

582425 Tosi aikajärjestelmät (3ov) Luento 2 – Ajoittaminen: RM & EDF

Tiina Niklander

Sisältö

- n **Johdanto ja termistöä**
 - n Dynaaminen, staattinen, ylikuormitus
- n **Ajoituksen perusmenetelmiä**
 - n Kello-ohjattu
 - n Esim. staattinen taulukkopohjainen, jaksollinen
 - n Prioriteettiperustainen
 - n Esim. RM, EDF
- n **Ajoitettavuusanalyysi**

Ajoittaminen

- n Ajoitus on *toteuttamiskelpoinen* (feasible), jos se toteuttaa kaikki sovelluksen rajoitteet (constraint) annetulle joukolle tapahtumia.
- n Joukko tapahtumia on *ajoituskelpoinen* (schedulable), jos on olemassa vähintään yksi ajoitus algoritmi, joka pystyy muodostamaan toteuttamiskelpoisen ajoituksen.
- n Ajoitus algoritmi on *optimaalinen* (optimal) ottaen huomioon ajoituskelpoisuuden, jos se pystyy aina löytämään toteuttamiskelpoisen ajoituksen tilanteessa, missä jokin toinenkin algoritmi pystyy siihen.

Dynaaminen vai staattinen ajoitus

- **Dynaaminen sallii**
 - uusia töitä
 - vaihtelevan suoritusjärjestyksen
- **Dynaamisuuden ongelmana on**
 - varmennettavuus ja
 - ennustettavuus
- **Staattinen kiinnittää**
 - työt
 - töiden ajoituksen etukäteen
- **Staattisuutta käytetään kriittisissä järjestelmissä, koska**
 - matemaattisesti todennettavissa

Staattisen ajoituksen rajoitukset:

- n **Keskeyttämätön ajoittaminen (non-preemptive)**
 - n Tapahtuman käynnistyttyä tapahtumaa ei voi keskeyttää (preempt) toinen tapahtuma.
- n **Ahne ajoittaja (greedy)**
 - n Tapahtuman käynnistyminen estää muiden, esim. alemman prioriteetin tapahtumien ajon.
- n **Ei prosessorin jakoa**
 - n Prosessori voi suorittaa vain yhtä tapahtumaa kerrallaan.
- n **Ei dynaamista tapahtumien rinnakkaisuutta**
 - n Tapahtumaa voidaan suorittaa vain yhdellä prosessorilla kerrallaan
- n **Ei tapahtuman siirtoa (migration)**
 - n Tapahtuma ei voi vaihtaa prosessoria kesken suorituksen.

Ylikuormitus

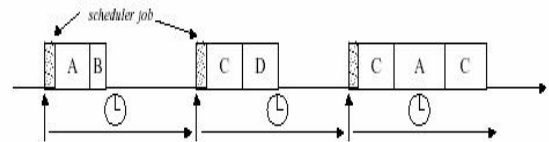
- n **Mikä on ylikuormitus?**
 - n Kun ajoittaja, edes ennustavasti, ei voi tehdä toteuttamiskelpoista ajoitusta
- n **Miten toivutaan?**
 - n Osa tehtävistä on jätettävä suorittamatta
- n **Suoritusaikaiset ajoittajat eivät selviä ylikuormitustilanteista kovinkaan hyvin.**

Ajoituksen perusmenetelmät

- n Kello-ohjattu
 - n suoritus etenee kellon määräämässä tahdissa
 - n Yleensä staattinen ajoitus, mutta voi olla myös staattinen vain yhden jakson ajan
- n Prioriteettipohjainen ajoitus
 - n Tehtävät suoritetaan vuorotellen, suoritusvuoroon valitaan se, jolla korkein prioriteetti
 - n Prioriteetti voi olla staattinen tai dynaaminen, ajoitus on dynaaminen

Kello-ohjatut ajoittajat (clock-driven schedulers)

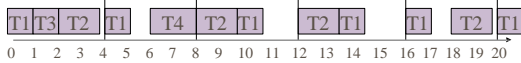
- Ajoituspäätökset tehdään tiettyinä määriteltynä aikahetkinä, jotka on tyypillisesti valittu etukäteen.
- Esimerkki: Ajoittaja herää tietyn ajan välein ja muodostaa osan ajoituksesta.



Staattinen taulukkopohjainen ajoitus

| | d | e |
|----|----|-----|
| T1 | 4 | 1 |
| T2 | 5 | 1.8 |
| T3 | 20 | 1 |
| T4 | 20 | 2 |

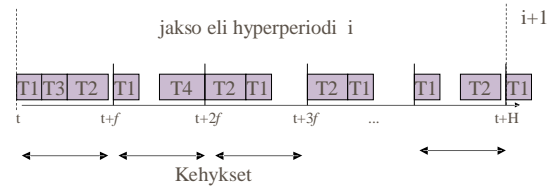
- Etukäteen tunnetut jaksolliset tehtävät
- Ajoitustaulukko etukäteen koko hyperperiodille, jota sitten toistetaan



Taulukon alkio: (0,T1),(1,T3),(2,T2),(3,8,I),(4,T1),....,(19,8,I)

Jaksollinen ajoitus

- n Ajoituspäätökset säännöllisin väliajoin
- n Kehys f (kahden päätöshetken väli)
- n Kehyksen koko?



Kehyksen koko:

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

- n Alaraja - riittävän suuri
 - n Yksittäinen työ mahtuu kokonaan kehykseen
 - n Tällöin ei tarvita keskeytyksiä (preemption)
- n Yläraja
 - n Ajoittajan toiminnan varmistamiseksi on kunkin työn aloitusajan (t') ja aikarajan (D_i) väliin mahdollista aina kokonainen kehys.

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

Kehysten määrä

- n Kehysten lukumäärän on oltava tasan jaollinen yhden jakson sisällä
 - n Kehysraja ja jakson raja osuvat samaan kohtaan
 - n Jakson pituuden pitää olla joku hyperperiodi H :n tekijöistä (eli jonkun tehtävän T_i jakso p_i on tasan jaollinen f :llä)

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

Kehyksen koon määrittäminen

| | d | e |
|----|----|-----|
| T1 | 4 | 1 |
| T2 | 5 | 1.8 |
| T3 | 20 | 1 |
| T4 | 20 | 2 |

- Hyperperiodi $H = 20$
- Alaraja: $f \geq \max(1, 1.8, 1, 2)$
- Kokovaihtoehdot 2,4,5,10,20
- Yläraja: $2f\text{-syt}(p_i, f) \leq D_i$
ainoastaan $f=2$ käy



Muutos staattiseen taulukkoon verrattuna on, että nyt työt alkavat kehysrajoilta

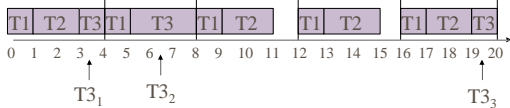
Staattinen jaksollinen ajoitus

- n Kehyksen koon määrittäminen
 - n kaavat edellä
- n Töiden ositus
 - n joskus tarkoituksenmukaista
- n Töiden sijoittelu aikajanelle
 - n tämä määrää suoritusjärjestyksen
- n Töiden sijoittelusta syntyy ajoitustaulu

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 $\Rightarrow f=4$**



Jaksollinen ajoitus (esim 3)

- n Tapahtumat (4,0.5), (5,1.0), (10,2), ja (24,9)
- n Hyperperiodi $4=2*2, 5, 10=5*2, 24=2^3*3$ siis hyperperiodi $H=2^3*3*5=120$
- n Tästä joukosta voidaan suoraan veikata, että pisin työ joudutaan sijoittamaan useampaan kehykseen (epävirallinen perustelu $9 > 4$), koska
- n Tavoitteena on löytää sellainen kehyskoko, että ainakin yksi kehys mahtuu jokaisen työn aloituksen ja lopetusrajan sisään (eli pitää olla varmasti ≤ 4)

| | p | e |
|----|----|-----|
| T1 | 4 | 0.5 |
| T2 | 5 | 1.0 |
| T3 | 10 | 2 |
| T4 | 24 | 9 |

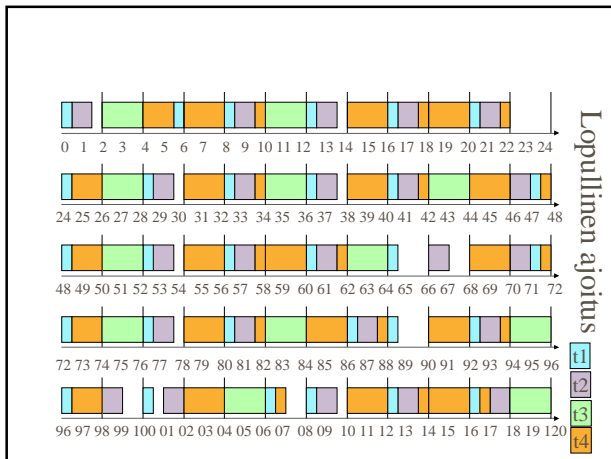
H=120

Esimerkki 3 jatkuu

- n Unohdetaan tuo suurin hetkiseksi ja tutkitaan vain joukkoa (4,0.5), (5,1), ja (10,2)
- n Nyt mahdollinen vaihtoehto on vain 2
- n Kehyksen yläraja (koko 3): $2*3\text{-pyj}(4,3) = 2*3-1 = 5$ joka ei ole pienempi kuin 4
- n Kehyksen yläraja (koko 2): $2*2\text{-pyj}(4,2) = 2$
- n Siis käytetään kehyksen kokoa 2!
- n Kehyksen koko 4 ei toimi työlle (5,1)

Esim 3: Suurimman työn osittaminen

- n Työ (24,9) täytyy jakaa osiin ja sallia sen 'keskeyttäminen'. Osituksessa pyritään mahdollisimman pieneen määrään paloja.
- n Suurin osan koko on 2, koska se on kehyksen koko.
- n Ositus 7:ään osaan on ilmeisesti toimivin vaihtoehto: 1,5+2+0,5+2+0,5+2+0,5



- ### Sisältö
- n Johdanto ja termistöä
 - n Ajoituksen perusmenetelmiä
 - n Kello-ohjattu
 - n Esim. staattinen taulukkopohjainen, jaksollinen
 - ➔ n **Prioriteettiperustainen**
 - n Rate-monotonic, Deadline-monotonic
 - n Earliest-deadline-first, Least-slack-time
 - n Ajoitettavuusanalyysi

- ### Prioriteettiperustaiset ajoittajat
- n Ajoituspäätökset tehdään tiettyjen tapahtumien herätteen perusteella
 - n Tapahtuma tulee järjestelmään
 - n Prosessori tulee vapaaksi
 - n Prosessori on kiireinen aina, kun on työtä tehtäväksi.
 - n Esimerkkejä: FIFO, LIFO, EDF

- ### Prioriteettiperustaisten luokittelu
- n Staattinen prioriteetti (sekä tehtävä että työ)
 - n Rate-monotonic (RM)
 - n Deadline-monotonic (DM)
 - n Dynaaminen prioriteetti
 - n EDF - mutta yksittäisellä työllä staattinen
 - n LST - sekä työllä että tehtävällä dynaaminen

- ### Rate-monotonic (RM)
- n Käyttää staattisia prioriteetteja
 - n Prioriteetti määräytyy tapahtuman taajuudesta (frequency)
 - n Tapahtumat, joilla on lyhyemmät jaksot saavat korkeamman prioriteetin.
 - n Teoreettisesti hyvin tutkittu (1 prosessorille)
 - n Riittävä ajoitettavuudesta voidaan suorittaa lineaarisessa ajassa (tietyn ehdoin).
 - n Tarkka ajoitettavuudesta on NP-täydellinen ongelma.
 - n Optimaalinen verrattuna kaikkiin staattista prioriteettia käyttäviin menetelmiin, jos aikaraja on sama kaikilla tapahtumilla.

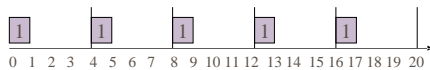
- ### Rate monotonic
- Ajoittaa suoritusvuoroon aina sen, jonka prioriteetti (eli taajuus) on korkein
- | | d | e |
|----|----|---|
| T1 | 4 | 1 |
| T2 | 5 | 2 |
| T3 | 20 | 5 |
-

Rate monotonic

- Ajoittaa suoritusvuoroon aina sen, jonka prioriteetti (eli taajuus) on korkein

| | d | e |
|----|----|---|
| T1 | 4 | 1 |
| T2 | 5 | 2 |
| T3 | 20 | 5 |

- Ensin sijoitellaan T1

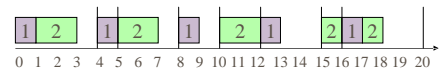


Rate monotonic

- Ajoittaa suoritusvuoroon aina sen, jonka prioriteetti (eli taajuus) on korkein

| | d | e |
|----|----|---|
| T1 | 4 | 1 |
| T2 | 5 | 2 |
| T3 | 20 | 5 |

- Ensin sijoitellaan T1
- Sitten T2 vapaisiin kohtiin



Rate monotonic

- Ajoittaa suoritusvuoroon aina sen, jonka prioriteetti (eli taajuus) on korkein

| | d | e |
|----|----|---|
| T1 | 4 | 1 |
| T2 | 5 | 2 |
| T3 | 20 | 5 |

- Ensin sijoitellaan T1
- Sitten T2 vapaisiin kohtiin
- Lopuksi T3

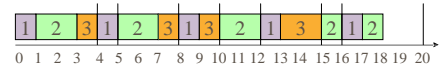


Rate monotonic

- Ajoittaja toimiessaan valitsee kullakin tapahtumahetkellä suoritukseen sen, jonka prioriteetti (eli taajuus) on korkein

| | p/d | e |
|----|-----|---|
| T1 | 4 | 1 |
| T2 | 5 | 2 |
| T3 | 20 | 5 |

- Esim. ajanhetkellä 16: kun T1 tulee suoritukseen, se keskeyttää T2:n suorituksen.

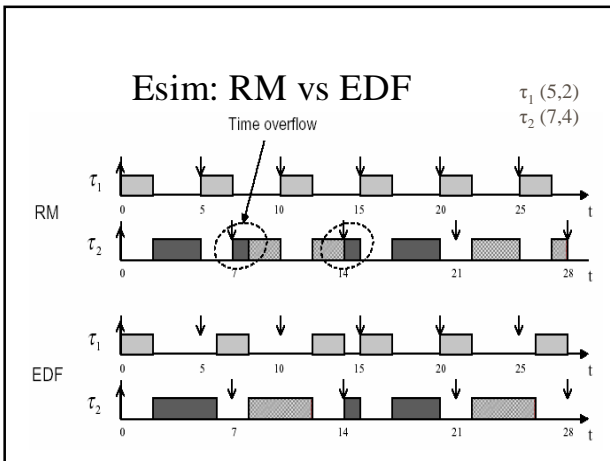


Deadline monotonic (DM)

- Algoritmi kuten rate monotonic, mutta
- Prioriteetiksi asetetaan työn suhteellinen aikaraja, jolloin se, jonka aikaraja on pienin pääsee ensin suoritukseen.
- RM ja DM ovat identtiset, jos kaikkien tapahtumien aikarajat ovat niiden jaksojen mittaiset.
- Esim tapahtumajoukko $T_1(50,50,25,100)$, $T_2(0,62.5,10,20)$ ja $T_3(0,125,25,50)$ on ajoitettavissa DM, mutta ei RM-menetelmällä

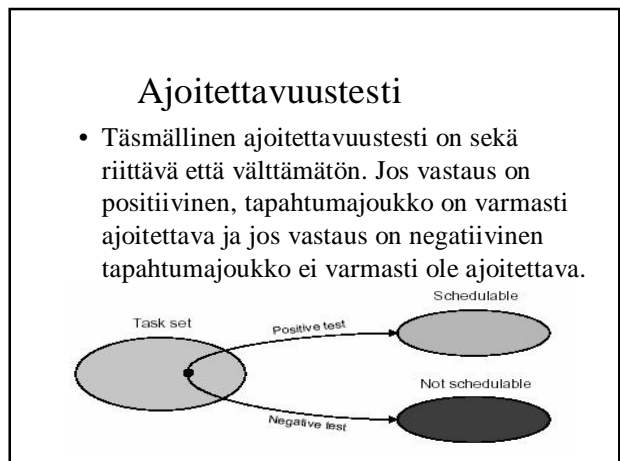
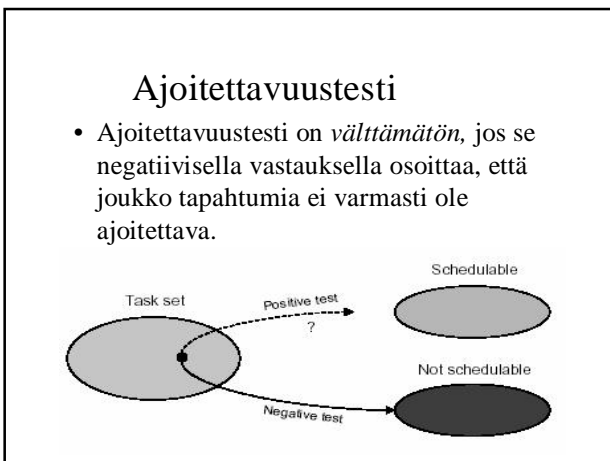
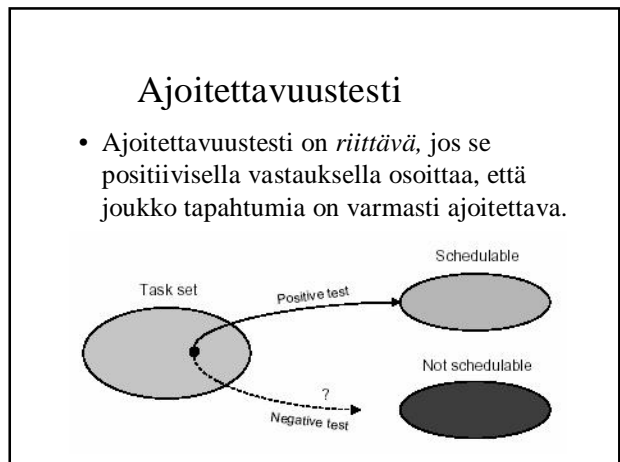
Earliest-deadline-first (EDF)

- EDF (Earliest Deadline First) –algoritmi käyttää dynaamista prioriteettia
 - Työ, jonka absoluuttinen aikaraja on lähinnä saa korkeimman prioriteetin.
- Teoreettisesti hyvin tutkittu (1 prosessorille)
 - Täsmällinen ajoitettavuudesta voidaan suorittaa lineaarisessa ajassa (tietyin ehdoin).
- EDF on optimaalinen verrattuna kaikkiin dynaamisiin prioriteettien käyttöihin menetelmiin, kun
 - yksi prosessori,
 - keskeyttävä suoritus sallitaan ja
 - ei kilpailua resursseista



- ### LST – Least Slack Time
- n LST on EDF:lle käänteinen; optimaalinen samoin rajoituksin
 - n Ajoitetaan vuoroon se, joka voi odottaa huonoimmin
 - n Vaatii
 - n tarkan tiedon laskennan kestosta
 - n joka hetki laskemaan kulloisenkin viivytelyajan aina kaikille töille

- ### Ajoitettavuusanalyysi
- n Tavoitteena selvittää, voidaanko annettu joukko tapahtumia ajoittaa annetulla suoritusajaisella ajoittajalla siten, että kaikki tapahtumien ilmentymät päättyvät aikarajoihin mennessä.
 - n Ajoitettavuusanalyysi tyypillisesti sisältää ajoitettavuustestin, joka on sovitettu käytettävälle ajonajaisella ajoittajalle.



Ajoitettavuustestin menetelmiä

- n Prosessorin käyttöasteen analyysi
 - n prosessoriajan (murto-)osa, joka käytetään tapahtumajoukon suoritukseen ei saa ylittää tiettyä rajaa.
 - n Sopii staattisen/dynaamisen prioriteetin ajoittajiin yksiprosessorijärjestelmässä.
- n Vastausaika-analyysi
 - n Jokaisen tapahtuman pahimman tapauksen suoritusaikaa verrataan tapahtuman aikarajaan
 - n Sopii staattisen prioriteetin ajoittajiin (kuten RM).
- n Prosessorin vaatimusanalyysi
 - n Yhteenlaskettu prosessointivaatimus tapahtumajoukolle tietyssä aikavälissä ei saa ylittää aikaväliä.
 - n Vain EDF-menetelmälle yksiprosessorijärjestelmässä.

Prosessorin käyttöaste

- Käyttöaste U joukolle periodisia tapahtumia on se murto-osa prosessorikapasiteetista, joka käytetään tapahtumien suoritukseen.
- Koska e_i/p_i on se osa, joka käytetään yhden tapahtuman T suoritukseen, $n:n$ tapahtuman yhteenlaskettu käyttöaste on:

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

Ajoitettava käyttöaste (schedulable utilization)

- n Ajoitusmenetelmä pystyy ajoittamaan, minkä tahansa jaksollisen työjoukon, kunhan niiden yhteenlaskettu käyttöaste on menetelmän ajoitettavaa käyttöastetta pienempi (tai yhtä suuri).
- n Kaikille ajoitusmenetelmille voidaan laskea ajoitettava käyttöaste, jolla saadaan tuo yo. tae. Se on riittävä ehto, mutta ei aina välttämätön.

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:

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

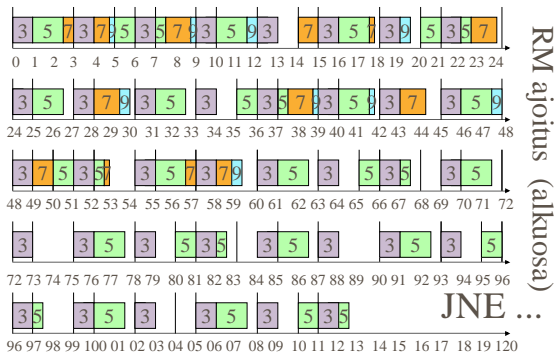
Esimerkki: ajoitettavuustesti

- n Tehtäväjoukko
 - n (3,1), (5,1.5), (7,1.25) ja (9,0.5)
- n Käyttöaste
 - n $1/3 + 1.5/5 + 1.25/7 + 0.5/9 = 0.85$

| | d | e |
|----|---|------|
| T1 | 3 | 1 |
| T2 | 5 | 1.5 |
| T3 | 7 | 1.25 |
| T4 | 9 | 0.5 |

- n $0.85 < 1 \Rightarrow$ On ajoitettavissa EDF:llä
- n $0.85 > 4(2^{1/4} - 1) = 0.757 \Rightarrow$ RM ? ehkei – kokeillaan (tai tehdään toinen analyysi)

(3,1), (5,1.5), (7,1.25) ja (9,0.5)



Aikavaatimusanalyysi (Time-Demand Analysis)

- Tarkastellaan yksi kerrallaan töitä T_i korkeimmasta prioriteetista alimpaan

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

- Työt voidaan ajoittaa, kun $\forall i$

$$\exists w_i(t) \leq t, \text{ jollekin } t \leq d_i \leq p_i$$

Aikavaatimusanalyysi

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

Maksimivasteajat
(kuvasta, merkitty
pienellä pallolla)

T1 1
T2 2.5
T3 4.75
T4 9

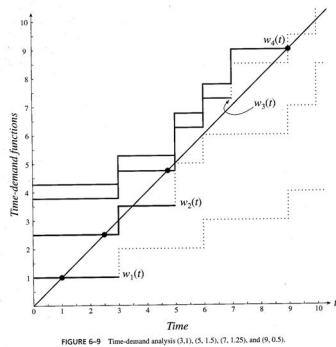


FIGURE 6-9 Time-demand analysis (3,1), (5, 1.5), (7, 1.25), and (9, 0.5).

Analyysin ongelma:

Suorituksen estyminen

- Alemman prioriteetin työ voi
 - Estää keskeyttämisenä (esim. systeemikutsu)
 - Käyttää poissulkevasti jaettua resurssia
- Korkeamman prioriteetin työ joutuu odottamaan
 - => ns. prioriteetin kääntyminen
- Estetty korkeamman prioriteetin työ saattaa myös ylittää aikarajansa odotuksen vuoksi
 - Siksi ajoitettavuusanalyseissä tarkasteltava myös matalamman prioriteetin töiden keskeyttämättömien suoritusjaksojen kestoja

Estyminen analyysin kannalta

- Korkeamman prioriteetin työn voi estää vain yksi alempi ennen sen pääsyä suoritukseen
- Joten lisätään arvioissa korkeamman prioriteetin työn suoritusajaksi alempien estoajien maksimi. Työn pisin estymisaika (blocking time) on tällöin

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

Kun työt järjestetty prioriteettien mukaan ja θ_i on yhden eston kesto

Estyminen ja EDF

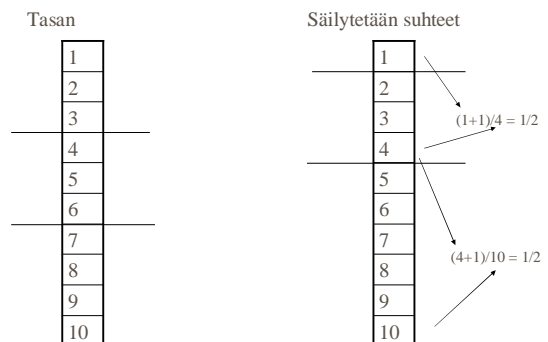
- Työ $J_k(.,d_k)$ voi estää työtä $J_i(.,d_i)$ vain jos $d_k > d_j$
 - Ainoastaan tällöin sen prioriteetti on pienempi eli aikaraja on myöhäisempi.
 - Toisaalta sen on täytyy olla jo suorituksessa, jotta estäminen olisi mahdollista, eli sen aloitusajan täytyy olla aikaisempi.

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

Prioriteettiluokkien määrä

- n Analyysit olettavat ääretöntä määrää prioriteetteja
 - n Samalla arvolla ei ole 'eriarvoisia' töitä
- n Käytännössä prioriteettiluokkien määrä rajoitettu – joskus 8, usein 32, 127 tai 255.
- n Miten työt sijoitellaan rajalliseen prioriteettiavaruuteen?
 - n jaetaan tasan
 - n säilytetään suhteet

Prioriteettien jako



Yhteenvetona:

Kello-ohjatut staattiset ajoitukset

- n Hyödyt:
 - n Ei tarvita rinnakkaisuuden hallintaa erikseen
 - n Voidaan määrittää ja arvioida etukäteen
- n Haitat:
 - n Kaikkien töiden aloitusajat on kiinnitettävä etukäteen
 - n Koko työkuorma on tunnettava ennakoon, ei salli dynaamisuutta
 - n Tukee suhteellisen huonosti sekakuormia
- n Erinomainen, jos yo. haitat eivät ole este

Yhteenvetona:

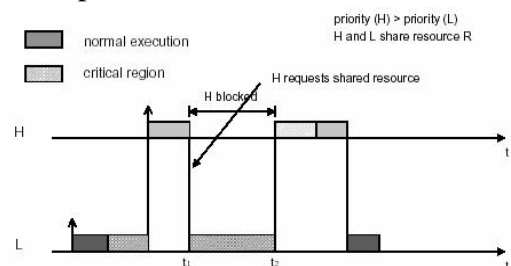
Prioriteettipohjaiset ajoitukset

- n Algoritmit: RM, DM, EDF ja LST
- n Ajoitettavuustesti:
 - n EDF: laskennallinen käyttöaste ≤ 1
 - n RM: laskennallinen käyttöaste $\leq \approx 0.693$
- n RM:n tapauksessa ehto on riittävä, mutta ei välttämätön
- n EDF:lle ehto on sekä riittävä että välttämätön

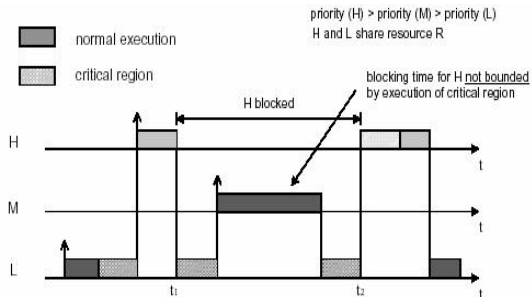
Mitä voi mennä pieleen ?

- n Resurssit, kriittiset alueet, odotus
- n Prioriteetin kääntyminen, lukkiumat
- n Keskeytymättömät kriittiset alueet

Odotusongelma (Blocking problem)



Prioriteetin kääntäminen (Priority inversion)

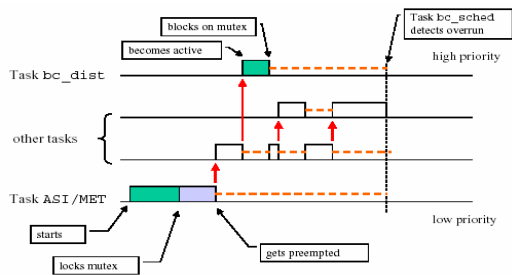


Mars Pathfinder



- Laskeutui 4.7.1997
- Kohtaa ohjelmisto-ongelmia (software glitches).
- Pathfinder kokee toistuvia RESET:jä meteorologisen tiedon keruussa.
- Prioriteetin kääntymisestä aiheutuvia ajoitusten ylityksiä.
- http://research.microsoft.com/~mbj/Mars_Pathfinder/Mars_Pathfinder.html

Prioriteetin kääntäminen Mars Pathfinder



Prioriteetin kääntymisen välttäminen

- Keskeyttämättömät kriittiset alueet
 - Luovat tarpeetonta odotusta.
 - Käyttökelpoisia vain **lyhyille** kriittisille alueille.
- Sisääntuloprotokollia kriittiselle alueelle
 - Prioriteetin perintä (Priority Inheritance Protocol).
 - Prioriteetin kattomenetelmä (Priority Ceiling Protocol).