

# Tosiakajärjestelmät: Luento 3

## Epäsäännöllisten töiden ajoitus

Tiina Niklander  
20.3.2006 (päiv. 22.3.)

## Sisältö

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

## Järjestelmämalli

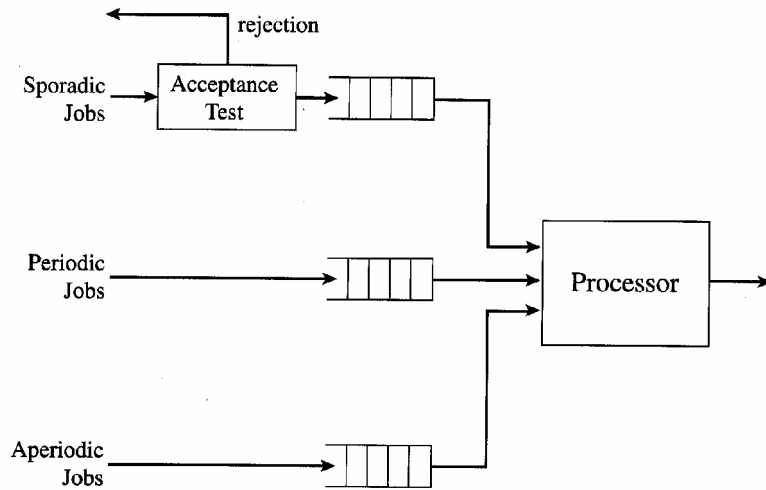


FIGURE 7-1 Priority queues maintained by the operating system.

## Sporadisten ja jaksottomien ajoitus

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

## Kellopohjainen ajoitus ja epäsäännölliset työt

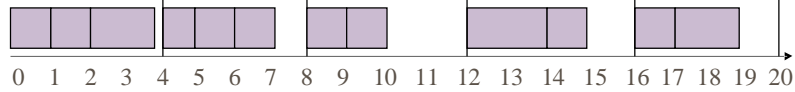
- n Työluokkien tärkeysjärjestys:
  - n Taulukoidut jaksolliset
  - n Sporadiset (näillä on aikarajat)
  - n Jaksottomat (aperiodiset, ei aikarajaa)
- n Muokataan jaksollista ajoittajaa siten, että
  - n Jaksollisilta jääneet vapaat aikajaksot käytetään sporadisille ja jaksottomille
- n Sporadisille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

## Epäsäännöllisten tehtävien ajoitus jaksollisten sekaan

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

## Esimerkki: Slack Stealing

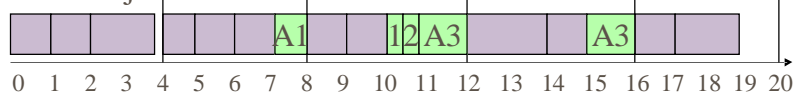
Jaksollisten ajoitus



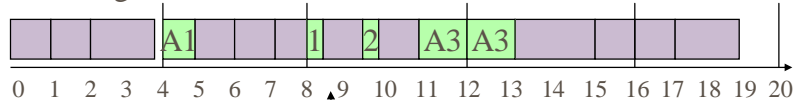
Jaksottomat tehtävät



Perusvuorottaja



Slack Stealing vuorotus



Vuoroon otetaan jaksollinen, kun jaksottomia ei ole

## Kehyksen ylitys

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

## Hyväksymistesti

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

## Hyväksymistesti

- 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  $\sigma$  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 järjestyksessä

## Hyväksymistesti: tarvittava tieto

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

## Hyväksymistestin 'algoritmi'

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

## Hyväksymistesti: A)

- n Kehyksien 'vapaat ajat' yli kehysjaksojen eli  $\sigma(i,j) \quad \forall i,j: i \leq j$
- n Kaikkien jo hyväksytyjen töiden tiedot eli  $\forall k \leq n_s \quad S_k(d_k, e_k)$
- n Kaikista jo aloitetuista töistä tähänastisen suorituksen kesto  $\xi_k$
- n Näillä voidaan laskea  $\sigma_c(t,l)$

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

## Hyväksymistesti: B)

- n Työt, joiden aikaraja tarkasteltavan jälkeen  $\forall k \leq n_s \text{ ja } d_k > d \quad S_k(d_k, e_k)$
- n Näiden suoritusmahdollisuudesta poistetaan  $e$ :n verran aikaa eli  $\sigma_k' = \sigma_k - e$
- n Uusi työ voidaan hyväksyä vain kun  $\sigma_k' \geq 0$

## Prioriteettipohjainen ajoitus ja epäsäännölliset työt

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

## Vaihtoehtoiset ajoitusmekanismit

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



## Sisältö

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

## Terminologiaa

- n Jaksollinen palvelin (periodic server)  $S(p_s, e_s)$ 
  - n  $e_s$  on suorituskiintiö eli budjetti (execution budget)
  - n Palvelimen *koko* on sen osuus käyttöasteesta eli
$$\tilde{u}_s = e_s / p_s$$
  - n Suorituskiintiö *täydennetään* aina jakson alussa
  - n Palvelin on *vireessä* aina, kun sillä on jonossa suoritettavia töitä
  - n Palvelin on valmiina suoritukseen, kun se on vireessä ja sillä on kiintiötä vielä jäljellä
  - n Kiintiötä kuluu suorituksen aikana yksi per aikayksikkö

## Jaksollisista palvelimista

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

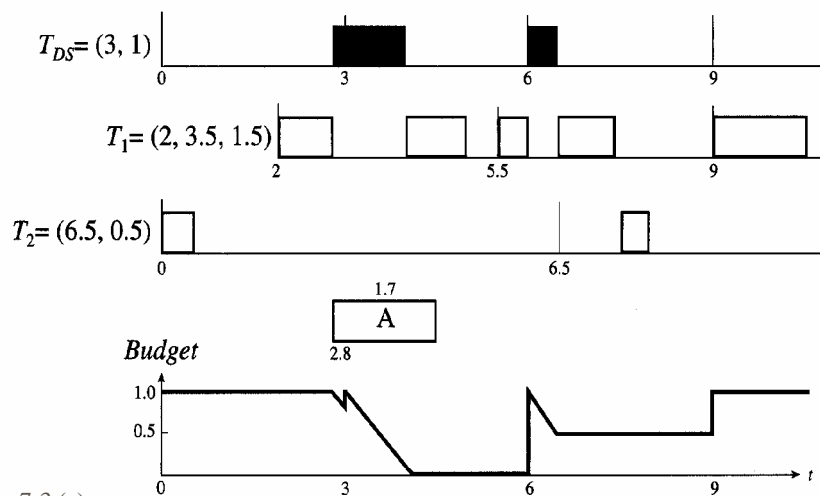
## ”Osa-aika” palvelin – Deferrable Server

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

## Osa-aikapalvelin: Kulutus- ja täydennyssäännöt

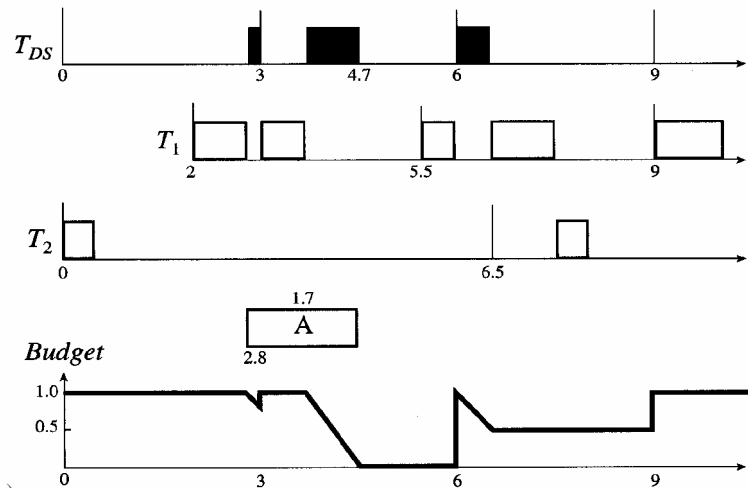
- n Kulutussääntö (Consumption Rule):
  - n Palvelimen suoritusaikaa kuluu yksi yksikkö kutakin suoritettua aikayksikköä kohti
- n Täydennyssääntö (Replenishment Rule):
  - n Palvelimen suoritus aika palautetaan maksimiin aina kunkin suoritusjakson aluksi eli budjetiksi asetetaan  $e_s$  aina kellon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

## Esimerkki (RM-menetelmä)



Kuva 7-3 (a)

## Esimerkki (EDF-menetelmä)



Kuva 7-3 (b)

## Miten määrätään palvelimen suorituskiintiö?

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

## Edellinen 'kriittinen tilanne'

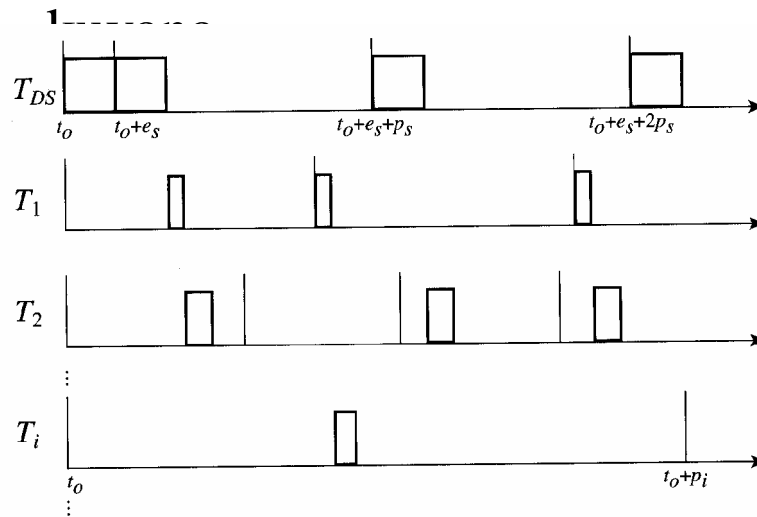


FIGURE 7-5 A segment of an arbitrary fixed-priority schedule after a critical instant of  $T_i$ .

## Aikavaatimusanalyysi (RM)

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

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

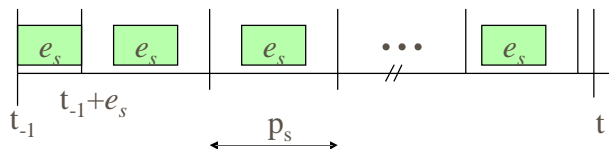
- n Oletus: osa-aikapalvelimen prioriteetti on suurin ja jakson pituus siis pienin

## 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

$$e_s + \left\lceil \frac{t - t_{-1} - e_s}{p_s} \right\rceil e_s$$

- 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  $n$  riippumaton keskeytettävää jaksollista tehtävää ja osa-aikapalvelin, jonka jakso on  $p_s$ , suorituskäyttö  $e_s$  ja käyttöaste  $u_s$ , jos

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

Koko järjestelmä

Jaksolliset työt, joiden suhteellinen aikaraja ennen  $T_i$

osa-aikapalvelin

(Kirjan Teoreema 7.3)

## Sporadinen palvelin (kiinteä prio)

- n 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.
- n Sporadinen palvelin käyttäytyy siivommin tilanteessa, jossa osa-aikapalvelin varaa suorittimen kaksinkertaisen kiintiön ajaksi

## 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 täydennys hetki säilyy  $t_r + p_s$
  - n jos palvelin oli vapaa, niin täydennys hetkeä 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  $\min(t_e + p_s, t_b)$

## Sporadinen palvelin: esimerkki

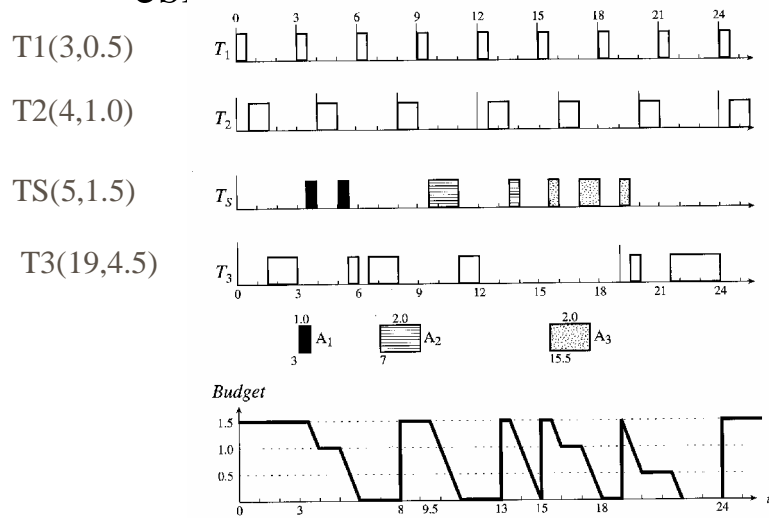


FIGURE 7-8 Example illustrating the operations of a simple sporadic server:  $T_1 = (3, 0.5)$ ,  $T_2 = (4, 1.0)$ ,  $T_3 = (19, 4.5)$ ,  $T_5 = (5, 1.5)$ .



## Lisää piirteitä

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

## Sporadinen/tausta-ajo palvelin EDF-ajoitettussa järjestelmässä (7.3.3)

- n Palvelimella itsellään on myös oltava aikaraja  $d$ 
  - n Aikaraja asetetaan, vain jos palvelimella on töitä jonossa, muuten sen arvo on määräämättä
- n Kulutussäännöt: Kiintiötä kuluu vain kun jompikumpi sääntö pätee, muuten kiintiö säilyy
  - n Palvelin on suorituksessa
  - n 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-ajoitettu järjestelmässä

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

## Sporadinen/tausta-ajo palvelin EDF-ajoitettu järjestelmässä

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

## Sporadinen/tausta-ajo palvelin EDF-ajoitettussa järjestelmässä

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

## Lisää palvelimia

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

## Lisää palvelimia

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

## Vakio käyttöaste -palvelin

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

## Koko kaistanleveys -palvelin

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

## WFQ (Weighted Fair-Queuing)

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

## Reiluus?

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

## Sisältö

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

# Järjestelmämalli

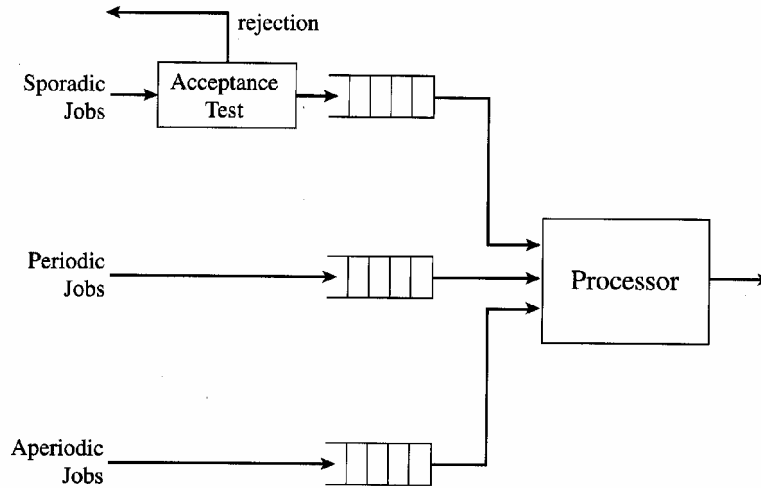


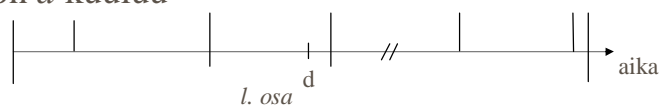
FIGURE 7-1 Priority queues maintained by the operating system.

## Sporadisten töiden hyväksymis-testi kun käytössä EDF-ajoitus

- n Ensimmäinen saapuva työ  $S_1(t, d, e)$  hyväksytään, jos  $e/(d-t) \leq 1-\Delta$ , missä  $\Delta$  on kaikkien sporadisten töiden sallittu yhteinen maksimitiheys
- n HUOM:  $d$  jakaa aikavälin kahteen osaan  $I_1$  ja  $I_2$ .  $I_1$  :n tiheys  $\Delta_1 = e/(d-t)$  ja  $I_2$  :n  $\Delta_2 = 0$ .
- 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



## Integroitu ajoittaminen

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

## Yhteenvetona:

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



## Yhteenveto (jatkuu)

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