582425 Tosiaikajärjestelmät
Luento 12: Kertaus

Tiina Niklander
Kevät 2006

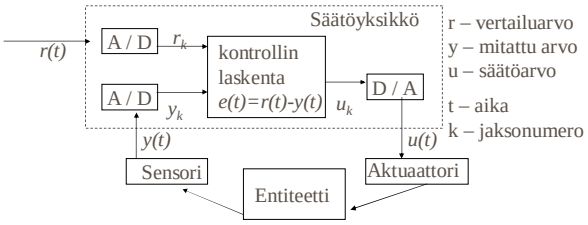
Kurssin rakenne: yleiskuva ...

Johdanto *(Liu 1-3)*	RM & EDF *(Liu 4-6)*
Jaksollisuus ja jaksotettavuus *(Liu 7)*	Mallinnus ja mittaaminen
Resurssit *(Liu 8)*	Luotettavuus ja turvallisuus
Sanomien vuorotus verkossa *(Liu 11)*	RT-protokollia *(Liu 11 osittain)*
Moniprosessit *(Liu 9)*	Tosiaikakj:t *(Liu 12 osittain)*
Tosiaikatietokannat	Kertaus

Tosiaikajärjestelmien luokittelu

- Tosiaika-järj.
  - Kovat (hard)
    - Aina toimiva (fail operational) — *Lennonohjaus*
    - Aina turvallinen (fail safe) — *Junien ohjaus, esim. Semaforit*
  - Pehmeät (soft)
    - Suuri saatavuus (high availability) — *Telepalvelut (puhelun välitys)*
    - Suuri eheys (high integrity) — *Pankkipalvelut*

Ohjausjärjestelmän malli



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

Ajoitus

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

Tehtävän kuvaus

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

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

Käyttöaste U,u

- Yhden jaksollisen tehtävän käyttöaste on

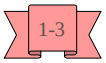
$$u_i = e_i / p_i$$

- Koko järjestelmän tai sen osan käyttöaste (n tapahtumaa)

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

- Ajoitettava käyttöaste – vuorotusmenetelmän perusteella laskettu järjestelmän käyttöaste

Kirjan luvut 1-3:



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

Kello-ohjattu ajoittaminen

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

Kehyksen koko

- Alaraja

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

- Kehysten määrä

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

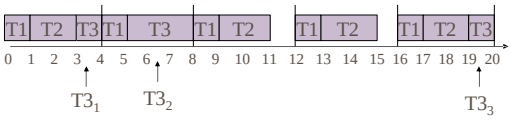
- Yläraja

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

Kehyksen koon määrääminen (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 => 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

- Menetelmiä
  - Rate Monotonic
  - Deadline Monotonic
  - Earliest Deadline First
  - Least Slack First
- Ajoitettavuusanalyysi
  - Käyttöasteen avulla
  - 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:

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

Aikavaativuusanalyysi

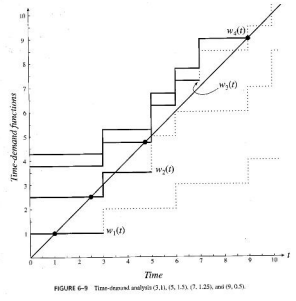
$$w_i(t) = ei + \sum_{k=1}^{i-1} \left\lceil \frac{t}{p_k} \right\rceil e_k, \text{ kun } 0 < t \leq p_i$$

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)



Estyminen analyysin kannalta

- Korkeamman prioriteetin työn voi estää vain yksi alempi ennen sen pääsyä suoritukseen
- 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

- 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.

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

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.

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

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 $D_k > D_i$

Kirjan luvut 4-6

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

Sporadiset ja epäsäännölliset tehtävät ja työt

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

Hyväksymistesti sporadisille töille

- Tehdään vuorotus ennen suoritusta
- Tarvitaan:
  - nykyinen ajoitus ja töiden parametrit
  - Uuden työn aikaraja ja maksimisuoritusaika S(d,e)
- Jos järjestelmästä löytyy riittävästi vapaata aikaa σ ennen aikarajaa, työ voidaan hyväksyä eli

  $e \le \sigma_c(t,l)$,

  missä t on saapumisaika ja l viimeinen ehyt kehys <d
- Jos samanaikaisesti useita saapujia, käsitellään ne aikarajan mukaisessa (EDF) järjestyksessä

Hyväksymistestin ’algoritmi’

- Ensimmäinen vaihe:
  - Riittävätkö tyhjät aikajaksot suorittamiseen

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

  $$0 \le \sigma_k' = \sigma_k - e$$

Järjestelmämalli

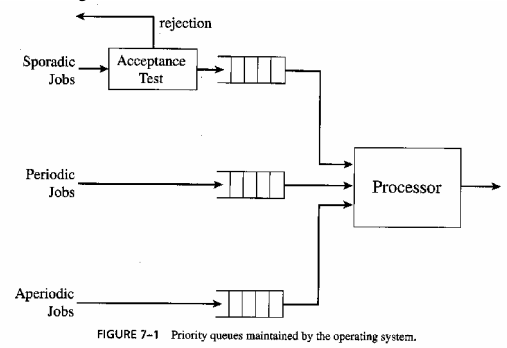


FIGURE 7–1 Priority queues maintained by the operating system.

Osa-aikapalvelin (Deferrable Server): Kulutus- ja täydennyssäännöt

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

Osa-aikapalvelin: Aikavaativuusanalyysi (RM)

- 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 $0 < t \le p_i$

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

Ajoitettavuustesti (EDF)

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

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

Jaksolliset työt — osa-aikapalvelin

(Kirjan Teoreema 7.3)

Sporadinen palvelin: kulutus

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

Sporadinen palvelin: täydennys

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

- Ensimmäinen saapuva työ S(t, d, e) hyväksytään, jos $e/(d-t) \leq 1-\Delta$, missä Δ on kaikkien sporadisten töiden sallittu yhteinen maksimitiheys
  - HUOM: d jakaa aikavälin kahteen osaan I_1 ja I_2. I_1:n tiheys on $e/(d-t)$ ja I_2:n 0.
- Yleinen tapaus: järjestelmässä on jo n_s aiemmin hyväksyttyä työtä.Työ hyväksytään, jos

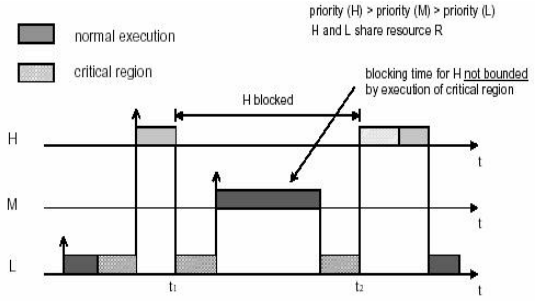
$$e/(d-t) + \Delta_k \leq 1-\Delta,$$

kaikille k= 1,...,l, missä l on sen aikavälin indeksi, johon d kuuluu

Jaettujen resurssien käyttö

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

Prioriteetin kääntyminen (Priority inversion)



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

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

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)

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.

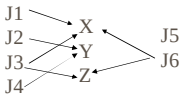
Estyminen ja prioriteetin perintä sekä prioriteettikatto

- 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)

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

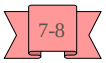
Irroitusten katto

- 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.

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

Kirjan luvut 7-8



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

Moniprosessorijärjestelmät

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

RMFF algoritmi

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

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

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

RMST algoritmi

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

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

RMST algoritmi

- Ajoitettavuusehto prosessorille on

$$u_i + U_k \le \max(\ln 2, 1 - \zeta_k \ln 2), \text{ missä } \zeta_k = \max X_l - \min X_l$$

- 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

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

WFQ vuorotusperiaate

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

WFQ: Lopetusnumeron (fn) laskeminen

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

Delay Earliest-Due-Date (D-EDD)

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

D-EDD: yhteyden muodostus

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

D-EDD: Pakettien käsittely

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

$$a^e_{i,j} = \max\left(a^e_{i,j-1} + p_i, a_{i,j}\right)$$

- Paikallinen aikaraja on siis

$$d_{i,j} = a^e_{i,j} + D_{i,k}$$

Jitter-EDD

- Jitter-EDD on D-EDD:n muunnelma, jossa pienennetään sanoman kulkuajan variaatiota.
  - D-EDD:ssä tuo vaihtelu on varsin suuri, jopa

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

- 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.
  - Saapuvalle paketille lasketaan lähetysaika (ready time), jolloin vasta se laitetaan lähetysjonoon (D-EDD laittoi jonoon heti)

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

WRR (Weighted Round-Robin)

- Kustakin saapuvasta yhteydestä lähetetään yhdellä kierroksella korkeintaan maks. sanomia eteenpäin
- Oletukset:
  - Ei globaalia kelloa tai aikarajan mukaan järjestettyä jonoa
  - Ei tehtävien (tai töiden) välisiä riippuvuuksia
  - Yhteyksillä on vakiotahti (constant bit rate)
  - Datavirtaa (message stream) kuvataan (p_i,e_i,D_i)
    - 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

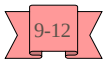
- Prioriteettipohjaisia käytäntöjä

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+ρ(*e*+1)	≤D	≤D
End-to-end jitter	*const* × ρ	*const* × ρ	*const*
Buffer-space requirem.	*const* × ρ	*const* × ρ	*const*

Tosiaikakäyttöjärjestelmät

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

Kirjan luvut 9-12



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

Kurssikoe

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