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

Tiina Niklander

Sisältö

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

Ajoittaminen

- Ajoitus on *toteuttamiskelpoinen* (feasible), jos se toteuttaa kaikki sovelluksen rajoitteet (constraint) annetulle joukolle tapahtumia.
- Joukko tapahtumia on *ajoituskelpoinen* (schedulable), jos on olemassa vähintään yksi ajoitusalgoritmi, joka pystyy muodostamaan toteuttamiskelpoisen ajoituksen.
- Ajoitusalgoritmi 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:

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

Ylikuormitus

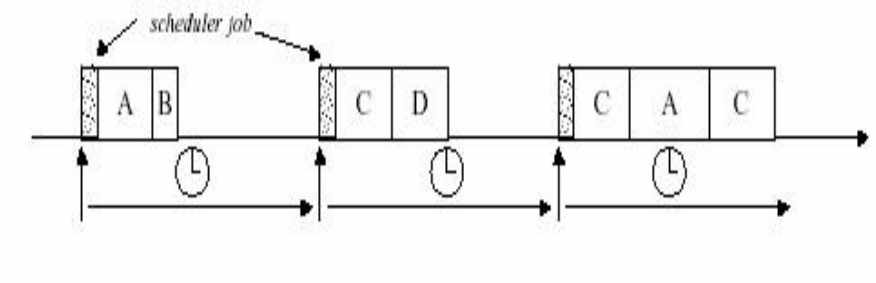
- Mikä on ylikuormitus?
  - Kun ajoittaja, edes ennustavasti, ei voi tehdä toteuttamiskelpoista ajoitusta
- Miten toivutaan?
  - Osa tehtävistä on jätettävä suorittamatta
- Suoritusaikaiset ajoittajat eivät selviä ylikuormitustilanteista kovinkaan hyvin.

Ajoituksen perusmenetelmät

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

Kello-ohjatut ajoittajat (clock-driven schedulers)

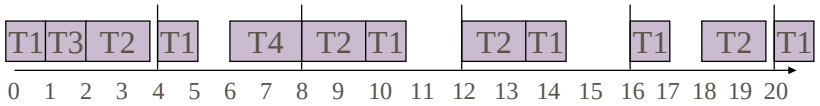
- Ajoituspäätökset tehdään tiettyinä määriteltyinä aikahetkinä, jotka on tyypillisesti valittu etukäteen.
- Esimerkki: Ajoittaja herää tietyn ajan välein ja muodostaa osan ajoituksesta.



Staattinen taulukkopohjainen ajoitus

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

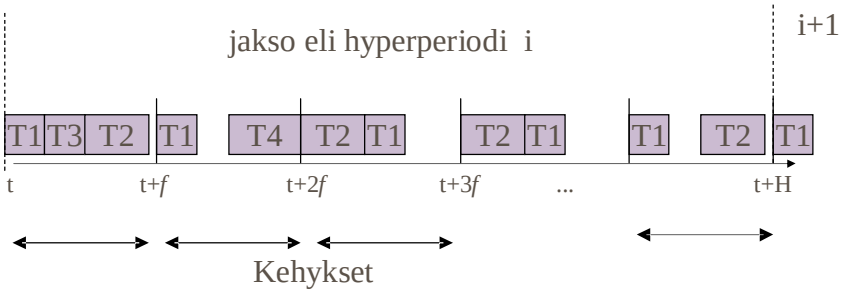
- Etukäteen tunnetut jaksolliset tehtävät
- Ajoitustaulukko etukäteen koko hyperperiodille, jota sitten toistetaan



Taulukon alkiot: (0,T1),(1,T3),(2,T2),(3.8,I),(4,T1),...,(19.8,I)

Jaksollinen ajoitus

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



Kehyksen koko:

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

- Alaraja - riittävän suuri
  - Yksittäinen työ mahtuu kokonaan kehykseen
  - Tällöin ei tarvita keskeytyksiä (preemption)
- Yläraja
  - Ajoittajan toiminnan varmistamiseksi on kunkin työn aloitusajan (t´) ja aikarajan (D_i) väliin mahduttava aina kokonainen kehys.

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

Kehysten määrä

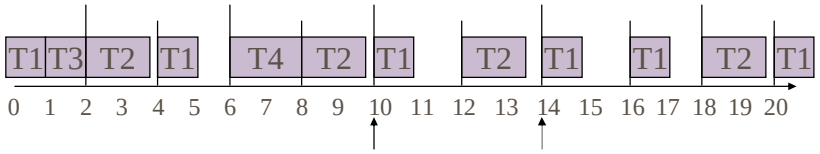
- Kehysten lukumäärän on oltava tasan jaollinen yhden jakson sisällä
  - Kehysraja ja jakson raja osuvat samaan kohtaan
  - Jakson pituuden pitää olla joku hyperperiodi H:n tekijöistä (eli jonkun tehtävän T_i jakso p_i on tasan jaollinen *f*:llä

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

Kehyksen koon määrääminen

	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

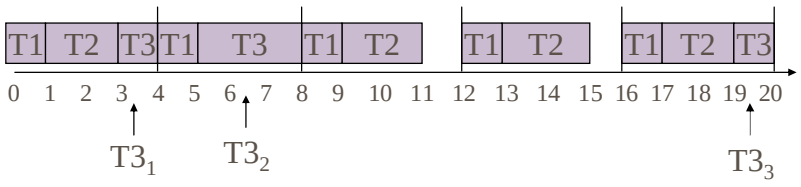
Staattinen jaksollinen ajoitus

- Kehyksen koon määritys
  - kaavat edellä
- Töiden ositus
  - joskus tarkoituksenmukaista
- Töiden sijoittelu aikajanalle
  - tämä määrää suoritusjärjestyksen
- Töiden sijoittelusta syntyy ajoitustaulu

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: $2f$-syt(p_i,f) $\leq D_i : f \leq 4$
- T3 jaettava osiin ! 1,3,1 => f=4



Jaksollinen ajoitus (esim 3)

- Tapahtumat (4,0.5), (5,1.0), (10,2), ja (24,9)
- Hyperperiodi 4=2*2, 5, 10=5*2, 24=2^3*3 siis hyperperiodi H= 2^3*3*5=120
- Tästä joukosta voidaan suoraan veikata, että pisin työ joudutaan sijoittamaan useampaan kehykseen (epävirallinen perustelu 9 > 4), koska
- 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

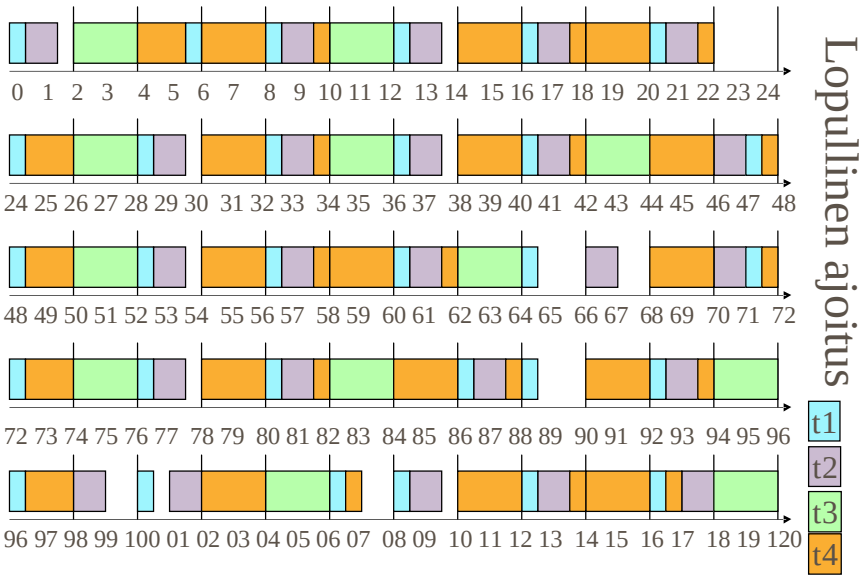
H=120

Esimerkki 3 jatkuu

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

Esim 3: Suurimman työn osittaminen

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



Sisältö

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

Prioriteettiperustaiset ajoittajat

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

Prioriteettiperustaisten luokittelu

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

Rate-monotonic (RM)

- Käyttää staattisia prioriteetteja
  - Prioriteetti määräytyy tapahtuman taajuudesta (frequency)
  - Tapahtumat, joilla on lyhyemmät jaksot saavat korkeamman prioriteetin.
- Teoreettisesti hyvin tutkittu (1 prosessorille)
  - Riittävä ajoitettavuustesti voidaan suorittaa lineaarisessa ajassa (tietyin ehdoin).
  - Tarkka ajoitettavuustesti on NP-täydellinen ongelma.
  - 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

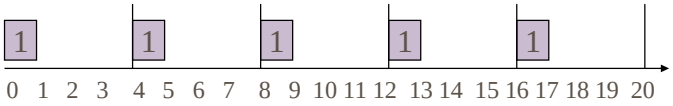
	d	e
T1	4	1
T2	5	2
T3	20	5

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

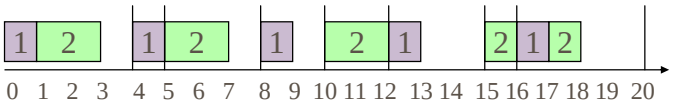


Rate monotonic

- 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

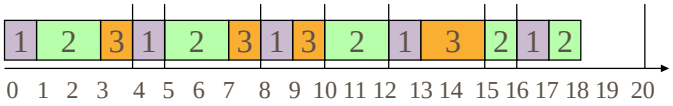


Rate monotonic

- 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

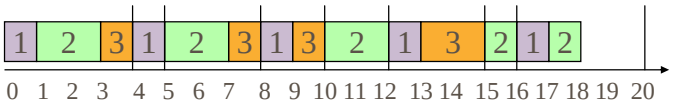


Rate monotonic

- 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)

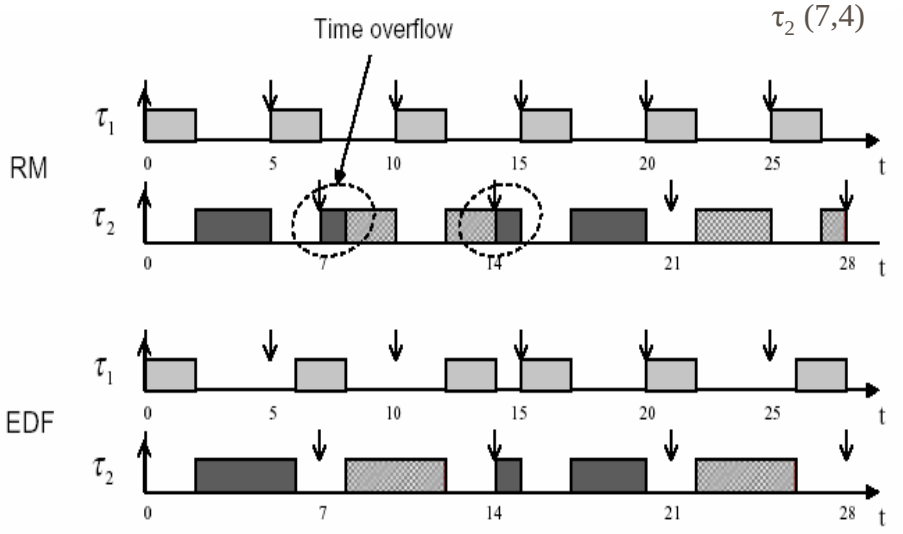
- Algoritmi kuten rate monotonic, mutta
- Prioriteetiksi asetetaan työn suhteellinen aikaraja, jolloin se, jonka aikaraja on pienin pääsee ensin suoritukseen.
- RM ja DM ovat identtiset, jos kaikkien tapahtumien aikarajat ovat niiden jaksojen mittaiset.
- Esim tapahtumajoukko T_1 (50,50,25,100), T_2 (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 ajoitettavuustesti voidaan suorittaa lineaarisessa ajassa (tietyin ehdoin).
- EDF on optimaalinen verrattuna kaikkiin dynaamista prioriteettia käyttäviin menetelmiin, kun
  - yksi prosessori,
  - keskeyttävä suoritus sallitaan ja
  - ei kilpailua resursseista

Esim: RM vs EDF

τ_1 (5,2)
τ_2 (7,4)



LST – Least Slack Time

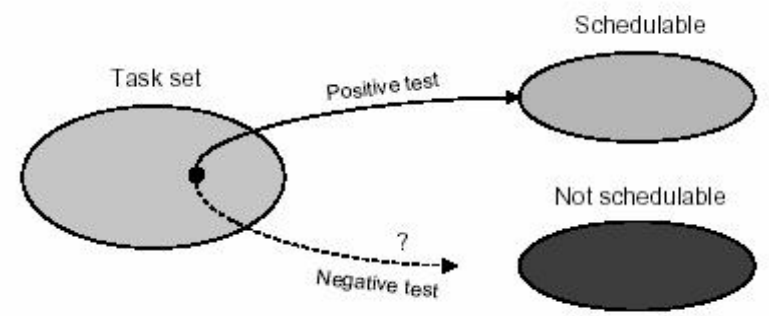
- LST on EDF:lle käänteinen; optimaalinen samoin rajoituksin
- Ajoitetaan vuoroon se, joka voi odottaa huonoimmin
- Vaatii
  - tarkan tiedon laskennan kestosta
  - joka hetki laskemaan kulloisenkin viivyttelyajan aina kaikille töille

Ajoitettavuusanalyysi

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

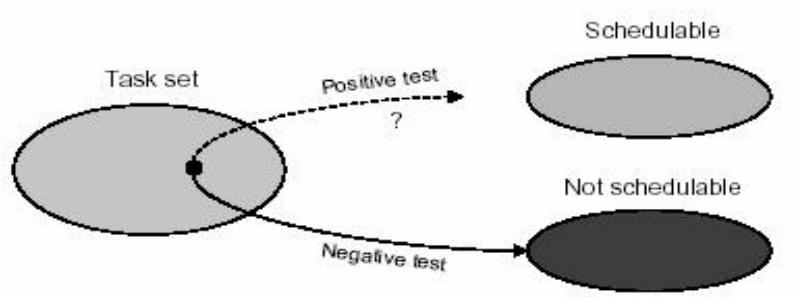
Ajoitettavuustesti

- Ajoitettavuustesti on *riittävä*, jos se positiivisella vastauksella osoittaa, että joukko tapahtumia on varmasti ajoitettava.



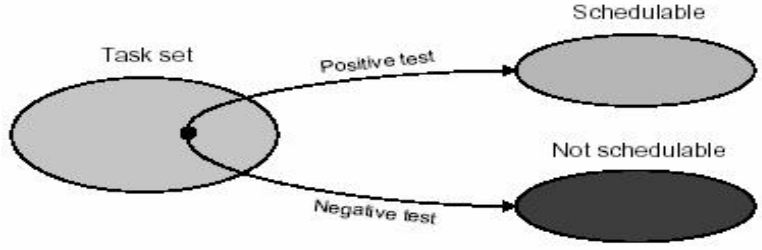
Ajoitettavuustesti

- Ajoitettavuustesti on *välttämätön*, jos se negatiivisella vastauksella osoittaa, että joukko tapahtumia ei varmasti ole ajoitettava.



Ajoitettavuustesti

- Täsmällinen ajoitettavuustesti on sekä riittävä että välttämätön. Jos vastaus on positiivinen, tapahtumajoukko on varmasti ajoitettava ja jos vastaus on negatiivinen tapahtumajoukko ei varmasti ole ajoitettava.



Ajoitettavuustestin menetelmiä

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

Prosessorin käyttöaste

- Käyttöaste U joukolle periodisia tapahtumia on se murto-osa prosessorikapasiteetistä, 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)

- Ajoitusmenetelmä pystyy ajoittamaan, minkä tahansa jaksollisen työjoukon, kunhan niiden yhteenlaskettu käyttöaste on menetelmän ajoitettavaa käyttöastetta pienempi (tai yhtä suuri).
- 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:

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

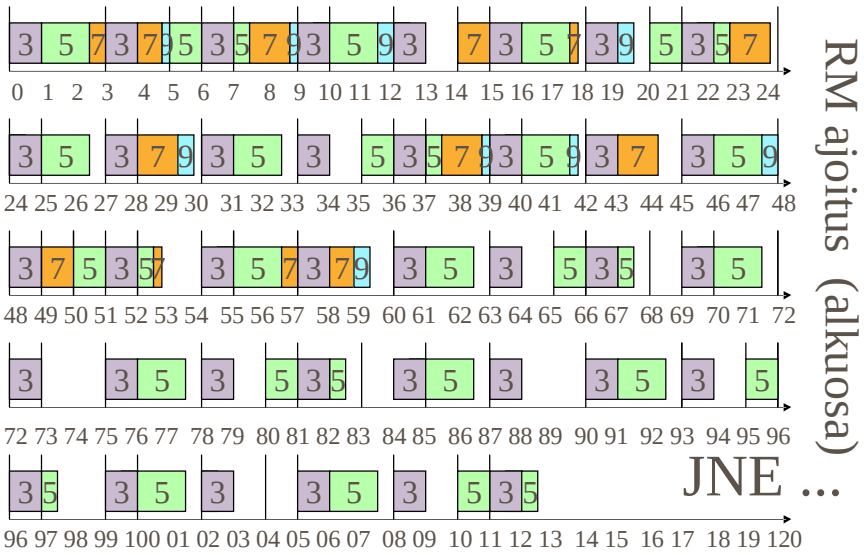
Esimerkki: ajoitettavuustesti

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

- 0.85 <1 => On ajoitettavissa EDF:llä
- 0.85 >4($2^{1/4}$-1)= 0.757 => 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

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

- Työt voidaan ajoittaa, kun $\forall i$

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

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

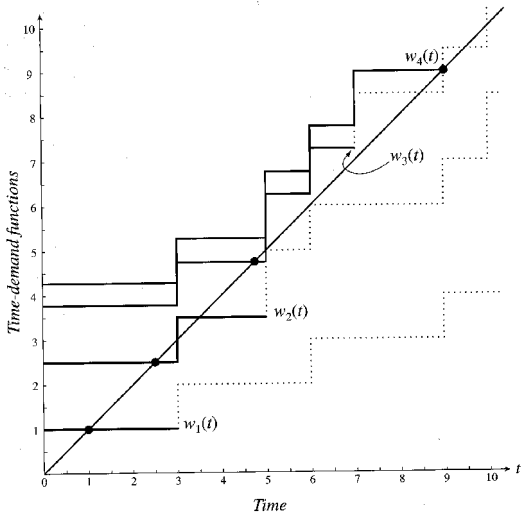


FIGURE 6–9 Time-demand analysis (3,1), (5, 1.5), (7, 1.25), and (9, 0.5).

Analyysin ongelma: Suorituksen estyminen

- Alemman prioriteetin työ voi
  - Estää keskeyttämisensä (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 ajoitettavuusanalyyseissä 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 arviossa korkeamman prioriteetin työn suoritusaikaan alempien estoaikojen maksimi. Työn pisin estymisaika (blocking time) on tällöin

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

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.

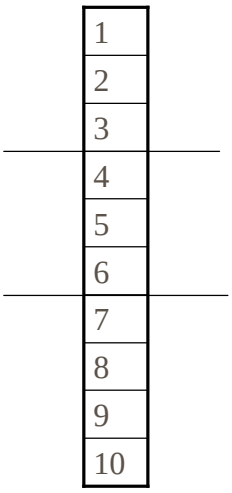
$$U_{EDF} = \sum_{i=1}^{n} \frac{e_i}{\min(p_i, d_i)} + \frac{b_i}{\min(p_i, d_i)} \le 1$$

Prioriteettiluokkien määrä

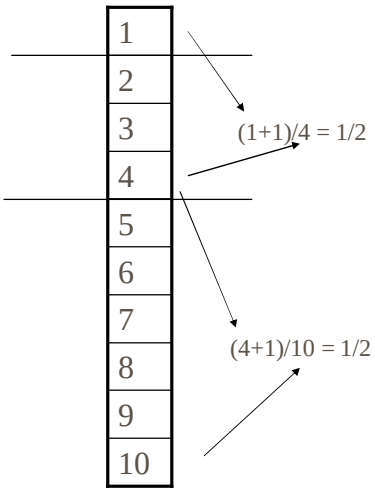
- Analyysit olettavat ääretöntä määrää prioriteetteja
  - Samalla arvolla ei ole ’eriarvoisia’ töitä
- Käytännössä prioriteettiluokkien määrä rajoitettu – joskus 8, usein 32, 127 tai 255.
- Miten työt sijoitellaan rajalliseen prioriteettiavaruuteen?
  - jaetaan tasan
  - säilytetään suhteet

Prioriteettien jako

Tasan



Säilytetään suhteet



Yhteenvetona: Kello-ohjatut staattiset ajoitukset

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

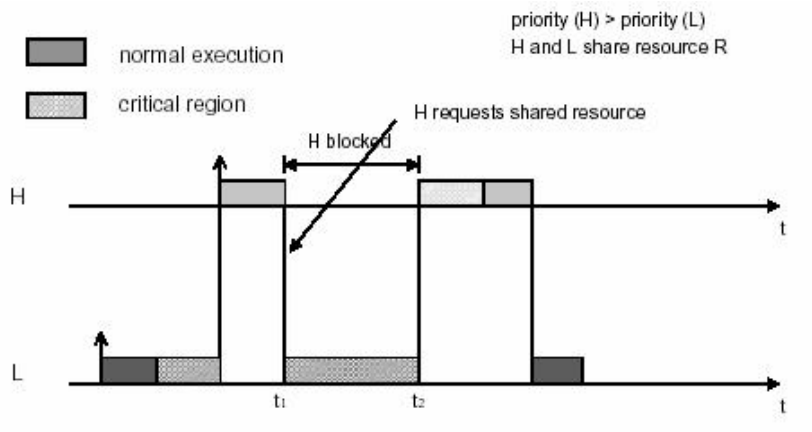
Yhteenvetona: Prioriteettipohjaiset ajoitukset

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

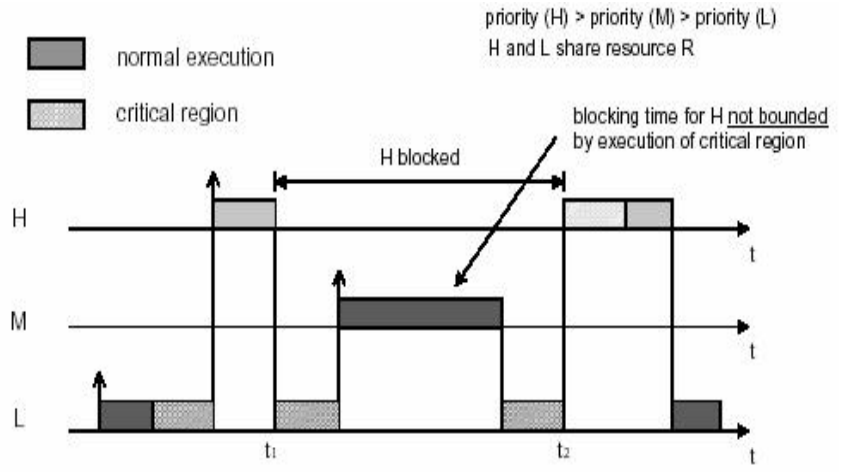
Mitä voi mennä pieleen ?

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

Odotusongelma (Blocking problem)



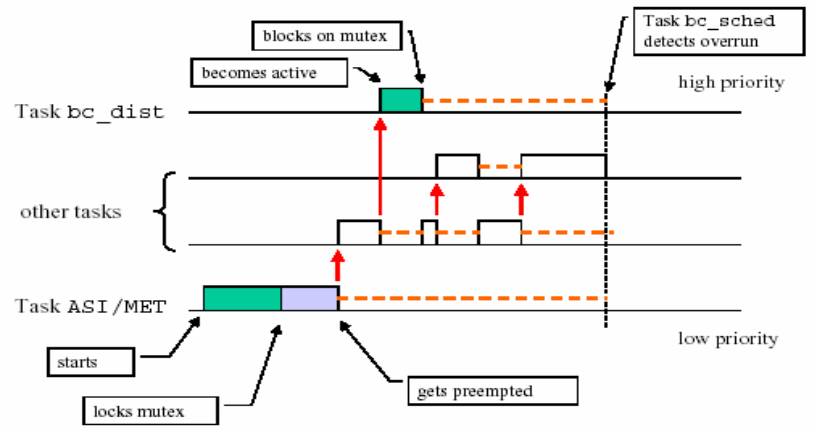
Prioriteetin kääntyminen (Priority inversion)



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 kääntyminen Mars Pathfinder



Prioriteetin kääntymisen välttäminen

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