Szenzorhálózatok alkalmazása valósidejű (visszacsatolt) rendszerekben

Összeállította:

Orosz György

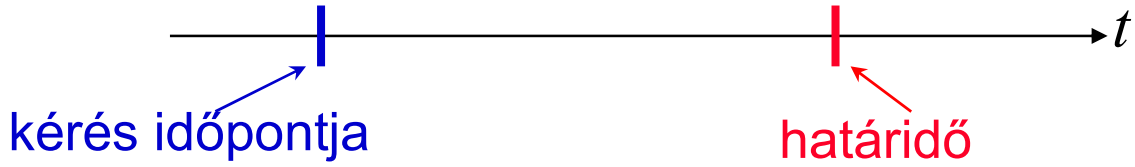
2011. 11. 09.

Hivatkozások

[1] Karl Henrik Johansson, „Control over Wireless Networks: Research Challenges and Case Studies”, *GRASP Seminar Series University of Pennsylvania*, Philadelphia, USA, March 24, 2006
[2] Karl Henrik Johansson: „Wireless Control Opportunities and Challenges”, *Automationsdagarna*, Stockholm,4-5 Feb, 2009
[3] Fredrik Linnarsson, Peng Cheng, Bengt Oelmann, „Analysis of the IEEE 802.15.4 Standard for a Wireless Closed Loop Control System for Heavy Duty Cranes”, *IEEE Second International Symposium on Industrial Embedded Systems - SIES'2007*, Lisbon, Portugal, 4-6 July 2007. pp 164-169
[4] Nick Ploplys, Andrew Alleyne, „Closed Loop Over a Wireless Network (CLOWN) ”

Valósidejű működés

- A feladatot egy adott időintervallumon (határidőn) belül be kell fejezni



- Soft real-time
  - Határidő elmulasztása nem jár súlyos következménnyel
  - Tipikusan sec nagyságrendű határidők (pl. honlap letöltés)
  - Várható válaszidőre tervezhetünk
- Hard real-time
  - Határidő elmulasztása súlyos következménnyel jár (pl. fékparancs)
  - Worst-case válaszidőre tervezünk
  - Tipikusan msec nagyságrendű határidők
    - Lehet hosszabb is: pl. június 20-dikán be kell adni a diplomatervet

Szenzorhálózati alkalmazási példák

- Soft real-time
  - Lassan változó mennyiségek érzékelése és szabályozása
    - Hőmérséklet érzékelés (épületek felügylete)
    - Fényerősség érzékelés
  - Általában több másodperces időállandó
- Hard real-time
  - Biztonságkritikus alkalmazás
    - Pl. tűzérzékelés
  - Gyorsan változó jelek
    - Audió és videójelek átvitele
    - Akusztikus jelek érzékelése
      - Pl. lokalizáció: elképzelhető beavatkozás is a pozíció alapján
  - **Visszacsatolt rendszerek**
    - Beavatkozás a mérések alapján: fontos, hogy milyen gyors a beavatkozás hatása
    - Mozgó robotok irányítása
    - Autók egymáshoz képest mért helyzetének szabályozása
    - Beavatkozás akusztikus jelek alapján: aktív zajcsökkentés

Alkalmazási példák

- Az egyes egységek (pl. autók) szenzorok segítségével érzékelik a környezetüket. Az érzékelt jelek alapján egymással vezeték nélkül kommunikálva irányítják az egységek helyzetét, sebességét.
- Vezetéknélküli kommunikáció kritikus: az egységek nem lehetnek fizikailag összekapcsolva.
- Folyamatosan figyelni kell a beavatkozások hatását: visszacsatolás
- Előnyök:
  - Utak hatékonyabb kihasználása
  - Veszélyes területek felderítése
- Valós idejű működés fontos:
  - Körülmények gyors változása
  - Emberek testi épsége

Járművek kooperatív irányítása (távolság tartása)

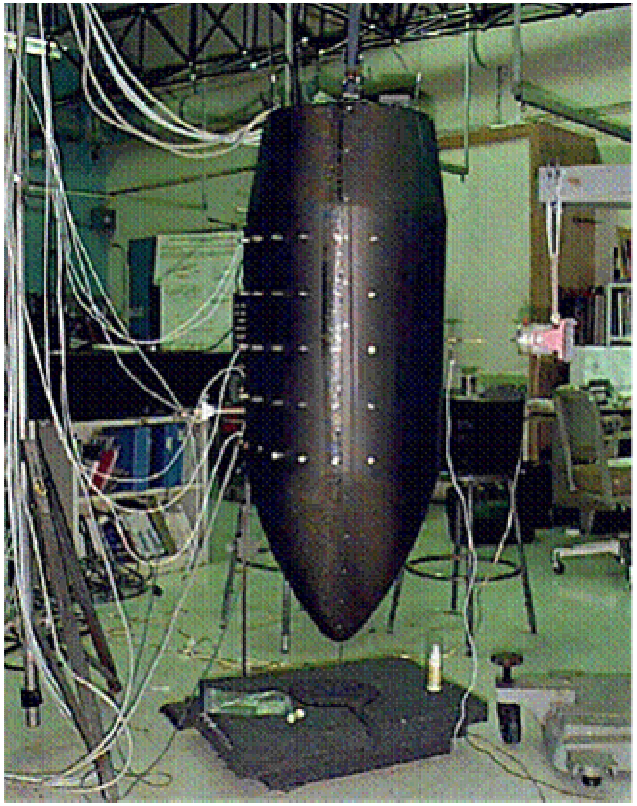
[2]

Mobil egységek irányítása (személyzet nélkül)

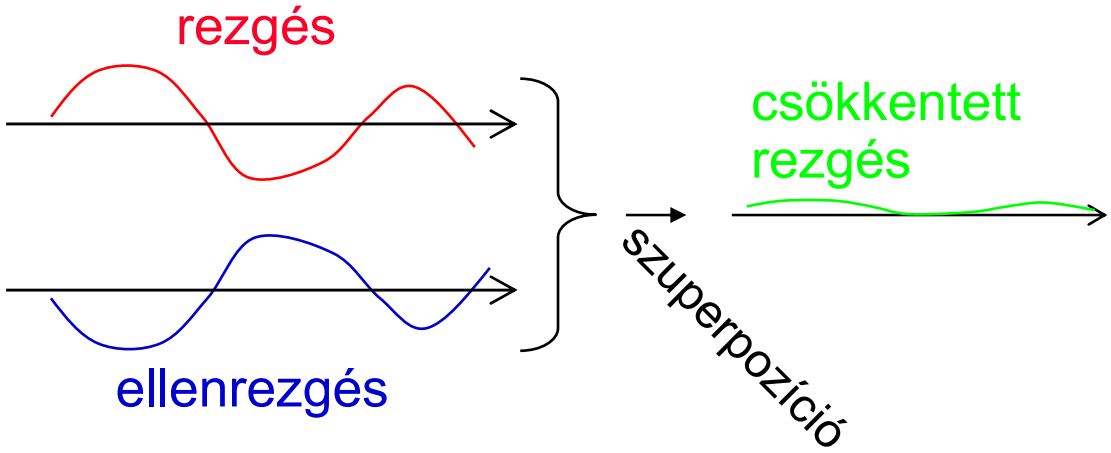
[1]

Alkalmazási példák

Aktív zaj-/rezgéscsökkentés rakétában



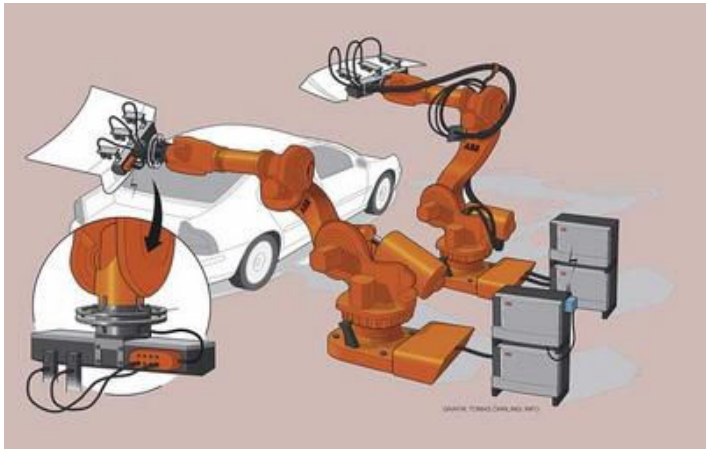
- Aktív zaj- / rezgéscsökkentés:
  - Ellenhullámmal (ellenzaj) kioltjuk a káros rezgéseket / hanghullámokat
  - Főleg kisebb frekvencián hatékony
- Előny:
  - Kisebb tömegű szigetelés
- Folyamatosan érzékelni kell a rezgéseket: visszacsatolás



Alkalmazási példák

- Jelek érzékelése és átvitele mozgó egységek között:
  - Mozgó egységek közötti adatátvitel egyszerűbb vezeték nélkül.
  - Kontaktusok megbomlásának (kontakthibák) elkerülése

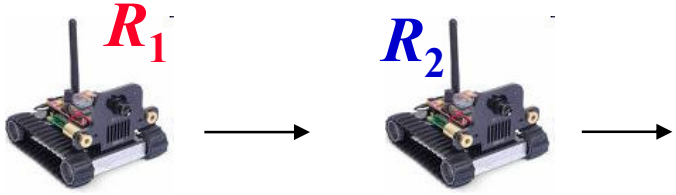
[3]



[1]

Kooperatív vezetéknélküli visszacsatolt rendszerek

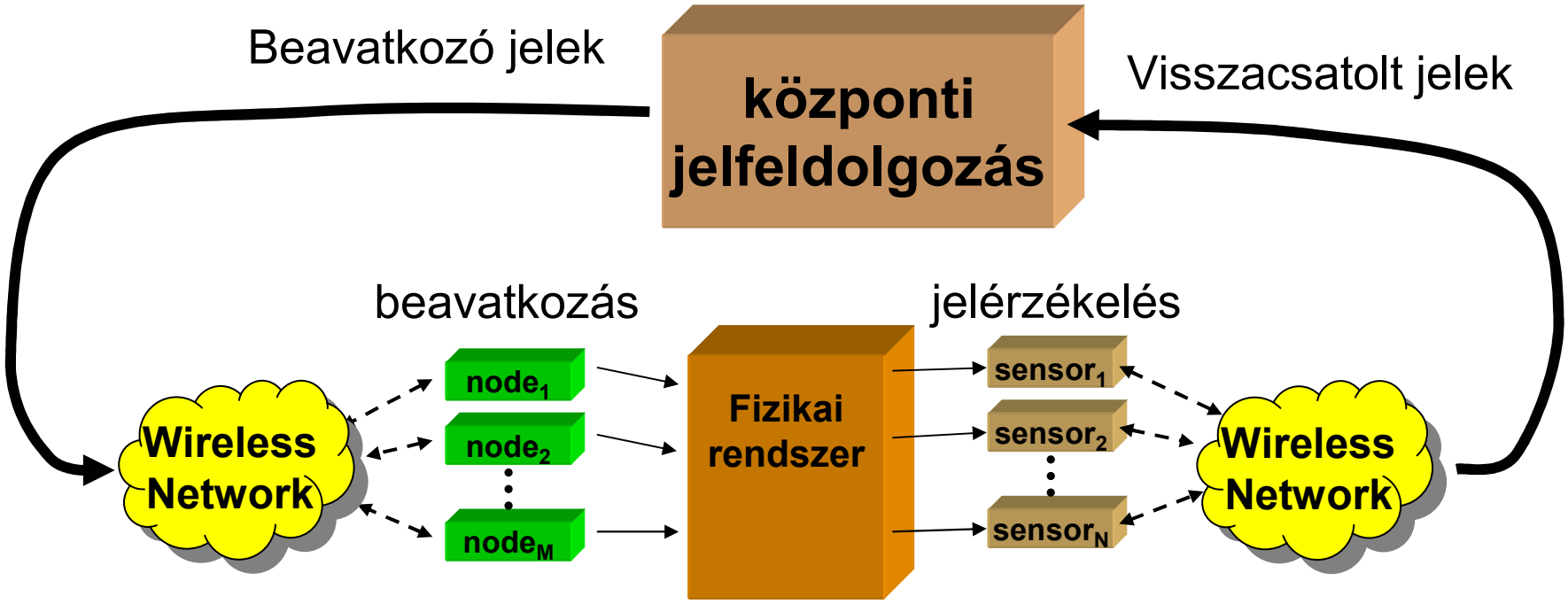
- Nem egyszerűen az egységek egyedi irányítása/szabályozása
- Fontos a kommunikáció: kooperativitás
- Pl.: robotok pozíciójának szabályzása: egymás közötti távolság tartása
  - Mindkét egység méri a távolságot és autonóm módon próbál adott d_0 távolságot tartani. Gond: távolságmérést hiba terheli:
    - Véletlen hiba: idő során átlagolódik
    - Ofszet (rendszeres hiba): nagy problémát okoz
      - Legyen a két egység megkívánt távolsága d_0 és a valódi távolság d=d_0
      - R_1 egység h_1 hibával mér, R_2 egység h_2 hibával mér (legyen h_1<0 és h_2>0)
      - Ha nincsen kooperáció:
        - R_1 egység közeledik R_2 egységhez
        - R_2 egység távolodik R_1 egységtől
        - Végeredmény: „kergetik” egymást
        - Meg kell egyezni egy távolságban: nem elegendő különálló szenzorok alkalmazása



Vezetéknélküli szenzorhálózatok valósidejű visszacsatolt rendszerekben

- Egyszerű telepíthetőség
  - Nem kell meglévő infrastruktúra
  - Egyszerűbben skálázható, nem kell új csatlakozási pontokat kialakítani
  - Kisebb telepítési költség
- Mobil elrendezés
  - Vezeték: fizikai korlátozás
  - Kevesebb korlátozás a mozgásban: pl. robotok pozícionálása, mozgó (rész)egységek felügyelete
  - Egyszerűbb átkonfigurálni a rendszert
- Monitorozás és diagnosztika egyszerűbb: nem kell kívülről csatlakozni
- Elosztott inteligencia
  - Előfeldolgozás lehetséges már a szenzornál
  - Tehermentesíti a központi egységet
  - Alapvető döntés már helyben is megszülethet
- Biztonságkritikus alkalmazásban redundancia: vezeték megrongálódása esetén másodlagos átviteli csatorna

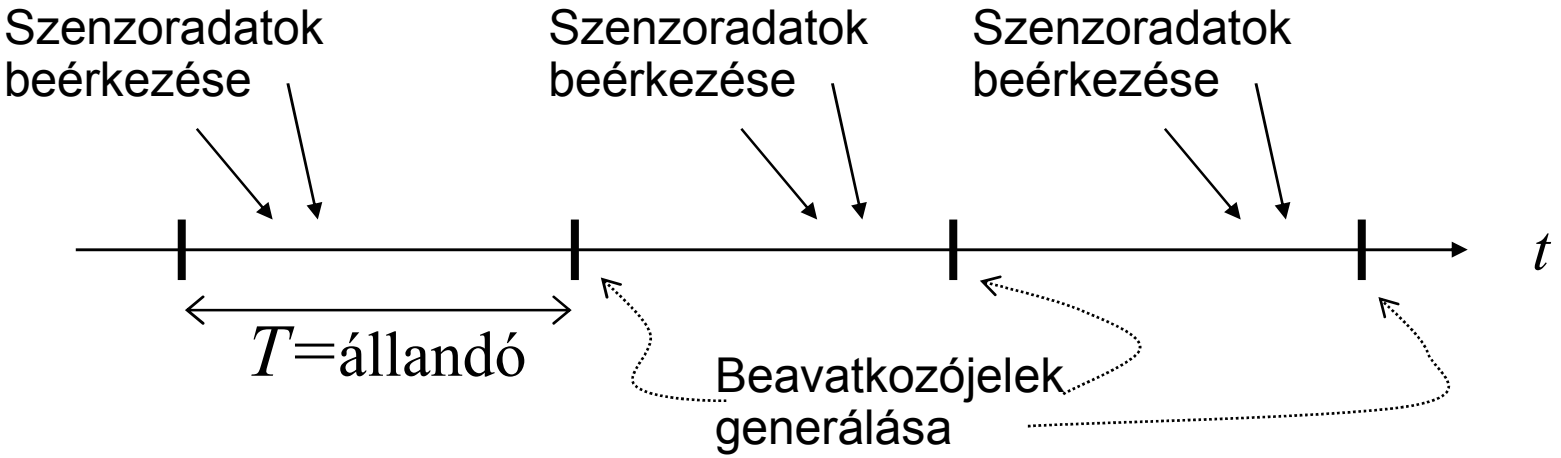
Visszacsatolt rendszerek felépítése



- Szenzor: érzékelés általában kis energiával, így kézenfekvőbb terület
- Beavatkozás: nagyobb energia, így kell tápvezeték, de a jelvezetékek helyettesíthetőek.
  - A meglévő tápellátás kedvező az energiagazdálkodás szempontjából

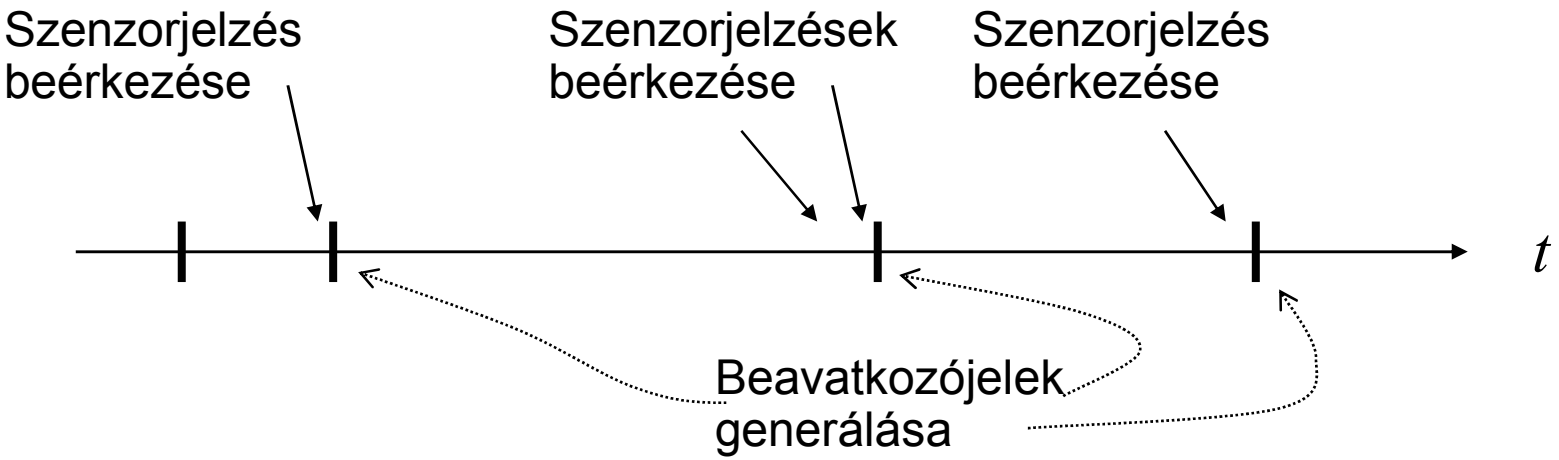
Valósidejű rendszerek jellemző működési modelljei

- Periodikus (idővezérelt) működés
  - A szenzorok adott T időközönként szolgáltatják az adatokat
  - A központi feldolgozó egység T időközönként fogadja az adatokat és legenerálja a beavatkozó jeleket (válaszjeleket)
  - Jól kidolgozott algoritmusok
  - Determinisztikus működés
  - Az előadás során nagyrészt idővezérelt rendszerekkel foglalkozunk



Valósidejű rendszerek jellemző működési modelljei

- Eseményvezérelt működés
  - A központi egység az adatok beérkezésekor egyből feldolgozza azokat és legenerálja a válaszjeleket.
  - Általában bonyolultabb algoritmust eredményez
  - A szenzorok csak adott esemény bekövetkeztekor jeleznek (pl. a szemben lévő jármű adott távolságnál közelebb került)
  - Nem egyenközű a feldolgozás
  - Részleges mérések alapján dönt a központi egység



Real-time rendszerekben való alkalmazás kihívásai

- Kevés a tapasztalat, és az elméleti háttér még nem teljesen kidolgozott
- (Változó) késleltetés
  - Okai:
    - Szoftveres késleltetések
    - Rádiós csatorna foglaltsága
    - Közeghozzáférési stratégia (pl. p-perzisztens CSMA: véletlen várakozás az adás kezdése előtt)
  - A determinisztikus viselkedést rontja
  - Nem kiszámítható a rendszer
- Elosztott érzékelés és jelfeldolgozás
  - Lazán csatolt rendszerek (adatcsere nem közvetlen)
  - Szinkronizáció

Real-time rendszerekben való alkalmazás kihívásai

- Adatvesztés
  - Elkerülhetetlen a rádiós átvitel miatt
  - Javítható pl. acknowledge csomagokkal. Megéri?
- Bizonytalan adatátvitel: bizonytalan információk alapján hozunk döntéseket
- Kis adatátviteli sávszélesség
  - Tipikus sávszélesség <1Mbps
  - Kis fogyasztás ← kevés kommunikáció
  - Sok szenzor
    - Megosztott csatorna
    - A sávszélesség megoszlik az összes szenzor között

Tervezési irányelvek

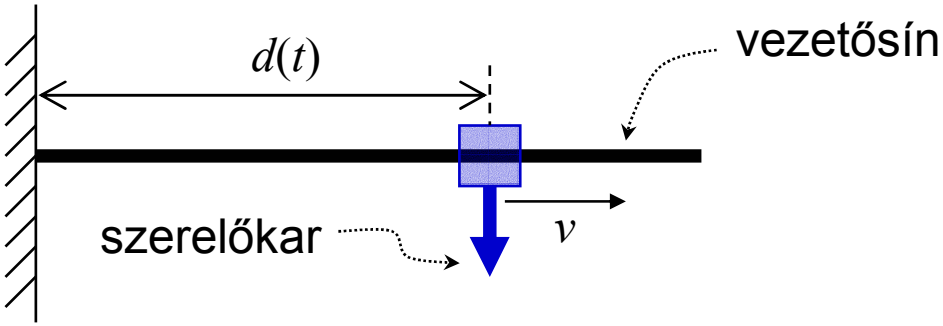
- Nem tekinthetünk el a hálózat hatásától és az alkalmazás specialitásaitól.
- Hagyományos megközelítések:
  - Control-aware networking: megtervezzük a jelfeldolgozó rendszert és próbáljuk minimalizálni a hálózat hatását megfelelő kommunikációs réteg megtervezésével.
  - Network-aware control: a hálózat által kínált lehetőségeket adott peremfeltételeknek tekintjük és emellett tervezzük meg a visszacsatolási stratégiát.
- A két elvet egymással párhuzamosan érdemes alkalmazni a legjobb eredmény elérése érdekében.
- A szabályozás minősége függ a hálózat teljesítőképességétől és paramétereitől.

Késleltetés problémája

- A késleltetést minimalizálni kell mert az csökkenti a reakcióidőt, akár súlyos problémákat is okozhat. Hétköznapi példák:
  - Zuhany: a víz hosszú késleltetés után jut ki a zuhanyrózsán, így a hőfok beállítása hosszú iteráció után lehetséges pl. a sima csaptelephez képest. Ha kapkodunk és gyorsan változtatjuk a víz hőmérsékletét, akkor a zuhanyrózsához érő víz túl forró lehet, mire ezt észleljük. (Ez is visszacsatolt rendszer: a víz érzékelt hőmérséklete alapján állítjuk a vízcsapot)
  - Szoftverek késleltetése: egy gombra kattintva, pl. ablak bezárása, nem reagál a szoftver (nem zárul be az ablak). Ha túl sokáig nem történik semmi, akkor arra gondolhatunk, hogy nem észlelte a kattintást és újra rákattintunk a gombra. Amennyiben csak a késleltetés miatt nem történt semmi, akkor egy idő múlva az első kattintás hatására bezárul az aktuális ablak, a második kattintás pedig az alatta lévő ablakon történik, így azt is bezárjuk. (visszacsatolás: folyamatosan észleljük a beavatkozásunk, tehát a kattintás, hatását, és ennek megfelelően cselekszünk)
- A fix késleltetés kompenzálható (pl. tudjuk, hogy mennyi a szoftverek maximális késleltetése)

Példa a késleltetés kompenzálására

- Legyen adott egy fix $\Delta\tau$ késleltetés a szenzor és a központi egység között. Ez meghatározható előzetes mérésekkel, vagy a hálózat paramétereiből és az egységek működéséből számítható.
- Feladat: egy adott test pozíciójának beállítása (pl. gyártó robot karja).

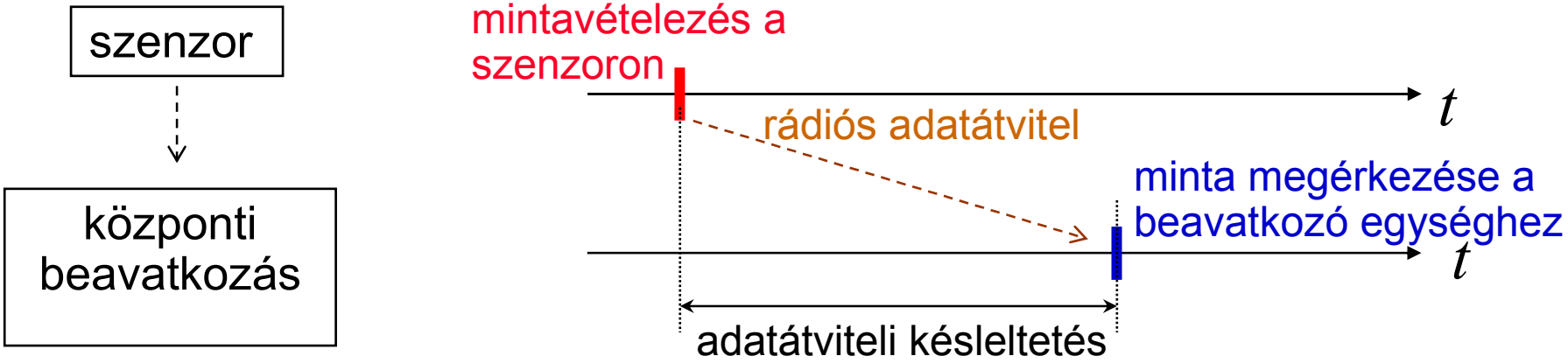


- A kar pozícióját [$d(t)$] és sebességét [$v(t)$] a szenzor elküldi a központi egységnek.
- A szabályozó egység megkapja az adatokat a $t' = t+\Delta\tau$ időpontban.
- A valós pozíció meghatározható a t' időpontban: $d(t') \approx d(t)+v(t)\Delta\tau$
- A döntés időpontjában (t') tehát ismert a pozíció becslése: $d(t')$

Változó késleltetés problémája

- Konstans késleltetés esetén a fenti gondolatmenet kiterjeszthető általános esetre is, pl. ha a rendszer az állapotegyenletével adott → kiszámítható a viselkedése
- Network aware control stratégia: késleltetés figyelembe vétele
- A késleltetés akár minden adatküldésnél egyenként is mérhető.
- A változó késleltetés különösen a központi vezérlő és a beavatkozó szerv között gond, mert ott nem kompenzálható a már központilag kiszámított beavatkozó jel. (intelligens beavatkozó esetén a beavatkozó egység is kompenzálhat)
- Változó késleltetés: súlyosabb probléma, mint a késleltetés (nem kompenzálható)
- Az előre számított / mért $\Delta\tau$ késleltetéstől eltér a valódi késleltetéstől
- A valódi késleltetés nem ismert
  - Amennyiben az előző példában $\Delta\tau$ bizonytalansága $h_{\Delta\tau}$=$10\mathrm{msec}$, és pl. a kar sebessége $1\mathrm{m/s}$, akkor a pozícióbecslés bizonytalansága $1\mathrm{cm}$. Nagy késleltetés instabilitást okoz. A probléma analitikusan is vizsgálható.

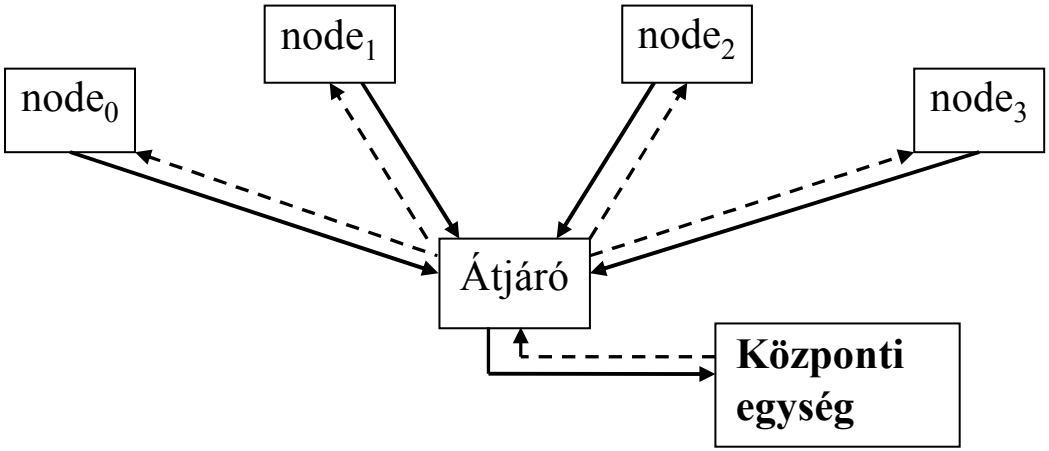
Állandó késleltetés biztosítása



- A késleltetés rontja a teljesítményt, de a változó késleltetés ennél is rosszabb: kiszámíthatatlan működés.
- Cél: a késleltetés állandó szinten tartása.
- Control aware networking: biztosítjuk az állandó késleltetést
- Feltétel: determinisztikus hálózati működés.

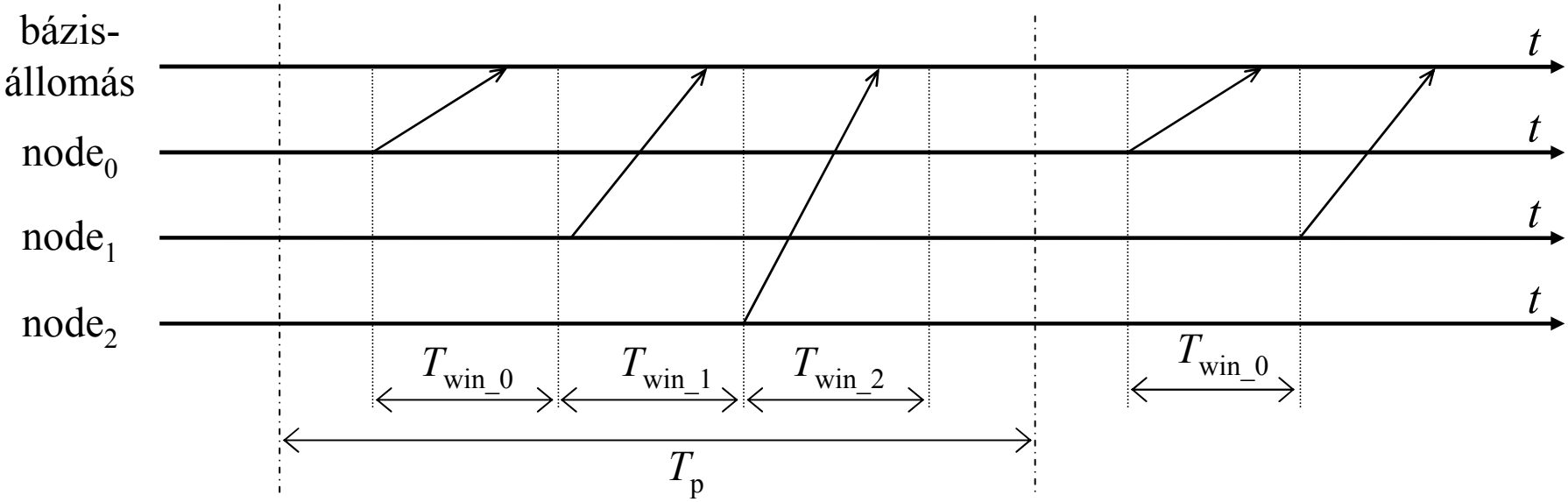
Állandó késleltetés biztosítása

- Determinisztikus működés:
  - Időosztásos hálózat: nincsen véletlenszerű adatküldés, ami zavarná a rendszert
  - Minden egység kap saját időszeletet
    - Token ring hálózat: egymásnak adják át az adás jogát az egyes node-ok
    - Polling: a központi egység (vagy a vele kapcsolatban lévő bázisállomás) egymás után kérdezi le a szenzorokat.
- Előnyös a csillag struktúra: IEEE 802.15.4. (ZigBee fizikai rétege) definiálja ezt a struktúrát



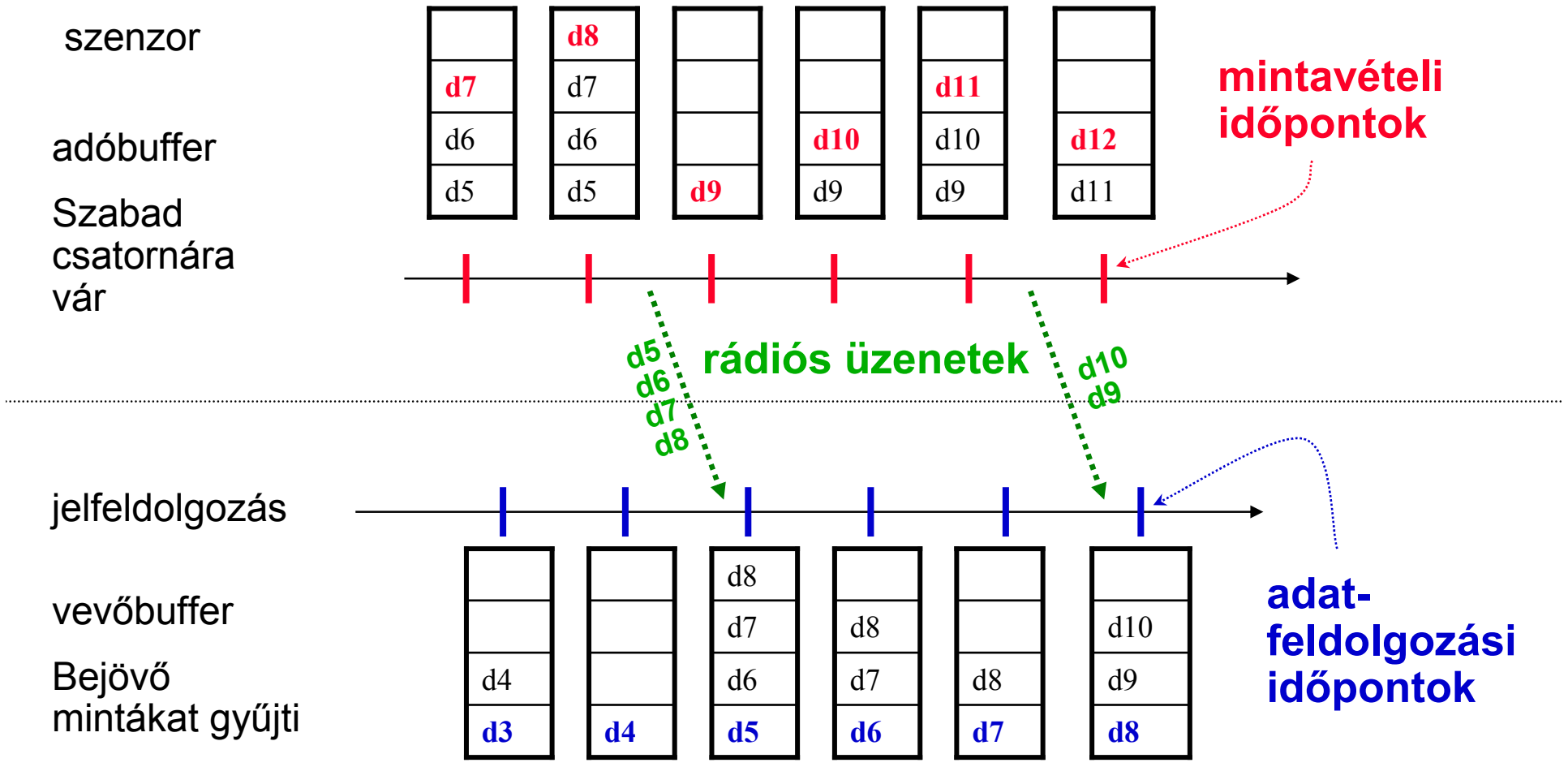
Időosztásos működés

- Minden egység saját időszeletet kap
- Az időszeletek periodikusan ismétlődnek
- Rossz a kihasználtság, ha nem egyenlő a mintavételezési sebesség: kimaradó üres időszeletek, ahol az adott node nem küld adatot
- Az időszeletek között érdemes bizonyos időt hagyni az üzenetek feldolgozására

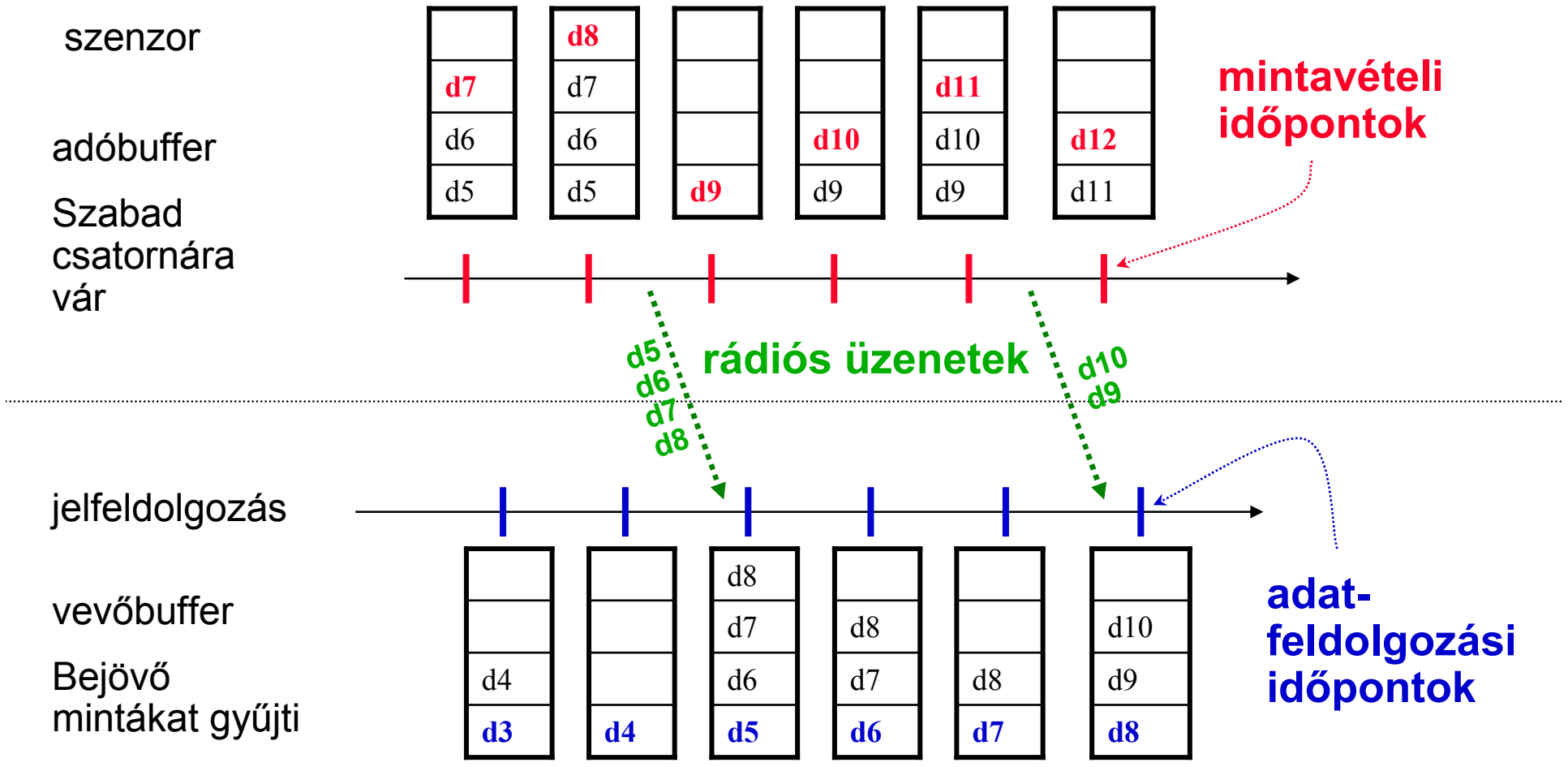


T_{win_i}: i. mote időkerete
T_p: egy hálózati periódus

Determinisztikus működés buffereléssel



Determinisztikus működés buffereléssel

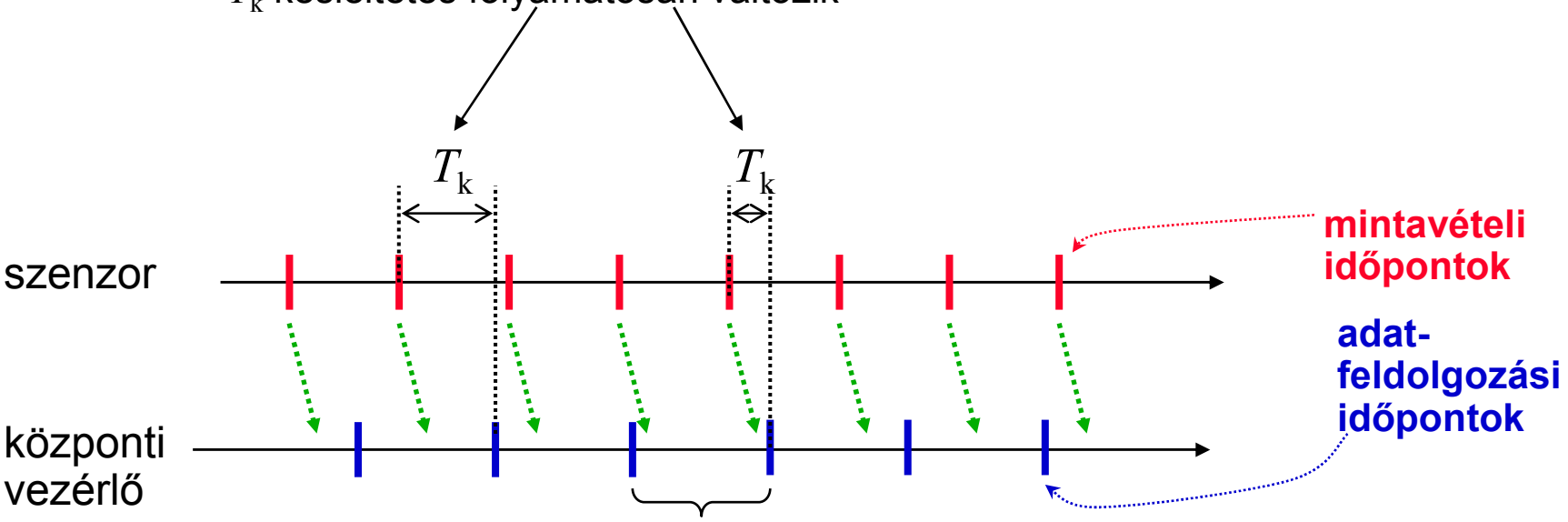


Determinisztikus működés bufferelléssel

- Amennyiben a hálózat átlagos adatátviteli sebessége elegendő, de például a közeghozzáférés ingadozása miatt az átviteli idő ingadozhat. Ez kiküszöbölhető bufferelléssel.
- A bufferelés egy általános technika
  - Az átlagos késleltetést megnöveli, mert legalább annyi adatot kell bufferelni, mint amekkora a leghosszabb késleltetés. ☹
  - Determinisztikussá teszi a viselkedést. ☺
  - Példák egyéb területekről
    - Internetes médiafile-ok lejátszása: bizonyos adatmennyiséget előre letárolunk, így egyenletes a lejátszás
    - Hordozható CD lejátszók rázkódásvédelme

Szenzorok szinkronizálása

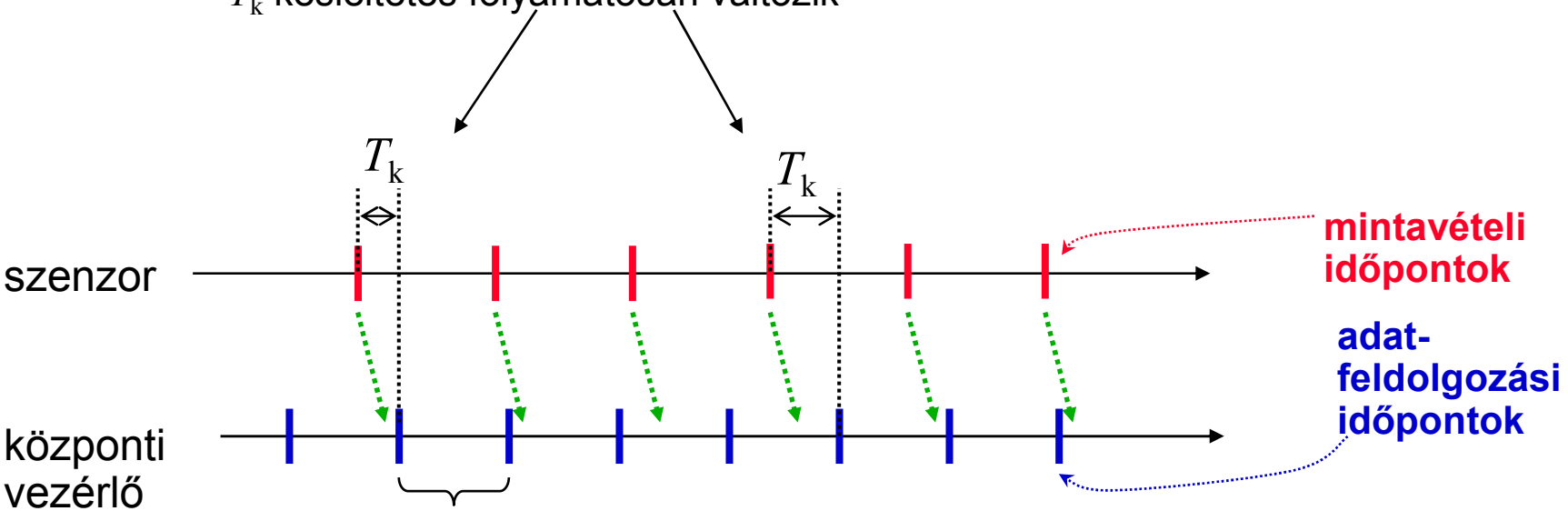
- Probléma: a kvarcok hibája miatt eltérő mintavételezési és adatfeldolgozási sebesség
  - A mintavételezési és adatfeldolgozási pontok elcsúsznak egymáshoz képest még állandó adatátviteli késleltetés esetén is így a késleltetés ingadozik
  - Adat **túl**- vagy alul**csordulás** következhet be a feldolgozás során
  - T_k késleltetés folyamatosan változik



kieső adat: két feldolgozási időpont között két adat érkezett

Szenzorok szinkronizálása

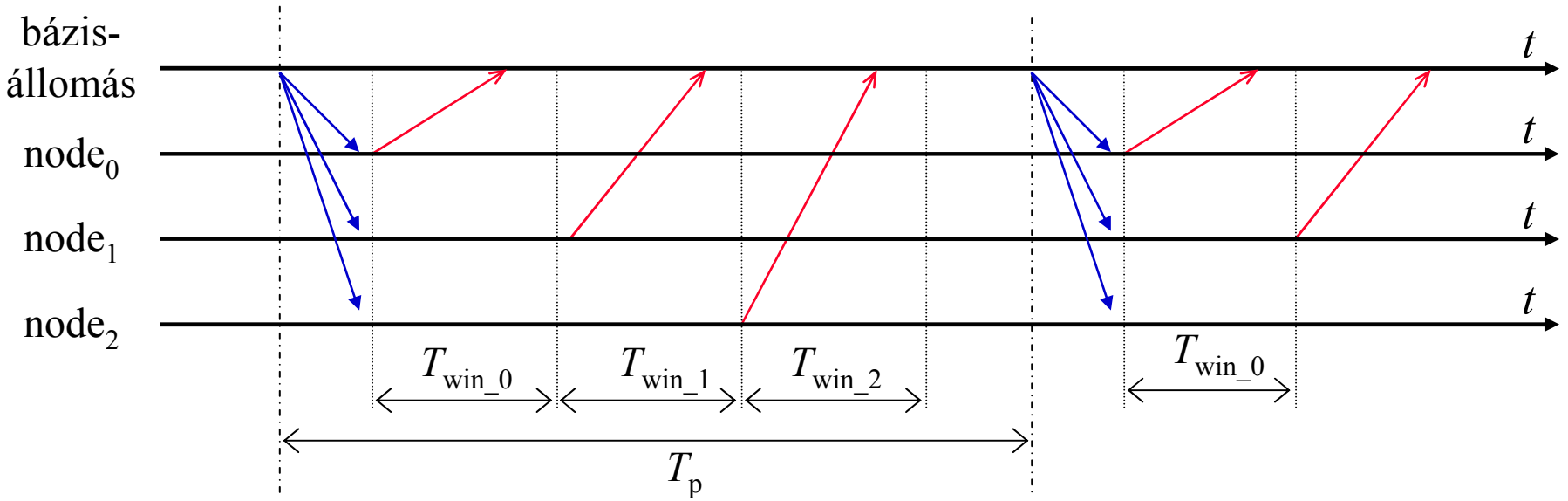
- Probléma: a kvarcok hibája miatt eltérő mintavételezési és adatfeldolgozási sebesség
  - A mintavételezési és adatfeldolgozási pontok elcsúsznak egymáshoz képest még állandó adatátviteli késleltetés esetén is így a késleltetés ingadozik
  - Adat túl- vagy **alulcsordulás** következhet be a feldolgozás során
  - T_k késleltetés folyamatosan változik



hiányzó adat: két feldolgozási időpont között nem érkezett adat

Szenzorok szinkronizálása lekérdezéssel

- A szenzorok nem autonóm módon mintavételeznek, hanem a központi egység minden adatfeldolgozás során lekérdezi a szenzorok értékét.
- Minden szenzor egyszerre kapja a lekérdezést
- Probléma: folyamatos lekérdezés szükséges



T_{win_i}: i. mote időkerete
T_p: egy hálózati periódus

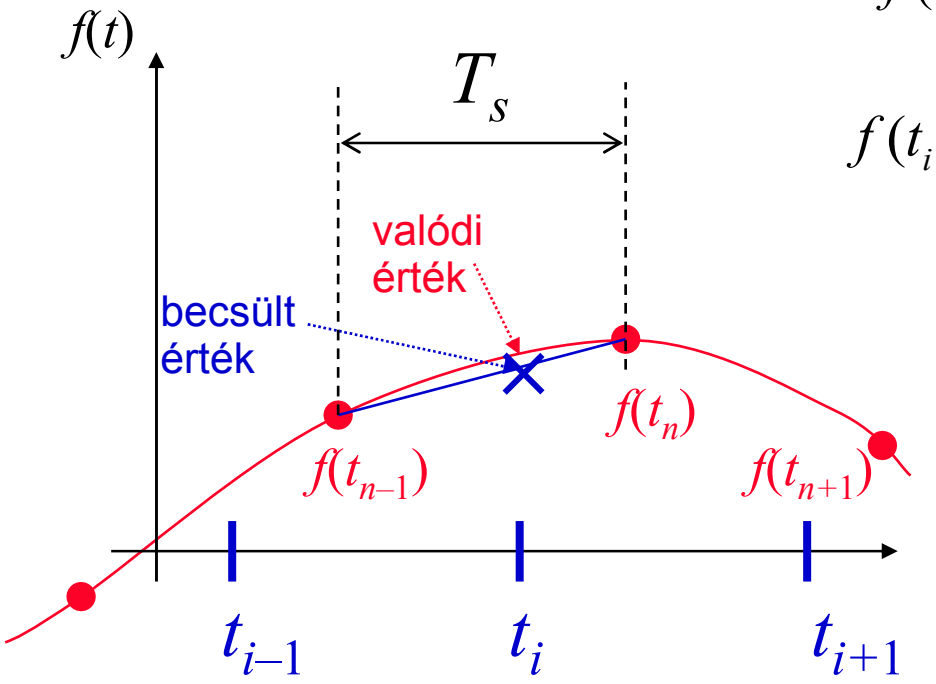
lekérdezés
adatküldés

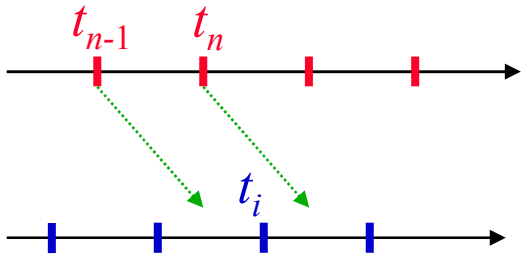
Szinkronizáció interpolációval

- A jelfeldolgozási időpontokban a központi egység valamilyen interpolációs technikával becsli a jel értékét. Példa: lineáris interpoláció:

$$f(t_i) = f(t_{n-1}) + \left[f(t_n) - f(t_{n-1})\right]\frac{t_i - t_{n-1}}{T_s}$$

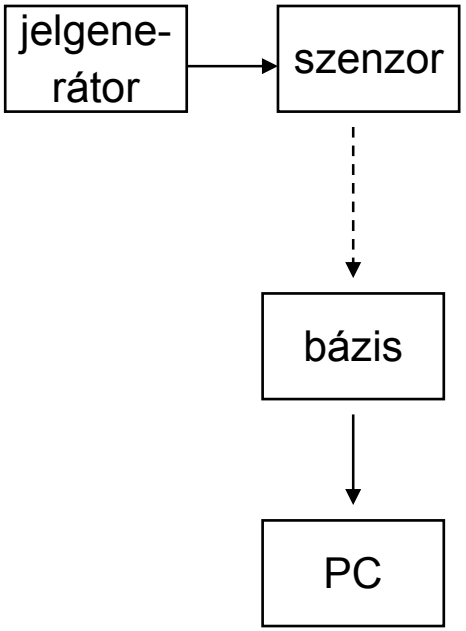
$$f(t_i) = (1-a)f(t_{n-1}) + af(t_n), \quad a = \frac{t_i - t_{n-1}}{T_s}$$

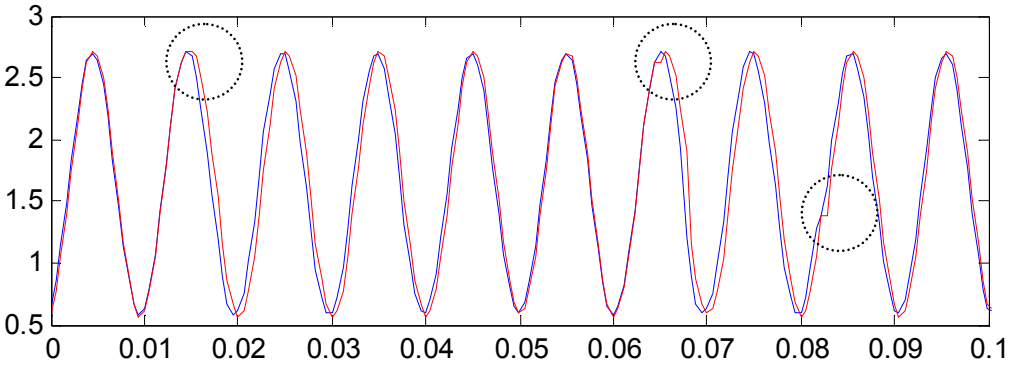


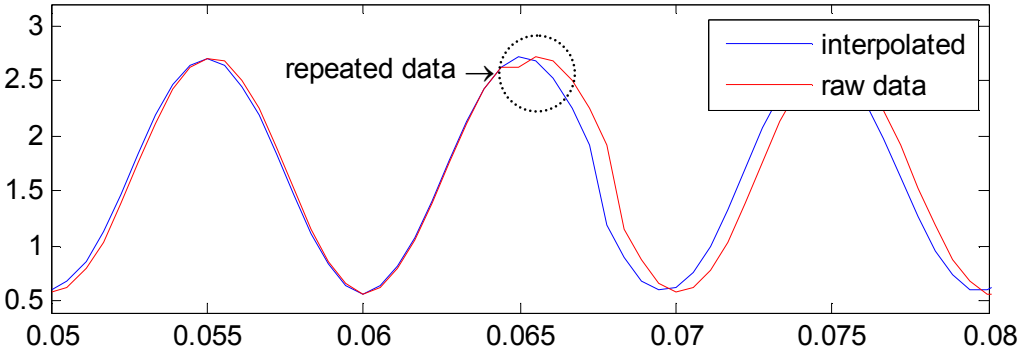


Mérési eredmény lineáris interpolációra

Elrendezés:

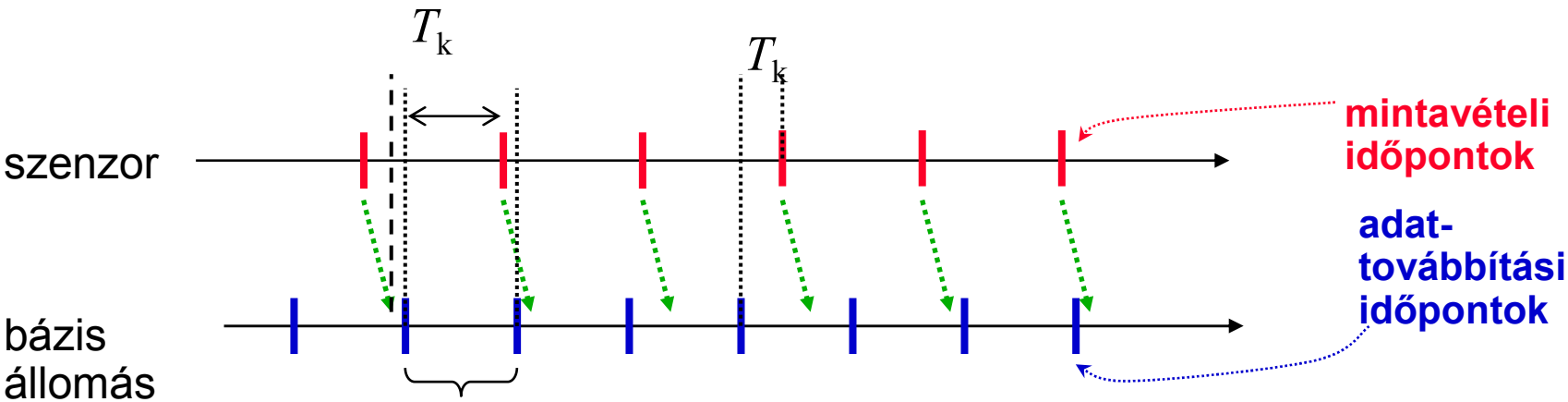






Mérési eredmény lineáris interpolációra

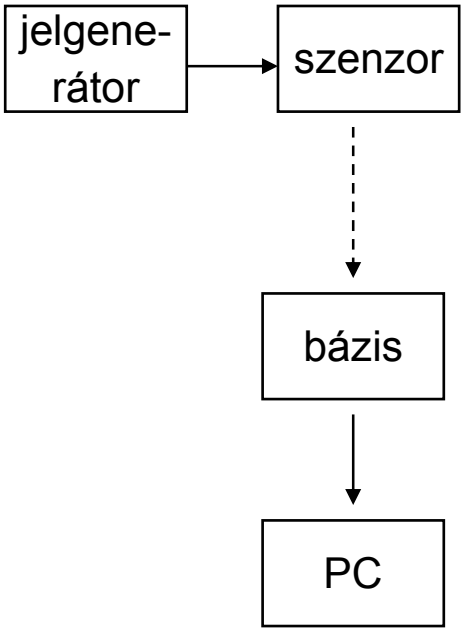
- Időzítési adatok
  - A szenzor 1800Hz-en mintavételezi a jelgenerátor jelét (100Hz-es szinusz)
  - A mintavételezett jelet rádión elküldi a bázisállomáshoz
  - A bázisállomás 1820Hz-es sebességgel továbbítja a legfrissebb mintát és a minta érkezési idejét a PC felé
- A PC-n loggoljuk az adatokat (idő+adat) és elvégezzük az interpolációt



ismétlődő adat: két feldolgozási időpont között nem érkezett adat

Mérési eredmény lineáris interpolációra

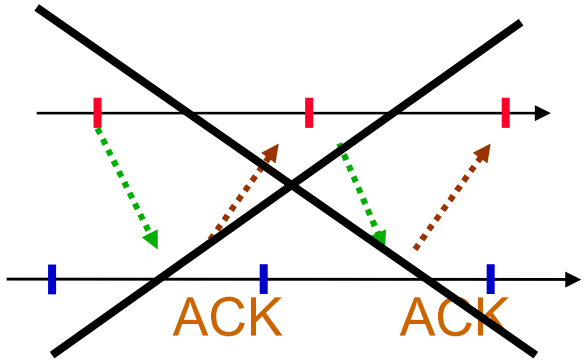
Elrendezés:

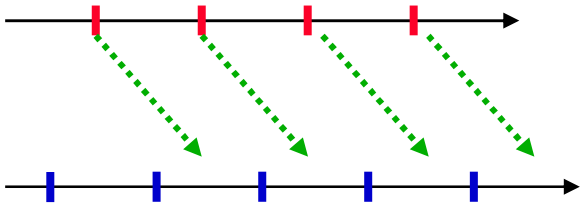


- A szenzor 1800Hz-en mintavételezi a jelgenerátor jelét (100Hz-es szinusz)
- A mintavételezett jelet rádión elküldi a bázisállomáshoz
- A bázisállomás 1820Hz-es sebességgel továbbítja a legfrissebb mintát és a minta érkezési idejét a PC felé
- A PC-n loggoljuk az adatokat és elvégezzük az interpolációt
- Az ábrán látható, hogy vannak ismétlődő minták (piros grafikon), mivel a szenzor lassabban szolgáltatja az adatokat a bázisállomás felé, mint ahogyan a bázisállomás szolgáltatja az adatokat a PC felé.
- A kék grafikon a lineáris interpolációval kapott jelet mutatja. Látható, hogy az interpoláció javítja a jel minőségét, megszünteti a nagy ugrásokat
- Gyorsabban változó jelek esetén a lineáris interpoláció kevésbé használható, növelni kell az interpoláció fokszámát. Ez növeli a késleltetést, mert több mintát kell használni ☹

Adatátviteli módok
Nyugtázott vs. folyamatos

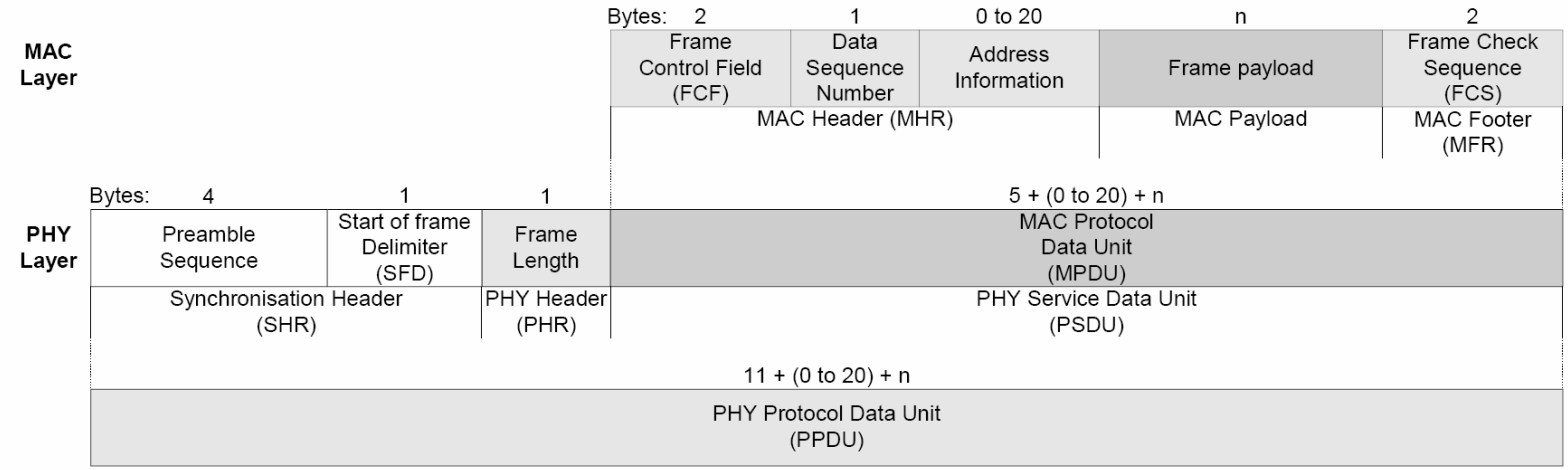
- Az adatvesztés elkerülhetetlen (pl.: csukott szemmel vezetünk)
- Javítható acknowledge (ACK) üzenetekkel. TCP jellegű adatátvitel.
- Probléma:
  - ACK üzenetek sávszélességet foglalnak
  - Torlódást okozhat ha ACK-ra nem kapunk választ vagy többször kell ismételni→ a késleltetés változik.
- Sebességkritikus esetekben folyamatos adattovábbítás ACK nélkül (UDP jellegű adatátvitel: általában pl. multimédiás alkalmazásokban használatos)
- A sávszélességet inkább új adatok továbbítására használjuk nem ACK küldésére és fogadására (az üzenet várása is fogyasztás).
- Visszacsatolt rendszerek esetén számítható, hogy milyen adatvesztési arány mellett biztosítható még a rendszer stabilitása.





Adatátviteli módok
Egyedi vs. tömbös adatátvitel

- Az adatátvitel általában valamilyen csomag alapú rendszeren történik.
- A csatorna kihasználása annál kisebb, minél kevesebb hasznos adatot (payload) továbbítunk egy csomagban.

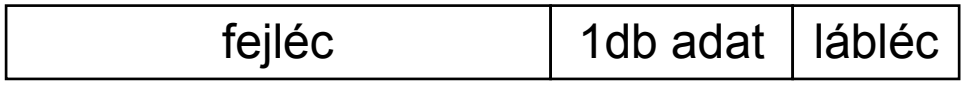


- Példa: IEEE 802.15.4. által definiált szerkezet. Egy csomagban egy adat
- Legjobb esetben 11byte overhead (+interframe space: IFS = 6byte)
- 8 bites (1byte) adatok esetén az eredő üzenethossz 18byte, tehát a hatásfok folyamatos küldés esetén alig több, mint 5,5%!

Adatátviteli módok
Egyedi vs. tömbös adatátvitel

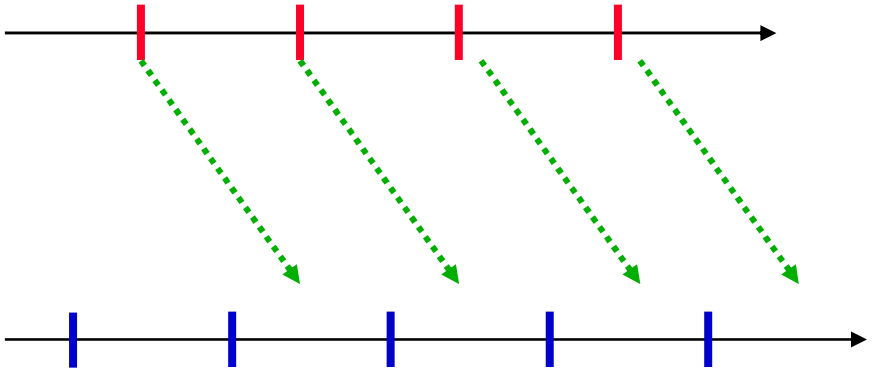
- Egy csomagban egy adat: minden adat elküldésénél megjelenik az overhead.
- Kis csatornakihasználtság.

csomagformátum:



Időzítési diagram:

Adatküldés minden mintavétel után



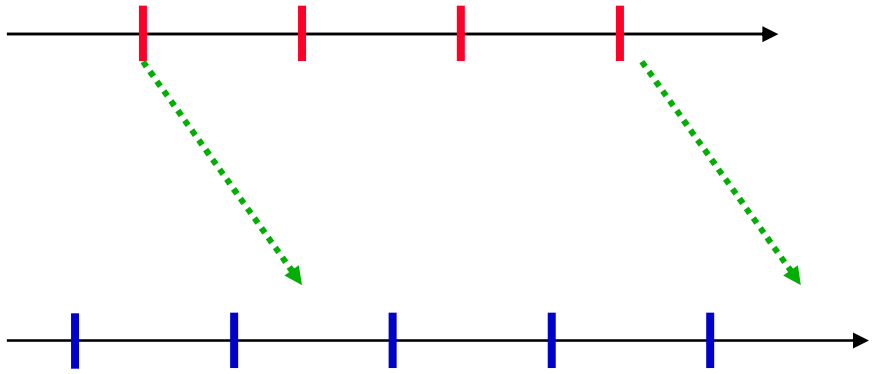
Adatátviteli módok
Egyedi vs. tömbös adatátvitel

- Egy csomagban több adat: egy byte hasznos adatra átlagosan kevesebb overhead jut.
- A kihasználtság nő.
- Példa: 30byte adat (1byte/adat) + (11+6)byte overhead esetén a hatásfok 30/47≈64%
- Probléma: növekvő késleltetés, nő a rendszer tehetetlensége és a beállási idő

csomagformátum:

Időzítési diagram:

Adatküldés nem kell minden mintavétel után

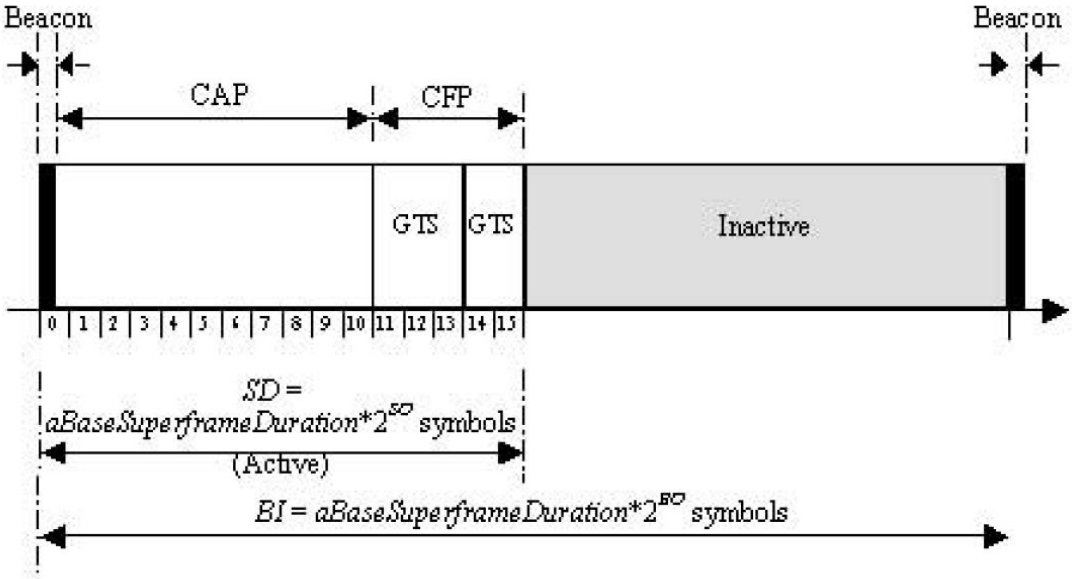


A hálózat adatátviteli sebességének tervezése

- A szenzorok mintavételi frekvenciája alapvetően meghatározza a megkövetelt adatsebességet.
- Példa:
  - 600Hz mintavételi frekvencia + előző példában adott paraméterek.
  - 10 db szenzor
  - 1db szenzor másodpercenként 20 (600/30) csomagot küld, egy csomag 47byte=376bit, összesen ez 7520bit/másodperc
  - 10 szenzor esetén 75.2kbps
- A mintavételi frekvencia meghatározása:
  - Mintavételi tétel: egy BW sávszélességű jelet legalább $f_s > 2BW$ frekvenciával kell mintavételezni
  - Szabályozástechnikai adatátviteli tétel:
    - Legyen adott egy rendszer állapotegyenlete: $\mathbf{x}' = \mathbf{Ax} + \mathbf{Bu}$
    - Az $\mathbf{A}$ mátrix legnagyobb sajátértéke legyen λ_{max}
    - A rendszer stabilizálásához szükséges legkisebb adatátviteli sebesség: $R > \log_2(\lambda_{max})$
    - Ennél mindenképpen nagyobb sebesség kell, ez egy abszolút alsó korlát
    - A rendszer legkisebb időállandójánál kb. 3-5-ször gyakoribb mintavételezés kell

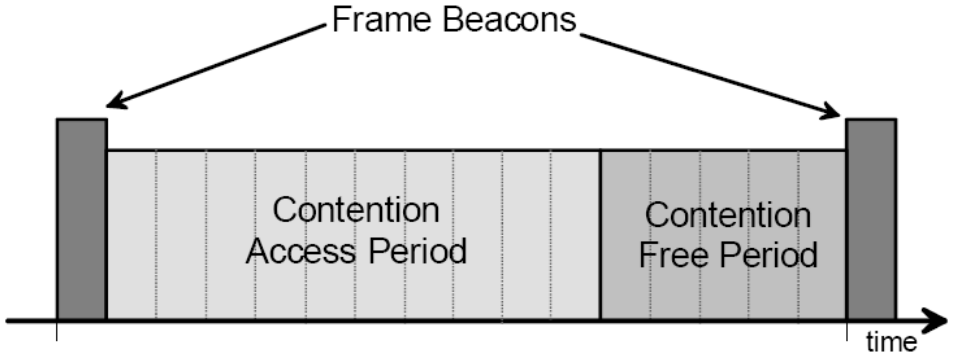
Valósidejű működés támogatása az IEEE 802.15.4. szabványban

- Mindenképpen keretezett (beacon based) adatátvitel kell a determinisztikusság miatt
- Egy keret: aktív és inaktív szakasz
- Időkritikus rendszerekben az inaktív szakaszt érdemes nullára állítani
- SO (Superframe Order) és BO (Beacon Order) a hálózat állítható paraméterei
- *aBaseSuperframeDuration* = *aBaseSlotDuration***aNumSuperframeSlots* = 60symbols*16 = 960symbols
- *aBaseSuperframeDuration*, *aBaseSlotDuration* és *aNumSuperframeSlots*: rendszer paraméterei



Valósidejű működés támogatása az IEEE 802.15.4. szabványban

- Az IEEE 802.15.4. szabvány bizonyos szinten támogatja a valósidejű működést.
- Csillag topológia ajánlott. A vezérlő a PAN coordinator.
- Az adatátvitel 16 db egyenlő időszeletre (slot) oszlik két beacon üzenet között.
- A keret első felében (Contention Access Period: CAP) versengés folyik a csatornáért: CSMA
- Második szakasz: Contention Free Period (CFP): nincsen versengés a csatornáért
- GTS: Guaranteed Time Slot: a CFP-ben található, fixen kiosztott időszeletek.
- Maximum 7 db GTS áll rendelkezésre
- A GTS-eket kérni kell a koordinátortól. Ez a folyamat hosszabb időt is igénybe vehet: nem érdemes dinamikusan kezelni, mert a kérést a CAP-ban kell megtenni, melynek nem determinisztikus a működése.



Valósidejű működés támogatása az IEEE 802.15.4. szabványban

- Példák: 2.4GHz, 250kbps →1symbol=4bit, tehát 1byte=2symbols (lásd a szabvány tárgyalásánál)
- SO=BO=0 → BI=SD= *aBaseSuperframeDuration**2^0=960symbols=480byte
  - A hálózatban 7 darab node van, mindegyik egy GTS slot-ot kap
  - 480byte elküldése 250kbps-on T_F=15.36ms:
    - T_F egy keret hossza → ekkora az adatátviteli késleltetés
    - Egy-egy szenzor T_F időközönként küldi az adatait
  - Egy slot 30byte(*aBaseSlotDuration*=60symbols), ebből legyen 20byte overhead→10 byte hasznos adat
  - Egy slot hossza T_{slot}=(30*8)bit/250kbps=0.96ms
  - Ha egy minta 1 byte, ez azt jelenti, hogy egy csomagban egy szenzor N=10 mintát tud elküldeni.
  - A csomagokat T_F időközönként küldik, tehát a mintavételi időköz: $T_s=T_F/N$=1.536ms, tehát a max. mintavételi frekvencia $f_s=1/T_s$=651Hz
  - A kihasználtság növelhető a frame méretének növelésével, mert így az overhead relatív aránya csökken.

Példa folytatás