Tosiaikajärjestelmät – Luento 5: Resurssien hallinta ja prioriteetit

Tiina Niklander

Jaetut resurssit

- Useat tapahtumat jakavat ohjelma-/laitteisto-olioita, joissa keskinäinen poissulkeminen on välttämätöntä.
- Ratkaisuja:
  - Ajonaikainen resurssien käyttösäännöstö, jolla konfliktit ratkaistaan ajonaikaisesti käyttäen dynaamisia tapahtumien prioriteettien muutoksia.
  - Etukäteen muodostettu resurssien ajoitus, joka muodostaa ei päällekkäisen tapahtumien suoritusikkunan estäen ajonaikaiset konfliktit.

Jaettujen resurssien vuorottajan tehokkuus ?

- ”When there are mutual exclusion constraints in a system, *it is impossible to find* an optimal on-line scheduling algorithm (unless it is clairvoyant).” (Mok, 1983)
- ”The problem of deciding feasibility for set of periodic tasks which use semaphores to enforce mutual exlusion is *NP-hard*.” (Mok, 1983).

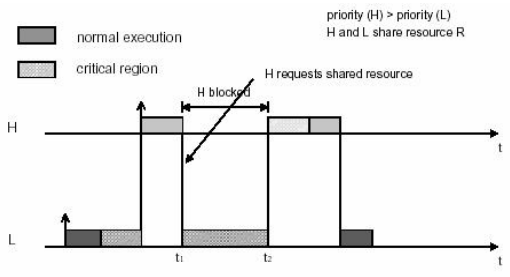
Mitä voi mennä pieleen ?

- Resurssit, kriittiset alueet, odotus
- Prioriteetin kääntyminen
  - korkeamman prioriteetin työ joutuu odottamaan alemman prioriteetin työtä
- Lukkiumat (deadlocks)
  - kaksi tehtävää odottaa toisen varaamaa resurssia
- Irroittamattomat kriittiset alueet
  - resurssin käyttäjää ei voi siirtää pois suorittimelta

Resurssikilpailu ja estyminen

- Resurssien käyttö usein poissulkevaa, eli vain varaaja saa käyttää, muut odottavat
- Yleensä ensimmäisenä pyytänyt saa resurssin
- Merkitään tehtävien kriittiset alueet seuraavasti [R,η;e], missä R on resurssi, η on haluttu lukumäärä ja e on kriittisen alueen kesto
- Esim: [X,1; 5 [Y,2; 2]][Z,3; 1]

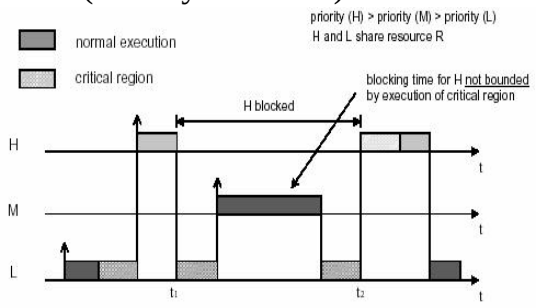
Odotusongelma (Blocking problem)



Estyminen ja odotuksen kesto

- Ilman kilpailua korkeamman prioriteetin työ ohittaa alemman prioriteetin työn, **mutta**
- Jos alemmalla on hallussaan jaettu resurssi on korkeamman prioriteetin työn odotettava
- Kuinka kauan?
  - rajoitettu aika – vain resurssi käytön verran
  - rajattomasti – aikaa ei voida ennustaa
- Ns. prioriteetin kääntyminen

Prioriteetin kääntyminen (Priority inversion)



Prioriteetin kääntymisen välttäminen

- Irroittamattomat kriittiset alueet
  - Luovat tarpeetonta odotusta.
  - Käyttökelpoisia **lyhyille** kriittisille alueille.
- Varausprotokollia kriittiselle alueelle
  - Prioriteetin perintä (Priority Inheritance Protocol).
  - Prioriteettikatto (Priority Ceiling Protocol).

Irroittamattomat kriittiset alueet (nonpreemptive critical sections, NPCS)

- Suoritetaan vain sitä tehtävää, joka käyttää jotain jaettua resurssia, muut odottavat
- Korkeimman prioriteetin maksimiodotus kestää muiden pisimmän yhtenäisen kriittisen alueen suorituksen ajan

$$b_i(rc) = \max_{i+1 \le k \le n} (c_k)$$

Irroittamattomat kriittiset alueet

- Hyödyt
  - yksinkertainen – ei rinnakkaisuuden hallintaa
  - ei lukkiudu
- Haitat
  - Korkeimman prioriteetin työ saattaa joutua odottamaan myös sellaista alemman prioriteetin työtä, joka ei kilpaile sen kanssa samoista resursseista

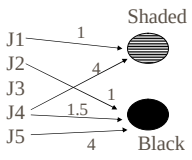
Esimerkkien työkuorma

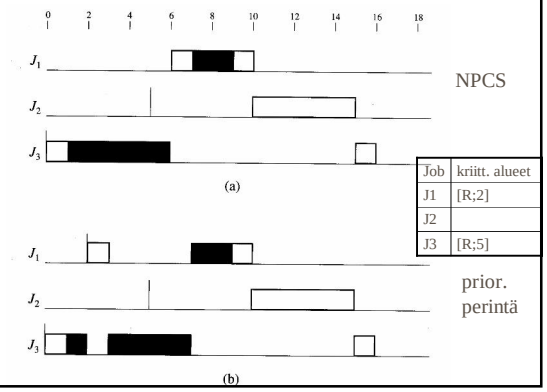
Job	ri	ei	πi	kriitt. alueet
J1	7	3	1	[Shaded;1]
J2	5	3	2	[Black;1]
J3	4	2	3	
J4	2	6	4	[Shaded;4 [Black;1.5]]
J5	0	6	5	[Black;4]

r – aloitushetki

e – suoritusaika

π – prioriteetti





NPCS

Job	kriitt. alueet
J1	[R;2]
J2	
J3	[R;5]

prior. perintä

Liu kuva 8-7

Prioriteetin perintä

- Idea: Jos tapahtuma T estää (blocks) yhden tai useamman korkeampi prioriteettisen tapahtuman etenemisen, tapahtuma T perii väliaikaisesti korkeimman estetyn tapahtuman prioriteetin.
- Hyödyt
  - Estää prioriteetiltaan keskitasoa olevan tapahtuman keskeyttämästä tapahtumaa T.
- Haitat
  - Prioriteetin perintä voi aiheuttaa lukkiuman
  - Linkitetty odotus (chained blocking)

Prioriteetin perinnän säännöt

- Vuorotus
  - vuorotetaan irroittavasti ja prioriteetti-pohjaisesti hetkellisen prioriteetin perusteella
- Varaus
  - Työ saa resurssin R jos R on vapaa, muuten joutuu odottamaan
- Prioriteetin perintä
  - Kun työ J_h joutuu odottamaan resurssia R, resurssin varannut työ J_l perii J_h:n prioriteetin $\pi(t)$ kunnes vapauttaa R:n. Vapautuksen yhteydessä J_i:n prioriteetiksi palautetaan varaushetken prioriteetti $\pi(t')$

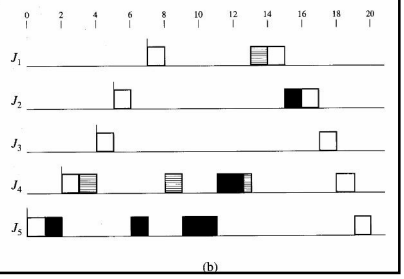
Prioriteetin perintä

Job	ri	ei	πi
J1	7	3	1
J2	5	3	2
J3	4	2	3
J4	2	6	4
J5	0	6	5

Job	kriitt. alueet
J1	[Sh;1]
J2	[Bl;1]
J3	
J4	[Sh;4 [Bl;1.5]]
J5	[Bl;4]

Job	r_i	e_i	π_i	Critical Sections
J_1	7	3	1	[*Shaded*; 1]
J_2	5	3	2	[*Black*; 1]
J_3	4	2	3	
J_4	2	6	4	[*Shaded*; 4 [*Black*; 1.5]]
J_5	0	6	5	[*Black*; 4]

(a)

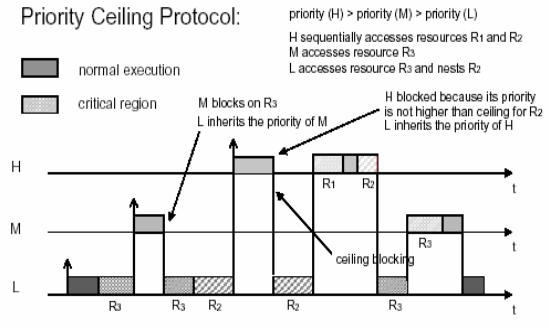


Liu Kuva 8-8

Prioriteettikatto

- Idea
  - Jokaiselle resurssille asetetaan prioriteettikatto $\Pi(R_i)$ yhtä suureksi kuin sen korkeimman tapahtuman prioriteetti, joka tarvitsee tätä resurssia ja siis saa lukita resurssin.
  - Tapahtuma T saa siirtyä kriittiselle alueelle ja varata resurssin vain, jos sen prioriteetti on korkeampi kuin kaikkien muiden samanaikaisten tapahtumien sillä hetkellä varaamien resurssien prioriteettikatot $\Pi(t)$.
  - Jos tapahtuma T estää yhden tai useamman korkeampi prioriteettisen tapahtuman, se väliaikaisesti perii korkeimman estetyn tapahtuman prioriteetin.

Prioriteettikatto



Prioriteettikatto – säännöt

- Vuorotus
  - vuorotetaan irroittavasti ja prioriteetti-pohjaisesti hetkellisen prioriteetin perusteella
- Varaus
  - Työ J saa resurssin R vain jos R on vapaa ja J:n prioriteetti on korkeampi kuin $\Pi(t)$, muuten joutuu odottamaan
- Prioriteetin perintä
  - kuten aiemmin, J perii korkeimman odottajan prioriteetin kunnes J on vapauttanut kaikki resurssit, joiden prioriteettikatto on sama tai korkeampi kuin peritty prioriteetti.

Esimerkki: prioriteetin katto

Job	ri	ei	πi
J1	7	3	1
J2	5	3	2
J3	4	2	3
J4	2	6	4
J5	0	6	5

Job	kriitt. alueet
J1	[Sh;1]
J2	[Bl;1]
J3	
J4	[Sh;4 [Bl;1.5]]
J5	[Bl;4]

Liu kuva 8-10

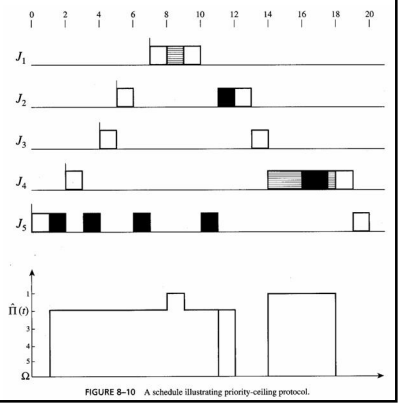


FIGURE 8–10 A schedule illustrating priority-ceiling protocol.

Prioriteettikatto

- Hyödyt
  - Lukkiutumaton: prioriteettikatot estävät lukkiumat
  - Ei linkitettyä odotusta: korkeimman prioriteetin tapahtuma voi estyä korkeintaan yhden kriittisen alueen keston ajaksi.
- Haitat
  - ?

Lukkiutumisten esto?

- Resursseilla ja prosesseilla sama 'prioriteettijärjestys'
- Ajanhetkellä t, työn J prioriteetti $\pi(t)$ voi olla aidosti korkeampi kuin senhetkinen prioriteettikatto $\Pi(t)$ vain kun
  - J ei aio varata ja käyttää mitään tällä hetkellä varattuja resursseja ja
  - J:tä korkeampi prioriteettiset työt eivät myöskään aio käyttää tällä hetkellä varattuna olevia resursseja
- Näistä seuraa, että J ei voi joutua lukkiumaan minkään resursseja jo varanneen työn kanssa

Kattoprioriteetti

- Voitaisiinko prioriteetin katto –menetelmää yksinkertaistaa?
- Entä jos tehtävillä on yhteinen suoritusaikainen pino ja vain päällimmäinen on suorituksessa?
  - Työn irroittava (preempt) uusi työ sijoittuu pinossa irroitetun työn päälle
- Kattoprioriteetti (tai pinoperustainen kattoprior.)
  - Uusi menetelmä, joka hyödyntää tätä ideaa
  - Rajoitus: työt eivät saa 'keskeyttää' itseään

Pinoperustainen kattoprioriteetti (Stack-Based Priority-Ceiling protocol)

- Nykyhetken kattoarvon päivitys
  - Arvo päivitetään aina resurssin varauksen ja vapautuksen yhteydessä
  - Jos kaikki resurssit vapaat, niin arvo on Ω.
- Vuorotussääntö:
  - Suoritukseen saapuvan työn täytyy odottaa, kunnes sen prioriteetti on kattoarvoa suurempi
  - Työt vuorotetaan käyttäen irroittavaa prioriteettiperustaista menetelmää
- Varaussääntö:
  - Suorituksessa oleva työ voi aina varata pyytämänsä resurssin

Esimerkki

Job	ri	ei	πi
J1	7	3	1
J2	*4.8*	3	2
J3	4	2	3
J4	2	6	4
J5	0	6	5

Job	kriitt. alueet
J1	[Sh;1]
J2	[Bl;*1.2*]
J3	
J4	[Sh;4 [Bl;1.5]]
J5	[Bl;4]

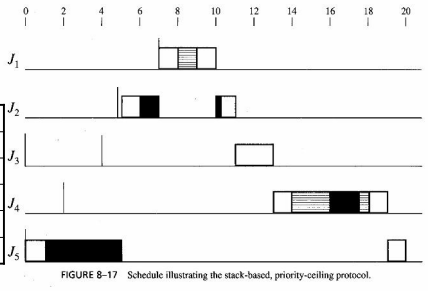


FIGURE 8–17 Schedule illustrating the stack-based, priority-ceiling protocol.

Liu kuva 8-17

Kattoprioriteetti (Ceiling-Priority)

- Toimii kuten pinoperustainen kattoprior.
- Vuorotussääntö
  - Työtä suoritetaan sen alkuperäisellä prioriteetilla, jos työ ei ole varannut mitään resurssia. Saman prion työt FIFO-periaatteella
  - Minkätahansa resurssin varanneen työn tilap. prioriteetti on yhtä suuri kuin sen varaamien resurssin suurin kattoprioriteetti
- Varaussääntö
  - Suorituksessa oleva työ voi aina varata pyytämänsä resurssin

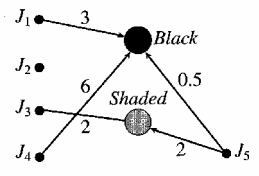
Irroitusten katto

- Aiemmat menetelmät (prioriteetin perintä, prioriteetin katto ja kattoprioriteetti) soveltuvat parhaiten kiinteille prioriteeteille
- Käytetään irroittamisia prioriteettien sijaan
- Taustana
  - Korkeamman prioriteetin työ voi haluta matalamman varaamaa resurssia vain, jos se saa irroittaa suorituksessa olevan matalamman
  - Tietyille dynaamista prioriteettia käyttäville järjestelmille (esim. takarajoihin perustuvat) on mahdollista etukäteen selvittää jaksollisten töiden mahdolliset irroitustilanteet.

Irroitustaso (preemption level)

- Merkitään työn J_i irroitustasoa ψ_i
- Oikeellisuusehto:
  - Jos π_i on korkeampi kuin π_k ja $r_i > r_k$, niin ψ_i on korkeampi kuin ψ_k
- Tavoitteena on siis saada järjestys töiden irroitustasojen välille siten, että korkeampiprioriteettinen työ ei voi irroittaa sellaista työtä, jolla on varattuna korkeamman haluama resurssi

Irroitustaso – esimerkki



Jobs	r_i	e_i	π_i	ψ_i
J_1	5	5	1	1
J_2	7	3	2	1
J_3	4	4	3	2
J_4	0	9	4	3
J_5	3	4	5	3

FIGURE 8–19 A schedule according to the preeption-ceiling protocol.

Esim. työ J1 varaa resurssin Black 3 aikayksikön ajaksi

Liu kuva 8-19

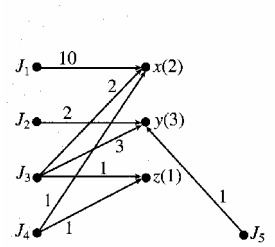
Irroitusten katto

- Vuorotus ja perintäsäännöt
  - kuten prioriteetin katto menetelmässä
- Varaussäännöt
  - Pyydetty resurssi on varattuna, → työ odottaa
  - Resurssi on vapaa
    - Työ saa resurssin, jos työn irroitustaso $\psi(t)$ on korkeampi kuin senhetkinen irroituskatto $\Psi(t)$
    - Jos työn irroitustaso ei ole korkeampi, mutta työllä on jo hallussaan resurssi, jonka irroituskatto on juuri tuo $\Psi(t)$, niin työ saa resurssin, muuten joutuu odottamaan

Useita resurssiyksiköitä

- Entä jos samanlaisia resursseja on useita?
  - Laaditaan resurssille kattoarvojen vektori
  - Merkitään $\Pi(R_i,k)$, missä $k \leq \upsilon_i$, ja k kuvaa vapaiden resurssiyksiköiden määrää
- Yhden resurssin säännöstöt pitää vain muokata ottamaan huomioon moniresurssin kattoarvo

Esimerkki – useita resurssiyksiköitä



Resources	$x(2)$	$y(3)$	$z(1)$
Units Required by J_1	1	0	0
J_2	0	2	0
J_3	2	3	1
J_4	1	0	1
J_5	0	1	0
$\Pi(*, 0)$	π_1	π_2	π_3
$\Pi(*, 1)$	π_3	π_2	Ω
$\Pi(*, 2)$	Ω	π_3	Ω
$\Pi(*, 3)$	Ω	Ω	Ω

FIGURE 8–21 Example of priority ceilings of multiple-unit resources.

Liu kuva 8-21

(nyt kaaren numero kuvaa varattavien yksiköiden määrää)

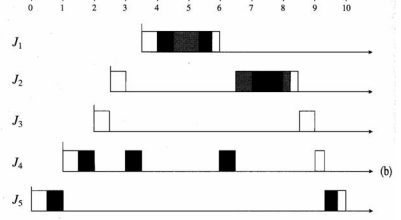
Prioriteetin perintä?

- Miten päätetään mikä useista resurssiyksiköitä varaavista alemmista töistä saa periä korkeamman prioriteetin?
- Perintäsääntö:
  - Prioriteetin perii korkeimman prioriteettikaton töistä se, jonka oma alkuperäinen prioriteetti on korkein

Esimerkki – useita resurssiyksiköitä

	no. of units	units required					$\Pi(*, k), k =$					
		J_1	J_2	J_3	J_4	J_5	0	1	2	3	4	5
Black	5	2	4	0	1	1	1	1	2	2	Ω	Ω
Shaded	1	1	1	0	0	1	1	Ω	Ω	Ω	Ω	Ω

(a)



(b)

Liu kuva 8-22

Käsitellyt menetelmät

- Irroittamaton kriittinen alue (nonpreemptive critical section)
- Prioriteetin perintä
- Prioriteetin katto
- Kattoprioriteetti
- Irroitusten katto

Lisämateriaali (luvattu laskareissa ke 5.4.)

- Estymisajan laskenta resurssikilpailun ja prioriteettien perusteella (kirjan luku 8.5.4)
- Ajoitettavuusanalyysi
  - Kiinteät prioriteetit (RM): Aikavaativuus-analyysi ja estymisaika
    - Perusversio ilman estymistä luku 6.6.2
    - Perusversio ja estyminen luku 6.8.1
  - Dynaamiset prioriteetit (EDF)
    - Ajoitettavuusanalyysi käyttöasteen perusteella
    - Tähän lisätään estymisen vaikutus (luku 6.8.1 kaava 6.21)

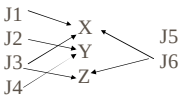
Estyminen ja prioriteetin perintä sekä prioriteettikatto

- Vain alemman prioriteetin työ voi estää resurssikilpailun kautta. Siksi taulukosta Es ei tarvitse täyttää vasenta alakolmiota.
- Työn maksimiestymisaika on sitä vastaavan rivin maksimiarvo.
- Estävät (alempi prio) työt muodostavat sarakkeet. Työ voi
  - Estää suoraan – kun se kilpailee samasta resurssista
  - Estää prioriteetin perinnällä – kun se varauksen kautta perii ylemmän prioriteetin
- Taulukon täyttäminen:
  - Vasen osa (Suoraan) – nämä poimitaan tiedoista
  - Oikea osa (Perinnällä) – voidaan päätellä vasemman sarakkeen perusteella

Estyminen ja prioriteetin perintä sekä prioriteettikatto

Job	kriitt. alueet
J1	[X;10]
J2	[Y;1]
J3	[X;6 [Y;2]]
J4	[Y;5]
J5	
J6	[Z;4][X;2]

Es	Estää suoraan					Prioriteetin perinnällä				
	J2	J3	J4	J5	J6	J2	J3	J4	J5	J6
J1		6			2					
J2	*		5			*	6			2
J3		*			4		*	5		2
J4			*					*		4
J5				*					*	4



HUOM: J3 käyttää X ja Y resursseja yhtä aikaa, kun J6 käyttää Z ja X resursseja peräkkäin. (Katso hakasulkuja)

Aikavaativuusanalyysi (RM) ja estymisaika

- Kuhunkin työhön kohdistuva estymisaika vaikuttaa vain siihen itseensä. Se tulee siis työn oman suoritusajan lisäksi tuohon kaavaan.
- Korkeampi prioriteettiset työt keskeyttävä työn vain oman suorituksensa ajaksi. Siinä ei enää tarkastella niihin kohdistuvaa estymistä

HUOM. Työt jakson-pituuden mukaan prioriteettijärj.

$$w_i(t) = e_i + b_i + \sum_{k=1}^{i-1} \left\lceil \frac{t}{p_k} \right\rceil e_k, \quad \text{aikavälillä } 0 < t \le \min(D_i, p_i)$$

Ajoitettavuusanalyysi (EDF) ja estymisaika

- EDF:llä ajoitettaville töille estyminen määrätään samoin kuin kiinteän prioriteetin tehtäville, mutta prioriteettina käytetään suhteellista aikarajaa Di.
- Kirjan teoreema 6.18: EDF skeduloinnissa työ J_k (suht. aikaraja (D_k) voi estää työn J_i (suht. aikaraja D_i), vain jos $D_k > D_i$
  - J_k voi estää vain, jos prioriteetti on pienempi, eli kun $d_k > d_i$
  - Estääkseen J_k:n pitää olla jo suorituksessa eli $r_k < r_i$
  - Molemmat epäyhtälöt voivat päteä samanaikaisesti vain, kun $D_k > D_i$

Ajoitettavuusanalyysi (EDF) ja estymisaika

- Koko tehtäväjoukko on ajoitettavissa, jos minkään tehtävän estymisaika ei aiheuta kokonaiskuorman päällä ylikuormitusta!
- Estäjänä voivat toimia vain työt, joiden suhteellinen aikaraja on suurempi.

$$\forall i \; U + \frac{b_i}{\min(D_i, p_i)} \le 1, \text{ missä } U = \sum_{k=1}^{n} \frac{e_k}{\min(D_k, p_k)}$$