

582425 Tosiakajärjestelmät

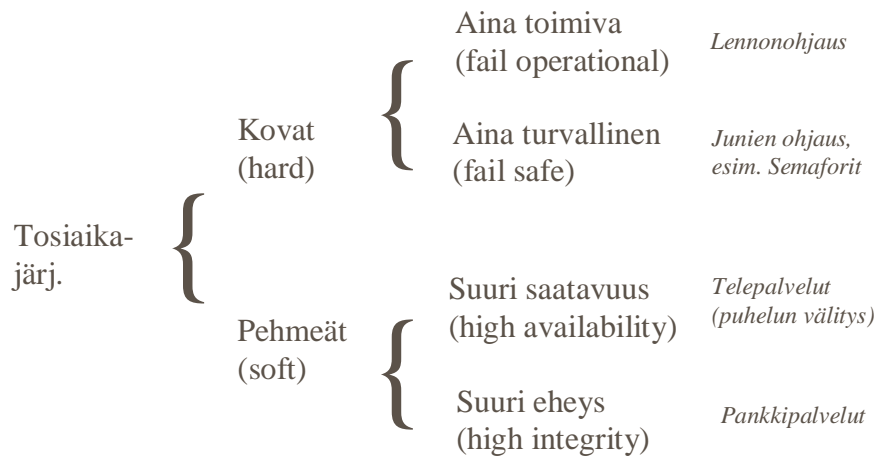
Luento 12: Kertaus

Tiina Niklander
Kevät 2006

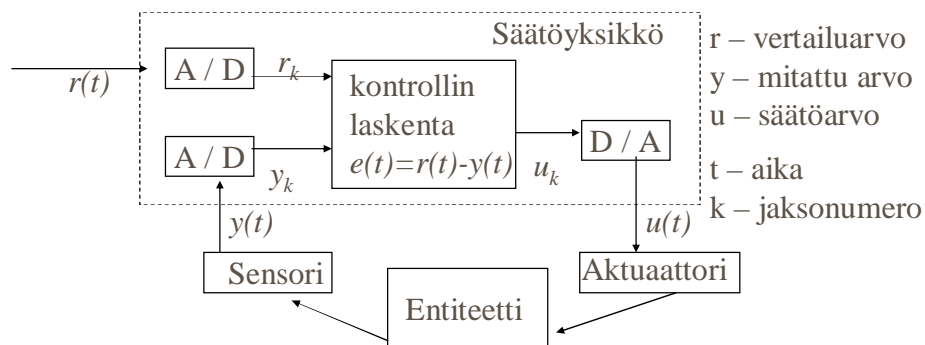
Kurssin rakenne: yleiskuva ...

Johdanto (<i>Liu 1-3</i>)	RM & EDF (<i>Liu 4-6</i>)
Jaksollisuus ja jaksotettavuus (<i>Liu 7</i>)	Mallinnus ja mittaaminen
Resurssit (<i>Liu 8</i>)	Luotettavuus ja turvallisuus
Sanomien vuorotus verkossa (<i>Liu 11</i>)	RT-protokollia (<i>Liu 11 osittain</i>)
Moniprosessit (<i>Liu 9</i>)	Tosiakaj:t (<i>Liu 12 osittain</i>)
Tosiakatietokannat	Kertaus

Tosi aikajärjestelmien luokittelu



Ohjausjärjestelmän malli



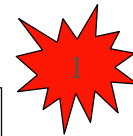
- Joka jaksolla mitataan uusi arvo ja katsotaan kuinka kaukana se on vertailuarvosta. Sitten lasketaan tarvittavat säädöt ja jatketaan

Ajoitus

- n Jaksollinen vai jaksoton
- n Tapahtuman suorituksen keskeyttäminen
- n Kello vai prioriteetti
- n Ahne vai reilu
- n Staattinen vai dynaaminen
- n WCET analyysi
- n RM – Rate monotonic
- n EDF -Earliest deadline first
- n Prioriteetin kääntyminen
- n Käyttöaste

Tehtävän kuvaus

$$J_i(\phi, p, e, d)$$



- n Aloitusaika r_i
 - n vakio, aikaväli, tilastollinen jakauma
- n Suoritusaika e_i
 - n Suorituksen kesto vaihtelee kuten aloitusaika
 - n minimi ja maksimikesto $[e_i^-, e_i^+]$
- n Jakso p_i
 - n on lyhin kahden työn aloituksen välinen aika
- n Vaihe ϕ_i
 - n Tehtävän ensimmäisen aloituksen aika eli $\phi_1 = r_{i,1}$

Käyttöaste U, u



- n Yhden jaksollisen tehtävän käyttöaste on

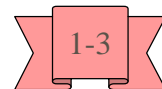
$$u_i = e_i / p_i$$

- n Koko järjestelmän tai sen osan käyttöaste (n tapahtumaa)

$$U = \sum_{i=1}^n u_i = \sum_{i=1}^n \frac{e_i}{p_i}$$

- n Ajoitettava käyttöaste – vuorotusmenetelmän perusteella laskettu järjestelmän käyttöaste

Kirjan luvut 1-3:



- n Luku 1: Typical Real-Time Applications
 - n EI: 1.1.2. More complex control-law comp.
 - n EI: 1.3 Signal processing
 - n EI 1.4.2 Multimedia applications
 - n Muut aliluvut: yleiskäsitys asiasta riittää
- n Luku 2: Hard versus Soft Real-Time Systems
 - n Kokonaan
- n Luku 3: A Reference Model
 - n EI: 3.4 Precedence constraints and data dependency
 - n EI: lisämateriaalia *3.5, *3.6.3, *3.6.4, * 3.7.2

Kello-ohjattu ajoittaminen



- n Staattinen taulukkopohjainen ajoitus
 - n Taulukko yhdelle hyperperiodille, jota toistetaan
- n Jaksollinen ajoitus
 - n Ajoituspäätökset säännöllisin väliajoin
 - n Kehys f (kahden päätöshetken väli)
- n Epäsäännöllinen työ jaksollisten sekaan
 - n Slack Stealing
 - n Hyväksymistestaus

Kehyksen koko

- n Alaraja

$$f \geq \max_{1 \leq i \leq n} (e_i)$$

- n Kehysten määrä

$$\lfloor p_i / f \rfloor - p_i / f \geq 0$$

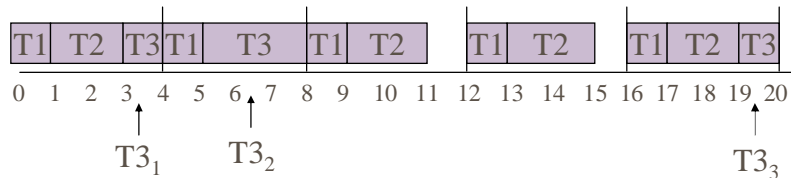
- n Yläraja

$$2f - \text{syt}(p_i, f) \leq D_i$$

Kehyksen koon määrittäminen (esim 2)

	p	e	d
T1	4	1	4
T2	5	2	7
T3	20	5	20

- Hyperperiodi $H = 20$
- Alaraja: $f \geq \max(1, 2, 5)$
- Kokovaihtoehdot 2,4,5,10,20
- Yläraja: $2f\text{-syt}(p_i, f) \leq D_i : f \leq 4$
- T3 jaettava osiin ! 1,3,1 $\Rightarrow f=4$



Slack Stealing – Jouston kähmintä

- Kunkin kehyksen sisällä ajoitetaan jaksottomat kehyksen vapaaseen osaan jaksollisten edelle.
- Jos jaksottomia ei ole, niin suoritetaan jaksollisia.

Prioriteettipohjaiset ajoitukset

- n Menetelmiä
 - n Rate Monotonic
 - n Deadline Monotonic
 - n Earliest Deadline First
 - n Least Slack First

- n Ajoitettavuusanalyysi
 - n Käyttöasteen avulla
 - n Aikavaativuusanalyysi



Ajoitettavuustestaus (EDF): Käyttöasteen avulla

- Riittävä ja myös välttämätön ehto EDF (ja LST) ajoitukselle:

$$U_{EDF} = \sum_{i=1}^n \frac{e_i}{\min(p_i, d_i)} \leq 1$$

HUOM: Jakajassa on joko jakson pituus tai työn suhteellinen aikaraja jakson alusta

Ajoitettavuustestaus (RM) käyttöasteen avulla

- Riittävä ehto RM (ja DM) ajoitukselle on:

$$U_{RM} \leq n(2^{1/n} - 1)$$

- Konservatiivinen raja saadaan:

$$\lim_{n \rightarrow \infty} n(2^{1/n} - 1) = \ln 2 \approx 0.693$$

Aikavaativuusanalyysi



$$w_i(t) = e_i + \sum_{k=1}^{i-1} \left\lceil \frac{t}{p_k} \right\rceil e_k, \text{ kun } 0 < t \leq p_i$$

voidaan ajoittaa, kun

$$\forall i \exists w_i(t) \leq t,$$

jollekin $t \leq d_i \leq p_i$

- T1 (3,1)
- T2 (5,1.5)
- T3 (7,1.25) ja
- T4 (9,0.5)

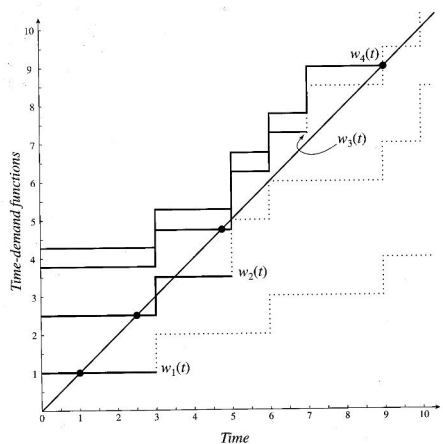


FIGURE 6-9 Time-demand analysis (3,1), (5, 1.5), (7, 1.25), and (9, 0.5).

Estyminen analyysin kannalta

- n Korkeamman prioriteetin työn voi estää vain yksi alempi ennen sen pääsyä suoritukseen
- n Joten lisätään arviossa korkeamman prioriteetin työn suoritusaikaan alempien estoaikojen maksimi, näin yhden työn pisin estymisaika (blocking time) on

$$b_i(np) = \max_{i+1 \leq k \leq n} \theta_k$$

Kun työt järjestetty prioriteettien mukaan ja θ_i on yhden eston kesto

Aikavaativuusanalyysi (RM) ja estymisaika

- n Kuhunkin työhön kohdistuva estymisaika vaikuttaa vain siihen itseensä. Se tulee siis työn oman suoritustajan 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

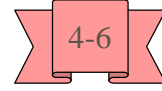
- Koko tehtäväjoukko on ajoitettavissa, jos minkään tehtävän estymisaika ei aiheuta kokonaiskuorman päällä ylikuormitusta!
- Estäjänä voivat toimia vain työt, joiden suhteellinen aikaraja on suurempi.

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

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 $D_k > D_i$

Kirjan luvut 4-6



- n Luku 4: Commonly Used Approaches ...
 - n EI: 4.6, 4.7
 - n 4.8. ja 4.9 ei teoreemoja (4.4, 4.5)
- n Luku 5: Clock-Driven Scheduling
 - n EI: 5.5.2, 5.6.3, *5.8
 - n Kuvissa olleita algoritmeja ei kysytä
- n Luku 6: Priority-Driven Scheduling of Periodic Taska
 - n EI: 6.3.1, 6.5.1, 6.5.3, 6.6,
 - n EI: *6.7.2, *6.7.3, 6.7.4, *6.7.5
 - n EI: 6.8.5, 6.8.6., 6.8.7

Sporadiset ja epäsäännölliset tehtävät ja työt



- n Hyväksymistesti
- n Jaksollisia (tai jaksottuvia) palvelimia
 - n Osa-aikapalvelin (deferrable server)
 - n Sporadinen palvelin
 - n Sporadinen/tausta-ajo palvelin

Hyväksymistesti sporadisille töille

- n Tehdään vuorotus ennen suoritusta
- n Tarvitaan:
 - n nykyinen ajoitus ja töiden parametrit
 - n Uuden työn aikaraja ja maksimisuoritus aika $S(d,e)$
- n Jos järjestelmästä löytyy riittävästi vapaata aikaa σ ennen aikarajaa, työ voidaan hyväksyä eli
$$e \leq \sigma_c(t,l),$$
missä t on saapumisaika ja l viimeinen ehyt kehys $< d$
- n Jos samanaikaisesti useita saapujia, käsitellään ne aikarajan mukaisessa (EDF) järjestyksessä

Hyväksymistestin 'algoritmi'

- n Ensimmäinen vaihe:
 - n Riittävätkö tyhjät aikajaksot suorittamiseen
$$e \leq \sigma_c(t,l) = \sigma(t,l) - \sum_{d_k \leq d} (e_k - \xi_k)$$
- n Toinen vaihe:
 - n Myöhästyisikö joku jo hyväksytty sporadinen työ, jonka takaraja tämän jälkeen?

$$0 \leq \sigma'_k = \sigma_k - e$$

Järjestelmämalli

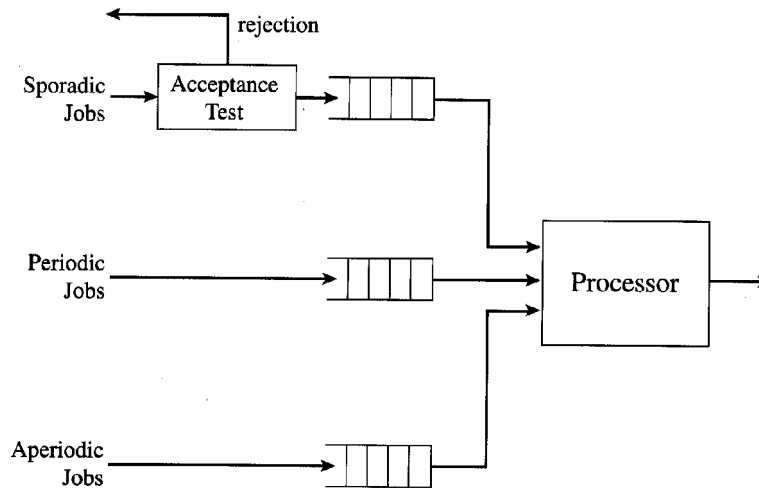


FIGURE 7-1 Priority queues maintained by the operating system.

Osa-aikapalvelin (Deferrable Server):

Kulutus- ja täydennyssäännöt

- n Kulutussääntö (Consumption Rule):
 - n Palvelimen suoritusaikaa kuluu yksi yksikkö kutakin suoritettua aikayksikköä kohti
- n Täydennyssääntö (Replenishment Rule):
 - n Palvelimen suoritus aika palautetaan maksimiin aina kunkin suoritusjakson aluksi eli budjetiksi asetetaan e_s aina kellon ollessa kp_s , $k=0,1,2,\dots$
 - n Huom: suoritus aikaa ei voi säästää seuraavaan jaksoon

Osa-aikapalvelin: Aikavaativuusanalyysi (RM)

- n Kaavassa huomioitu estoaika b_i ja osa-aikapalvelimen aiheuttama estyminen:

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

kun $0 < t \leq p_i$

- n Oletus: osa-aikapalvelimen prioriteetti on suurin ja jakson pituus siis pienin

Ajoitettavuustesti (EDF)

- n Jaksollinen tehtävä T_i on ajoitettavissa EDF-menetelmällä järjestelmässä, jossa on n riippumaton keskeytettävää jaksollista tehtävää ja osa-aikapalvelin, jonka jakso on p_s , suorituskiintiö e_s ja käyttöaste u_s , jos

$$\sum_{k=1}^{i-1} \frac{e_k}{\min(d_k, p_k)} + u_s \left(1 + \frac{p_s - e_s}{d_i} \right) \leq 1$$

↖
Jaksolliset työt

↖
osa-aikapalvelin

(Kirjan Teoreema 7.3)

Sporadinen palvelin: kulutus

- n Kiintiötä kuluu yksi yksikkö per aikayksikkö, kun joko
 - n palvelin on suorituksessa tai
 - n palvelin on ollut jossain vaiheessa suorituksessa edellisen täydennyshetken (t_r) jälkeen ja palvelimella on edelleen töitä suoritusjonossa
- n Mikäli kumpikaan ehto ei täyty, niin kiintiötä ei kulu
- n HUOM: Kiintiötä siis kulutetaan, vaikka palvelin ei suorittaisikaan mitään tehtävää.

Sporadinen palvelin: täydennys

- n Aluksi (ja aina täydennettäessä) kiintiö = e_s ja t_r = nykyhetki
- n Ajanhetkellä t_r , kun palvelin saa ensimmäisen kerran suoritusvuoron täydennyksen jälkeen
 - n jos palvelin odotti korkeamman prioriteetin töitä, niin seuraava täydennyshetki säilyy $t_r + p_s$
 - n jos palvelin oli vapaa, niin täydennyshetkeä siirretään siten, että uusi hetki on $t_r + p_s$, kun $t_r < t_f$
- n Normaalijaksojen ulkopuolella täydennys tehdään, jos
 - n Korkeamman prioriteetin työt ovat olleet suorituksessa koko jakson p_s , niin täydennys tehdään heti kiintiön tyhjennyttyä
 - n Jos koko järjestelmä oli jakson aikana tyhjäkäynnillä ajanhetkeen t_b asti, niin täydennys tehdään hetkellä $\min(t_e + p_s, t_b)$

Sporadisten töiden hyväksymis-testi kun

käytössä EDF -vuorotus

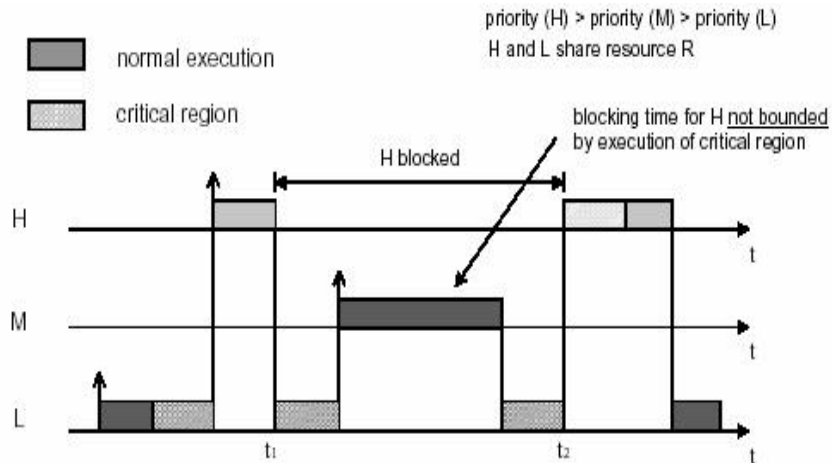
- n Ensimmäinen saapuva työ $S(t, d, e)$ hyväksytään, jos $e/(d-t) \leq 1-\Delta$, missä Δ on kaikkien sporadisten töiden sallittu yhteinen maksimitiheys
 - n HUOM: d jakaa aikavälin kahteen osaan I_1 ja I_2 . I_1 :n tiheys on $e/(d-t)$ ja I_2 :n 0.
- n Yleinen tapaus: järjestelmässä on jo n_s aiemmin hyväksytyä työtä. Työ hyväksytään, jos
$$e/(d-t) + \Delta_k \leq 1-\Delta,$$
kaikille $k=1, \dots, l$, missä l on sen aikavälin indeksi, johon d kuuluu

Jaettujen resurssien käyttö



- n Prioriteetin kääntyminen
- n Kääntymisen välttäminen
 - n Irroittamattomat kriittiset alueet
 - n Prioriteetin perintä (Priority Inheritance Protocol).
 - n Prioriteettikatto (Priority Ceiling Protocol)
 - n Kattoprioriteetti (Ceiling Priority)

Prioriteetin kääntyminen (Priority inversion)



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

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

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 prioriteetiltaan keskitasoa olevan tapahtuman keskeyttämästä tapahtumaa T.
- n Haitat
 - n Prioriteetin perintä voi aiheuttaa lukkiuman
 - n Linkitetty odotus (chained blocking)

Prioriteettikatto

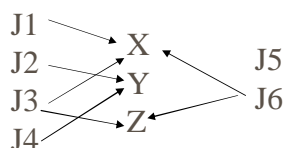
- n Idea
 - n Jokaiselle resurssille asetetaan prioriteettikatto $\Pi(R_i)$ yhtä suureksi kuin sen korkeimman tapahtuman prioriteetti, joka tarvitsee tätä resurssia ja siis saa lukita resurssin.
 - n Tapahtuma T saa siirtyä kriittiselle alueelle ja varata resurssin vain, jos sen prioriteetti on korkeampi 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.

Estyminen ja prioriteetin perintä sekä prioriteettikatto

- 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]												
J3	[X;6 [Y;2]]	J1		6			2						
J4	[Y;5]	J2	*		5			*	6				2
J5		J3		*			4		*	5			2
J6	[Z;4][X;2]	J4			*					*			4
		J5				*					*		4



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

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

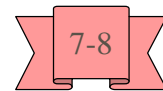
Irroitusten katto

- 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 Tietuille dynaamista prioriteettia käyttäville järjestelmille (esim. takarajoihin perustuvat) on mahdollista etukäteen selvittää jaksollisten töiden mahdolliset irroitustilanteet.

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 yhtäsuuri kuin $\Psi(t)$, niin työ saa resurssin, muuten joutuu odottamaan

Kirjan luvut 7-8



- n Luku 7: Scheduling Aperiodic and Sporadic Jobs in Priority-Driven Systems
 - n Ei SpSL servers (luvun 7.3.2 osa)
 - n EI: 7.4.1, 7.4.2, 7.4.3, 7.4.4.
 - n EI: *7.5, *7.6, *7.7.2,
 - n EI: 7.8.2, 7.9
- n Luku 8: Resources and Resource Access Control
 - n EI: *8.7.3, 8.9, *8.10

Moniprosessorijärjestelmät



- n Kaksi vaihtoehtoista mallia:
 - n Päästä-päähän (työ vaihtaa prosessoria)
 - n Etäsuorituksia (työ yhdessä paikassa, mutta voi pyytää resursseja tilapäisesti muualta)
- n Töiden sijoittelu prosessorille
 - n RMFF - Rate-Monotonic First Fit
 - n RMST - Rate-Monotonic Small Tasks
- n Prosessorin paikallinen vuorotusmekanismi
- n Globaali synkronointi

RMFF algoritmi

- n First Fit muunnelma
- n Rate-monotonic ajoitus prosessoreilla
 - n Käytetään lokeron koon ylärajana RM:n tarkkaa ylärajaa

$$u_i + U_k \leq n(2^{1/n} - 1)$$

- n Järjestetään työt jakson pituuksien perusteella:
 - n Lyhimmät ensin
 - n Tämä on linjassa RM:n priorisoinnin mukaan

RMST algoritmi

- n Rate-Monotonic Small Tasks (RMST)
 - n Hyödyntää RM:n käyttäytymistä harmoonisten jaksojen kanssa.
 - n Kun tehtävien jaksojen pituudet lyhimmän monikertoja, niin käyttöasteen maksimi kasvaa merkittävästi (jopa ykköseen)
- n Työt sijoitellaan jaksonpituudesta lasketun arvon X_i määräämässä suuruusjärjestyksessä

$$X_i = \log_2 p_i - \lfloor \log_2 p_i \rfloor$$

RMST algoritmi

- n Ajoitettavuusehto prosessorille on $u_i + U_k \leq \max(\ln 2, 1 - \zeta_k \ln 2)$, missä $\zeta_k = \max X_l - \min X_l$
- n Työt ovat sijoiteltavissa m :lle prosessorille, jos työkuormasta laskettu käyttöaste on pienempi kuin

$$U_{RMST} = (m - 2)(1 - u_{\max}) + 1 - \ln 2, \text{ kun } m > 2$$

missä u_{\max} on töiden suurin yksittäinen käyttöaste

Tosiaikainen tietoliikenne



- n Sanomien kuljetus päästä päähän
 - n Taataan kuljetusaika kehyksille
 - n Käytetään WCET arvioita
- n Lähetettävänä sanoman valinta (yhteystasolla)
 - n WFQ (Weighted Fair-Queueing)
 - n Delay Earliest-Due-Date (D-EDD)
 - n Jittered-EDD
- n Lähetettävänä koneen valinta (fyysinen taso)
 - n CSMA/CD (carrier sense multiple access / collision detection) – esim. Ethernet
 - n TDMA (time division multiple access)

WFQ vuorotusperiaate

- n Pyrkii takaamaan kullekin aktiiviselle yhteydelle sille luvatusuhteellisen osuuden linkin kapasiteetista
- n Yhteyksien saapuville ja eteenpäin lähteville paketeille lasketaan lopetusnumeroita. Paketit lähetetään näiden numeroiden mukaisessa (EDF) järjestyksessä.
- n Lopetusnumero kasvaa yhteyden suhteellisen osuuden mukaisesti.

WFQ: Lopetusnumeron (fn) laskeminen

- n 1) Tyhjään linkkiin saapuva (yhteyden i) ensimmäinen paketti:
 - n $t_{i1} = t; U_b += u_i; fn_i += e/u_i;$
- n 2) Seuraavat paketit (kun linkki on kiireinen)
 - n Yhteyden i ensimmäinen paketti
 - n Linkin laskuri $FN += (t - t_{i1}) / U_b$
 - n $fn_i = \max(FN, fn_i) + e/u_i;$ ja SFN-jonoon (fn_i, i)
 - n $t_{i1} = t; U_b += u_i;$
- n 3) Yhteyden i paketin lähetys päättyi
 - n Lisää yhteyden i paketteja jonossa
 - n $fn_i += e/u_i$ ja SFN-jonoon (fn_i, i)
 - n Ei tällä hetkellä lisää paketteja i:ssä (-> idle)
 - n Linkin laskuri $FN += (t - t_{i1}) / U_b$
 - n $t_{i1} = t; U_b -= u_i;$

Delay Earliest-Due-Date (D-EDD)

- n Perustuu EDF:ään
- n Päästä-päähän yhteyden muodostuksessa yhden paketin kuljettamiseen käytettävissä oleva aika jaetaan linkkien kesken, kukin linkki saa vähintään tarvitsemansa minimin.
- n Saapuvat viestit jaksotetaan suhteellisen saapumisaikansa mukaisesti. (Näin estetään varattua laajempi kaistan käyttö)
- n Muistuttaa sporadisen palvelimen erikoistapausta

D-EDD: yhteyden muodostus

- n Lähettäjä tekee aloitteen. Se lähettää 'request-for-connection'-viestin, jossa kuvataan tuleva liikenne (p_i, D_i)
- n Jokainen matkalle osuva kytkin (switch)
 - n Hyväksymispäätös ja alustava kapasiteetinvaraus
- n Vastaanottaja tarkistaa reitin kelvollisuuden
 - n Kun yhteys voidaan muodostaa, vastaanottaja määrää kytkimien paikallisen aikarajat (jakaa ylimääräisen ajan)
- n Paluuviestistä kytkimet
 - n poimivat uudet aikarajansa, tekevät pysyvät varaukset

D-EDD: Pakettien käsittely

- n Vuonvalvonta on keskeistä. EDF ei selviä ylikuormasta, joten sitä ei saa syntyä.
- n Saapuvan paketin paikallista aikarajaa ei lasketa todellisesta saapumisajasta vaan ns. efektiivistä saapumisajasta

$$a_{i,j}^e = \max(a_{i,j-1}^e + p_i, a_{i,j})$$

- n Paikallinen aikaraja on siis

$$d_{i,j} = a_{i,j}^e + D_{i,k}$$

Jitter-EDD

n Jitter-EDD on D-EDD:n muunnos, jossa pienennetään sanoman kulkuajan variaatiota.

n D-EDD:ssä tuo vaihtelu on varsin suuri, jopa

$$\sum_{j=1}^k (D_{i,k}) - k$$

n Jitter-EDD ei ole yhtä ahne. Sanomaa ei välttämättä aina lähetetä heti kun voitaisiin. Lähetyshetkien väli pyritään vakioimaan.

J-EDD: toiminta

- Yhteyden muodostus kuten D-EDD
- Paketin käsittely:
 - Vaihtelun tasoittamiseksi lähetettävään pakettiin lisätään tieto 'ahead-time' eli kuinka paljon se on etuajassa suhteessa aikarajaansa.
 - Saapuvalla paketilla lasketaan lähetysaika (ready time), jolloin vasta se laitetaan lähetysjonoon (D-EDD laittoi jonoon heti)

$$r_{i,j} = \max(a_{i,j}^e, a_{i,j} + ah_{i,j})$$

WRR (Weighted Round-Robin)

- n Kustakin saapuvasta yhteydestä lähetetään yhdellä kierroksella korkeintaan maks. sanomia eteenpäin
- n Oletukset:
 - n Ei globaalia kelloa tai aikarajan mukaan järjestettyä jonoa
 - n Ei tehtävien (tai töiden) välisiä riippuvuuksia
 - n Yhteyksillä on vakiotahdi (constant bit rate)
 - n Datavirtaa (message stream) kuvataan (p_i, e_i, D_i)
 - n p ja D kuten ennenkin, mutta e on nyt viestien määrä yhdellä kierroksella (instance)

Greedy-WRR

- Kullakin yhteydellä on paino wt_i
- Yhden kierroksen kuluessa yhteyden i viesteistä lähetetään edelleen korkeintaan wt_i kappaletta
- Yhteydet käsitellään vuorotellen (siitä RR) ja kaikki painon mukaan mahtuvat viestit lähetetään edelleen
- Kierroksen maksimipituus RL on kiinteä

$$RL \geq \sum_{i=1}^n wt_i$$

$$RL < \min(p_i)$$

$$wt_i \geq \left\lceil \frac{e_i}{\lfloor p_i / RL \rfloor} \right\rceil$$

Yhteenveto

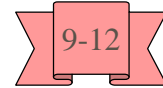
- Prioriteettipohjaisia käytäntöjä

Performance Measures	WFQ	Delay-EDD	Jitter-EDD
Acceptance Test	$O(1)$	$O(1)$	$O(1)$
Sched. Complexity	$O(n)$	$O(\log n)$	$O(\log n)$
End-to-end delay bound	$E/u + \rho(e+1)$	$\leq D$	$\leq D$
End-to-end jitter	$const \times \rho$	$const \times \rho$	$const$
Buffer-space requirem.	$const \times \rho$	$const \times \rho$	$const$

Tosiaikakäyttöjärjestelmät

- n Palveluja:
 - n Prosessit, säikeet
 - n Aikapalvelut
- n POSIX
- n Host / target ohjelmakehitys
- n Esimerkkejä
 - n VxWorks
 - n LynxOS
 - n QNX
 - n RT-Linux

Kirjan luvut 9-12



- n Luku 9: Multiprocessor Scheduling ...
 - n VAIN 9.1 – 9.2.1 (s. 330-344)
- n Luku 10:EI
- n Luku 11: Real-Time Communication
 - n VAIN 11.1. – 11.3.1 (s.420 – 447)
- n Luku 12: Operating Systems
 - n EI: *12.4, *12.5, 12.7.1

Kurssikoe

- n Ma 8.5. klo 16.00 Exactum A111
- n Vaihtoehtoista aikaa ei järjestetä ellen saa selkeätä erillistä pyyntöä!
- n Kokeessa
 - n Sekä laskuharjoitusten kaltaisia
 - n että teoriapainotteisempia tehtäviä