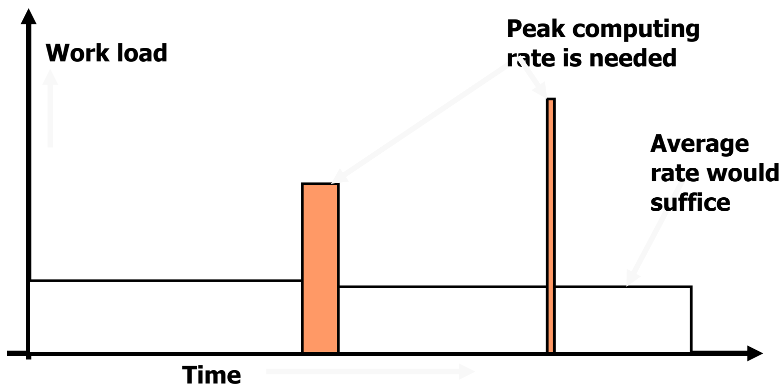
Graduate Operating Systems
(Embedded Systems & Scheduling)

Fall 2020

Paper "DVS"

- Real-Time Systems
- Dynamic Voltage Scaling (DVS, DFS)
- Over-designed systems (peak performance)
- Periodic task model
- Earliest Deadline First (EDF)
- Rate Monotonic Scheduling (RM)
- Schedulability test

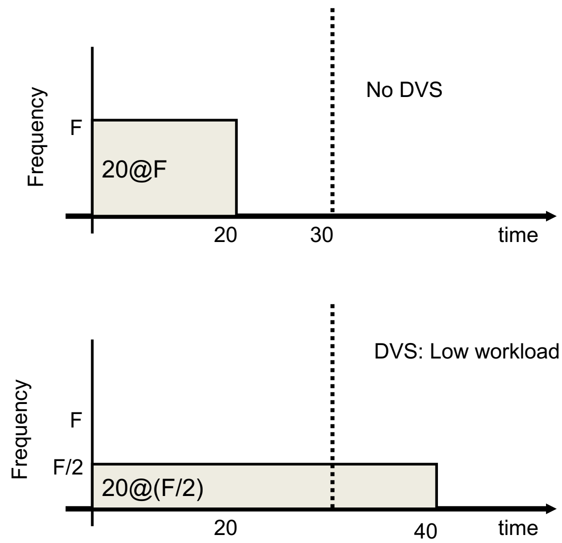
Peak vs. Average Performance



DVS Fundamentals

- Processors are based on CMOS technology where dynamic power is the bottleneck
- Dynamic power (due to switching activity)
  - Power depends on V^2 and f
  - Achievable f depends on V
- Energy = $P * t_{execution}$

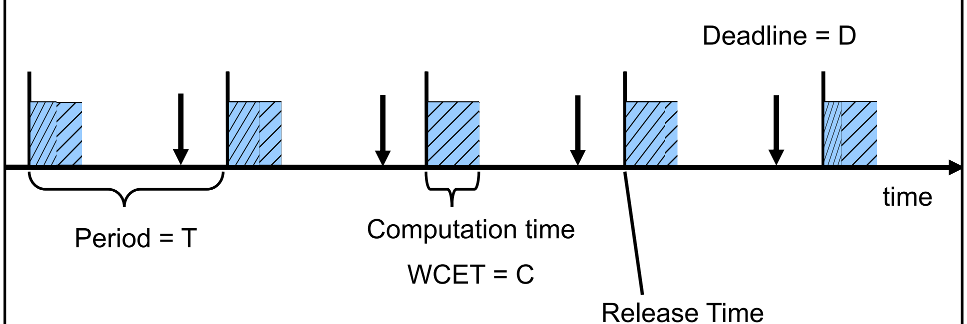
DVS Fundamentals



Periodic Task Model

Task = {T, C, D}

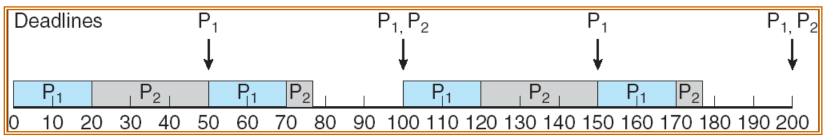
jobs (j_1, j_2, j_3, …)



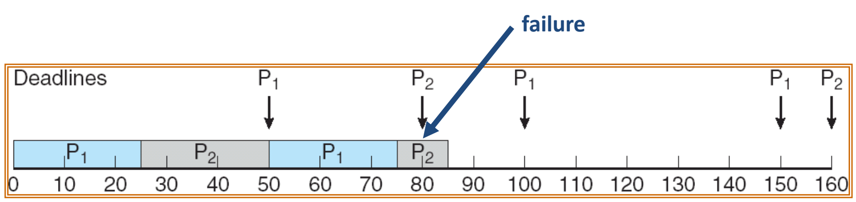
RMS (Rate Monotonic Scheduling)

Process P_1: service time = 20, period = 50, deadline = 50

Process P_2: service time = 35, period = 100, deadline = 100



Missed Deadlines with RMS



Process P_1: service time = 25, period = 50, deadline = 50
Process P_2: service time = 35, period = 80, deadline = 80

RMS is guaranteed to work if

$$u = \sum_{i=1}^{N} \frac{t_i}{p_i} \le N\left(\sqrt[N]{2} - 1\right);$$

N = number of processes

sufficient condition

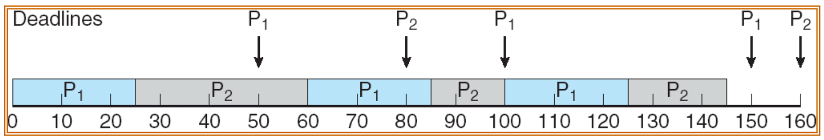
$$\lim_{N \to \infty} N\left(\sqrt[N]{2} - 1\right) = \ln 2 \approx 0.693147$$

N	$N\left(\sqrt[N]{2} - 1\right)$
2	0,828427
3	0,779763
4	0,756828
5	0,743491
10	0,717734
20	0,705298

EDF (Earliest Deadline First)

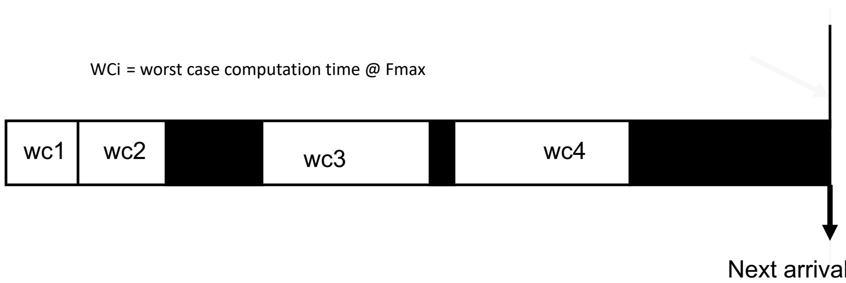
Process P_1: service time = 25, period = 50, deadline = 50

Process P_2: service time = 35, period = 80, deadline = 80



Static Voltage Scaling EDF: Motivation

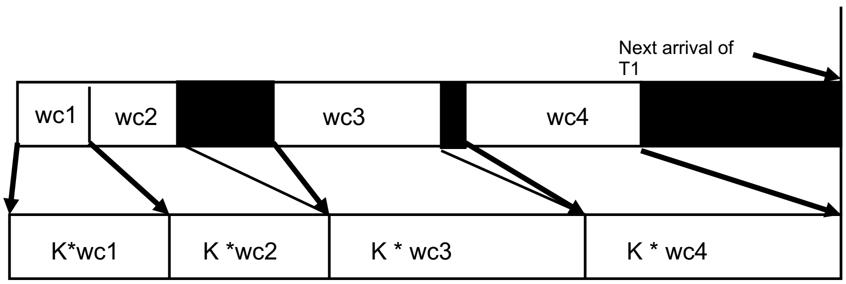
WCi = worst case computation time @ Fmax



Holes in the schedule imply:

EDF Test: $\sum(wc_i/p_i) < 1$ at frequency = F_{max}

Static Voltage Scaling EDF



EDF Test:

$\sum(wc_i/p_i) < 1$ at maximum frequency = F_{max}

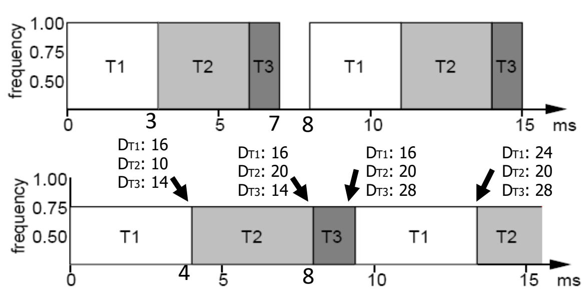
Static-VS EDF Test:

$K^* [\sum(wc_i/p_i)] = 1$ at frequency = F_{max}/K

Static EDF: Example

Task	Computing Time	Period
1	3 ms	8 ms
2	3 ms	10 ms
3	1 ms	14 ms

Available frequencies:
1.00, 0.75



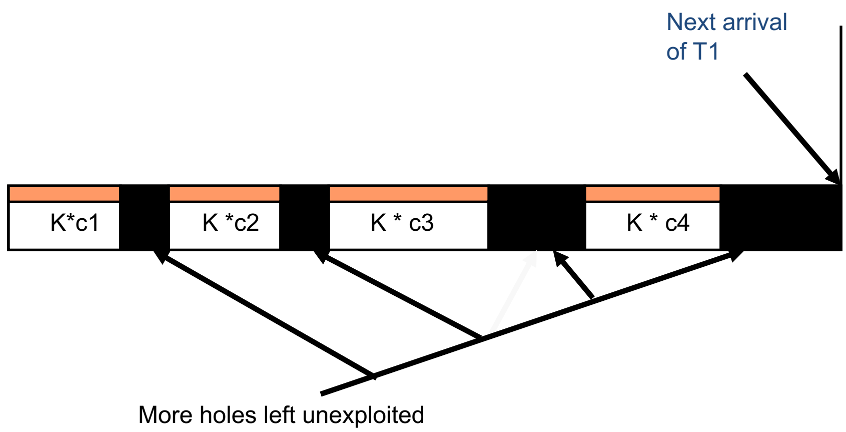
Schedulability test for $\alpha = 1.00$

3/8 + 3/10 + 1/14 ≤ 1 ---> **Return true**

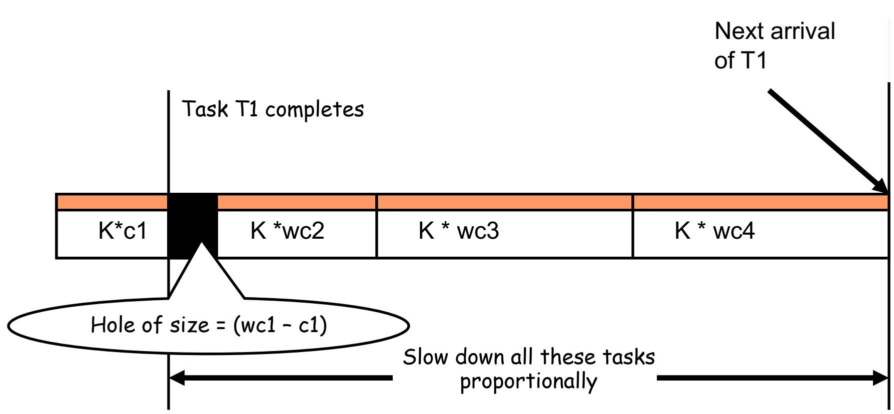
Schedulability test for $\alpha = 0.75$

3/8 + 3/10 + 1/14 ≤ 0.75 ---> **Return true**

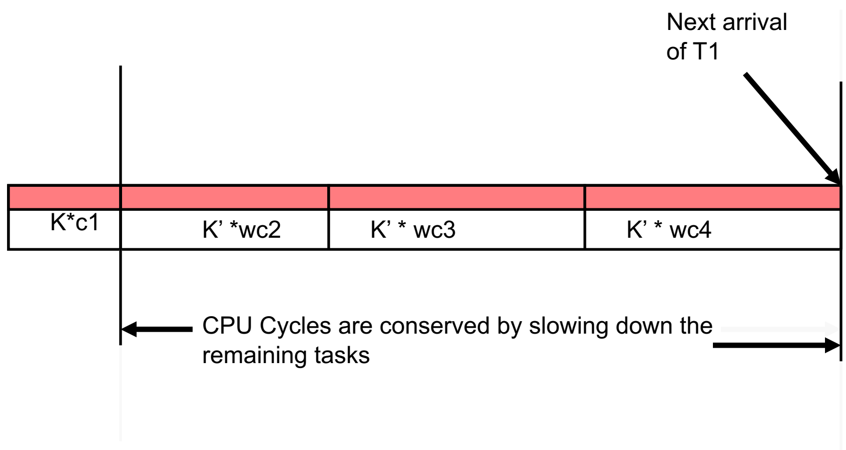
What if $C_i < WC_i$?



What if $C_i < WC_i$?



What if $C_i < WC_i$?

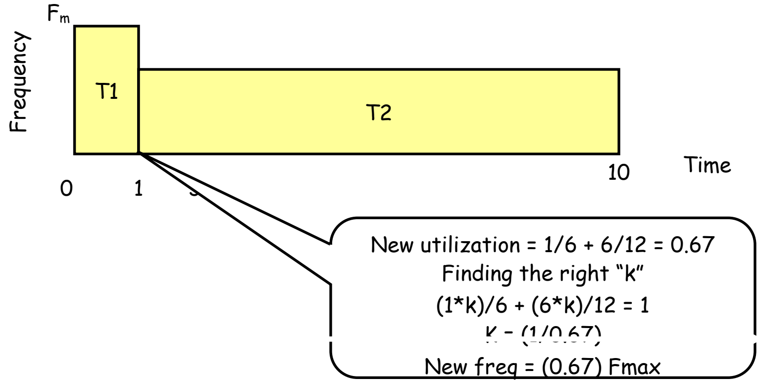


Cycle Conserving RT-DVS

- When a task set completes its first release, compare real execution time with worst case specified initially.
- Any idle time in that period can be used to conserve energy.
- Rescale frequency that avoid idle cycles, surplus time is used to run other remaining tasks at lower frequency.

Cycle Conserving EDF: Example

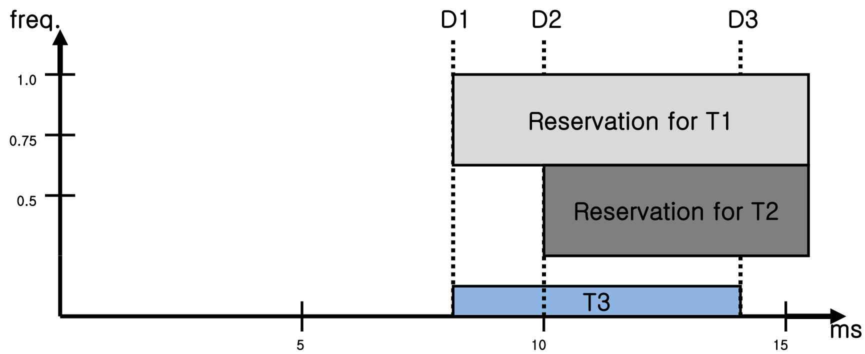
Task set @ (Fmax): T1 = (3,6) and T2 = (6,12)
U = 3/6 + 6/12 = 1 @ (Fmax)



Look-Ahead EDF

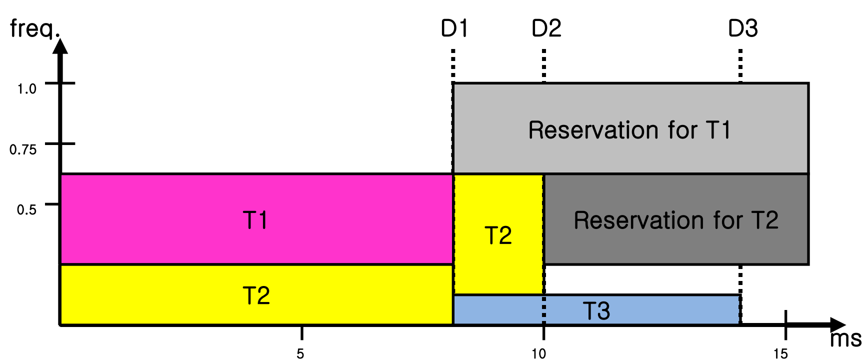
- Defer as much works as possible and set initially to the minimum possible frequency.
- Hence at later stage if a task uses much less than it worst case, deferred work may never be needed.
- It ensures that there are sufficient cycles available for each task to meet its deadline after reserving cycles for higher priority jobs.
- Best saving of energy.

Look-ahead EDF Step 1 of 6



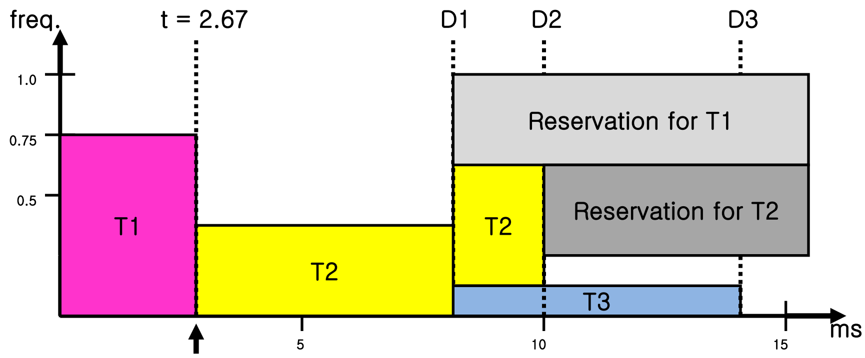
(a) Plan to defer T3's execution time until after D1 but by D3

Look-ahead EDF Step 2 of 6



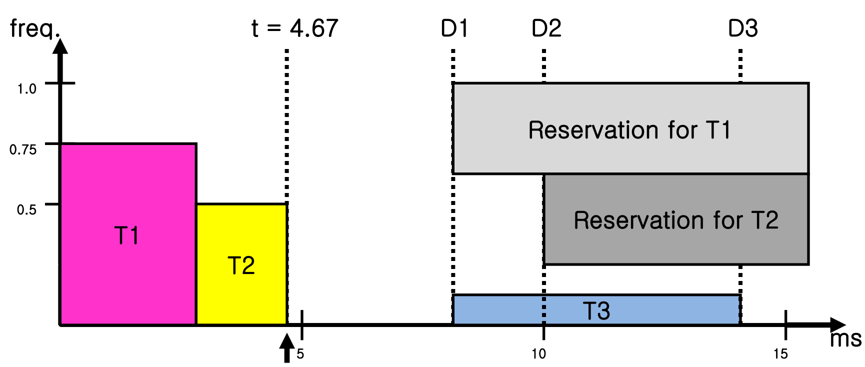
(b) Find u so as to finish T1 by D1 and T2 by D2

Look-ahead EDF Step 3 of 6



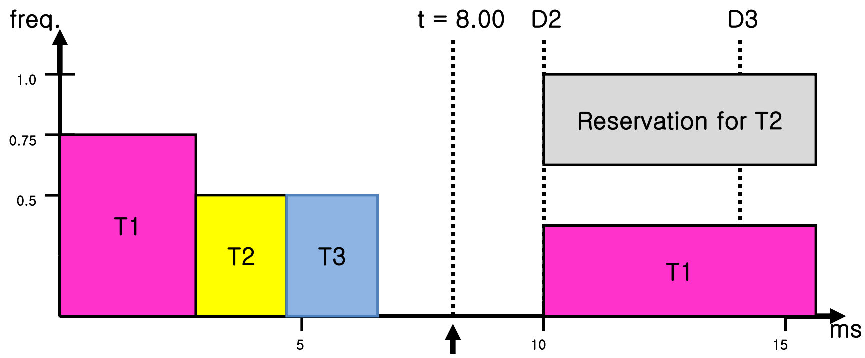
(c) u = 0.75, T1 finishes earlier, find new u for T2 to finish by D2

Look-ahead EDF Step 4 of 6



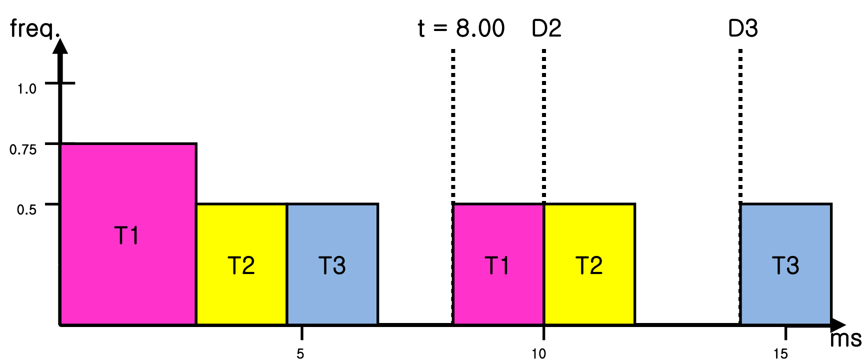
(d) α = 0.5, T2 finishes earlier, enough time until D1, but EDF is work conserving, launch T3 at u = 0.5

Look-ahead EDF Step 5 of 6



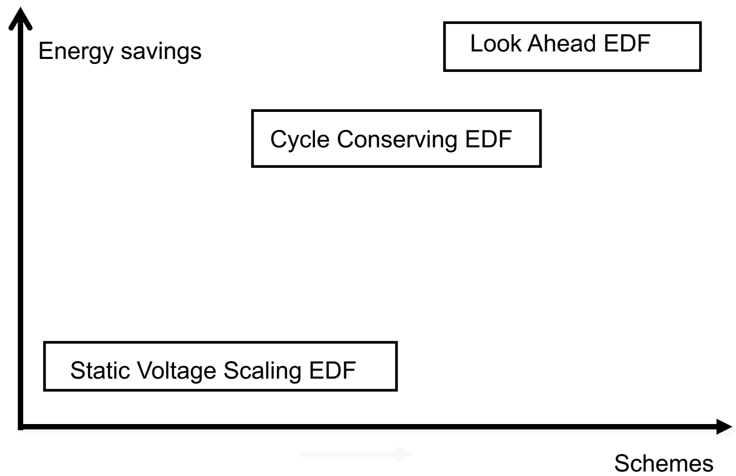
(e) Guess for T1 again

Look-ahead EDF Step 6 of 6



(f) u= 0. 5, every task is dynamically scheduled successfully

Relative Performance



Implementation

- Scheduler hooking in the kernel

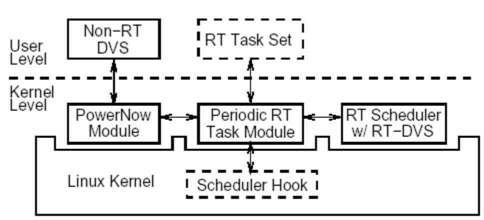


Figure 14: Software architecture for RT-DVS implementation