TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Arvutitehnika instituut
ati.ttu.ee

Sardsüsteemid
(Embedded Systems)

VII Loeng
Planeerimine

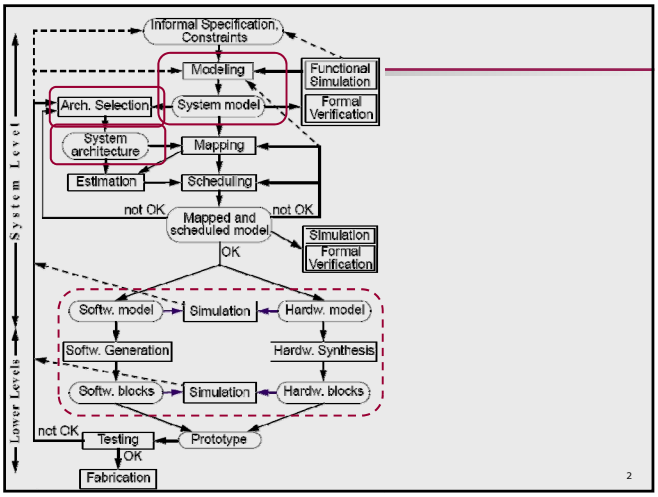
www.pld.ttu.ee/IAF0042

Gert Jervan
Arvutitehnika instituut

ati.ttu.ee/~gerje

Graphics: © Alexandra Nolte, Gesine Marwedel, 2003

Some materials: © Petru Eles



2

Ülevaade

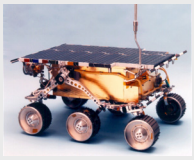
- ✓ Reaalajasüsteemid
- ✓ Ülesannete planeerimine
- ✓ RTOS

TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

3

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"." …



TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

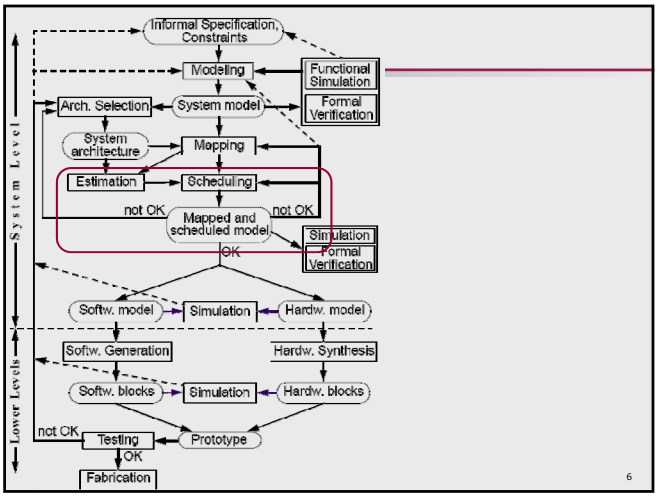
4

TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Arvutitehnika instituut
ati.ttu.ee

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

Gert Jervan



6

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:
  - Kontrollsüsteemid, tööstussüsteemid, lennundus, autondus, meditsiin, tuumaenergia, militaar, telekommunikatsioon, multimeedia, ...

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

Nõrgad ja ranged reaalaja süsteemid

- Ajalised piirangud on tüüpiliselt esitatud piir-aegadega (*dedline*), mis määravad ära aja, millal ülesande (*task*) täitmine peab lõppema.
- Ülesandele seatav piir-aeg võib olla:
  - Range (*hard dealine*): tuleb täielikult ja alati saavutada. Mittesaavutamine 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

Range, kindel, nõrk...

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

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

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**).

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

Keskmised 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:

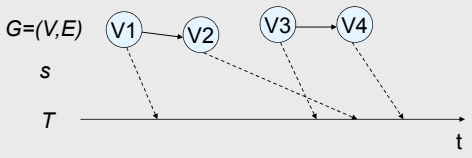
    Kas kõik ülesanded ja nendega seotud kommunikatsioon on planeeritav olemasoleval arhitektuuril, nõnda et piir-ajad on täidetud?

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.

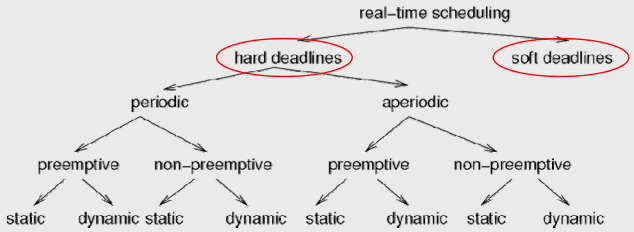
Ülesannete omadused

- Mida me eeldame, et teame ülesande kohta?
  - Arvutusaeg (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

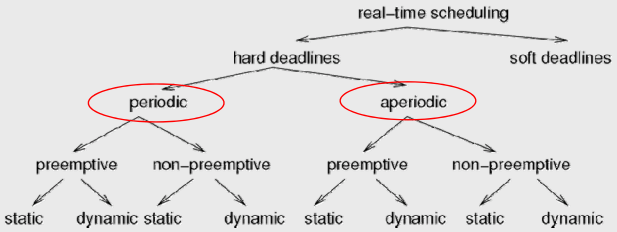
Ülesannete omadused (2)

- 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 andmevoo mudeleid)
  - Ressursside sõltuvused
    - Jagatud ressursid: protsessorid, siinid, perifeeriaseadmed, puhvrid jne.

Planeerimisalgoritmide klassifikatsioon

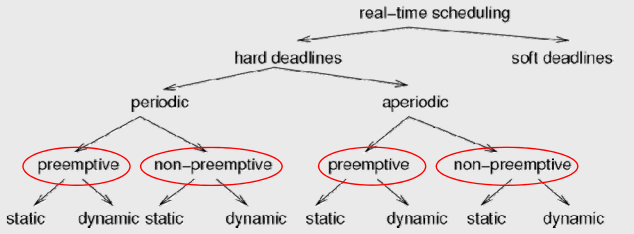


Planeerimisalgoritmide klassifikatsioon



- 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 välistele 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 dispetžer protsessori aja jagamiseks. Timer kasutab disaini loomise käigus genereeritud tabelit.

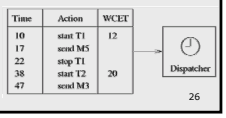
Time	Action	WCET
10	start T1	12
17	send M5	
22	stop T1	
38	start T2	20
47	send M3	

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-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 homogeensedte multiprotsessorsüsteemide jaoks
    - Algoritmid heterogeensete multiprotsessorsüsteemide jaoks

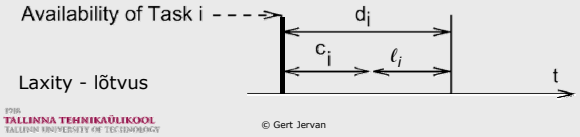
1918 TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY

Arvutitehnika instituut
ati.ttu.ee

Planeerimine ilma järgnevuste arvestamiseta

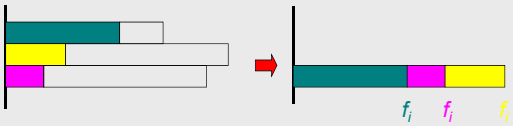
Aperioodiline planeerimine

- Let $\{T_i\}$ be a set of tasks. Let:
  - c_i be the execution time of T_i,
  - d_i be the **deadline interva**l, 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 $\ell_i = d_i - c_i$



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))*.

Earliest Deadline First (EDF)

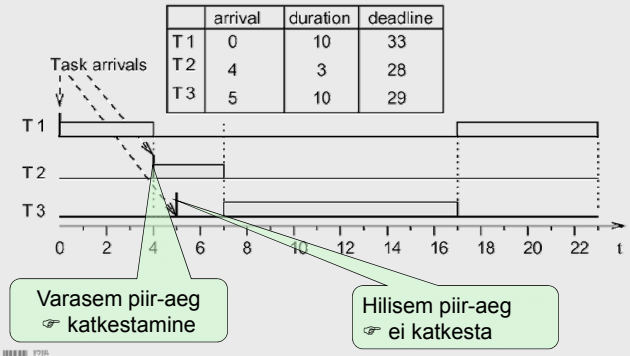
- Erinevad saabumisajad. Katkestamine võib vähendada hilinemist
- Horni teoreem:
  - [Horn74]: Given a set of n independent tasks with arbitrary arrival times, any algorithm that at any instant executes the task with the earliest absolute deadline among all the ready tasks is optimal with respect to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

EDF algoritm

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

EDF - näide

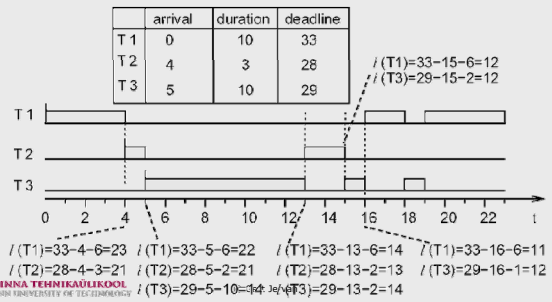
	arrival	duration	deadline
T1	0	10	33
T2	4	3	28
T3	5	10	29



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



LL (LST) omadused

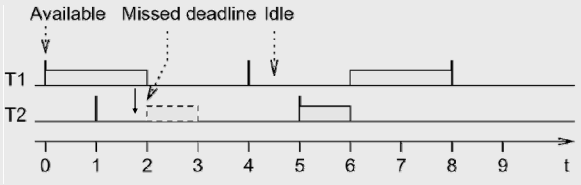
- Ei piisa planeerija väljakutsumisest (ja lõtvuse arvutamisest) ainult ülesannete saabumise hetkel
- Planeerija väljakutsumise lisakulu
- Palju context switch'e
- Avastab üle minevad piir-ajad varakult
- Optimaalne monoprotsessor süsteemidele
- Dünaamilised prioriteedid → ei saa kasutada fikseeritud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

Mittekatkestav planeerimine

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



Me peame teadma tulevikku!

1918 TALLINNA TEHNIKAÜLIKOOL TALLINN UNIVERSITY OF TECHNOLOGY

Arvutitehnika instituut
ati.ttu.ee

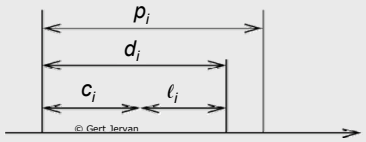
Perioodiline planeerimine

Staatiline tsükliline 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 → $T^h = T$
- Kui ülesannetel on erinevad perioodid $T_1, T_2, \ldots, T_n$, siis T^h on $T_1, T_2, \ldots, T_n$ 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):

$$\mu \leq m$$

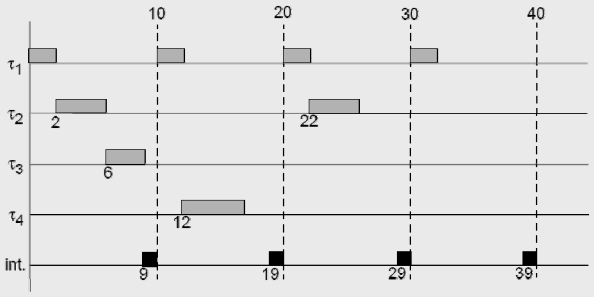
Staatiline tsükliline planeerimine (2)

- 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

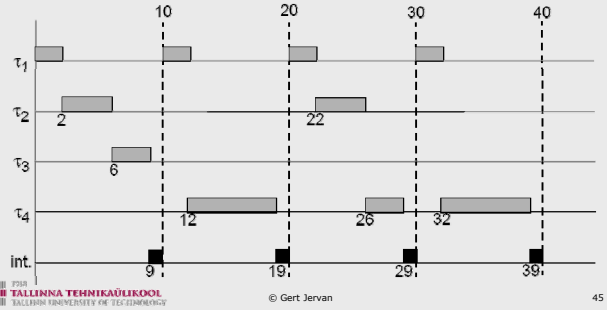
$T^h = \text{LCM}(10, 20, 40) = 40$

Staatiline tsükliline planeerimine (3)

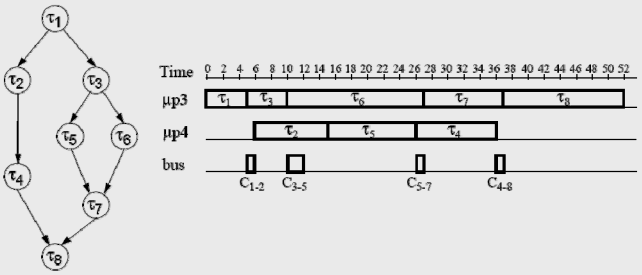


Staatiline tsükliline planeerimine (4)

- Sama näide, kuid $\tau 4$ 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

- 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 kommunikatsiooniinfrastruktuur 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

Sard- ja reaalaja OSid

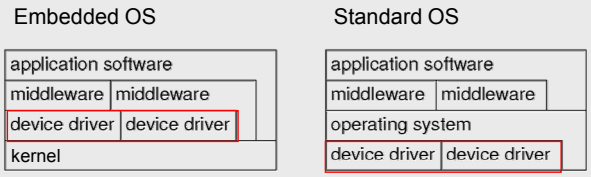
Gert Jervan

Sard OS - nõudmised

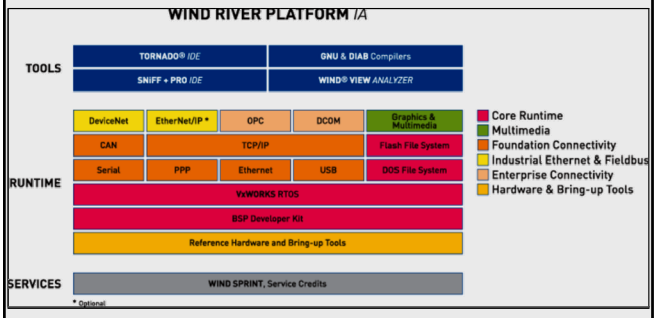
- Konfigureeritavus
  Mitte ükski reaalaja OS (RTOS) ei sobi kõikide süsteemide jaoks, meil ei ole võimalust hoida süsteemis kasutamata funktsionaalsust ☞ vajadus kofigureeritavuse järgi.
  - Lihtsam moodus: eemalda mittevajalik funktsionaalsus (linker?).
  - Tingimuslik kompileerimine (kasutade #if ja #ifdef käske).
  - Dünaamilised andmed võidakse asendada staatiliste andmetega.
  - Kompileerimisaegne hindamine.
  - Objektorienteeritus.
- Suurte süsteemide, kus mitmeid erinevaid OSe verifitseerimine võib olla keeruline:
  - Iga tuletatud OS tuleb põhjalikult testida;
  - Näiteks probleem eCos'iga: (open source RTOS from Red Hat) 100 kuni 200 konfigureerimise punkti[Takada, 01].

Sard OS – nõudmised (2)

- Ketast ja võrguühendust hallatakse läbi ülesannete, mitte draiverite kaudu.
- Paljud sardsüsteemid on ilma ekraani, klaviatuuri, hiireta.
- Põhimõtteliselt ei ole olemas ühtegi seadet (välja arvatud süsteemi timer), mida peaksid toetama kõiks OSi versioonid.



Näide: WindRiveri platvorm autotööstusele



© Windriver

Sard OS – nõudmised (3)

- Kaitsemehhanismid ei ole alati vajalikud:
  Sardsüsteem on mingi kindla ülesande jaoks, mittetestitud tarkvara ei laadita peaaegu kunagi, tarkvara loetakse usaldusväärseks.
  (NB! Kaitsemehhanisme võib vaja minna ohutuse ja turvalisuse tagamiseks).

Eraldi I/O operatsioone ei ole vaja ning ülesannetel võib olla oma I/O:

Example: Let `switch` be the address of some switch
Simply use

```
load register,switch
```

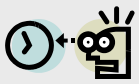
instead of OS call.

Sard OS – nõudmised (4)

- Katkestusi võivad tekitada kõik protsessid Standard OSi puhul oleks see suur risk töökindlusle.
  - sardtarkvara kohta võib eeldada, et see on testitud,
  - kuna kaitsmine ei ole oluline ja
  - kuna hea kontroll erinevate seadmete üle on vajalik,
  - on võimalik lasta katkestustel alustada ja peatada ülesandeid (hoides ülesannete algusaadresse katkestuste tabelis).
  - On märgatavalt effektiivsem, kui seda teha läbi OSi.

Sard OS – nõudmised (5)

- Reaalaeg:
  - Paljud sardsüsteemid on reaalajasüsteemid, seetõttu tuleb nendes süsteemides kasutada reaalaja OSe (RTOS).



RTOS

- **Def.:** (A) real-time operating system is an operating system that supports the construction of real-time systems
- The following are the three key requirements:
  - **The timing behavior of the OS must be predictable.**
    ∀ services of the OS: Upper bound on the execution time!
    RTOSs must be deterministic:
    - unlike standard Java,
    - short times during which interrupts are disabled,
    - contiguous files to avoid unpredictable head movements.

    [Takada, 2001]

RTOS (2)

- **OS must manage the timing and scheduling**
  - OS possibly has to be aware of task deadlines; (unless scheduling is done off-line).
  - OS must provide precise time services with high resolution.

  [Takada, 2001]

Aeg

- Aeg on reaalaja süsteemides kesksel kohal
- Tegelik aeg on reaalsetes numbrites
- Kaks standardit, mida kasutatakse:
  - **International atomic time TAI**
    (french: temps atomic internationale)
    Free of any artificial artifacts.
  - **Universal Time Coordinated (UTC)**
    UTC is defined by astronomical standards
- UTC ja TAI on identsed alates 01.01.1958.
- Sellest ajast alates on lisatud 30 sekundit.

Sisemine sünkroniseerimine

- Sünkroniseeritakse ühe “master clock-iga”
  - Tüüpiliselt algkäivitusel
- Hajutatud sünkroniseerimine:
  - Kogutakse informatsiooni naabritelt
  - Arvutatakse parandusväärtus
  - Muudetakse aega

- Esimese sammu täpsus on sõltuv:
  - Rakenduste tasemel: ~500 µs - 5 ms
  - OSi kernel: 10 µs - 100 µs
  - Kommunikatsiooni riistvara: < 10 µs

Väline sünkroniseerimine

- Väline sünkroniseerimine tagab ühilduvuse globaalse ajaga.
- Viimasel ajal kasutatakse selleks ennekõike GPSe
- GPS pakub TAI ja UTC ajainformatsiooni.
- Täpsus on ca 100 ns.

GPS hiir

© Dell

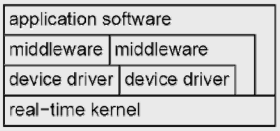
RTOS (3)

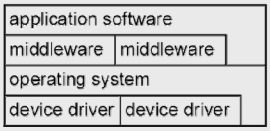
- **OS peab olema kiire!**
  Vajalik praktikas

[Takada, 2001]

RTOSi kernelid

- Eristatakse
  - Reaalaja kerneleid ja standard OSide muudetud kerneleid





- **Eristatakse**
  - Üldised RTOSid and RTOSid spetsiaalse rakendusvaldkonna jaoks,
  - Standardsed APId (e.g. POSIX RT-Extension of Unix, ITRON, OSEK) or spetsiaalsed APId.

RTOS kernelite funktsionaalsus

- Sisaldab
  - Protsessori haldus
  - Mälu haldus
  - Timeri haldus

  } Ressursside haldus
  - Ülesannete haldus (resume, wait etc),
  - Ülesannete vaheline kommunikatsioon ja sünkroniseerimine

Vahevara - middleware

- Reaalaja andmebaasid
- Ligipääs kaugemale olevatele objektidele

Reaalaja andmebaasid

- Eesmärk: hoida ja pakkuda püsivat infot
- Tehing = jada lugemis/kirjutamisoperatsioone
- Muutused ei ole püsivad kuni need ei ole kinnitatud
- Tehingutele esitatavad nõudmised (“ACID”):
  - **A**tomic: state information as if transaction is either completed or had no effect at all.
  - **C**onsistent: Set of values retrieved from several accesses to the data base must be possible in the world modeled.
  - **I**solation: No user should see intermediate states of transactions
  - **D**urability: results of transactions should be persistent.

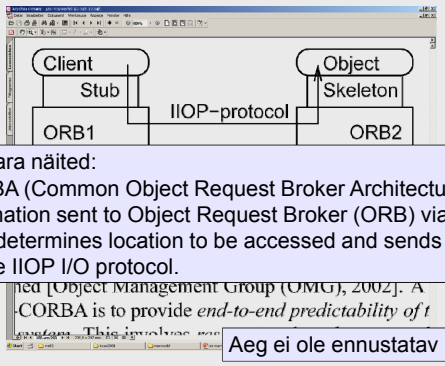
Reaalaja andmebaasid (2)

- ✓ Reaalaja andmebaaside loomisega seotud probleemid:
  - Tehinguid võidakse suvalisel ajahetkel katkestada, ilma et oleks kinnitatud
  - Kõvaketastega töötamine on ääretult ettearvamatu

✓ Võimalikud lahendused

1. Kettavabad andmebaasid
2. Nõrgendatud ACID nõudmised

Kaugemate objektidega töötamine



Tarkvara näited:
CORBA (Common Object Request Broker Architecture).
Information sent to Object Request Broker (ORB) via local stub.
ORB determines location to be accessed and sends information via the IIOP I/O protocol.

Aeg ei ole ennustatav

Reaalaja CORBA

- ✓ Väga oluline, et RT-CORBA pakuks
  - Ennustatavust fikseeritud prioriteetidega süsteemis.
    - See sisaldab kliendi ja serveri vahel lõimede prioriteetide austamist, et lahendada ressurssidele konkureerimist
    - ja operatsioonide latentsuse piiramist.
  - Lõimede prioriteete ei ole vaja jälgida, kui lõimed saavutavad välistava (mutually exclusive) ligipääsu ressurssidele (priority inversion).

Message passing interface (MPI)

- ✓ Message passing interface (MPI): alternative to CORBA
- ✓ MPI/RT: a real-time version of MPI [MPI/RT forum, 2001].
- ✓ MPI-RT does not cover issues such as thread creation and termination.
- ✓ MPI/RT is conceived as a potential layer between the operating system and standard (non real-time) MPI.

1918 TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Arvutitehnika instituut
ati.ttu.ee

Küsimusi?

Gert Jervan
ati.ttu.ee/~gerje