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

- Peak computing rate is needed
- Average rate would suffice

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 = $P \cdot t_{\text{execution}}$
DVS Fundamentals

No DVS

DVS: Low workload

Periodic Task Model

Task = \{T, C, D\}

jobs (j_1, j_2, j_3, \ldots)

Period = T

Deadline = D

Computation time
WCET = C

Release Time
RMS (Rate Monotonic Scheduling)

Process P₁: service time = 20, period = 50, deadline = 50
Process P₂: service time = 35, period = 100, deadline = 100

Missed Deadlines with RMS

Process P₁: service time = 25, period = 50, deadline = 50
Process P₂: service time = 35, period = 80, deadline = 80

RMS is guaranteed to work if

\[ N \left( \sqrt{2} - 1 \right) \leq \sum_{i} \frac{t_i}{p_i} \]

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

---

<table>
<thead>
<tr>
<th>N</th>
<th>N(\sqrt{2} - 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.828427</td>
</tr>
<tr>
<td>3</td>
<td>0.779763</td>
</tr>
<tr>
<td>4</td>
<td>0.756828</td>
</tr>
<tr>
<td>5</td>
<td>0.743491</td>
</tr>
<tr>
<td>10</td>
<td>0.717734</td>
</tr>
<tr>
<td>20</td>
<td>0.705298</td>
</tr>
</tbody>
</table>
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

$W_{Ci} =$ worst case computation time @ $F_{\text{max}}$

Holes in the schedule imply:

EDF Test: $\sum (W_{Ci}/p_i) < 1$ at frequency $= F_{\text{max}}$
**Static Voltage Scaling EDF**

EDF Test:
\[
\sum \left( \frac{wc}{p} \right) < 1 \quad \text{at maximum frequency} = F_{\text{max}}
\]

Static-VS EDF Test:
\[
K \times \sum \left( \frac{wc}{p} \right) = 1 \quad \text{at frequency} = F_{\text{max}}/K
\]

**Static EDF: Example**

<table>
<thead>
<tr>
<th>Task</th>
<th>Computing Time</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3 ms</td>
<td>8 ms</td>
</tr>
<tr>
<td>2</td>
<td>3 ms</td>
<td>10 ms</td>
</tr>
<tr>
<td>3</td>
<td>1 ms</td>
<td>14 ms</td>
</tr>
</tbody>
</table>

Available frequencies: 1.00, 0.75

Schedule test for \( \alpha = 1.00 \):
\[
\frac{3}{8} + \frac{3}{10} + \frac{3}{14} \leq 1 \rightarrow \text{Return true}
\]

Schedule test for \( \alpha = 0.75 \):
\[
\frac{3}{8} + \frac{3}{10} + \frac{3}{14} \leq 0.75 \rightarrow \text{Return true}
\]
What if $C_i < WC_i$?

More holes left unexploited

Next arrival of $T_1$

What if $C_i < WC_i$?

Slow down all these tasks proportionally

Hole of size = $(wc_i - c_i)$

Task $T_1$ completes

Next arrival of $T_1$
What if $C_i < WC_i$?

Next arrival of $T_1$

$K^*c_1$  $K' * wc_2$  $K' * wc_3$  $K' * wc_4$

CPU Cycles are conserved by slowing down the remaining tasks

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)

New utilization = 1/6 + 6/12 = 0.67
Finding the right “k”
(1*k)/6 + (6*k)/12 = 1
k = (1/0.67)
New freq = (0.67) 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.
(a) Plan to defer T3’s execution time until after D1 but by D3

(b) Find u so as to finish T1 by D1 and T2 by D2
(c) $u = 0.75$, $T_1$ finishes earlier, find new $u$ for $T_2$ to finish by $D_2$

Look-ahead EDF Step 4 of 6

(d) $\alpha = 0.5$, $T_2$ finishes earlier, enough time until $D_1$, but EDF is work conserving, launch $T_3$ 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

Energy savings

- Look Ahead EDF
- Cycle Conserving EDF
- Static Voltage Scaling EDF

Schemes

Implementation

- Scheduler hooking in the kernel

Figure 14: Software architecture for RT-DVS implementation