DESIGN OF RTOS SYSTEMS

Metrika za ocjenu performansi RTOS

Prosjećno vrijeme odziva:

$$\overline{t_r} = \frac{1}{n}\sum_{i=1}^{n}(f_i - a_i)$$

Ukupno vrijeme kompletiranja:

$$t_c = \max_i(f_i) - \min_i(a_i)$$

Težinska suma vremena kompletiranja:

$$t_w = \sum_{i=1}^{n} w_i f_i$$

DESIGN OF RTOS SYSTEMS

Maksimalno zakašnjenje:

$$L_{max} = \max_i (f_i - d_i)$$

Maksimalni broj zakašnjelih taskova:

$$N_{late} = \sum_{i=1}^{n} miss(f_i)$$

gdje:

$$miss(f_i) = \begin{cases} 0 & \text{if } f_i \leq d_i \\ 1 & \text{otherwise} \end{cases}$$

DESIGN OF RTOS SYSTEMS

Adekvatnost metrike

Prosjećno vrijeme odziva nije adekvatno za tvrde RT taskove, pošto nema procjene perioda ili deadlajna.

Ukupno vrijeme kompletiranja takodjer nije adekvatno.

Težinska suma vremena kompletiranja može biti relevantna ako taskovi imaju različitu važnost.

Maksimalno zakašnjenje: ova metrika je u redu ako se resursi mogu dodavati sve dok maksimalno zakašnjenje ne postane=0.

DESIGN OF RTOS SYSTEMS

Rasporedjivanje anomalija

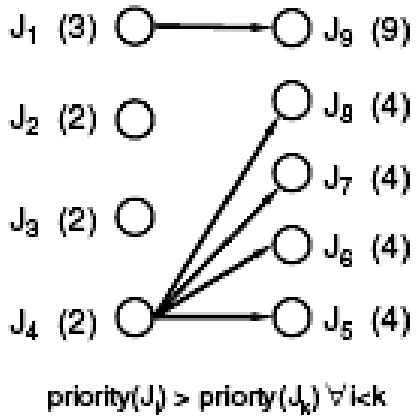
Teorem Grahama

Ako je skup taskova optimalno rasporedjen na multiprocesorskoj arhitekturi, sa nekim doznačavanjem prioriteta, sa fiksnim brojem procesora, fiksnim vremenima izvršenja i ograničenjima precedencije (prethodjenja), tada povećanje broja procesora, smanjivanje vremena izvršenja, ili slabljenje ograničenja precedencije može dovesti do povećanja dužine rasporedjivanja.

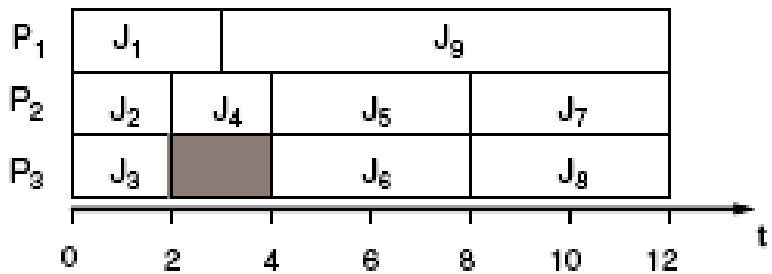
Pokažimo to na slijedećim primjerima:

DESIGN OF RTOS SYSTEMS

Dodavanje procesora može povećati dužinu rasporedjivanja: Predpostavimo skup taskova sa precedencijama prikazanim na slijedećoj slici:

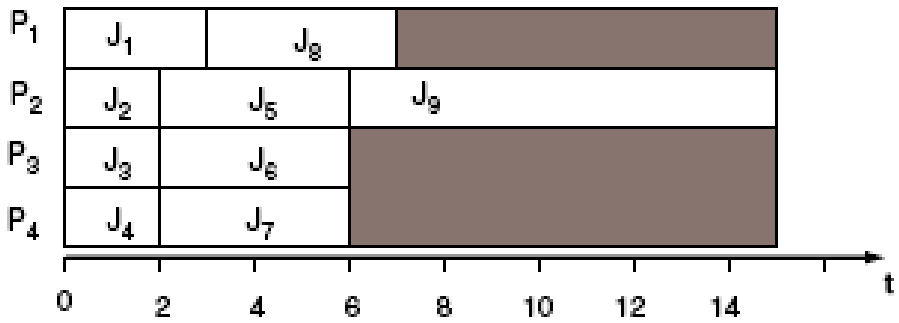


Optimalno rasporedjivanje na troprocesorskoj mašini je dato kao na slijedećoj slici:

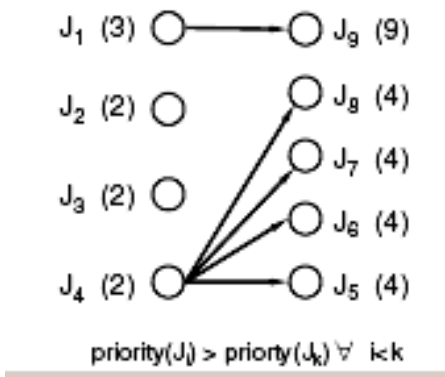


DESIGN OF RTOS SYSTEMS

Na 4-ro procesorskoj mašini rasporedjivanje će postati gore:



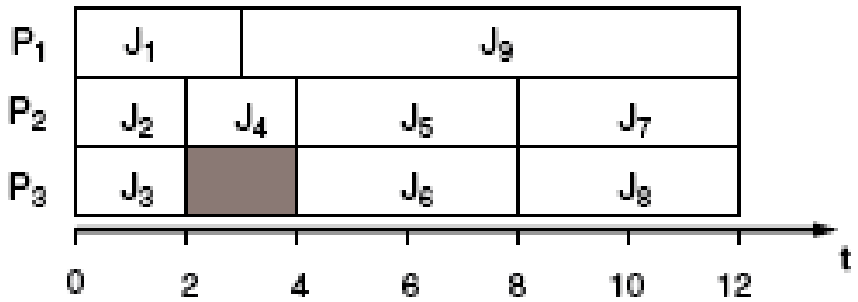
Rasporedjivanje skupa taskova na četvero procesorskoj mašini je prikazano slijedećim dijagramom:



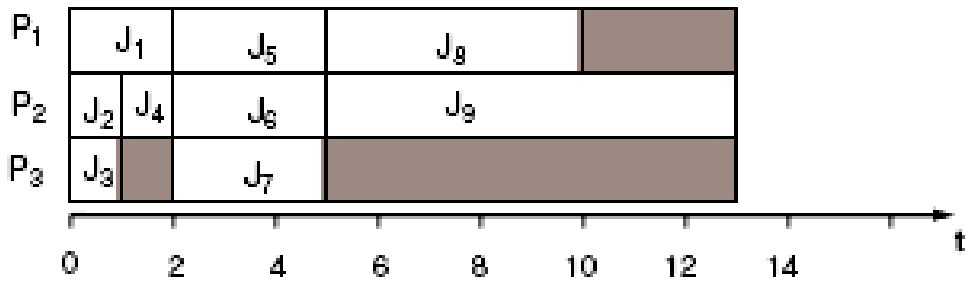
DESIGN OF RTOS SYSTEMS

Smanjenje vremena računanja povećava vrijeme rasporedjivanja

Pogledajmo prvo rasporedjivanje sa originalnim vremenima računanja:



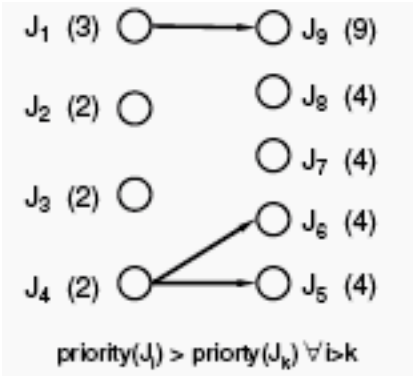
a onda rasporedjivanje sa svim vremenima računanja reduciranim za 1 jedinicu vremena:



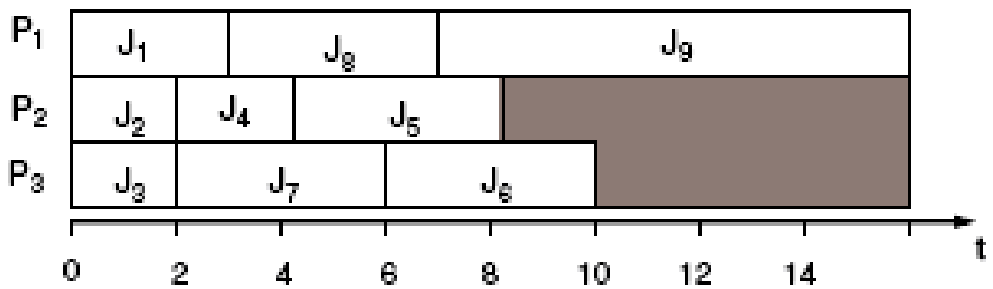
DESIGN OF RTOS SYSTEMS

Slabljenje ograničenja precedencije povećava vrijeme rasporedjivanja:

Oslabljena precedencija:



Rasporedjivanje za oslabljenu precedenciju:



DESIGN OF RTOS SYSTEMS

Rasporedjivanje aperiodičnih taskova:

Klasifikaciona šema Grahama:
Algoritmi rasporedjivanja su klasificirani sa tripletom:

$$\alpha \mid \beta \mid \gamma$$

gdje :

α opisuje mašinsko okruženje na kojem skup taskova treba da se rasporedjuje (jednoprocesorski, distribuirane arhitekture,...)

β opisuje taskove i karakteristike resursa (priemptivni, sinhrone aktivnosti, itd..)

γ indicira kriterij optimalnosti (mjeru performanse), koja će se slijediti kod rasporedjivanja.

DESIGN OF RTOS SYSTEMS

Primjeri klasifikacije

- 1 ! prec! Lmax
  - jednoprocesorska mašina
  - task je setovan sa ograničenjima precedencije
  - minimizira maksimalno zakašnjenje
- 3 !no-preemt ! Σf_i
  - troprocesorska mašina
  - nije dozvoljena priempcija
  - minimizira sumu vremena završavanja

- 2 ! sync ! $\sum$ $Late_i$
  - dvoprocesorska mašina
  - taskovi imaju sinhrona vremena dospjeća, i nemaju drugih ograničenja.
  - minimiziraju broj zakašnjelih taskova

DESIGN OF RTOS SYSTEMS

Jackson-ov algoritam

Klasifikacija : 1 ! sync! L_{max}

- jedan procesor
- task se sastoji od pojedinačnih poslova, ima sinhrona vremena pristizanja,
- mogu imati različita vremena
- minimiziraju broj zakašnjelih taskova

Druga ograničenja nisu uključena

Iz ovoga slijedi da su taskovi nezavisni , tj.

- nema relacija precedencije
- nema djeljenih resursa

DESIGN OF RTOS SYSTEMS

Predpostavke:

- svi taskovi su aktivirani u trenutku t=0

slijedi da za svaki i : J_i može biti karakterizirano sa

- vremenom računanja C_i
- relativnim (= apsolutnim) deadlajnom D_i

Slijedi da : Skup taskova J = $\{J_i(C_i, D_i) \mid i = 1 : n\}$

Ovo je algoritam rasporedjivanja poznat kao : najraniji datum ili earleast due date (EDD) , čiji je autor Jackson.

Teorem 3 (Jackson-ovo pravilo)

Za dati skup od n nezavisnih taskova, svaki algoritam koji izvršava taskove u redoslijedu neopadajućih deadlajna je optimalan u odnosu na minimizaciju maksimalnih zakašnjenja.

DESIGN OF RTOS SYSTEMS

Dokaz:

Neka je σ optimalno rasporedjivanje proizvedeno sa bilo kojim algoritmom A , gdje A $\neq$ EDD

Slijedi da $\exists\ J_a, J_b\ ,\ d_a \leq d_b : J_b \rightarrow J_a$

Neka je σ' dobijeno iz σ zamjenom J_a i J_b , tj. $J_a \rightarrow J_b$.

Preostaje da se pokaže da:

zamjenjujući pozicije J_a i J_b ne može povećati $L_{max.}$

Ako je tako σ se može transformisati sa konačnim brojem transpozicija u σ_{EDD} i pošto ni jedna transpozicija ne može povećati L_{max} , σ_{EDD} je optimalno.

DESIGN OF RTOS SYSTEMS

Dokaz Jackson-ovog pravila

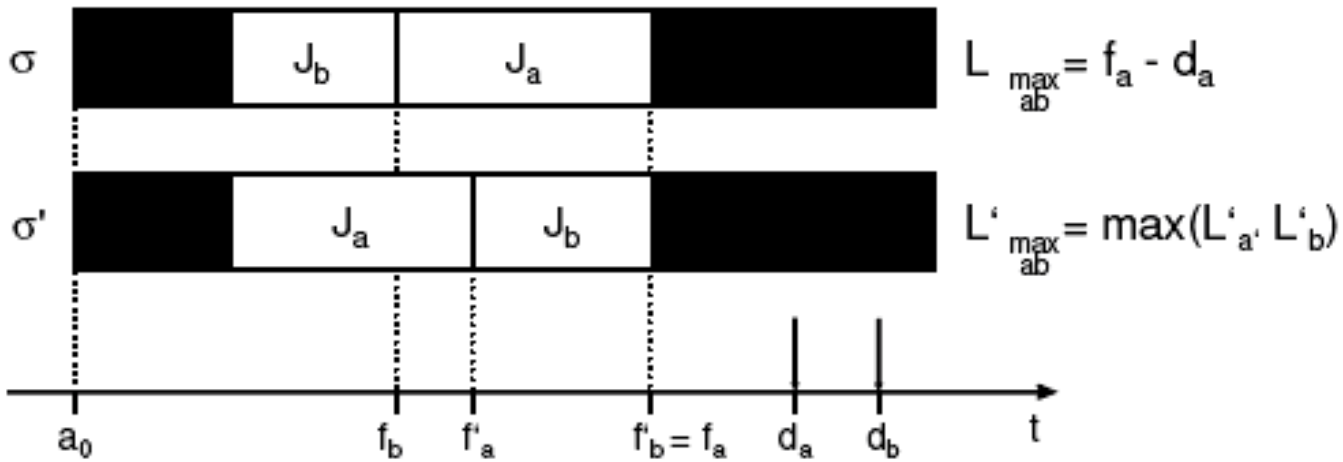
U σ slijedi da vrijedi : Lmax (a, b) = fa - da
(d_a je raniji deadlajn i J_a je kasniji task)

U σ' L'max (a, b) = max (L'a, L'b)

case 1: $L'_a \geq L'_b \Rightarrow L'_{max}(a,b) = f'_a - d_a$
since $f'_a < f_a \Rightarrow L'_{max}(a, b) < L_{max}(a, b)$
case 2: $L'_a \leq L'_b \Rightarrow L'_{max}(a, b) = f'_b - d_b = f_a - d_b$
since $d_a < d_b \Rightarrow L'_{max}(a, b) < L_{max}(a, b)$

DESIGN OF RTOS SYSTEMS

Dokaz Jackson-ovog pravila (ilustracija)



if ($L'_a \geq L'_b$) then $L'^{max}_{ab} = f'_a - d_a < f_a - d_a$

if ($L'_a \leq L'_b$) then $L'^{max}_{ab} = f'_b - d_b < f_a - d_a$

$L'^{max}_{ab} < L^{max}_{ab}$

u obadva slučaja:

DESIGN OF RTOS SYSTEMS

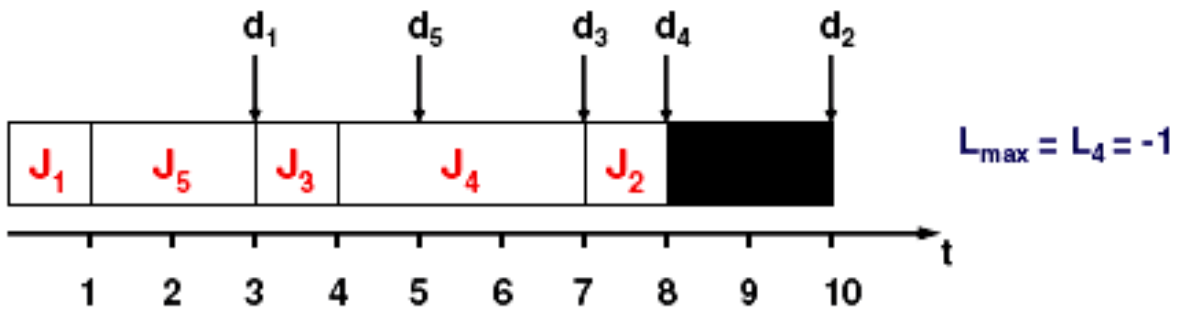
Kompleksnost Jackson-ovog algoritma:

Kompleksnost Jackson-ovog algoritma samo zbog sortiranja taskova po rastučem deadlajnu.

Complexity (EDD) = Complexity (sorting) = O (n log n)

Primjer 1 za algoritam EDD

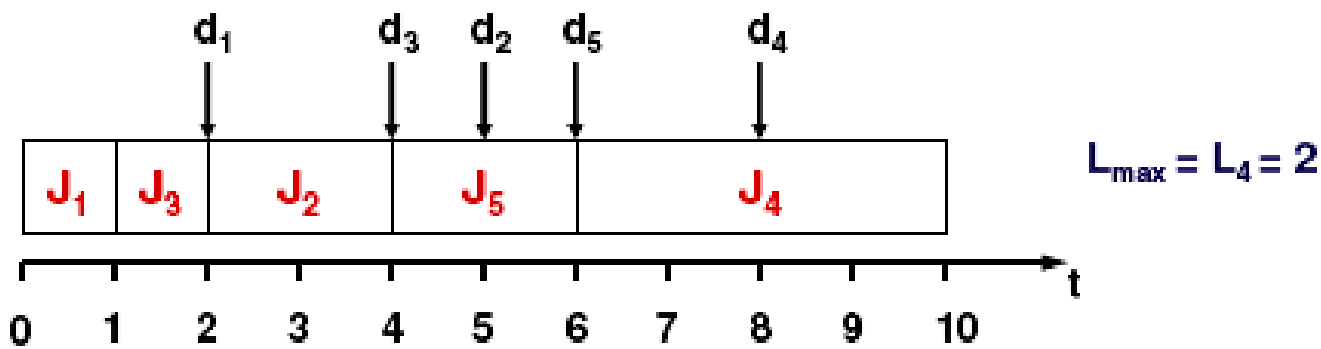
	J_1	J_2	J_3	J_4	J_5
C_i	1	1	1	3	2
d_i	3	10	7	8	5



DESIGN OF RTOS SYSTEMS

Primjer br. 2 za EDD algoritam (nije ostvarivo)

	J_1	J_2	J_3	J_4	J_5
C_i	1	2	1	4	2
d_i	2	5	4	8	6



$L_{max} = L_4 = 2$

Opaska: EDD ne može garantirati ostvarivo rasporedjivanje. On samo garantuje da ako postoji ostvarivo rasporedjivanje onda će ga algoritam EDD naći.

DESIGN OF RTOS SYSTEMS

Garancija u slučaju EDD algoritma

Pokazaćemo da za svako i izmedju 1 : n : $f_i \leq d_i$

Predpostavićemo da taskovi $J_1 : J_n$ su izlistani sa rastućim deadlajnima.

$$\Rightarrow f_i = \sum_{k=1}^{i} c_k$$

Tj. test garancije znači da se verificira slijedećih n uslova :

$$\forall i = 1 : n : \sum_{k=1}^{i} c_k \leq d_i$$

DESIGN OF RTOS SYSTEMS

Hornov algoritam

Sada ćemo da relaksiramo ograničenje sinhronog pristizanja taskova:
U isto vrijeme dozvolićemo priempciju, što bi trebalo da olakša rasporedjivanje.

Klasifikacija : 1 ! Preem! L_{max}

- mono procesor
- dinamičko pristizanje , priempcija dozvoljena
- minimizacija maksimalnog zakašnjenja

Prinicip rasporedjivanja : najraniji deadlajn prvi (EDF)

DESIGN OF RTOS SYSTEMS

Teorem 4 (Hornov)

Za dati skup od n nezavisnih taskova sa arbitrarnim vremenima pristizanja, bilo koji algoritam koji u bilo kojem trenutku izvršava task sa najranijim absolutnim deadlajnom izmedju svih spremnih taskova je optimalan u smislu minimizacije maksimalnog zakašnjenja.

Kompleksnost po tasku je : umetanje novo prispjelog taska u uredjenu listu na propisan način : O (log n)

Kompleksnost za n taskova je dakle : O (n log n)

DESIGN OF RTOS SYSTEMS

Dokaz Hornovog algoritma

Neka je A arbitrarni algoritam koji proizvodi σ , $\sigma \neq \sigma_{EDF}$

priempcija je dozvoljena , slijedi da svaki task se može izvršavati u razdvojenim vremenskim intervalima.

Predpostavimo postojanje vremenskih odrezaka dužine jedne vremenske jedinice:

Neka je σ (t) = task koji se izvršava u odresku [t, t+1)

E (t) = spremni task koji, u trenutku t, ima najraniji deadlajn

t_E (t) = vrijeme (≥t) kod kojeg slijedeći odrezak od E(t) počinje svoje izvršenje u tekućem rasporedu.

$$\sigma \neq \sigma_{EDF} \Rightarrow \exists\, t \text{ in } \sigma : \sigma(t) \neq E(t)$$

DESIGN OF RTOS SYSTEMS

Mi smo pokazali da izmjenjujući pozicije od σ(t) i E (t) ne može se povećati maksimalno zakašnjenje. Ako σ starta kod t=0 i D= max [d_{ij}] , σ_{EDF} se može dobiti iz σ pomoću transpozicije D.

Dokaz Hornovog algoritma : zamjenjivanje (interchange)

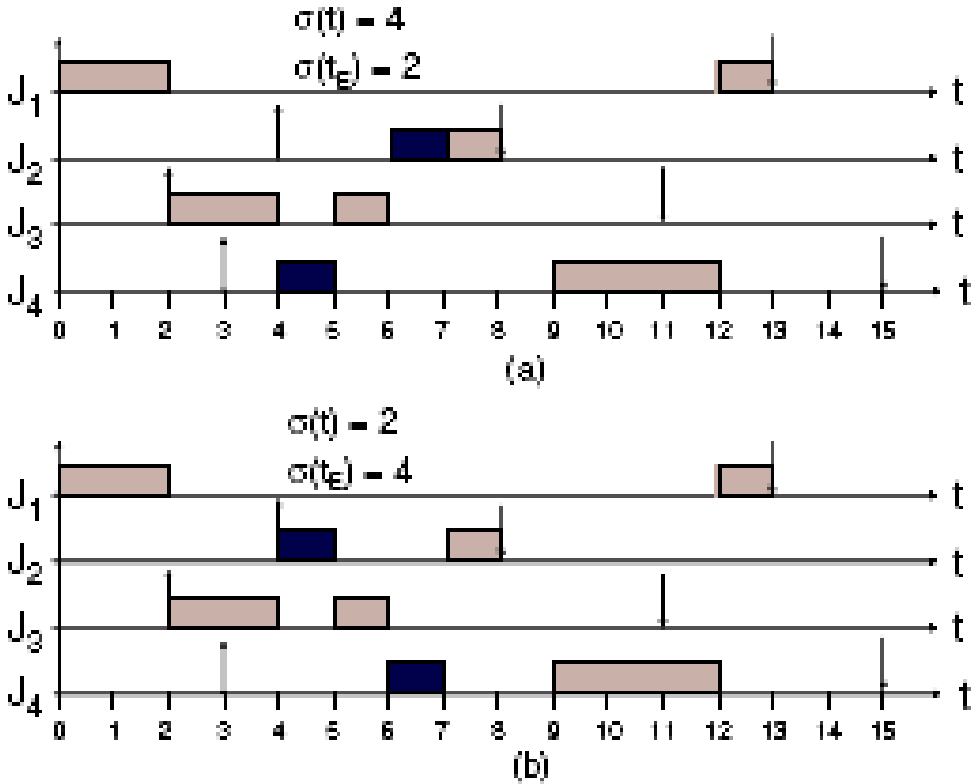
Opaska : mi ćemo na kratko zamjeniti i pisati t_E umjesto t_E (t)

```
Algorithm : interchange
 {
      for( t=0 to D-1 ) {
         if (σ(t) ≠ E(t) ) {
              σ(t_E) = σ(t) ;
              σ(t)   = E(t) ;
         }
      }
 }
```

tj. za svaki odrezak vremena t , provjerava se da li je σ (t) = E (t). Ako nije odrezak kod t i E (t) se zamjenjuju.

DESIGN OF RTOS SYSTEMS

Ilustracija zamjenjivosti



Dokaz optimalnosti EDF algoritma

(a) rasporedjivanje σ u trenutku t = 4

(b) novo rasporedjivanje dobijeno poslije transpozicije

DESIGN OF RTOS SYSTEMS

Zamjenjivanje taskova u slučaju Horna je manje više isto onome kod Jackson-ovog slučaja.

Odatle, sa argumentima koji su korišteni za dokaz Jackson-ovog algoritma, slijedi da zamjenjivanje maksimalne zakašnjenosti neće je povećati.

Iz ovoga proističe zaključak da je EDF optimalno rješenje.

DESIGN OF RTOS SYSTEMS

Sačuvavanje rasporedivosti

Corollary

Ako je σ rasporedivo tada je i σ_{EDF} rasporedivo

Dokaz

Mi ćemo pokazati da svaka transpozicija sačuvava rasporedivost.
U svakom trenutku vremena :
svaki vremenski odrezak je :

- bilo predvidiv (anticipiran)
- ili odgodjen do t_E

Ako je odrezak vremena anticiparan sa ostvarivošću ovoga , task će se sačuvati.
Predpostavimo da je odrezak vremena od J_i je odgodjen do t_E

DESIGN OF RTOS SYSTEMS

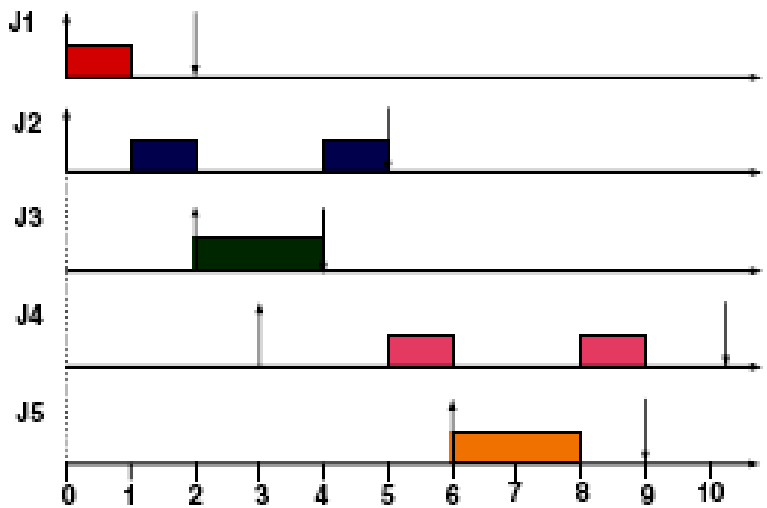
σ ostvarivo => $(t_E + 1) \leq d_E$

$$\Rightarrow (\forall i : d_E \leq d_i) \quad (t_E + 1) \leq d_i$$

slijedi da J_i odloženo do t_E je ostvarivo.

Primjer za EDF rasporedjivanje

	J_1	J_2	J_3	J_4	J_5
a_i	0	0	2	3	6
c_i	1	2	2	2	2
d_i	2	5	4	10	9



DESIGN OF RTOS SYSTEMS

Garancija u EDF algoritmu

Kod EDF algoritma test garancije je neophodan uvjek kada novi task prispije.
Predpostavimo da J je tekuči skup aktivnih taskova, garantirani J_{new} je novo prispjeli task.

J_{new} se može prihvatiti ako J' = J U { J_{new} } je rasporediv.

Mi moramo testirati : da za svako J_i , i : n : $f_i \leq d_i$
Predpostavimo da J' je uredjen po rastučem deadlajnu.

Taskovi su priemptabilni, slijedi da kada novi J_{new} stigne u trenutku t, neki taskovi su mogli biti parcijalno izvršeni.
Neka C_i (t) je preostali najgori slučaj vremena izvršenja od J_i.

DESIGN OF RTOS SYSTEMS

U trenutku t najgori slučaj vremena završetka J_i se može izračunati kao :

$$f_i = \sum_{k=1}^{i} C_k(t)$$

Rasporedivost se može garantirati ako su slijedeći uslovi tačni :

$$\forall i = 1:n: \left(\sum_{k=1}^{i} C_k(t) \right) \leq d_i$$

Pošto je $f_i = f_{i1} + C_i(t)$ test dinamičke garancije se može provesti u vremenu O (n).

DESIGN OF RTOS SYSTEMS

Algoritam za EDF test garancije

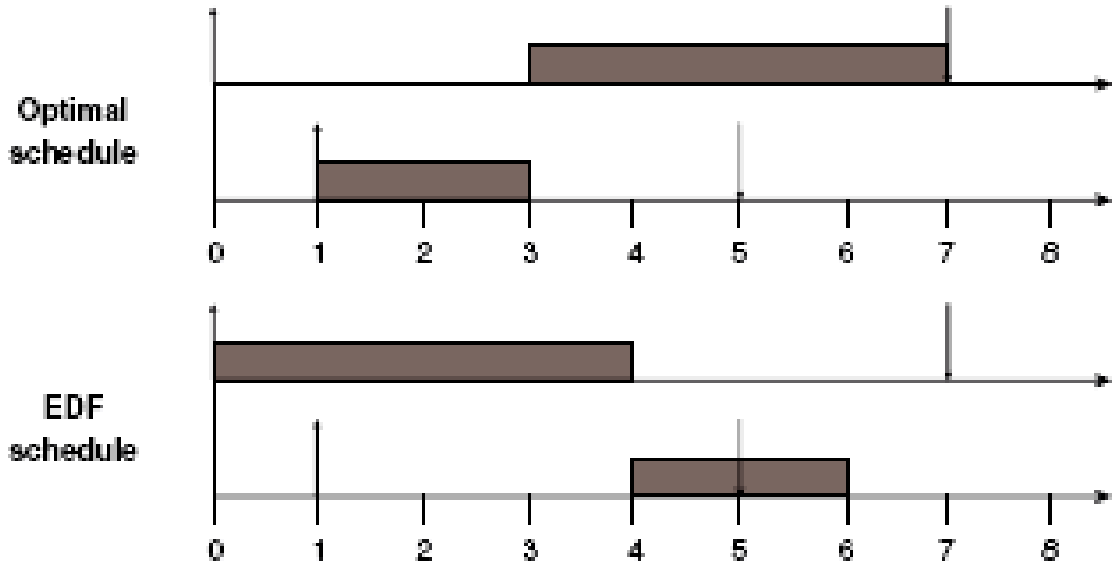
```
Algorithm : EDF guarantee( I, J_new)
{
        J' = J ∪ {J_new }:        /* ordered by deadline */
        t = current-time ():
        f_0 = 0 ;
        for ( each J_i ∈ J' ) {
             f_i  =  f_i-1 + C_i( t ) ;
             If ( f_i > d_i )  return (INFEASIBLE) ;
        }
        return(FEASIBLE) ;
}
```

DESIGN OF RTOS SYSTEMS

Neprieemptivno rasporedjivanje

Predpostavimo arbitrarna vremena pristizanja i bez priempcije. U tom slučaju EDF algoritam rasporedjivanja nije optimalan.

	J_1	J_2
a_i	0	1
c_i	4	2
d_i	7	5



DESIGN OF RTOS SYSTEMS

O ovom prethodnom primjeru, procesor mora biti besposlen (idle) u untervalu [0,1] da bi se dobilo optimalno rasporedjivanje.

•Ako vremena pristizanja nisu apriori poznata, ni jedan online algoritam ne može odlučiti da li procesor treba biti idle.

•Bilo je pokazano da u slučaju non-idle rasporedjivanja, EDF je još uvjek optimalan algoritam (Jeffray, Starat, Martel , 1991).

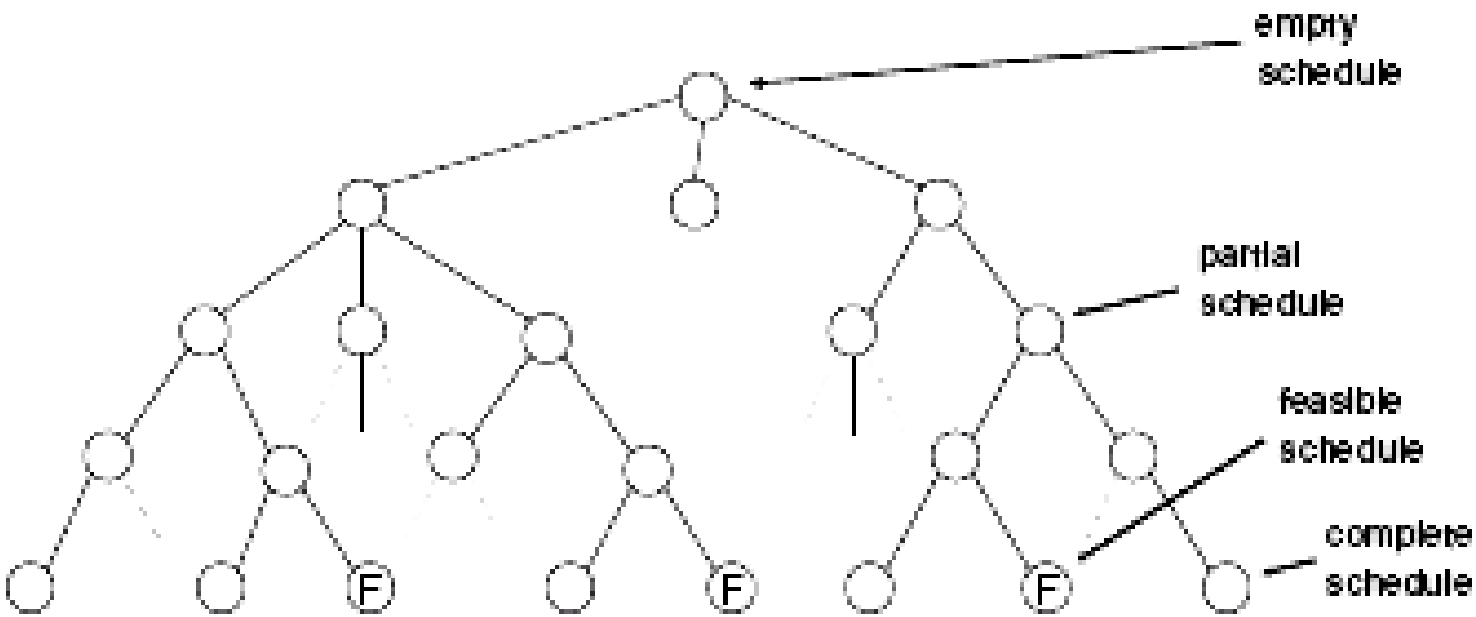
DESIGN OF RTOS SYSTEMS

Drvo traženja

Predpostavimo da su vremena prispjeća a priori poznata.

Tada traženje za ostvarivo rasporedjivanje podrazumjeva tehniku probe i greške (trial and error).

Obićno se izvršava na drveću traženja (search trees):



DESIGN OF RTOS SYSTEMS

Kompleksnost algoritama drveta traženja

U svakom koraku traženja novi task se unosi u drvo, rezultirajući u novom parcijalnom rasporedjivanju.

Neka je *n* broj taskova
slijedi da je dubina drveta *n*

Pošto svaki list čvora predstavlja jednu permutaciju aranžmana taskova, postoji n! lista čvorova.

Najgori slučaj znači da se analizira n! staza dužine *n* , odakle slijedi da je kompleksnost reda O (n * n!)

DESIGN OF RTOS SYSTEMS

Heuristički algoritmi traženja

Red O (n* n!) se ne može dozvoliti u realističnim sistemima i zbog toga prostor traženja se mora reducirati.
Jedan od mogućih algoritama redukcije pretraživanja je granaj i ograniči (branch –and- bound) algoritam.

Razmotričemo dva primjera ovakvih algoritama:

- Bratley je pokušao da skreše drvo koristeći dodatne informacije.
- Metod izvornog kernela (spring – kernel) koji pokušava da slijedi samo obečavajuće staze.

- Opaska: Ovakvi heuristički algoritmi mogu proizvesti ostvarivo rasporedjivanje u polinomijalnom vremenu, ali oni ne garantiraju da će ga i naći.

DESIGN OF RTOS SYSTEMS

Bratley-ev algoritam

Ovaj algoritam rasporedjivanja je tipa:

1! Non-preem! Feasible

Što znači da je za jedan procesor, nepriemtivan i ostvariv.

Ovaj algoritam je jednostavna tehnika odsjecanja (pruning) drveta. Grana se napušta ako:

* dodavanje još jednog čvora tekučoj stazi će prouzrokovati da se deadlajn ne ispuni
* ostvarivo rasporedjivanje je nadjeno na tekučoj stazi.

DESIGN OF RTOS SYSTEMS

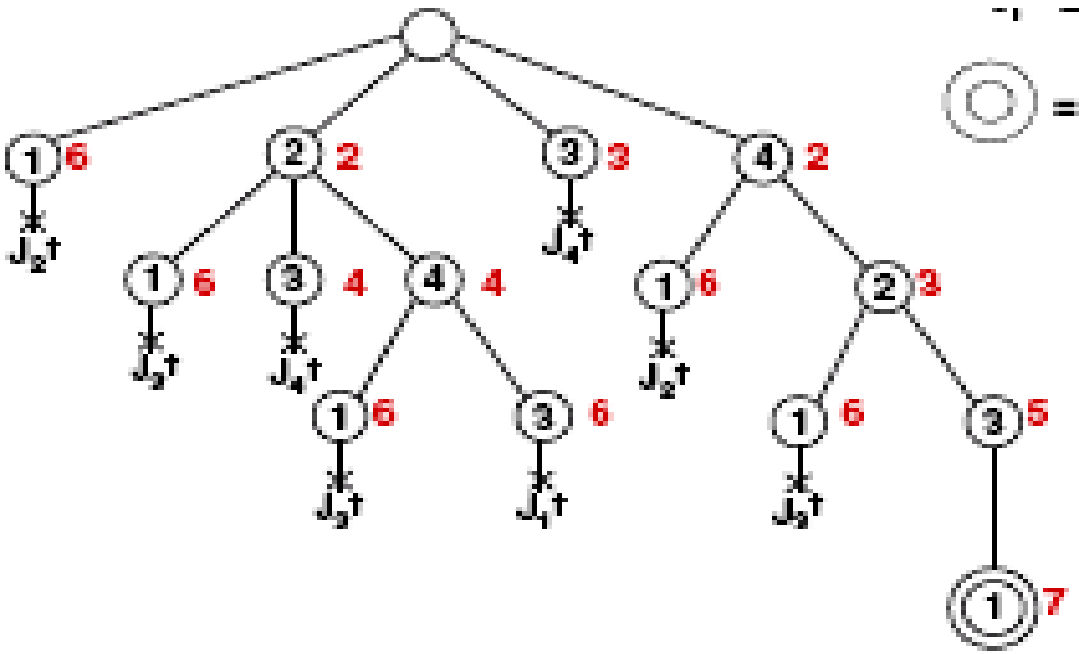
Primjer:

Na slijedećem primjeru oznake koje se koriste imaju slijedeća značenja:

broj u čvoru = rasporedjenom tasku

broj van čvora = vremenu završetka

J_i^* je task koji je propustio svoj deadlajn



= ostvarljivo rasporedjivanje

	J_1	J_2	J_3	J_4
a_i	4	1	1	0
c_i	2	1	2	2
d_i	7	5	6	4

DESIGN OF RTOS SYSTEMS

Spring algoritam

Spring je RT kernel koji je dizajniran od strane Stankovića i Ramamarith-a sa MIT univerziteta u Amherstu.
Spring je ekstremno fleksibilan algoritam, i podržava takodjer i distribuirane sisteme. Spring se primjenjuje na svakom nivou drveta traženja, i radi se o heurističkoj funkciji koja služi da se odluči koju stazu treba nastaviti slijediti.
Primjeri za ove heurističke funkcije su :

H = a	First Come First Served (FCFS)
H = C	Shortest Job First (SJF)
H = d	Earliest Deadline First (EDF)
H = Test	Earliest Start Time First (ESTF)
H = d + w C	EDF + SJF
H = d + w Test	EDF + ESTF

DESIGN OF RTOS SYSTEMS

Razmatranja ograničenja resursa u Spring algoritmu

Svaki task mora deklarisati binarni niz (array) resursa:

R_i= [R_1 (i) ,……, R_r (i)] , gdje R_j (i) =1 označava resurs j koji je potreban tasku i.

Za dato parcijalno rasporedjivanje, algoritam odredjuje najranije vrijeme kada specifični resurs je raspoloživ:

EAT_k (earleast available time za resurs k) tj . Najranije raspoloživo vrijeme za resurs k.

Task ne može startati ranije nego što je vrijeme kada su svi njegovi resursi raspoloživi:

$$\text{Test}(i) = \max_k [\, a_i, \max EAT_k]$$

DESIGN OF RTOS SYSTEMS

Ograničenja precedencije

Faktor E (eligibility) se dodaje na H (gdje eligible znači – ima pravo, kandidat je, može da konkuriše)

$E_i = 1$: znači da su svi prethodnici u grafu precedencije kompletirani

$E_i = \infty$: u svim ostalim slučajevima.

Dok proširuje parcijalno rasporedjivanje, algoritam odredjuje da li dobijeno rasporedjivanje je strogo ostvarivo, tj. ostvarivo takodjer i ako se proširi i sa bilo kojim od preostalih taskova.

Ako je parcijalno rasporedjivanje nadjeno koje nije strogo ostvarivo, algoritam se zaustavlja i izvještava da je task označen kao nerasporediv.

Kompleksnost : ostvarivo rasporedjivanje koje je dostignuto kroz n čvorova , kod svakog čvora se može izračunati kao

n x H , i slijedi da je kompleksnost ovog algoritma reda $O(n^2)$

DESIGN OF RTOS SYSTEMS

Modifikacije Spring algoritma

Spring algoritam napreduje pravolinijski , i ukoliko dodje do njegovog neuspjeha, to može da bude indikacija postojanja lokalnog trapa (zamke).
Riješenje za ovo je metod ***backtrackinga*** (praćenja unatrag)

Ako parcijalno rasporedjivanje je nadjeno da nije strogo ostvarivo, onda se može krenuti jedan korak unatrag i promjeniti slijedeće najbolje riješenje od H.

Redukcija kompleksnosti
Ne primjenjivati H na sve preostale taskove nego samo na k onih preostalih taskova sa najmanjim deadlajnima.
Ako je k konstantno i malo onda je kompleksnost reda $O(n)$

DESIGN OF RTOS SYSTEMS

Rasporedjivanje sa ograničenjima precedencije

U opštem slučaju sa stanovišta kompleksnosti ovo je NP-hard problem (nondeterministic polynomial time).
Za neke specijalne slučajeve moguči su algoritmi sa polinomijalnim vremenom.

Posljednji deadlajn prvi (LDF – latest deadline first) je jedan od njih koji je tipa :

1! Precedence sync! L_{max}

Za dati skup J od n taskova i diagram koji opisuje njihove relacije precedencije, predpostavlja se da su vremena pristizanja simultana.

DESIGN OF RTOS SYSTEMS

Algoritam LDF gradi red rasporedjivanja od repa ka vrhu na slijedeći način:

* izmedju taskova koji nemaju nasljednike, ili sa svim nasljednicima već izabranim i stavljenim u red.
* LDF izabire onaj sa najmanjim deadlajnom da bude posljednji rasporedjen, i ovo pravilo poštuje dok nisu svi taskovi rasporedjeni.

U vremenu izvršenja (runtime) taskovi se vade iz glave reda tako da prvi task koji je umetnut u red bit će posljednji izvršen.

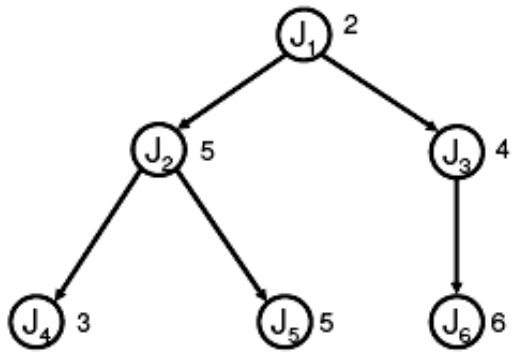
DESIGN OF RTOS SYSTEMS

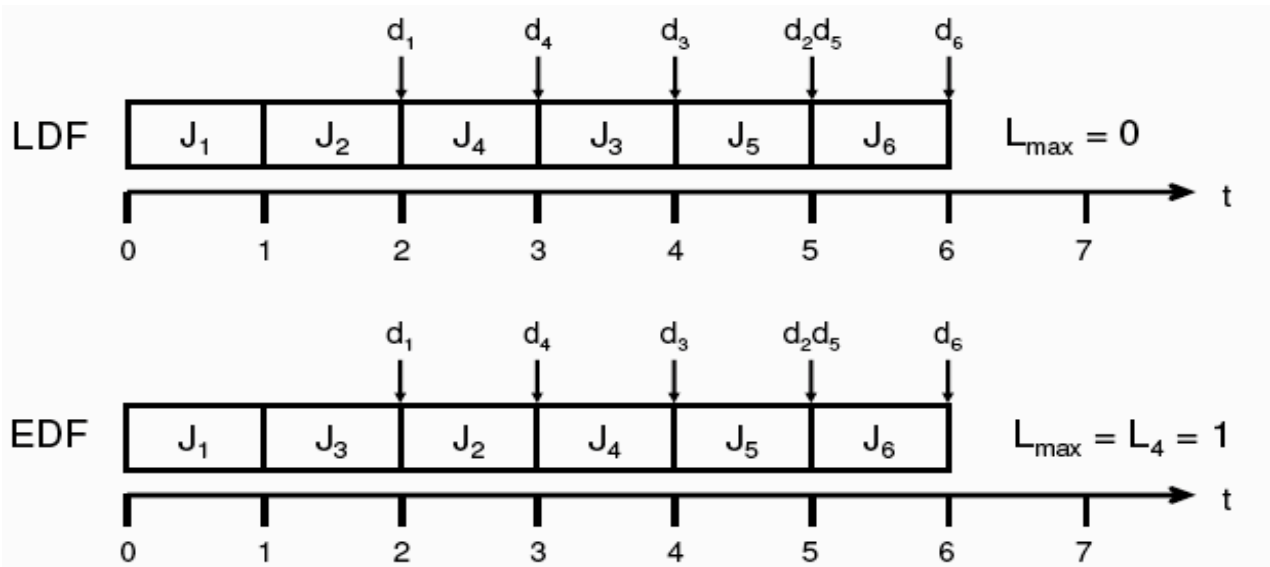
Primjer LDF algoritma rasporedjivanja i njegova kompleksnost

Kompleksnost je reda O (n^2) , pošto za svaki od n koraka mora se posjetiti graf precedencije da bi se našao podskup Γ koji nema nasljednika.

Primjer ovakvog rasporedjivanja je :

	J_1	J_2	J_3	J_4	J_5	J_6
C_i	1	1	1	1	1	1
d_i	2	5	4	3	5	6





DESIGN OF RTOS SYSTEMS

Dokaz optimalnosti LDF

Neka je J kompletan skup taskova , $\Gamma \subseteq J$ je podskup taskova bez nasljednika.

Neka je J_Γ task u Γ sa posljednjim deadlajnom d_Γ .
Predpostavimo da je σ bilo koje rasporedjivanje koje ne slijedi LDF algoritam.
Posljednji rasporedjeni task J_k nije onaj sa najkasnijim deadlajnom , tj. $D_k < d_i$.

Γ je particionirano u $\Gamma = A \cup \{J_l\} \cup B \cup \{J_k\}$, aranžirano u ovom raedoslijedu u σ .

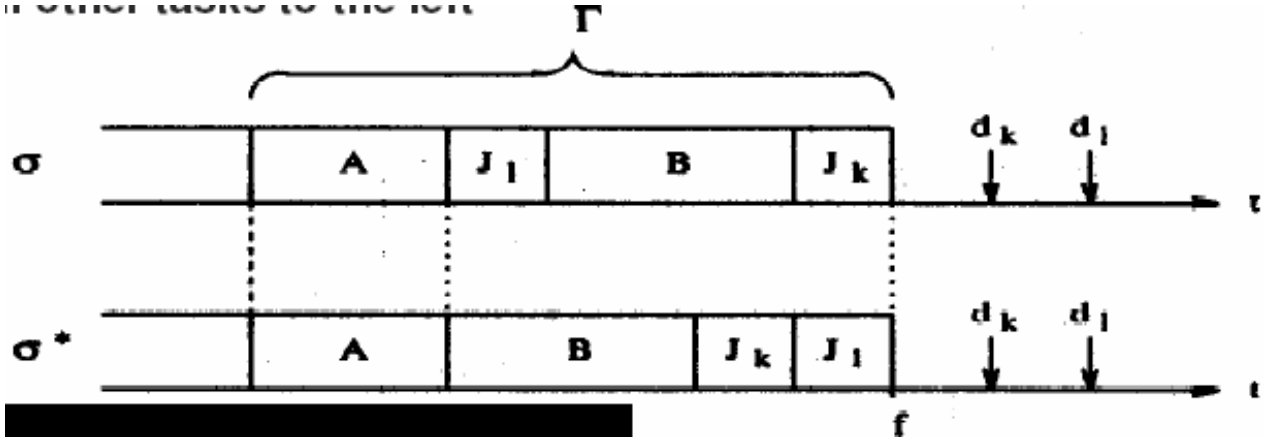
Maksimalno zakašnjenje za Γ je veće ili jednako $L_k = f - d_k$ sa

$$f = \sum_{i=1}^{k} c_i$$

DESIGN OF RTOS SYSTEMS

Potrebno je dokazati da ostavljanje J_i na kraj za rasporedjivanje ne može povećati maksimalno zakašnjenje u Γ.

Neka je σ^* dobijeno iz σ pomjeranjem J_i na kraj reda i zatim pomicanjem svih drugih taskova uljevo kao što je prikazano na slijedećoj slici:



Svaki argument od $L^*_{max}(\Gamma)$ nije veći od $L_{max}(\Gamma)$ pošto :

- $L^*_{max}(A) = L_{max}(A) \le L_{max}(\Gamma)$ pošto A nije pomaknuto
- $L^*_{max}(B) \le L_{max}(B) \le L_{max}(\Gamma)$ jer B starta ranije u σ^*
- $L^*_k \le L_k \le L_{max}(\Gamma)$ because J_k pošto J_k starta ranije u σ^*
- $L^*_l = f - d_l \le f - d_k \le L_{max}(\Gamma)$ pošto $d_k \le d_i$

DESIGN OF RTOS SYSTEMS

Pomjeranje J_i na kraj rasporedjivanja ne povećava L_{max} (Γ) , tj. rasporedjivanje J_i kao posljednjeg taska po algoritmu posljednjeg deadlajna će minimizirati L_{max} (Γ).
Ponavljanje ove procedure sa preostalih n-1 taskova $\in J - \{J_l\}$
i tako uzastopno kompletira ovaj dokaz.

EDF algoritam sa ograničenjima precedencije

Uzmimo da je ovaj algoritam tipa :

1! Prec, preem ! L_{max}

Predpostavimo n taskova sa ograničenjima precedencije i dinamičkim aktivacijama.
Ovaj problem je rješiv u polinomijalnom vremenu , samo u slučaju kada je priempcija dozvoljena.

DESIGN OF RTOS SYSTEMS

Rješenje Chetto, Silly i Bouchentouf

Transformišimo skup J zavisnih taskova u skup J^* nezavisnih taskova sa adekvatnim modifikacijama vremenskih parametara.

Nakon toga ćemo primjeniti EDF algoritam.
Transformacija obezbjedjuje da:
J^* je rasporedivo , iz čega slijedi da je i J rasporedivo i sva ograničenja su zadovoljena.

Modifikacije vremena oslobadjanja (release) i deadlajna u smislu da niti jedan task ne može startati prije svojih preteča i ne može priemptirati svoje nasljednike su uključena (ovo se ne odnosi na priempciju drugih taskova osim nasljednika).

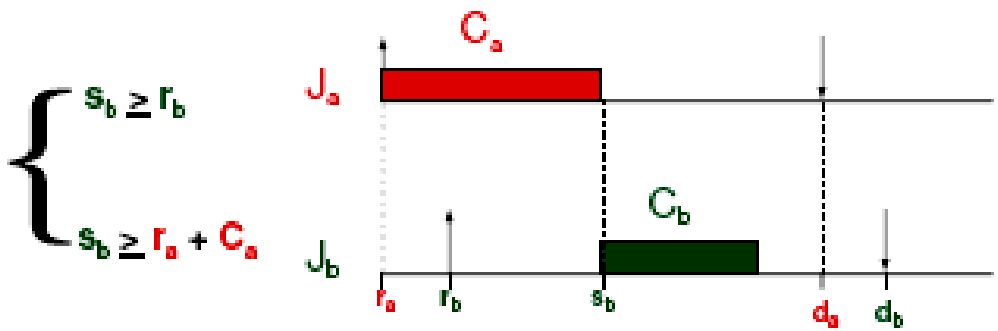
DESIGN OF RTOS SYSTEMS

Modifikacija vremena oslobadjanja

Neka su data dva taska J_a i J_b i neka $J_a \rightarrow J_b$.

Slijedeća dva uslova moraju biti zadovoljena:

* $s_b \geq r_b$ (J_b ne može biti startano prije njegovog vremena oslobadjanja)
* $s_b \geq r_a + c_a$ (J_b ne može biti startano sve dok J_a nije završio)

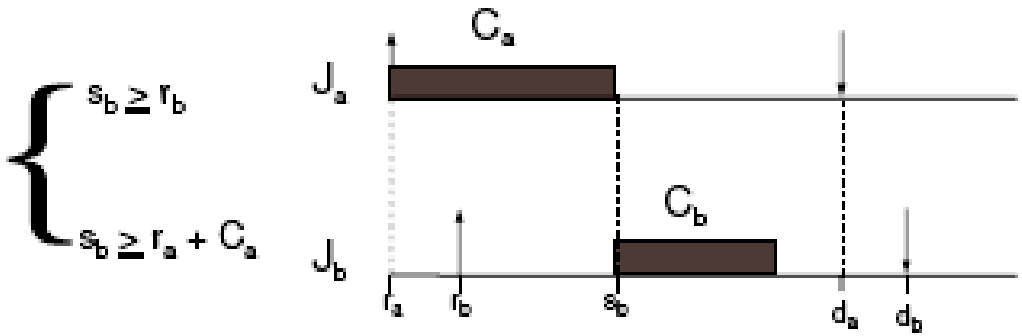


Ako $J_a \rightarrow J_b$ onda vrijeme oslobadjanja J_b može biti zamjenjeno sa :

$$max(r_b, r_a + C_a)$$

DESIGN OF RTOS SYSTEMS

Modifikacija algoritma vremena oslobadjanja



Ako J_a prethodi J_b onda vrijeme oslobadjanja J_b se može zamjeniti sa $\max (r_b , r_a + c_a)$

Novo vrijeme oslobadjanja za J_b je :

$$r^*_b = \max (rb , ra + ca)$$

Nivo kompleksnost algoritma modifikacije vremena oslobadjanja je: $O (n^2)$

DESIGN OF RTOS SYSTEMS

Procedura ovog algoritma je :

1. Za bilo koji početni mod grafa precedencije , postaviti $r^*_i = r_i$

2. Izabrati J_i sa vremenom oslobadjanja koje nije bilo još modificirano ali su svi neposredni taskovi preteče J_h modificirani.
   ako ne postoji niti jedan task : izaći iz uslova

3. $r_i^* = \max [(r_i, \max(r_h^* + c_h: \ J_h \rightarrow J_i)]$

4. Vratiti se na korak 2

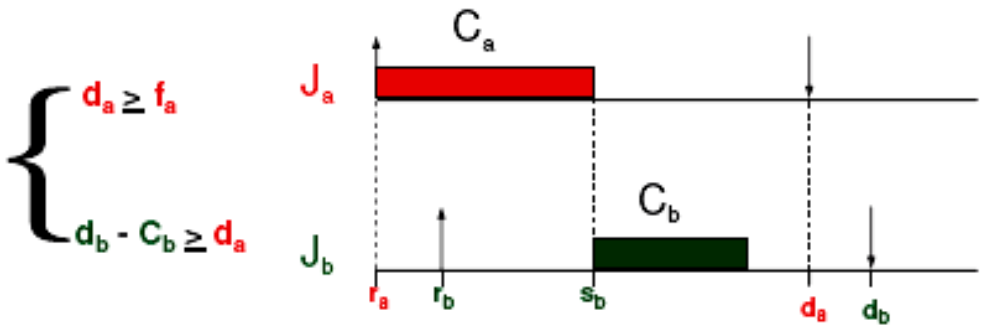
DESIGN OF RTOS SYSTEMS

Modifikacija deadlajna

Neka su data dva taska J_a i J_b , i neka $J_a \rightarrow J_b$

Kod svakog ostvarivog rasporedjivanja, koje zadovoljava uslove ograničenja, slijedeća dva uslova moraju biti ispunjena:

* $f_a \leq d_a$ (J_a mora završiti prije svog deadlajna)
* $f_a \leq d_b - c_b$ (J_a mora završiti prije maksimalnog startnog vremena J_b)

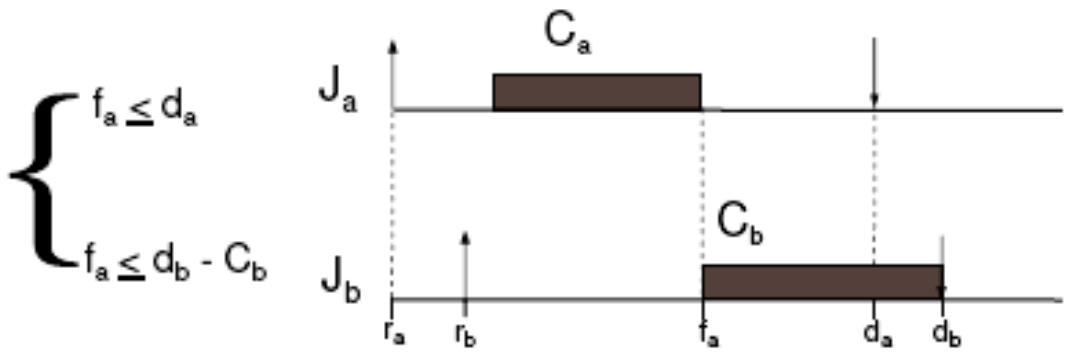


Ako Ja $\rightarrow$ J_b tada deadlajn od J_a se može zamjeniti sa

$$\min (d_a, d_b + c_b)$$

DESIGN OF RTOS SYSTEMS

Modifikacija deadlajna za razmatrani primjer će izgledati kao na narednoj slici:



Ako $J_a \rightarrow J_b$ tada deadlajn za J_a se može zamjeniti sa $\min(d_a, d_b - c_b)$

Novi deadlajn za J_a je sada: $d_a^* = \min (d_a, d_b - c_b)$

Kompleksnost algoritma modifikacije deadlajna je : $O(n^2)$

DESIGN OF RTOS SYSTEMS

Procedura ovog algoritma je slijedeća:

1. Za bilo koji terminalni čvor grafa precedencije, postaviti $d_i^* = d_i$

2. Izabrati J_i sa deadlajnom koji nije još modificiran ali su deadlajni svih neposrednih nasljednika J_k modificirani.
   ako niti jedan ne egzistira: izaći iz konture

3. $d_i^* = \min [d_i, \min (d_k^* - c_k : J_i \rightarrow J_k)]$

4. Vratiti se na korak 2

DESIGN OF RTOS SYSTEMS

Optimalnost EDF sa ograničenjima precedencije

1. Ako je J^* rasporedivo slijedi da je i J rasporedivo
   Ako je J^* rasporedivo slijedi da svi modificirani taskovi startaju nakon r_i^*, i kompletiraju se prije ili u trenutku d_i^*.

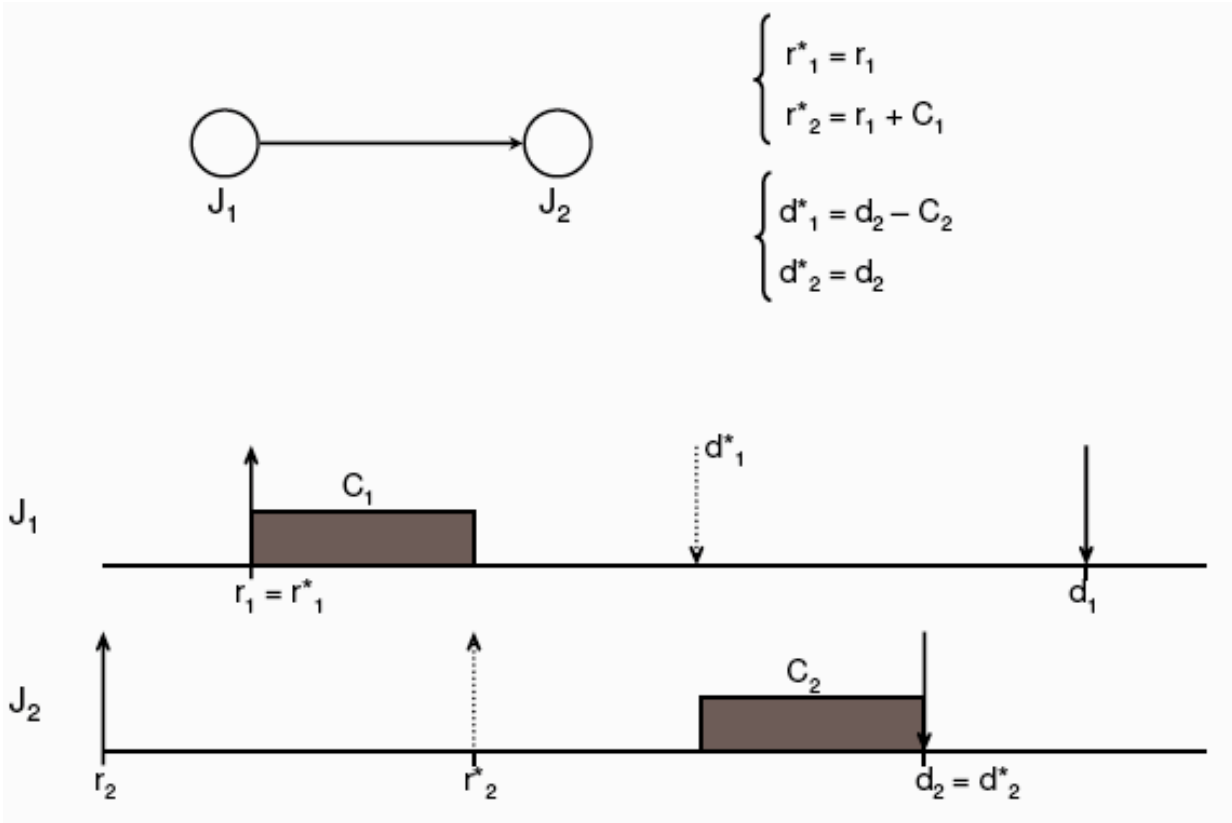
Pošto je $r_i^* \geq r_i$ i $d_i^* \leq d_i$ su rasporedivi , pošto su rasporedivosti za J^* podskup onih za J .

2. Transformacija sačuvava ograničenje precedencije.
   Predpostavimo da $J_i < J_j$ tada slijedi da je $r_i^* \leq r_j^*$ i $d_i^* \leq d_j^*$
   tj. ako je J_i u izvršenju tada svi nasljednici od J_i ne mogu startati ranije nego r_i^* pošto je $r_i^* \leq r_j^*$.
   Ovi taskovi takodjer ne mogu priemptirati J_i pošto je $d_i^* \leq d_j^*$ i , u skladu sa EDF algoritmom, procesor je doznačen spremnom tasku sa najranijim deadlajnom.

Iz 1. i 2. slijedi da su ograničenja tajminga i precedencije koji su specificirani za J su garantirani sa rasporedivošću J^*.

DESIGN OF RTOS SYSTEMS

Optimalnost EDF sa ograničenjima precedencije - primjer



DESIGN OF RTOS SYSTEMS

Poredjenje algoritama rasporedjivanja za aperiodske taskove

	sinhrona aktivacija	priemtivna asinh. Aktivac.	ne-primept Asinh. aktiv
Nezavisni	EDD (Jackson'55) O(n logn) Optimal	EDF (Horn '74) O (n^2) Optimal	Tree search (Bartly '71) O (n n!) Optimal
Ograničenja precedencije	LDF (Lawler '73) $O(n^2)$ Optimal	EDF* (Chetto et al.'90) $O(n^2)$ Optimal	Spring (Stankovic& Ramamritham '87) $O(n^2)$ Heuristic

DESIGN OF RTOS SYSTEMS

Rasporedjivanje periodičnih taskova

Aspekti koji će biti detaljnije razmatrani su:

- rasporedjivanje RM sa dokazima optimalnosti
- deadlajn monotono rasporedjivanje (DM)
- EDF sa deadlajnom ≤ periodu

Hipoteze koje pojednostavljuju analizu su:

A1 Instance taskova τ_i se regularno aktiviraju sa konstantnom brzinom. Interval izmedju dvije uzastopne aktivacije je period T_i taska.

DESIGN OF RTOS SYSTEMS

A2 Sve instance od τi imaju isto vrijeme izvršenja u najgorem slučaju C_i.

A3 Sve instance taskova τi imaju iste relativne deadlajne D_i i vrijedi da je $D_i = T_i$ (tj. deadlajn je jednak periodu).

A4 Svi taskovi u Γ su nezavisni (tj. nema relacija precedencije, nema ograničenja resursa).

A5 Ni jedan task ne može suspendirati samog sebe (napr. Za I/O).

A6 Svi taskovi se oslobadjaju čim prispiju.

A7 Svi dodatci (overheads) zbog RTOS sistema se predpostavljaju da su nula.

DESIGN OF RTOS SYSTEMS

Analiza gore uvedenih predpostavki

A1 i A2 : ovo nisu neka velika ograničenja u praksi (aktivacija nekih rutina u regularnim intervalima)

A3 i A4 : mogu biti i suviše ozbiljna ograničenja za praktične aplikacije .

Karakterizacija periodičnih taskova

U predpostavkama A1, A2, A3 i A4 taskovi τ_i mogu biti u potpunosti karakterizirani sa :

* fazom Φ
* periodom T_i
* vremenom računanja u najgorem slučaju C_i

DESIGN OF RTOS SYSTEMS

Odavde slijedi :

$$\Gamma = [\tau_i(\phi_i, T_i, C_i), i = 1, \ldots, n]$$

(vrijeme oslobadjanja od τ_i)

$$r_{i,k} = \phi_i + (k-1)T_i$$

$$d_{i,k} = \phi_i + kT_i = r_{i,k} + T_i$$

(deadlajn od τ_{ik})

Vrijeme odziva jedne instance $R_{i,k} = f_{i,k} - r_{i,k}$

Kritična instanca taska : je vrijeme oslobadjanja instance taska koje rezultira u najvećem vremenu odziva

Kritična vremenska zona taska : interval izmedju kritične instance i vremena odziva odgovarajuće instance taska

DESIGN OF RTOS SYSTEMS

Relativni jitter oslobadjanja taska je :

maksimalna devijacija startnog vremena izmedju dvije uzastopne instance :

$$RRJ_i = \max_k \; | (s_{i,k} - r_{i,k}) - (s_{i,k-1} - r_{i,k-1}) |$$

Apsolutni jitter oslobadjanja taska je :

$$ARJ_i = \max_k (s_{i,k} - r_{i,k}) - \min_k (s_{i,k} - r_{i,k})$$

Relativni završni jitter taska je:

$$AFJ_i = \max_k (f_{i,k} - r_{i,k}) - \min_k (f_{i,k} - r_{i,k})$$

DESIGN OF RTOS SYSTEMS

Periodični task se naziva ***ostvarivim*** , ako sve njegove instance se završavaju unutar njihovih deadlajna.

Skup taskova Γ se naziva ostvarivim ili rasporedivim ako su svi taskovi u Γ ostvarivi.

Vidjeli smo da je kod periodičnih taskova faktor iskoristivosti procesora bio definisan za n procesa kao:

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i}$$

U se može poboljšati sa :

* povećanjem vremena izračunavanja taskova
* smanjenjem perioda taskova

DESIGN OF RTOS SYSTEMS

do maksimalne vrijednosti do koje je Γ rasporedivo.

Ova granica rasporedivosti zavisi od
* skupa taskova (posebne relacije izmedju perioda taskova)
* algoritma koji se koristi za rasporedjivanje taskova

U_{ub} (Γ, A) = gornja granica faktora iskorištenja procesora za skup taskova Γ pod datim algoritmom A.

$U = U_{ub}$ (Γ, A) $\rightarrow$Γ je puno iskorištenje procesora

Ovo znači da :

Γ je rasporedivo koristeći A ali bilo koje povećanje vremena računanja u nekom od taskova će učiniti taj skup neostvarivim.

Za dati algoritam A , najmanja gornja granica U_{lub} (A) je minimum faktora iskorištenja nad čitavim skupom taskova , koji u potpunosti iskorištava procesor.

DESIGN OF RTOS SYSTEMS

Ova veličina omogućava da se lagano verificira rasporedivost skupa taskova :

$U_{ub}(\Gamma_i) \leq U_{lub}(A) \Rightarrow \Gamma_i$ taskovi su rasporedivi

$U_{ub}(\Gamma_i) > U_{lub}(A) \Rightarrow \Gamma_i$ mogu biti rasporedivi , ako su periodi taskova u pogodnoj medjusobnoj relaciji

$U_{ub}(\Gamma_i) > 1 \Rightarrow \Gamma_i$ taskovi nisu rasporedivi

DESIGN OF RTOS SYSTEMS

Dokaz nerasporedivosti u slučaju U > 1

Neka je $T := T_1 * T_2 * \ldots\ldots * T_n$

$U > 1 \rightarrow U\,T > T$ tj.

$$\sum_{i=1}^{n} \frac{T}{T_i} C_i > T$$

T/T_i predstavlja broj puta task τ_i je izvršen u intervalu T

$(T/T_i)\,C_i$ = totalnom vremenu računanja zahtjevanom od strane taska τ_i u intervalu T.

$$\sum_{i=1}^{n} \frac{T}{T_i} C_i$$

je ukupni zahtjev za vremenom računanja koje zahtjevaju svi taskovi u intervalu T. Jasno da to vrijeme ne može biti veće od T ako hoćemo da Γ bude rasporedivo.

DESIGN OF RTOS SYSTEMS

Deadlajn monotono rasporedjivanje (DM – deadline monotonic)

Predpostavka koju smo napravili u početku i koja se koristila kod RM i EDF algoritma je bila

relativni deadlajn = periodu (A3)

Algoritam rasporedjivanja DM oslabljava ovu predpostavku ali je još uvjek statičan.

Kod DM algoritma , svaki periodični task τ_i se karakteriše sa četiri parametra :

* faza Φ_i
* vrijeme računanja u najgorem slučaju C_i (isto za sve instance)
* relativni deadlajn D_i (isti za sve instance)
* period T_i

DESIGN OF RTOS SYSTEMS

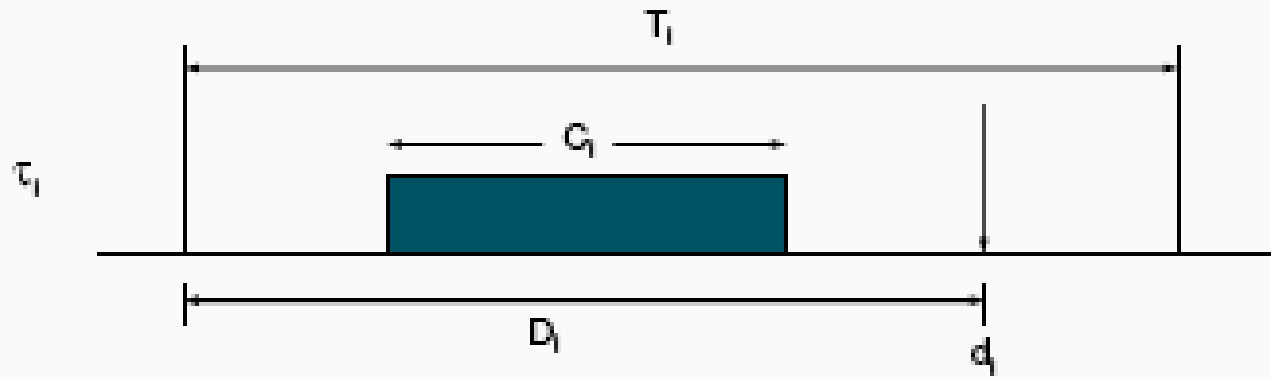
DM rasporedjivanje

Relacije medju parametrima taskova su :

$C_i \leq D_i \leq T_i$

$r_{i,k} = \phi_i + (k-1)\ T_i$

$d_{i,k} = r_{i,k} + D_i$



DESIGN OF RTOS SYSTEMS

DM rasporedjivanje

* DM algoritam doznačuje svakom tasku prioritet koji je inverzno proporcionalan njegovom relativnom deadlajnu.

* U svakom trenutku task sa najkraćim relativnim vremenom se izvršava.

* Pošto je RM DM priemtivni algoritam, task koji se trenutno izvršava može biti priemptiran od novo prispjelog taska ako ima kraći deadlajn.

* DM je optimalni algoritam u smislu optimalnosti RM. Dokaz je sličan onome kod dokazivanja optimalnosti RM algoritma.

DESIGN OF RTOS SYSTEMS

Analiza rasporedivosti DM algoritma

Prvi pristup

Primjeniti RM test rasporedivosti , sa periodima taskova reduciranim na njihove deadlajne.

$$\sum_{i=1}^{n} \frac{C_i}{D_i} \leq n\left(2^{1/n} - 1\right)$$

Medjutim ovo je isuviše pesimistično.

DESIGN OF RTOS SYSTEMS

Opaska :

* Najgori slučaj zahtjeva za procesorom se javlja kada svi taskovi su oslobodjeni simultano (tj. kod njihovih kritičnih instanci)
* Za svaki task τ_i suma njihovih vremena procesiranja i interferencija (priempcija) nametnuta od strane taska većeg nivoa prioriteta mora biti $\leq D_i$.

Predpostavimo da su taskovi uredjeni po rastućim relativnim deadlajnima, t.j $\tau_i < \tau_j \iff D_i < D_j$

Sada je test definisan sa :

$$\forall i : 1 \leq i \leq n : C_i + I_i \leq D_i$$

with

$$I_i = \sum_{j=1}^{i-1} \left\lceil \frac{D_i}{T_j} \right\rceil C_j$$

funkcije limitiranja kao interferirajuća instanca može interferirati i kod početka perioda

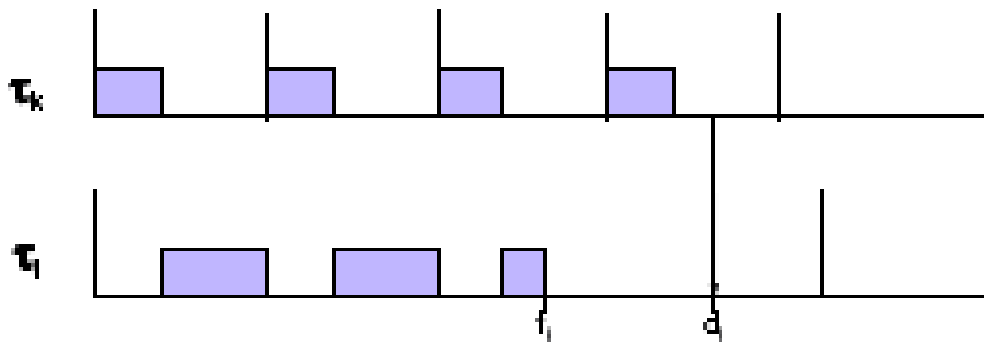
I_i je interferencija na τ_i

DESIGN OF RTOS SYSTEMS

Izvedeni test je dovoljan ali nije neophodan.

Razlog je u tome što je I_i izračunato sa osiguranjem da svaki task većeg prioriteta τ_i inreferira tačno $[D_i/T_i]$ puta sa τ_i.

Stvarna interferencija može biti i manja nego I_i pošto τ_i može da se završi i ranije, kao što je to prikazano na slijedećoj slici:



DESIGN OF RTOS SYSTEMS

Metod Andsley et.al , ze analizu rasporedivosti DM algoritma

Najveće vrijeme odziva taska τ_i je :

$$R_i = C_i + I_i$$
$$I_i = \sum_{j=1}^{i-1} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$
$$R_i = C_i + \sum_{j=1}^{i-1} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

za svako izračunavanje C_i potrebno je izračunavanje sume R_j/T_j i puta

Dakle vidimo da imamo rekurentnu jednačinu u kojoj sa R_i pojavljuje i na lijevoj i desnoj strani jednačine.

Vrijeme odziva u najgorem slučaju za task τ_i je dato sa najmanjom vrijednošću R_i koja zadovoljava gornju jednačinu.

DESIGN OF RTOS SYSTEMS

Iterativni metod

Neka je R^k_i k-ta procjena od R_i.

I^k_i je interferenca za task τ_i na intervalu [R^k_i]

$$I_i^k = \sum_{j=1}^{i-1} \left\lceil \frac{R_i^k}{T_j} \right\rceil C_j$$

Na bazi ovoga se može definisati jedan iterativni proces kako slijedi:

1. $R^0_i = C_i$ je prva tačka u vremenu kada task τ_i bi se mogao kompletirati .
2. Izračunati I^k_i na intervalu [0, R^k_i] koristeći gornji izraz.
3. Ako je $I^k_i + C_i = R^k_i$ tada R^k_i je odziv najgoreg slučaja.

vrijeme za τ_i : $R_i = R^k_i$

inače $R^{k+1}_i = I^k_i + C_i$: goto 2

Ostvarivost taska τ_i je garantovana $\leftrightarrow R_i \leq D_i$

DESIGN OF RTOS SYSTEMS

Algoritam za opisani dokaz je slijedeći :

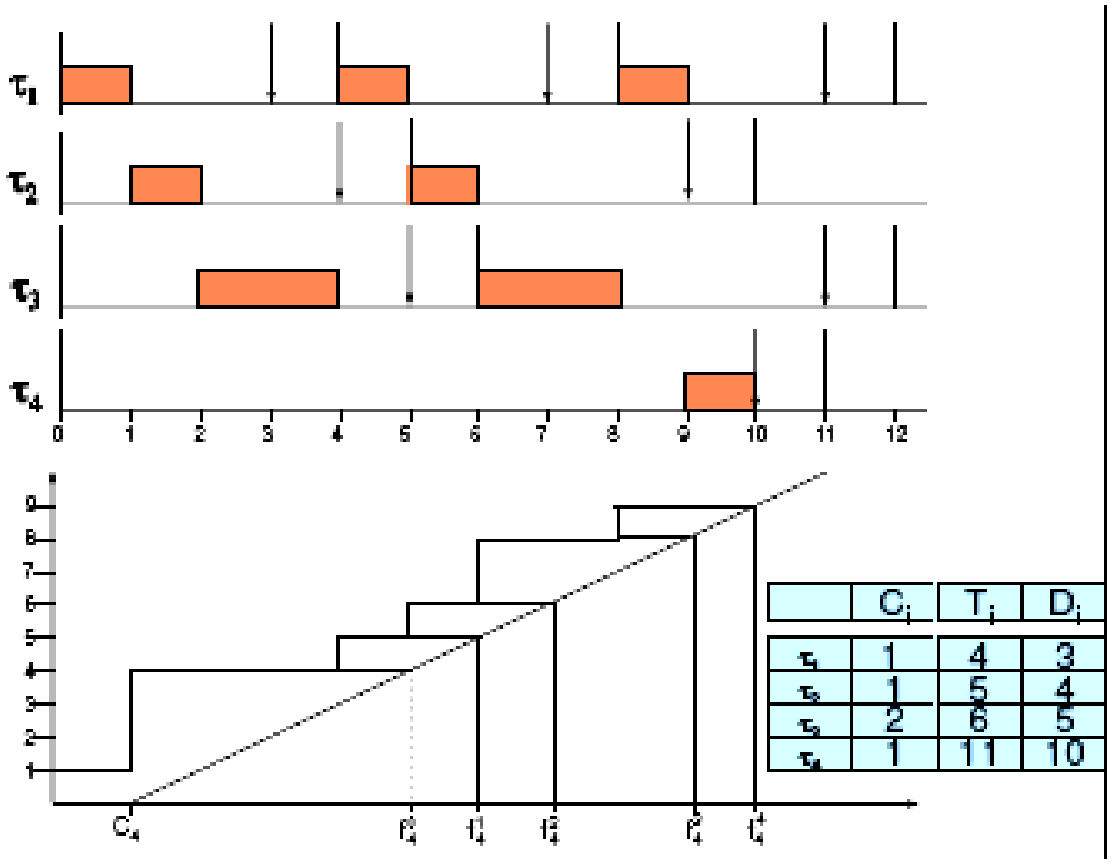
```
DM-guarantee( Γ ) {
    for(each τ_i ∈ Γ ) {
        I=0;
        do {
            R = I + C_i;
            if (R > D_i)      return(UNSCHEDULABLE) ;
            I = Σ_{j=1}^{i-1} ⌈R / T_j⌉ C_j;
        } while (I+C_i>R);
    }
return(SCHEDULABLE);
}
```

DESIGN OF RTOS SYSTEMS

Primjer za DM analizu rasporedivosti

Posmatrajmo slijedeći skup taskova:

	C_i	T_i	D_i
τ_1	1	4	3
τ_2	1	5	4
τ_3	2	6	5
τ_4	1	11	10



Step 0: $R_4^0 = C_4 = 1,$

ali $I^0_4 = 4$ i $I^0_4 + C_4 > R^0_4$
dakle τ_4 ne završava u R^0_4

Step 1: $R_4^1 = I_4^0 + C_4 = 5,$

ali $I^1_4 = 5$ i $I^1_4 + C_4 > R^1_4$
dakle τ_4 ne završava u R^1_4

Step 2: $R_4^2 = I_4^1 + C_4 = 6,$

ali $I^2_4 = 6$ i $I^2_4 + C_4 > R^2_4$
dakle τ_4 ne završava u R^2_4

Step 3: $R_4^3 = I_4^2 + C_4 = 7,$

ali $I^3_4 = 8$ i $I^3_4 + C_4 > R^3_4$
dakle τ_4 ne završava u R^3_4

Step 4: $R_4^4 = I_4^3 + C_4 = 9,$

ali $I^4_4 = 9$ i $I^4_4 + C_4 > R^4_4$
dakle τ_4 ne završava u R^4_4

Step 5: $R_4^5 = I_4^4 + C_4 = 10,$

ali $I^5_4 = 9$ i $I^5_4 + C_4 > R^5_4$
dakle τ_4 završava u $R_4 = 10$

DESIGN OF RTOS SYSTEMS

Pristup tražnje procesora kod EDF algoritma

Tražnja procesora (processor demand) je poopštenje zahtjeva za procesnim vremenom u sistemima upravljanim deadlajnom.

Tražnja procesora taska τ_i u bilo kojem intervalu [t, t+L] je količina procesnog vremena koju zahtjeva τ_i u intervalu [t, t+L] koji mora da se završi prije t+ L.

(u deadlajn baziranim sistemima, to je vrijeme procesiranja koje se zahtjeva u intervalu [t, t+L] koje mora biti izvršeno prije deadlajna ≤ t+ L).

Predpostavimo skup periodičnih taskova sa deadlajn = periodima , koji se pozivaju u trenutku t = 0. Tada kumulativni zahtjev za procesorom u intervalu [0, L] je dat sa:

$$C_p(0,L) = \sum_{i=1}^{n} \left\lfloor \frac{L}{T_i} \right\rfloor C_i$$

DESIGN OF RTOS SYSTEMS

Teorema Jefferey & Stone

Skup periodičnih taskova je rasporediv pomoću EDF algoritma ako i samo ako za sve L , L ≥ 0

$$L \geq \sum_{i=1}^{n} \left\lfloor \frac{L}{T_i} \right\rfloor C_i$$

Što se može interpretirati riječima : Kumulativna tražnja procesora ne prevazilazi njegovo raspoloživo vrijeme

Može se pokazati da je gornja jednačina ekvivalentna klasičnom već ranije izvedenom uslovu :

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1$$

DESIGN OF RTOS SYSTEMS

Sumarne karakteristike algoritama za rasporedjivanje periodičnih taskova

- Ograničenja na nezavisne i priemptabilne periodične taskove.
- Riješenje za fiksno i dinamičko doznačavanje prioriteta.
- EDF algoritam je optimalan medju svim sa doznačavanjem dinamičkog prioriteta.
- U slučaju da su Deadlajni = periodima , test garancije je reda O(n) , koristeći iskoristivost procesora, , što se može primjeniti na RM i EDF algoritme.
- U slučajevima kada su deadlajni < periodima algoritmi za garanciju rasporedivosti su u polinomijalnom vremenu.

  - ako su prioriteti fiksni, koristi se analiza vremena odziva
  - ako su dinamički prioriteti, koristi se zahtjev na procesor.

DESIGN OF RTOS SYSTEMS

	$D_i = T_i$	$D_i \leq T_i$
Statički prioriteti	**RM** **Pristup korištenja procesora** $U \leq n(2^{1/n} - 1)$	**DM** **Pristup vremena odziva** $\forall i \; R_i = C_i + \sum_{j=1}^{i-1} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \leq D_i$
Dinamički prioriteti	**EDF** **Pristup iskoristivost procesora** $U \leq 1$	**EDF*** **Pristup tražnje za procesorom** $\forall L > 0 \quad L \geq \sum_{i=1}^{n} \left(\left\lfloor \frac{L - D_i}{T_i} \right\rfloor + 1 \right) C_i$