Beágyazott információs rendszerek

Valós idejű kommunikáció

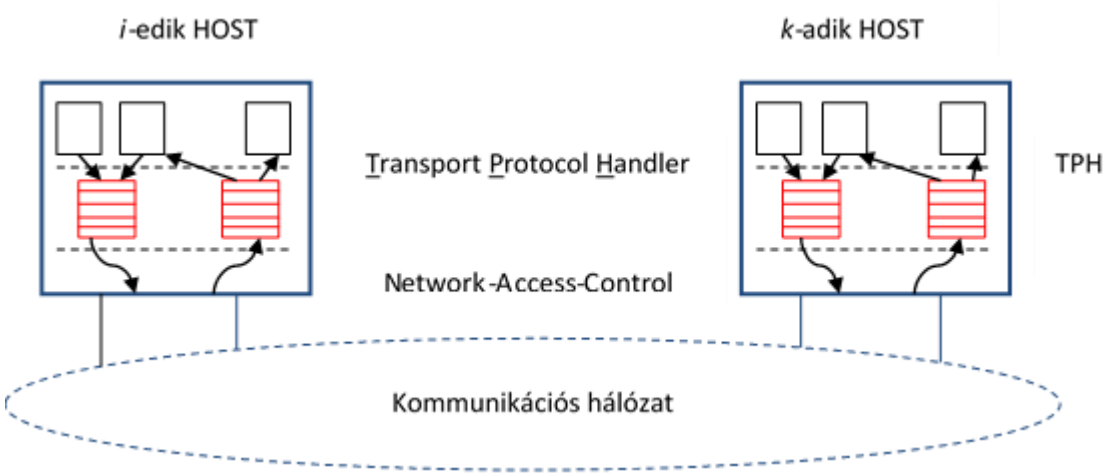
2020. november 5.

6. Valós idejű kommunikáció

Emlékeztető

Általában bonyolul mechanizmusok, várólisták jellemzőek. Valós idejű követelmények nehezen teljesíthetőek.

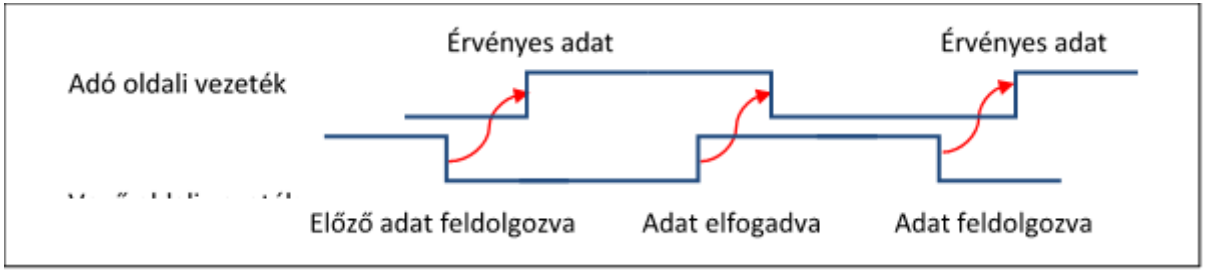
Az általános séma:



Az időviszonyok kritikus volta a fizikai szinten is jól azonosítható.

Aszinkron kommunikáció esetén szinkronizálás kell, ez a “handshaking”.

A kétvezetékes handshake:



A kommunikáció sebesség- és időviszonyait az adó és a vevő sebessége és egyéb feladatai együtt határozzák meg, hiszen az adat vevő oldali “feldolgozásáig” újabb adat továbbításában az adó nem gondolkodhat.

Követelmények:

1. Lehetőleg kis protokoll késleltetés (protocol latency) és jitter (latency jitter). (Latency árnyaltabb jelentése: lappangás, homály, elrejtettség).
2. Komponálhatóság: segíteni kell az időbeni követelmények teljesülését: HOST ↔ CNI (Communication Network Interface), időszakos tűzfal szerep, HOST önálló tesztelhetősége.
3. Flexibilitás: gépkocsi funkciók időbeni működése extrákkal, extrák nélkül …

Emlékeztető

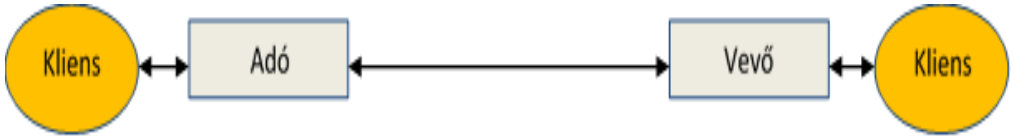
4. Hibadetektálás: Jósolható és hibatűrő kommunikáció kell. End-to-end nyugtázás [illegible] zárjon automatikusan, ha az állítását lehetővé tevő vezeték elszakad, de erről menjen értesítés a központnak.
5. Struktúra: pont-pont kapcsolat kezelhetetlen bonyolultságú kábelezéssel jár, helyette busz és gyűrű.

Az adatáramlás szabályozása (flow control):

Explicit forgalomszabályozás:

Példa: PAR (Positive Acknowledgement or Retransmission) protokollok:

Több változat van, de ezek közösek az alábbiakban:
(1) Az adóoldali kliens kezdeményez.
(2) A vevő jogosult késleltetni.
(3) A hibát az adó detektálja.
(4) Hibajavítás időbeni redundanciával.



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.

Emlékeztető

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.)

Megjegyzés:

Az adó oldalon a vétel visszaigazolása és a vevőoldalon az adat elfogadás időpontja között jelentős eltérés lehet.

Példa: Token vezérelt buszon az üzenettovábbítás ideje $\mathbf{1}\ ms$, a token körüljárás ideje $\mathbf{10}\ ms$.

A beállítandó time-out: $10 + 1 + 10 + 1 = \mathbf{22}\ ms$, hiszen worst-case esetben, ha éppen elment a token $10\ ms$-ot kell várni, erre jön az üzenettovábbítás $1\ ms$-a, majd a visszaigazoláskor ugyanez ismétlődhet.

A $d_{min} = 1\ ms$, a $d_{max} = (ismétlések\ száma) * timeout + 10\ ms + 1\ ms$.

Ha kétszer ismétlünk (azaz háromszor próbálkozunk), akkor d_{max}= **55** *ms*.

Ezekkel néhány jellemző a következőképpen alakul:

- **jitter**= $d_{max} - d_{min} = 54\ ms$.
- **akció késleltetés**, ha van globális óra: $d_{max} = 55\ ms$.
- **akció késleltetés**, ha nincs globális óra: $2 * d_{max} - d_{min} = 109\ ms$.
- **a hibadetektálás késleltetése:** 3*time-out: 66 *ms*.

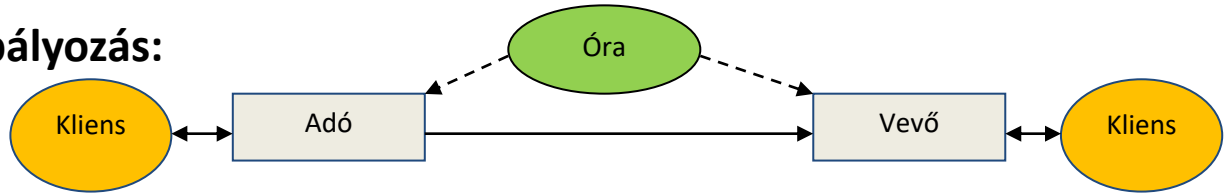
A PAR protokoll és a számpélda azt illusztrálja, hogy az ún. **explicit forgalomszabályozás** valós idejű alkalmazásokban **kedvezőtlen lehet** a nagymértékű jitter, akció késleltetés és hibadetektálás késleltetés miatt.

Emlékeztető

A PAR protokollnak sokféle változata van, de mindegyik a következőkön alapszik:

(1) A kommunikációt az az adóoldali kliens kezdeményezi.
(2) A vevő jogosult késleltetni az adót a kétirányú kommunikációs közegen keresztül.
(3) A kommunikáció hibáját az adó detektálja, nem a vevő.
A vevő nem kap arra vonatkozóan információt, hogy mikor történt a hiba.
(4) A hiba javítására időredundanciát használnak, amely növeli a protokoll késleltetést.

Implicit forgalomszabályozás:



A kommunikáció **idővezérelt.** Az adó és a vevő is rendelkezik egy tervezési időben elkészült időrendi táblázattal (“vasúti menetrend”). Ebben egyértelmű az adás és egyidejűleg a vétel időpontja/időintervalluma. Az adó az óraütés vezérlésére „kitolja” az üzenetet, a vevő pedig “behúzza” (push-pull jellegű működés). Ez a logika sok esetben **jobban illeszkedik a valós idejű követelményekhez!** A hibadetektálás például a vevő által azonnal lehetővé válik, ha a várt adat nem érkezik meg. (Az adó részéről ez egy ún**. fail-silent** üzemmódban létet jelent, azaz hibás állapotát azzal jelzi, hogy **nem küld üzenetet**.) **Globális időalap kell!**

Az adó az „óraütés” kezdeményezésére csak meghatározott időpontokban ad, **nincs handshake**.
A hibadetektálás a vevő dolga: tudja, hogy mikor kell/kellett volna üzenetnek érkeznie.
A hibatűrés **aktív redundanciával** valósul meg: *k* fizikai üzenet kópia, **ha legalább egy megérkezik**, addig sikeres.
A csatorna **egyirányú,** ami többszereplős esetben előnyös.

Emlékeztető

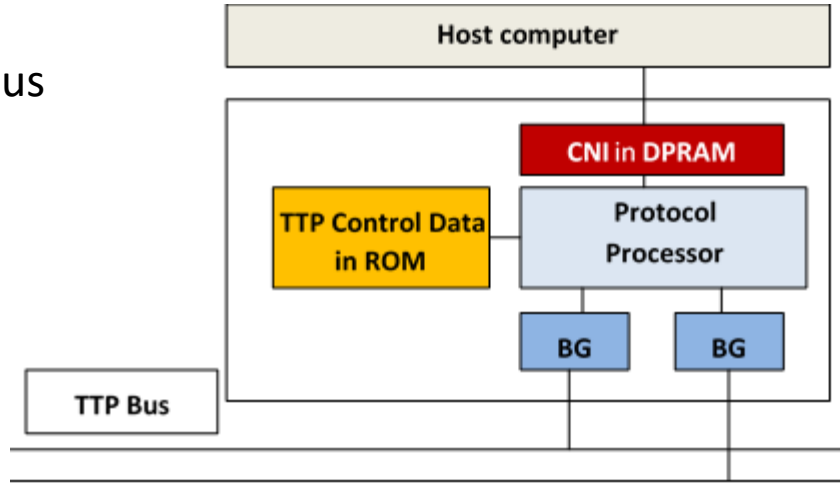
Az idővezérelt architektúra (Time Triggered Architecture, TTA) és az idővezérel
(Time-Triggered Protocols, TTP) Hard real-time (HRT) rendszerek implementálására szolgál.

Két változata van: **a TTP/C**, amely hibatűrő HRT rendszerekhez készült, és a **TTP/A**, amely olcsó ipari alkalmazások esetén jön számításba (pl. terepi busz (field bus) alkalmazásokban).

- A rendszer hibatűrő egységekből (FTU: **Fault Tolerant Unit**) felépülő **fürt** (cluster).
- Minden FTU cluster egy, kettő, vagy több csomópontból áll, amelyeket a kommunikációs hálózat köt össze.
- Minden csomópont két részrendszerből, a **host számítógépből** és a **kommunikációs vezérlőből** (**CC**) áll.
- A kommunikációs hálózat interfész (**CNI**) a csomóponton belüli interfész a host és a kommunikációs vezérlő (**CC**) között. A **CNI** egy dual-portos RAM memória (DPRAM). Az adat integritást a **Non-Blocking Write** (**NBW**) Protocol biztosítja (lásd később).

A kommunikációs vezérlő lokális memóriája tartalmazza az üzeneteket leíró listát (**Message Description List: MEDL**), amely meghatározza, hogy mely időpontban küldhet a csomópont üzenetet, ill. mely időpontban várhat más csomópontból. A MEDL méretét a fürt-ciklus mérete határozza meg.

- A TTP vezérlő független hardverként ún. **Bus Guardian egységeket** is tartalmaz, amelyek figyelik a vezérlő busz-hozzáférési mintáit, és leállítják a vezérlő működését, ha a szabályos hozzáférési minták időzítése megsérül.

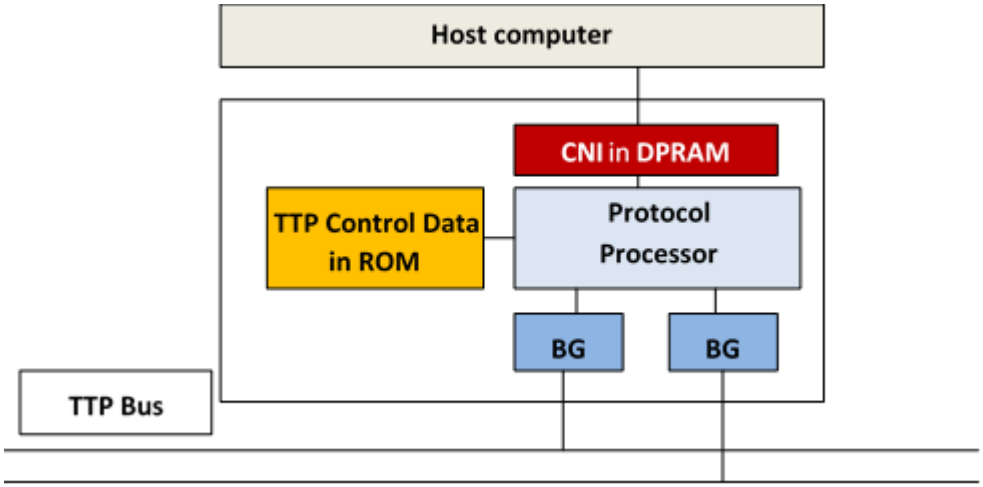


Emlékeztető

Fontos tulajdonságok:

(**1**) A **TTP** egy időosztásos-többszörös-hozzáférésű (**TDMA**) protokoll.

(**2**) A komponálhatóságot szolgálja, hogy a kommunikációs vezérlő **autonóm**, amelyet a **MEDL** és **a globális óra** vezérel.



A host számítógépek hibája nem tudja befolyásolni a kommunikációs rendszert, mert vezérlő jel nem megy át a **CNI**-n és a **MEDL** nem férhető hozzá a **host** felől.

(**3**) A kommunikáció módja **tervezési időben dől el** (olyan, mint a vasúti menetrend), mindenki előre tudja, hogy mikor kap, ill. mikor küld üzenetet. Ha hiányzik/elmarad az üzenet, akkor **azonnal detektálható** a hiba.

(**4**) Az üzenet azonosítása (**naming**): az **üzenet** és **küldőjének neve** nem kell, hogy része legyen az üzenetnek, a MEDL-ből kinyerhető. Ugyanannak az RT változónak **más és más nevet** adhatunk az egyes host-ok szoftverében.

(**5**) **Visszaigazolás:** előzetesen tudjuk, minden helyesen működő vevő veszi a helyesen működő adó üzenetét. Amint egy vevő visszaigazol egy üzenetet, arra lehet következtetni, hogy az üzenet kiküldése helyesen történt és azt minden helyesen működő vevő megkapta.

(**6**) **Hiba esetén** "hallgatás" az időtartományban: a TTP feltételezi, hogy a csomópontok támogatják a "**fail silence**" absztrakciót az időtartományban, ami azt jelenti, hogy egy csomópont vagy küld üzenetet a helyes időpontban, vagy nem küld semmit. A csomópontnak ezt a tulajdonságát a TTP vezérlőn belül a **Bus Guardian** (**BG**) valósítja meg.

Az amplitúdó tartományban a hibakezelés a host felelőssége, a TTP csak CRC-t bizt

Emlékeztető

A CNI felépítése:

A CNI az idővezérelt architektúra **legfontosabb interfésze**, mert ez az egyetlen interfész, amely a **host szoftvere által látható**.

A **Status Register**-eket a TTP vezérlő írja, a **Control Register**-eket pedig a host.

Status Registers	Control Registers
(S1) Global Internal Time	(C1) Watchdog
(S2) Node Time	(C2) Timeout Register
(S3) Message Description List	(C3) Mode Change Request
(S4) Membership	(C4) Reconfiguration Request
(S5) Status Information	(C5) External Rate Correction

S1: A fürt közös órája két bájton.
S2: a vezérlő saját órája.
S3: **MEDL** Pointer.
S4: annyi bitből áll, ahány csomópont van a fürtben. Ha egy bit "**TRUE**", akkor működött az illető csomópont a legutolsó kommunikációs időszeletben, ha "**FALSE**", akkor nem működött.

C1: A host periodikusan frissíti, a vezérlő ellenőrzi. Ha elmarad a frissítés, akkor a vezérlő – hibát sejtve – leállítja az üzenetküldést.
C2: A host írja, lejártakor megszakítást okoz. Például a host a fürt órájához szinkronizálhatja magát egy előírt későbbi időben.
C3: Például új ütemezésre lehet áttérni ennek segítségével.
C4: Meghibásodás esetén szerepcsere kezdeményezhető.
C5: külső óra szinkronizálást (pl. GPS) tesz lehetővé.

Emlékeztető

A Message Description List (MEDL) felépítése

Node Time	Address	D	L	I	A
Mikor	*Mit: Az üzenet címe*	*irány*	*hossz*		

I: azt adja meg, hogy inicializálással kapcsolatos üzenet, vagy normál üzenet.

A: egy további paraméter-mező, amely a változtatásokkal (mode changes) kapcsolatos információkat tartalmaz.

A hibatűrő egységek (Fault-Tolerant Units) rendeltetése egy csomópont hibájának a maszkolása.

Ha a csomópont a **fail-silent** absztrakciót valósítja meg, akkor a **csomópontok duplikálása** elegendő **egyszeres csomópont-hiba tolerálására**.

Ha a csomópont nem valósítja meg a **fail-silent** absztrakciót, de lehet **érték-hibája** a CNI-nél, akkor **háromszoros moduláris redundancia** (TMR: triple modular redundancy) valósítandó meg. Ez „**háromból kettő**” szavazással maszkolja az érték-hibát.

Ha a csomópont hiba esetén fellépő viselkedéséről **semmit sem tudunk**, azaz akár **bizánci típusú hiba** is felléphet, akkor **négy csomópont** tudja maszkolni a hibát.

A protokoll tervezés alapvető konfliktusai

A kiegyensúlyozott protokoll tervezés törekszik számos szempont összeegyeztetésére. Vannak azonban olyanok, amelyek nem egyeztethetők össze.

Néhány példa buszon történő kommunikáció esetében:

Konfliktus: Külső vezérlés ↔ komponálhatóság

Egy valós idejű elosztott rendszerben minden csomóponthoz tartozik egy **host számítógép**, aminek van egy **kommunikációs hálózat interfésze** (Communication Network Interface: CNI).

Az időtartománybeli komponálhatóság megkívánja, hogy:

- a CNI teljesen specifikált legyen az időtartományban;
- a rendszerbe további csomópontok integrálása semmilyen változást nem idéz elő az egyes CNI-k időbeni tulajdonságait illetően,
- minden host időbeni tulajdonságai a CNI-től függetlenül tesztelhetők.

Ha az időbeni tulajdonságok nincsenek benne a CNI specifikációjában, akkor, például azért, mert az üzenetküldés időpontja *külső* és *ismeretlen* információ a kommunikációs rendszer számára, nem lehetséges a komponálhatóság biztosítása az időtartományban.

Az **eseményvezérelt rendszerek** esetében az időbeni vezérlés jelei **külső forrásból** származnak, ezért a csomópontok host számítógépeiben az **alacsonyszintű komponálhatóság nem biztosítható.**

Példa: Ha a csomópontok mindegyike bármikor versenyezhet az egyetlen kommunikációs csatorna birtoklásáért, akkor képtelenség elkerülni az **ütközések következtében** fellépő **átviteli késleltetést** bármilyen okos közeg-hozzáférési protokollt alkalmazunk is. A járulékos átviteli/kommunikációs késleltetések érvényteleníthetik a valós idejű képek időbeni pontosságát.

Konfliktus: Flexibilitás ↔ hibadetektálás

A flexibilitás azt jelenti, hogy a csomópont viselkedése **nincs előzetesen korlátozva**.

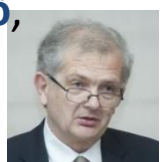
Replikátum nélküli architektúrában **a hiba detektálása csak akkor lehetséges**, ha az aktuális viselkedés összevethető a várt viselkedésre vonatkozó előzetes (a priori) ismerettel.
Ha ilyen ismeret nem áll rendelkezésre, akkor a hálózat nem védhető meg a hibás csomópontjától.

Példa: Ha egy eseményvezérelt rendszerben periodikus működést nem tételezhetünk fel, **ha nincs az üzenetküldés gyakoriságának valamilyen korlátja,** akkor nem kerülhető el, hogy egy (esetlegesen hibás) csomópont ne sajátítsa ki a hálózatot.

Példa: Ha egy csomópontot nem kényszerítünk arra, hogy rendszeres időközönként "**szívdobbanás**" jellegű üzeneteket küldjön, akkor nem lehetséges a csomópont hibáját detektálni (valamilyen korlátozott késleltetéssel).

Konfliktus: Sporadikus adat ↔ periodikus adat

Egy valós idejű protokoll lehet hatékony **periodikus adatra** és **sporadikus adatra**, de egyidejűleg mindkettőre nem. Periodikus adat továbbítása (Pl. szabályozási hurkok koordinálására használt adatok esetében) minimális kommunikációs késleltetéssel (latency jitter) kell, hogy megtörténjen. Mivel a periodikus adat továbbításának ideje **előzetesen ismert**, ezért **konfliktusmentes** ütemezések készíthetők off-line.
A **sporadikus adatokat** kérésre, és minimális késleltetéssel kell továbbítani.
Ha a külső kérés időpontja egybeesik a periodikus adat továbbítási időpontjával, akkor el kell dönteni, hogy melyiket késleltetjük. Bármelyik esetben a **kommunikációs késleltetés nő**, azaz mindkét cél egyidejű kielégítése **nem lehetséges**.



Konfliktus: Egy pontról történő vezérlés ↔ hibatűrés

Minden protokoll, amelyet **egy pontról** vezérlünk, **egy meghibásodási ponttal** rendelkezik. Ez egyértelmű egy központi master-en alapuló kommunikáció protokoll esetében.

De ilyen módon viselkedik minden időpillanatban egy **token-passing rendszer** is: ha token-t birtokló csomópont **elromlik**, akkor nincs tovább kommunikáció mindaddig, amíg nem detektáljuk a token elvesztését járulékos time-out mechanizmussal, és **helyre nem állítjuk** a token-t. Ez időt vesz igénybe, és megszakítja a valós-idejű kommunikációt.

Bizonyos értelemben a token helyreállítás nemtriviális problémája kapcsolódik a központi master átkapcsolása egy standby masterre problémához egy multi-master protokollban.

Konfliktus: Valószínűségi hozzáférés ↔ a replikátum determinizmusa

Ha közeg-hozzáférés valószínűségi mechanizmusokkal (pl. a konfliktus feloldás véletlen számok alkalmazásával) történik, akkor aktív redundancia igénye esetén nem garantálható, hogy a replikátum a versengő csomópontok közül ugyanazt hozza ki győztesnek.
A replikátum determinizmusa nélkül a replikátum eltérő helyes eredményre juthat, ez azonban a rendszer egészében inkonzisztenciát eredményez.

Teljesítőképesség határok TT rendszerekben:

Tegyük fel, hogy az egy-egy üzenet továbbítására szánt keretek 20 µs időtartamúak, és 80%-os a sávszélesség kihasználás, tehát 5 µs az ún. inter-frame-gap. Ez a 25 µs gyakoriság 40 000 üzenet/sec üzenettovábbítási sebességet tesz lehetővé.
Ha 10 csomópontot foglal magába a fürt(klaszter), akkor ez csomópontonként 4kHz-es mintavételi frekvenciát jelent. Természetesen a 20 µs alatt átvihető adatmennyiség a sávszélesség függvénye.

Példa: 5Mbit/s sávszélesség esetén $5*10^6*20*10^{-6}$ = 100 bit (~12 byte) vihető át.

Példa: 1Gbit/s sávszélesség esetén $1*10^9*20*10^{-6}$ = 20 000 bit (2500 byte) vihető át.

Szinkronizáció ET és TT rendszerek között

A csomópont host gépe eseményvezérelt (ET) módon működik, a hálózat pedig idővezérelt (TT). Ez utóbbi azt jelenti, hogy a hálózati kommunikáció interfésze (CNI) nem blokkolható következmények nélkül. A hálózat felőli írást vizsgáljuk.

NBW: Non-blocking Write Protocol:

Egy író (CNI), több olvasó (a host taskjai) van a rendszerben.

A CNI **blokkolás nélkül** (hand-shake nélkül) átírja a DPRAM tartalmát, ami azonban egybeeshet egy olvasással → **inkonzisztencia** léphet fel.

Ha az olvasó észleli az interferenciát, akkor megpróbálja újra mindaddig, amíg meg nem kapja a konzisztens verziót. Az olvasási próbálkozások száma korlátos kell legyen.

A protokollnak szüksége van egy együttfutó vezérlő mezőre (**Concurrency Control Field: CCF**), amelyhez a hozzáférést kölcsönösen kizáró módon, hardverrel kell garantálni.

Ezt nullára kell inicializálni és az író által inkrementálni az írás megkezdése előtt, majd a végén is. Az olvasó az olvasás művelet előtt olvassa a CCF-et és ha páratlan, akkor azonnal ismétel, ha nem, akkor az olvasás végén ellenőrzi, hogy változott-e a CCF, azaz történt-e írás időközben, ha igen, akkor újból próbálkozik.

Inicializálás: CCF:=0

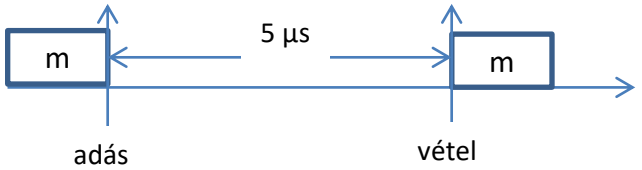
Írás:

```
Start:   CCF_old:=CCF;
         CCF:=CCF_old+1;
         <írás>
         CCF:=CCF_old+2;
```

Olvasás:

```
Start:   CCF_begin:=CCF;
         if CCF_begin=odd then goto Start;
         <olvasás>
         CCF_end:=CCF;
         if CCF_end ≠ CCF_begin then goto Start;
```

Az olvasási próbálkozások száma korlátos, ha az írások közötti idő lényegesen nagyobb, mint maga az írás, vagy az olvasás.



Kommunikációs közeg jellemzők:

A csatorna jellemzők:

(1) sávszélesség 10kbit-sec -> 10 Mbit/sec pl. autóban, vezetéken, Gbit/sec üvegszálon.

(2) terjedési sebesség/késleltetés: 300 000 km/sec, 1 láb/nsec. Kábelben ennek 2/3-a. Pl.: 5 µsec kell 1km megtételéhez.

(3) csatorna bit hosszúsága: azon bitek száma, amelyek a terjedési késleltetés alatt átérnek. Pl. 100 Mbit/s sávszélesség mellett 200 m hosszú kábelen a bit hosszúság 100 bit, mert a terjedési késleltetés ezen a hosszon 1 µsec.

(4) adat hatékonyság: buszon történő kommunikáció esetén ki kell várni minimálisan egy terjedési késleltetésnyi időt ahhoz, hogy az adó újból adhasson (a buszon lévő tartalmat kimerevítjük addig, amíg a vevő nem olvassa be onnan az adatot).

Az adat hatékonyság < $m/(m+bl)$, ahol m az üzenethossz, bl pedig a csatorna bithosszúsága.

Pl. 1 km hosszú buszon, 100 Mbit/s sávszélesség mellett bl = 500, ha az üzenet 100 bit, akkor az adat hatékonyság: 100/(100+500) = 16.6%.

Megjegyzések a kommunikáció témaköréhez:

1. A fizikai réteg szintjén a kommunikáció:

(1) aszinkron: ha szinkronizációra alkalmas jelszint-átmenet csak az üzenet elején van. Pl. az UART (Universal Asznchronous Receiver Transmitter) ilyen: általában rövid üzenet, pl. 10bit, így olcsó oszcillátor is elég (pl. 10^{-2}sec/sec).
(2) szinkron: menet közben is szinkronizál, mert vannak erre használható szintátmenetei.
Megjegyzés: Figyeljük meg, hogy az aszinkron és szinkron itt mást jelent, mint általában!

Példák:

NRZ kód (non-return-to-zero)***:*** nem szinkronizáló

11010001: "1" magas szint, "0" alacsony szint
Manchester kód: szinkronizáló

11010001: "1" felfutó él, "0" lefutó él két órajel között "félúton": az éppen következő bit dönti el, hogy az "órajel" idejében vissza kell-e futni a másik szintre, vagy sem. Hátránya, hogy 1/2 bit-cellával jellemezhető: kétszer is változhat a jel, míg más kódokban legfeljebb egyszer.

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.

Időszinkronizáció vezetéknélküli hálózatokban

A szinkronizáció osztályai/fajtái:

- *Külső, belső* (pontosság, együttfutás);
- *Időtartam szerint*: (1) folytonos (mindig fennáll), (2) kérésre (on demand);
- *Eseményvezérelt:* az időbélyeg előállításához csak akkor kell szinkronizált idő, amikor az esemény bekövetkezett (Post-facto synchronization);
- *Idővezérelt:* akkor használjuk, amikor több szenzor adatára van szükségünk egy adott időpontra;
  (1) közvetlen (immediate): Vegyen mintát azonnal, és rendeljen hozzá időbélyeget;
  (2) Előre megadott (anticipated) időpontbeli mintavétel, amit akkor használunk, ha az adattovábbítás párhuzamos megvalósítása nehézségekbe ütközik, vagy ha több ugrás (hop) szükséges. (A megfelelő minőségű szinkronizált állapotot elérése csak az előre megadott időpontra esedékes.) (Pre-facto synchronization);
- *Érintettség* (Scope): minden csomópont vagy csak egy részhalmaz:
  (1) hálózati topológia szerinti;
  (2) időbeni érintettség.
- *Ütem, ofszet szinkronizáció*: (1) Ütem: a csomópontok azonos időintervallum-hosszúságot mérnek (pl. egy objektum feltűnésének és eltűnésének időpontjai között eltelt idő; (2) Ofszet: minden érintett csomópont azonos időt mutat.
- *Időskála szinkronizáció*: helyi idő transzformálása egy másik csomópont helyi idejébe.
- *Óra szinkrozináció*: ütem és ofszet szinkronizáció vagy folyamatos szinkronizáció.
- *Időpillanatok szerint*: idő+bizonytalanság (valószínűség eloszlás): $t = 5$ + bizonytalanság
- *Időintervallumok szerint*: $t \in [4.5, 5.5]$, nagyon jó, ha garantálni lehet.