

# Tosiaikajärjestelmät – Luento 5: Resurssien hallinta ja prioriteetit

Tiina Niklander

## Jaetut resurssit

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

## Jaettujen resurssien vuorottajan tehokkuus ?

- n ”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)
- n ”The problem of deciding feasibility for set of periodic tasks which use semaphores to enforce mutual exclusion is *NP-hard*.” (Mok, 1983).

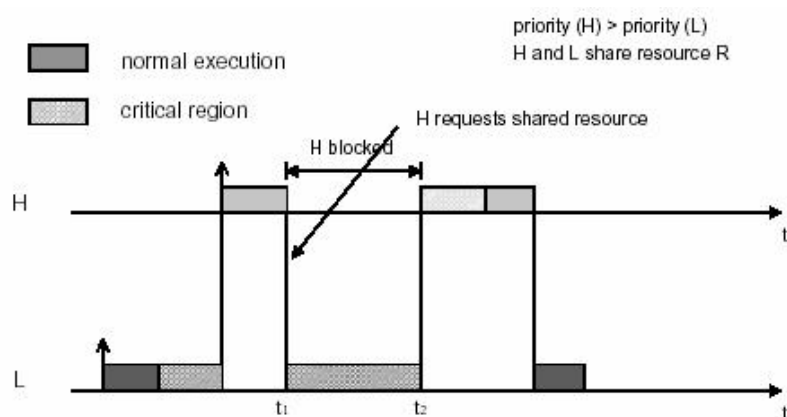
## Mitä voi mennä pieleen ?

- n Resurssit, kriittiset alueet, odotus
- n Prioriteetin kääntyminen
  - n korkeamman prioriteetin työ joutuu odottamaan alemman prioriteetin työtä
- n Lukkiumat (deadlocks)
  - n kaksi tehtävää odottaa toisen varaamaa resurssia
- n Irroittamattomat kriittiset alueet
  - n 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  $[R, \eta; e]$ , missä  $R$  on resurssi,  $\eta$  on haluttu lukumäärä ja  $e$  on kriittisen alueen kesto
- Esim:  $[X, 1; 5 [Y, 2; 2]][Z, 3; 1]$

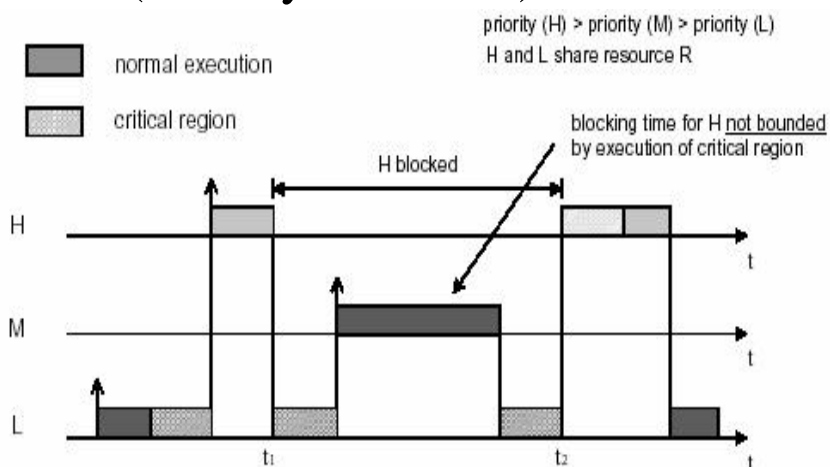
## Odotusongelma (Blocking problem)



## Estyminen ja odotuksen kesto

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

## Prioriteetin kääntyminen (Priority inversion)



## Prioriteetin kääntymisen välttäminen

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

## Irroittamattomat kriittiset alueet (nonpreemptive critical sections, NPCS)

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

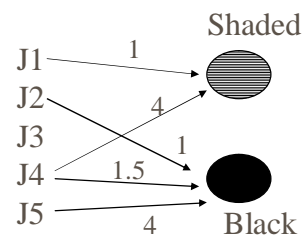
$$b_i(rc) = \max_{i+1 \leq k \leq n} (c_k)$$

## Irroittamattomat kriittiset alueet

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

## Esimerkkien työkuorma

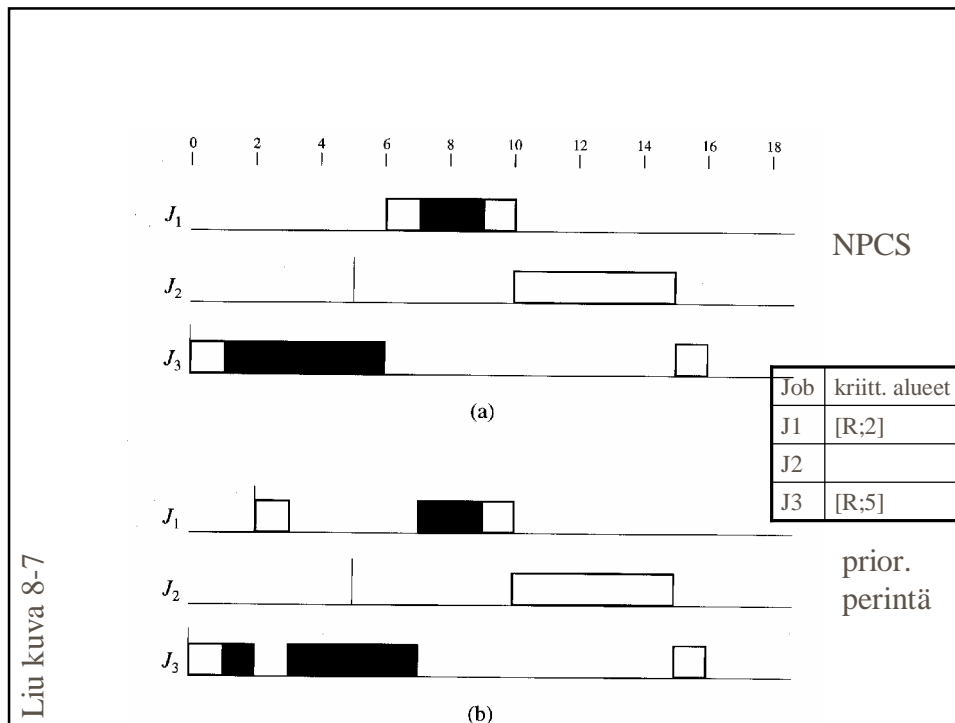
Job	$r_i$	$e_i$	$\pi_i$	kriitt. alueet
J1	7	3	1	[Shaded;1]
J2	5	3	2	[Black;1]
J3	4	2	3	
J4	2	6	4	[Shaded;4 [Black;1.5] ]
J5	0	6	5	[Black;4]



$r$  – aloitushetki

$e$  – suoritusaika

$\pi$  – prioriteetti



## Prioriteetin perintä

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

# Prioriteetin perinnän säännöt

- n Vuorotus
  - n vuorotetaan irrottavasti ja prioriteetti-pohjaisesti hetkellisen prioriteetin perusteella
- n Varaus
  - n Työ saa resurssin R jos R on vapaa, muuten joutuu odottamaan
- n Prioriteetin perintä
  - n 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  $J_l$ :n prioriteetiksi palautetaan varaushetken prioriteetti  $\pi(t')$

## Prioriteetin perintä

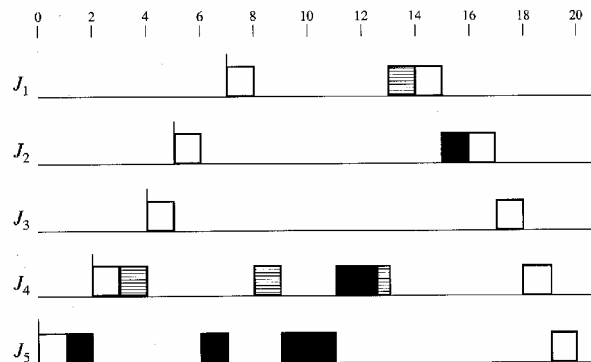
Job	$r_i$	$e_i$	$\pi_i$
J1	7	3	1
J2	5	3	2
J3	4	2	3
J4	2	6	4
J5	0	6	5

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

Liu Kuva 8-8

Job	$r_i$	$e_i$	$\pi_i$	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)



(b)



# Prioriteettikatto

## n Idea

- n Jokaiselle resurssille asetetaan prioriteettikatto  $\Pi(R_i)$  yhtä suureksi kuin sen korkeimman tapahtuman prioriteetti, joka tarvitsee tätä resurssia ja siis saa lukita resurssin.
- n 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  $\Pi(t)$ .
- n Jos tapahtuma T estää yhden tai useamman korkeampi prioriteettisen tapahtuman, se väliaikaisesti perii korkeimman estetyn tapahtuman prioriteetin.

# Prioriteettikatto

## Priority Ceiling Protocol:

priority (H) > priority (M) > priority (L)

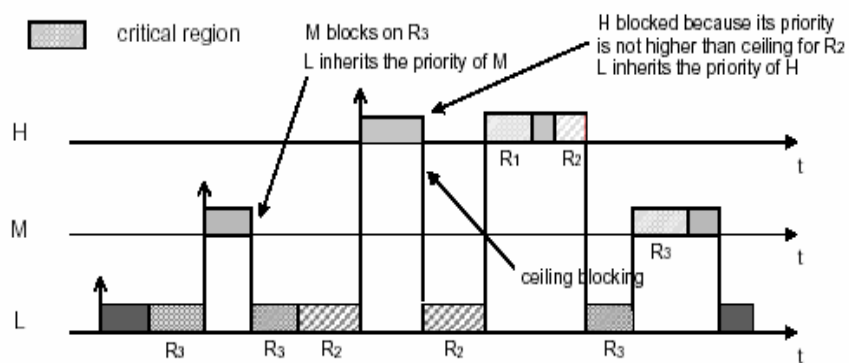
H sequentially accesses resources R1 and R2

M accesses resource R3

L accesses resource R3 and nests R2

■ normal execution

■ critical region



## Prioriteettikatto – säännöt

- n Vuorotus
  - n vuorotetaan irrottavasti ja prioriteetti-pohjaisesti hetkellisen prioriteetin perusteella
- n Varaus
  - n Työ J saa resurssin R vain jos R on vapaa ja J:n prioriteetti on korkeampi kuin  $\Pi(t)$ , muuten joutuu odottamaan
- n Prioriteetin perintä
  - n 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	$r_i$	$e_i$	$\pi_i$
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

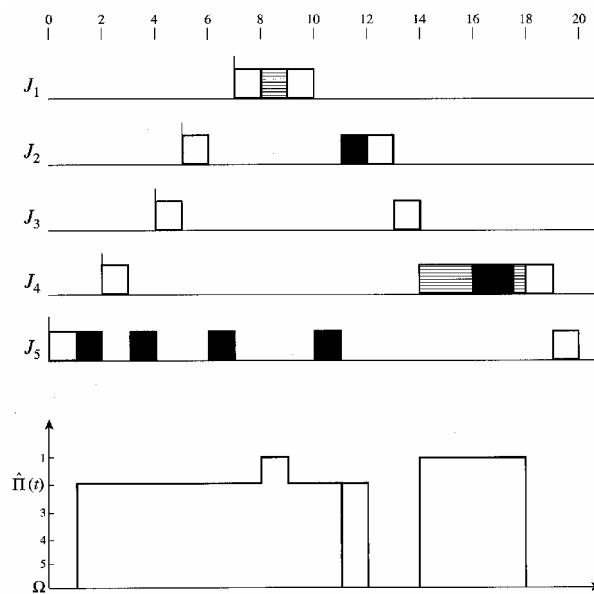


FIGURE 8-10 A schedule illustrating priority-ceiling protocol.

## Prioriteettikatto

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

## Lukkiutumisten esto?

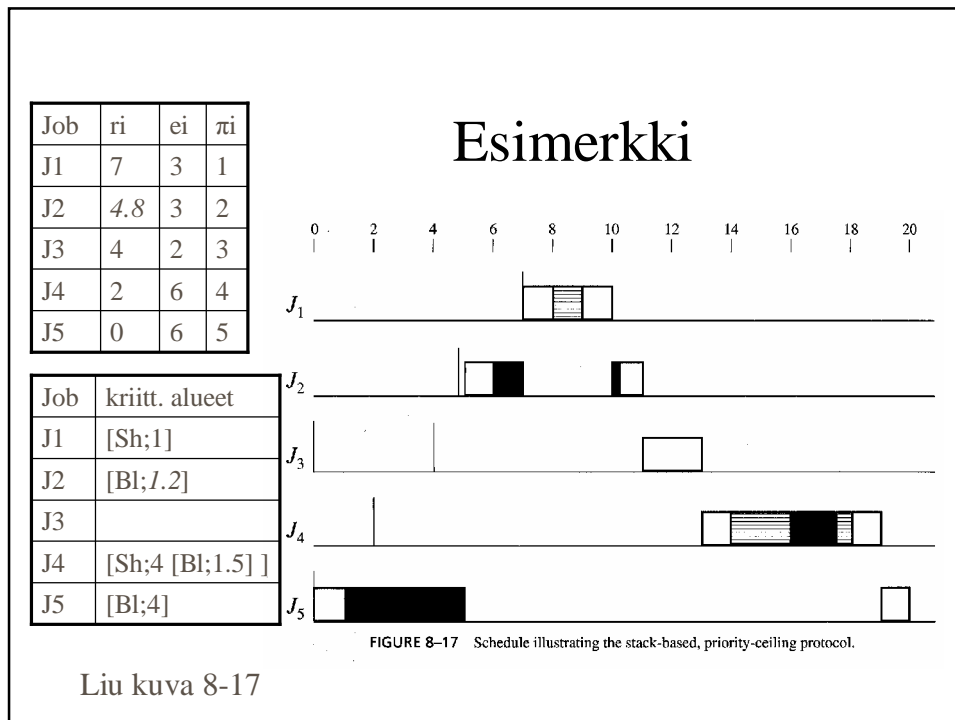
- n Resursseilla ja prosesseilla sama 'prioriteettijärjestys'
- n Ajanhetkellä  $t$ , työn  $J$  prioriteetti  $\pi(t)$  voi olla aidosti korkeampi kuin senhetkinen prioriteettikatto  $\Pi(t)$  vain kun
  - n  $J$  ei aio varata ja käyttää mitään tällä hetkellä varattuja resursseja ja
  - n  $J$ :tä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

## Kattoprioriteetti

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

## Pinoperustainen kattoprioriteetti (Stack-Based Priority-Ceiling protocol)

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



## Kattoprioriteetti (Ceiling-Priority)

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

## Irroitusten katto

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

## Irroitustaso (preemption level)

- n Merkitään työn  $J_i$  irroitustasoa  $\psi_i$
- n Oikeellisuusehto:
  - n Jos  $\pi_i$  on korkeampi kuin  $\pi_k$  ja  $r_i > r_k$ , niin  $\psi_i$  on korkeampi kuin  $\psi_k$
- n 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

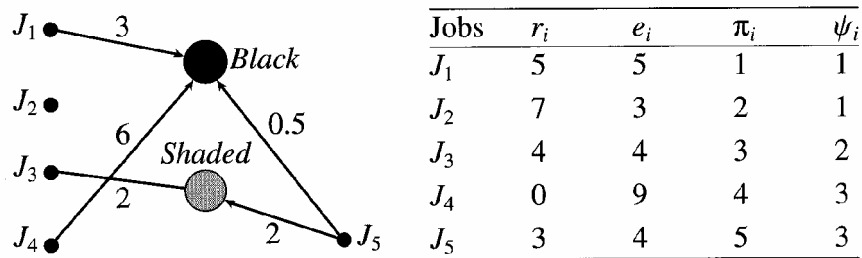


FIGURE 8-19 A schedule according to the preemption-ceiling protocol.

Esim. työ  $J_1$  varaa  
resurssin Black  
3 aikayksikön ajaksi

Liu kuva 8-19

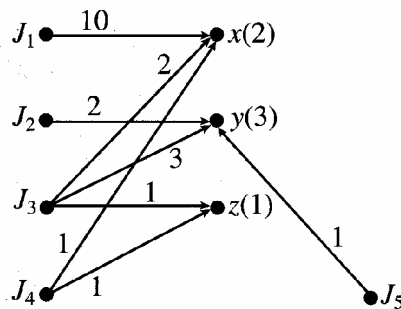
## Irroitusten katto

- n Vuorotus ja perintäsäännöt
  - n kuten prioriteetin katto menetelmässä
- n Varaussäännöt
  - n Pyydetty resurssi on varattuna,  $\rightarrow$  työ odottaa
  - n Resurssi on vapaa
    - n Työ saa resurssin, jos työn irroitustaso  $\psi(t)$  on korkeampi kuin senhetkinen irroituskatto  $\Psi(t)$
    - n 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

- n Entä jos samanlaisia resursseja on useita?
  - n Laaditaan resurssille kattoarvojen vektori
  - n Merkitään  $\Pi(R_i, k)$ , missä  $k \leq v_i$ , ja  $k$  kuvaa vapaiden resurssiyksiköiden määrää
- n Yhden resurssin säännöt 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
$\Pi(*, 0)$		$\pi_1$	$\pi_2$	$\pi_3$
$\Pi(*, 1)$		$\pi_3$	$\pi_2$	$\Omega$
$\Pi(*, 2)$		$\Omega$	$\pi_3$	$\Omega$
$\Pi(*, 3)$		$\Omega$	$\Omega$	$\Omega$

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

(nyt kaaren numero kuvaa varattavien yksiköiden määrää)

Liu kuva 8-21

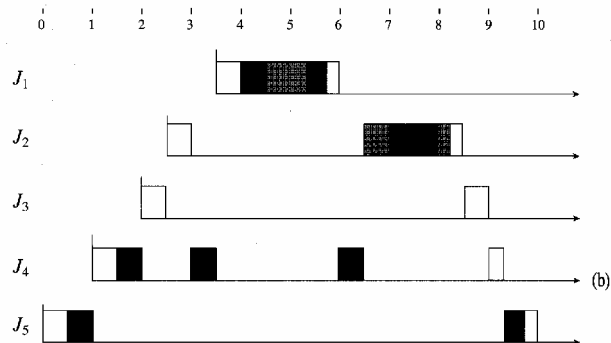


## Prioriteetin perintä?

- n Miten päätetään mikä useista resurssiyksiköitä varaavista alemmista töistä saa periä korkeamman prioriteetin?
- n Perintäsääntö:
  - n 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
<i>Black</i>	5	2	4	0	1	1	1	2	2	$\Omega$	$\Omega$	
<i>Shaded</i>	1	1	1	0	0	1	1	$\Omega$	$\Omega$	$\Omega$	$\Omega$	$\Omega$



Liu kuva 8-22

## Käsitellyt menetelmät

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

## Lisämateriaali

### (luvattu laskareissa ke 5.4.)

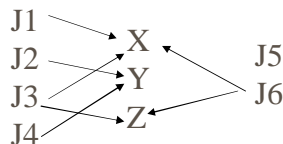
- n Estymisajan laskenta resurssikilpailun ja prioriteettien perusteella (kirjan luku 8.5.4)
- n Ajoitettavuusanalyysi
  - n Kiinteät prioriteetit (RM):  
Aikavaativuus-analyysi ja estymisaika
    - n Perusversio ilman estymistä luku 6.6.2
    - n Perusversio ja estyminen luku 6.8.1
  - n Dynaamiset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

## Estyminen ja prioriteetin perintä sekä prioriteetikatto

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

## Estyminen ja prioriteetin perintä sekä prioriteettikatto

Job	kriitt. alueet	Es	Estää suoraan					Prioriteetin perinnällä				
			J2	J3	J4	J5	J6	J2	J3	J4	J5	J6
J1	[X;10]											
J2	[Y;1]			6			2					
J3	[X;6 [Y;2]]		*		5			*	6			2
J4	[Y;5]			*			4		*	5		2
J5					*					*		4
J6	[Z;4][X;2]					*					*	4



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

## Aikavaativuusanalyysi (RM) ja estymisaika

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

HUOM.  
Työt jakson-  
pituuden  
mukaan  
prioriteettijärj.

$$w_i(t) = e_i + b_i + \sum_{k=1}^{i-1} \left\lceil \frac{t}{p_k} \right\rceil e_k,$$

aikavälillä  $0 < t \leq \min(D_i, p_i)$

## Ajoitettavuusanalyysi (EDF) ja estymisaika

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

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