# Power-Aware Scheduling with Dynamic Priorities

**Raj Rajkumar**

Lecture #11



## Outline

- Static voltage scaling with EDF
- Cycle-conserving Dynamic Voltage Scaling
- Look-Ahead Dynamic Voltage Scaling

# Quiz 2 Statistics

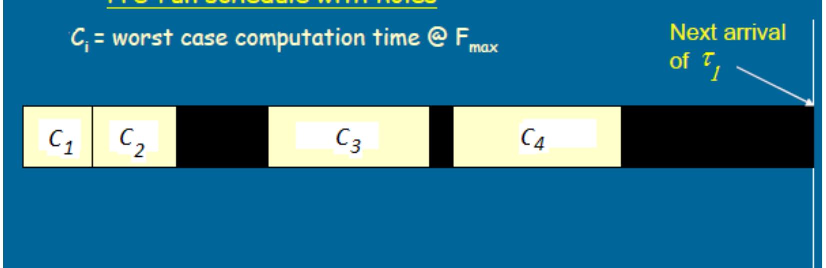
| | |
|---|---|
| Average | 65.49 |
| Std. Deviation | 9.85 |
| Max | 77 |
| Min | 40 |



# Static Voltage Scaling w/ EDF: Motivation

Pre-run schedule with holes

$C_i$ = worst case computation time @ $F_{max}$




Holes in the pre-run schedule imply:

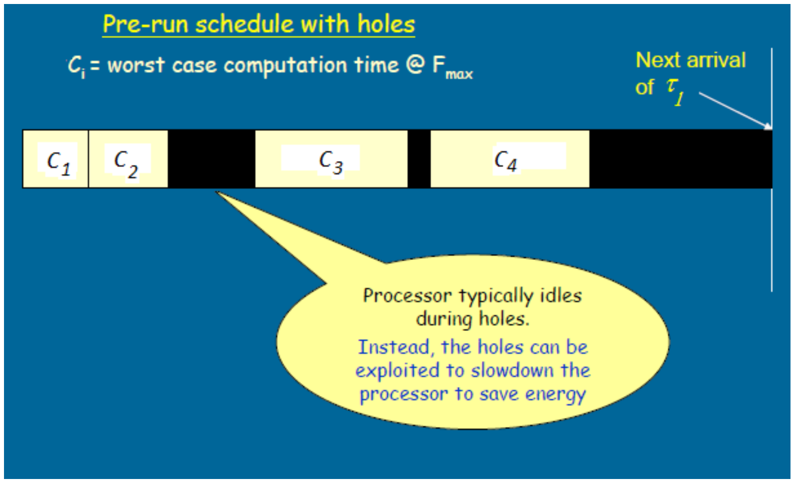
EDF Test:

$$\sum( Ci/Ti ) < 1 \quad \text{at frequency} = F_{max}$$

In other words, whenever $\sum( Ci/Ti ) < 1$ there are holes in the EDF schedule

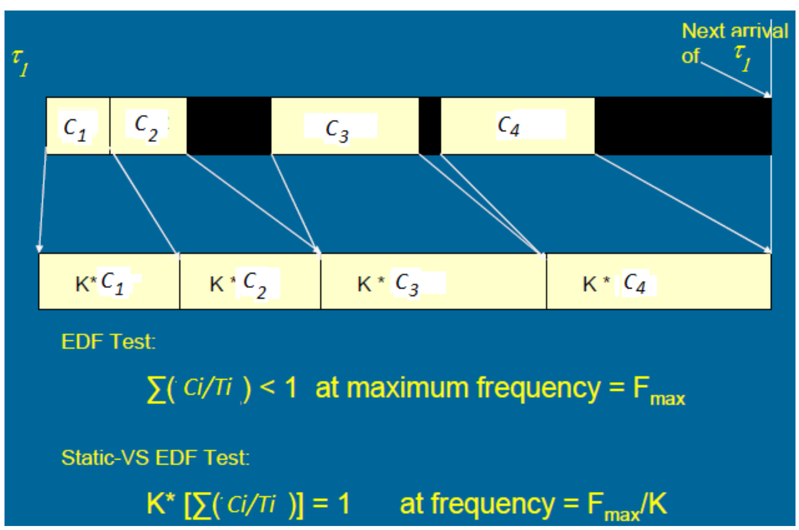
## Static Voltage Scaling w/ EDF: Exploiting "Holes"

"Holes" in the schedule correspond to idle times on the processor.






## Static Voltage Scaling w/ EDF

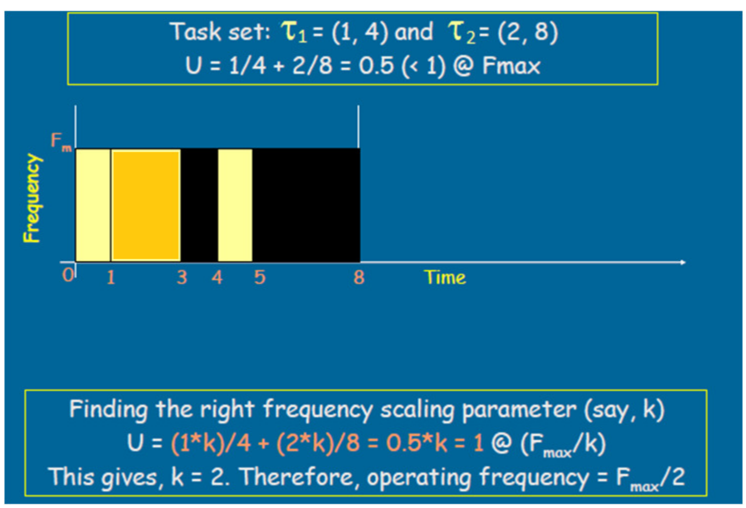


## Static Voltage Scaling: Example Taskset

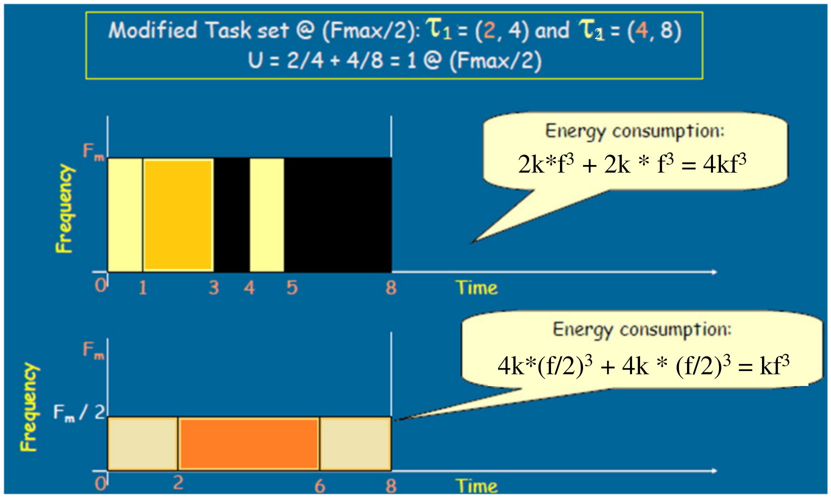
- Taskset: $\tau_1$ = (1, 4) and $\tau_2$ = (2, 8)
- U = 1/4 + 2/8 = 0.5 @ $F_{max}$
- What is the value of scaling factor *k* at which the taskset is still schedulable @ ($F_{max}$ / *k*)?
  - U = (1*k*)/4 + (2*k*)/8 = k (1/4 + 2/8) = 1
  - Solving for *k*,
    - *k* = 2
  - Therefore, we should operate at *f* = $F_{max}$ / 2 in order to meet all task deadlines.



## Static Voltage Scaling: Example (1 of 2)

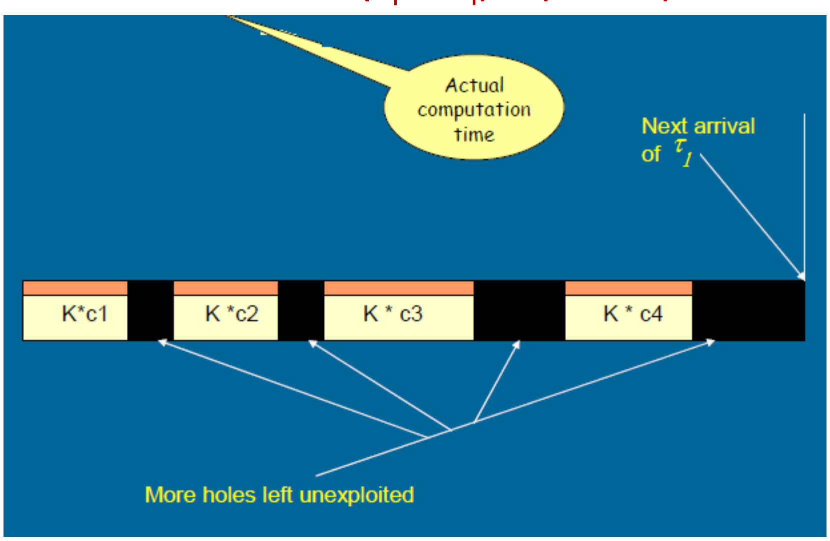


## Static Voltage Scaling: Example (2 of 2)

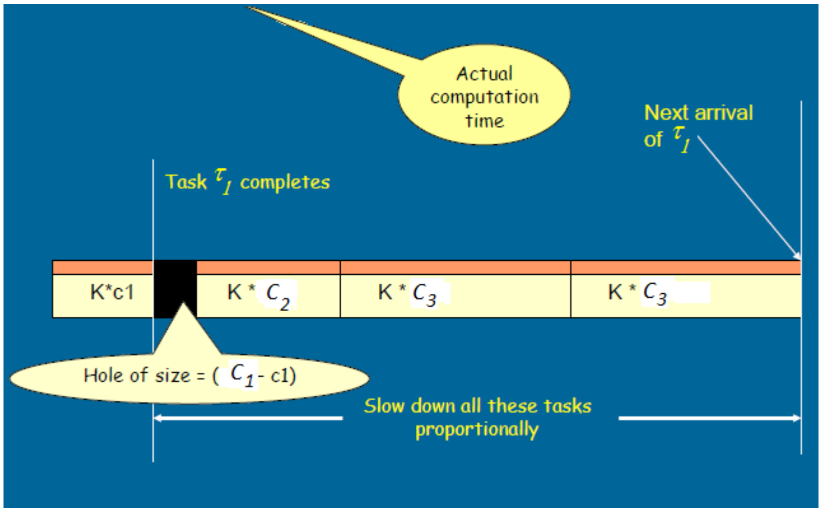





## What if ($c_i < C_i$)? (1 of 3)

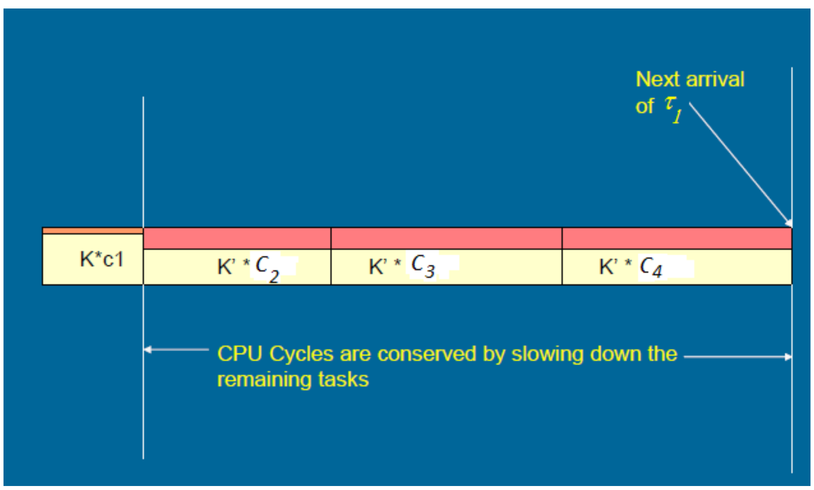


## What if ($c_i < C_i$)? (2 of 3)






## What if ($c_i < C_i$)? (3 of 3)



# Cycle-Conserving EDF

- **Idea**: When a task instance is released, we know its worst-case execution time but not its actual execution time. Let's assume worst-case execution on arrival.
- When the task instance completes, the actual cycles used are compared against the worst case, and cycles saved are given to run other remaining tasks at lower frequency.

***select_frequency():***

use lowest frequency such that $U^1 + U^2 + \ldots U^n <= 1$

***upon task_release ($\tau_i$):***

set $U^i$ to $C_i$ / $T_i$

select_frequency()

***upon task_completion($\tau_i$):***

set $U^i$ to $c_i$ / $T_i$ // $c_i$ is the actual cycles used in this invocation

select_frequency()

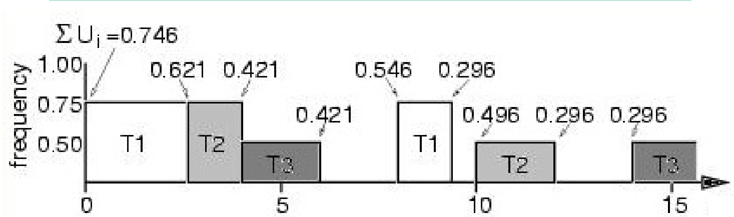


# Cycle-Conserving EDF Example

| Task | $C_i$ | Period | $U_i$ |
|---|---|---|---|
| 1 | 3 | 8 | 0.375 |
| 2 | 3 | 10 | 0.300 |
| 3 | 1 | 14 | 0.071 |

Actual Execution Times at $f_{max}$

| Task | Instance #1 $c_i$ | Instance #2 $c_i$ |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 1 | 1 |
| 3 | 1 | 1 |

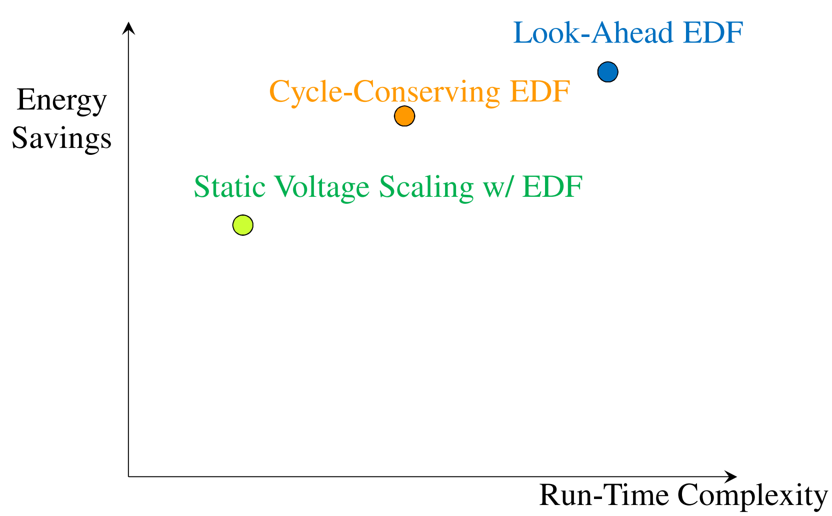


## Look-Ahead EDF

- Defers as much work as possible.
- Sets the operating frequency to meet the minimum work that must be done now to ensure all future deadlines to be met.



## Relative Performance



## Conclusions

- RT-DVS schemes are designed to ensure
  - predictability while saving as much energy as possible in real-time systems.