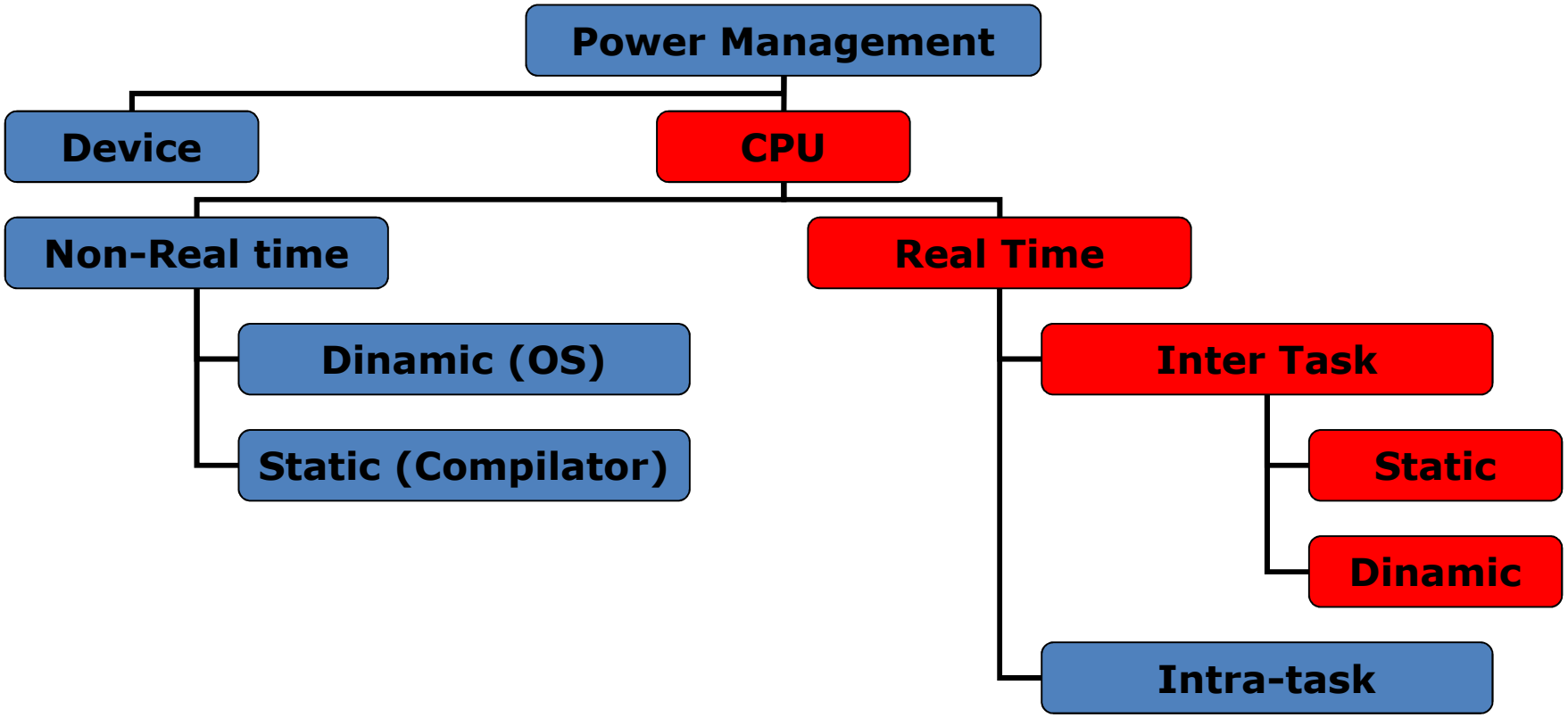
Sisteme Incorporate

Cursul 10

Dynamic Voltage Scaling

Taxonomia Power Management



Care sunt problemele real-time PM?

- Sistemele embedded moderne
  - Au unul sau mai multe procesoare puternice
  - Memorie de capacitate mare
  - Set bogat de periferice si interfete
    - LCD + touchscreen
    - Bluetooth, Wireless 802.11b/g
    - USB, Firewire
    - GSM, GPS etc.
  - Dar merg pe baterii

Abordari pentru PM

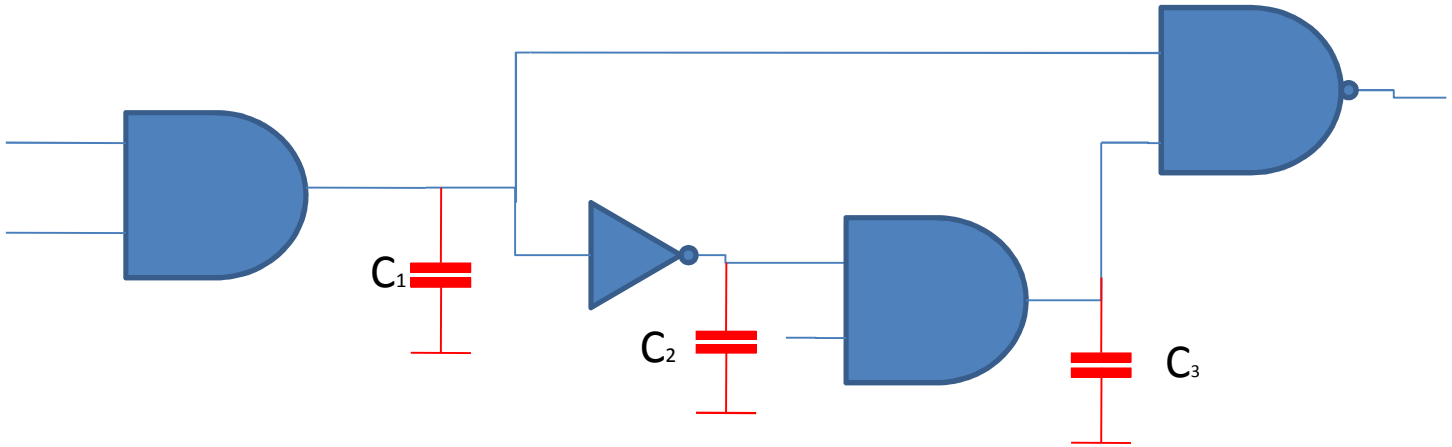
- Memorie
  - Partajeaza memoria in mai multe circuite
  - Deconecteaz-o atunci cand n-o folosesti
- Periferice
  - PM pe mai multe niveluri
  - Deconectare atunci cand nu sunt folosite

Processor Power Management

- Procesor
  - Pot sa dezactivez anumite subsisteme
    - Dificil (cunostinte serioase despre setul de instructiuni)
    - Depinde de arhitectura procesorului
  - Observatii:
    1. Performanta maxima a unui sistem embedded este cu cateva ordine de marime mai mare decat productivitatea medie a unui regim de lucru sustinut.
    2. Perioada de productivitate maxima a sistemului este cu cateva ordine de marime mai mica decat perioada de procesare minima.

Processor Power Management

- Performanta = f(frecventa)
- Pentru logica CMOS
  - frecventa ~ tensiunea de alimentare
- De ce?
  - Capacitatea parazita



Processor Power Management

- Dynamic Voltage Scaling (DVS)
  - Planificare facuta de OS (solutie universala)
  - Tine cont de constrangerile Real-Time (RT-DVS)
    - Periodicitatea task-urilor.
    - Prioritati diferite.
    - Respectarea termenelor limita.
  - Algoritmi de planificare DVS
    - Statici (RMS)
    - Dinamici (EDF)

Scalarea Statica a Tensiunii

- Observatii
  - Scaleaza frecventa de operare cu un factor α ($0 < \alpha \leq 1$); WCET pentru un task devine = c/α
  - Perioadele si deadline-urile raman neafectate
- Algoritm de selecte a frecventei minime de operare (α minim) pentru care planificatorul RMS/EDF atinge toate termenele limita
- Frecventa de operare este setata static si nu se schimba in timpul executiei setului de task-uri

Scalare Statica: EDF

- Conditia necesara si suficienta pentru EDF

$$U = \frac{C_1}{P_1} + \ldots\ldots\ldots + \frac{C_n}{P_n} \leq 1$$

- Cu factorul de scalare al frecventei α

$$\frac{C_1}{\alpha P_1} + \ldots\ldots\ldots + \frac{C_n}{\alpha P_n} \leq \alpha$$

- Algoritm: alege factorul minim α pentru care $U \leq \alpha$

Scalare Statica: RMS

- Conditia suficienta (dar nu si necesara):

$$c_1 \times \left\lceil \frac{p_i}{p_1} \right\rceil + c_2 \times \left\lceil \frac{p_i}{p_2} \right\rceil + \ldots\ldots\ldots + c_i \times \left\lceil \frac{p_i}{p_i} \right\rceil \leq p_i$$

- Cu factorul α de scalare:

$$c_1 \times \left\lceil \frac{p_i}{p_1} \right\rceil + c_2 \times \left\lceil \frac{p_i}{p_2} \right\rceil + \ldots\ldots\ldots + c_i \times \left\lceil \frac{p_i}{p_i} \right\rceil \leq \alpha \times p_i$$

Algoritm de planificare statica

EDF_test (α):

if ($C_1/P_1 + \cdots + C_n/P_n \leq \alpha$) return true;

else return false;

RM_test (α):

if ($\forall T_i \in \{T_1, \ldots, T_n | P_1 \leq \cdots \leq P_n\}$

$\lceil P_i/P_1 \rceil * C_1 + \cdots + \lceil P_i/P_i \rceil * C_i \leq \alpha * P_i$)

return true;

else return false;

select_frequency:

use lowest frequency $f_i \in \{f_1, \ldots, f_m | f_1 < \cdots < f_m\}$

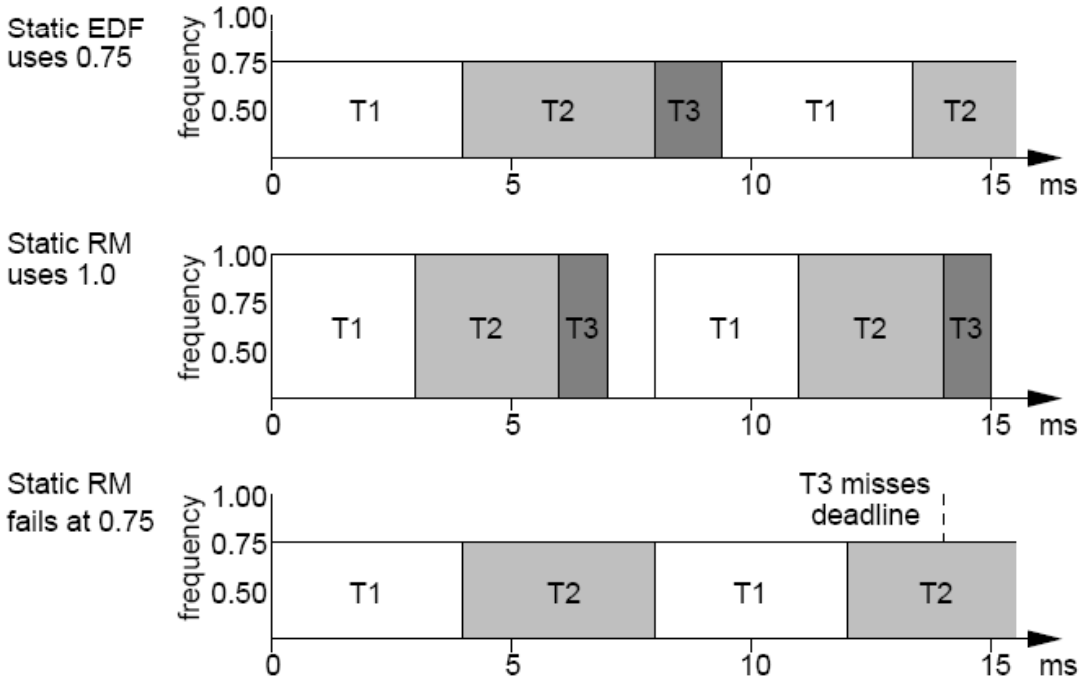
such that RM_test(f_i/f_m) or EDF_test(f_i/f_m) is true.

Scalare Statica. Exemplu

Timpi calculati pentru α=1

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

$$U = 3/8 + 3/10 + 1/14 = 0.746$$



Cycle-conserving RT-DVS

- Cand un task este preemptat
  - Nu se cunosc timpii de executie
  - Estimare conservatoare: WCET = c
- La completarea unui task
  - Calculeaza ciclii nefolositi
  - Se pot folosi acesti cicli pentru alte task-uri prin reducerea frecventei procesorului in perioada lor de operare
- Nu trebuie sa violeze nici un deadline

Cycle-conserving EDF

- La preemptarea unui task
  - Calculeaza utilizarea totala estimand la WCET
- La terminarea unui task
  - Recalculeaza utilizarea folosind timpii de executie masurati in ciclul anterior
  - Foloseste aceasta valoare pentru a scala frecventa

$$\frac{cc_1}{P_1} + \ldots + \frac{cc_i}{P_i} \ldots\ldots + \frac{C_n}{P_n} \leq \alpha$$

Cycle-conserving EDF

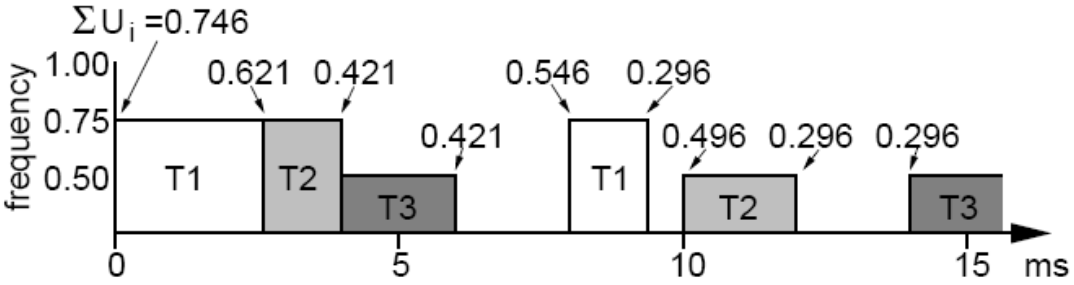
- Exemplu algoritm EDF:

select_frequency():
    use lowest freq. $f_i \in \{f_1, \ldots, f_m | f_1 < \cdots < f_m\}$
    such that $U_1 + \cdots + U_n \leq f_i/f_m$

upon task_release(T_i):
    set U_i to C_i/P_i;
    select_frequency();

upon task_completion(T_i):
    set U_i to cc_i/P_i;
        /* cc_i is the actual cycles used this invocation */
    select_frequency();

Exemplu



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

Trei frecvente setabile:
1, 0.75 si 0.5*fmax.

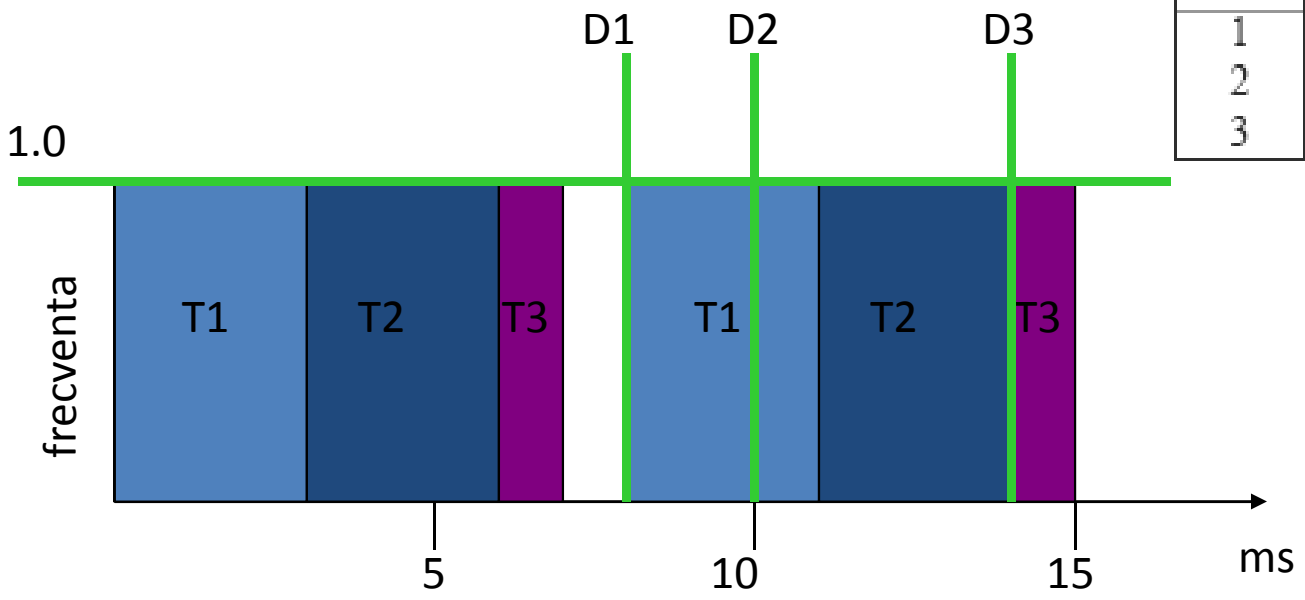
Task	Invocation 1	Invocation 2
1	2 ms	1 ms
2	1 ms	1 ms
3	1 ms	1 ms

- U = 3/8 + 3/10 + 1/14 = 0.746
- U = 2/8 + 3/10 + 1/14 = 0.621
- U = 2/8 + 1/10 + 1/14 = 0.421
- U = 2/8 + 1/10 + 1/14 = 0.421
- U = 3/8 + 1/10 + 1/14 = 0.546
- U = 1/8 + 1/10 + 1/14 = 0.296
- U = 1/8 + 3/10 + 1/14 = 0.496

Cycle-conserving RMS

- Complexitatea pentru testarea planificabilitatii unei scheme RMS este $O(n^2)$ unde n este numarul de procese
  - Nu este fezabil; compromite simplitatea RMS
- Abordare mai buna:
- Incepe cu RMS scalat static
- Asigura-te ca RMS cu conservare de cicli are intotdeauna rezultate mai bune (sau cel putin aceleasi) cu RMS scalat static (care foloseste WCET pentru a estima deadline-urile)

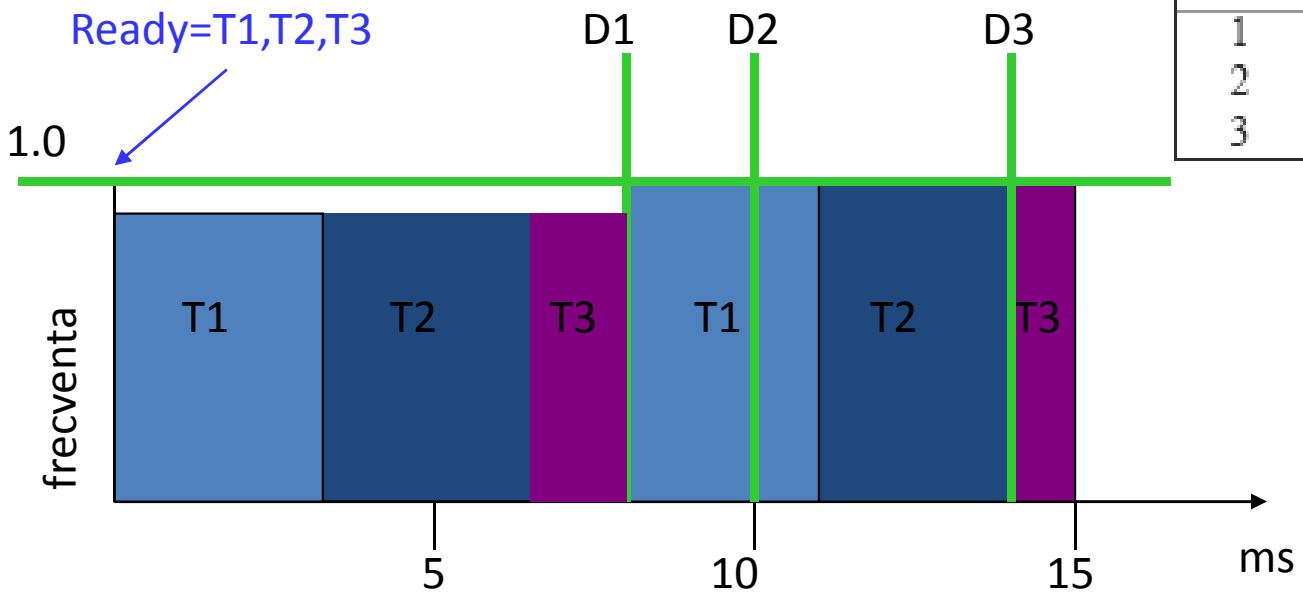
Exemplu



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

- Incepe cu RMS static
- In acest caz α = 1.0

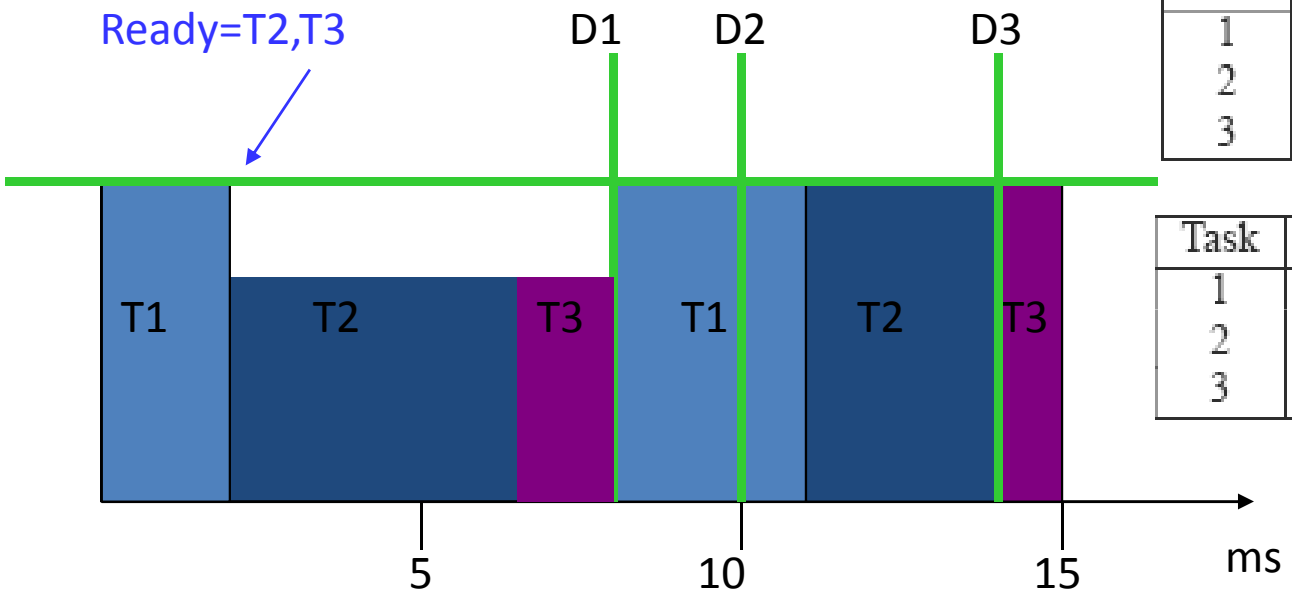
Exemplu



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

- Presupune ca toate procesele trebuie sa termine pana la primul deadline
- Munca totala pana la D1 = 3+3+1 = 7
- Utilizarea pana la D1 = 7/8 = 0.875
- Trebuie sa rulez la 1.0 * fmax

Exemplu

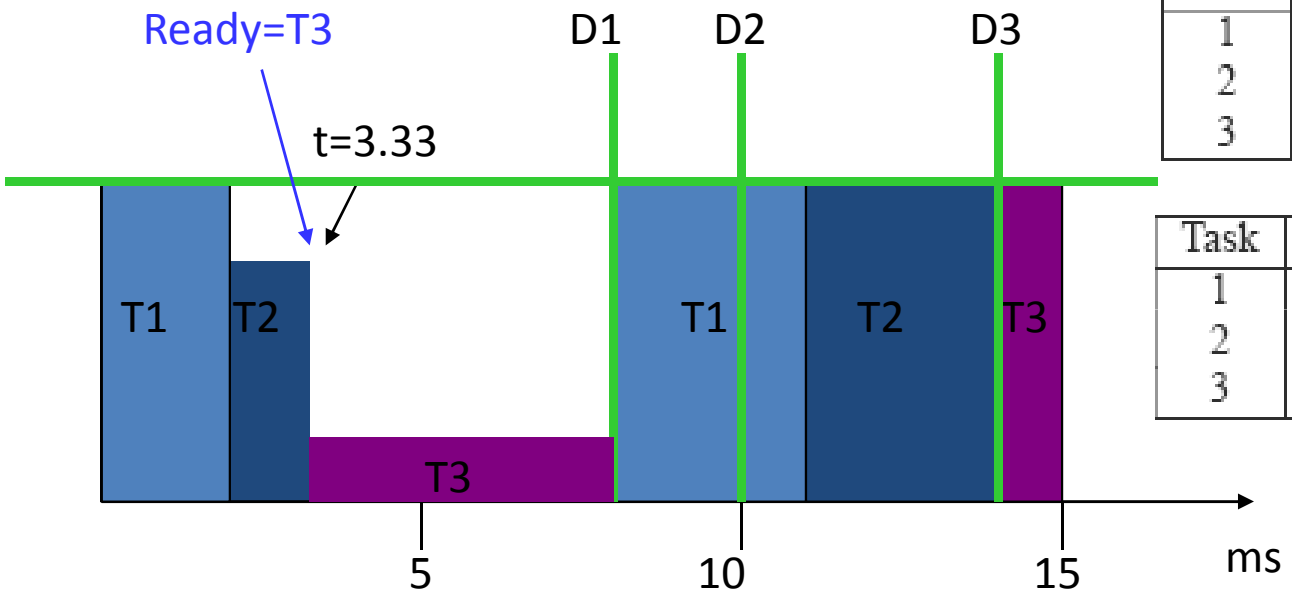


Task	Computing Time	Period
1	3 ms	8 ms
2	3 ms	10 ms
3	1 ms	14 ms

Task	Invocation 1	Invocation 2
1	2 ms	1 ms
2	1 ms	1 ms
3	1 ms	1 ms

- Presupune ca toate procesele trebuie sa termine pana la primul deadline
- Cantitatea de procesare ramasa D1 = 0+3+1 = 4
- Utilizare pana la D1 = 4/(8-2) = 0.667
- Pot sa rulez la 0.75 * fmax

Exemplu

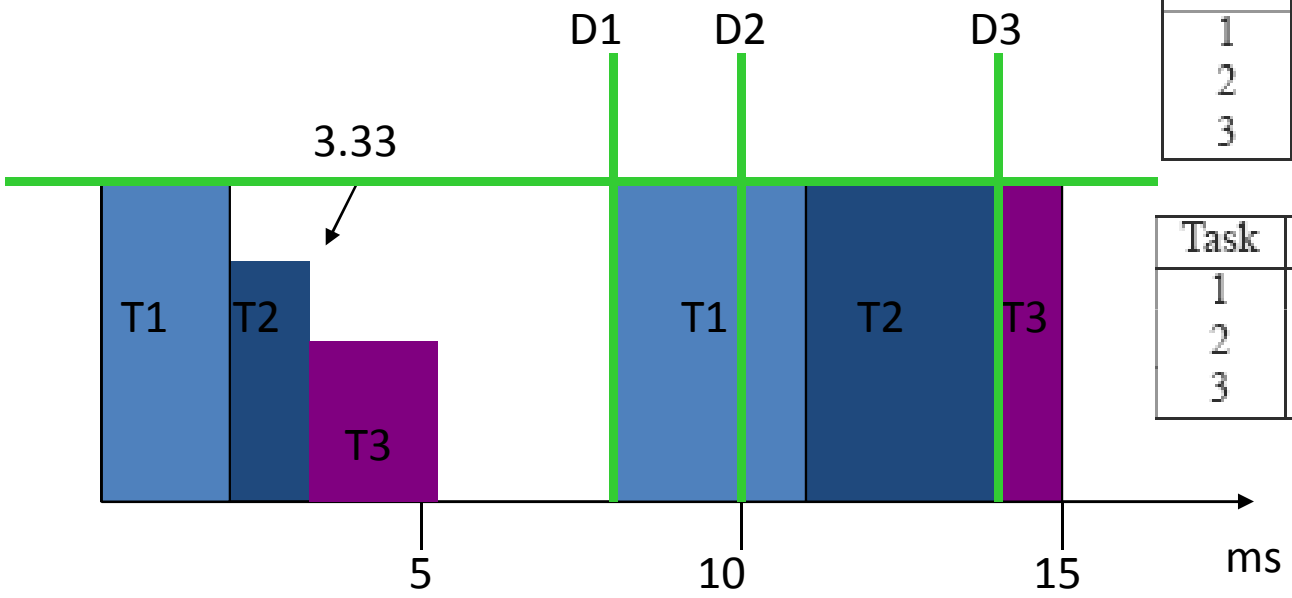


Task	Computing Time	Period
1	3 ms	8 ms
2	3 ms	10 ms
3	1 ms	14 ms

Task	Invocation 1	Invocation 2
1	2 ms	1 ms
2	1 ms	1 ms
3	1 ms	1 ms

- Presupune ca toate procesele trebuie sa termine pana la primul deadline
- Procesare ramasa pana la D1 = 0+0+1 = 1
- Utilizare pana la D1 = 1/(8-3.33) = 0.214
- Pot rula la 0.5

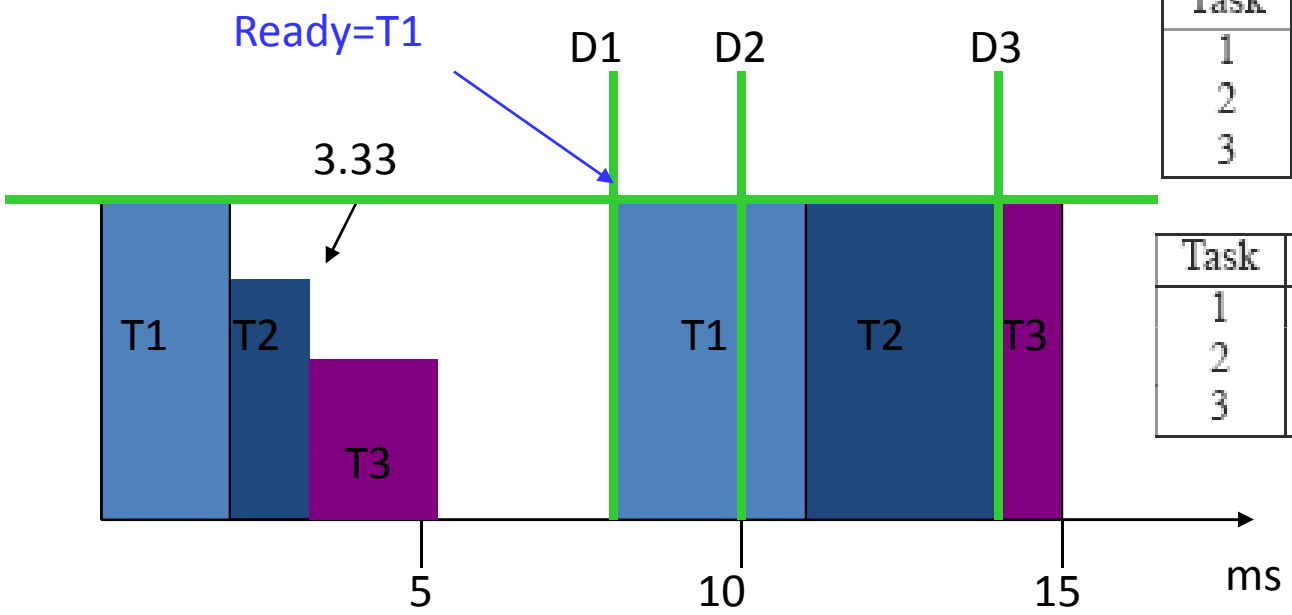
Exemplu



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

Task	Invocation 1	Invocation 2
1	2 ms	1 ms
2	1 ms	1 ms
3	1 ms	1 ms

Exemplu

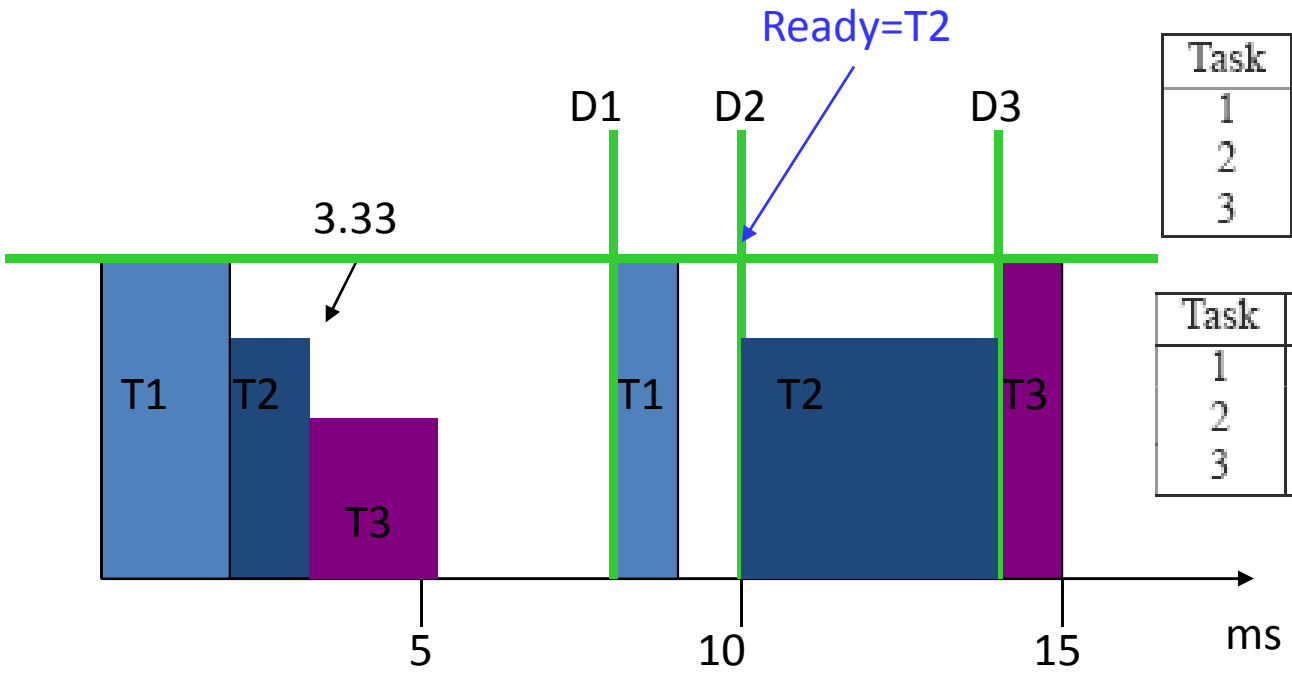


Task	Computing Time	Period
1	3 ms	8 ms
2	3 ms	10 ms
3	1 ms	14 ms

Task	Invocation 1	Invocation 2
1	2 ms	1 ms
2	1 ms	1 ms
3	1 ms	1 ms

- Presupune ca toate procesele trebuie sa termine pana la primul deadline
- Procesare ramasa pana la D2 = 3+0+0 = 3
- Utilizare pana la D2 = 3/(10-8) = 1.5
- Trebuie sa rulez la 1.0

Exemplu

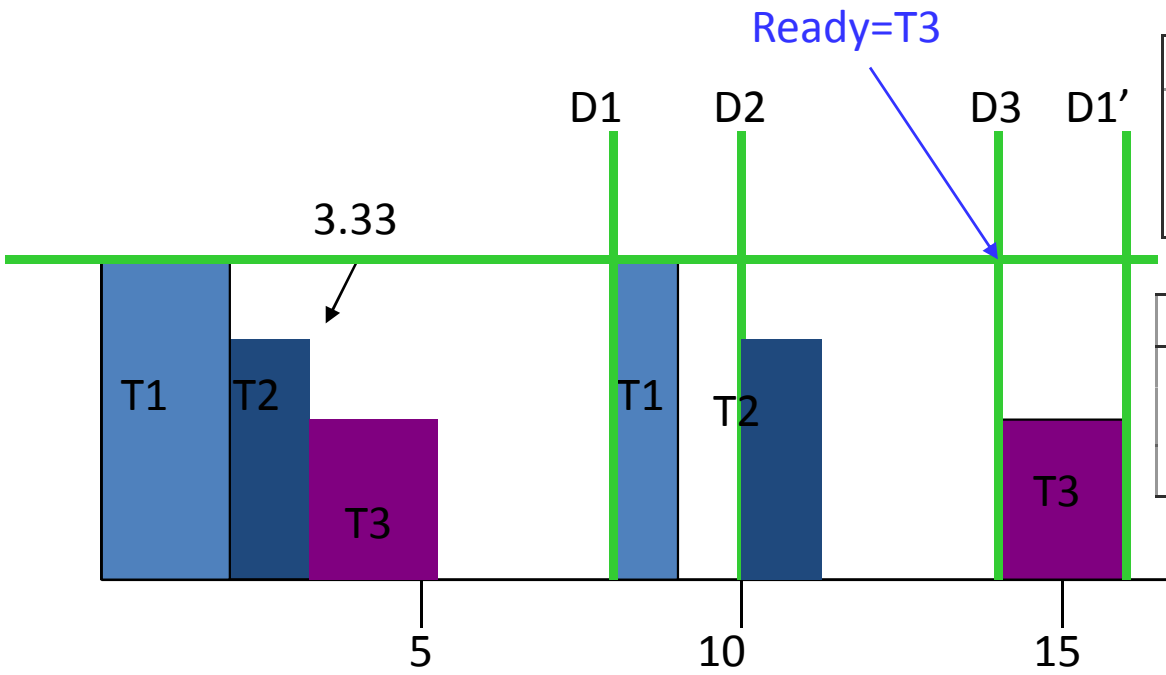


Task	Computing Time	Period
1	3 ms	8 ms
2	3 ms	10 ms
3	1 ms	14 ms

Task	Invocation 1	Invocation 2
1	2 ms	1 ms
2	1 ms	1 ms
3	1 ms	1 ms

- Presupune ca toate procesele trebuie sa termine pana la primul deadline
- Procesare ramasa pana la D3 = 0+3+0 = 3
- Utilizare pana la D3 = 3/(14-10) = 0.75
- Trebuie sa rulez la 0.75

Exemplu



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

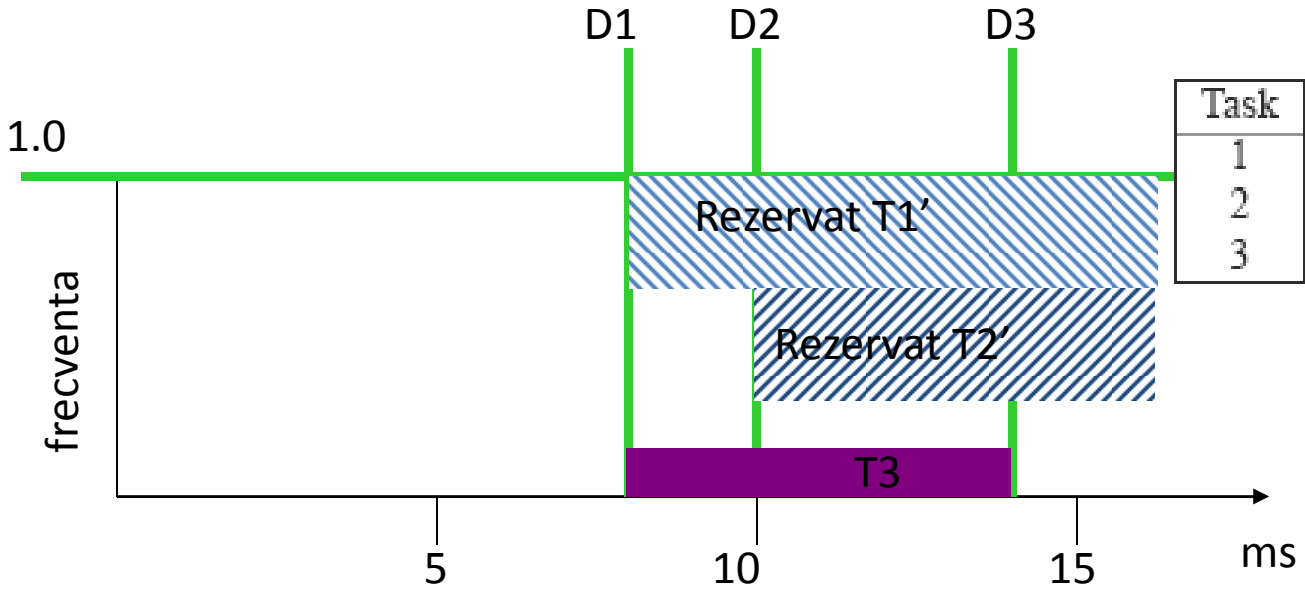
Task	Invocation 1	Invocation 2
1	2 ms	1 ms
2	1 ms	1 ms
3	1 ms	1 ms

- Presupune ca toate procesele trebuie sa termine pana la primul deadline
- Procesare ramasa pana la D1' = 0+0+1 = 1
- Utilizare pana la D1' = 1/(16-14) = 0.5
- Rulez la 0.5

Look-ahead RT-DVS

- Abordarea cu conservare de cicli porneste de la cel mai rau caz posibil si ruleaza la frecventa maxima. Reduce frecventa/voltajul doar cand un task termina mai devreme
- Politica de alocare look-ahead incearca sa amane cat mai mult posibil taskurile si sa seteze frecventa de operare pentru cantitatea de procesare minima care trebuie indeplinita la momentul actual.
- Se uita la task-uri in ordinea inversa EDF
- Chiar daca ruleaza la frecventa minima acum, poate rula la frecventa maxima mai tarziu

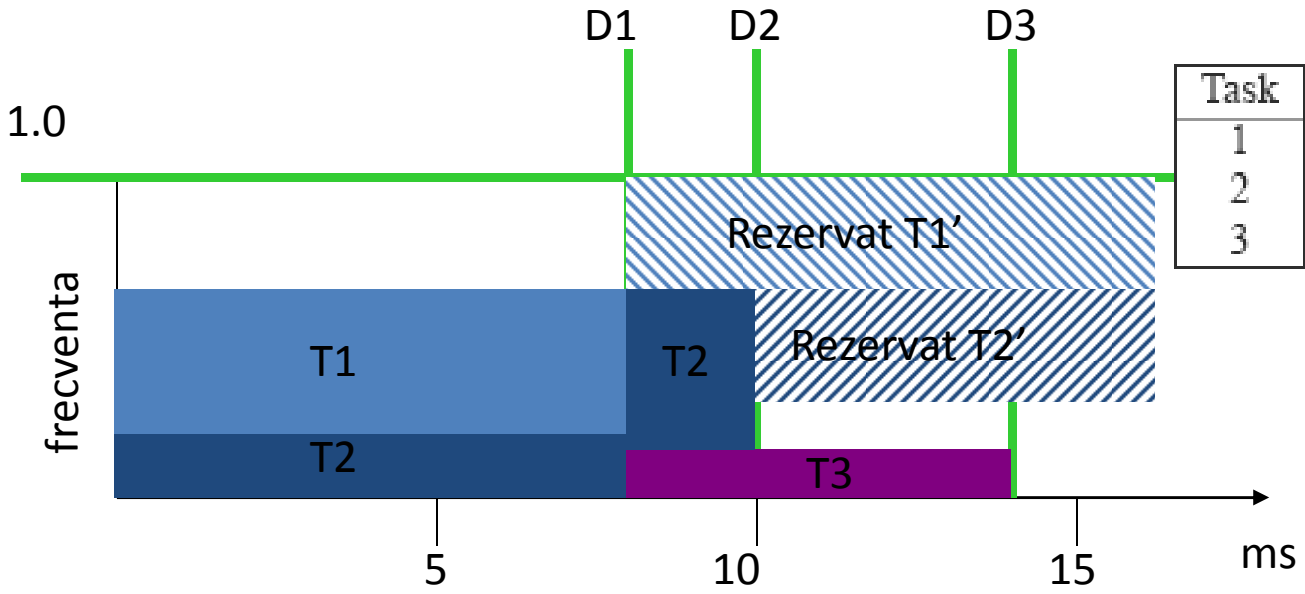
Look-ahead EDF



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

- Care este sarcina minima pana la D1?
- T1, T2 si T3 nu pot fi amanate dupa D1
  - T1 nu poate fi executat dupa D1
  - T3: intra in competitie cu invocarile ulterioare T1', T2'
  - U ramas dupa T1', T2' = 1 – (3/8 + 3/10) = 0.325
  - Timp ramas pentru T3 intre D1-D3 = 0.325*6=1.95
  - T3 poate fi planificat in totalitate intre D1 si D3

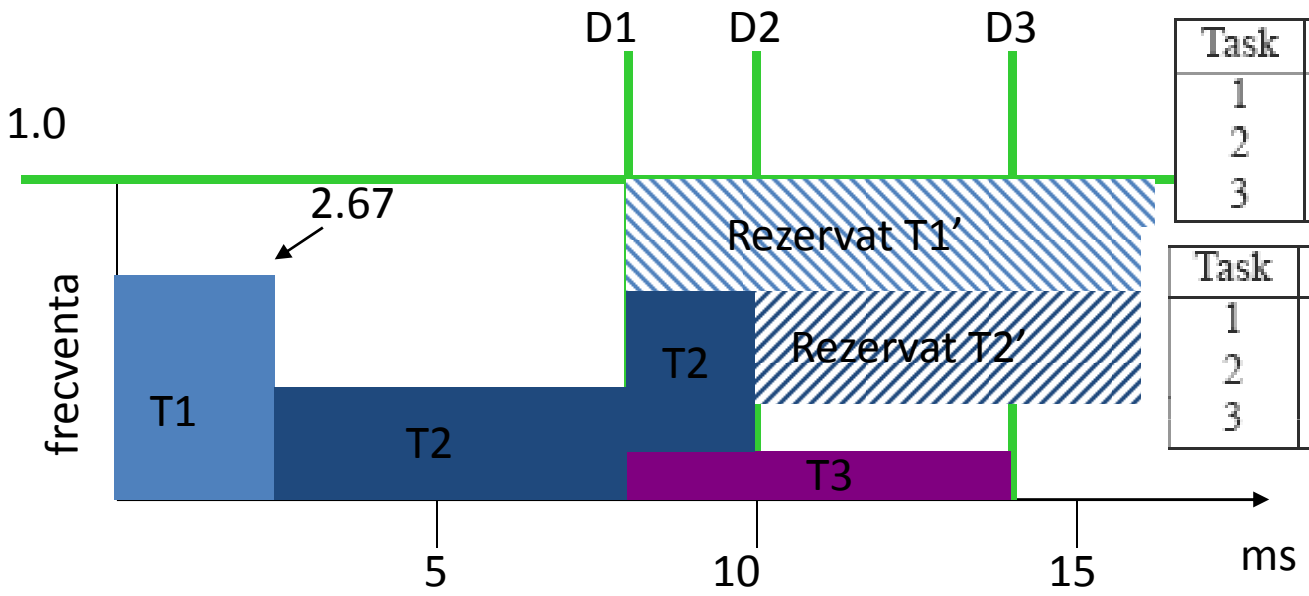
Look-ahead EDF



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

- Trebuie sa planificam T2 intre D1 si D2
- U ramasa dupa T1', T3 = 1 – (3/8 + 1/14) = 0.553
- Timp ramas pentru T2 intre D1-D2 = 0.553*2=1.107
- Timp ramas cu T2 inainte de D1 = 3 – 1.107 = 1.893
- Utilizare inainte de D1 = (1.893 + 3) / 8 =0.611
- T1 trebuie rulat la 0.75

Look-ahead EDF

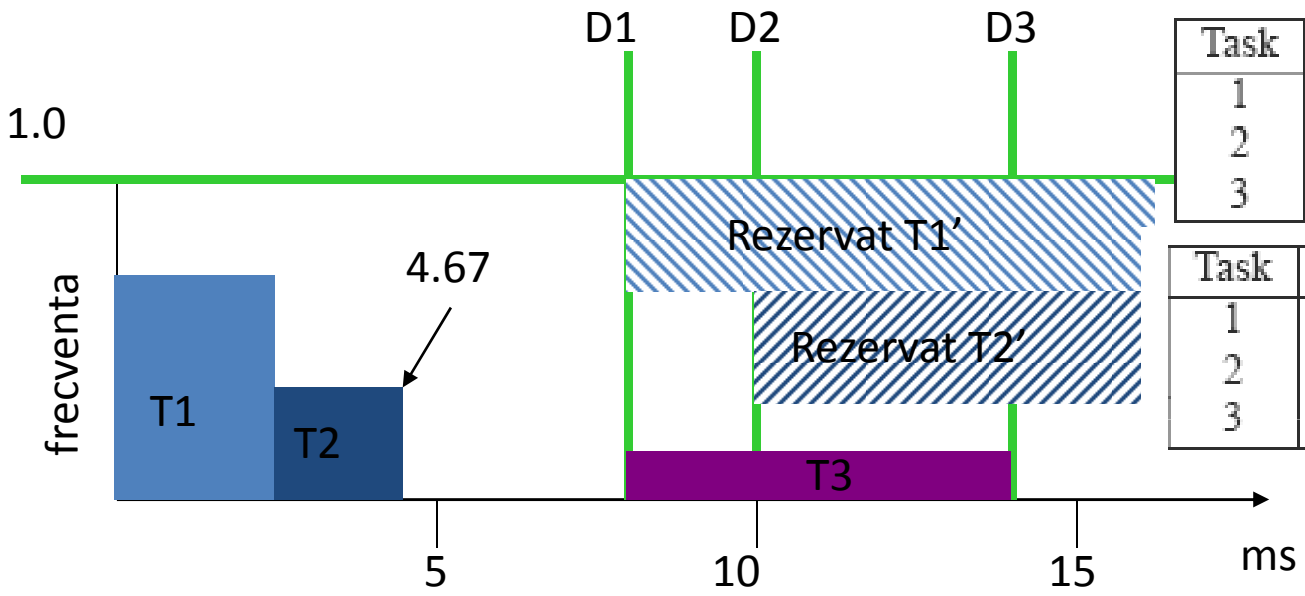


Task	Computing Time	Period
1	3 ms	8 ms
2	3 ms	10 ms
3	1 ms	14 ms

Task	Invocation 1	Invocation 2
1	2 ms	1 ms
2	1 ms	1 ms
3	1 ms	1 ms

- Trebuie sa planificam T2 intre D1 si D2
- U ramas dupa T1', T3 = 1 – (3/8 + 1/14) = 0.553
- Timp ramas pentru T2 intre D1-D2 = 0.553*2=1.107
- Timp ramas cu T2 inainte de D1 = 3 – 1.107 = 1.893
- Utilizare inainte de D1 = (1.893 + 0) / (8 – 2.67) = 0.36
- T2 poate rula la 0.5

Look-ahead EDF

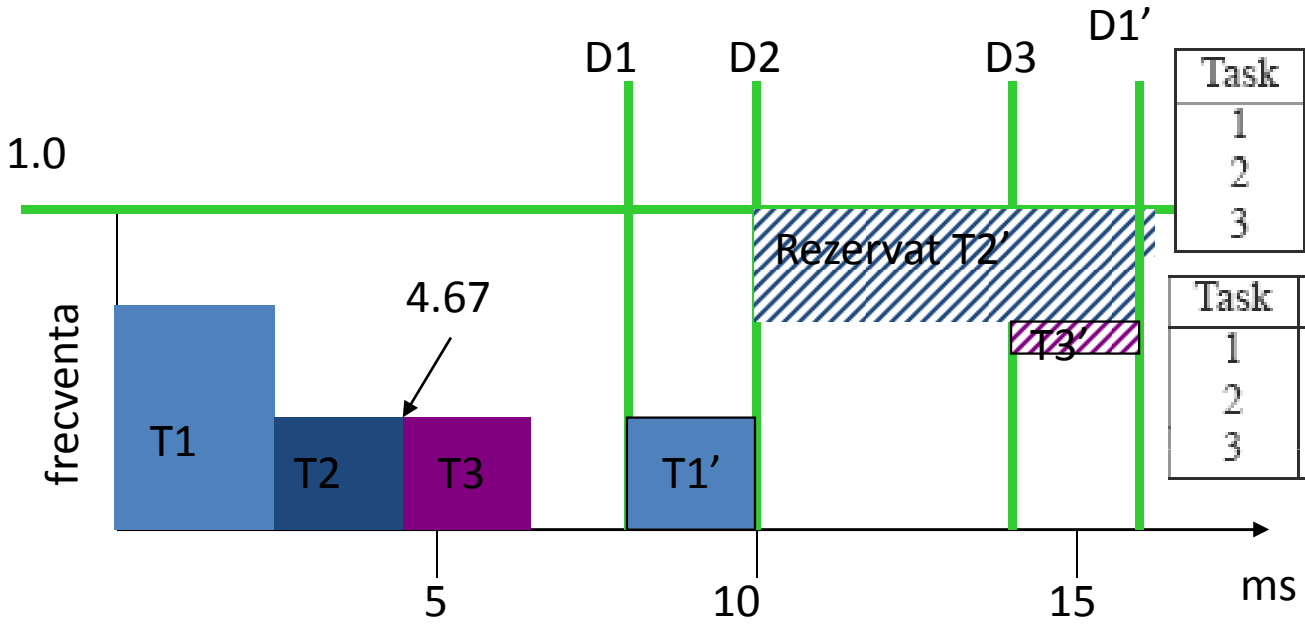


Task	Computing Time	Period
1	3 ms	8 ms
2	3 ms	10 ms
3	1 ms	14 ms

Task	Invocation 1	Invocation 2
1	2 ms	1 ms
2	1 ms	1 ms
3	1 ms	1 ms

- Nu trebuie sa mai rulam nimic pana la D1
- EDF economiseste timpii de lucru
- Ruleaza T3 la frecventa minima

Look-ahead EDF

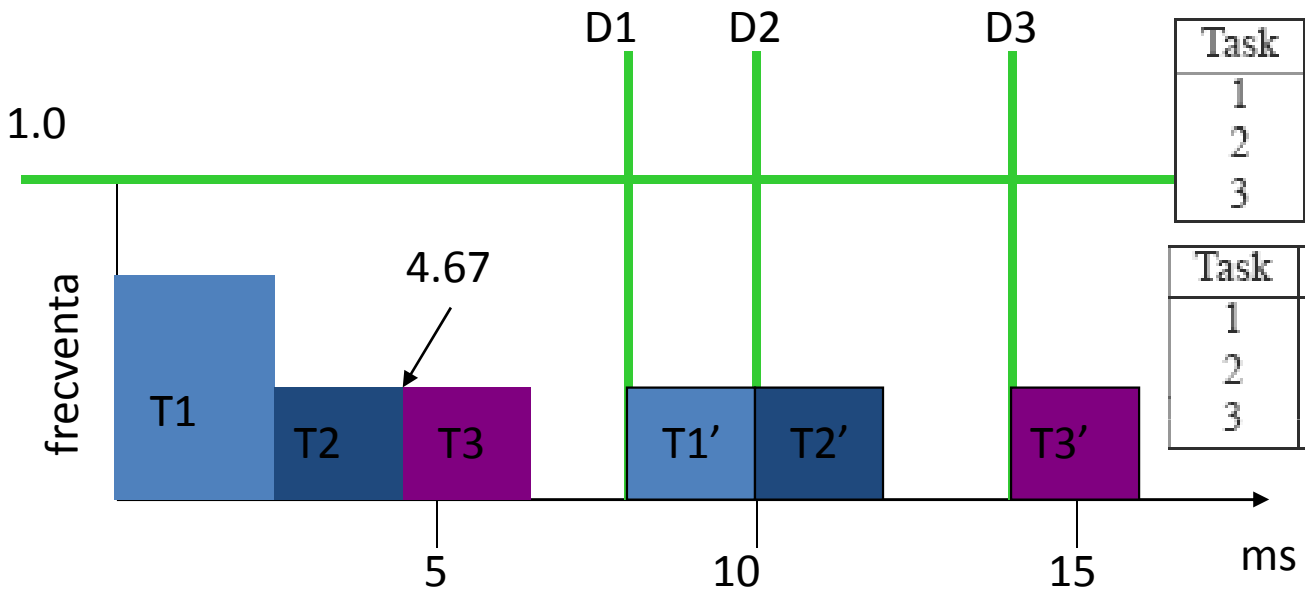


Task	Computing Time	Period
1	3 ms	8 ms
2	3 ms	10 ms
3	1 ms	14 ms

Task	Invocation 1	Invocation 2
1	2 ms	1 ms
2	1 ms	1 ms
3	1 ms	1 ms

- U ramas dupa T2’ si T3’ = 1 – (3/10 + 1/14) = 0.63
- Timp ramas pentru T1’ intre D2-D1’ = 0.63*6 = 3.77
- T1’ poate fi amanat complet pana dupa D2
- Pentru a optimiza, ruleaza T1’ la frecventa minima

Look-ahead EDF



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

Task	Invocation 1	Invocation 2
1	2 ms	1 ms
2	1 ms	1 ms
3	1 ms	1 ms

Consumul de energie

- Pentru exemplul dat

Algoritm RT-DVS	Energie
Nici unul(EDF simplu)	1.0
RMS static	1.0
EDF static	0.64
Cycle-conserving EDF	0.52
Cycle-conserving RMS	0.71
Look-ahead EDF	0.44