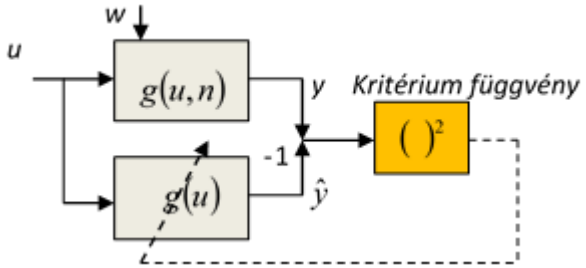
Beágyazott információs rendszerek

Második zárthelyire készülés

2020. november 20.

1. Mutassa be a polinomiális regresszió eljárását köbösnek tűnő függvénykapcsolat (például CMOS processzor villamos-teljesítmény igényének frekvenciafüggése), két ismeretlen paraméter és *N* független mérés feltételezésével (max. 2 pont)! Vezesse le az ismeretlen paraméterek kiszámítására alkalmas összefüggéseket (max. 3 pont)! Indokolja meg, hogy miért előnyös a négyzetes hibakritérium és a paramétereiben lineáris modell együttes alkalmazása (max. 2 pont)!



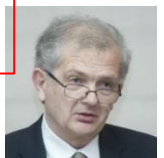
Ilyenkor $y_n = a_0 + a_3 u_n^3 + w_n$, ahol w_n *az additív zaj,* $n = 0,1,\dots,N-1$, $z = Ua + w$.

A legkisebb négyzetes hibájú becslő összefüggéseit használva:

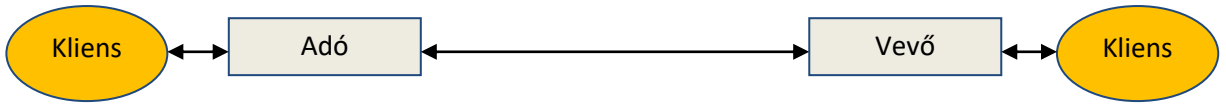
$$\hat{a}_{LS} = [U^T U]^{-1} U^T z$$

$$z = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{bmatrix} = \begin{bmatrix} 1 & u_0^3 \\ 1 & u_1^3 \\ \vdots & \vdots \\ 1 & u_{N-1}^3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_3 \end{bmatrix} + \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{N-1} \end{bmatrix} \quad U^T U = \begin{bmatrix} N & \sum_{n=0}^{N-1} u_n^3 \\ \sum_{n=0}^{N-1} u_n^3 & \sum_{n=0}^{N-1} u_n^6 \end{bmatrix}; \; U^T z = \begin{bmatrix} \sum_{n=0}^{N-1} y_n \\ \sum_{n=0}^{N-1} u_n^3 y_n \end{bmatrix};$$

$$\begin{bmatrix} \hat{a}_0 \\ \hat{a}_3 \end{bmatrix} = \frac{1}{\frac{1}{N}\sum_{n=0}^{N-1} u_n^6 - \left(\frac{1}{N}\sum_{n=0}^{N-1} u_n^3\right)^2} \begin{bmatrix} \frac{1}{N}\sum_{n=0}^{N-1} u_n^6 & -\frac{1}{N}\sum_{n=0}^{N-1} u_n^3 \\ -\frac{1}{N}\sum_{n=0}^{N-1} u_n^3 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{N}\sum_{n=0}^{N-1} y_n \\ \frac{1}{N}\sum_{n=0}^{N-1} u_n^3 y_n \end{bmatrix};$$



2. Mutassa be a PAR protokollok működését az adó- és vevőoldali programlépések felsorolásával (max. 2 pont)! Hogyan határozható meg a hibadetektálás késleltetése (max. 1 pont)?



Adó oldali program:

(1) Az ismételt küldések számlálóját nullázzuk.
(2) Indítjuk a visszaigazoláshoz rendelt time-out számlálót.
(3) Indítjuk az üzenetet.
(4) A time-out-on belül fogadjuk a visszaigazolást.
(5) Értesítjük a klienst a sikeres adattovábbításról.

Ha nincs visszaigazolás a time-out-on belül:

(a) Ellenőrizzük az ismételt küldések számlálóját, hogy elérte-e a maximumát.
(b) Ha igen, akkor megszakít minden tevékenységet, és hibát jelez a kliensnek.
(c) Ha nem, akkor inkrementálja az ismételt küldések számlálóját, és visszatér a fenti (2) ponthoz.

Vevő oldali program:

(1) Üzenet érkezésekor ellenőrzi, hogy ez az üzenet érkezett-e már korábban.
(2) Ha nem, akkor visszaigazol, és értesíti a kliensét.
(3) Ha igen, akkor csak visszaigazol. (Ilyenkor az előző visszaigazolás time-out időn túl érkezhetett az adóhoz, ha egyáltalán megérkezett.

A hibadetektálás késleltetése:

(Az ismétlések száma+1)*timeout

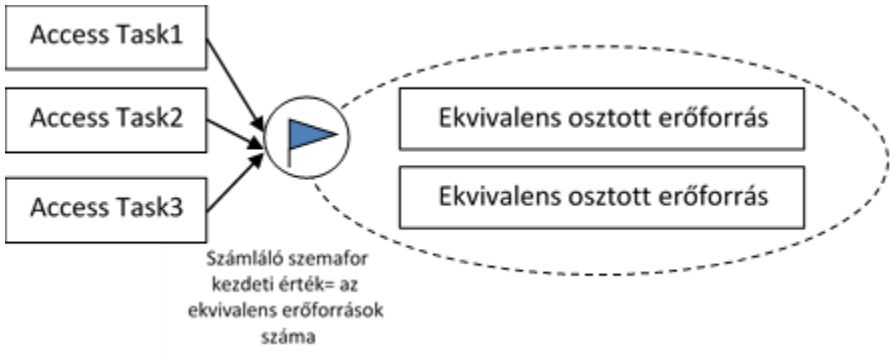
3. Alkalmas programrészlet megadásával mutassa be az IT-vel kiegészített ciklikus programszervezés módszerét (max. 2 pont)! Mekkora worst-case válaszidő érhető el ezzel a módszerrel (max. 1 pont)?

IT-vel kiegészített ciklikus programszervezés: nem lekérdezéssel, hanem megszakítással jelzünk.

```
FLAG A, B;
void interrupt A_Handler() { Handle_HW_A(); A=TRUE; }
void interrupt B_Handler() { Handle_HW_B(); B=TRUE; }
void interrupt C_Handler() { Handle_HW_C(); C=TRUE; }
void main() {
while (TRUE){
 if A {A=FALSE; Service_A(); }
 if B { B=FALSE; Service_B(); }
 if C { C=FALSE; Service_C(); }
...
}
}
```

- max. válaszidő: tA+ tB+ tC+ ...(+ IT) csak a jelzés gyorsul, a kiszolgálás nem

4. Mi a kezdeti beállítása annak a számláló típusú szemafornak, amely öt ekvivalens erőforrás és az azokat felhasználni kívánó taszkok futását hivatott szinkronizálni (max. 1 pont)? Egy szemafor foglalásának (beállítástól függően) milyen következményei lehetnek a taszkokra nézve (max. 1 pont)?



A foglalás következményei:

- wait forever → a task blokkolva addig, amíg nem szabadítjuk fel
- wait with a timeout → a task blokkolva addig, amíg nem szabadítjuk fel vagy lejár a time-out
- do not wait → megkéri a szemafor token-t, de ha nem szabad, akkor nem blokkolódik

5. Hasonlítsa össze beágyazott operációs rendszerek és az asztali számítógépek operációs rendszerek főbb jellegzetességeit (max. 2 pont)! Sorolja fel a beágyazott valós-idejű kernelek feladatait (max. 1 pont)! Mire szolgál a beágyazott virtualizáció (max. 1 pont)?

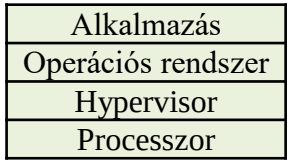
Összehasonlítás:

Rendszerindulás, programkomponensek szervezése, védelem, skálázhatóság különbségeinek kifejtése a honlapon megtalálható jegyzet szerint.

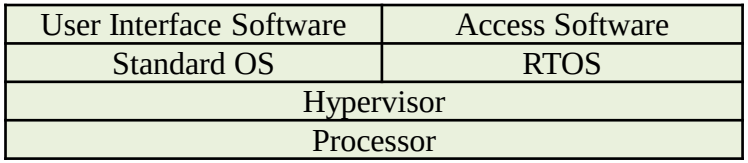
A Kernel feladatai:

- A konkurens (kvázi-parallel) feladatok végrehajtása task-ok vagy szálak (threads) formájában:
  - a task állapotok kézbentartásával és a task-ok sorba állításával,
  - a task preempciók végrehajtásával (gyors context switching) és gyors IT kezeléssel,
- A CPU ütemezése (a határidők garantálása, a task várakozások minimalizálása, a számítási teljesítmény méltányos szétosztása);
- A task-ok szinkronizálása (kritikus szakaszok, szemaforok, monitorok, kölcsönös kizárás);
- A task-ok közötti kommunikáció (bufferelés);
- A valós-idejű óra belső referenciaként történő támogatása.

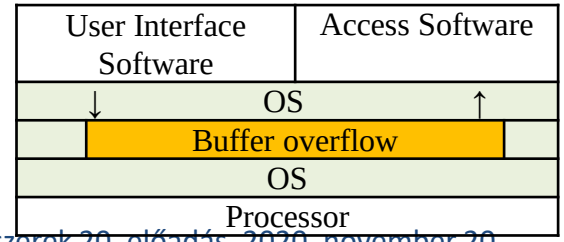
Vitualizáció: a szoftver hordozhatóságot szolgálja, azaz fusson különféle hardvereken. A virtuális gép (VM: Virtual Machine) olyan szoftver környezetet biztosít az adott szoftver számára, mintha tényleges hardveren futna az alábbi struktúrában:

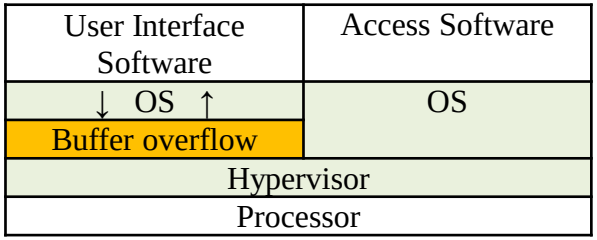


Konkurens operációs rendszerek virtuális gépen



A biztonság növelése virtualizáció alkalmazásával

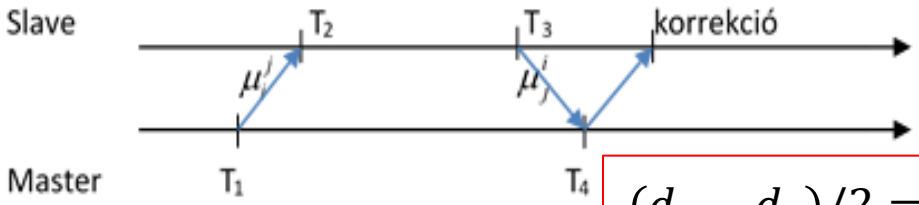




Licensz elválasztás virtualizáció alkalmazásával

User Interface Software	Access Software
Linux GPL	RTOS
Stub (csonk)	Driver
Hypervisor	
Processor	

6. Mutassa be a master-slave óraszinkronizáció lépéseit és összefüggéseit (max. 2 pont)! Milyen hibát követhetünk el akkor, ha a szinkronizálandó óra a szinkronizációt megelőzően sietett (max. 1 pont)?



$$d_1 = C_j(T_2) - C_i(T_1)$$
$$d_2 = C_i(T_4) - C_j(T_3)$$

$$(d_1 - d_2)/2 = \varepsilon_j + (\mu_i^j - \mu_j^i)/2 - (E_j^1 - E_j^2)/2 = \varepsilon_j + h_j,$$

Óra visszaállítás inkonzisztens időbélyegeket eredményezhet …

7. Hogyan működik, és mit szolgál a módosított frekvenciamodulációs kódolás (max. 2 pont)?

Módosított frekvenciamodulációs kód: szinkronizáló

11010001: órajel és adatjel pozíciók helyezkednek el egymást váltva. “1”: jelváltás történik, “0”: nem történik jelváltás az adatjel pozícióban. Ha több mint két “0” van egymás után, akkor az órajel pozícióban jelváltás lesz.

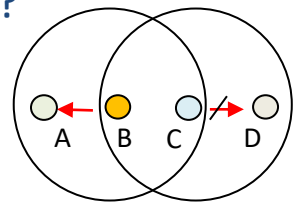
8. Két, külön-külön $K = 0.8$ készenléti tényezőjű részegység sorba kapcsolásával megvalósított funkció eredő megbízhatóságát redundancia alkalmazásával $K' \geq 0.9$ készenléti szintre emeltük. Milyen és hányszoros redundanciát kell alkalmaznunk az egyes részegységek szintjén (max. 3 pont)?

A sorbakapcsolás redundancia nélkül $0.8 * 0.8 = 0.64$ készenléti tényezőjű rendszert eredményez.

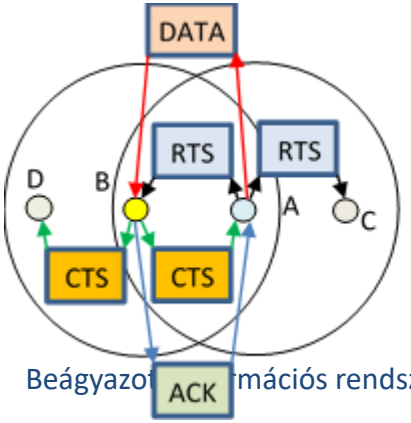
Részegységenként duplikálás $1 - (1 - 0.8)^2 = 0.96$ értékű készenléti tényezőt eredményez, amivel az eredő készenléti tényező: $0.96 * 0.96 = 0.9216 > 0.9$

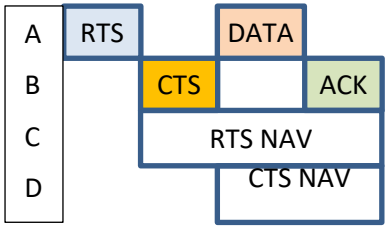
Kétszerezés és párhuzamos redundancia szükséges.

9. Mit nevezünk látható terminál problémának (max. 1 pont)? Mire szolgál, és hogyan működik az RTS/CTS algoritmus? Célszerűen mit kell tartalmazzon a RTS/CTS üzenet (max. 2 pont)?



B ad A-nak
C is szeretne adni D-nek!
C hallja B-t
C nem ad, bár nem okozna ütközést





- Az „A" adó RTS üzenetet küld ("B"-nek)
- A „B" vevő CTS üzenettel válaszol
- Az adó a CTS vétele után továbbítja az adatcsomagot

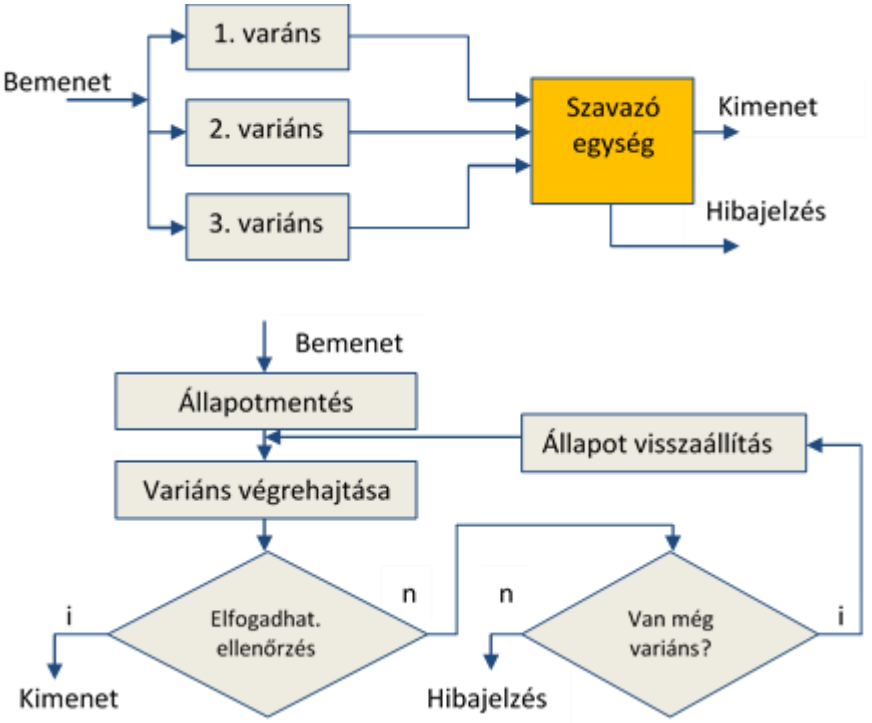
A többi csomópont RTS, CTS vétele után nem adhat!
(NAV = Network Allocation Vector)

Az RTS/CTS üzenet a vevő címét és az üzenet hosszát kell tartalmazza.

10. Mutassa be a szoftver hibák kezelésére kidolgozott *N*-verziós programozás és a javító blokkok technikája módszereket (max. 2 pont)? Hasonlítsa össze a két módszert az ellenőrzés és a végrehajtás módja, a szükséges időigény és a tolerált hibák számát tekintve (max. 2 pont)?

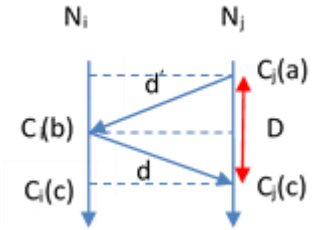
N-verziós programozás: aktív redundancia, a variánsok párhuzamos végrehajtása, többségi szavazás. Ha a variánsok kimeneteire elfogadhatósági tartományt is adunk, akkor a szavazó azt is ellenőrzi. A szavazó maga ún. egyszeres hibapont, azaz ha elromlik, akkor a funkció kiesik, de a szavazó egyszerű, ezért kisebb a kockázat.

Javító blokkok technikája: passzív redundancia, csak hibaesetén aktiválódik. A variánsok kimenetének elfogadhatóságát ellenőrizzük, ha erre nincs lehetőség, akkor a módszer nem alkalmazható. Ha hiba lép fel, akkor tartalék variáns soros végrehajtására kerül sor.



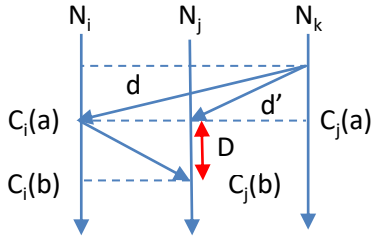
Tulajdonság/típus	N-verziós programozás	Javító blokkok
Ellenőrzés	Szavazás, relatív	Elfogadhatóság, abszolút
Végrehajtás	Párhuzamos	Soros
Időigény	Leghosszabb variáns v. time-out	Hibák számától függ
Redundancia aktiválása	Mindig	Csak hiba esetén
Tolerált hibák	[(N-1)/2]	N-1
Hibakezelés	Maszkolás	Helyreállítás

11. Elosztott rendszerben kétirányú (round-trip) kommunikáció alkalmazásával szinkronizálunk órákat. A szinkronizálandó órával mért round-trip ideje $10\ ms$, miközben a hálózatra specifikált üzenettovábbítási idő minimuma $d_{min} = 3\ ms$, a maximum pedig $d_{max} = 6\ ms$. Ilyen adatok mellett mekkora a szinkronizáció bizonytalansága (max. 3 pont)? Miért tud ezen javítani a referencia broadcasting szinkronizáció (max. 1 pont)?



Mivel $4 \leq d' \leq 6$, ill. $4 \leq d \leq 6$ tartományokban lehet úgy, hogy $d' + d = 10$, Ezért a szinkronizáció bizonytalansága $\pm 1\ ms$.

Azért javít, mert D nagyon jó közelítéssel megadja az üzenettovábbítási időt.



12. Idővezérelt architektúrát (TTA) alkalmazunk. A csomópontokat összekötő $100kbit/s$ sávszélességű terepi buszon, 90%-os sávszélesség kihasználtság mellett, minden egyes klaszter-körüljárás során, minden egyes csomópontból a többiek valamelyike felé keretenként 50 bájt információt akarunk továbbítani. Elhelyezhető-e 8 csomópont egy klaszteren belül, ha a csomópontok $40\ msec$ gyakorisággal szeretnének egy-egy keretet megkapni/frissíteni (max. 2 pont)?

Mivel 8 csomóból 400 bit 40 msec alatt $80kbit/s$ sávszélesség igénynek felel meg, ezért elhelyezhető.

Az első előadás diái: 2019. szeptember 11.
2-12. előadás és 1-6. gyakorlat óravázlatai 2017
Előadásvázlat kiegészítés: Total Bandwidth Server

Beágyazott rendszerek szoftver architektúrái
Beágyazott operációs rendszerek
Beágyazott rendszerek ellenőrzéstechnikája
Az idővezérelt architektúra

Szenzorhálózatok I.
Szenzorhálózatok II.
Szenzorhálózatok III.

Beágyazott információs rendszerek előadás diái 2020.09.09.
Beágyazott információs rendszerek elöadás diái 2020.09.10.
Beágyazott információs rendszerek előadás diái 2020.09.16.
Beágyazott információs rendszerek előadás diái 2020.09.17.
Beágyazott információs rendszerek előadás diái 2020.09.24.
Beágyazott információs rendszerek előadás diái 2020.09.30.
Beágyazott információs rendszerek előadás diái 2020.10.01.
Beágyazott információs rendszerek előadás diái 2020.10.07.
Beágyazott információs rendszerek előadás diái 2020.10.08.
Beágyazott információs rendszerek előadás diái 2020.10.14.
Beágyazott információs rendszerek előadás diái 2020.10.15.
Beágyazott információs rendszerek előadás diái 2020.10.21.
Beágyazott információs rendszerek előadás diái 2020.10.22.
Beágyazott információs rendszerek előadás diái 2020.10.28.
Beágyazott információs rendszerek előadás diái 2020.10.29.
Beágyazott információs rendszerek előadás diái 2020.11.04.
Beágyazott információs rendszerek előadás diái 2020.11.05.
Beágyazott információs rendszerek előadás diái 2020.11.18.
Beágyazott információs rendszerek előadás diái 2020.11.19.