Embedded Systems Laboratory

Catedra de Calculatoare
Facultatea de Automatica si Calculatoare

Sisteme Încorporate

Cursul 6

Software Power Management

Facultatea de Automatică și Calculatoare
Universitatea Politehnica București

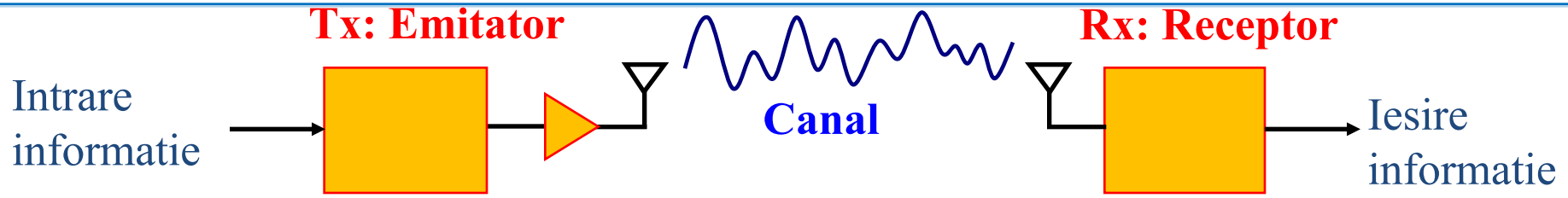
Recapitulare: Power Management

- Unde se poate face PM: H/W, Firmware, OS, Aplicatie, Utilizator
- Hardware si firmware nu pot aprecia starea globala a sistemului
- Utilizatorii nu stiu intotdeauna caracteristicile sistemului pe care lucreaza sau nu pot lua decizii
- Aplicatiile opereaza independent si sistemul de operare nu pune la dispozitia lor suficiente informatii
- PM la nivel de SO pare a fi cea mai buna solutie, dar
  - Trebuie sa puna la dispozitia aplicatiei starea sistemului la momentul respectiv pentru ca ele sa se poata adapta

Power Management pentru Wireless NIC

- Moduri de lucru: transmit, receive, idle, sleep, off
  - In general idle mode (nu se transmite si nu se receptioneaza) consuma aproape la fel de mult cat modul receive
  - Puterea de emisie in WLAN este 2x-3x mai mare decat puterea de receptie
    - Diferenta este d.p. cu distanta de emisie (puterea RF domina)
    - Puterea de emisie poate fi variata (dezavantaj: distanta efectiva de emisie)
  - Timpii de tranzitie intre stari sunt semnificativi
    - Transceiver IR: 10 μs sa intre in sleep mode, si 40 μs la wake-up
    - Placi “clasice” de retea: timpi de wake-up intre 100ms si 5s

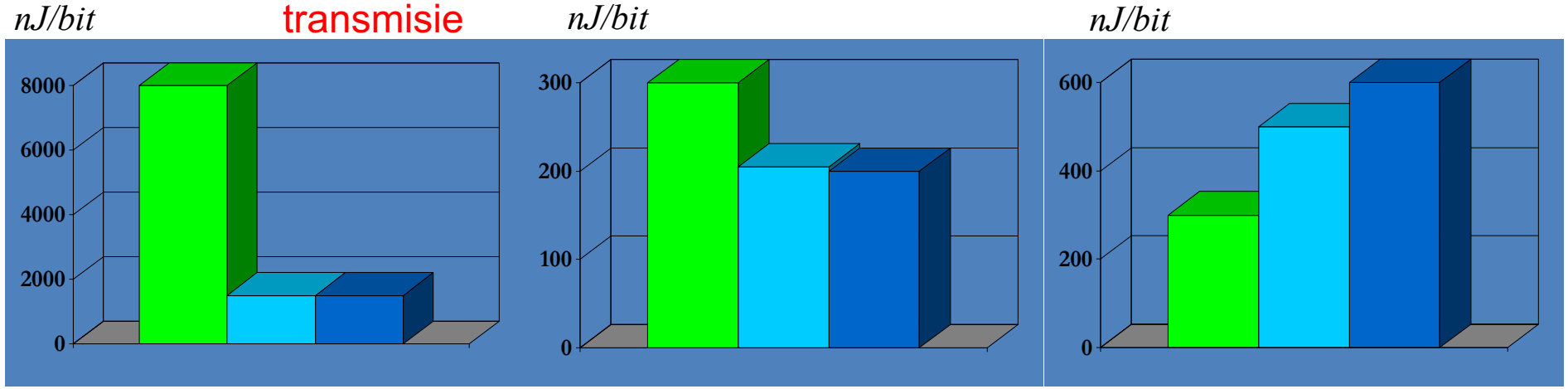
Consumul de energie radio



E_{elec}^{Tx} — Electronica de transmisie

E_{RF} — Amplificare

E_{elec}^{Rx} — Electronica de receptie

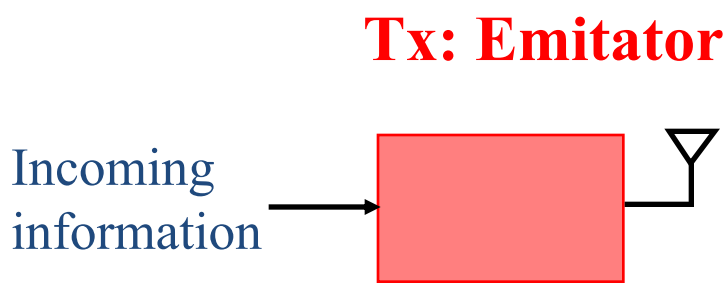


E_{RF} E_{elec}^{Tx} E_{elec}^{Rx} — ~ 1 km (GSM)

E_{RF} E_{elec}^{Tx} E_{elec}^{Rx} — ~ 50 m (WLAN)

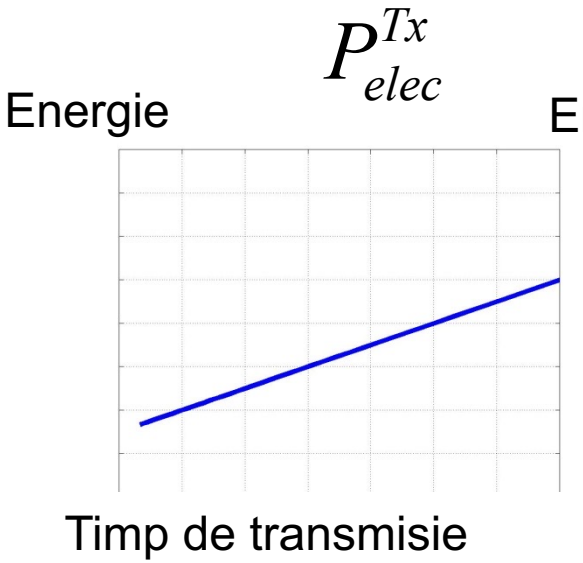
E_{RF} E_{elec}^{Tx} E_{elec}^{Rx} — ~ 10 m (Mote)

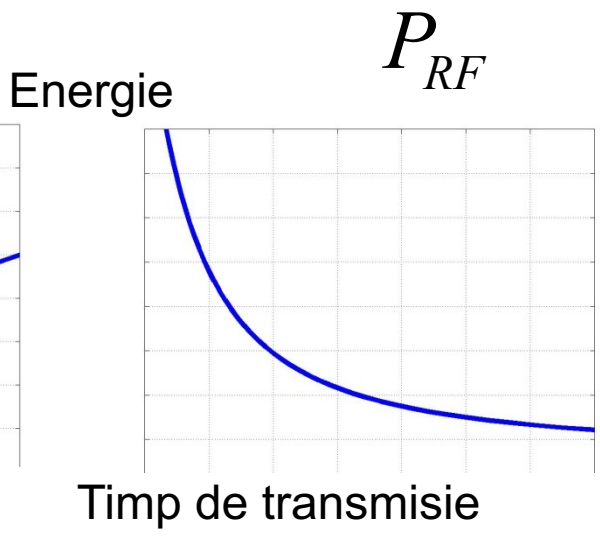
Consumul de energie la Emitator

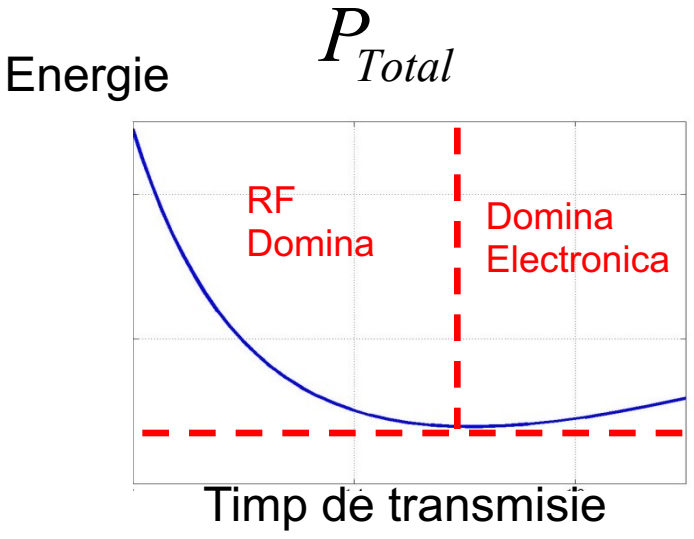


- Parametru de interes:
  - Consumul de energie pe bit transmis

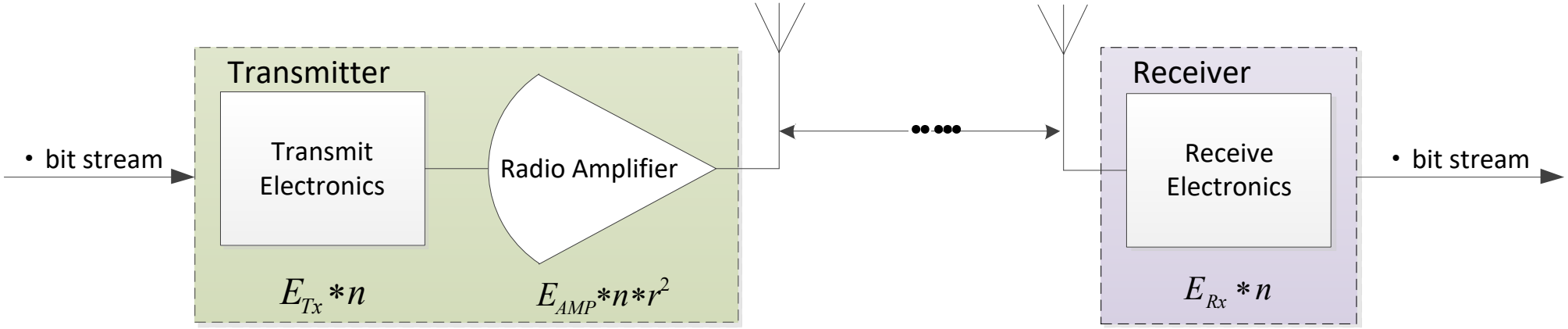
$$E_{bit} = \frac{P}{T_{bit}}$$







Model Transmisie Radio



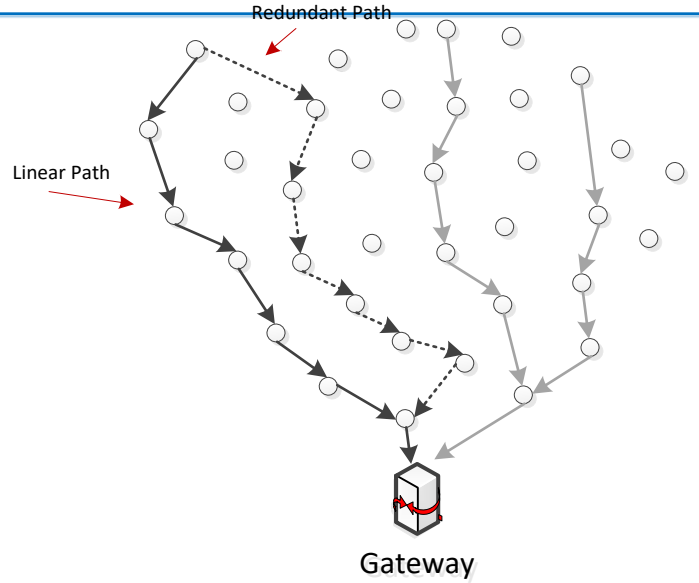
$$E_{Tx}(n,r) = E_{tc}(n) + E_{amp}(n,r) = n \cdot E_{trans.} + n \cdot \varepsilon_{amp} \cdot r^{\gamma}$$

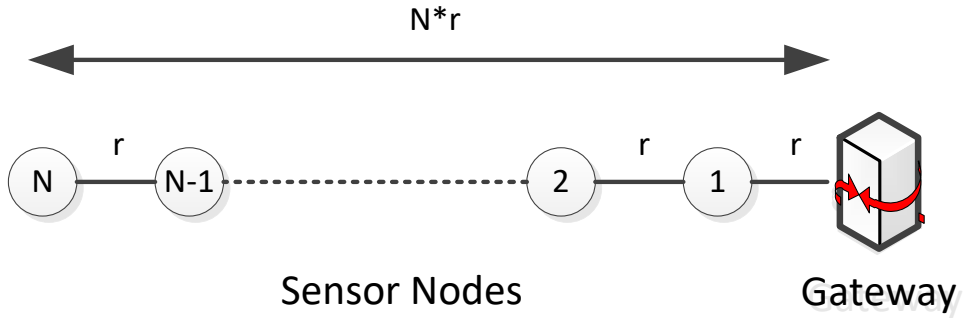
$$\varepsilon_{amp} = \frac{\frac{S}{N_r} \cdot NF_{Rx} \cdot N_0 \cdot BW \cdot \left(\frac{4\pi}{\lambda}\right)^{\gamma}}{G_{ant} \cdot \eta_{amp} \cdot R_{bit}}$$

$$E_{Rx}(n) = E_{rc}(n) = n \cdot E_{recv.}$$

$$PL = P_{Tx[dBm]} - P_{Rx[dBm]} = PL_0 + 10\gamma log_{10}\frac{d}{d_0} + X_g$$

Rețele Wireless Liniare





$$E_b(N,n,r) = E_{Tx}(n,N\cdot r) = n\cdot E_{trans.} + n\cdot \varepsilon_{amp}\cdot (N\cdot r)^{\gamma}$$

$$E_{MH}(N,n,r) = N\cdot E_{Tx}(n,r) + (N-1)\cdot E_{Rx}(n) = N\cdot n\cdot (E_{trans.} + \varepsilon_{amp}\cdot r^{\gamma}) + (N-1)\cdot n\cdot E_{recv.}$$
$$= n\cdot \left(N\cdot \left(E_{trans.} + E_{recv.} + \varepsilon_{amp}\cdot r^{\gamma}\right) - E_{recv.}\right)$$

$$E_b^{all}(n,r) = \sum_{i=1}^{N} E_{Tx}(n,i\cdot r) = n\cdot N\cdot E_{trans.} + n\cdot \varepsilon_{amp}\cdot r^{\gamma}\cdot \sum_{i=1}^{N} i^{\gamma}$$

$$E_{MH}^{all}(n,r) = \sum_{i=1} E_{MH}(i,n,r) = N\cdot E_{Tx}(n,r) + (N-1)\cdot E_{Rx}(n)$$

$$= \frac{N\cdot (N+1)}{2}\cdot n\cdot (E_{trans.} + \varepsilon_{amp}\cdot r^{\gamma}) + \frac{N\cdot (N-1)}{2}\cdot n\cdot E_{recv.}$$

- Când e mai bine să fac multi-hop?

$$E_b^{all}(n,r) > E_{MH}^{all}(n,r)$$

- Ipoteză simplificatoare:

$$E_{trans.} = E_{recv.} = E_{circ.}$$

$\gamma = 2$

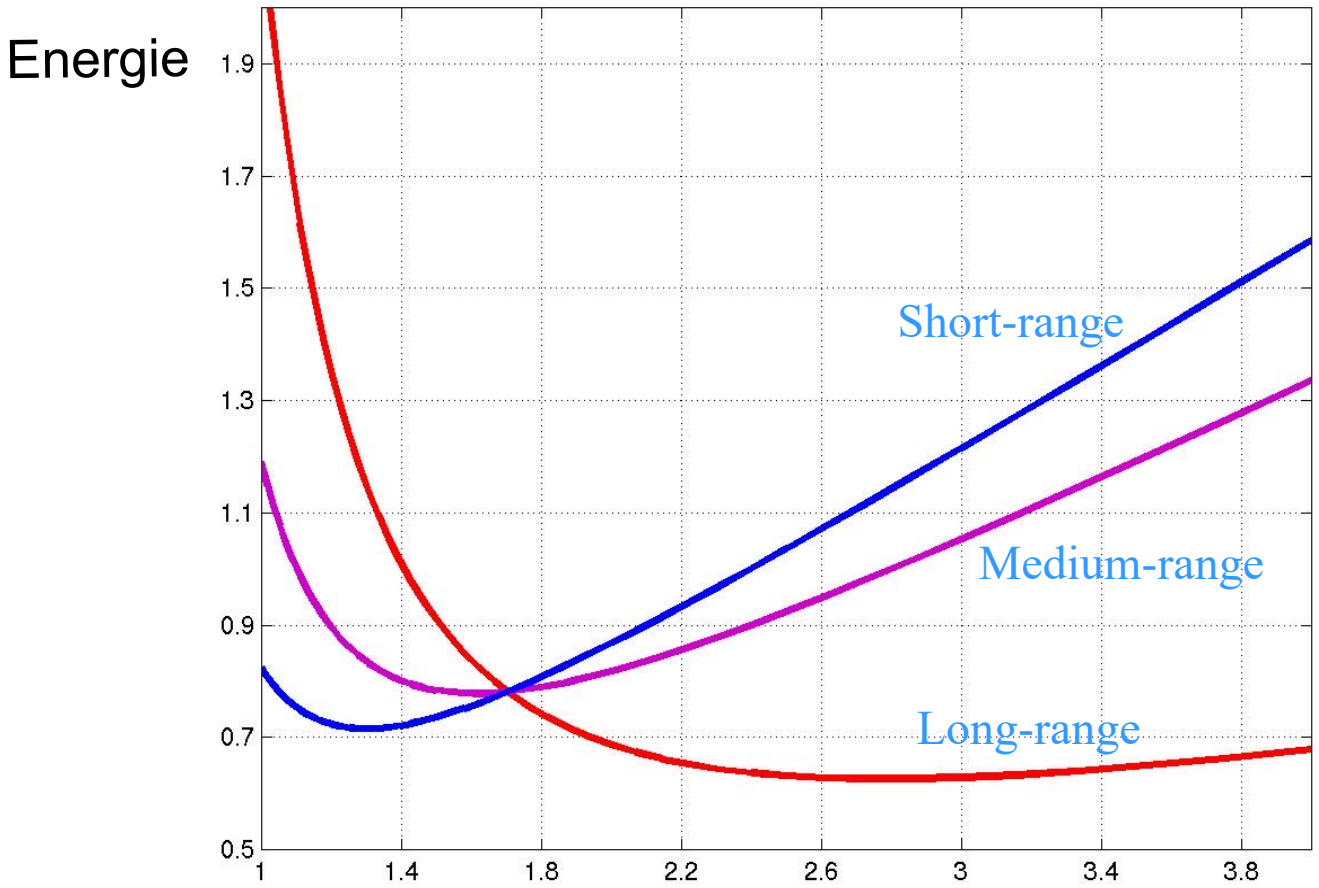
$$nNE_{circ.} + n\varepsilon_{amp}r^2\sum_{i=1}^{N} i^2 > \frac{N\cdot(N+1)}{2}n(E_{circ.} + \varepsilon_{amp}r^2) + \frac{N\cdot(N-1)}{2}nE_{circ.}$$

$$\frac{E_{circ.}}{\varepsilon_{amp}} < \frac{N+1}{3}r^2$$

$\gamma = 4$

$$\frac{E_{circ.}}{\varepsilon_{amp}} < \frac{(N+1)(6N^2+15N+16)}{30}r^4$$

Efectele Distantei de Transmisie



Strategii PM pentru Wireless NIC

- Strategiile similare celor pentru hard-disk si CPU
  - Tranzitiile sleep ↔ wakeup << ca la discuri
  - Pot fi implementate la nivel de protocol MAC e.g. 802.11
- Reducerea incarcarii NIC
  - Compresia de date
  - Oprirea emisiei in cazul unei rate de eroare mare
- Reducerea puterii de transmisie
  - Afecteaza direct BER (Bit Error Rate)

Power Management in Transceiverul Radio

- Circuitele digitale pot fi oprite/pornite usor
- Managementul unui transceiver RF nu e facil
- Oprirea unor circuite ale transceiverului (buffere, etaje de amplificare, oscilator) duce la probleme
  - Stabilitatea transmisiei
  - Introduce timpi de asteptare (PLL settle etc.)

802.11 Power Management

NIC poate fi activa tot timpul (CAM)

- Consum foarte mare (~1.5 Watt)
- Poate sa injumatateasca timpul de viata al unui handheld

Alternativ, poate folosi power-save mode (PSM)

- Daca nu exista pachete receptionare, interfata intra in sleep
- Wake-up periodic (beacon de 100 ms)
- Reduce consumul cu 70-80%

Mecanisme de Suport pentru Power Management

- PM bazat pe hardware si firmware este problematic
  - Nu tine cont de aplicatiile care ruleaza
    - » Ex. Ecranul se stinge in timpul unei prezentari
- Solutia: Incorporeaza informatiile despre aplicatii
  - Ia decizii de power management bazate pe datele stranse de aplicatie despre utilizator si modul in care acesta foloseste sistemul
- SO este locul potrivit pentru PM, luarea de decizii si coordonarea resurselor
  - Ex.: Arhitectura Microsoft OnNow si extensiile API pentru Windows98, Windows 2000

Microsoft OnNow

- Extensia a Windows API care permite aplicatiilor sa
  - Ia decizii de power management
  - Se adapteaza la un power-state curent
    - Afla daca sistemul ruleaza pe baterie ca sa reduca procesarea
    - Descopera starea hard-discului si amana operatiile I/O de prioritate redusa
- Necesita schimbari in hardware, BIOS, OS si softul de aplicatie
  - Standarde noi de magistrala si device h/w management
  - Interfata standard intre OS si hardware
    - ACPI (Intel & Toshiba)

Advanced Configuration and Power Interface (ACPI)

- Standard deschis pentru device configuration si power management (HP, Intel, Microsoft, Toshiba, Dell 1996)
  - Defineste interfete independente de platforma pentru descoperirea componentelor hardware, configurare, power management si monitorizare.
  - Foloseste tabele de descriptori
  - Continutul unei tabele pentru fiecare device:
    - Cerinte si capabilitati power management
    - Metode pentru a seta si a obtine power-state-ul curent
    - Setari hardware
    - Metode pentru setarea resurselor hardware

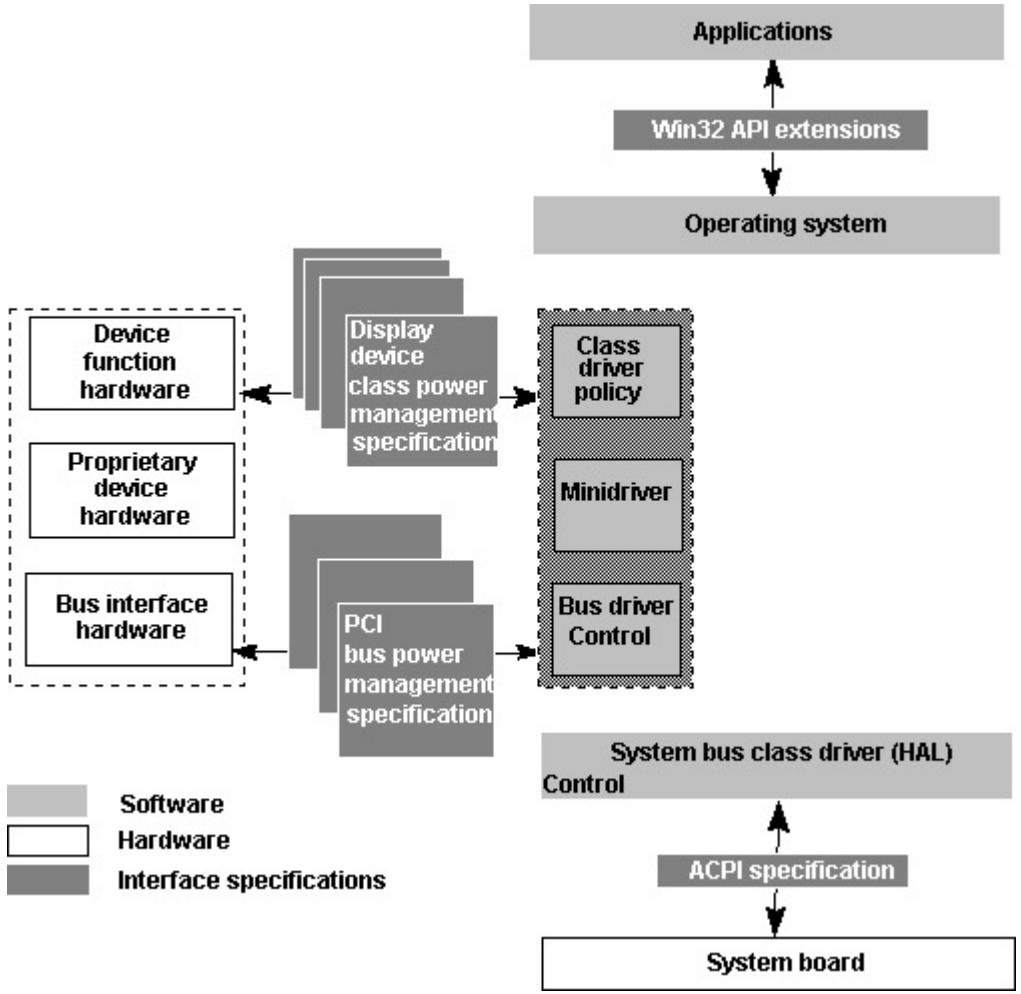
ACPI Power States

- **G0** (**S0**) *Working*
- **G1** *Sleeping*
  - **S1**: Toate memoriile cache sunt golite si CPU nu mai executa nici o instructiune. CPU si memoria RAM sunt alimentate; device-urile care nu indica explicit ca trebuie sa ramana operationale sunt oprite.
  - **S2**: CPU este oprit
  - **S3**: Cunoscuta sub numele de *Standby* sau *Sleep*. Mem. RAM este alimentata.
  - **S4**: *Hibernare*. Intregul continut al memoriei volatile este salvat in memoria ne-volatila (hard-disk) si sistemul este oprit.
- **G2** *Soft Off.* G2 este aproape acelasi lucru cu G3 *Mechanical Off*dar unele componente raman activate a.i. sistemul sa poata sa fie “trezit” de la tastatura, ceas intern, modem, LAN sau USB.
- **G3** *Mechanical Off*: Consumul de energie al sistemului ajunge aproape la zero (de obicei mai ramane alimentat doar ceasul de timp real din bateria proprie.)

Device & Processor states

- **Device states**
  - D0 *Fully-On* – starea normala de operare.
  - D1 si D2 sunt stari intermediare care variaza in functie de device.
  - D3 *Off* – dispozitivul nu raspunde la comenzi pe bus.
- **Processor states**
  - C0 – starea normala de operare.
  - C1 (*Halt*) este o stare in care procesorul nu executa instructiuni dar poate sa se intoarca la C0 aproape instantaneu
  - C2 (*Stop-Clock*) procesorul nu primeste semnal de ceas (revenirea la C0 dureaza mai mult).
  - C3 (*Sleep*) procesorul nu mentine coerenta memoriei cache. Unele procesoare au variatii ale acestei stari (Deep Sleep, Deeper Sleep, etc.) care difera prin timpul de wake-up.

Componentele OnNow

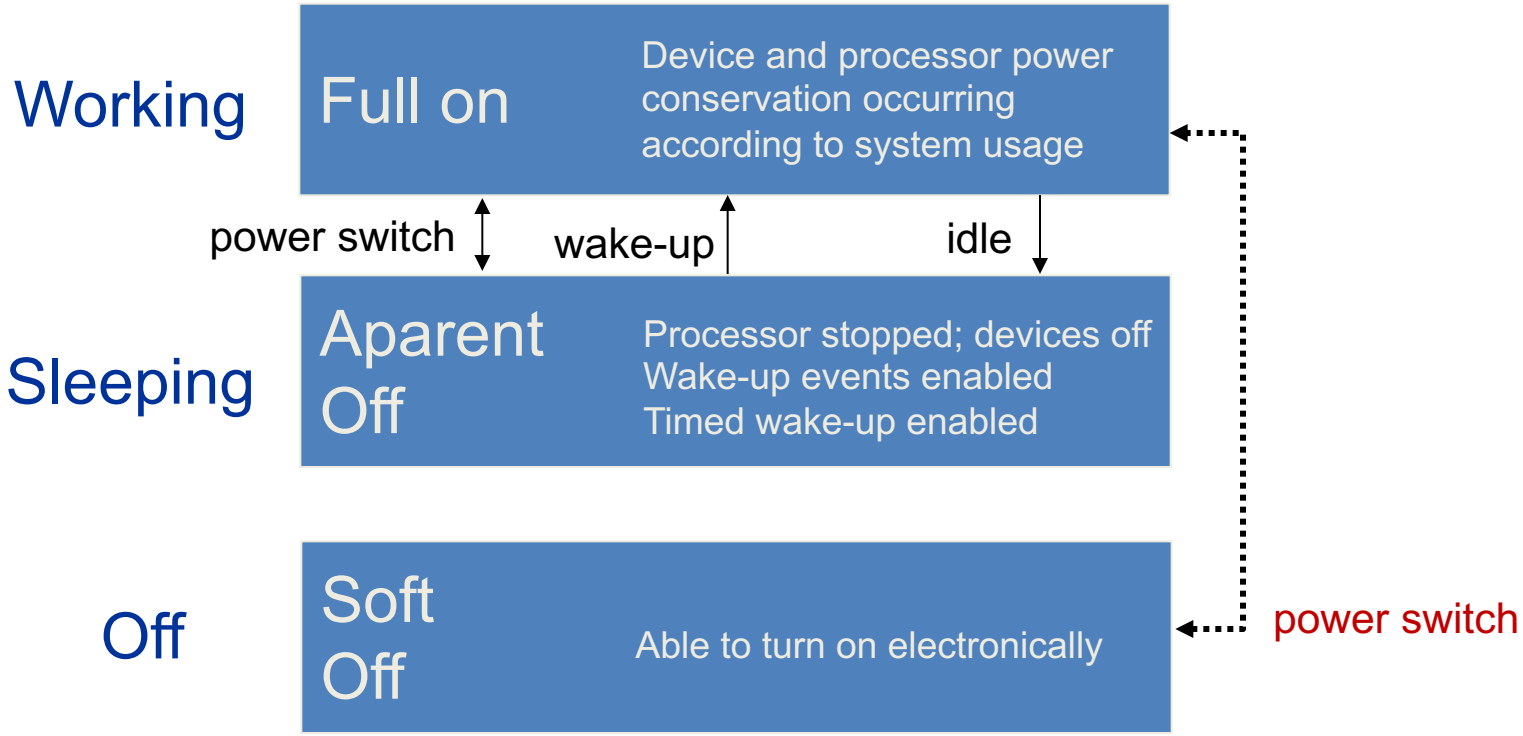


http://www.microsoft.com/whdc/archive/OnNowApp.mspx

Arhitectura OnNow

- Punctul de vedere al utilizatorului: sistemul este pornit/oprit
- In realitate sistemul trece prin mai multe stari in functie de politica sistemului de operare
- Stari globale
  - working: aplicatiile se executa normal
  - sleep: aplicatiile sunt oprite
    - SO monitorizeaza actiunile utilizatorului si aplicatiile pentru a determina intrarea-iesirea din aceasta stare
  - off: sistemul este oprit in totalitate

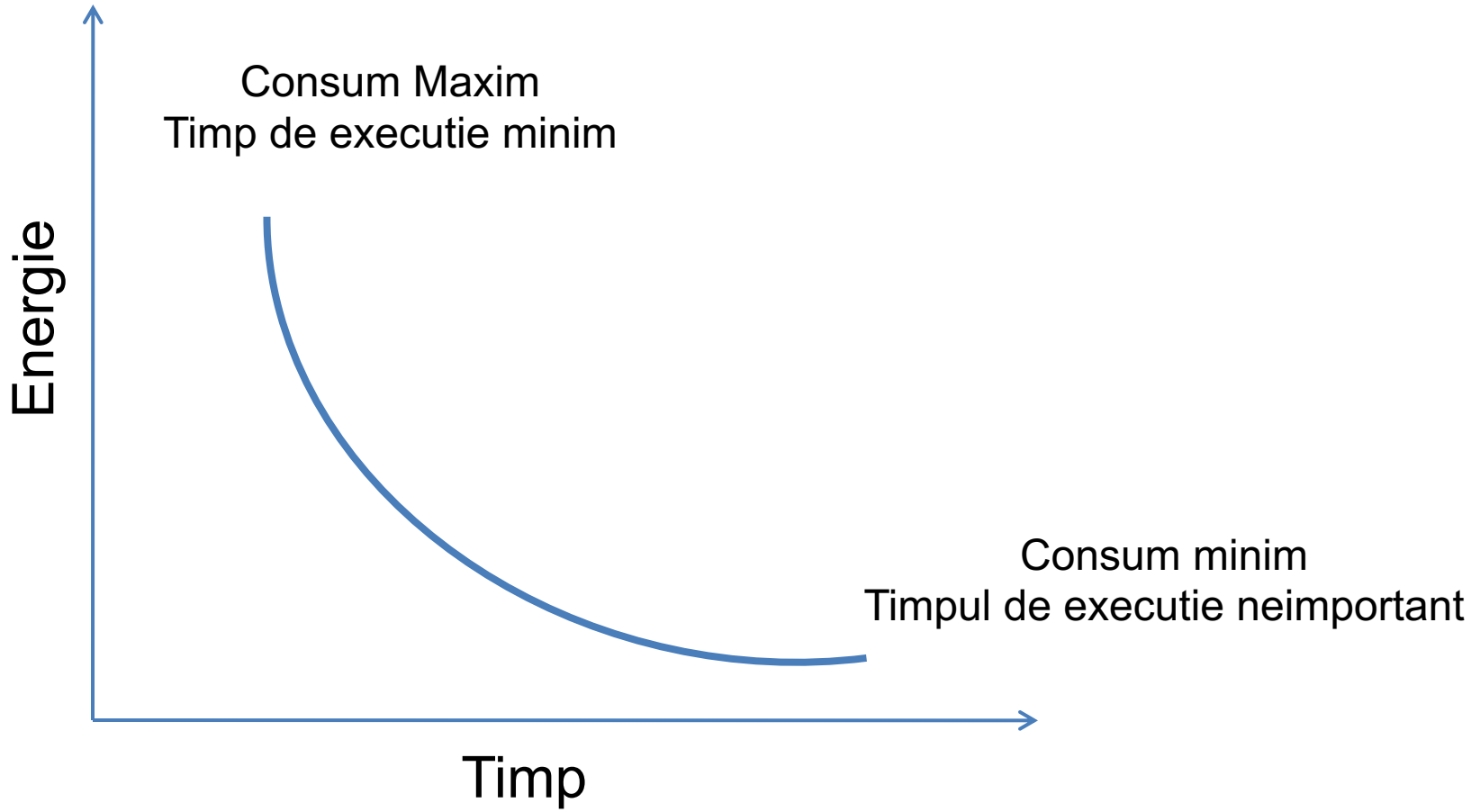
Arhitectura OnNow



Stari Globale OnNow

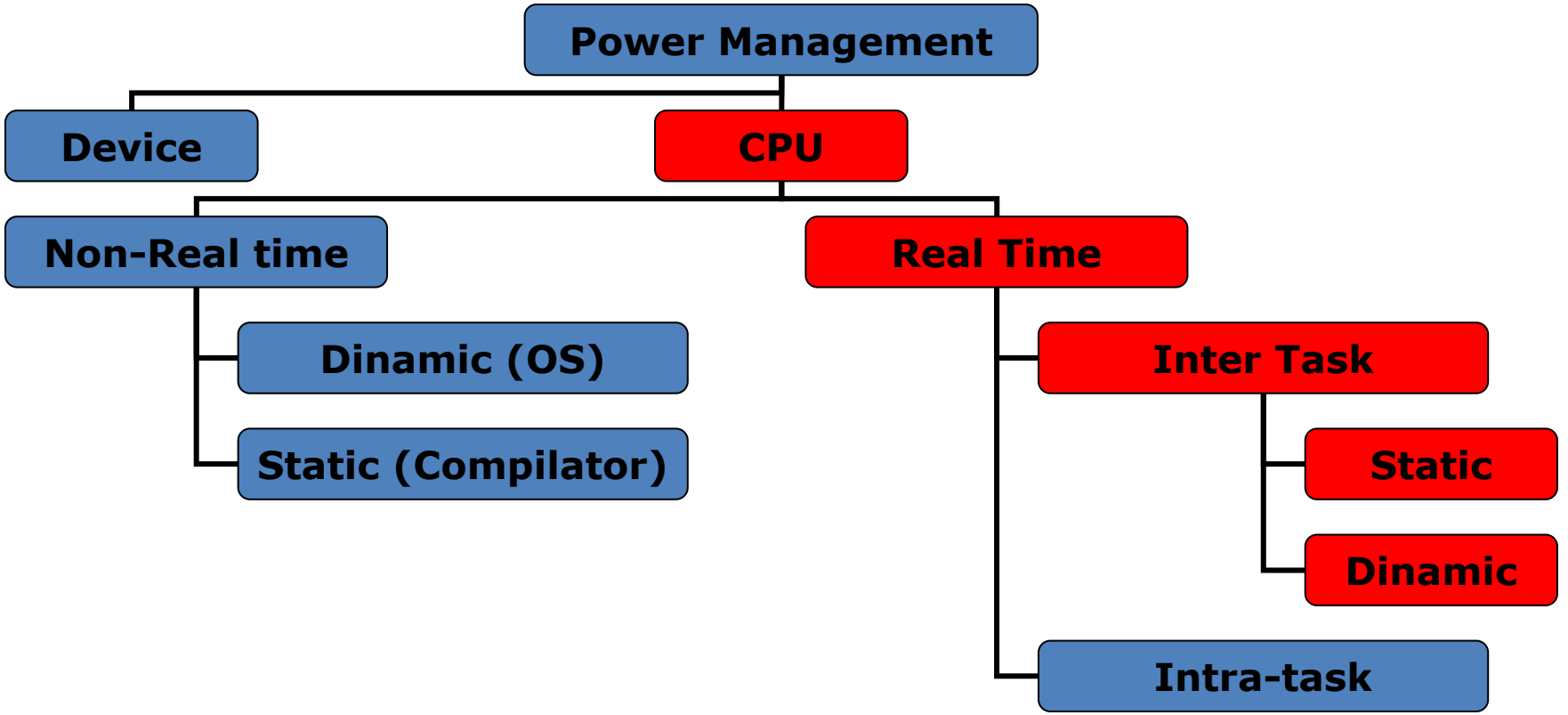
http://www.microsoft.com/whdc/archive/OnNowApp.mspx

Concluzie: Compromis intre consum si performanta



http://dilbert.com/strips/comic/1990-07-16/

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

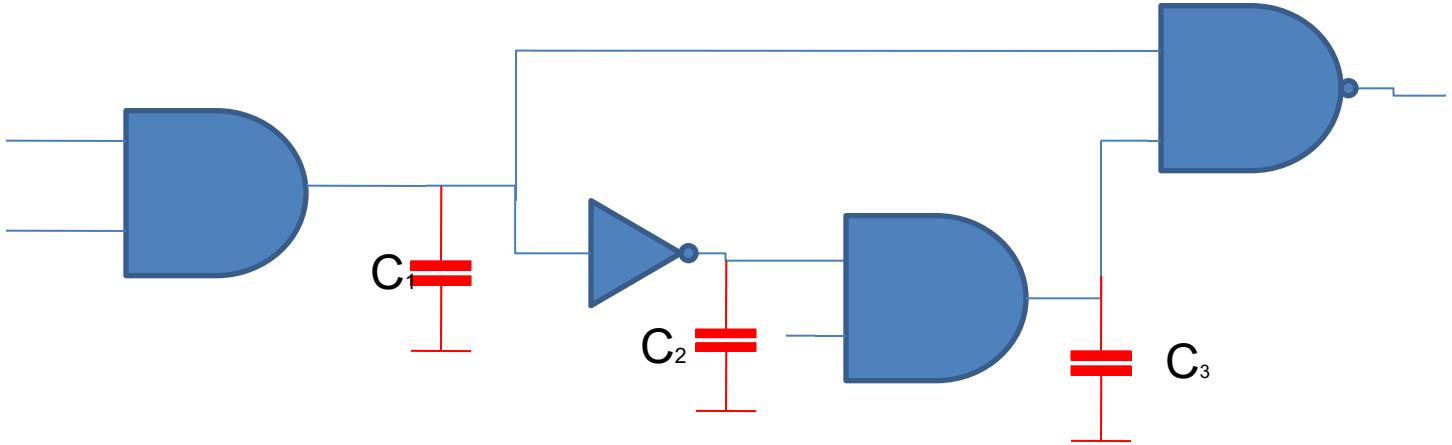
- Memorie
  - Partajeaza memoria in mai multe circuite
  - Deconecteaz-o atunci cand n-o folosesti
- Periferice
  - PM pe mai multe niveluri
  - Deconectare atunci cand nu sunt folosite

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

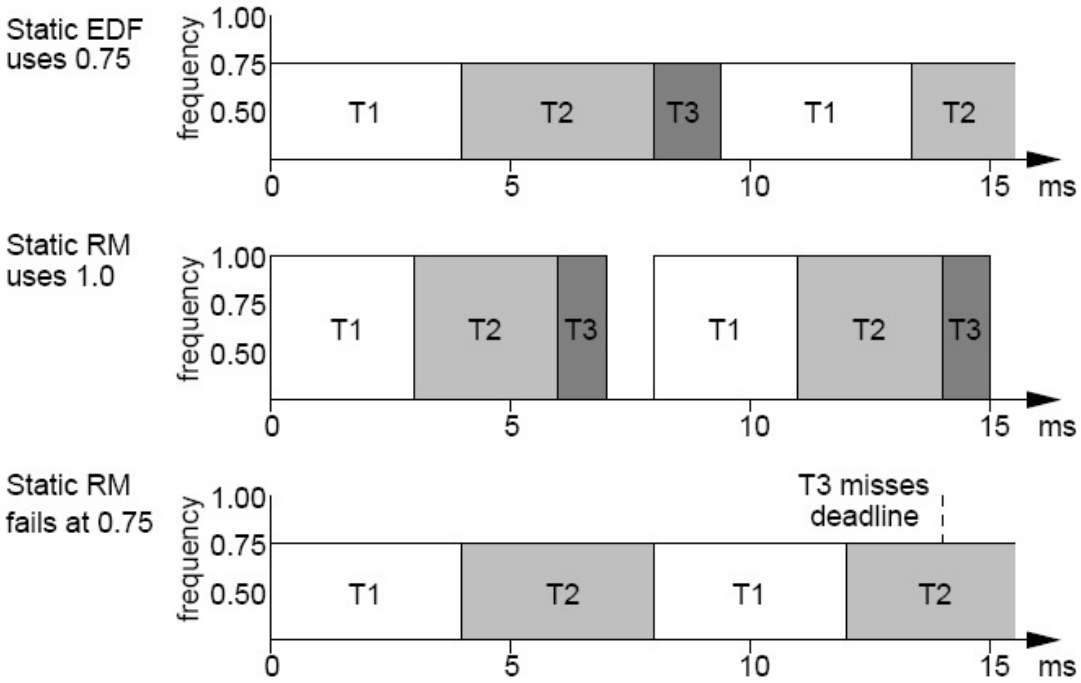
such that RM_test(f_i/f_m) or EDF_test(f_i/f_m) is true.

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

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

Cycle-conserving EDF

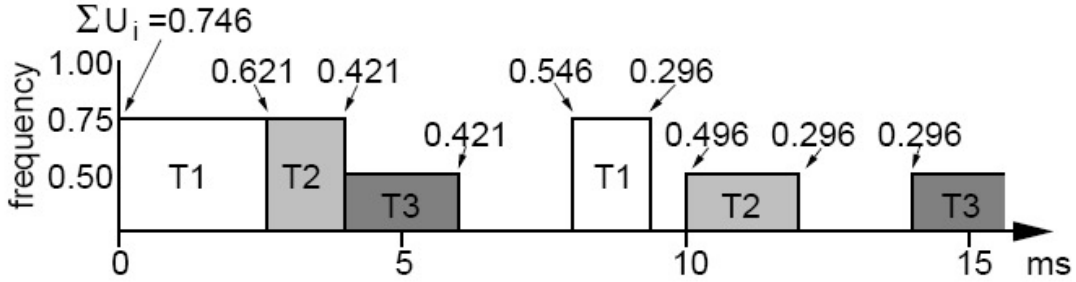
- Exemplu algoritm EDF:

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.

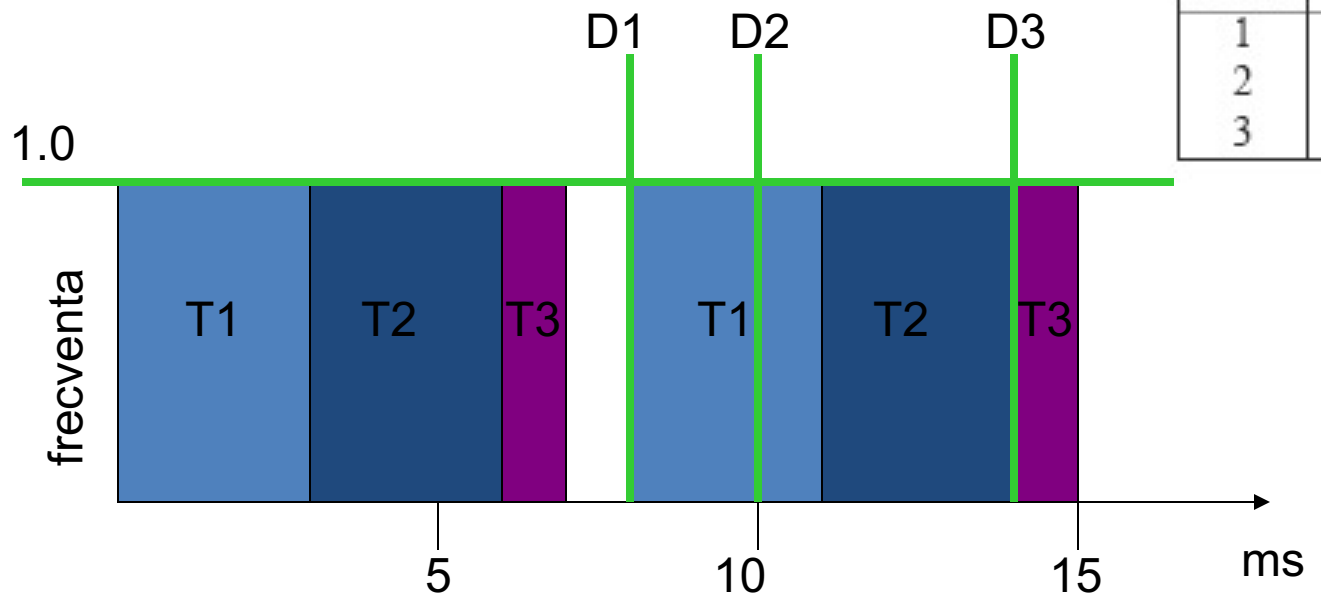
Task	Invocation 1	Invocation 2
1	2 ms	1 ms
2	1 ms	1 ms
3	1 ms	1 ms

- 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)

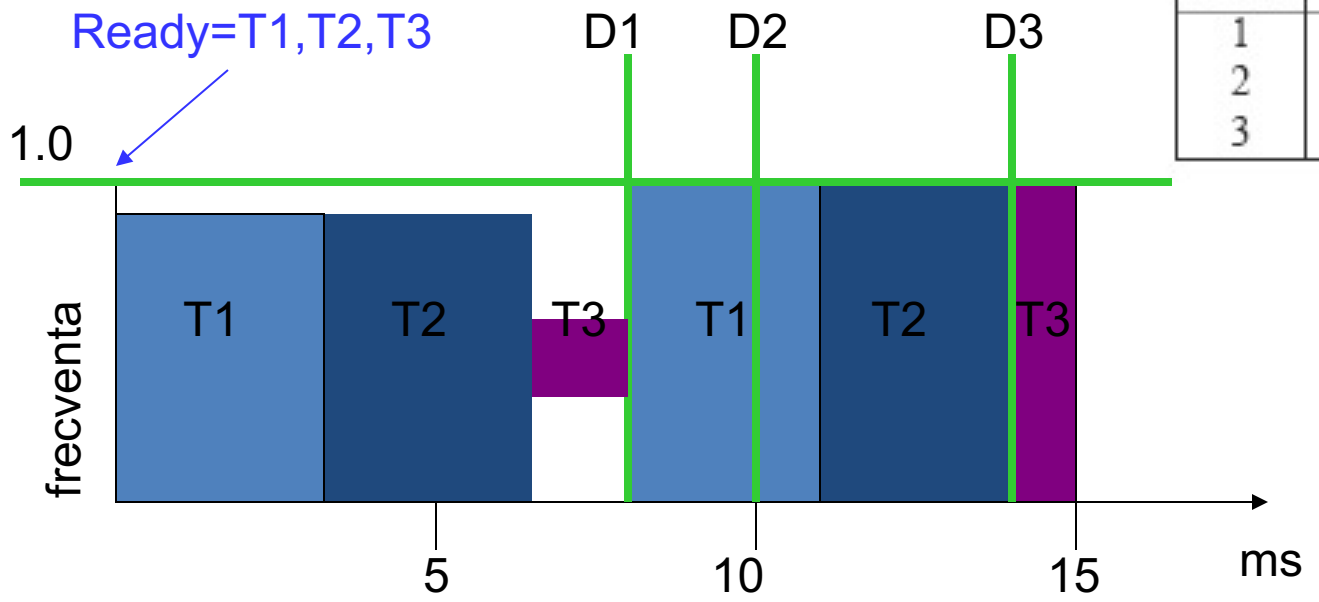
Exemplu



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

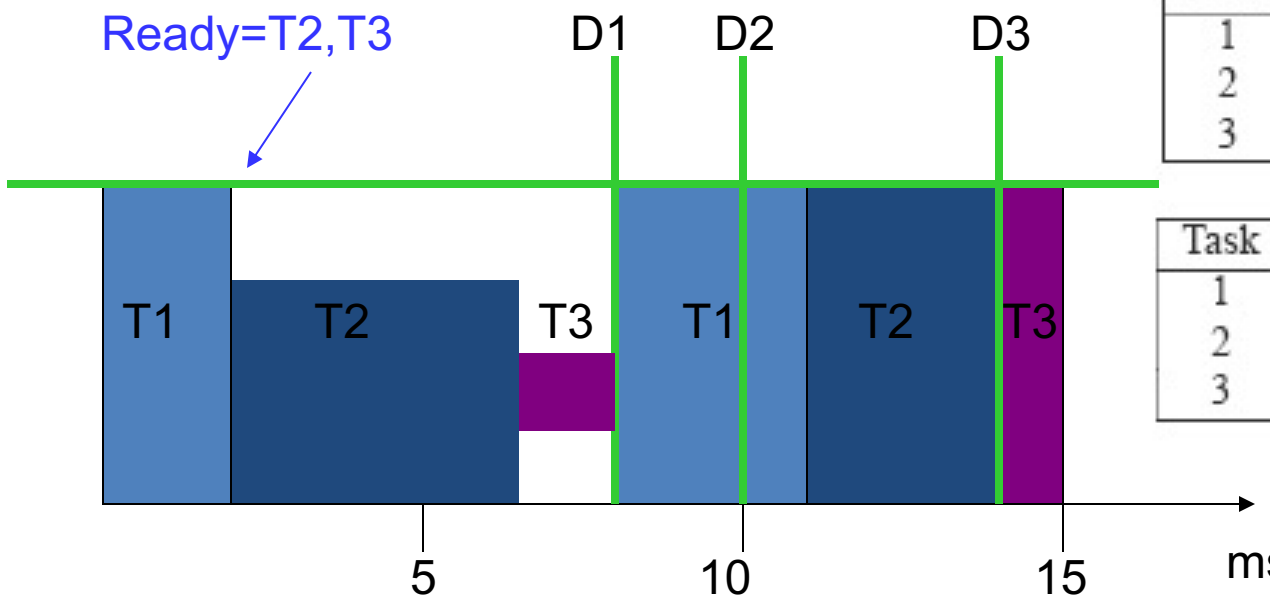
Exemplu



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

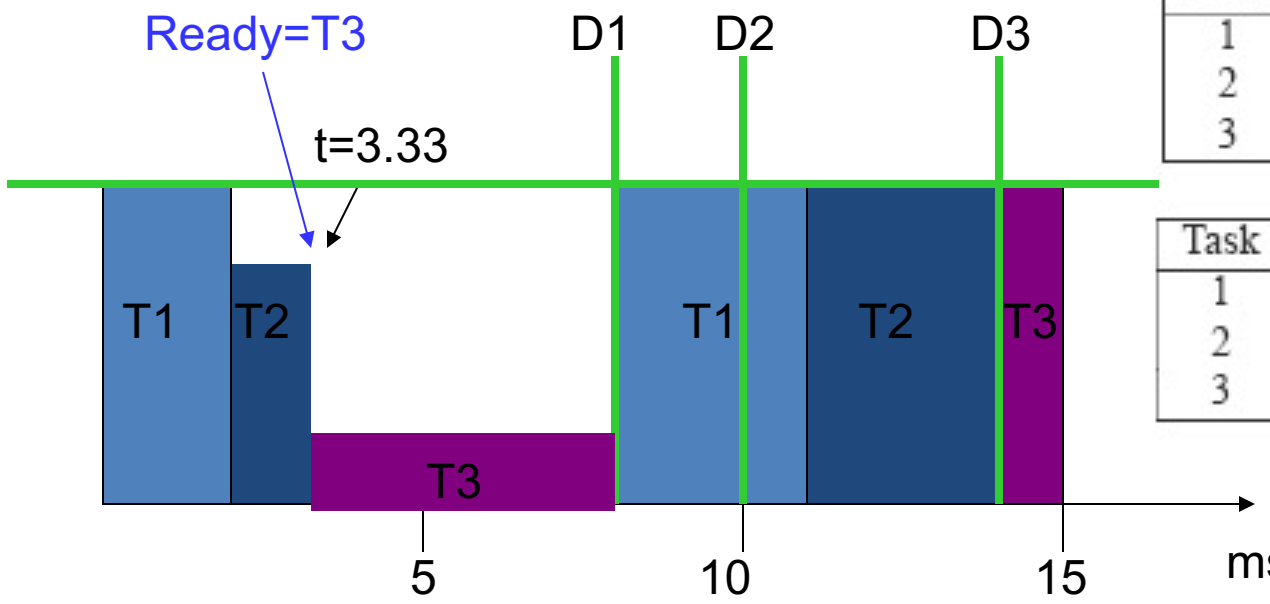


Task	Computing Time	Period
1	3 ms	8 ms
2	3 ms	10 ms
3	1 ms	14 ms

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

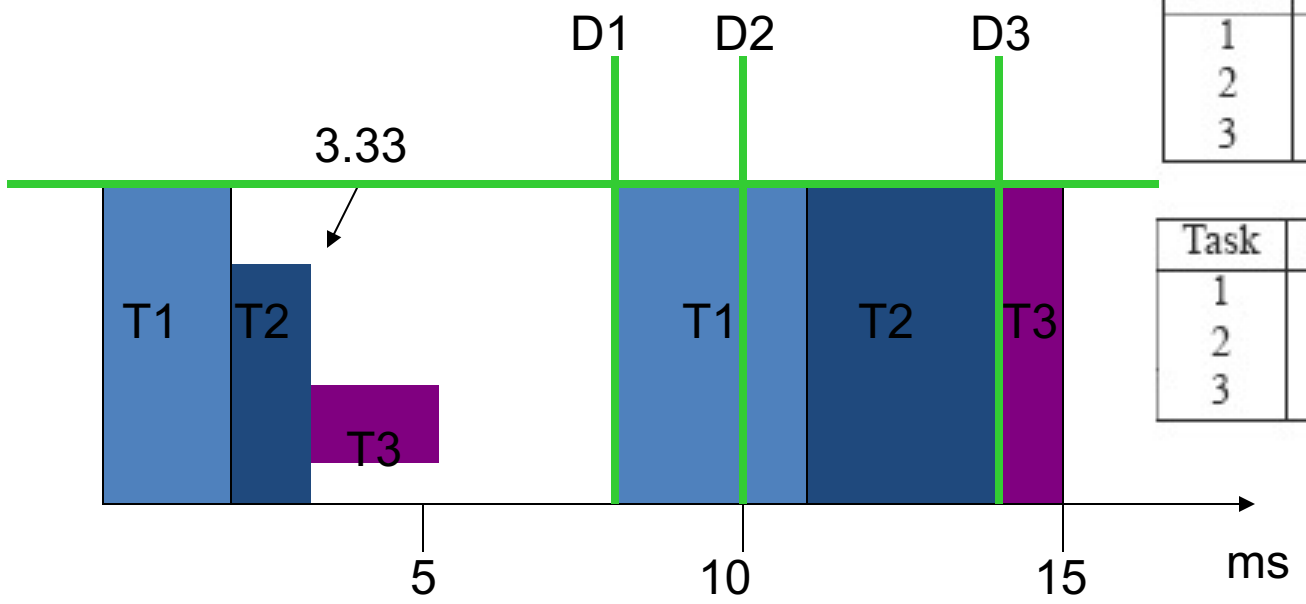


Task	Computing Time	Period
1	3 ms	8 ms
2	3 ms	10 ms
3	1 ms	14 ms

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

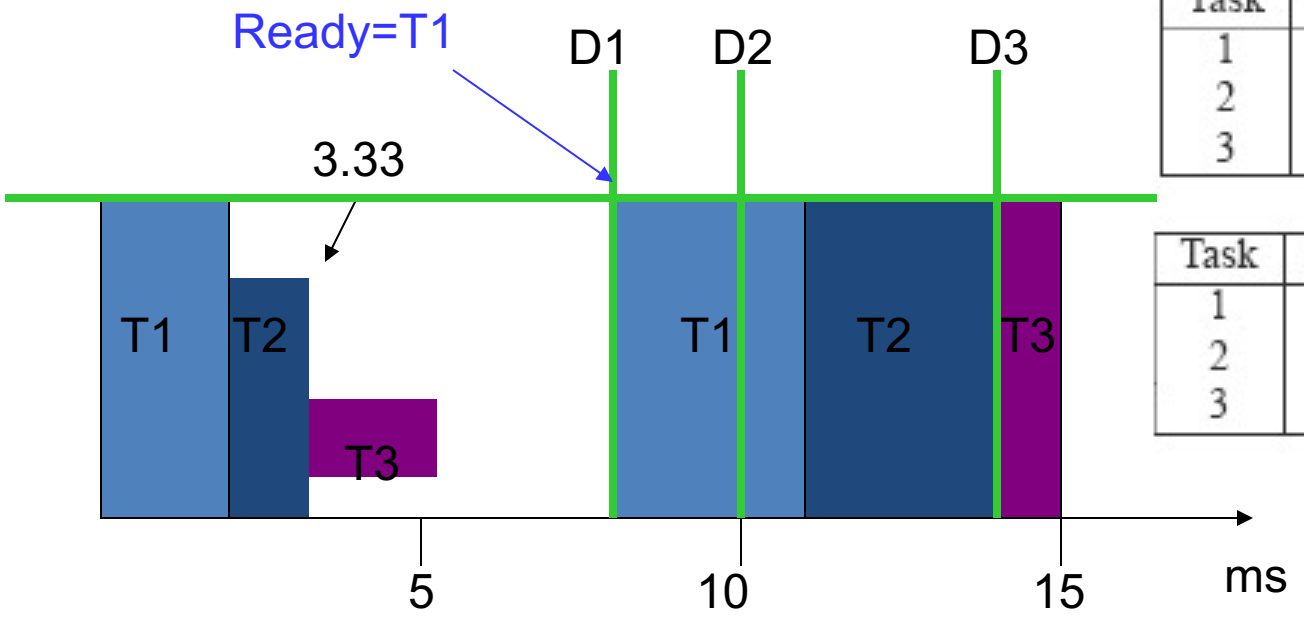
Exemplu



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

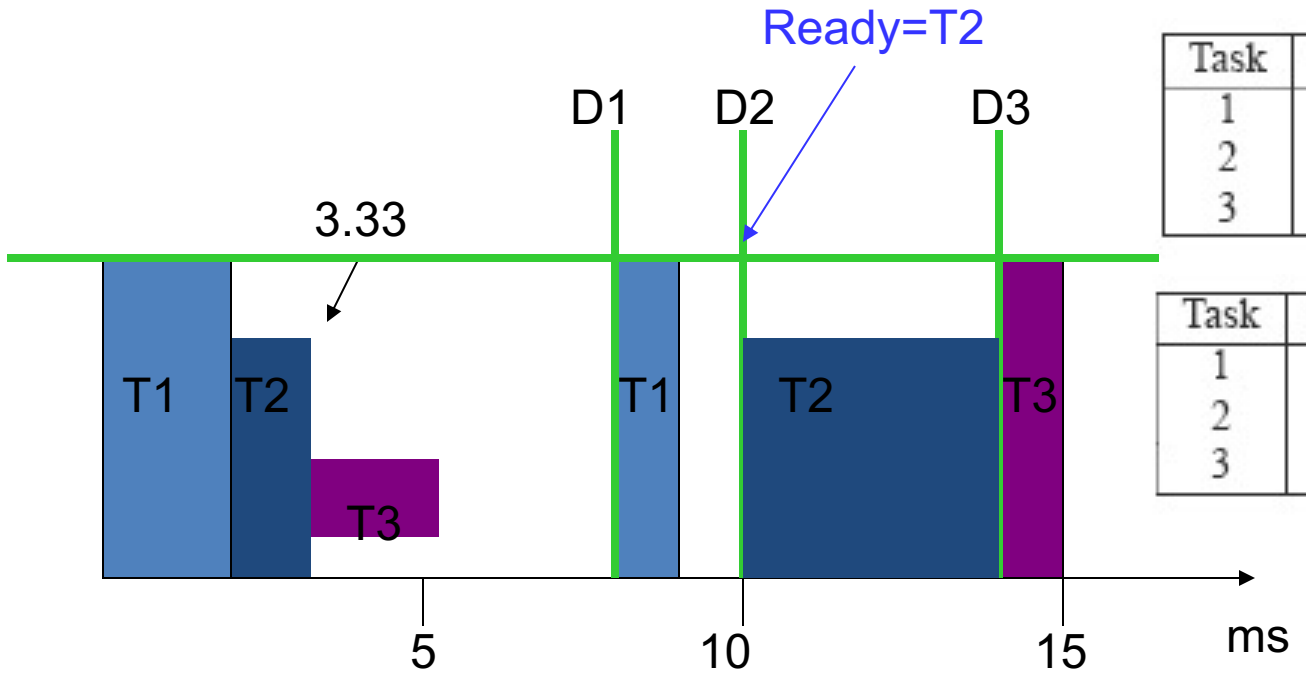


Task	Computing Time	Period
1	3 ms	8 ms
2	3 ms	10 ms
3	1 ms	14 ms

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

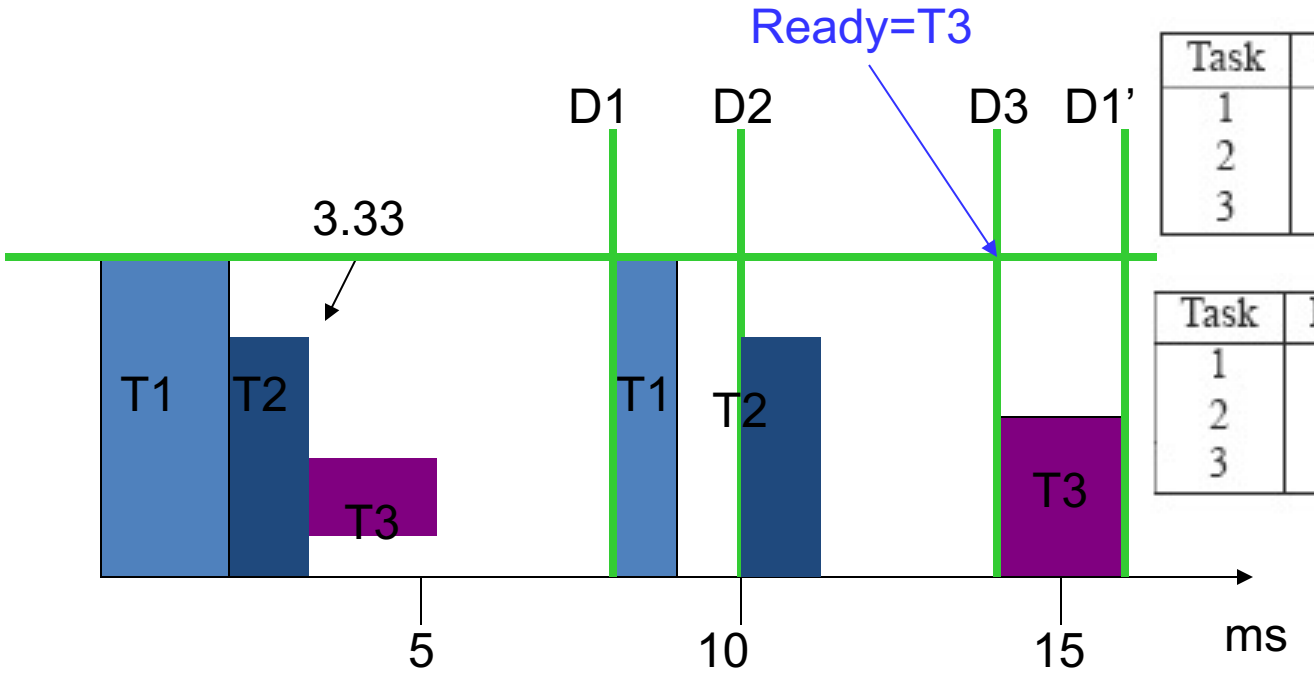


Task	Computing Time	Period
1	3 ms	8 ms
2	3 ms	10 ms
3	1 ms	14 ms

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

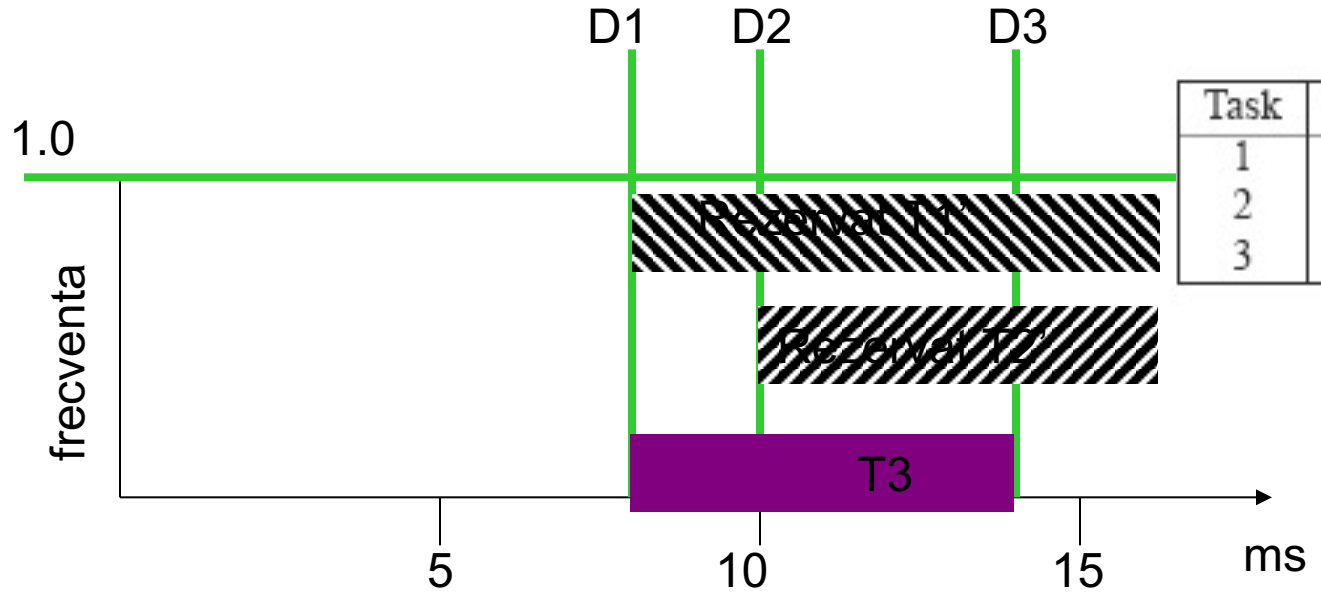
Task	Invocation 1	Invocation 2
1	2 ms	1 ms
2	1 ms	1 ms
3	1 ms	1 ms

- 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

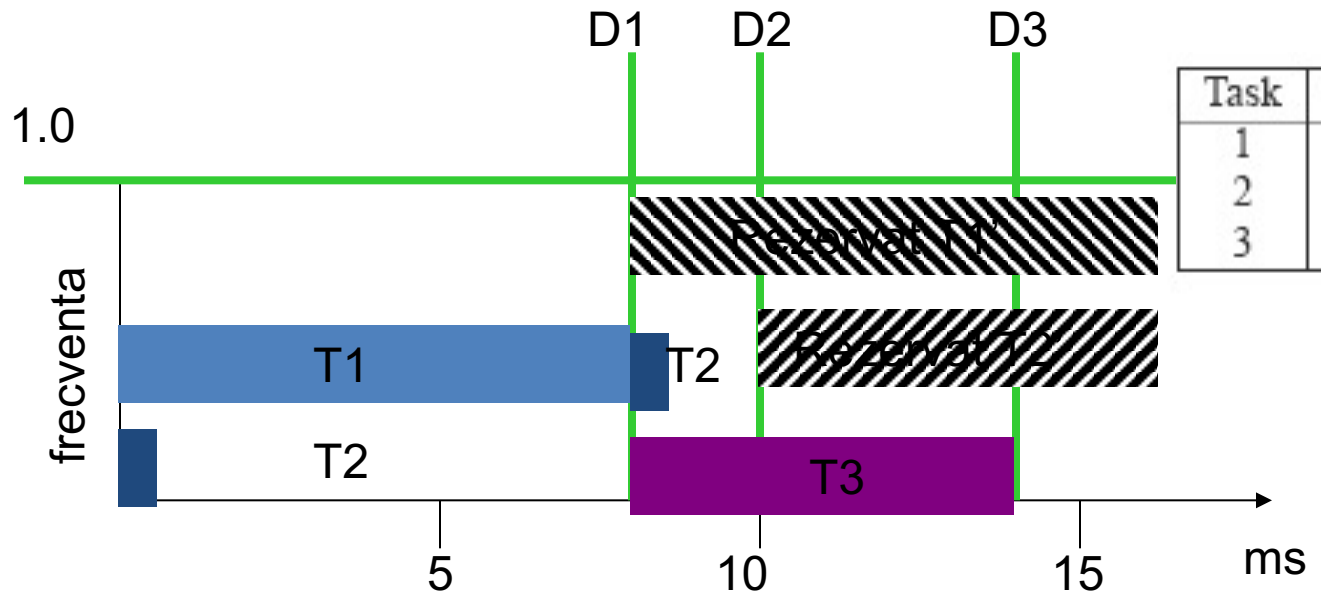
Look-ahead EDF



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

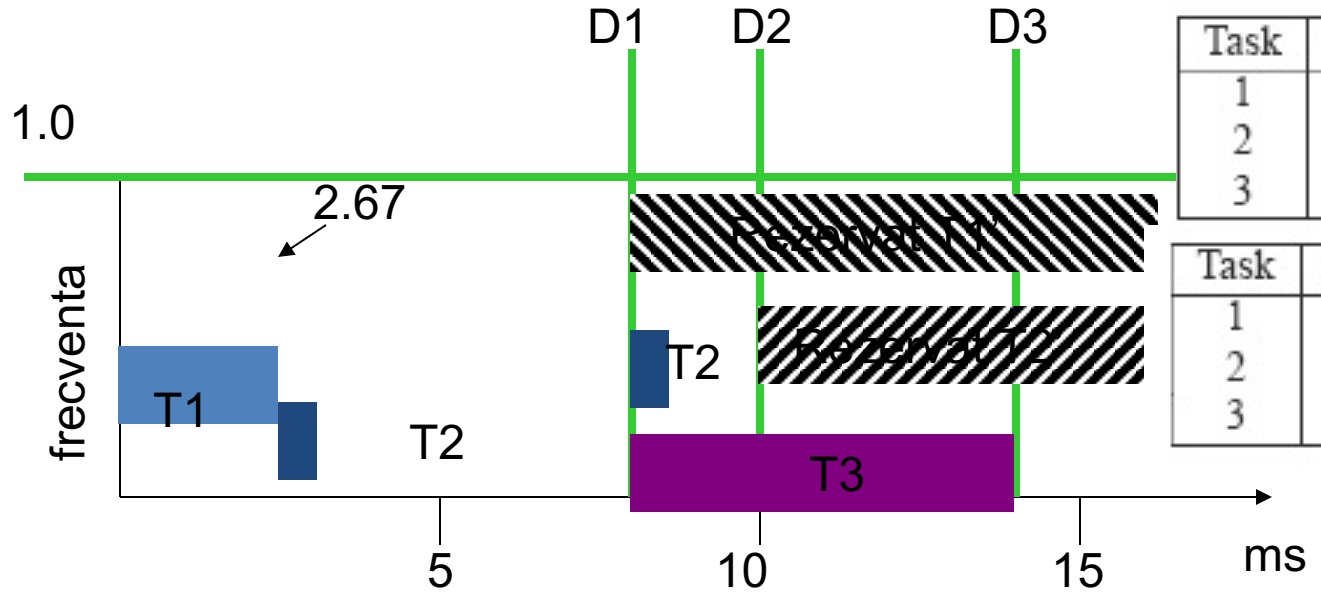
Look-ahead EDF



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

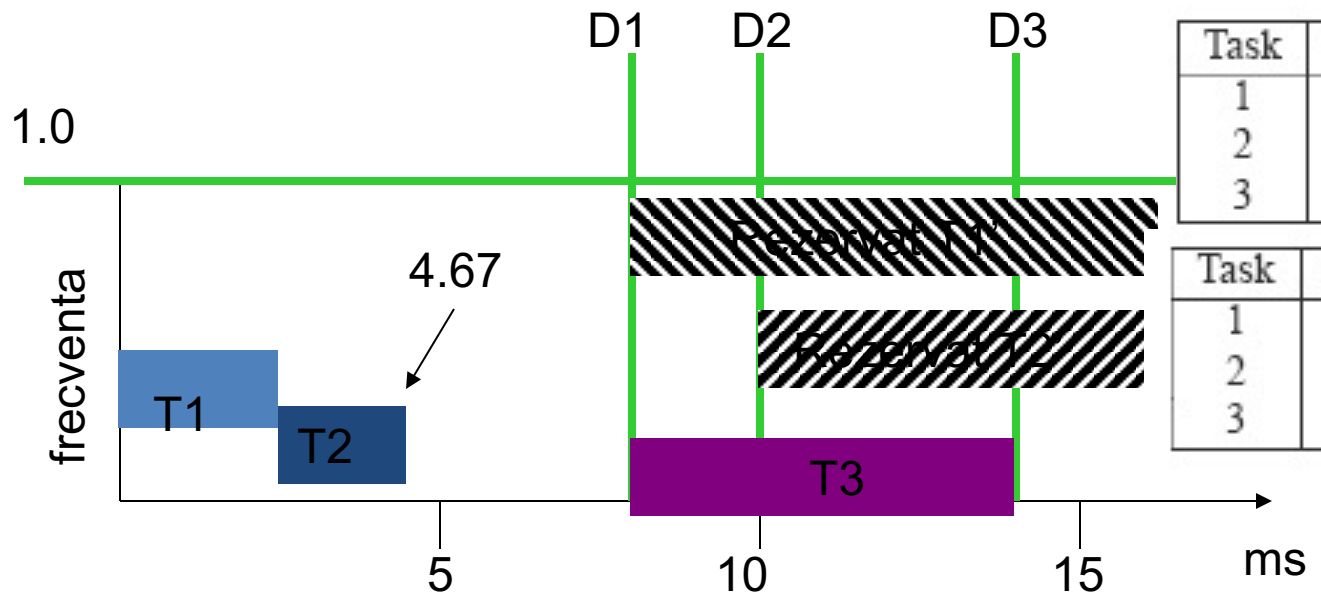


Task	Computing Time	Period
1	3 ms	8 ms
2	3 ms	10 ms
3	1 ms	14 ms

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

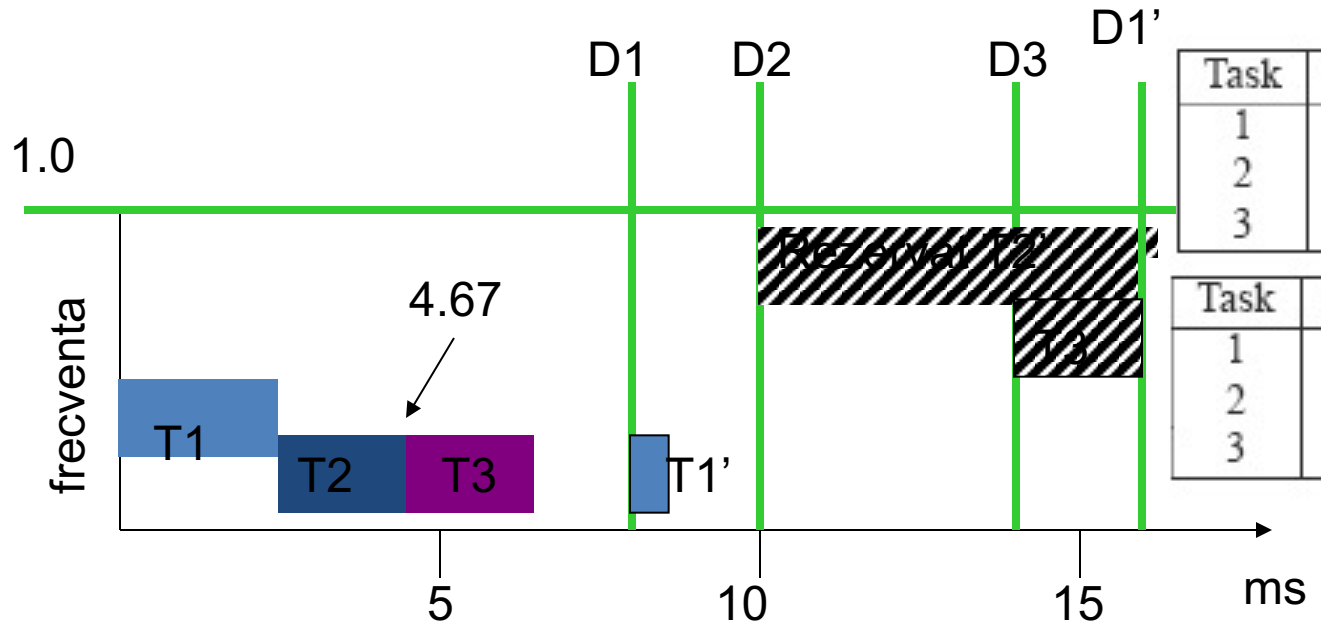


Task	Computing Time	Period
1	3 ms	8 ms
2	3 ms	10 ms
3	1 ms	14 ms

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

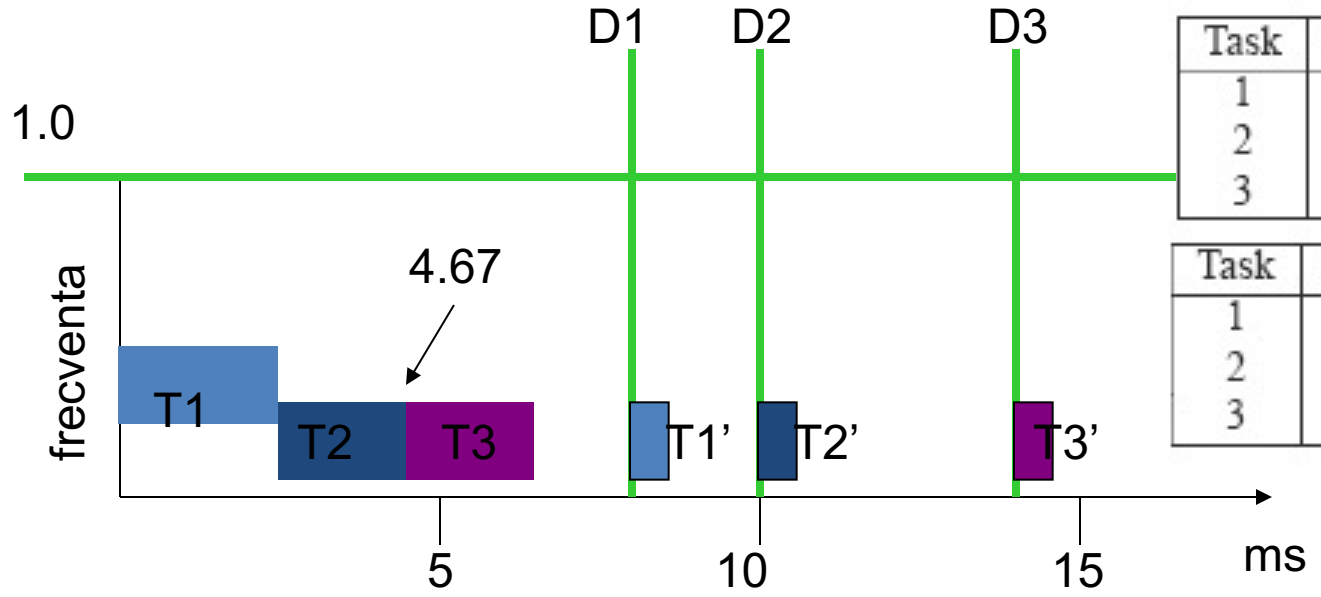


Task	Computing Time	Period
1	3 ms	8 ms
2	3 ms	10 ms
3	1 ms	14 ms

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