

1918
TALLINNA TEHNIKAÜLIKOOL
 TALLINN UNIVERSITY OF TECHNOLOGY

Arvutitehnika instituut
 ati.ttu.ee

Sardsüsteemid
 (Embedded Systems)

VII Loeng
 Planeerimine

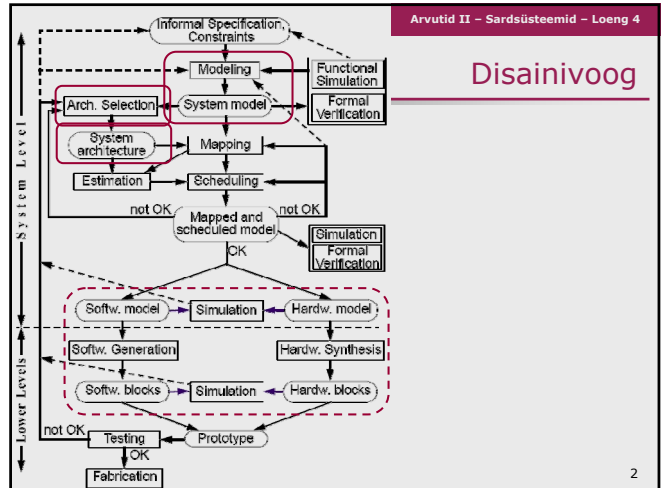
www.pld.ttu.ee/IAF0042

Gert Jervan
 Arvutitehnika instituut

www.pld.ttu.ee/~gerje

Some materials: © Petru Eles

Graphics: © Alexandra Nohle, Gesine Marwede, 2003



© Gert Jervan Arvutid II – Sardüsteemid – Loeng 4

Ülevaade

- ✓ Reaalajasüsteemid
- ✓ Ülesannete planeerimine
- ✓ RTOS

1918
TALLINNA TEHNIKAÜLIKOOL
 TALLINN UNIVERSITY OF TECHNOLOGY

3

© Gert Jervan Arvutid II – Sardüsteemid – Loeng 4

The MARS Pathfinder problem

✓ “But a few days into the mission, not long after Pathfinder started gathering meteorological data, the spacecraft began experiencing total system resets, each resulting in losses of data. The press reported these failures in terms such as “software glitches” and “the computer was trying to do too many things at once.” ...

1918
TALLINNA TEHNIKAÜLIKOOL
 TALLINN UNIVERSITY OF TECHNOLOGY

4

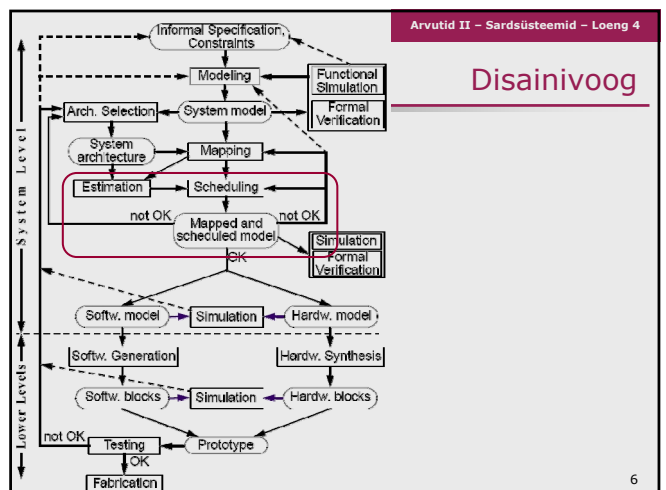
1918
TALLINNA TEHNIKAÜLIKOOL
 TALLINN UNIVERSITY OF TECHNOLOGY

Arvutitehnika instituut
 ati.ttu.ee

Sard-OS, vahevara, planeerimine
 (Embedded OS, middleware, scheduling)

Gert Jervan

Tallinna Tehnikaülikool
 Arvutitehnika instituut



© Gert Jervan Arvutid II – Sardüsteemid – Loeng 4

Reaalaja süsteemid

- ✓ Enamus sardsüsteeme on reaalaja süsteemid
 - Aeg:
 - Süsteemi korrektsus ei sõltu mitte ainult tulemuste loogilisest korrektsusest vaid ka ajast, millal need tulemused on saadud
 - Reaal-:
 - Reaktsioon välistele sündmustele peab toimuma samal ajal sündmusega. Süsteemi aeg peab olema mõõdetav samades ühikutes kui keskkonna aeg
- ✓ Näited:
 - Kontrollisüsteemid, tööstussüsteemid, lennundus, autondus, meditsiin, tuumaenergia, militaar, telekommunikatsioon, multimeedia, ...

1918 TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

7

© Gert Jervan Arvutid II – Sardüsteemid – Loeng 4

Reaalaja süsteemid – tüüpilised omadused

- ✓ Nad on aja-kriitilised
 - Ajaliste piirangute mittejärgimine võib vähendada teenuste kättesaadavust või viia katastroofiliste tagajärgedeni
- ✓ Sisaldavad mitmeid paralleelselt täidetavaid ülesandeid
 - Ülesanded jagavad ühiseid ressursse, nagu näiteks protsessor, kommunikatsioonikanalid. Suhtlevad omavahel. Seetõttu on üheks peamiseks probleemiks ülesannete planeerimine
- ✓ Töökindlus ning veakindlus on esmatähtsad
 - Palju on ohutus-kriitilisi rakendusi

1918 TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

8

© Gert Jervan Arvutid II – Sardüsteemid – Loeng 4

Nõrgad ja ranged reaalaja süsteemid

- ✓ Ajalised piirangud on tüüpiliselt esitatud piir-aegadega (*deadline*), mis määravad ära aja, millal ülesande (*task*) täitmine peab lõppema.
- ✓ Ülesandele seatav piir-aeg võib olla:
 - Range (*hard deadline*): tuleb täielikult ja alati saavutada. Mittesaavutamane võib tuua katastroofilised tagajärjed
 - Garanteerida eelnevalt ja off-line
 - Nõrk (*soft deadline*): ülesanne võib lõppeda peale sellele ette nähtud piir-aega, kuid tulemuse väärtus võib aja jooksul väheneda
 - Kindel (*firm deadline*): sarnane rangele, kuid ei järgne katastroofilisi tagajärgi. Tulemus ei oma peale piir-aega mingit väärtust

1918 TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

9

© Gert Jervan Arvutid II – Sardüsteemid – Loeng 4

Range, kindel, nõrk...

- ✓ Näited rangetest tegevustest
 - Andmete kogumine sensoriga
 - Kriitiliste tingimuste avastamine
 - Aktuaatorite juhtimine
 - Kriitiliste komponentide madala taseme juhtimine
- ✓ Näited nõrkadest tegevustest
 - Kasutajaliidese interpreteerimine
 - Klaviatuuri tulevase informatsiooni töötlemine
 - Ekraanidele edastatav informatsioon
 - Graafilised kasutajaliidesed
 - Aruannete salvestamine

1918 TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

10

© Gert Jervan Arvutid II – Sardüsteemid – Loeng 4

Ennustatavus

- ✓ Ennustatavus (*predictability*) on reaalajasüsteemide üks kõige tähtsamaid omadusi
- ✓ Ennustatavus tähendab, et on võimalik tagada nõuetega paika pandud piir-aegade täitmine:
 - Ranged piir-ajad on alati täidetud
 - Nõrgad piir-ajad on täidetud nii palju, et vajalik teenusekvaliteet (*quality-of-service – QoS*) on tagatud

1918 TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

11

© Gert Jervan Arvutid II – Sardüsteemid – Loeng 4

Täitmise ajad

- ✓ **Def.:** The **worst case execution time** (WCET) is an **upper bound** on the execution times of tasks.
 - The term is not ideal, since a program requiring the WCET for its execution does not have to exist (WCET is a **bound**).
- ✓ **Def.:** The **best case execution time** (BCET) is a lower **bound** on the execution times of tasks.
 - The term is not ideal, since a program running at the BCET for its execution does not have to exist (BCET is a **bound**).

1918 TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

12

Täitmise ajad (2)

- ✓ Keerukus:
 - Tavalisel juhul ei ole võimalik määratleda, kas piirang eksisteerib
 - "Klassikalistel" arhitektuuridel suhteliselt lihtne. Uutel, kus on konveierid, cache'id katkestused, virtuaalmälud, jne. on see märgatavalt keerukam
- ✓ Meetodid
 - Riistvara: vaja teada täpset ajalist käitumist
 - Tarkvara: vajalik vähemalt assembler. Kõrgtasemel sisuliselt võimatu

Keskised täitmise ajad

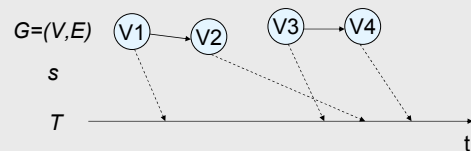
- ✓ Hinnangulised maksumuse ja jõudluse väärtused
 - Väga keeruline saada piisavalt täpseid hinnanguid
 - Täitmise aeg v. täpsus
- ✓ Täpsed maksumuse ja jõudluse väärtused
 - On saadavad tavaliste töövahenditega (näiteks kompilaatorid)
 - On nii täpsed, kui täpsed on sisendandmed

Ennustatavus (2)

- ✓ Ennustatavuse probleemid:
 - Kuidas määratleda iga ülesande WCET?
 - Kuidas määratleda kommunikatsiooni WCET?
 - Kuidas määratleda OSi poolt põhjustatud lisakulud (katkestuste töötlemine, ülesannete haldamine, context switching jne.)?
 - Kui kõik eelnev on vastuse leidnud, siis jääb veel SUUR KÜSIMUS:
Kas kõik ülesanded ja nendega seotud kommunikatsioon on planeeritav olemasoleval arhitektuuril, nõnda et piir-ajad on täidetud?

Reaalaja planeerimine

- ✓ Assume that we are given a task graph $G=(V,E)$.
- ✓ **Def.:** A **schedule** s of G is a mapping $V \rightarrow T$ of a set of tasks V to start times from domain T .



Schedule → programm

Planeerimine (2)

- ✓ Planeerimise probleem:
 - Millised ülesanded ja milline kommunikatsioon tuleb täita millisel ajahetkel antud ressursil, nii et ajalised piirangud on täidetud?
- ✓ Seega:
 - planeerimine peab täitma mitmeid nõudeid: ressursid, sõltuvused, piir-ajad
- ✓ Planeerimist tuleb teostada süsteemi loomise käigus korduvalt

Planeerimine (3)

- ✓ Teatud ülesannete hulk on planeeritav (*schedulable*), kui etteantud planeerimismeetodi puhul kõik piirangud saavad täidetud (mis tähendab, et lahendus planeerimise probleemile on leitav)
- ✓ Vähemalt rangete reaalajasüsteemide puhul tuleb planeerimist teha off-line'is, enne süsteemi tööle rakendamist.

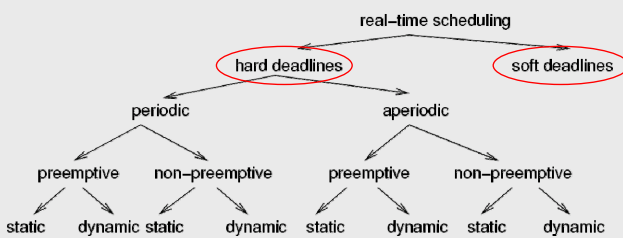
Ülesannete omadused

- ✓ Mida me eeldame, et teame ülesande kohta?
 - Arvutusaaeg (worst case), c .
Kommunikatsiooni puhul me eeldame, et teame kommunikatsiooniks kuluvat aega
 - Ülesande piir-aega d
 - Ülesande saabumise regulaarsust:
 - Perioodilised ülesanded, perioodiga T (identsete ülesannete lõputu jada)
 - Mitteperioodilised ülesanded, ilma fikseeritud saabumise perioodita:
 - Sporaadilised ülesanded: piiratud vähima saabumiste vahelise ajaga: piir-aegu saab garanteerida off-line'is
 - Kui neid piiranguid ei ole teada siis süsteem ei ole planeeritav

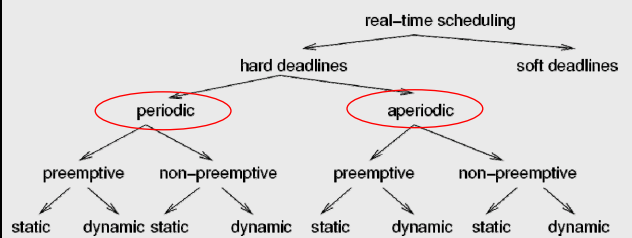
Ülesannete omadused (2)

- ✓ Ülesannete vahelised seosed
 - Järgnevused
 - Rakendusspetsiifilised järgnevused (ülesanne T2 tuleb alati täita peale ülesannet T1, sõltumata sellest, et T2 ei saa T1-lt mingeid andmeid)
 - Andmesõltuvused (meenutage andmevoos mudeleid)
 - Ressursside sõltuvused
 - Jagatud ressursid: protsessorid, siinid, perifeeriaesadmed, puhvid jne.

Planeerimisalgoritmide klassifikatsioon

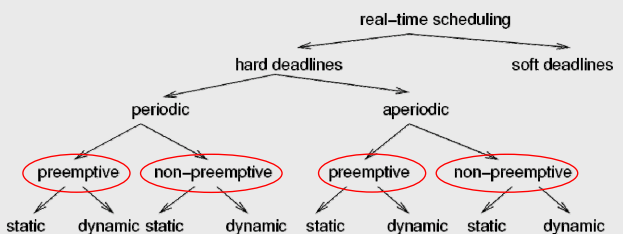


Planeerimisalgoritmide klassifikatsioon



- ✓ Perioodilisteks nimetatakse ülesandeid, mida täidetakse iga T ajaühiku tagant. Iga perioodilise ülesande järjekordset täitmist nimetatakse tööks (*job*)

Planeerimisalgoritmide klassifikatsioon

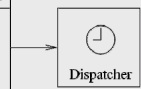


- ✓ Katkestavad (*preemptive*) planeerijad: kasutatakse, kui
 - Mõned ülesanded on pikkade täitmisaegadega, või
 - Reageerimine välissündmustele peab olema lühike
- ✓ Mitte-katkestavad (*non-preemptive*) planeerijad:
 - Kõik ülesanded töötavad, kuni on lõpetanud. Reageerimine väliste sündmustele võib võtta kaua aega

Dünaamiline/staatiline planeerimine

- ✓ Dünaamiline/online planeerimine:
 - Protsessori aja eraldamine toimub jooksvalt, põhinedes seni saabunud informatsioonil
- ✓ Staatiline/offline planeerimine:
 - Planeerimine, kus kasutatakse *a priori* teadmisi ülesannete saabumisest, täitmisaegadest, ja piir-aegadest. Eraldi dispetšer protsessori aja jagamiseks. Timer kasutab disaini loomise käigus genereeritud tabelit.

Time	Action	WCET
10	start T1	12
17	send M5	
22	stop T1	
38	start T2	20
47	send M3	



Planeerimisstrateegiad

- ✓ Staatiline tsükliline planeerimine (*static cyclic scheduling*)
 - Off-line'is genereeritakse tabel, mis sisaldab iga ülesande (kommunikatsiooni) aktiveerimise aegu. Tabelis olevat aktiveerimiste järgnevust korratakse tsükliliselt
- ✓ Prioriteetidepõhine planeerimine (*priority based scheduling*)
 - Ülesanded aktiveeritakse mingi sündmuse mõjul. Kui tekib konflikt, siis arvestatakse ülesannete prioriteetidega
 - Prioriteete võib määrata:
 - Staatiliselt (fikseeritakse off-line ja jäävad muutumatuks)
 - Dünaamiliselt (muutuvad täitmise käigus)

Time-triggered systems (1)

- ✓ In an entirely time-triggered system, the temporal control structure of all tasks is established **a priori** by off-line support-tools. This temporal control structure is encoded in a **Task-Descriptor List (TDL)** that contains the cyclic schedule for all activities of the node. This schedule considers the required precedence and mutual exclusion relationships among the tasks such that an explicit coordination of the tasks by the operating system at run time is not necessary. ..
- ✓ The dispatcher is activated by the synchronized clock tick. It looks at the TDL, and then performs the action that has been planned for this instant [Kopetz].

Time	Action	WCET
10	start T1	12
17	start M5	
22	stop T1	
36	start T2	20
47	start M5	



Time-triggered systems (2)

- ✓ ... **pre-run-time scheduling is often the only practical means of providing predictability in a complex system.** [Xu, Parnas].
- ✓ It can be easily checked if timing constraints are met. The disadvantage is that the response to sporadic events may be poor.

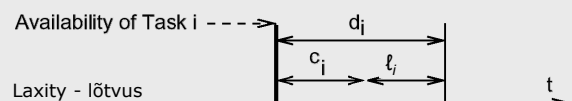
Tsentraalne ja hajutatud planeerimine

- ✓ Tsentraalne ja hajutatud planeerimine
 - Multiprotsessor planeerimine kas lokaalselt ühel või mitmel protsessoril
- ✓ Mono- ja multiprotsessor planeerimine
 - Lihtsamad planeerimisalgoritmid ühe protsessori jaoks
 - Keerukamad algoritmid mitme protsessori jaoks
 - Algoritmid homogeensete multiprotsessorsüsteemide jaoks
 - Algoritmid heterogeensete multiprotsessorsüsteemide jaoks

Planeerimine ilma järgnevuste arvestamiseta

Aperioodiline planeerimine

- ✓ Let $\{T_i\}$ be a set of tasks. Let:
 - c_i be the execution time of T_i ,
 - d_i be the **deadline interval**, that is, the time between T_i becoming available and the time until which T_i has to finish execution.
 - ℓ_i be the **laxity** or **slack**, defined as $\ell_i = d_i - c_i$



© Gert Jervan Arvutid II – Sardsüsteemid – Loeng 4

Üheprotsessoriline süsteem

- ✓ Võrdsed saabumisajad
 - Katkestamine on mõtetu
- ✓ **Earliest Due Date (EDD)**: Execute task with earliest due date (deadline) first.

EDD requires all tasks to be sorted by their (absolute) deadlines. Hence, its complexity is $O(n \log(n))$.

TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

31

© Gert Jervan Arvutid II – Sardsüsteemid – Loeng 4

Earliest Deadline First (EDF)

- ✓ Erinevad saabumisajad. Katkestamine võib vähendada hilinemist
- ✓ Horni teoreem:
 - [Horn74]: Given a set of n independent tasks with arbitrary arrival times, any algorithm that at any instant executes the task with the earliest absolute deadline among all the ready tasks is optimal with respect to minimizing the maximum lateness.

TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

32

© Gert Jervan Arvutid II – Sardsüsteemid – Loeng 4

EDF algoritmi

- ✓ EDF algoritmi:
 - Iga kord kui uus ülesanne saabub:
 - Lisatakse see valmis ülesannete järjekorda, sorteerituna nende absoluutsete piir-aeegade põhjal. Täidetakse ülesannet, mis on järjekorras esimene.
 - Kui saabunud ülesanne pannakse järjekorras esimeseks, siis hetkel täidetakse ülesanne katkestatakse.
- ✓ Lihtne lähenemine sorteeritud järjekordadega (iga saabuvat ülesannet võrreldakse kõigi ülesannetega) nõuab tööaega $O(n^2)$; (väiksem kahendotsingu puhul).

Sorteeritud järjekord

Ülesanne, mida täidetakse

TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

33

© Gert Jervan Arvutid II – Sardsüsteemid – Loeng 4

EDF - näide

	arrival	duration	deadline
T1	0	10	33
T2	4	3	28
T3	5	10	29

Task arrivals

Varasem piir-aeg ei katkesta

Hilisem piir-aeg ei katkesta

TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

34

© Gert Jervan Arvutid II – Sardsüsteemid – Loeng 4

Least laxity (LL), Least Slack Time First (LST)

- ✓ Prioriteedid pöördvõrdeline lõtvusega (mida vähem lõtv, seda kõrgem prioriteet), dünaamilised prioriteedid, katkestav

	arrival	duration	deadline
T1	0	10	33
T2	4	3	28
T3	5	10	29

$l(T1)=33-4-6=23$ $l(T1)=33-5-6=22$ $l(T1)=33-13-6=14$ $l(T1)=33-16-6=11$
 $l(T2)=28-4-3=21$ $l(T2)=28-5-2=21$ $l(T2)=28-13-2=13$ $l(T3)=29-16-1=12$
 $l(T3)=29-5-10=14$ $l(T3)=29-13-2=14$

TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

35

© Gert Jervan Arvutid II – Sardsüsteemid – Loeng 4

LL (LST) omadused

- ✓ Ei piisa planeerija väljakutsumisest (ja lõtvuse arvutamisest) ainult ülesannete saabumise hetkel
- ✓ Planeerija väljakutsumise lisakulu
- ✓ Palju context switch'e
- ✓ Avastab üle minevad piir-ajad varakult
- ✓ Optimaalne monoprotsessor süsteemidele
- ✓ Dünaamilised prioriteedid → ei saa kasutada fikseeritud prioriteetidega OSides
- ✓ Vajab informatsiooni täitmisaegade kohta

TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

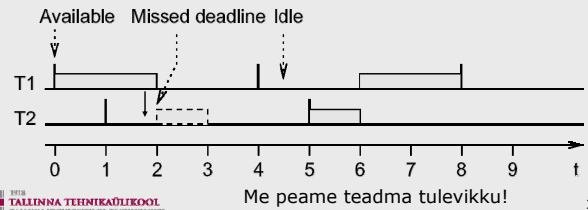
36

Mittekatkestav planeerimine

- ✓ Kui ülesannete katkestamine ei ole lubatud, siis optimaalsed programmid võivad jätta protsessori ootele, et lõpetada ülesanded, millel on lühikesed piir-ajad, kuid mis saavad hilja

Mittekatkestav planeerimine (2)

- ✓ T1: perioodiline, $c_1 = 2$, $p_1 = 4$, $d_1 = 4$
- ✓ T2: vahel saadaval ajahetkel $4*n+1$, $c_2 = 1$, $d_2 = 1$
- ✓ T1 peab alustama $t=0$
- ✓ Piiraeg ületatud, kuid programm oleks võimalik (kui alustada T2 esimesena) → planeerimine ei ole optimaalne



Perioodiline planeerimine

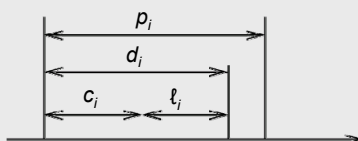
Tallinna Tehnikaülikool
Arvutitehnika instituut

Staatiline tsükline planeerimine

- ✓ Off-line'is genereerida aktiveerimisajad kõikidele ülesannetele
 - Need aktiveerimisajad määravad ära süsteemi käitumise üle (hüper)perioodi T^h
 - Seda jada korratakse tsükliliselt
- ✓ Kui kõikidel ülesannetel on sama periood $T \rightarrow T^h = T$
- ✓ Kui ülesannetel on erinevad perioodid T_1, T_2, \dots, T_n siis T^h on T_1, T_2, \dots, T_n vähim ühiskordne

Perioodiline planeerimine

- ✓ Olgu:
 - p_i ülesande T_i periood,
 - c_i ülesande T_i täitmisaeg
 - d_i ülesande piir-aja **intervall**, see on ajavahemik, millal ülesanne T_i on valmis täitmiseks ning seesama ülesanne T_i peab olema täidetud.
 - ℓ_i on lõtvus (**laxity** or **slack**), defineeritud kui $\ell_i = d_i - c_i$



Keskmine koormatus

Keskmine koormatus:

$$\mu = \sum_{i=1}^n \frac{c_i}{p_i}$$

Vajalik tingimus süsteemi planeeritavuse tagamiseks (m =protsessorite arv):

$$\mu \leq m$$

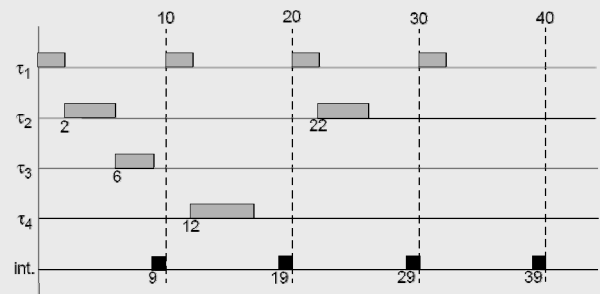
Staatiline tsükliline planeerimine (2)

- ✓ Näide: 4 sõltumatut ülesannet ühel protsessoril

	Period=deadline	Worst case comp. time
τ_1	10	2
τ_2	20	4
τ_3	40	3
τ_4	40	5
System management	10	1

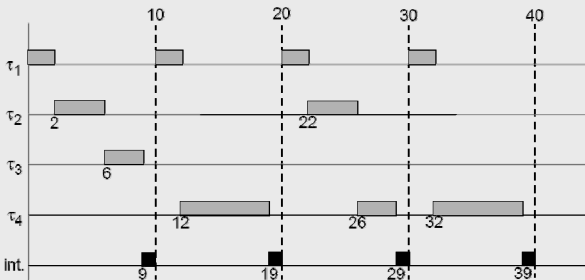
$$T^h = \text{LCM}(10, 20, 40) = 40$$

Staatiline tsükliline planeerimine (3)

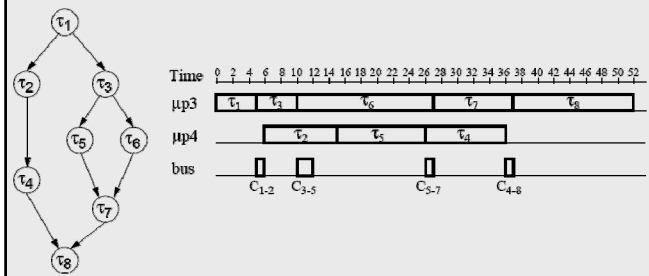


Staatiline tsükliline planeerimine (4)

- ✓ Sama näide, kuid τ_4 täitmisaeg on 17. Ilma katkestamiseta ei saa me ehitada programmi!



Staatiline tsükliline planeerimine (5)



List Scheduling

- ✓ Staatilise tsüklilise planeerimise jaoks sobib *list scheduling*:
 - Iga ressursi jaoks on olemas nimekiri valmis ülesanneteks. See sisaldab neid ülesandeid, mis on mõeldud täitmiseks sellel ressursil, kuid mis ei ole veel planeeritud. Samas, kõik nende eellased on planeeritud ja lõpetanud oma töö.
 - Ülesandeid võetakse nimekirjadest ning planeeritakse mingi prioriteedi funktsiooni alusel

Staatiline tsükliline planeerimine

- ✓ Mis on head küljed:
 - Väga hästi ennustatav
 - Kerge siluda
 - Väike lisakulu
- ✓ Mis on halvad küljed:
 - Ei ole painduv:
 - Kvaliteet väheneb märgatavalt kui täitmisajad erinevad eeldatutest
 - Kui lisatakse uusi ülesandeid tuleb kogu programm ümber arvutada
 - Katkestuste käsitlemine:
 - Staatiliselt eraldatud ajahetked
 - Tuleb vältida väga pikki hüperperioode
 - Üksikute ülesannete perioode peab ühtlustama → kunstlikult vähendatud perioodid → suurenenud koormus → protsessori aja raiskamine
 - Manuaalne tükeldamine, et mahuks perioodidesse

Prioriteetidel põhinev katkestav planeerimine

- ✓ Priority Based Preemptive Scheduling
 - Programmi ei genereerita ette valmis. Ülesanded käivituvad väliste sündmuste (näiteks signaalide, sõnumite) mõjul
 - Igal ajahetkel jookseb kõrgeima prioriteediga ülesanne. Kui korraga on valmis mitu ülesannet, siis valitakse kõrgeima prioriteediga ülesanne
 - Ülesandeid võib katkestada igal ajahetkel. Kui ülesanne on valmis täitmiseks ning tema prioriteet on kõrgem, kui hetkel täidetaval ülesandel, siis ülesande täitmine katkestatakse

Prioriteetidel põhinev katkestav planeerimine (2)

- ✓ Prioriteete võib omistada nii staatiliselt kui ka dünaamiliselt
- ✓ Kas ülesanne saab valmis enne oma piir-
aega? Sellele leiab vastuse planeeritavuse
analüüsiga (*schedulability analysis*)

Planeeritavuse analüüs

- ✓ Planeeritavust on võimalik analüüsida teatud juhtudel:
 - Ülesannete perioodid ja täitmisajad on teada
 - Ülesannete perioodid on staatilised või Kasutatakse teatud piiratud dünaamilist prioriteetide põhimõtet, nagu näiteks EDF
 - Ülesandeid täidetakse ühel protsessoril või Multiprotsessorsüsteemid, kus kommunikatsiooninfrastruktuur on ennustatava viitega (nagu näieks CAN, TDMA protokollid jne.)

The MARS Pathfinder problem (2)

- ✓ "VxWorks provides preemptive priority scheduling of threads. Tasks on the Pathfinder spacecraft were executed as threads with priorities that were assigned in the usual manner reflecting the relative urgency of these tasks."
- ✓ "Pathfinder contained an "information bus", which you can think of as a shared memory area used for passing information between different components of the spacecraft."
 - A bus management task ran frequently with high priority to move certain kinds of data in and out of the information bus. Access to the bus was synchronized with mutual exclusion locks (mutexes)."

The MARS Pathfinder problem (3)

- ✓ The meteorological data gathering task ran as an infrequent, low priority thread, ... When publishing its data, it would acquire a mutex, do writes to the bus, and release the mutex. ..
- ✓ The spacecraft also contained a communications task that ran with medium priority."

☞

High priority: retrieval of data from shared memory
 Medium priority: communications task
 Low priority: thread collecting meteorological data

The MARS Pathfinder problem (4)

- ✓ "Most of the time this combination worked fine. However, very infrequently it was possible for an interrupt to occur that caused the (medium priority) communications task to be scheduled during the short interval while the (high priority) information bus thread was blocked waiting for the (low priority) meteorological data thread. In this case, the long-running communications task, having higher priority than the meteorological task, would prevent it from running, consequently preventing the blocked information bus task from running. After some time had passed, a watchdog timer would go off, notice that the data bus task had not been executed for some time, conclude that something had gone drastically wrong, and initiate a total system reset. This scenario is a classic case of priority inversion."

Priority inversion on Mars

- ✓ Priority inheritance also solved the Mars Pathfinder problem: the VxWorks operating system used in the pathfinder implements a flag for the calls to mutex primitives. This flag allows priority inheritance to be set to “on”. When the software was shipped, it was set to “off”.

The problem on Mars was corrected by using the debugging facilities of VxWorks to change the flag to “on”, while the Pathfinder was already on the Mars [Jones, 1997].

VxWorks – WindRiver RTOS

