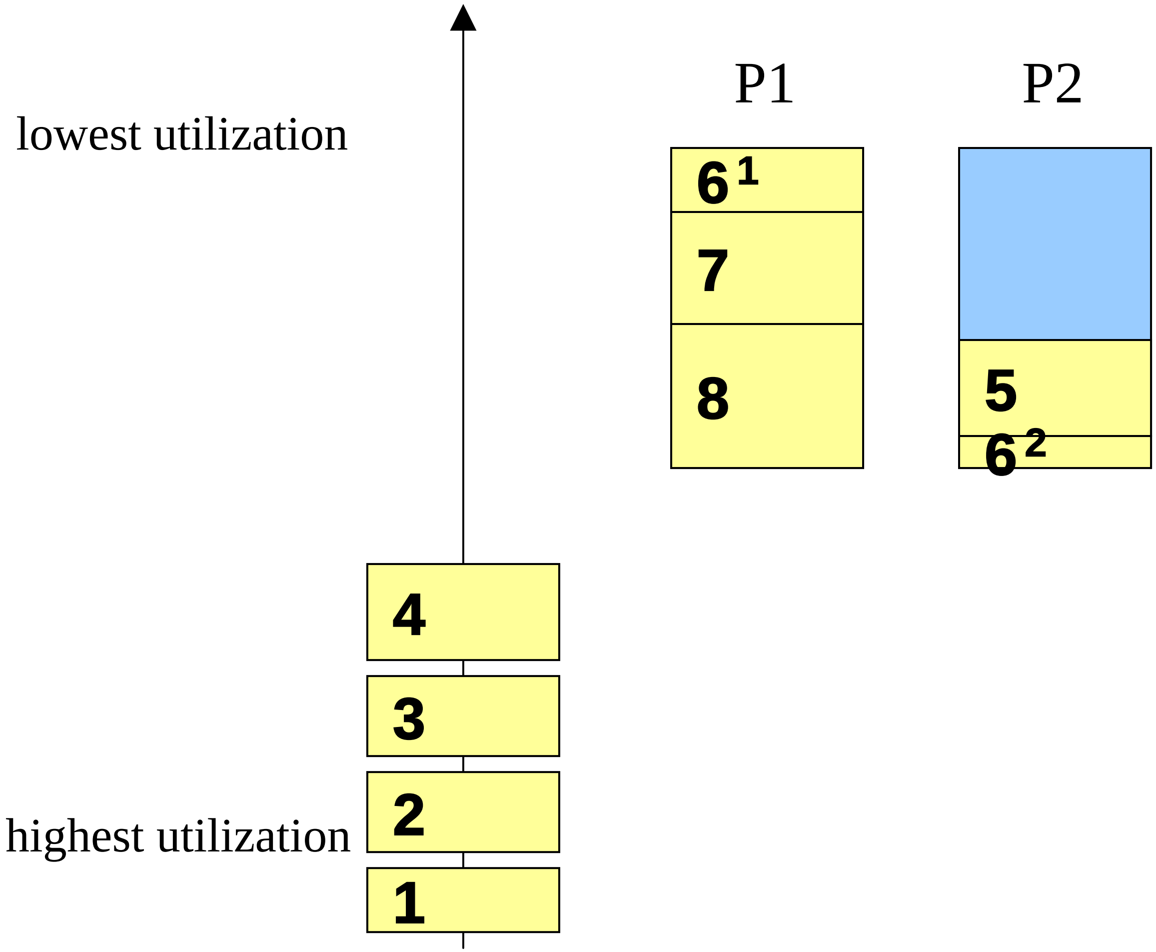
Lehoczky's Algorithm [ECRTS'09]

- pick up one processor, and assign as many tasks as possible



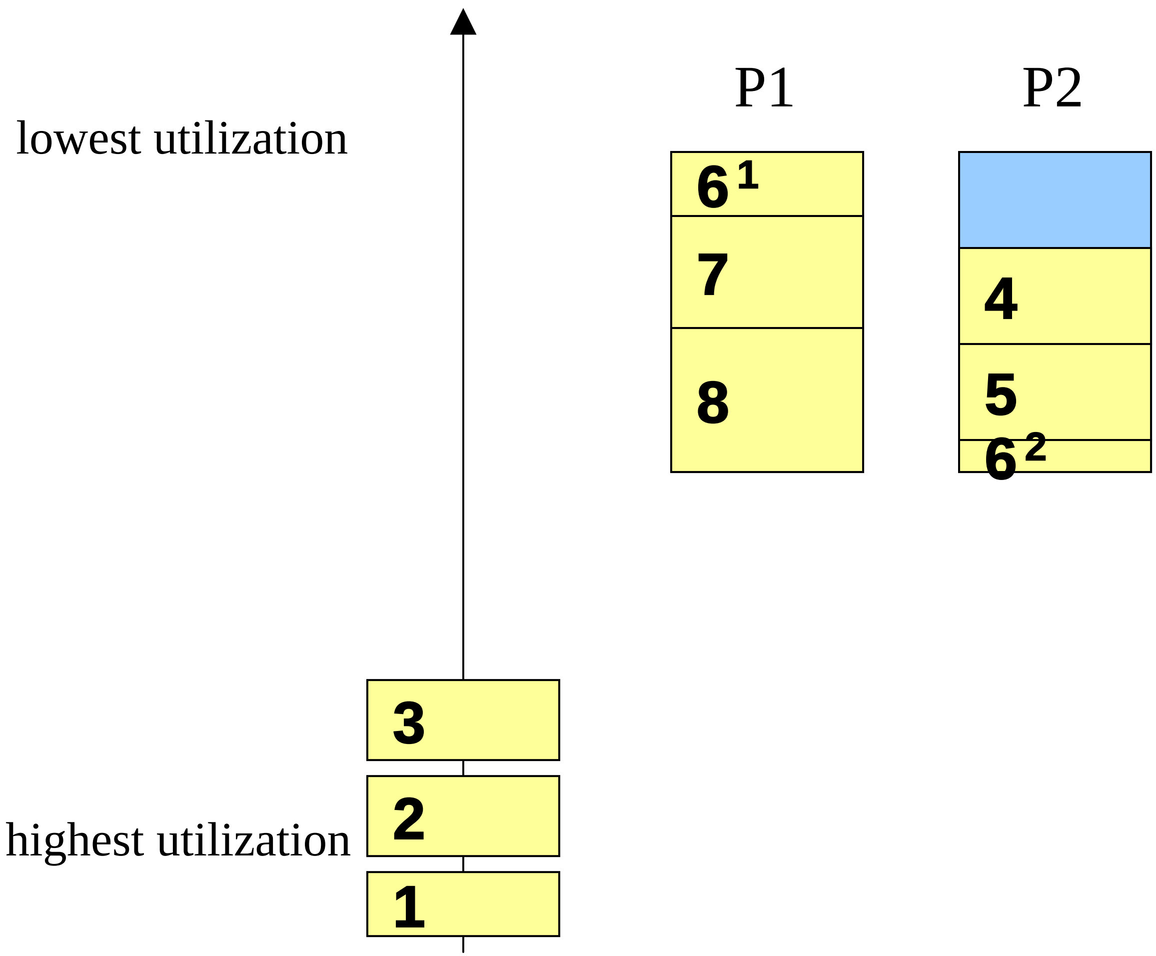
Lehoczky's Algorithm [ECRTS'09]

- pick up one processor, and assign as many tasks as possible



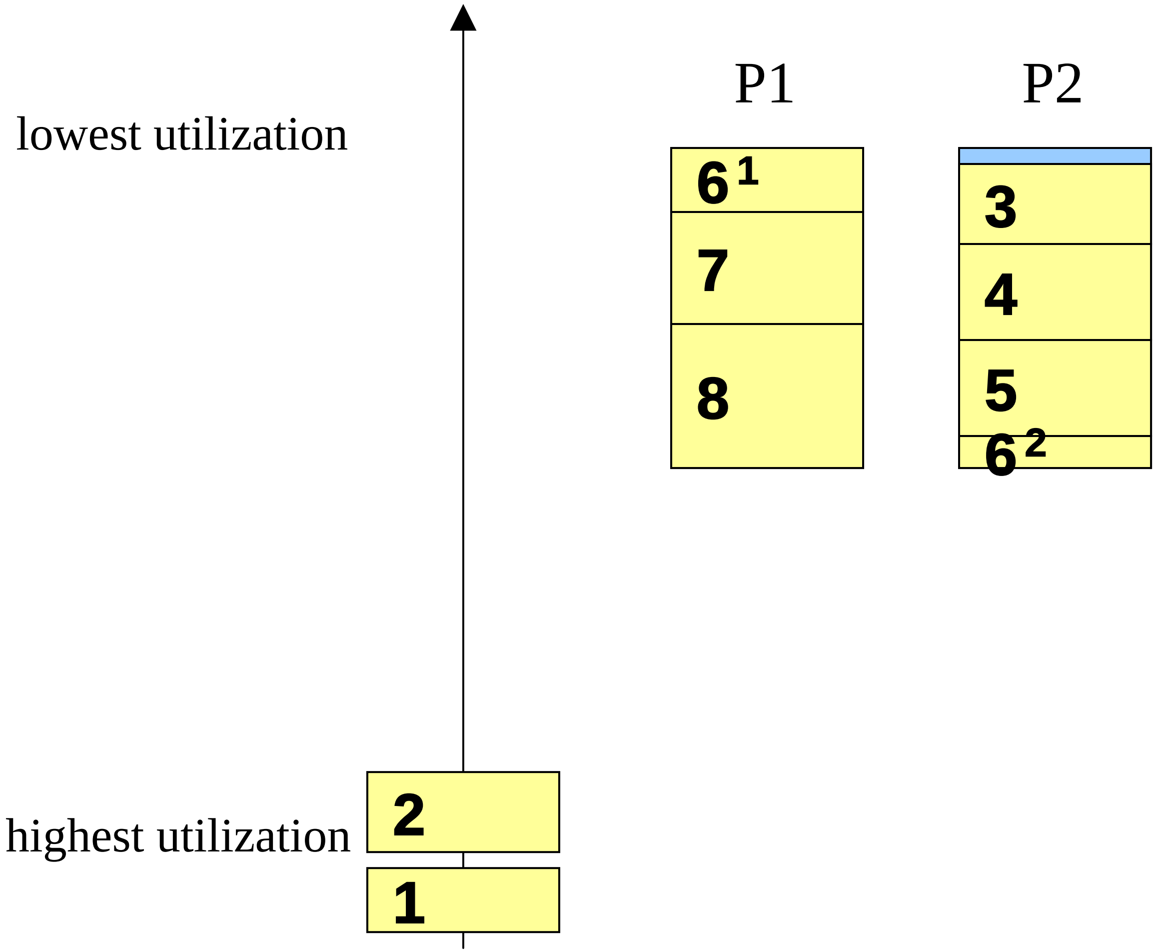
Lehoczky's Algorithm [ECRTS'09]

- pick up one processor, and assign as many tasks as possible



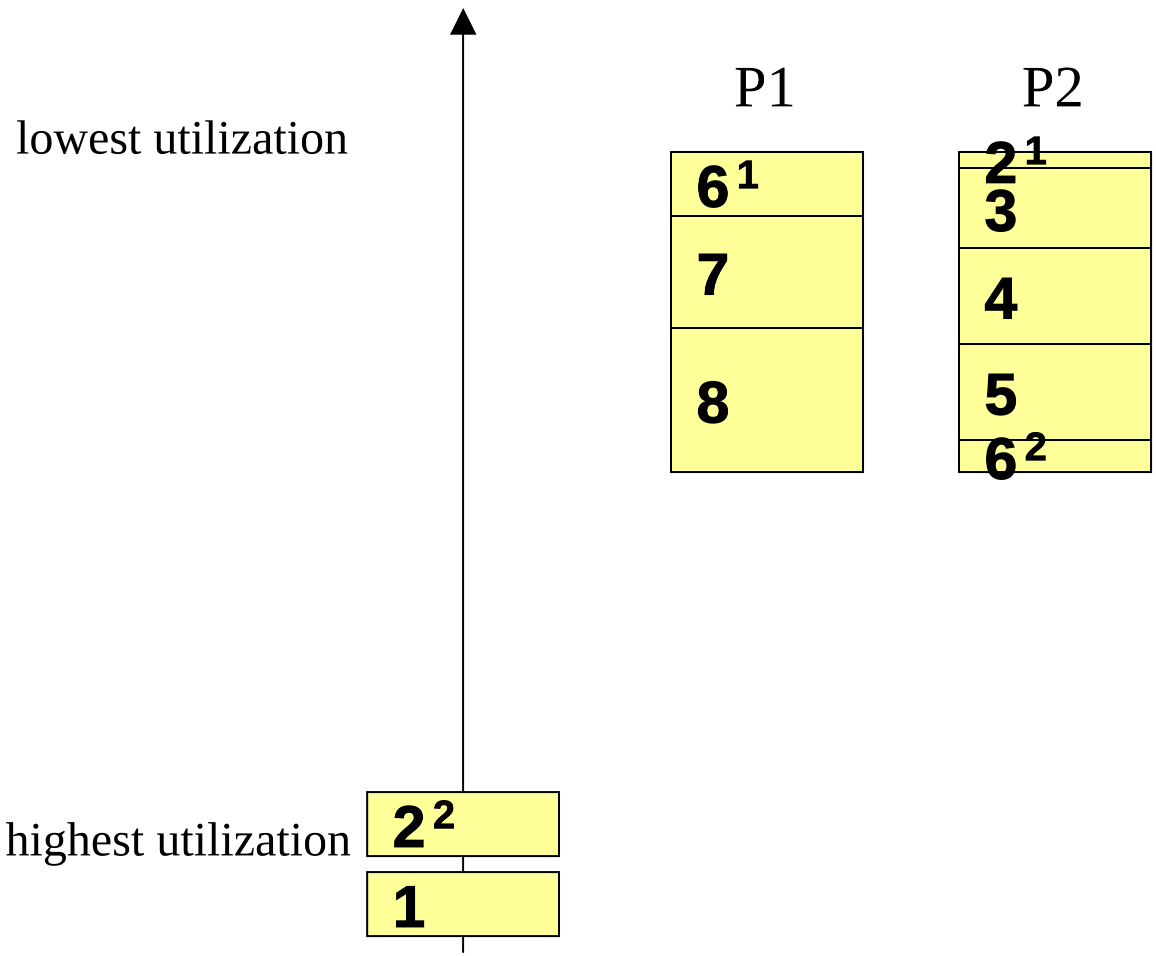
Lehoczky's Algorithm [ECRTS'09]

- pick up one processor, and assign as many tasks as possible



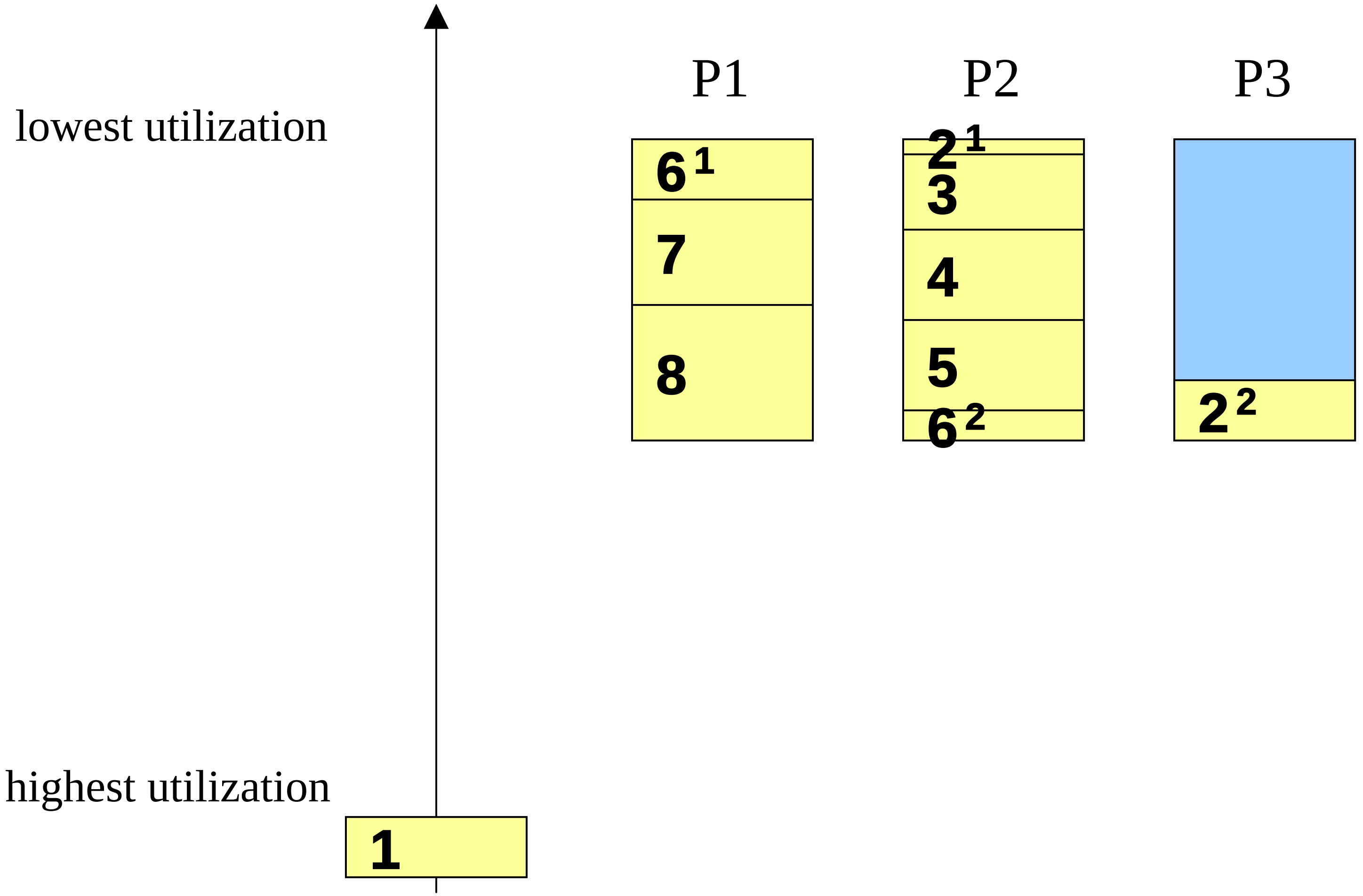
Lehoczky's Algorithm [ECRTS'09]

- pick up one processor, and assign as many tasks as possible



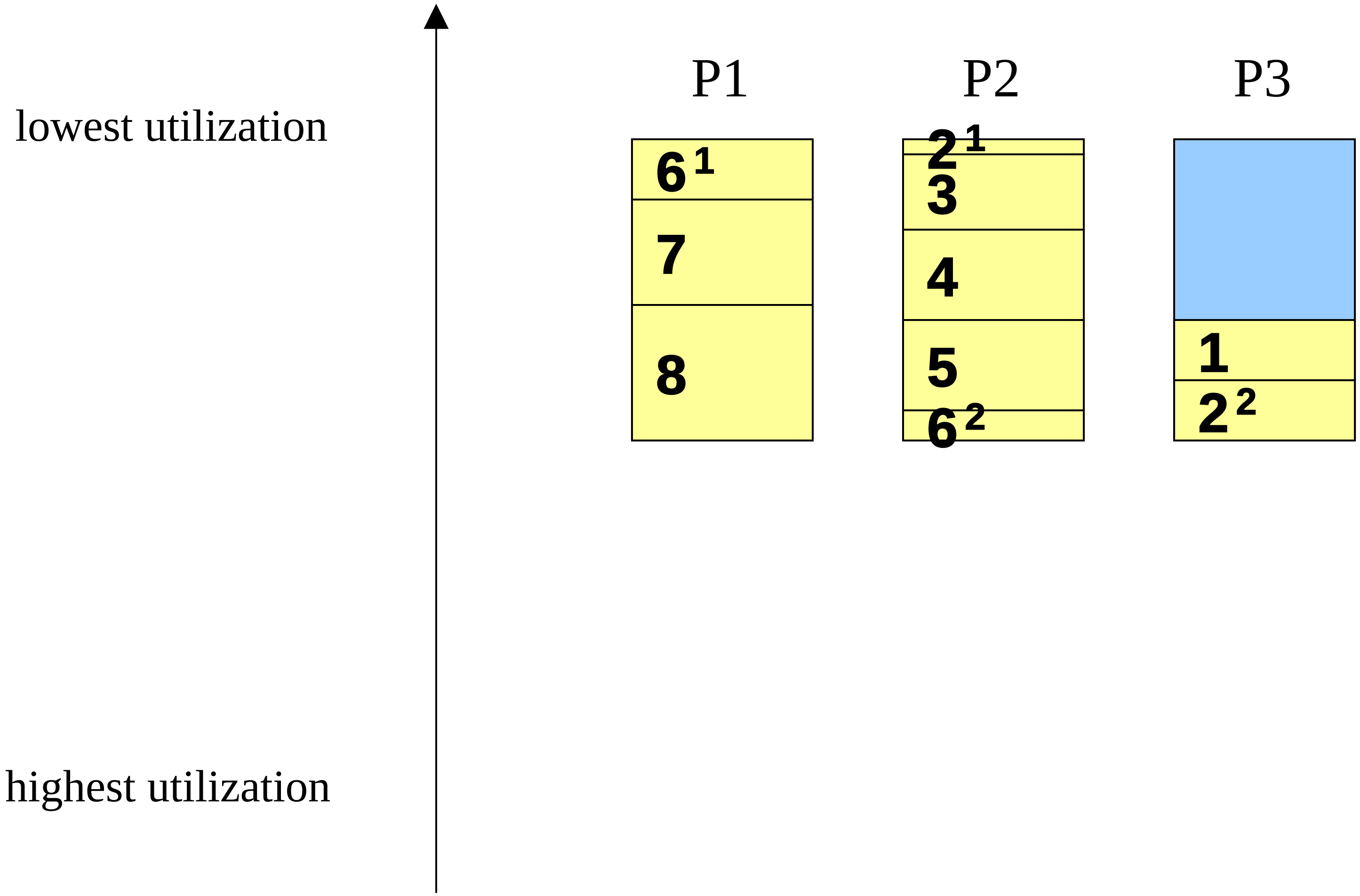
Lehoczky's Algorithm [ECRTS'09]

- pick up one processor, and assign as many tasks as possible



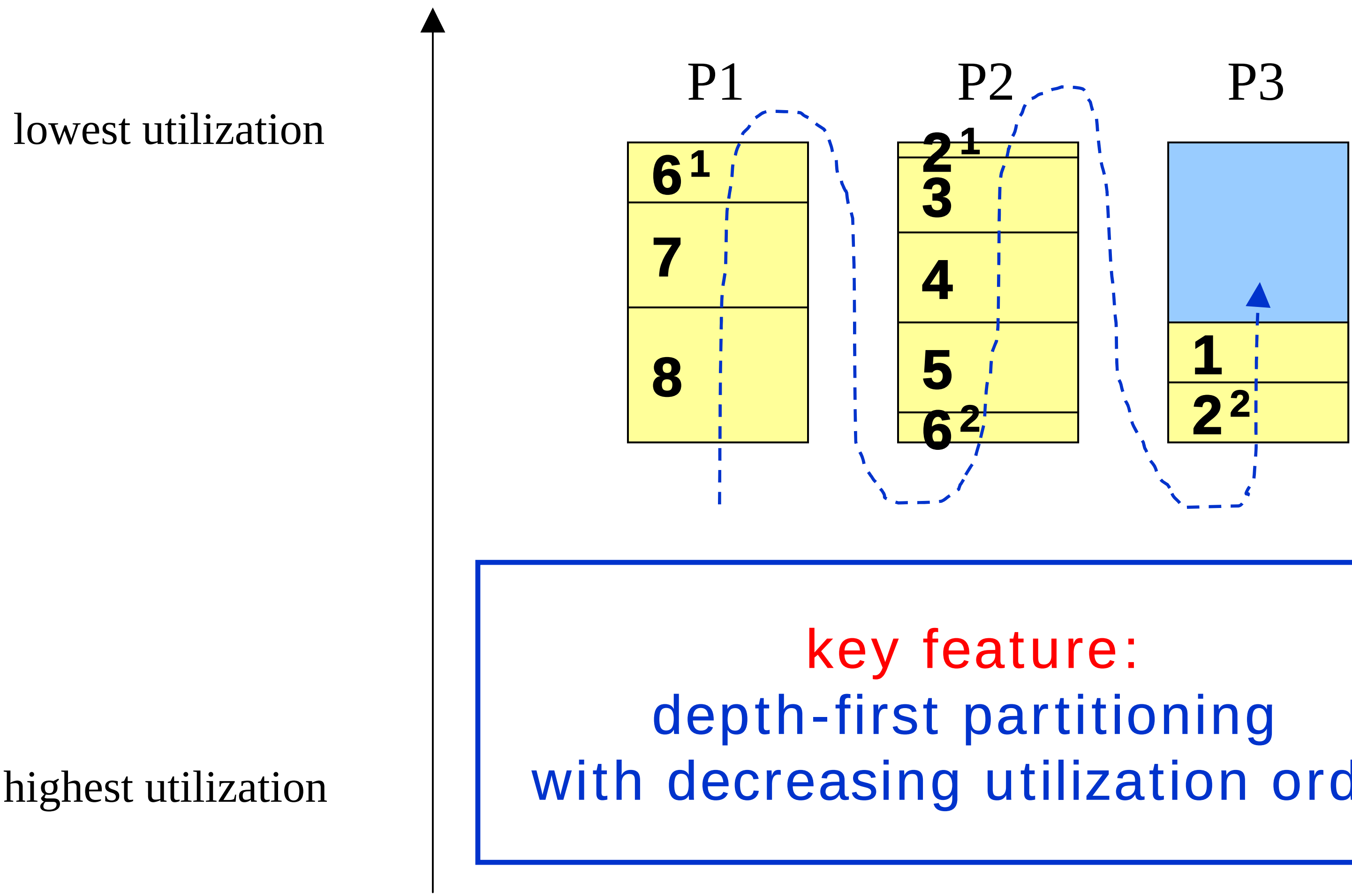
Lehoczky's Algorithm [ECRTS'09]

- pick up one processor, and assign as many tasks as possible



Lehoczky's Algorithm [ECRTS'09]

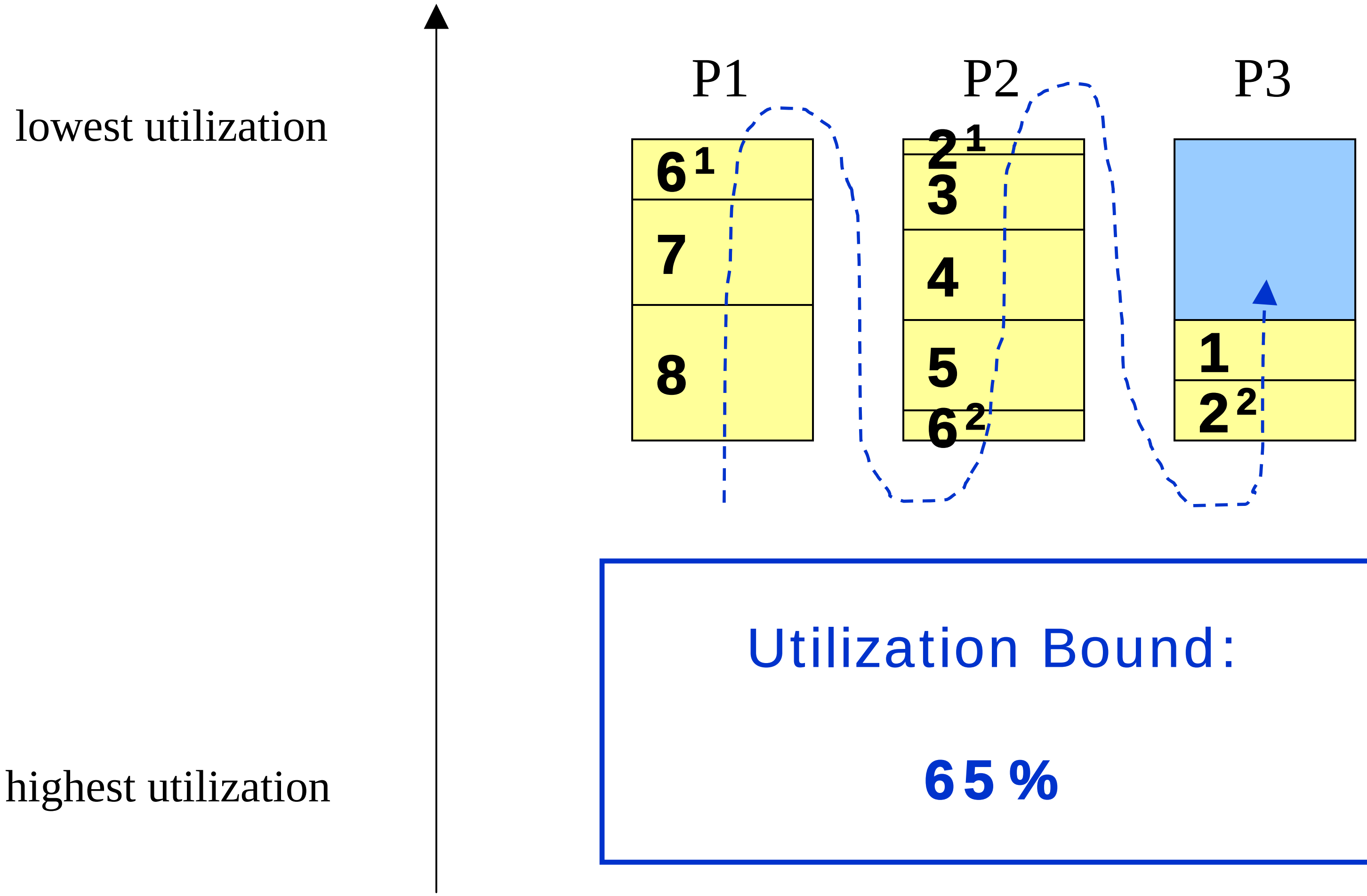
- pick up one processor, and assign as many tasks as possible



key feature:
depth-first partitioning
with decreasing utilization order

Lehoczky's Algorithm [ECRTS'09]

- pick up one processor, and assign as many tasks as possible



Utilization Bound:

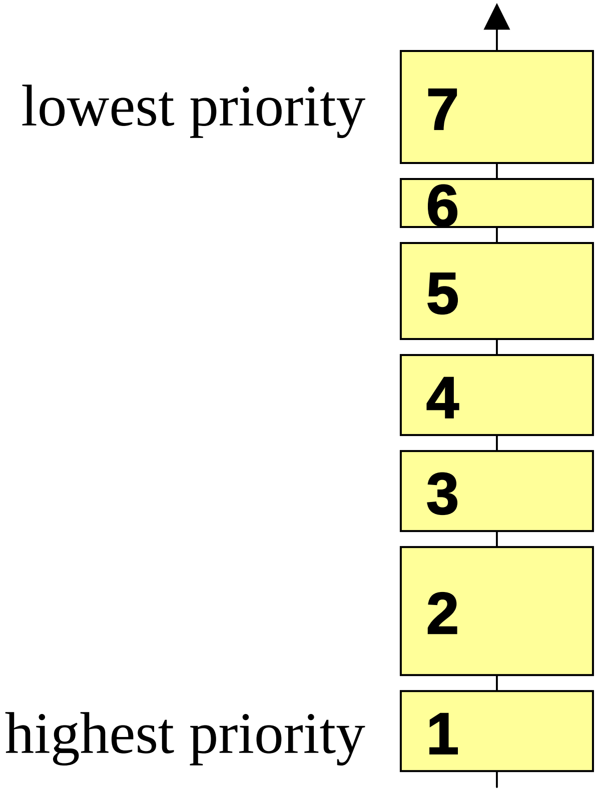
65%

Our Algorithm

width-first partitioning
with increasing priority order

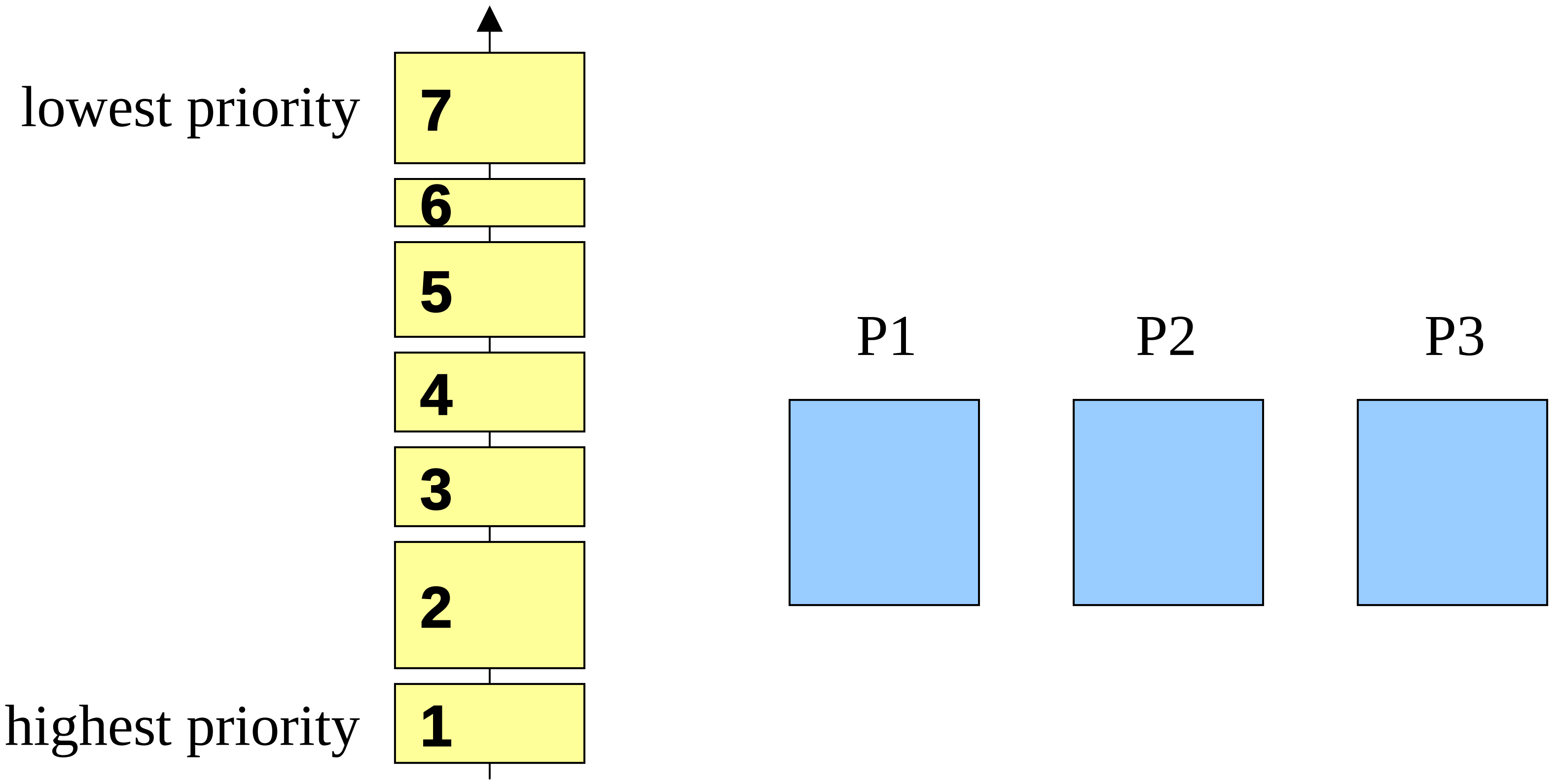
Our Algorithm

- sort all tasks in increasing priority order



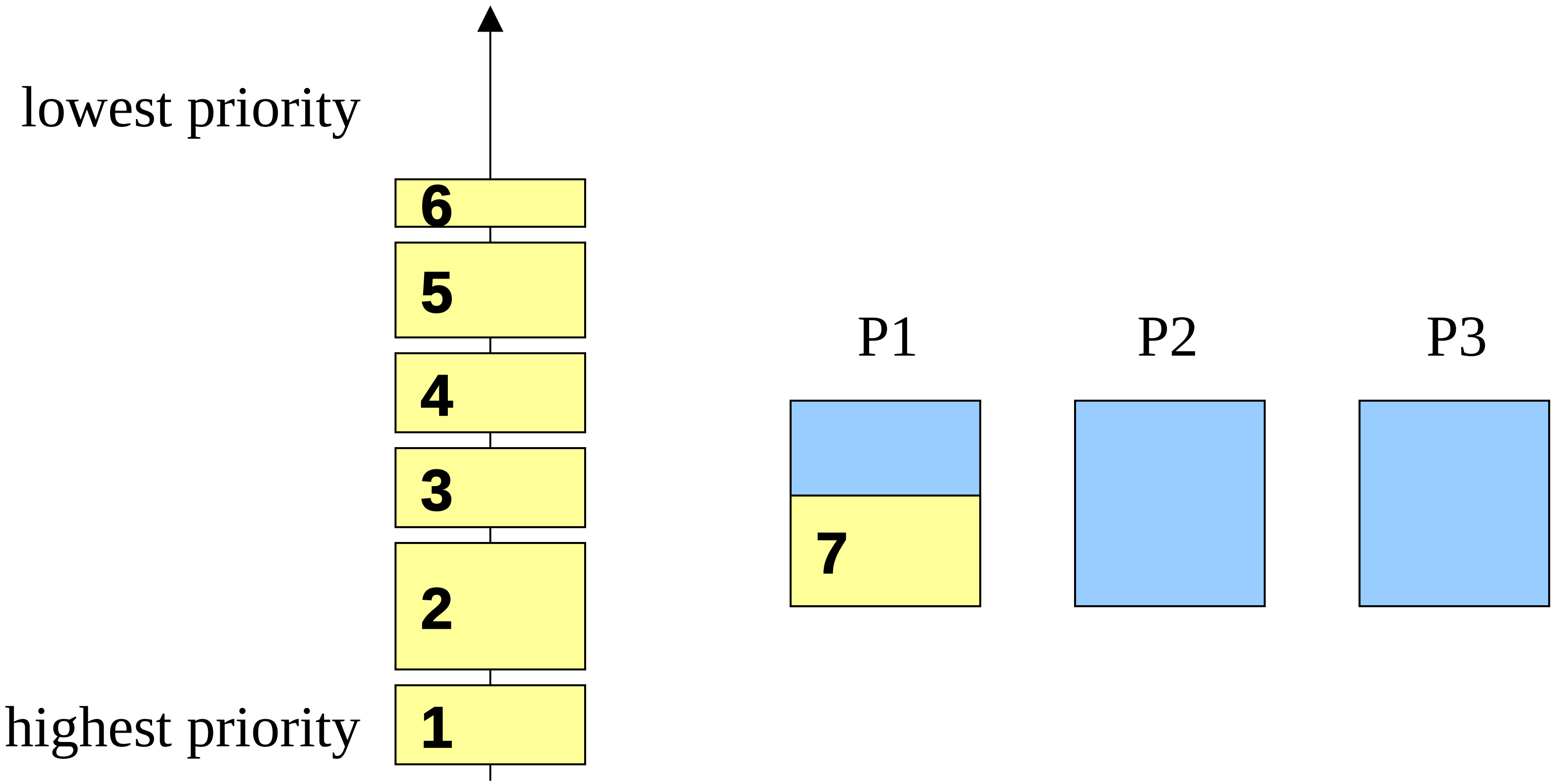
Our Algorithm

- select the processor on which the assigned utilization is the lowest



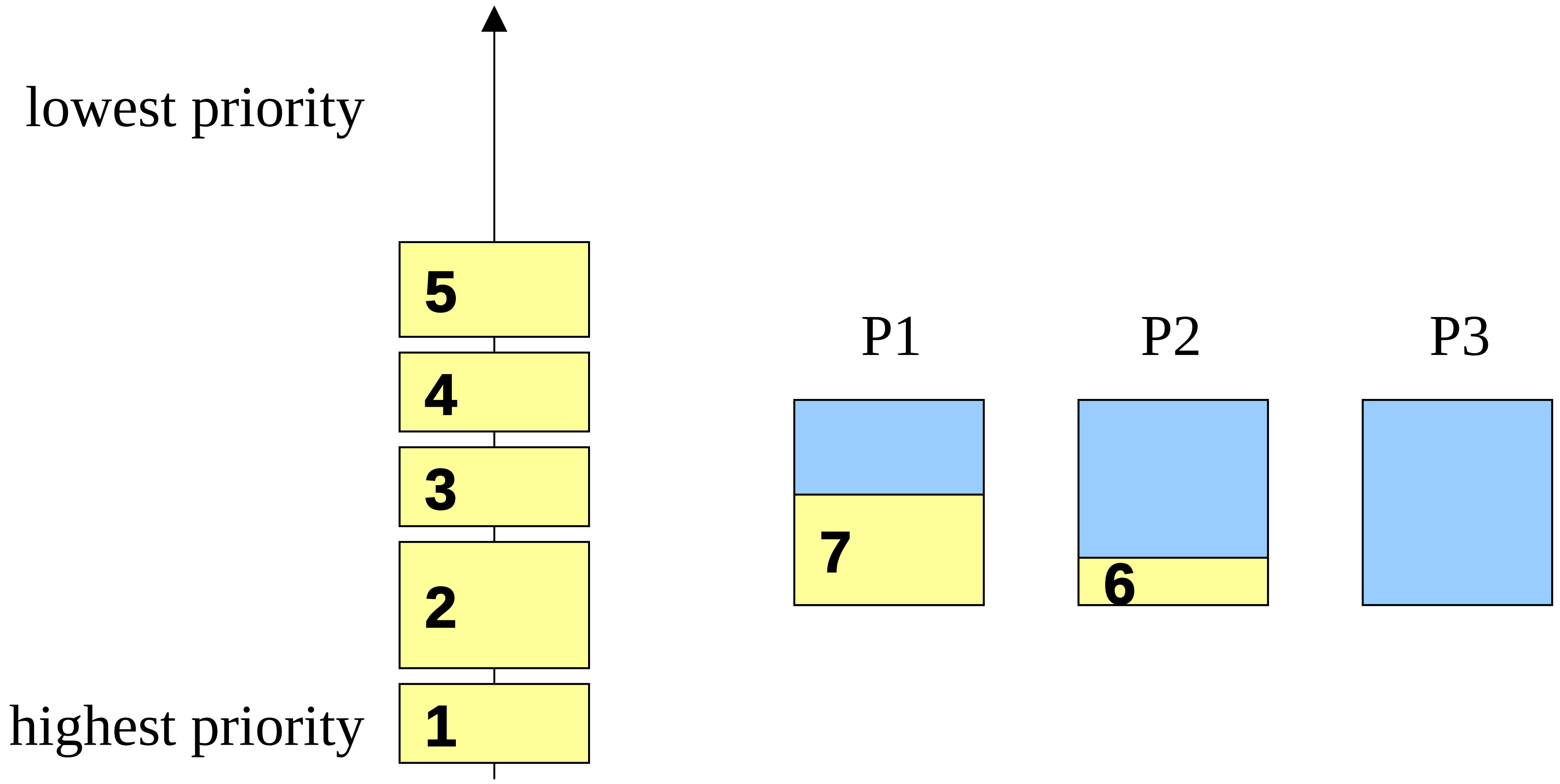
Our Algorithm

- select the processor on which the assigned utilization is the lowest



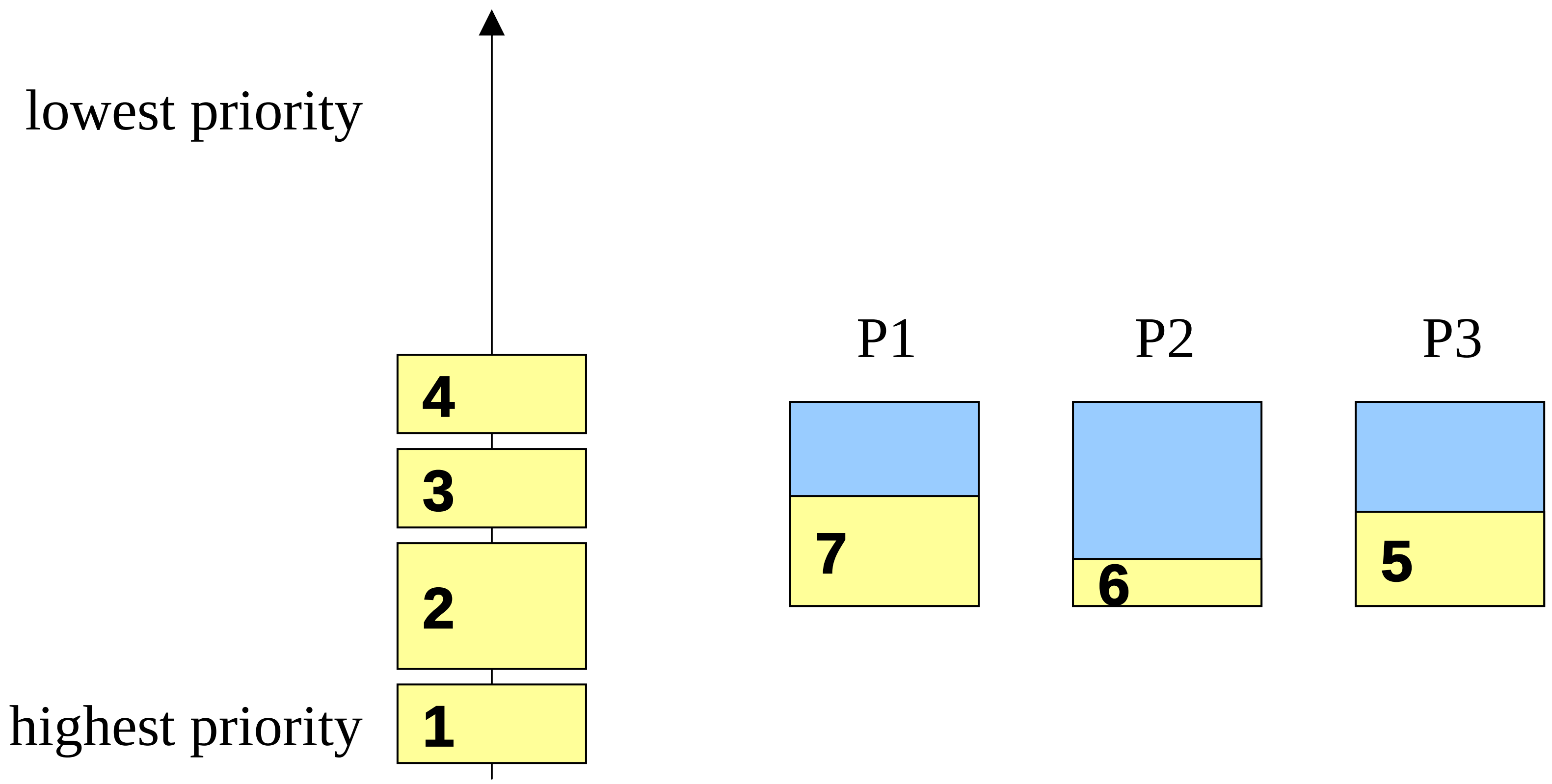
Our Algorithm

- select the processor on which the assigned utilization is the lowest



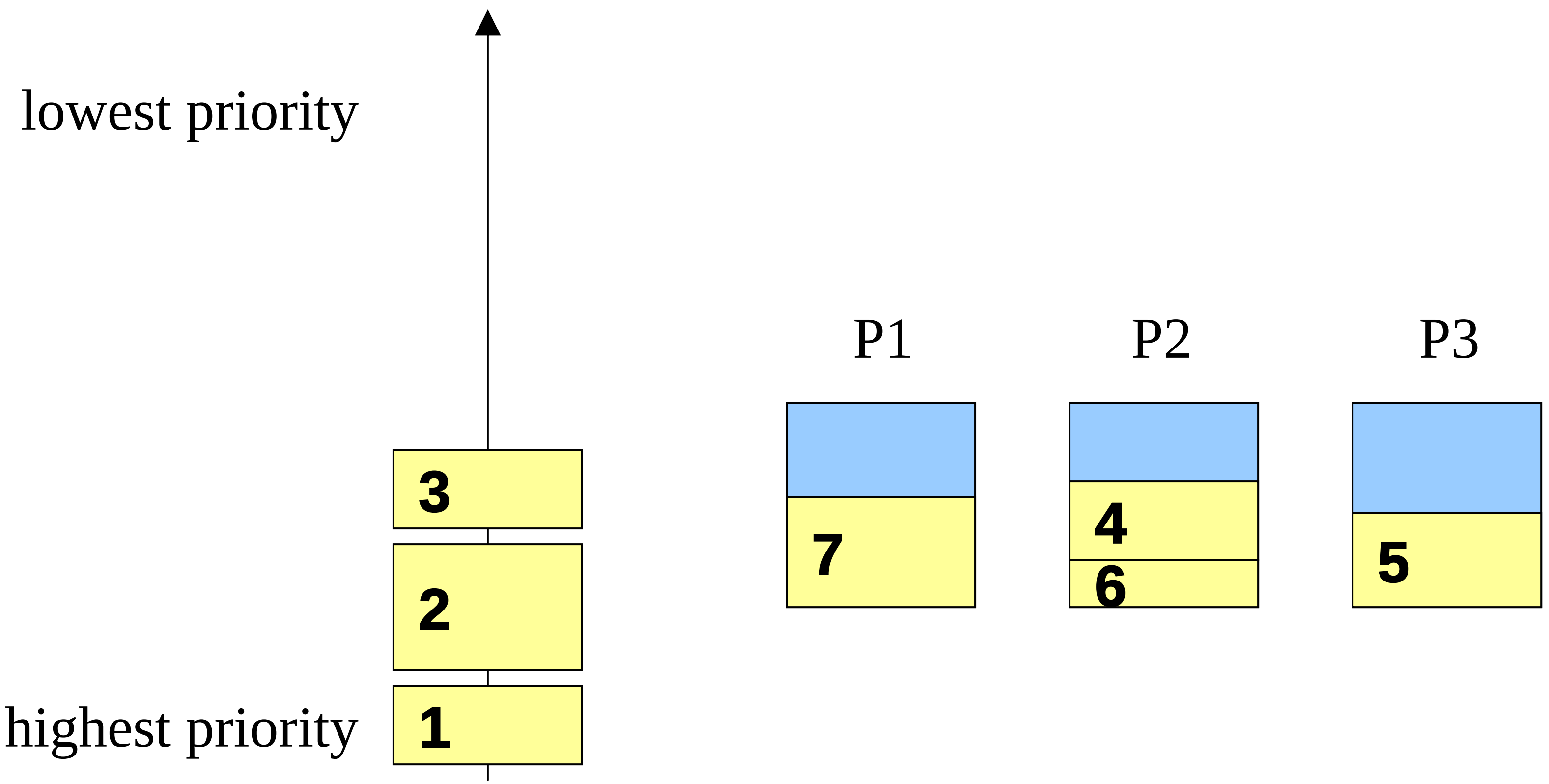
Our Algorithm

- select the processor on which the assigned utilization is the lowest



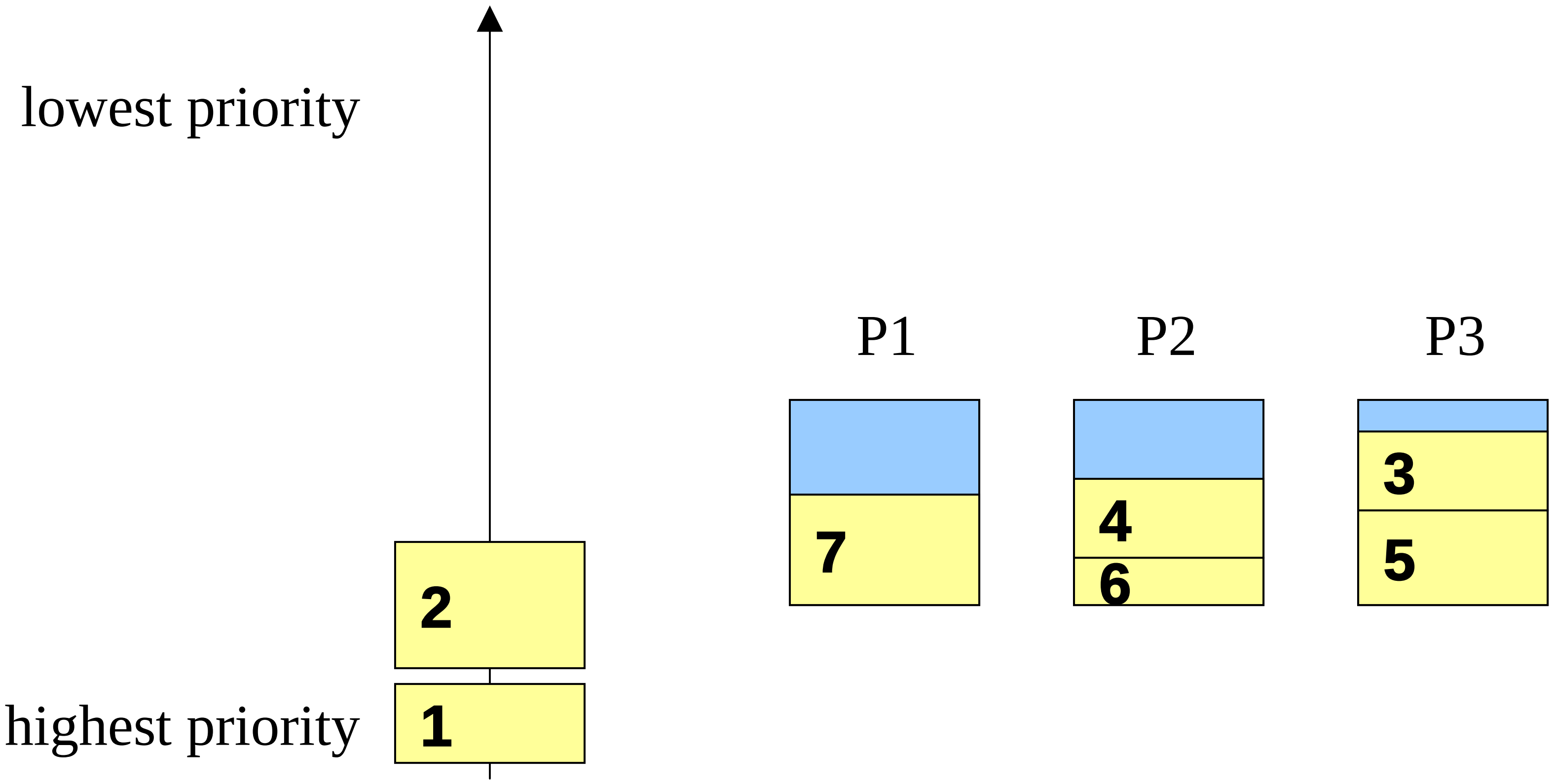
Our Algorithm

- select the processor on which the assigned utilization is the lowest



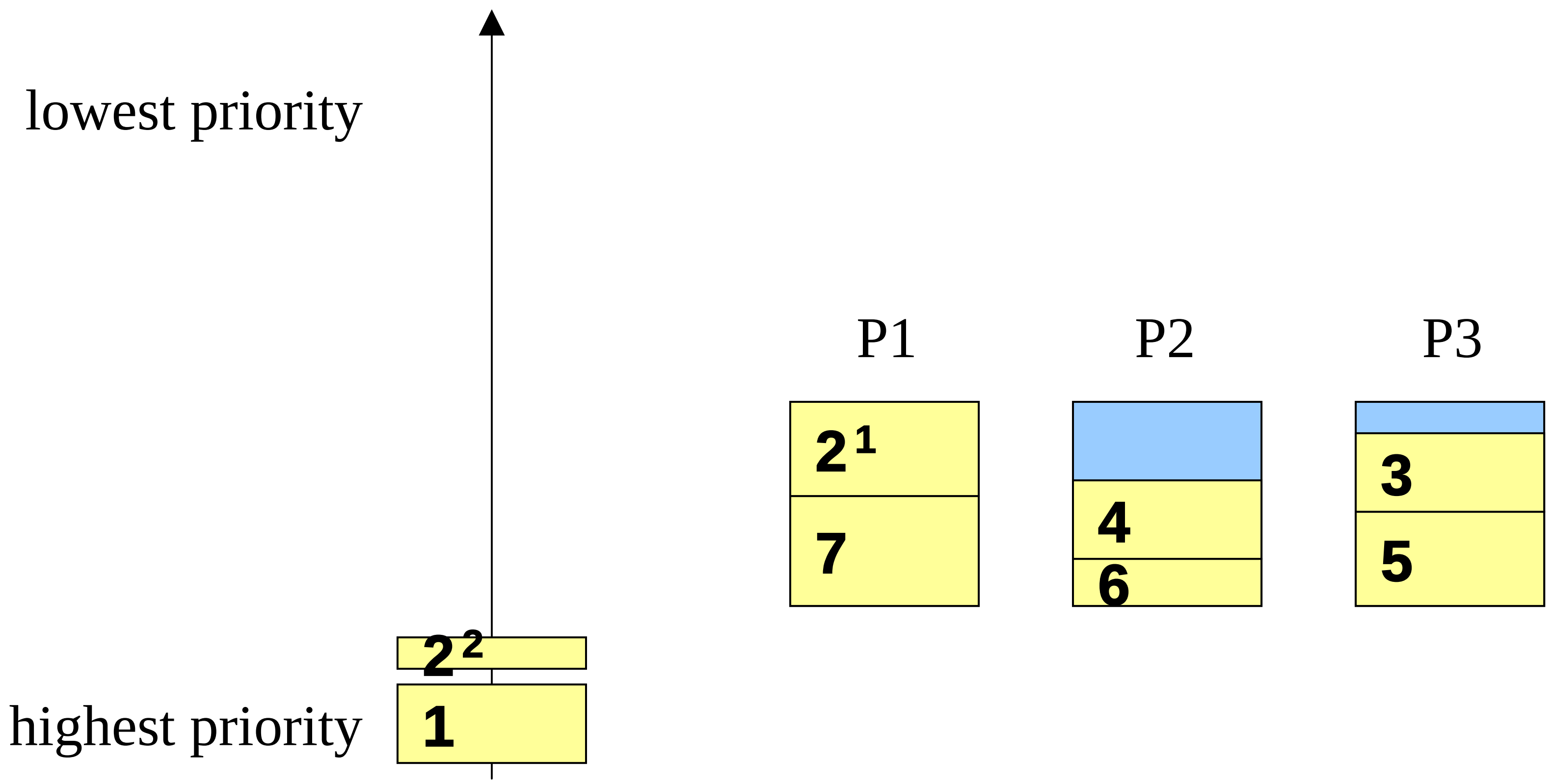
Our Algorithm

- select the processor on which the assigned utilization is the lowest



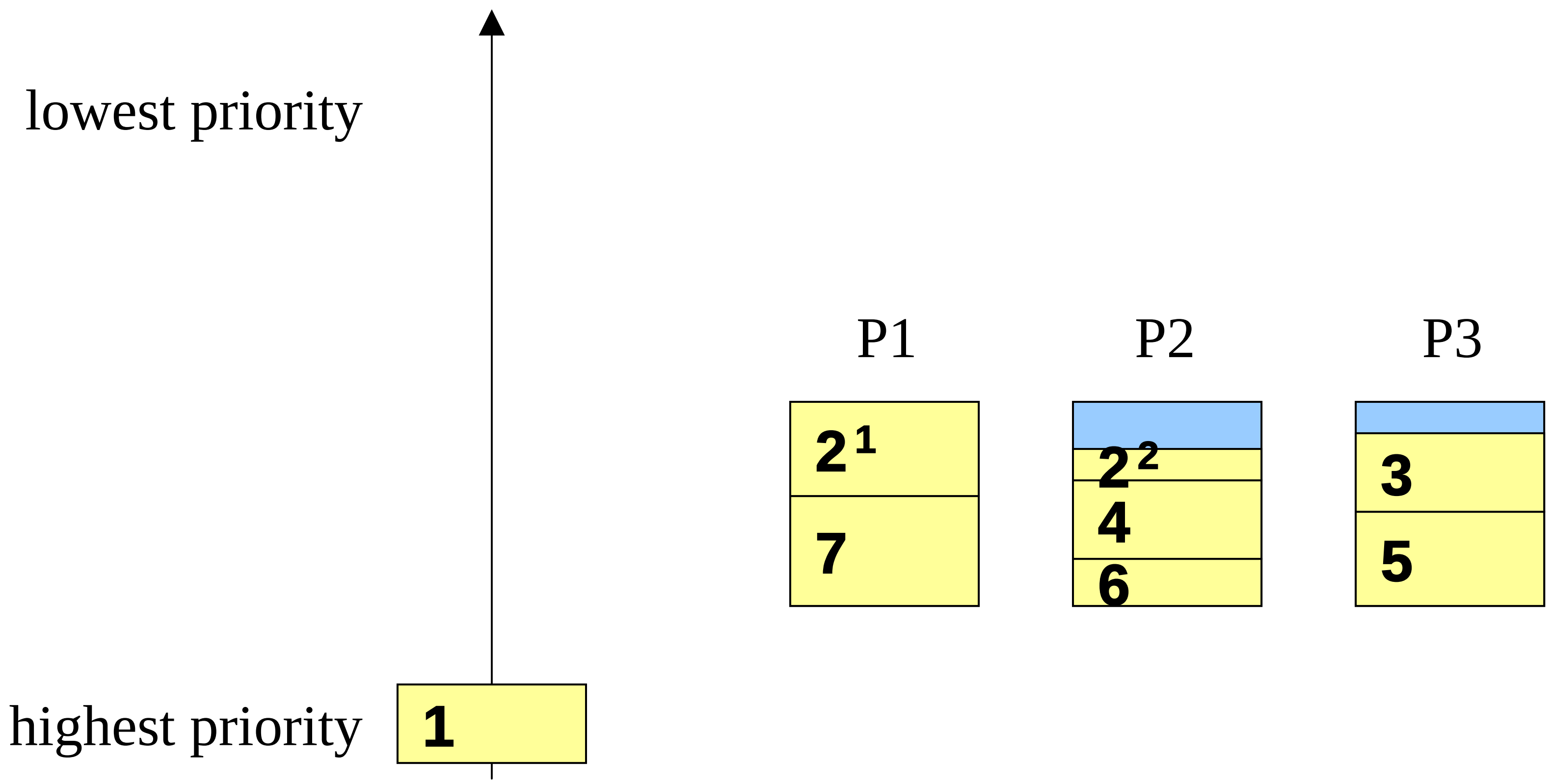
Our Algorithm

- select the processor on which the assigned utilization is the lowest



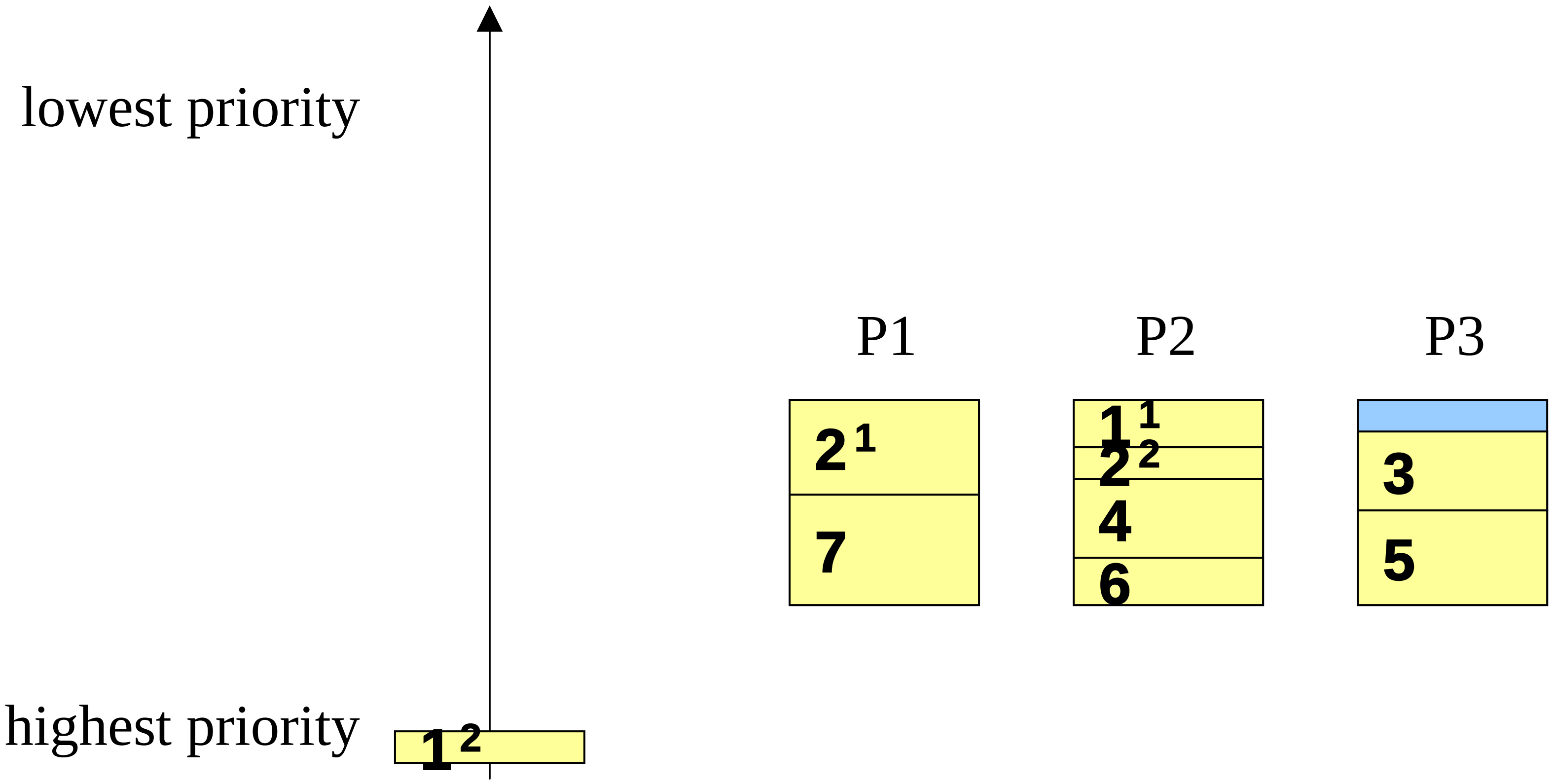
Our Algorithm

- select the processor on which the assigned utilization is the lowest



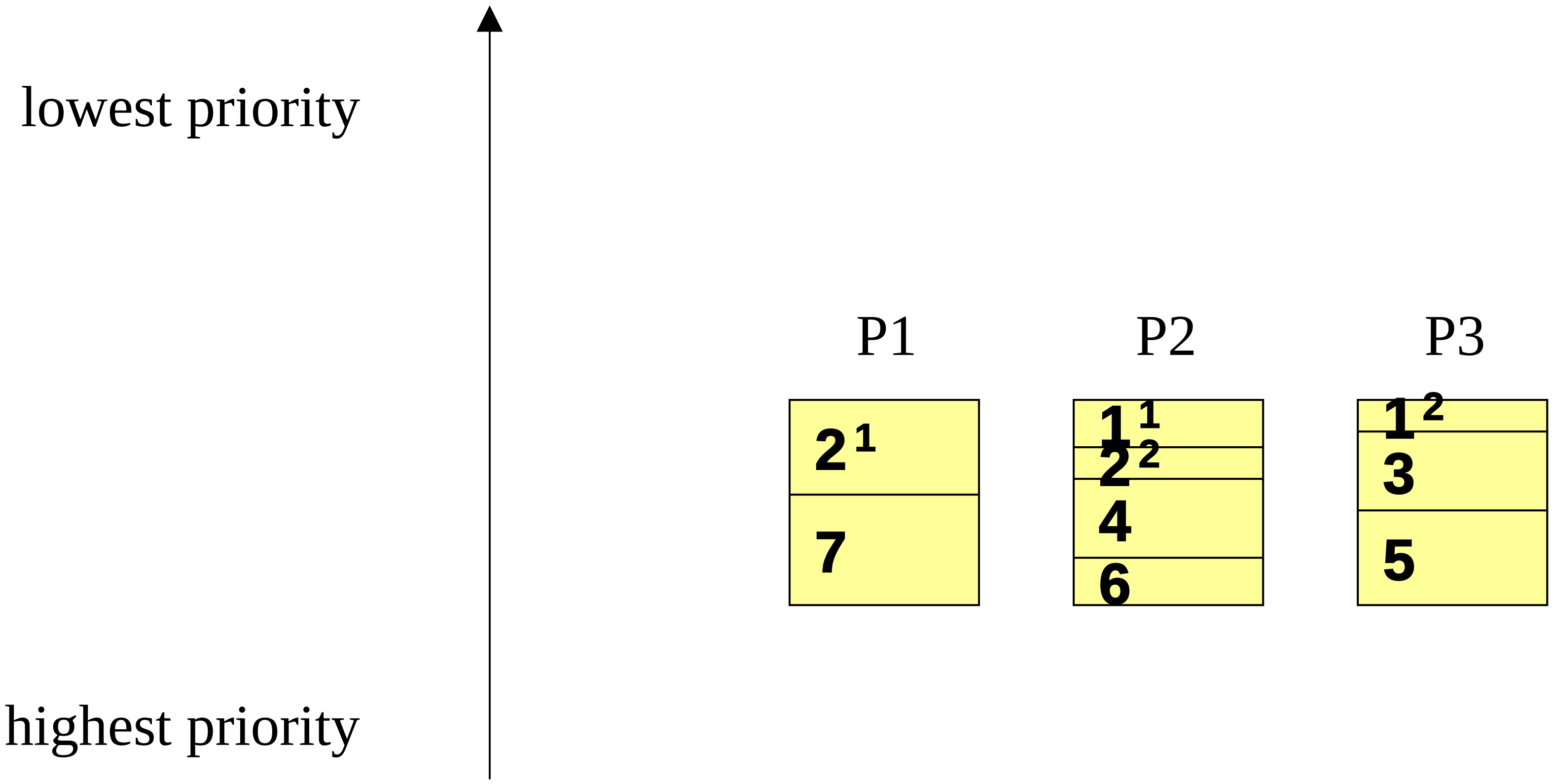
Our Algorithm

- select the processor on which the assigned utilization is the lowest



Our Algorithm

- select the processor on which the assigned utilization is the lowest



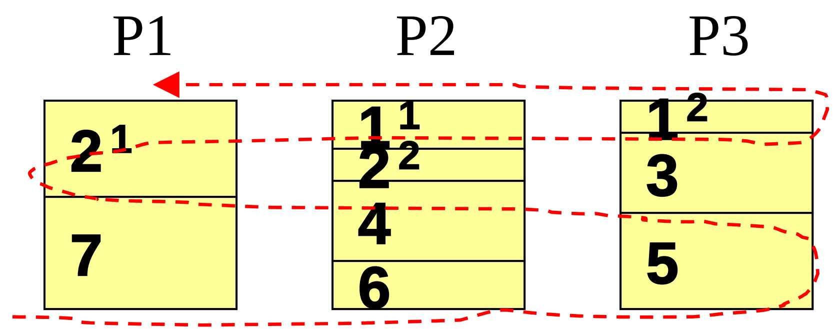
Our Algorithm

- select the processor on which the assigned utilization is the lowest

lowest priority

highest priority

key feature:
width-first partitioning with increasing prio order

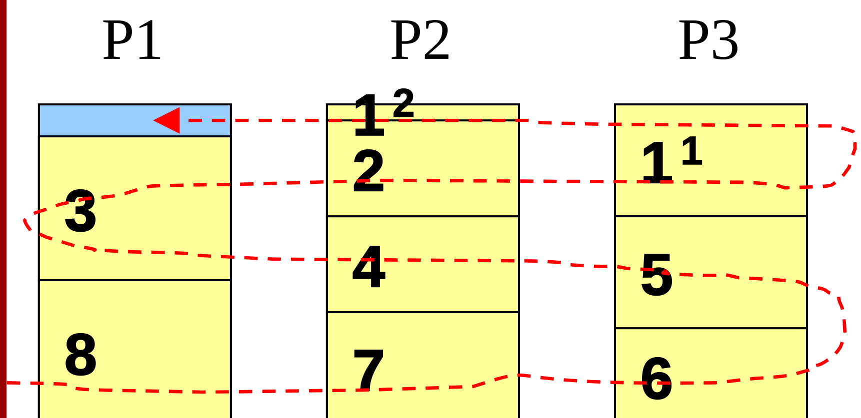


Comparison

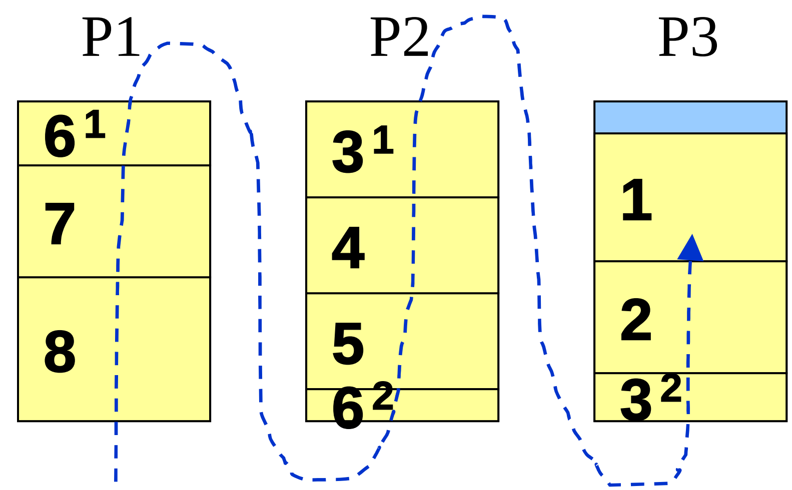
- maximal number of task splitting

  both are *M-1*

Ours: width-first
(increasing priority order)



Lehoczky's: depth-first
(decreasing utilization order)

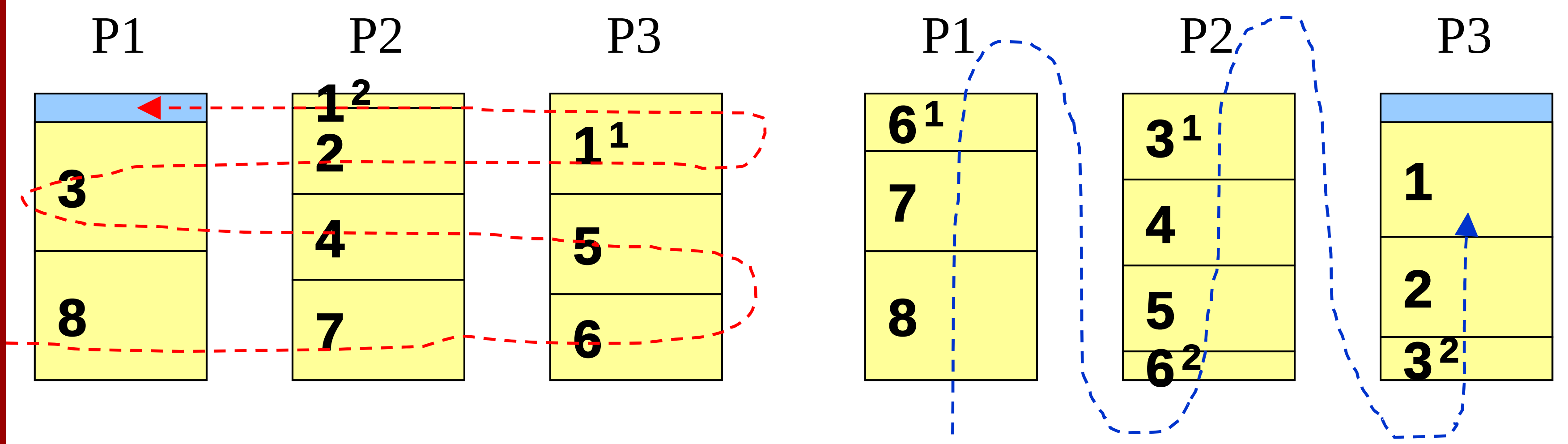


Comparison

- why is our algorithm better?

Ours: width-first
(increasing priority order)

Lehoczky's: depth-first
(decreasing utilization order)

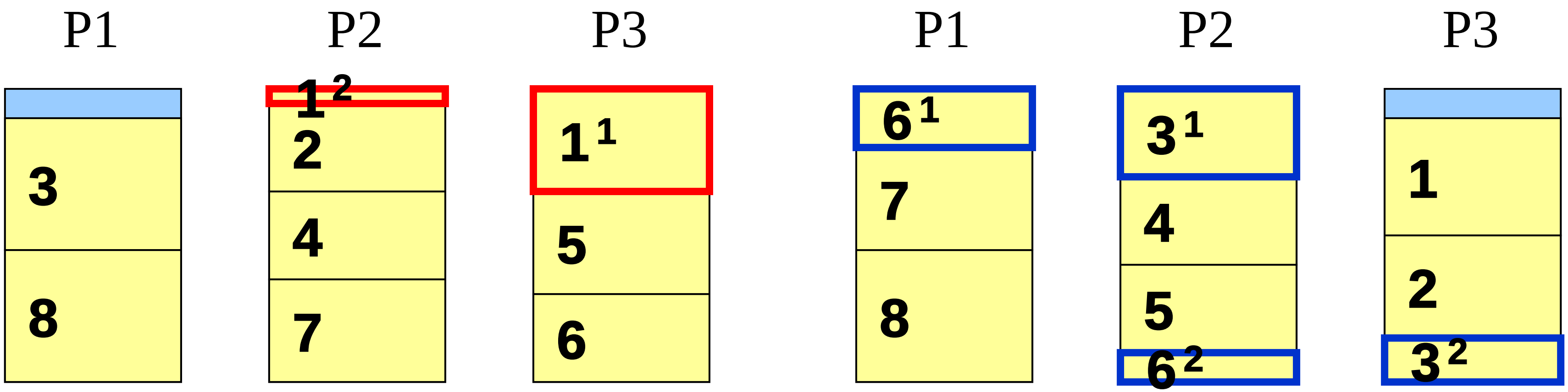


Comparison

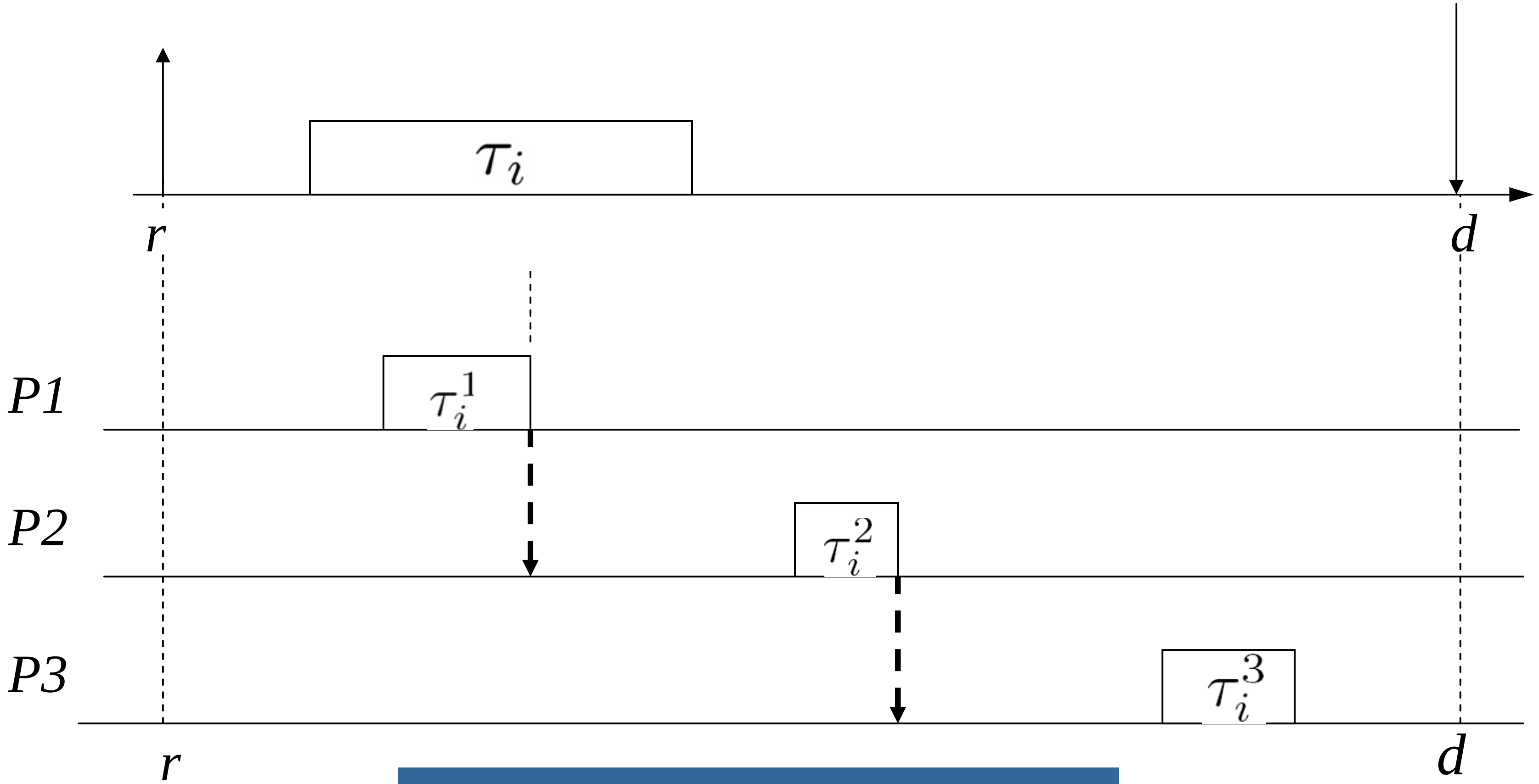
key point:
by our algorithm, split tasks generally have high priorities on each processor

Ours: width-first (increasing priority order)

Lehoczky's: depth-first (decreasing utilization order)

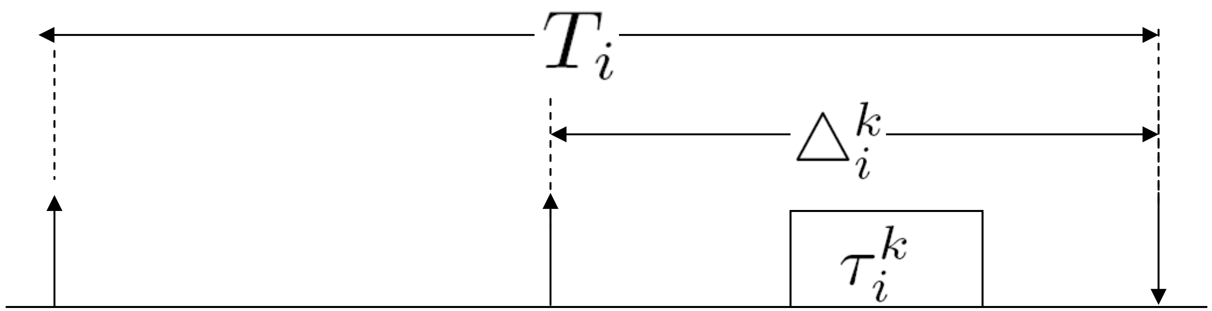


Split Task



subtasks should be executed in the correct order

Split Task



original utilization: $U_i^k = c_i^k / T_i$

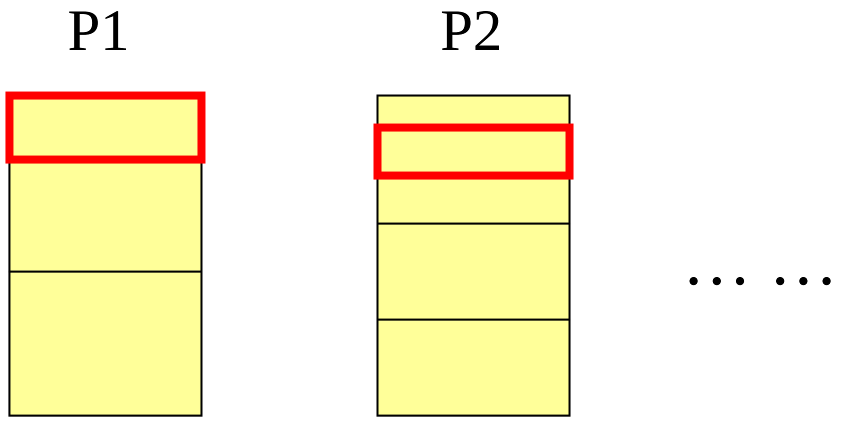
synthetic utilization: $V_i^k = c_i^k / \triangle_i^k$

$V_i^k > U_i^k$

split tasks cause "utilization increase"

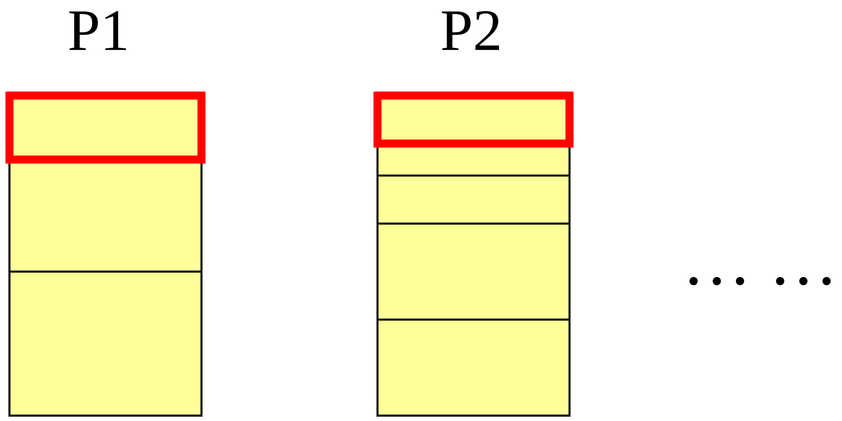
Our Algorithm

- intuition
  - high priority tasks have better chance to meet their deadlines



Our Algorithm

- intuition
  - an extreme scenario:
    - each subtask has the highest priority on each processor
    - can meet their deadlines anyway
    - no "utilization increase"



Theorem

for a task set in with each task τ_i satisfies

$$U_i \leq \frac{\Theta(N)}{1+\Theta(N)}$$

we have

$$\frac{\sum C_i/T_i}{M} \leq N(2^{1/N}-1)$$

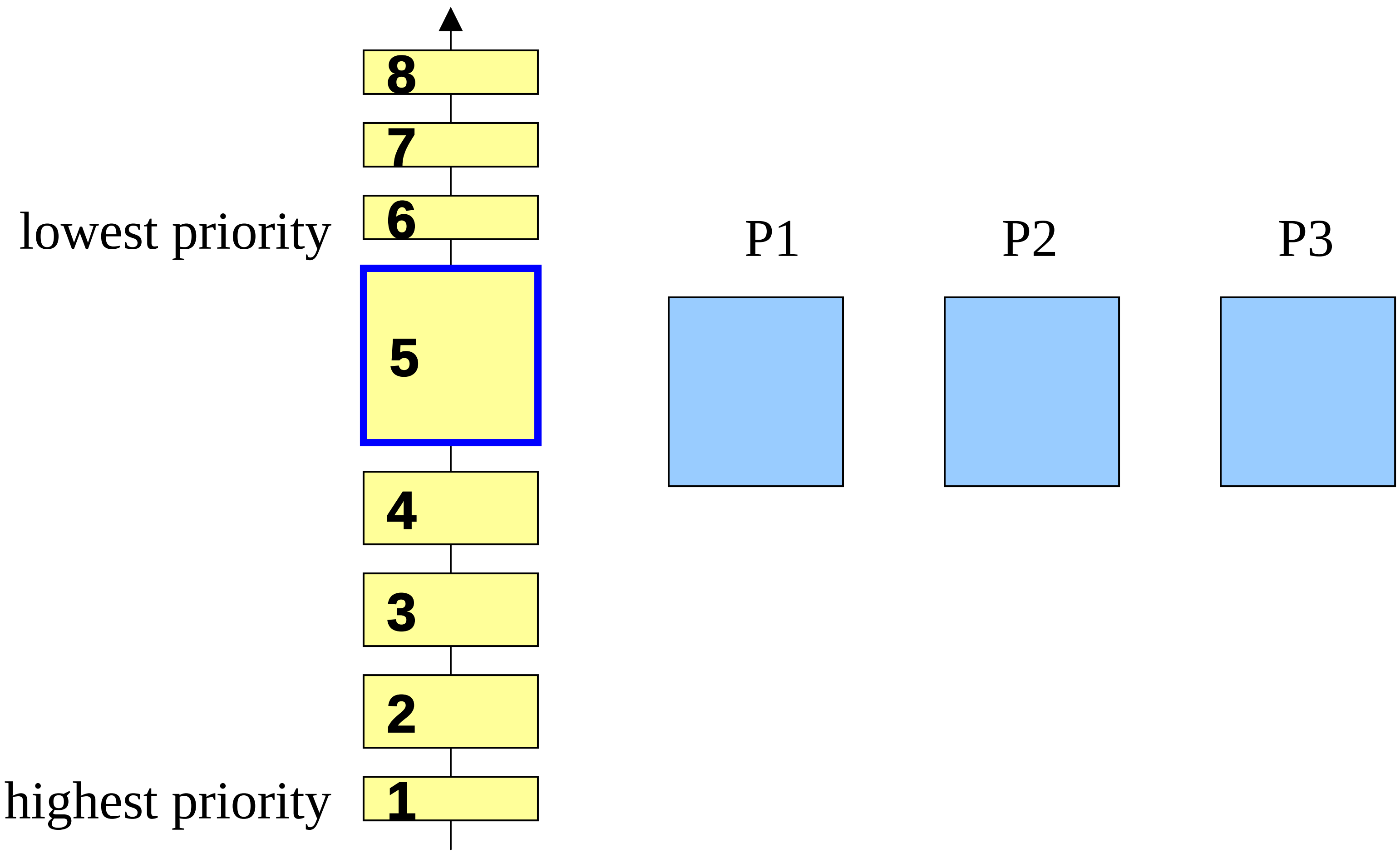
$$\Rightarrow \quad \text{the task set is schedulable}$$

$$\Theta(N) = N(2^{\frac{1}{N}}-1) \qquad \frac{\Theta(N)}{1+\Theta(N)} \doteq 0.41$$

reasonable constraint in real-life systems

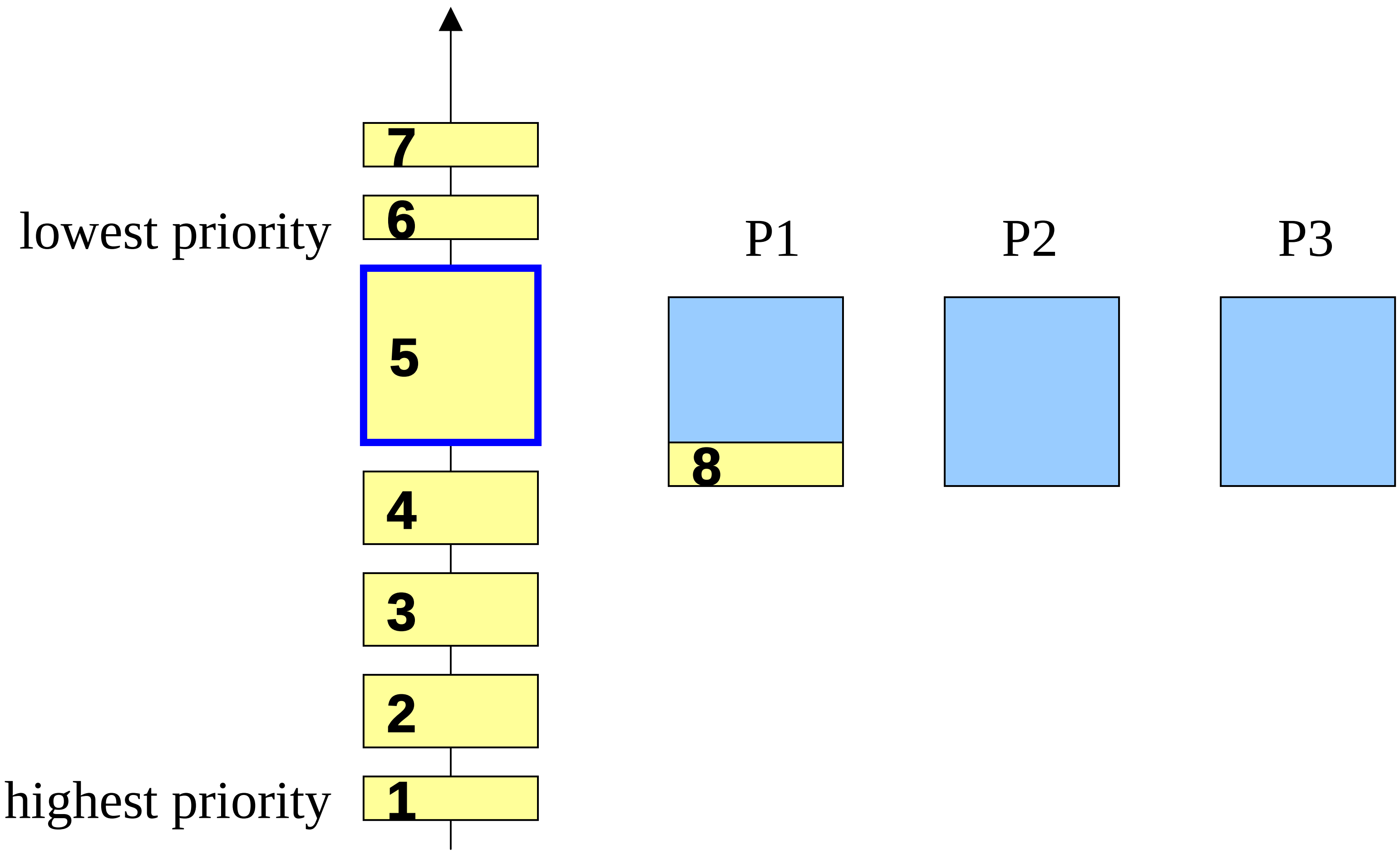
Our Algorithm

- problem of heavy tasks



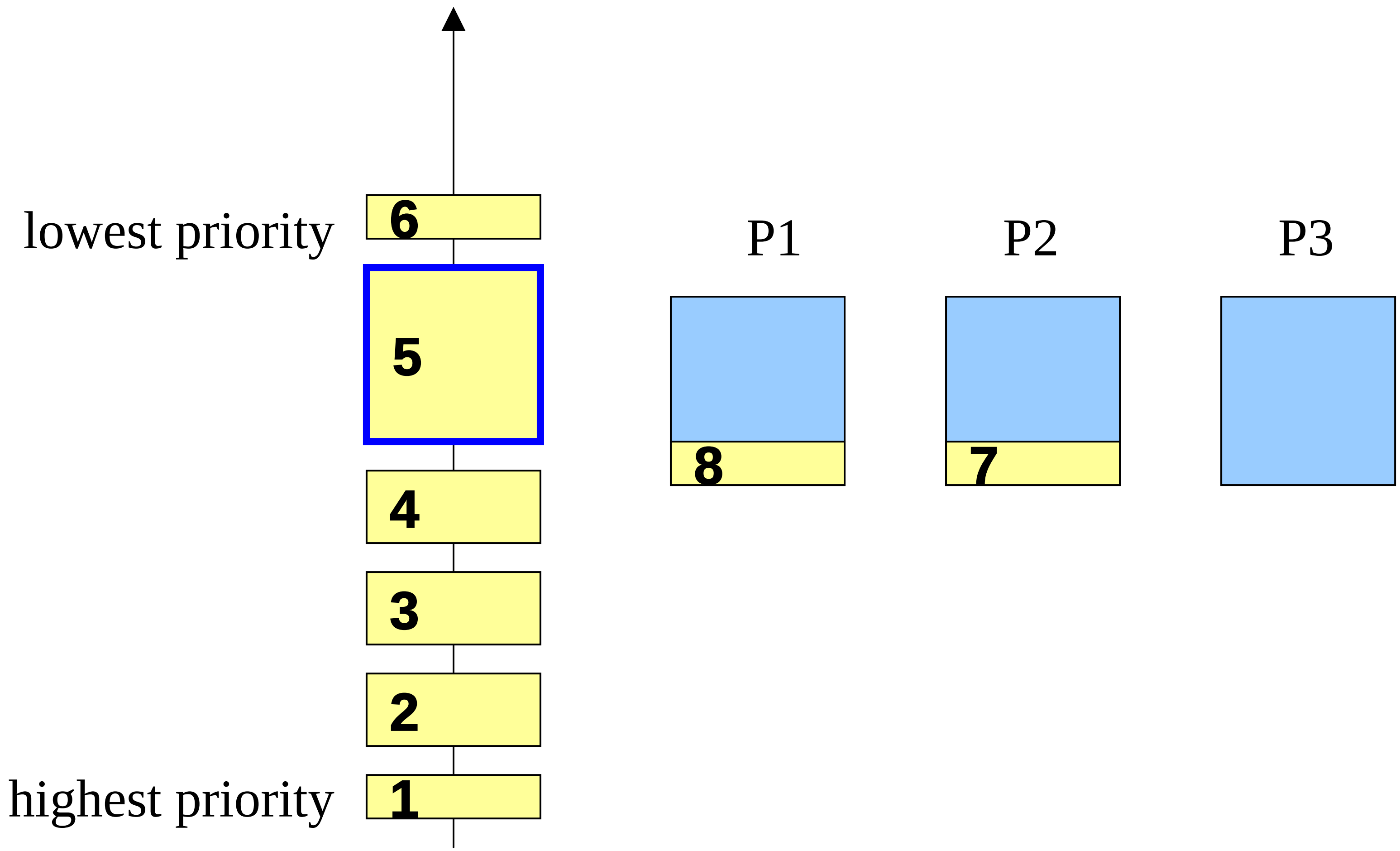
Our Algorithm

- problem of heavy tasks



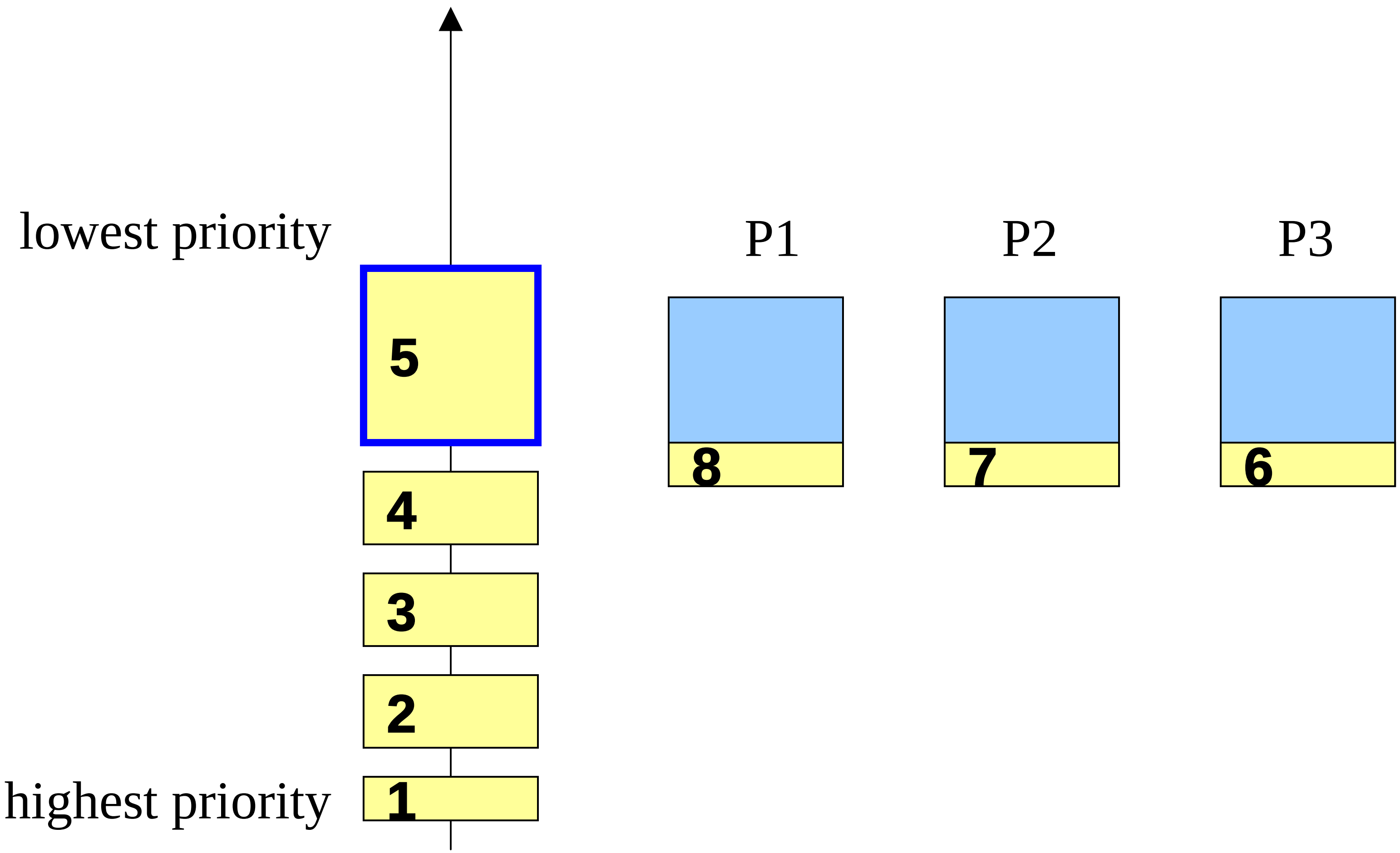
Our Algorithm

- problem of heavy tasks



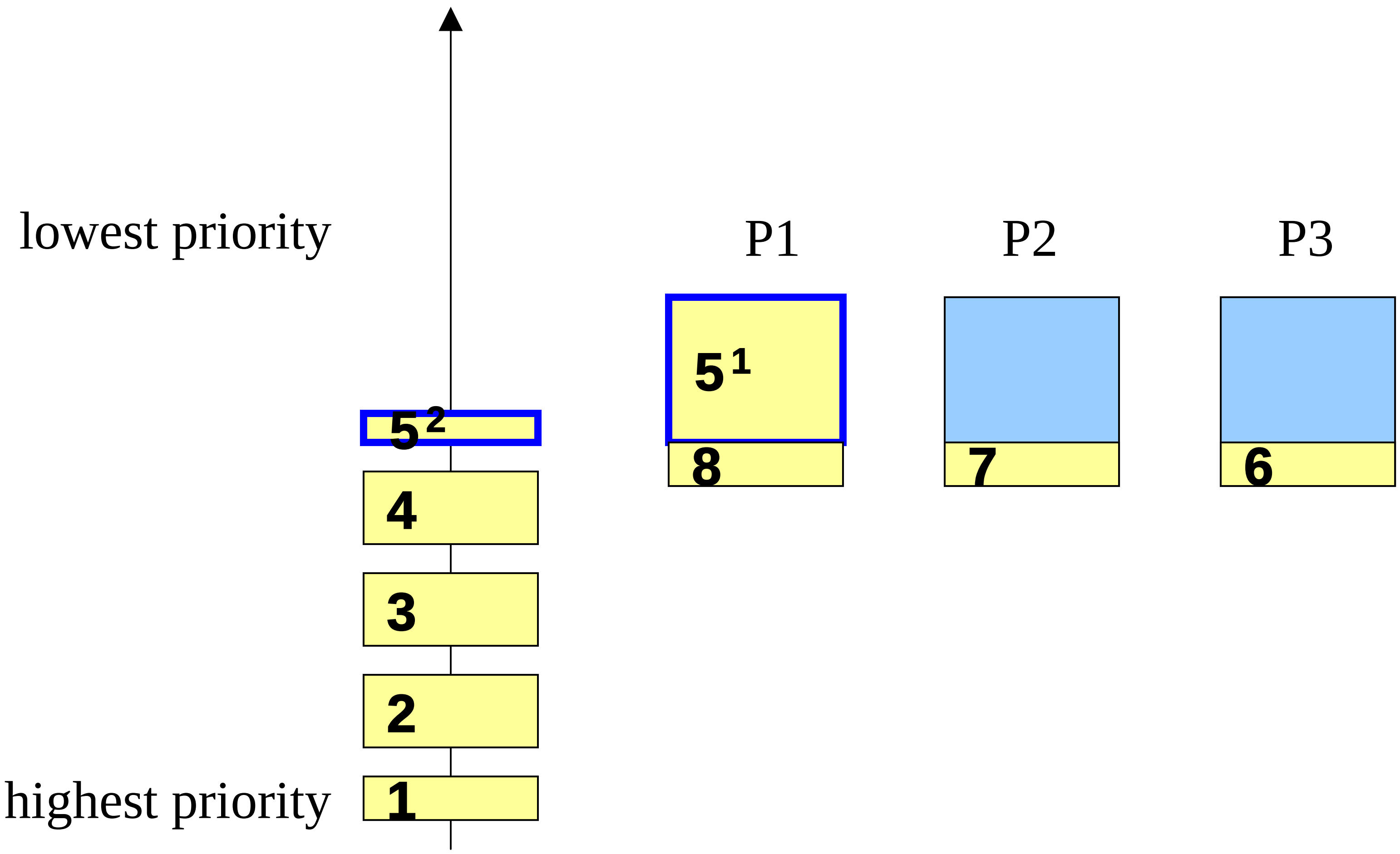
Our Algorithm

- problem of heavy tasks



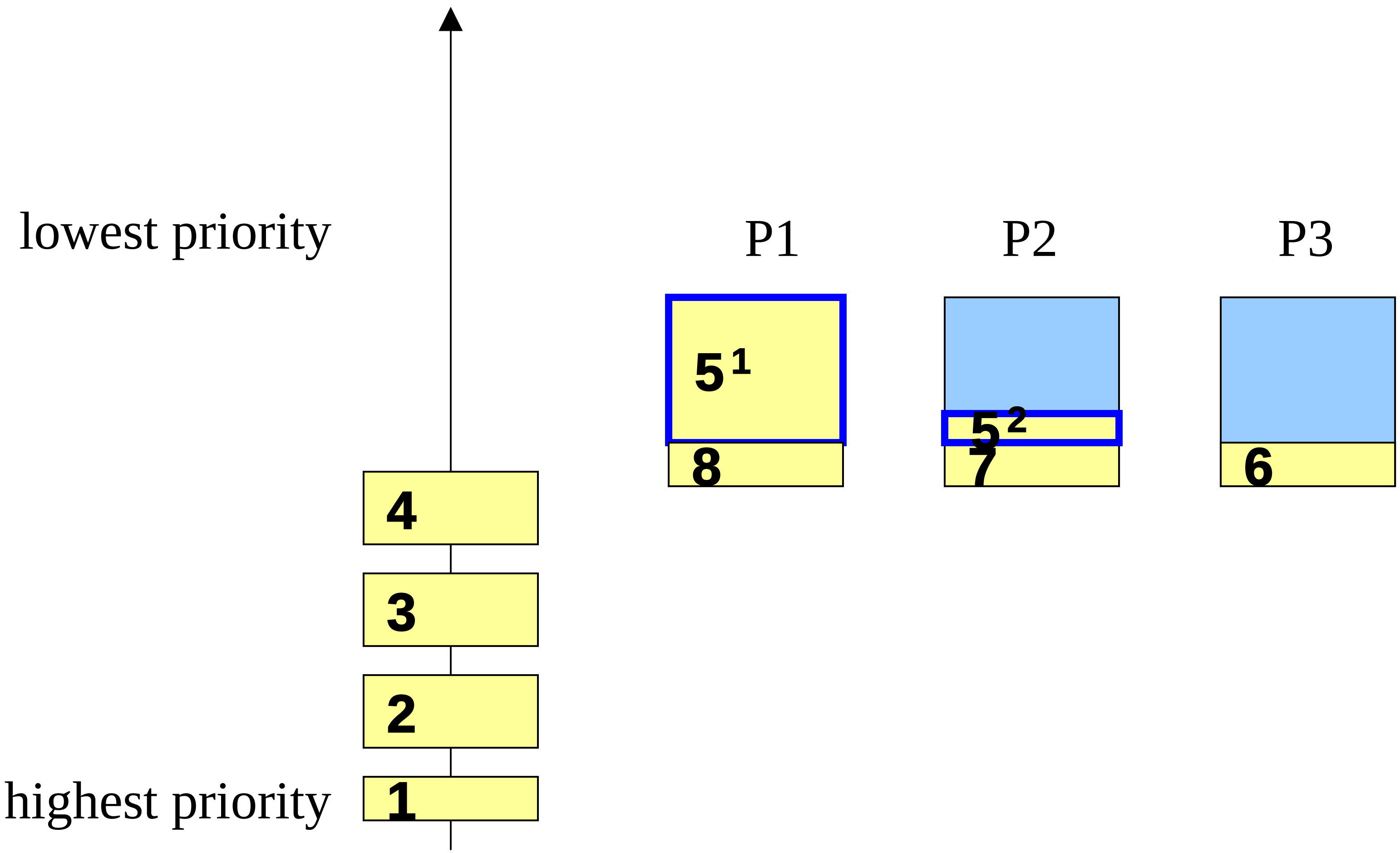
Our Algorithm

- problem of heavy tasks



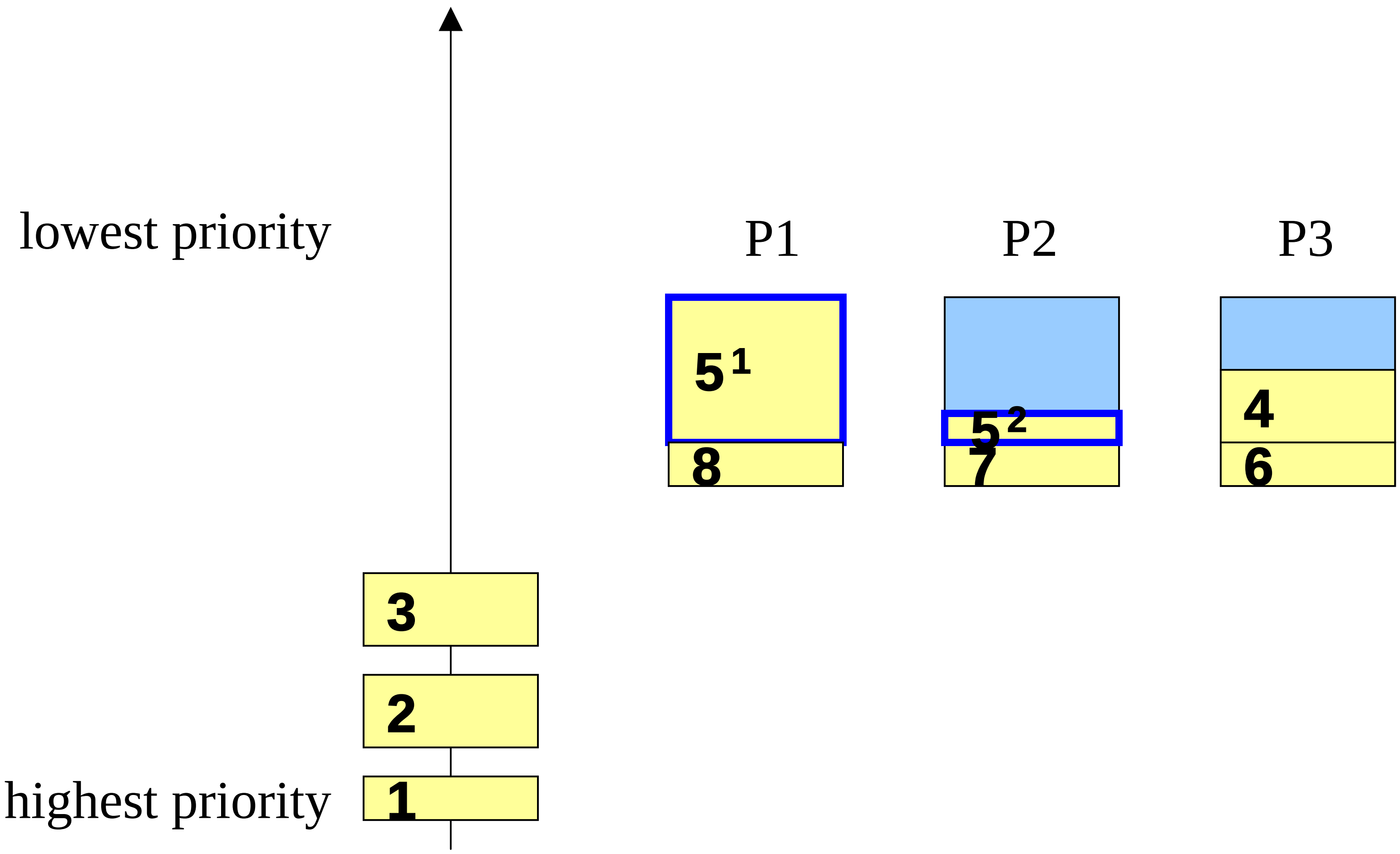
Our Algorithm

- problem of heavy tasks



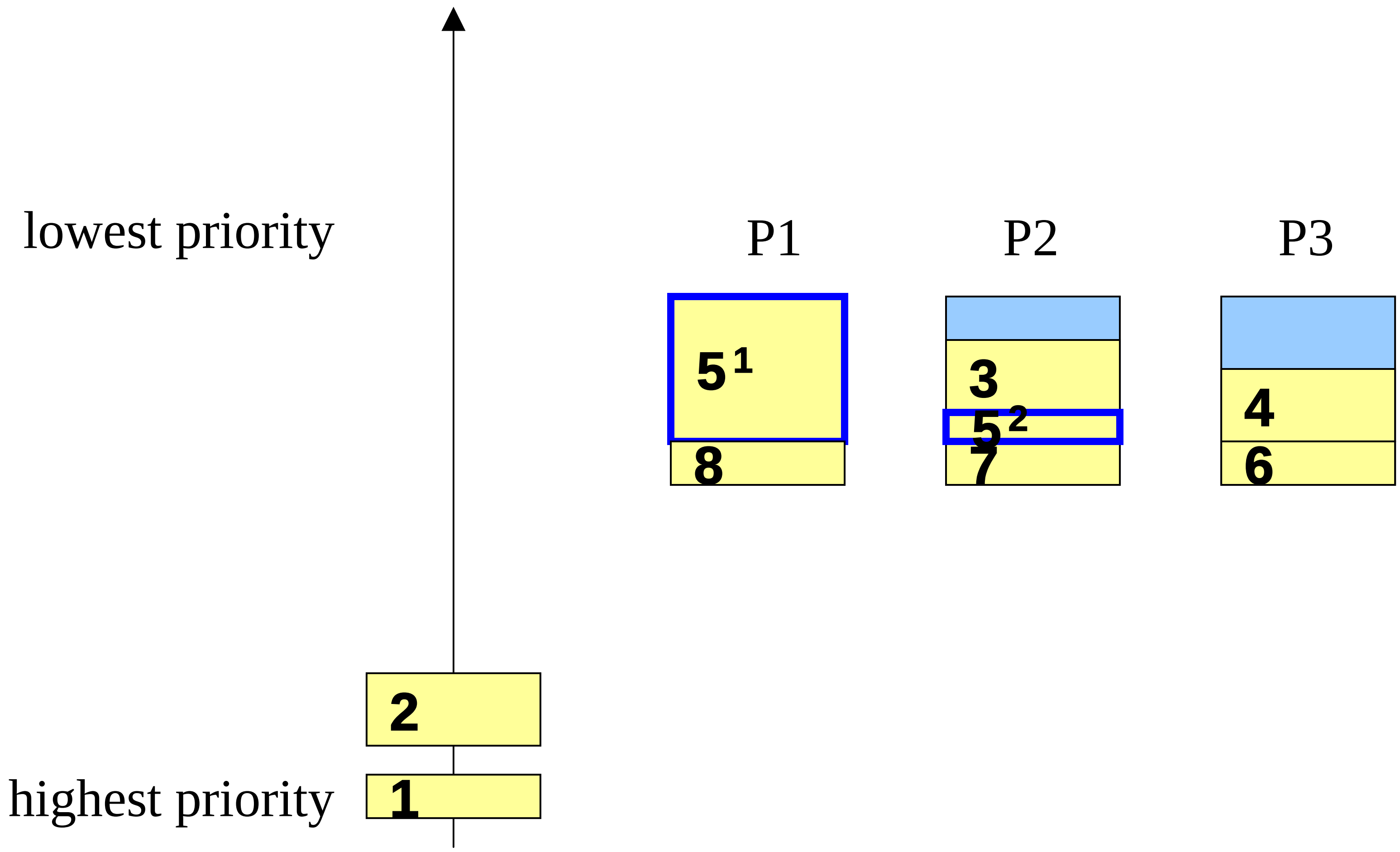
Our Algorithm

- problem of heavy tasks



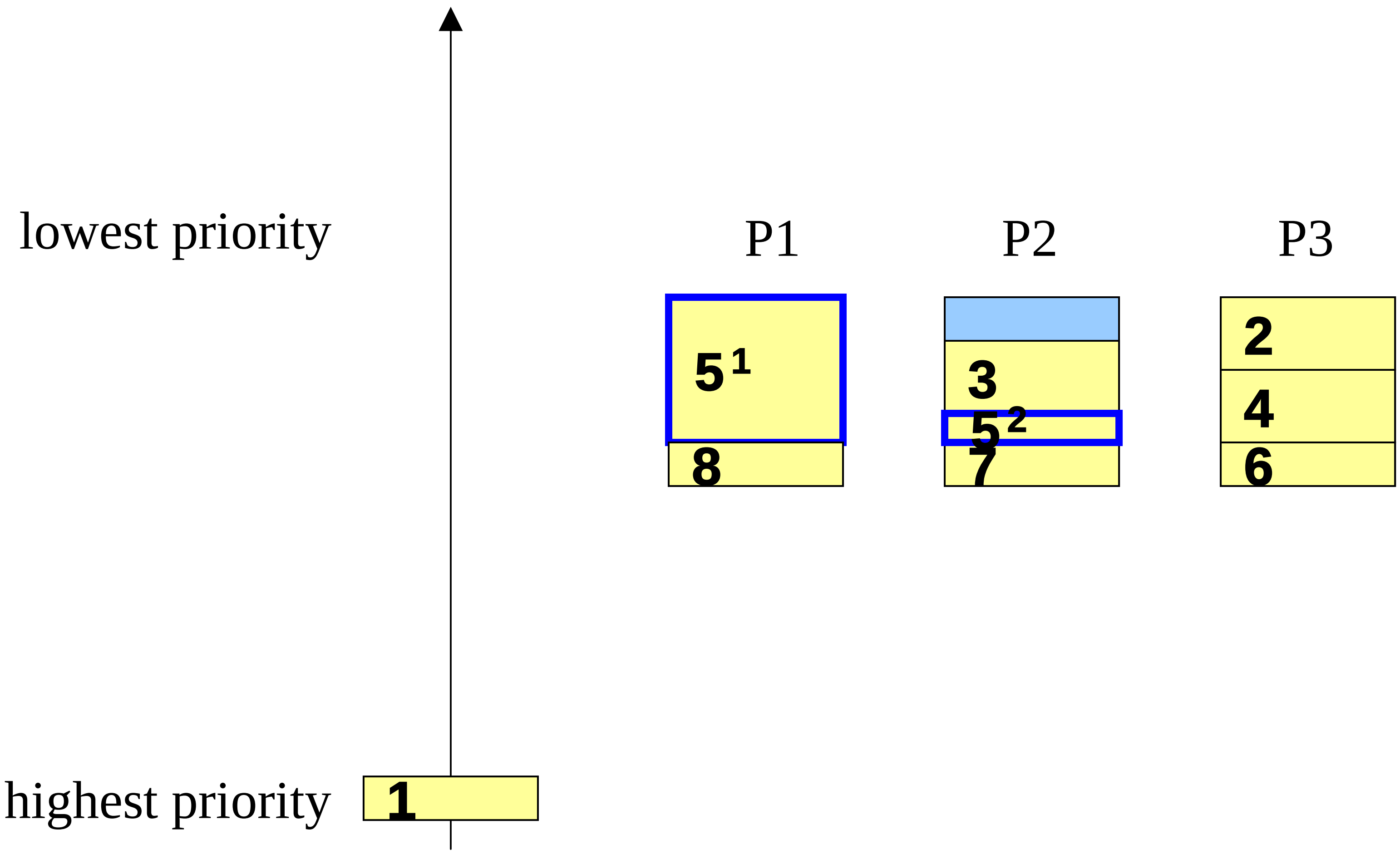
Our Algorithm

- problem of heavy tasks



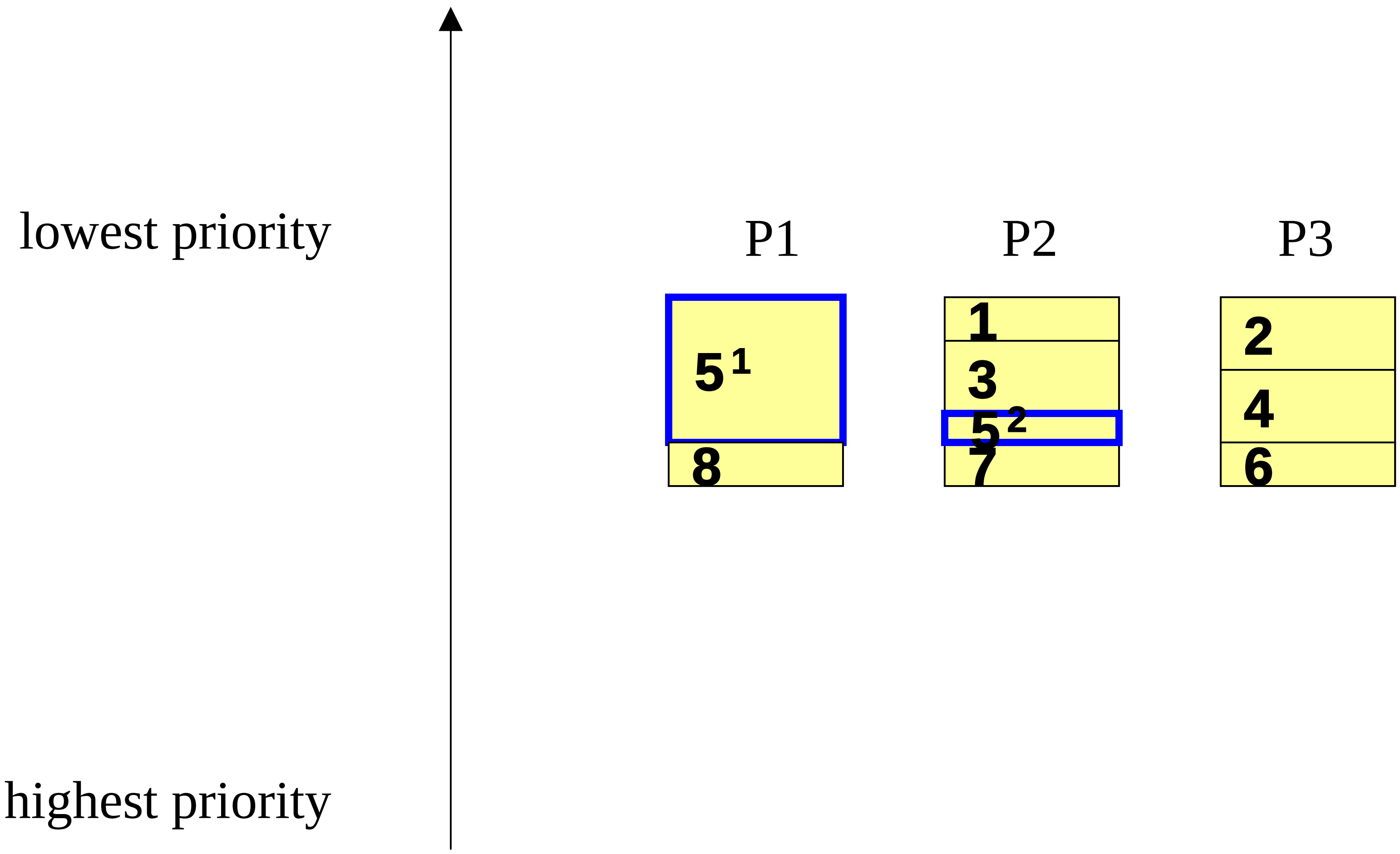
Our Algorithm

- problem of heavy tasks



Our Algorithm

- problem of heavy tasks



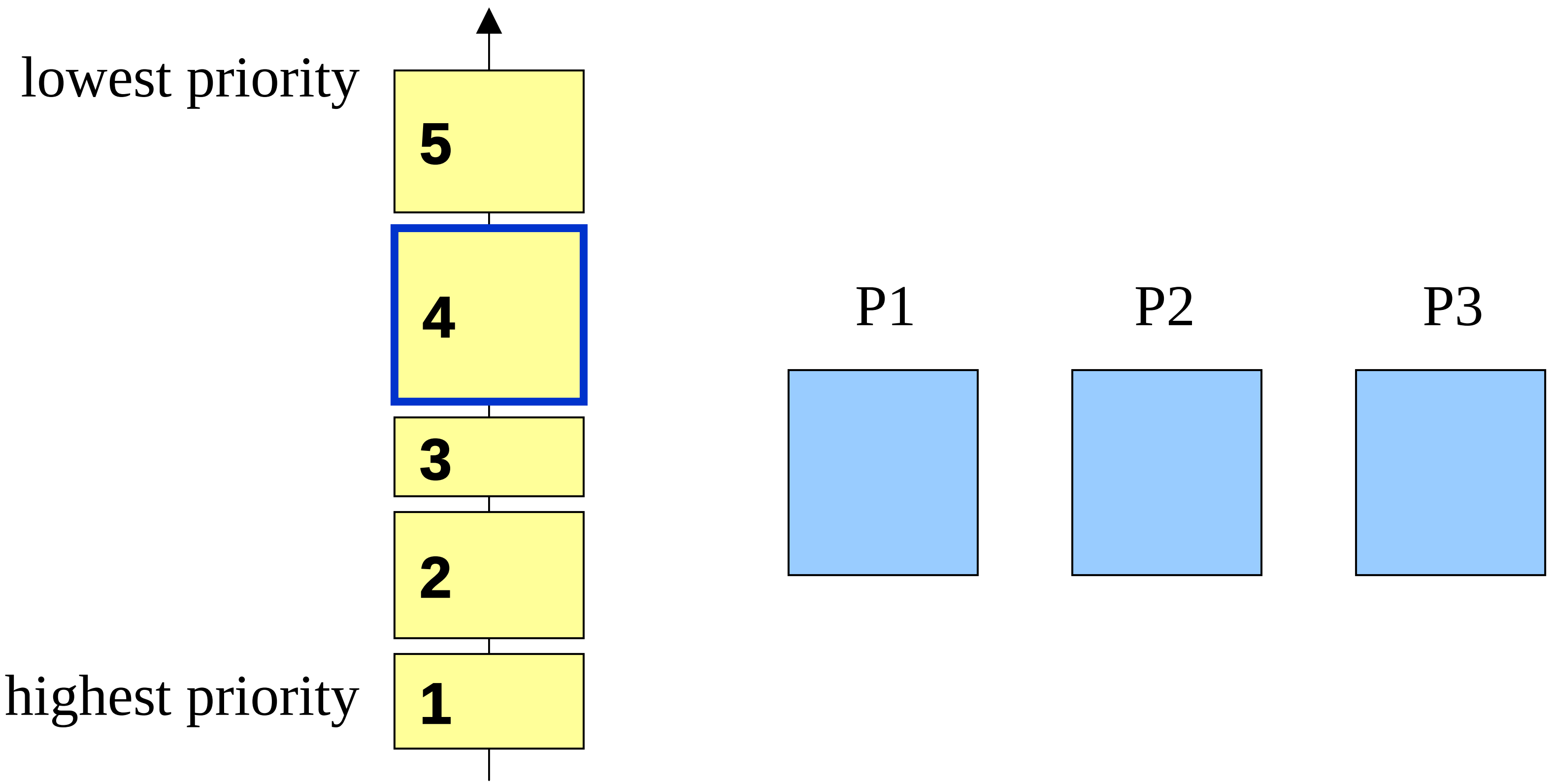
The idea works for all tasks!

- To get rid of the constraint

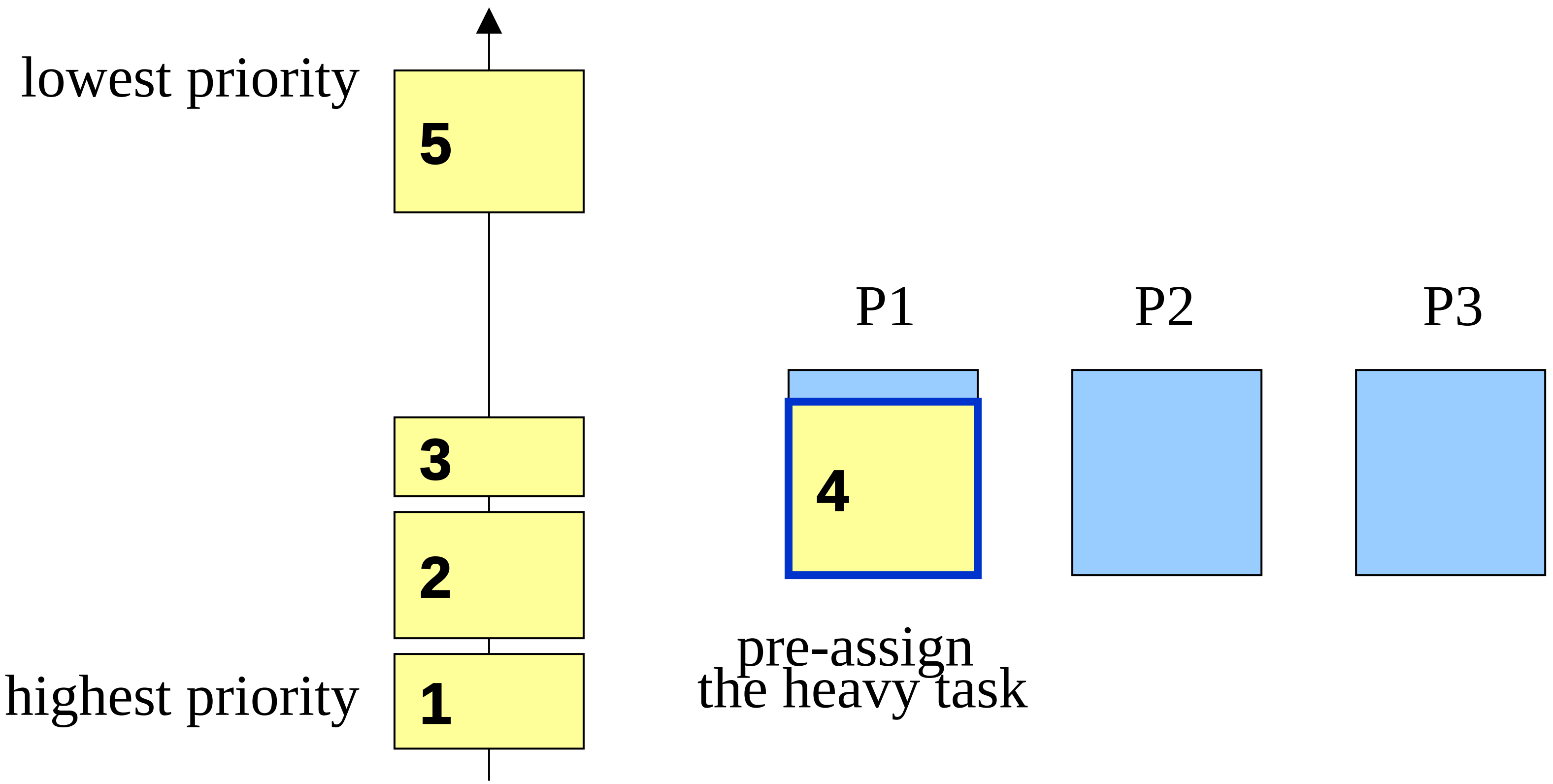
$$U_i \leq \frac{\Theta(N)}{1+\Theta(N)}$$

  - pre-assign tasks with high utilization

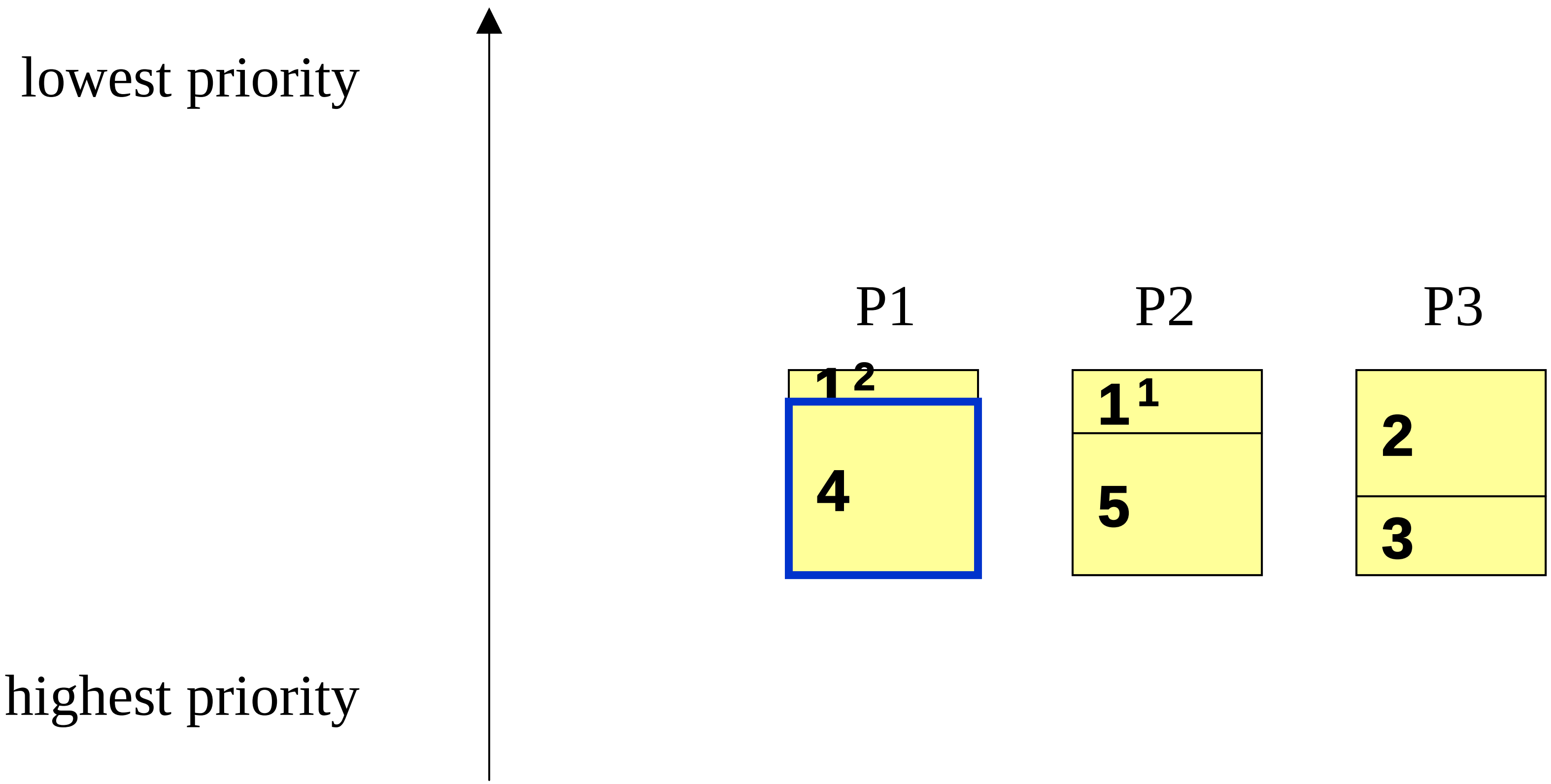
The idea works for all tasks!



The idea works for all tasks!



The idea works for all tasks!



Theorem

By introducing the pre-assignment to the algorithm, we have

$$\frac{\sum C_i/T_i}{M} \leq N(2^{1/N} - 1)$$

$$\Rightarrow \quad \text{the task set is schedulable}$$

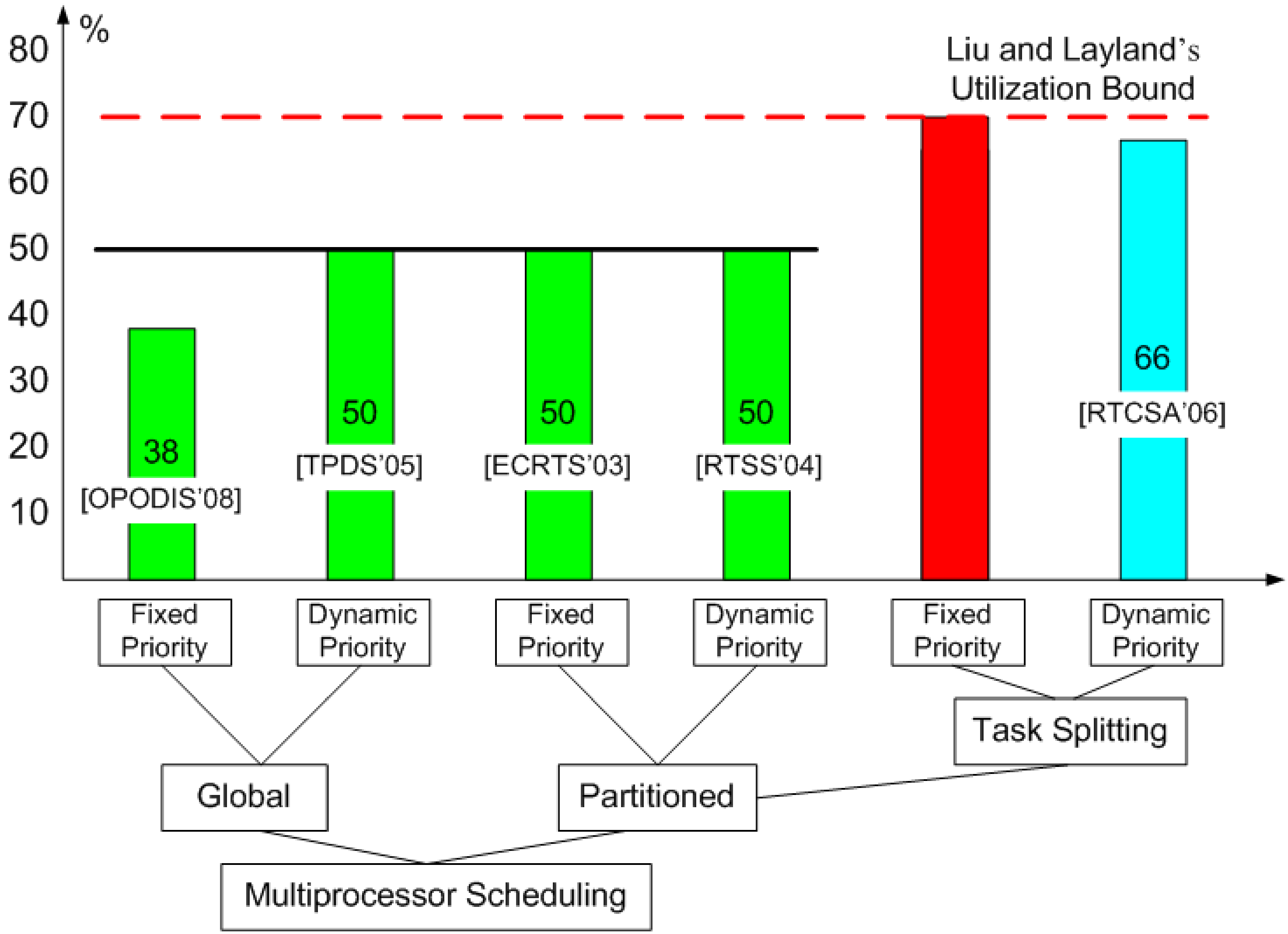
Conclusion

- Proposed multiprocessor scheduling algorithms with Liu and Layland's utilization bound

  - works on "light" task sets with a simple width-first algorithm

  - works on any task set with a hybrid algorithm
    - pre-assigning

Conclusion



%
80
70
60
50
40
30
20
10
Liu and Layland's
Utilization Bound
38
[OPODIS'08]
50
[TPDS'05]
50
[ECRTS'03]
50
[RTSS'04]
66
[RTCSA'06]
Fixed Priority
Dynamic Priority
Fixed Priority
Dynamic Priority
Fixed Priority
Dynamic Priority
Task Splitting
Global
Partitioned
Multiprocessor Scheduling

UPPSALA UNIVERSITET

THANKS!

UPPSALA UNIVERSITET

- **[Liu1973]** C.L. Liu and James Layland, Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment
- **[Andersson03ECRTS]** Bjorn Andersson, Jan Jonsson: The Utilization Bounds of partitioned and Pfair Static-Priority Scheduling on multiprocessors are 50%. ECRTS 2003: 33-40
- **[Andersson08OPODIS]** Bjorn Andersson: Global Static-Priority Preemptive Multiprocessor Scheduling with Utilization Bound 38%. OPODIS 2008: 73-88
- **[Andersson06RTCSA]** Bjorn Andersson, Eduardo Tovar: Multiprocessor Scheduling with Few preemption. RTCSA 2006: 322-334
- **[Andersson01RTSS]** Bjorn Andersson, Sanjoy K. Baruah, Jan Jonsson: Static-Priority Scheduling on multiprocessors. RTSS 2001: 193-202
- **[Baker05TPDS]** Theodore P. Baker: An Analysis of EDF Schedulability on a Multiprocessor. IEEE Trans. Parallel Distrib. Syst. 16(8): 760-768 (2005)
- **[Lakshmanan09ECRTS]** Karthik Lakshmanan, Ragunathan Rajkumar, John Lehoczky Partitioned Fixed-Priority Preemptive Scheduling for Multi-core Processors. ECRTS 20006
- **[Lopez04RTSS]** J. M. Lopez, J. L. Diaz, and D. F. Garcia, ”Utilization Bounds for EDF Scheduling on Real-Time Multiprocessor Systems”, RTSS 2004.
- **[Oh98]** D. Oh and T. P. Baker. Utilization bounds for n-processor Rate Monotone scheduling with static processor assignment. In Real-Time Systems, 1998.