Tosiaikajärjestelmät: Luento 3
Epäsäännöllisten töiden ajoitus

Tiina Niklander
20.3.2006 (päiv. 22.3.)

Sisältö

- Yleistä
- Jaksottomien (ei-tosiaikaisten) töiden jaksolliset vuorotuspalvelut
  - ”Osa-aika” palvelimet (deferrable servers)
  - Sporadiset palvelimet
  - ”Vakio käyttöaste” (constant utilization)
  - ”Koko kaistanleveys” (Total bandwidth)
  - ”Painotettu reilu odotus” (Weighted Fair-Queing)
- Sporadiset (tosiaikaiset satunnaiset) työt
- Kaksitasoinen integroitu vuorottaminen

Järjestelmämalli

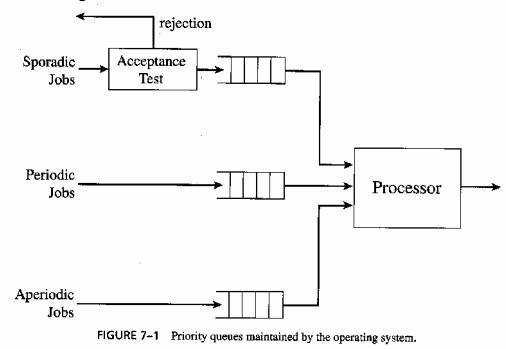


FIGURE 7–1 Priority queues maintained by the operating system.

Sporadisten ja jaksottomien ajoitus

- Kellopohjainen ajoitus jaksollisilla
  - Kellotettujen töiden sekaan sijoitellaan vapaisiin aikaviipaleisiin tausta-ajona.
- Prioriteettiperustainen ajoitus jaksollisilla
  - Alimman prioriteettitason työ (ns. tausta-ajo) voi joutua odottamaan kauan

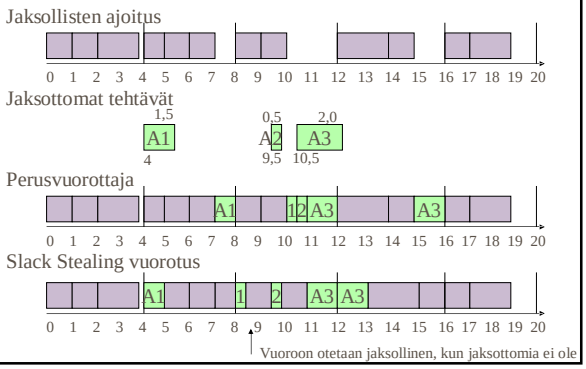
Kellopohjainen ajoitus ja epäsäännölliset työt

- Työluokkien tärkeysjärjestys:
  - Taulukoidut jaksolliset
  - Sporadiset (näillä on aikarajat)
  - Jaksottomat (aperiodiset, ei aikarajaa)
- Muokataan jaksollista ajoittajaa siten, että
  - Jaksollisilta jääneet vapaat aikajaksot käytetään sporadisille ja jaksottomille
- Sporadisille on taattava suoritus ajallaan
  - Hyväksymistesti ennen mukaanottoa

Epäsäännöllisten tehtävien ajoitus jaksollisten sekaan

- Satunnaiset ei-tosiaikaiset tehtävät ajoitetaan jaksollisilta tehtäviltä jääneisiin koloihin.
- **Slack Stealing**-menetelmä parantaa jaksottomien vasteaikaa.
  - Kunkin kehyksen sisällä ajoitetaan jaksottomat kehyksen vapaaseen osaan jaksollisten edelle.
  - Jos jaksottomia ei ole, niin suoritetaan jaksollisia.

Esimerkki: Slack Stealing



Kehyksen ylitys

- Mitä tehdään, jos kehyksen alussa on vielä edellisen kehyksen työ suorituksessa?
  - Annetaan jatkaa – riskinä, että tämäkin kehys myöhästyy
  - Lopetetaan työ, kirjataan ja jätetään tilanteen korjaus erilliselle toipumismekanismille
  - Siirretään työ jaksottomien jonon ensimmäiseksi, jolloin se suoritetaan loppuun vasta myöhemmin, aikarajansa jälkeen

Hyväksymistesti

- Sopii sekä jaksollisille että sporadisille
- Työ hyväksytään vain, jos se voidaan suorittaa ajoissa jo muiden hyväksyttyjen kanssa ilman, että mikään niistä ylittää aikarajansa.
- Ei-tosiaikaisten (jaksottomien) töiden palvelutasoa ei tässä yhteydessä tarkastella, koska ne voivat viipyä järjestelmässä, kunnes saavat suoritusvuoron
- Seuraavaksi tarkasteltava hyväksymistestaus liittyy nimenomaan jaksolliseen ajoittajaan (mm. kehykset ja niiden vapaat ajat).

Hyväksymistesti

- 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 järjestyksessä

Hyväksymistesti: tarvittava tieto

- Kehyksien vapaat ajat (slack) σ(i,j)
  - voidaan laskea ennen suoritusta ajoitustaulukon muodostuksen yhteydessä
  - Yhden hyperperiodin kehysten käsittely riittää
- Tapahtumien parametrit saadaan pyynnössä $S_k(d_k,e_k)$
- Kunkin tapahtuman suorituksen kesto ξ_k päivitetään aina suorituksen jälkeen
- Tapahtuman tarvitsema vapaa aika σ_k muuttuu ja päivitetään aina ja vain, kun uusia sporadisia tapahtumia hyväksytään

Hyväksymistestin ’algoritmi’

- Ensimmäinen vaihe:
  - Riittävätkö tyhjät aikajaksot suorittamiseen eli $e \le \sigma_c(t,l)$?
  - A) Tarkastetaan ensin työt, joiden aikaraja ennen tätä ehdokasta
- Toinen vaihe:
  - Myöhästyisikö joku jo hyväksytty sporadinen työ?
  - B) Varmistetaan loputkin sporadiset työt

Hyväksymistesti: A)

- Kehyksien ’vapaat ajat’ yli kehysjaksojen eli $\sigma(i,j)$ $\forall$ **i,j: i≤j**
- Kaikkien jo hyväksyttyjen töiden tiedot eli $\forall$ **k ≤ n$_S$** $S_k(d_k,e_k)$
- Kaikista jo aloitetuista töistä tähänastisen suorituksen kesto ξ_k
- Näillä voidaan laskea $\sigma_c(t,l)$

$$\sigma_c(t,l) = \sigma(t,l) - \sum_{d_k \le d} (e_k - \xi_k)$$

Hyväksymistesti: B)

- Työt, joiden aikaraja tarkasteltavan jälkeen $\forall$ **k ≤ n$_S$ ja d$_k$>d** $S_k(d_k,e_k)$
- Näiden suoritusmahdollisuudesta poistetaan *e*:n verran aikaa eli $\sigma_k' = \sigma_k - e$
- Uusi työ voidaan hyväksyä vain kun $\sigma_k' \ge 0$

Prioriteettipohjainen ajoitus ja epäsäännölliset työt

- Varataan oma kiintiö
- Töiden pitää olla ajoitettavissa kokonaisuudessaan
- Kiintiön käsittely?
  - Oma jaksollinen tehtävä, joka suorittaa sporadisia ja jaksottomia töitä (ns. palvelin)
  - Useita toteutusvaihtoehtoja

Vaihtoehtoiset ajoitusmekanismit

- Tausta-ajo (kuten kellotetuissa)
  - muuten hyvä, mutta odotusaika voi prioriteettien vuoksi olla huomattavan pitkä
- Keskeyttävä (ja korkein prioriteetti)
  - Mahdollisimman lyhyt palveluaika, mutta jos ei rajoiteta, niin saatetaan menettää jaksollisia töitä
  - Rajoitusmekanismeja: käyttöasteen yläraja tai slack stealing (jouston kähmintä)
- Kiertokysely (poll)
  - Kyselijällä on jaksollinen suoritus ja tietty sallittu suoritusaika
  - Suorittaa vuorotellen järjestelmään saapuneita jaksottomia töitä

Sisältö

- Yleistä
- Jaksottomien (ei-tosiaikaisten) töiden jaksolliset vuorotuspalvelut
  - ”Osa-aika” palvelimet (deferrable servers)
  - Sporadiset palvelimet
  - ”Vakio käyttöaste” (constant utilization)
  - ”Koko kaistanleveys” (Total bandwidth)
  - ”Painotettu reilu odotus” (Weighted Fair-Queing)
- Sporadiset (tosiaikaiset satunnaiset) työt
- Kaksitasoinen integroitu vuorottaminen

Terminologiaa

- Jaksollinen palvelin (periodic server) $S(p_s,e_s)$
  - e_s on suorituskiintiö eli budjetti (execution budget)
  - Palvelimen *koko* on sen osuus käyttöasteesta eli

$$\tilde{u}_s = e_s / p_s$$

  - Suorituskiintiö *täydennetään* aina jakson alussa
  - Palvelin on *vireessä* aina, kun sillä on jonossa suoritettavia töitä
  - Palvelin on valmiina suoritukseen, kun se on vireessä ja sillä on kiintiötä vielä jäljellä
  - Kiintiötä kuluu suorituksen aikana yksi per aikayksikkö

Jaksollisista palvelimista

- Mitä tehdään kiintiölle, jos ei palvelin ole vireessä?
  - nollataan – kiertokyselijä
  - säilytetään jakson ajan – kaistanleveyden säilyttävä palvelin (bandwidth-preserving)
- Palvelimet kuvataan käyttäen
  - kulutus- ja täydennyssääntöjä
- Palvelin siirretään pois suorituksesta, kun
  - sillä ei ole enää töitä jonossa (ns. idle) tai
  - se on kuluttanut suorituskiintiönsä tyhjäksi

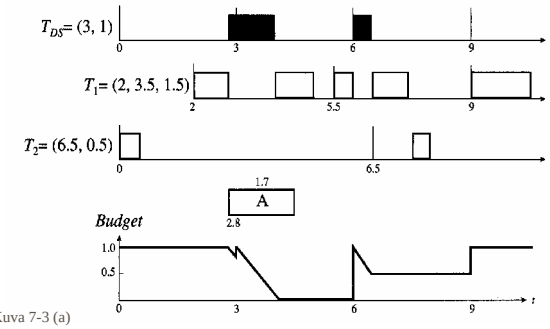
”Osa-aika” palvelin – Deferrable Server

- Jaksollinen palvelin epäsäännöllisille ei-tosiaikaisille töille
- Pitkäkestoisen ei-tosiaikaisen työn suoritus etenee palvelimen tarjoamilla suoritusjaksoilla, kunnes se saa joskus tarvitsemansa suoritusajan täyteen

Osa-aikapalvelin: Kulutus- ja täydennyssäännöt

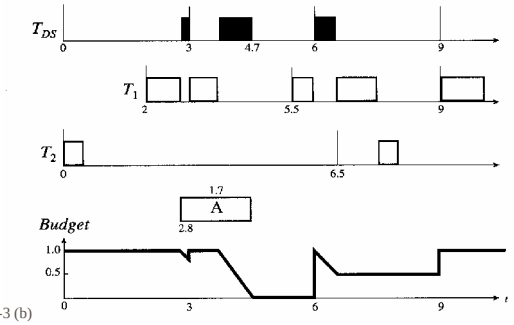
- Kulutussääntö (Consumption Rule):
  - Palvelimen suoritusaikaa kuluu yksi yksikkö kutakin suoritettua aikayksikköä kohti
- Täydennyssääntö (Replenishment Rule):
  - Palvelimen suoritusaika palautetaan maksimiin aina kunkin suoritusjakson aluksi eli budjetiksi asetetaan e_s aina kellon ollessa kp_s,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

Esimerkki (RM-menetelmä)



Kuva 7-3 (a)

Esimerkki (EDF-menetelmä)



Kuva 7-3 (b)

Miten määrätään palvelimen suorituskiintiö?

- Mikään muu työ ei saa ylittyä, joten edes hetkellistä ylikuormitusta ei voida sallia.
- Siis suurin mahdollinen viive palvelimesta, kun
  - Työ $J_{i,c}$ alkaa hetkellä t_0
  - Kaikki sitä korkeamman prioriteetin työt alkavat samalla ajanhetkellä
  - Myös osa-aikapalvelin saa töitä koko budjetikseen samalla ajanhetkellä t_0
  - Palvelimen suorituskiintiö täydennetään ajanhetkellä t_0+e_s

Edellinen ’kriittinen tilanne’

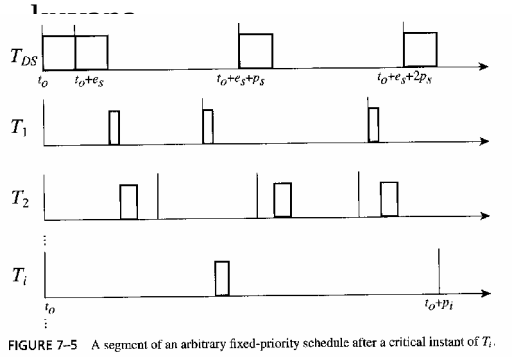


FIGURE 7–5 A segment of an arbitrary fixed-priority schedule after a critical instant of T_i.

Aikavaatimusanalyysi (RM)

- Kaavassa huomioitu (estoaika b_i ja osa-aikapalvelimen aiheuttama estyminen)

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

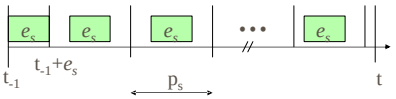
- Oletus: osa-aikapalvelimen prioriteetti on suurin ja jakson pituus siis pienin

Ajoitettavuus (kun käytössä EDF)

- Tarkastellaan osa-aikapalvelimen aiheuttamaan maksimiviivettä työ:n $J_{i,c}$ suoritukseen. J_i:n takaraja (deadline) on t. Osa-aikapalvelimen suurin mahdollinen kulutus aikavälillä (t_{-1}, t] on

$$e_s + \left\lfloor \frac{t - t_{-1} - e_s}{p_s} \right\rfloor e_s$$

- missä t_{-1} on sellainen ajanhetki, jolloin suoritin on viimeeksi ollut vapaa, tai suorittanut työtä, jonka takaraja on t:n jälkeen (eli prioriteetti on pienempi)



Ajoitettavuustesti

- 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

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

Koko järjestelmä — Jaksolliset työt, joiden suhteellinen aikaraja ennen T_i — osa-aikapalvelin

(Kirjan Teoreema 7.3)

Sporadinen palvelin (kiinteä prio)

- Sporadinen palvelin käyttäytyy kuten vastaava jaksollinen tehtävä (p_s,e_s), eli se ei millään aikavälillä koskaan käytä enempää aikaa kuin vastaava jaksollinen tehtävä käyttäisi.
- Sporadinen palvelin käyttäytyy siivommin tilanteessa, jossa osa-aikapalvelin varaa suorittimen kaksinkertaisen kiintiön ajaksi

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 $\min(t_e+p_s, t_b)$

Sporadinen palvelin: esimerkki

T1(3,0.5)

T2(4,1.0)

TS(5,1.5)

T3(19,4.5)

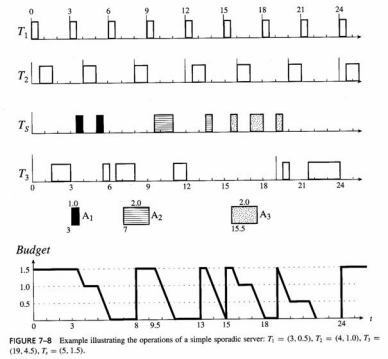


FIGURE 7–8 Example illustrating the operations of a simple sporadic server: $T_1 = (3, 0.5)$, $T_2 = (4, 1.0)$, $T_3 = (19, 4.5)$, $T_s = (5, 1.5)$.

Lisää piirteitä

- Sporadinen palvelin kuluttaa kiintiötään odottaessaan, jos se on ollut hetkenkin suorituksessa.
- Kuitenkin jos koko järjestelmä on tyhjä, niin palvelin voisi käyttää tämän ajan töiden suorittamiseen.
- Muutetaan ainoastaan kulutussääntöä hiukan:
  - Kiintiötä ei kulu suorituksen aikana, jos mikään jaksollinen työ ei ole odottamassa suoritukseen
- Näin saadaan Sporadinen/Tausta-ajo -palvelin

Sporadinen/tausta-ajo palvelin EDF-ajoitetussa järjestelmässä (7.3.3)

- Palvelimella itsellään on myös oltava aikaraja d
  - Aikaraja asetetaan, vain jos palvelimella on töitä jonossa, muuten sen arvo on määräämättä
- Kulutussäännöt: Kiintiötä kuluu vain kun jompikumpi sääntö pätee, muuten kiintiö säilyy
  - Palvelin on suorituksessa
  - Palvelimelle on asetettu takaraja *d*, palvelin on vapaa ja järjestelmässä ei ole odottamassa tai suorituksessa yhtään työtä, jonka takaraja olisi ennen *d*:tä.

Sporadinen/tausta-ajo palvelin EDF-ajoitetussa järjestelmässä

- Täydennyssäännöt (Sääntö R1):
  - Aluksi (ja myöhemmin täydennyksessä), kiintiö = e_s, t_r = nykyhetki; t_e ja d jätetään asettamatta
  - Aina kun t_e on asetettu $d=t_e+p_s$, mikä on myös seuraava täydennyshetki (t_e kuvaa jakson efektiivistä aloitushetkeä)

Sporadinen/tausta-ajo palvelin EDF-ajoitetussa järjestelmässä

- Täydennyssäännöt (Sääntö R2):
  - t_e:n arvon määrääminen:
    - Kun uusi jaksoton työ saapuu palvelimelle:
      - $t_e=t_r$, jos suorituksessa aikavälillä (t_r,t) on ollut vain töitä, joiden takaraja ennen t_r+p_s:ää
      - te=t, jos suorituksessa on samalla aikavälillä ollut yksikin työ, jonka aikaraja on t_r+p_s:n jälkeen
    - Täydennyshetkellä t_r:
      - Jos palvelin on vireessä $t_e=t_r$,
      - Jos töitä ei ole jonossa, niin t_e jää määräämättä

Sporadinen/tausta-ajo palvelin EDF-ajoitetussa järjestelmässä

- Täydennyssäännöt (Sääntö R3):
  - Täydennys tapahtuu täydennyshetkellä, paitsi jos
    - Palvelimen tullessa vireeseen ajanhetkellä t, seuraava täydennyshetki t_e+p_s oli jo. Tällöin kiintiö täydennetään heti, kun se on kulutettu.
    - Kiintiö täydennetään aina, kun koko järjestelmä on ollut vapaa (idle)

Lisää palvelimia

- Keskenään samankaltaisesti käyttäytyviä
  - ”Vakio käyttöaste” (constant utilization)
  - ”Koko kaistanleveys” (Total bandwidth)
  - ”Painotettu reilu odotus” (Weighted Fair-Queing)
- Nämä käyttäytyvät järjestelmässä kuten sporadiset työt, (sporadinen palvelin taas pyrki käyttäytymään kuin jaksollinen työ)
- Kaikilla näillä budjetti kuluu vain suorituksessa

Lisää palvelimia

- ”Vakio käyttöaste” (constant utilization)
  - Palvelimen käyttöön varataan tietty käyttöaste $\tilde{u}_s$, joka on siis palvelimen koko
- ”Koko kaistanleveys” (Total bandwidth)
  - Kiinteän käyttöasteen lisäksi osaa käyttää myös muuten vapaaksi jäävän ajan
- ”Painotettu reilu odotus” (Weighted Fair-Queuing)
  - Tunnetumpi tietoliikenteestä – pakettien lähetykseen pakettikytkentäisessä verkossa
  - WFQ pystyy takaamaan reiluutta, silloin kun useita samanaikaisia palvelimia

Vakio käyttöaste -palvelin

- Täydennyssäännöt
  - Aluksi: $e_s = 0$, ja d=0
  - Kun uusi jaksoton työ e saapuu tyhjälle palvelimelle:
    - Jos saapumishetki $t <$ d, älä tee mitään
    - Jos $t \geq$ d, aseta d= t+ $e/\tilde{u}_s$, ja $e_s = e$
  - Hetkellä d (takaraja)
    - Jos palvelin on vireessä, aseta d= d+ $e/\tilde{u}_s$, ja $e_s = e$
    - Jos palvelin on vapaa, älä tee mitään

Koko kaistanleveys -palvelin

- Täydennyssäännöt
  - Aluksi: $e_s = 0$, ja d=0
  - Kun uusi jaksoton työ e saapuu tyhjälle palvelimelle:
    - aseta d= max(d, t)+ $e/\tilde{u}_s$, ja $e_s = e$
  - Kun palvelimella suoritusvuorossa ollut työ päättyy ja on poistettu jonosta
    - Jos palvelin jää vireeseen, aseta d= d+ $e/\tilde{u}_s$, ja $e_s = e$
    - Jos palvelin jää vapaaksi, älä tee mitään

WFQ (Weighted Fair-Queuing)

- Jäljittelee Generalized Processor Sharing (GPS) algoritmia
  - ideaalinen algoritmi, joka takaa kaikille suoritettaville palvelimille kullakin äärettömän lyhyellä kierroksella äärettömän pienen, mutta palvelimen kokoon suhteutetun aikaviipaleen
  - Se siis suorittaa kaikkia palvelimia mahdollisimman reilusti ja tasaisella nopeudella
- WFQ:lla on vain suurempi granulariteetti
- Tarkemmin tietoliikenneosiossa

Reiluus?

- Tosiaikaisuus ja reiluus?? Nälkiintyminen?
- Reiluutta ei tarvita priorisoitujen töiden järjestämiseen, mutta entä jos useita jaksottomien töiden palvelimia?
- Algoritmi on *reilu*, kun se takaa kaikille vireessä oleville palvelimille niiden kokoon suhteutetun osuuden prosessoriajasta tarkasteltavalla aikavälillä

Sisältö

- Yleistä
- Jaksottomien ei-tosiaikaisten töiden jaksolliset vuorotuspalvelut
  - ”Osa-aika” palvelimet (deferrable servers)
  - Sporadiset palvelimet
  - ”Vakio käyttöaste” (constant utilization)
  - ”Koko kaistanleveys” (Total bandwidth)
  - ”Painotettu reilu odotus” (Weighted Fair-Queing)
- **Sporadiset työt**
- Kaksitasoinen integroitu vuorottaminen

Järjestelmämalli

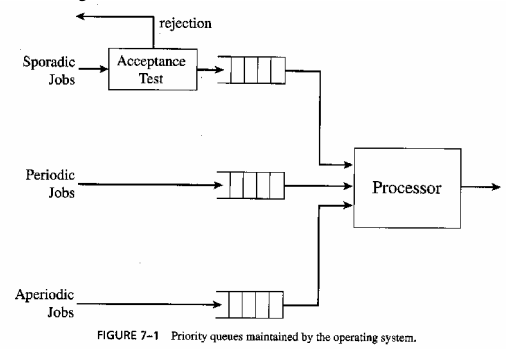


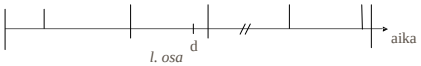
FIGURE 7–1 Priority queues maintained by the operating system.

Sporadisten töiden hyväksymis-testi kun käytössä EDF-ajoitus

- Ensimmäinen saapuva työ $S_1(t, d, e)$ 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 $\Delta_1 = e/(d-t)$ ja I_2 :n $\Delta_2 = 0$.
- 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



Integroitu ajoittaminen

- Käyttöjärjestelmän tasolla on käytössä joku ajoitusmekanismi (kuten RM tai EDF)
- Se vuorottaa palvelimia, jotka suorittavat hallinnoimiaan töitä kukin oman ajoitusmenettelynsä mukaan
- Tämänkin luennon palvelimet voisivat suorittaa useampia töitä ’samanaikaisesti’ ja vuorottaa ne haluamallaan tavalla palvelimelle annettuihin aikaviipaleisiin

Yhteenvetona:

- Käyttämällä erityisiä palvelimia epäsäännöllisille töille (sekä sporadiset että jaksottomat)
  - Taataan näille tietty osuus kapasiteetista
  - Vältetään näiden aiheuttamat haitat tärkeille jaksollisille töille ja
  - Yksinkertaistetaan koko järjestelmän ajoitettavuusanalyysiä

Yhteenveto (jatkuu)

- Tarkastellut palvelimet:
  - ”Osa-aika” palvelimet (deferrable servers)
  - Sporadiset palvelimet
  - ”Vakio käyttöaste” (constant utilization)
  - ”Koko kaistanleveys” (Total bandwidth)
  - ”Painotettu reilu odotus” (Weighted Fair-Queuing)