Óraszinkronizáció szenzorhálózatokban (TPSN és RBS algoritmusok)

Összeállította:

Orosz György

BME-MIT

- Referenciák:

[1] K. Kömer, P. Blum, L. Meier, “Time Synchronization and Calibration in Wireless Sensor Networks,” in *Handbook of Sensor Networks: Algorithms and Architectures*, Ivan Stojmenović, Ed., Wiley, 2005.

[2] D. L. Mills, “Internet Time Synchronization: The Network Time Protocol,” *IEEE Transactions on Communications*, COM 39, No. 10, Oct. 1991., pp. 1482–1493.

[3] S. Ganeriwal, R. Kumar, M. B. Srivastava, “Timing-sync Protocol for Sensor Networks,” *First ACM Conference on Embedded Networked Sensor Systems (SenSys)*,Los Angeles, California, USA, Nov. 2003.

[4] J. Elson, L. Girod, D. Estrin, “Fine-grained Network Time Synchronization Using Reference Broadcasts,” *Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002)*, Boston, MA. Dec. 2002.

[5] M. Maroti, B. Kusy, Gy. Simon, A. Ledeczi, “The Flooding Time Synchronization Protocol,” Technical Report ISIS-04-501, Institute for 96 Software Integrated Systems, Vanderbilt University, Nashville Tennessee, 2004.

[6] K. Römer, “Time Synchronization in Ad-Hoc Networks,” *ACM Symposium on Mobile Ad-Hoc Networking and Computing (MobiHoc 01)*, Oct. 2001.

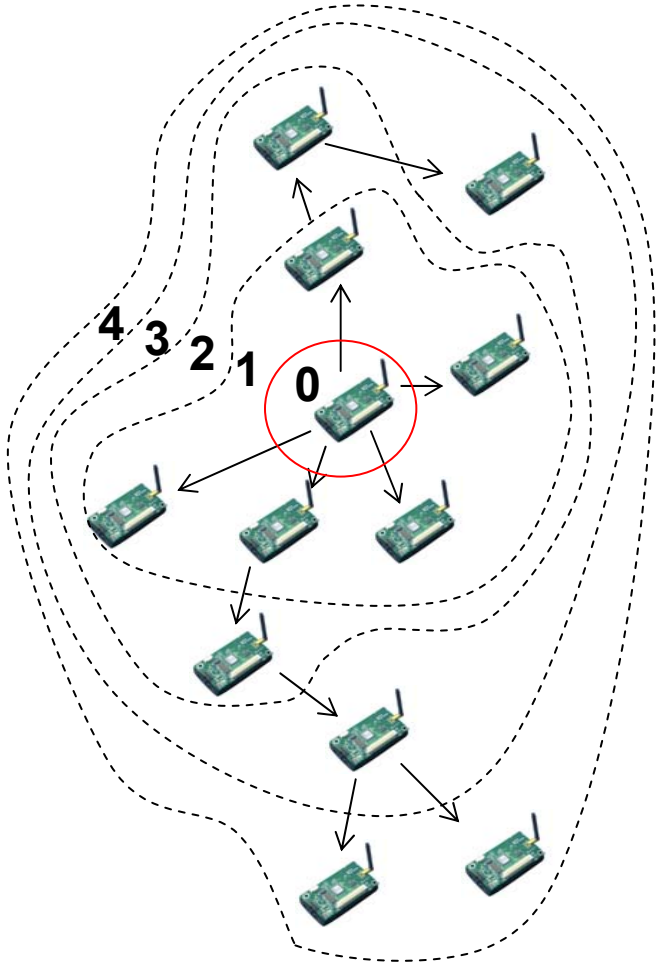
Timing-sync Protocol for Sensor Networks (TPSN) [3]

Általános tulajdonságok

- Mérési eredmények Mica mote:
  - kb. $20\mu\text{sec}$ átlagos pontosság
  - kb. $50\mu\text{sec}$ worst case pontosság
- Fa struktúrájú szinkronizációs algoritmus
- Egy referencia mote-hoz szinkronizálódik a hálózat (root)
- Két szakasz
  - Faépítés: hálózat struktúrájának felépítése
    - Fel van készítve a struktúra javítására is
    - Egyben adattovábbító hálózati felépítésként is használható
  - Páronkénti szinkronizáció: szülő → gyerek irány (küldő fogadó szinkronizáció) az élek mentén
- Post-facto szinkronizálás is lehetséges

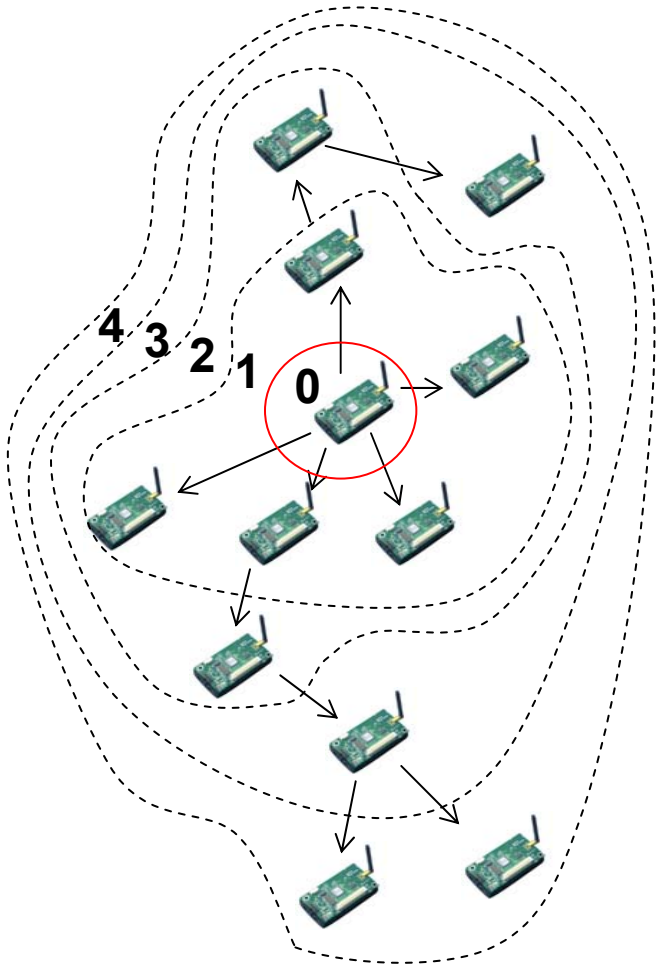
Kommunikációs struktúra

- Feltételezések:
  - Az i-edik szint kommunikálni tud az $(i-1)$-edik szinttel
  - Egyedi azonosítók szükségesek
  - Mindenkinek van min. 1 szomszédja (egyébként nem lenne része a hálózatnak)
  - Szimmetrikus kommunikáció
- Ajánlott faépítési algoritmus:
  - A root **`level_discovery`** üzenetet küld
  - Aki ezt megkapja, az 1. szintre sorolja magát és **`level_discovery`** üzenetet küld
  - Rekurzívan folytatódik a szintépítés
  - Aki egy adott szinthez tartozik, nem fogad több szintépítő üzenetet
  - Más faépítési algoritmus is használható nem biztos, hogy ez az optimális
- Szinkronizáció az élek mentén
  - Páronkénti szinkronizáció: kizárja a broadcast lehetőségét



Kommunikációs struktúra

- Problémák a hálózati felépítésben
  - Újonnan bekapcsolt node
  - Kieső node-ok: elvész a kapcsolat a roottal
- Újonnan bekapcsolt node:
  - N_i **`time-out`** ideig nincs szinthez rendelve: **`level_request`** üzenetet küld
  - Erre válaszolnak a szomszédok: elküldik a saját szintjüket
  - N_i a legalacsonyabb szinthez csatlakozik
- Kieső node (i-edik szinten)
  - (i+1)-edik szinten nem kapnak szinkronizációs üzentet
  - **`level_request`** üzenet küldése
  - Root esik ki:
    - Az 1. szinten valaki rootnak jelöli magát (algoritmus nincs definiálva)
    - Új faépítés indul



Szinkronizáció

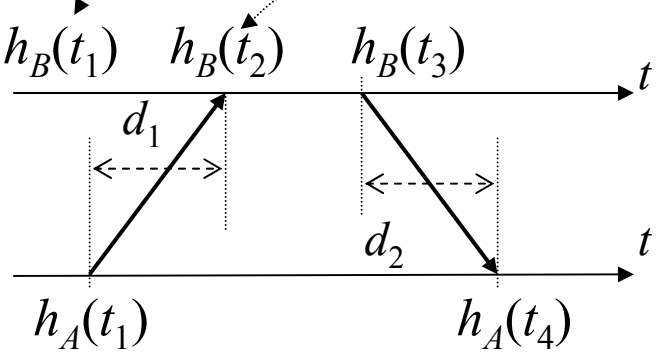
- Szinkronizációs idődiagram jelölései:
  - t_1, t_2 … : globális időpillanatok
  - $\Delta=O_{j,i}(t)=O_{B,A}(t)=O(t)= h_B(t)–h_A(t)$: ofszet, tehát a B és A órái közötti különbség példa: $h_B(t_1)=h_A(t_1)+\Delta$ illetve $h_B(t_4)=h_A(t_4)+\Delta$
  - d: üzenetküldés időtartama
- Az $(i+1)$-edik szint az i-edik szinthez szinkronizálódik: A node szinkronizációt kezdeményez ($t=t_1$ -kor). B node elküldi $h_B(t_2)$-t és $h_B(t_3)$-t, ebből számítható az ofszet.
- Alapfeltevések
  - Szimmetrikus csatorna: $d=d_1=d_2$ (hasonló eszközök és csatornaviszonyok)
  - Gyors üzenetváltás: $\Delta=O_{i,j}(t)=\text{const.}$ (rövid idő az üzenetváltás között→offszet=állandó)
- Δ meghatározása után az órák korrigálhatóak (fizikai vagy időbélyeg konverzió)

master

$B=N_i$

slave

$A=N_j$



$$h_B(t_2) = h_A(t_1) +\Delta+d \rightarrow d+\Delta = h_B(t_2)–h_A(t_1)$$

$$h_A(t_4)+\Delta = h_B(t_3) +d \rightarrow d–\Delta = h_A(t_4)–h_B(t_3)$$

ebből:

$$d = ½ \{[h_B(t_2)–h_A(t_1)]+[h_A(t_4)–h_B(t_3)]\}$$

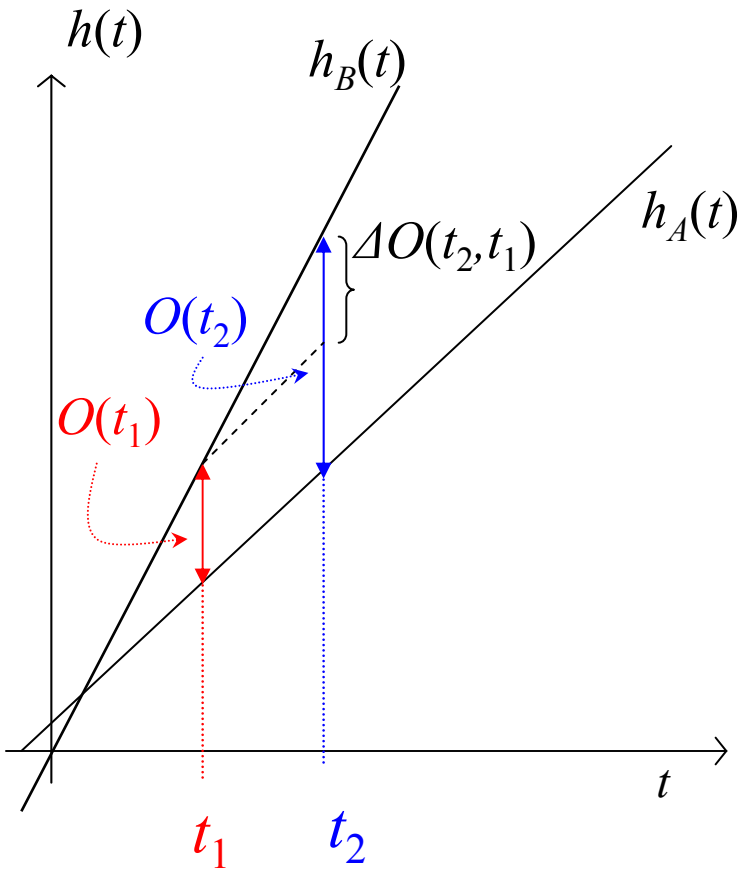
$$\Delta = ½ \{[h_B(t_2)–h_A(t_1)]–[h_A(t_4)–h_B(t_3)]\}$$

Szinkronizáció kommunikációs protokollja

- Root kezdeményez **`time_sync`** csomag küldésével
- Az 1. szinten véletlen idő múlva szinkronizációs üzenetcserét indítanak a root-tal
  - Véletlen idő: ütközés elkerülése
- Az üzenetcsere során szinkronizációs pontokat gyűjtenek a Δ ofszet meghatározásával
- A 2. szinten érzékelik az üzenetcserét, mivel kapcsolatban vannak az 1. szinttel
- A 2. szint ezt követően (véletlen idő beiktatása után) szinkronizációt kezdeményez az 1. szinttel
- A szinkronizációs folyamat a fent leírtak alapján folytatódik

Időalapok jellemzése (TPSN)
Levezetés: kiegészítő anyag

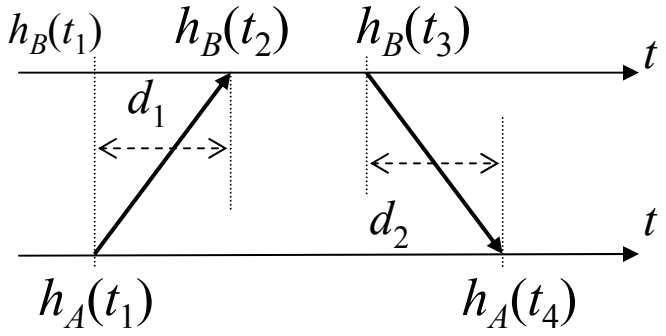
- Idővariáns ofszet
  - $O(t_1)$: ofszet a t_1 időpontban
  - $O(t_2)$: ofszet a t_2 időpontban
  - $\Delta O(t_2,t_1)$: ofszet változása a t_1 és t_2 időpontok között
    $O(t_1)+\Delta O(t_2,t_1)=O(t_2)$
  - $O(t_3)$: ofszet a t_3 időpontban
  - $O(t_4)$: ofszet a t_4 időpontban
  - $\Delta O(t_4,t_3)$: ofszet változása a t_3 és t_4 időpontok között
    $O(t_3)+\Delta O(t_4,t_3)=O(t_4)$
  - Probléma: Δ helyett $O(t_4)$-el kellene számolni, mert az az igazi ofszet



TPSN szinkronizáció hibaanalízise

Levezetés: kiegészítő anyag

- Algoritmusban figyelmen kívül hagyott paraméterek:
  - Drift és bizonytalanság a kommunikációs időkben
- $d_1 = S_A + P_{AB} + R_B$
  - d_1: üzenet küldés időtartama A-tól B-ig
  - S_A : üzenet elküldéséhez szükséges idő az A node-on
  - P_{AB}: jelterjedési idő
  - R_B: üzenet fogadásához szükséges idő a B node-on
- $d_2 = S_B + P_{BA} + R_A$
- Emlékeztető: $h_B(t_1)=h_A(t_1)+O(t_1)$, $h_B(t_4)=h_A(t_4)+O(t_4)$



$$h_B(t_2) = h_A(t_1) + O(t_1) + S_A + P_{AB} + R_B$$

$$h_A(t_4) + O(t_4) = h_B(t_3) + S_B + P_{BA} + R_A$$

$$O(t_4) = O(t_1) + \Delta O(t_4,t_1)$$

$\Delta O(t_4,t_1)$ lehet pozitív vagy negatív is

$$h_B(t_2) - h_A(t_1) = S_A + P_{AB} + R_B + O(t_4) - \Delta O(t_4,t_1)$$

$$h_A(t_4) - h_B(t_3) = S_B + P_{BA} + R_A - O(t_4)$$

kivonva a 2. egyenletet, szorozva ½ -el és használva Δ definícióját

$$\tfrac{1}{2}\{[h_B(t_2)-h_A(t_1)]-[h_A(t_4)-h_B(t_3)]\} = \Delta = \tfrac{1}{2}S^{UC} + \tfrac{1}{2}P^{UC} + \tfrac{1}{2}R^{UC} + O(t_4) - \tfrac{1}{2}\Delta O(t_4,t_1)$$

$\Delta - O(t_4) = \tfrac{1}{2}S^{UC} + \tfrac{1}{2}P^{UC} + \tfrac{1}{2}R^{UC} - \tfrac{1}{2}\Delta O(t_4,t_1)$: becsült ofszet hibája

ahol: $S^{UC} = (S_A - S_B)$, $P^{UC} = (P_{AB} - P_{BA})$, $R^{UC} = (R_B - R_A)$: eltérés az időzítések között

TPSN szinkronizáció hibaanalízise (magyarázat a levezetéshez)

- $h_B(t_2) = h_A(t_1) + O(t_1) + S_A + P_{AB} + R_B$ (Eq. 1)
  ugyanis:
  - $h_B(t_1)=h_A(t_1) + O(t_1)$ (Eq. 2)
    - Ez definíció: a B node lokális ideje megegyezik az A node lokális ideje + ofszet
    - Azért kell átszámítani az időpontot, mert a B node időskálájával fogunk dolgozni.
  - $h_B(t_2)=h_B(t_1)+d_1$ (Eq. 3)
    - Az első üzenet érkezésének időpontja [$h_B(t_2)$] a B node-on = Az első üzenet elküldésének időpontja [$h_B(t_1)$] a B node-on + az üzenettovábbítás ideje. d_1 helyébe beírható az előző dián található definíciója:
      - $d_1 = S_A + P_{AB} + R_B$ (Eq. 4)
    - (Eq. 3) egyenletbe behelyettesítve (Eq. 2) -t, és használva d_1 definícióját megkapjuk (Eq. 1)-et

TPSN szinkronizáció hibaanalízise (magyarázat a levezetéshez)

- $h_A(t_4) + O(t_4) = h_B(t_3) + S_B + P_{BA} + R_A$ (Eq. 5)
  ugyanis:
  - $h_B(t_4) = h_A(t_4) + O(t_4)$ (Eq. 6)
    - Ez definíció: a B node lokális ideje megegyezik az A node lokális ideje + ofszet
    - Azért kell átszámítani az időpontot, mert a B node időskálájával fogunk dolgozni.
  - $h_B(t_4) = h_B(t_3) + d_2$ (Eq. 7)
    - A második üzenet érkezésének időpontja [$h_B(t_4)$] a B node-on = A második üzenet elküldésének időpontja [$h_B(t_3)$] a B node-on + az üzenettovábbítás ideje. d_2 helyébe beírható a korábbi dián található definíciója:
      - $d_2 = S_B + P_{BA} + R_A$ (Eq. 8)
    - (Eq. 7) egyenletbe behelyettesítve (Eq. 6) -t, és használva d_2 definícióját megkapjuk (Eq. 5)-öt

Szinkronizáció hibaanalízise

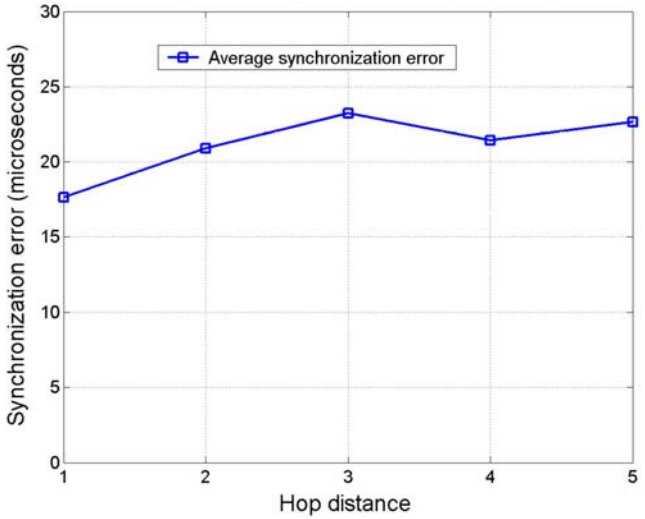
hiba = $\Delta - O(t_4) = ½S^{UC} + ½P^{UC} + ½R^{UC} - ½\Delta O(t_4,t_1) = ½(d_1 - d_2) - ½\Delta O(t_4,t_1)$

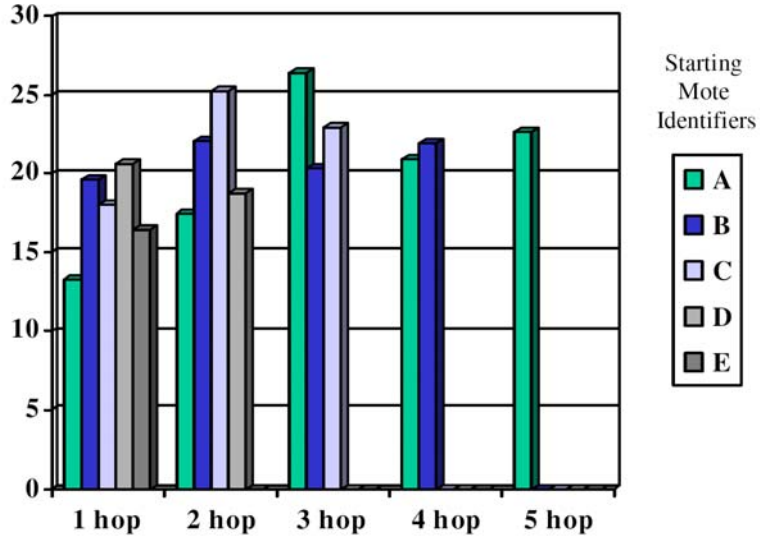
ahol: $S^{UC} = (S_A - S_B)$, $P^{UC} = (P_{AB} - P_{BA})$, $R^{UC} = (R_B - R_A)$: eltérés az időzítések között

- Δ értéke ideális esetben $O(t_4)$ lenne, mert ez alapján tudjuk t_4-ben korrigálni az időt, tehát a szinkronizáció hibája: $\Delta - O(t_4)$
- Láthatóan különbségük nem nulla, ennek okai:
  - Bizonytalanságok az időzítésben, pl: $S^{UC} = (S_A - S_B)$ az A illetve B üzenetének küldési időtartama közti különbség. Nem biztos hogy nulla pl. ha más a közeghozzáférési idő.
  - Összességében az $½(S^{UC} + P^{UC} + R^{UC}) = ½(d_1 - d_2)$ tagok az adatküldés és fogadás aszimmetriája miatt nem nullák.
  - Az üzenetváltások alatt is tovább csúszik egymáshoz képest a két óra, ezt az $½\Delta O(t_4,t_1)$ tag reprezentálja. Ez akkor okoz problémát, ha sok időt várnak az üzenetcserék között.
    - Pl. 20ppm hibájú kvarc esetén, amennyiben az üzenetváltások között 100msec telik el, úgy az $½\Delta O(t_4,t_1)$ értéke $1\mu\text{sec}$.
  - Legnagyobb bizonytalanság S^{UC} -ben van a közeghozzáférés miatt.
  - P^{UC}, tehát a terjedési idők különbsége elhanyagolható.

Multi-hop szinkronizáció

- A node-ok láncba szerveződve szinkronizálódnak egymáshoz, így közvetetten a root-hoz
- A hibák hop-onként akkumulálódnak
  - Véletlenszerű hibák pl. MAC
  - Determinisztikus részek pl. drift ($\frac{1}{2}\Delta O(t_4,t_1)$)
  - A hibák worst-case esetben hop-onként összeadódnak, de valójában nem ilyen rossz a helyzet, mert a véletlenszerű hibák átlagolódnak.
- A pontosság nem függ a hálózatban található node-ok számától csak a root-tól való távolságtól.





Abszolút hiba: előjelet nem vesszük figyelembe. Hiba páronként. Forrás: [3]

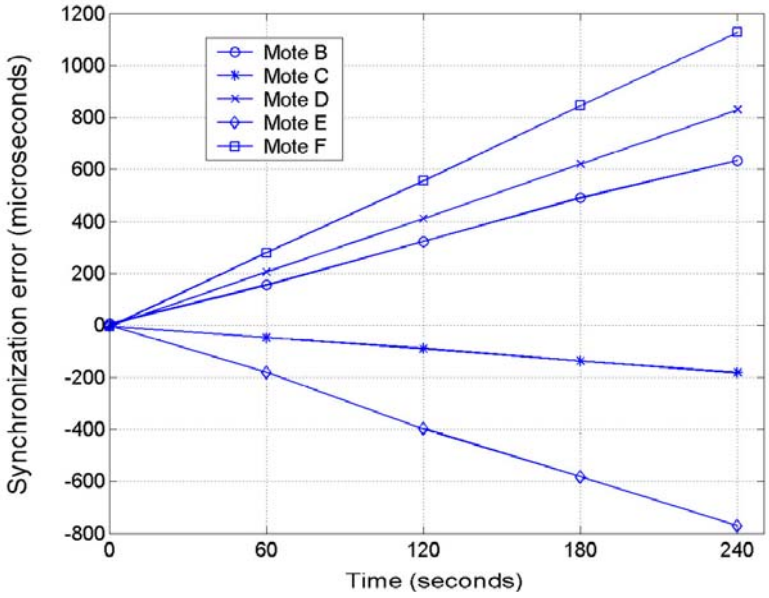
Szinkronizáció gyakorisága

- Az idő múlásával egyre jobban elcsúszhatnak a node-ok egymásól
- Szinkronizáció gyakorisága függ:
  - Minimálisan biztosítható szinkronizációs hiba
  - Az órák egymáshoz képesti elcsúszásának gyorsasága (drift)

Példa:

- 500μs megengedett hiba
- Algoritmusból származó hiba max: 50 μs
- Elcsúszás max.: 500 μs–50 μs=450 μs
- Az ábra alapján ez először kb. 60sec-nél következik be (MoteF és MoteE között)
- A szinkronizációk közötti idő növelhető a drift becslésével

Mérési eredmény:



Órák elcsúszása egymáshoz képest (szinkronizációs hiba). Forrás: [3]

Reference Broadcast Synchronization (RBS) [4]

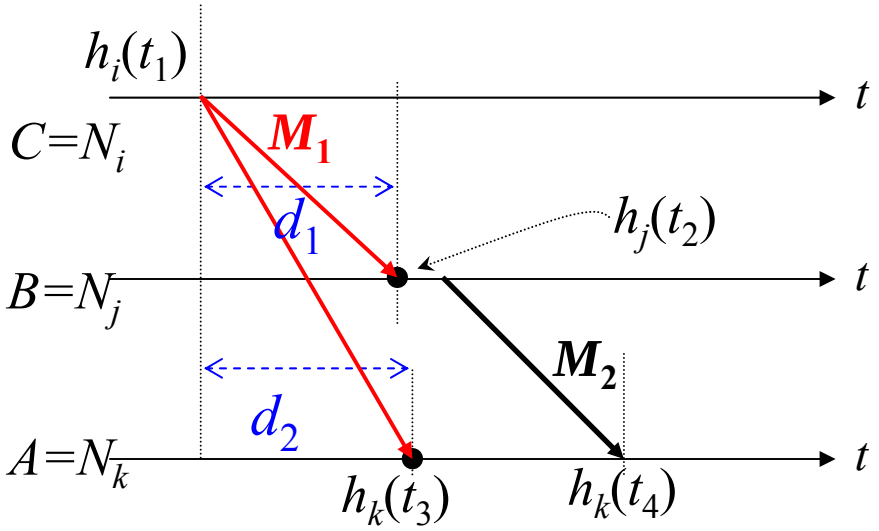
Reference Broadcast Synchronization

- Hagyományos megoldás: páronkénti szinkronizáció, aktivitás csak az érintett node-ok között
  - Gond: üzenet késleltetése, főleg a közeghozzáférés (adás megkezdése előtt) nagyon változó lehet.
- Receiver-receiver architektúra: a vevők egy csoportját szinkronizálja egymáshoz, a közeghozzáférési idő nem számít ☺
- Több node szinkronizálása egymáshoz, nem a hagyományos páronkénti szinkronizáció
- Külső esemény (rádiós üzenetek érkezése) szolgáltatja a szinkronizációs pontot
- Broadcast üzenetek kellenek
- Pl.: déli harangszó. Ha több ember hallja a harangszót, és a harang ütésének pillanatában mindenki megnézi az óráját, akkor ez alapján megállapítható, hogy mennyi az eltérés az órák között. Több mérés esetén becsülhető az órák driftje is.
  - Nem számít, ha a toronyóra késik, ha egyszerre halljuk a hangját akkor a többi órát egymáshoz képest pontosan tudjuk szinkronizálni
- Alapvetően időbélyeg konverziós szinkronizációt használ, de nem zárja ki bármilyen órahangolási (fizikai) szinkronizáció lehetőségét sem.

Reference Broadcast működése

- Broadcast üzenet küldése: N_i
- N_j és N_k feljegyzik az $\boldsymbol{M_1}$ üzenet érkezésének időpontját a lokális óráik alapján, ezek rendre $h_j(t_2)$ valamint $h_k(t_3)$
- N_j az $\boldsymbol{M_2}$ üzenetben elhelyezi $h_j(t_2)$-t (az üzenet érkezésének időpontja N_j órája alapján).
- Ez az időpont megegyezik az $\boldsymbol{M_1}$ üzenet érkezési idejével N_k lokális ideje alapján.
- A becslés alapján t_4-ben az ofszet meghatározható: $\Delta = h_k(t_3) - h_j(t_2)$, feltétel: $t_3 \approx t_2$
- Az ofszet alapján időbélyeg konverzió megvalósítható.

- Az üzenetcsere megoldható a bázisállomáson keresztül is, ha N_j és N_k nem „látja” egymást közvetlenül.



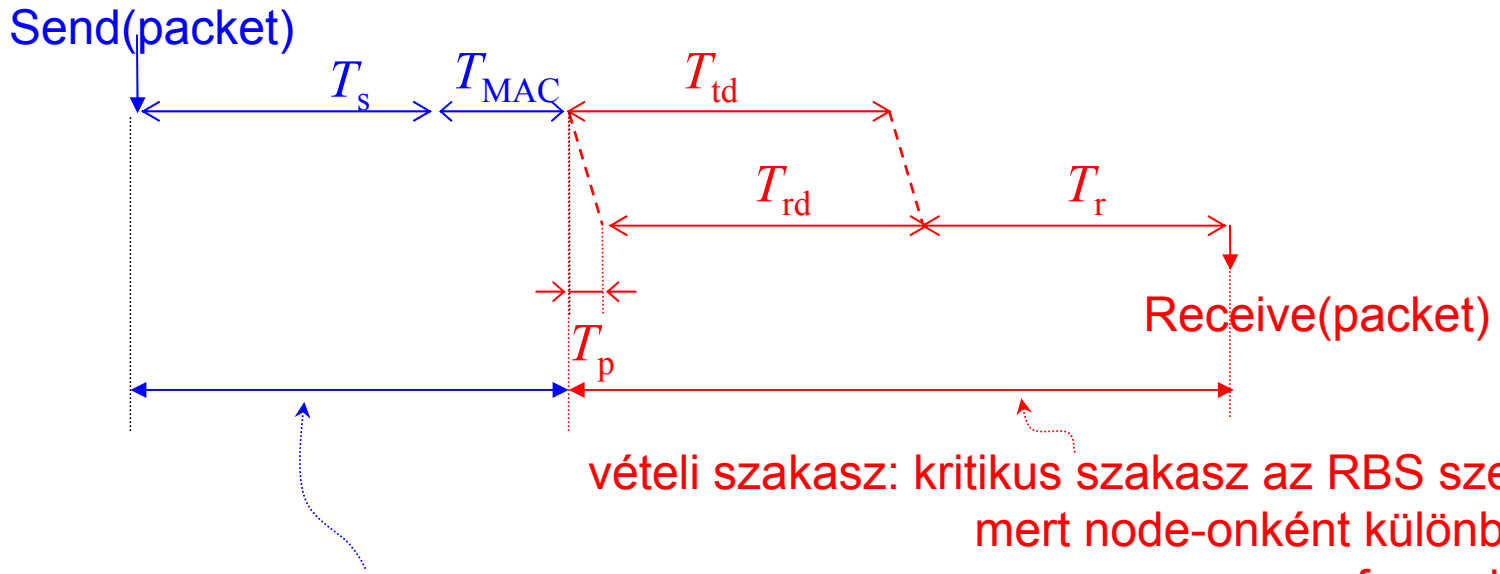
$\Delta = h_k(t_3) - h_j(t_2)$: becsült ofszet az órák között a $t_3 \approx t_2$ időpontban

Példa

- Egy városban, a Petőfi utcában lakik két szomszéd, legyenek Sz_1 és Sz_2. Egy reggelen a városon egy gyorshajtó száguldott keresztül és Sz_1 éppen a város egyik, Sz_2 pedig a város másik végén tartózkodott. A város hossza 5km. Az autó érkezési idejeként a következőket jegyezte fel a két szomszéd:
  - Szomszéd1 (Sz_1): 8:00:10 = h_1^{E1} ($E1$: a város egyik végén jár az autó)
  - Szomszéd2 (Sz_2): 8:06:40 = h_2^{E2} ($E2$: a város másik végén jár az autó)
- Mindkét szomszéd jelentette a rendőrségen az esetet, és a rendőrség az Ő beszámolóik alapján szeretné bizonyítani a gyorshajtást. Sikerül-e ezt megtenni?
  - A szomszédok által mért időkülönbség: τ=6min 30sec = 390sec = $h_2^{E1} - h_1^{E2}$
  - Ezen értékek alapján az autó sebessége v=5000m/390sec = 12.8m/s = 46.15km/h, tehát nem bizonyítható a gyorshajtás.
- Tudjuk azonban, hogy előző nap este tűzijáték volt a városban. Mivel a tűzijáték késett a meghírdetett időponthoz képest, így mindkét szomszéd folyamatosan az óráját figyelte, várva a tűzijátékot. Az első rakéta robbanása Sz_1 szerint 22:10:00-kor történt Sz_2 szerint viszont 22:11:20-kor. Mit mondhatunk ezek alapján a gyorshajtó sebességéről?
  - Ha a Reference Broadcast szinkronizációt szeretnénk felhasználni a megoldásra, akkor a referencia üzenet ebben az esetben a rakéta felrobbanása. Ezek alapján az órák közötti ofszet: $\Delta = h_2(t_{\text{rakéta}}) - h_1(t_{\text{rakéta}}) = 80\text{sec}$.
  - Valójában tehát Sz_2 az autót $h_1^{E2}=h_2^{E2}$–80sec=8:06:40–80sec=8:05:20-kor észlelte.
  - Emiatt az autó valójában: τ'=5min10sec=310sec idő alatt száguldott keresztül a városon.
  - A sebessége tehát v=5000m/310sec = 16.1m/s = 58km/h, ez alapján a gyorshajtás bizonyítható
- A példában a tűzijáték kezdete, mint esemény, tekinthető reference broadcastnak.
- Az órák ofszete ez alapján meghatározható.
- Az ofszet alapján korrigálni tudjuk az időbélyegeket.

Időzítés analízis RBS esetén

- Az üzenetküldés időzítése:



vételi szakasz: kritikus szakasz az RBS szempontjából mert node-onként különbözhet, így nem ugyanaz a referenciapont

Adás szakasza: minden node esetén megegyezik, így a szinkronizáció szempontjából nem kell figyelembe venni ☺

A példában a tűzijáték szervezőjének a rakéta kilövéséig eltöltött ideje. Pl. nem számít, hogy pontosan lövik-e ki, vagy 10 percet késik a várt kilövésig.

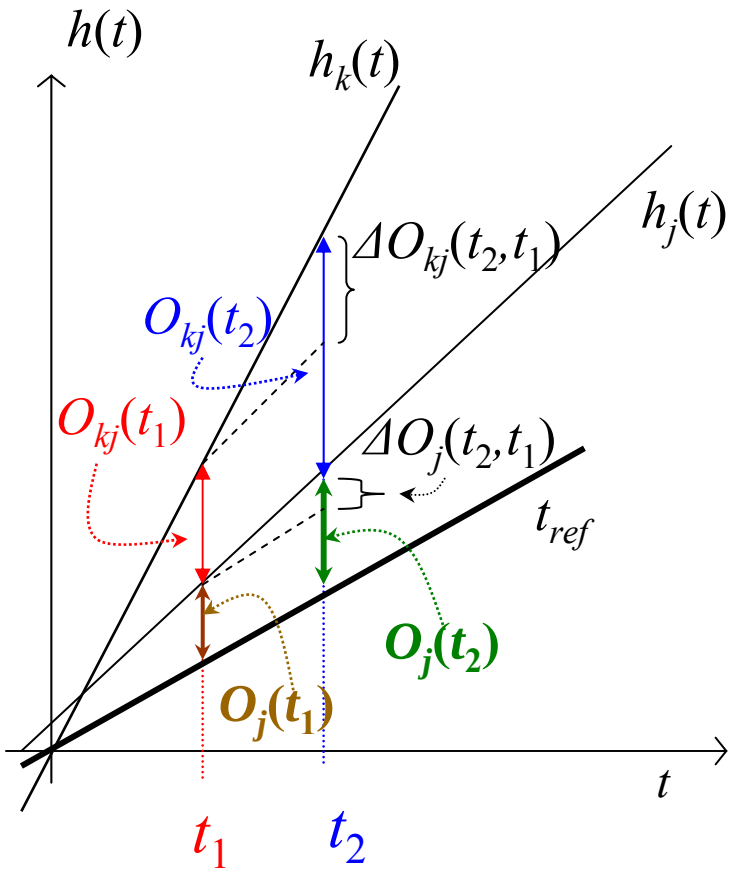
Pl. az egyik szomszédnak csak a felesége szól a tűzijáték kezdetéről, így Ő csak később néz rá az órájára. Vagy az egyik szomszéd távolabbról szemléli, így a hang lassabban ér el hozzá.

Időzítés analízis RBS esetén

- Az RBS szinkronizáció szempontjából kritikus szakasz a vételi szakasz.
- A vétel ideje kevésbé ingadozó, mint az adó oldali késleltetés, ugyanis a közeghozzáférés ideje erősen függ a csatorna foglaltságától.
- **Nagy előny: a nem determinisztikus közeghozzáférés nem befolyásolja a szinkronizáció pontosságát!**

Időalapok jellemzése (RBS)
Levezetés: kiegészítő anyag

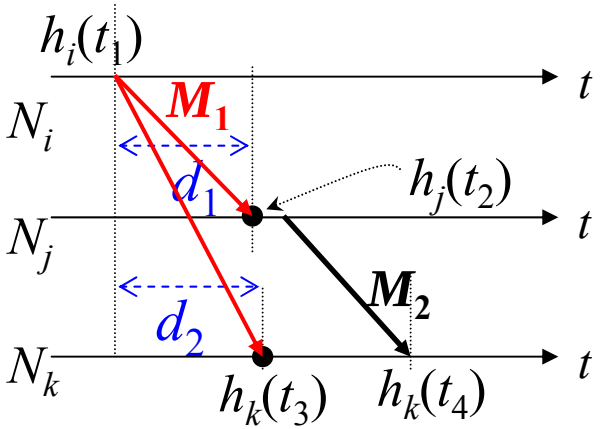
- Idővariáns ofszet
  - $O_{kj}(t_1)$: ofszet a t_1 időpontban N_k és N_j node között
  - $\Delta O_{kj}(t_2,t_1)$: ofszet változása a t_1 és t_2 időpontok között
    $\Delta O_{kj}(t_2,t_1)=O_{kj}(t_2) - O_{kj}(t_1)$
  - $O_j(t_1)$: N_j órájának ofszete a t_1 időpontban a referenciaórához képest
  - $\Delta O_j(t_2,t_1)$: ofszet változása a t_2 és t_1 időpontok között
    $\Delta O_j(t_2,t_1)=O_j(t_2) - O_j(t_1)$
  - Következő oldalon felhasználjuk, hogy:
    $O_{kj}(t_3)= O_{kj}(t_4)+\Delta O_{kj}(t_3,t_4)$



RBS szinkronizáció hibaanalízise

Levezetés: kiegészítő anyag

- Algoritmusban figyelmen kívül hagyott paraméterek:
  - Drift és bizonytalanság a kommunikációs időkben
- $d_1 = S_i + P_{ij} + R_j$
  - d_1: üzenet küldés időtartama i-től j-ig
  - S_i : üzenet elküldéséhez szükséges idő az N_i node-on
  - P_{ij}: jelterjedési idő
  - R_j: üzenet fogadásához szükséges idő az N_j node-on
- $d_2 = S_i + P_{ik} + R_k$
- Emlékeztető: $O_{kj}(t)=h_k(t) - h_j(t)$ és $O_j(t) = h_j(t) - t$



$t_2=t_1+S_i + P_{ij} + R_j \rightarrow h_j(t_2)= t_1 + O_j(t_2) + S_i + P_{ij} + R_j$ mivel $t_2=h_j(t_2)-O_j(t_2)$

$t_3=t_1+S_i + P_{ik} + R_k \rightarrow h_k(t_3)= t_1 + O_k(t_3) + S_i + P_{ik} + R_k$

$O_k(t_3)-O_j(t_2) = [O_k(t_3)-O_j(t_3)] + [O_j(t_3) -O_j(t_2)]= O_{kj}(t_3)+\Delta O_j(t_3,t_2)= O_{kj}(t_4)+\Delta O_{kj}(t_3,t_4) +\Delta O_j(t_3,t_2)$

Az első két egyenletet kivonva egymásból, és a harmadikat felhasználva:

$\Delta= h_k(t_3)-h_j(t_2)=O_{kj}(t_4)+\Delta O_{kj}(t_3,t_4) +\Delta O_j(t_3,t_2) + P^{UC} + R^{UC}$, ahol Δ definícióját használtuk

ahol: $P^{UC} = (P_{ik} - P_{ij})$ valamint $R^{UC} = (R_k - R_j)$

RBS szinkronizáció hibaanalízise (Magyarázat a levezetéshez)

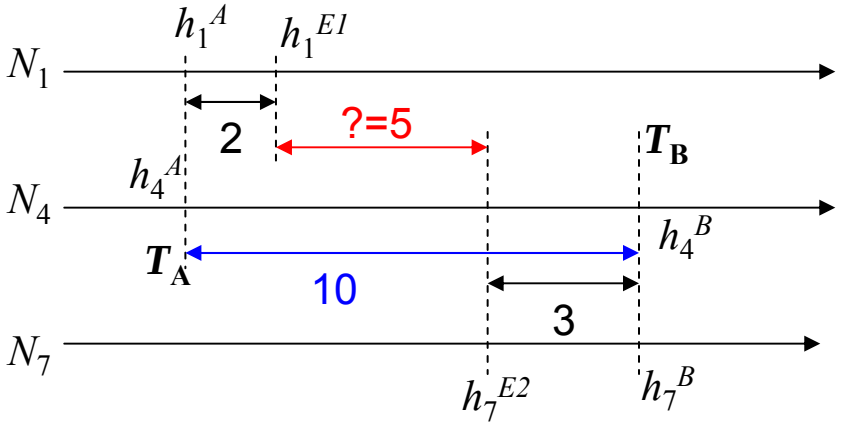
- $t_2=t_1+d_1=t_1+S_i+P_{ij}+R_j$ (Eq 1.)
  - Az ***M1*** üzenet érkezési időpontja az N_j node-on referencia idő szerint [t_2] = az üzenet elküldésének időpontja referencia idő szerint [t_1] + az üzenetküldés ideje [d_1] N_j node és N_j node között
  - Felhasználtuk d_1 definícióját: $d_1 = S_i + P_{ij} + R_j = t_2 - t_1$
  - Ebben az esetben a referencia időskálát használjuk, minden időpontot ide transzformálunk.
- $t_3=t_1+d_2=t_1+S_i+P_{ik}+R_k$ (Eq 2.)
  - Az ***M1*** üzenet érkezési időpontja az N_j node-on referencia idő szerint [t_2] = az üzenet elküldésének időpontja referencia idő szerint [t_1] + az üzenetküldés ideje [d_1] N_j node és N_k node között
  - Felhasználtuk d_2 definícióját: $d_2 = S_i + P_{ik} + R_k = t_3 - t_1$
  - Ebben az esetben is a referencia időskálát használjuk, minden időpontot ide transzformálunk.
- Vegyük észre: d_1 és d_2 kifejezésben az üzenet elküldésének ideje (S_i) megegyezik, csupán a terjedési idő illetve a fogadási idő (P és R) különbözik. Ennek oka, hogy az üzenetnek az N_j node-on töltött ideje (S_i) megegyezik, mert ugyanarról az üzenetről van szó. Ezután viszont az adott üzenet külön úton terjed míg N_j és N_k node feldolgozza.

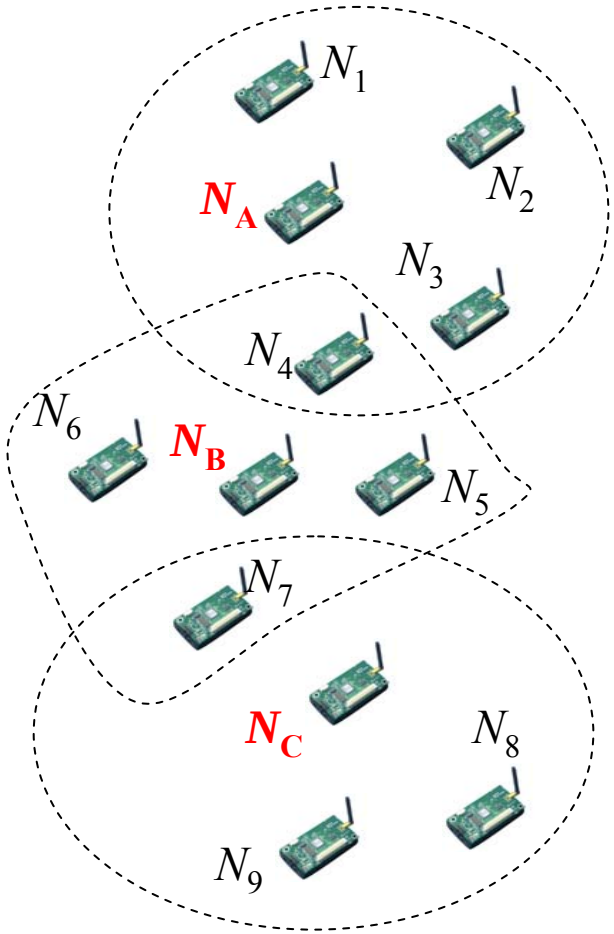
RBS szinkronizáció hibaanalízise

- $\Delta = h_k(t_3) - h_j(t_2) = O_{kj}(t_4) + \Delta O_{kj}(t_3,t_4) + \Delta O_j(t_3,t_2) + P^{UC} + R^{UC}$
- ahol
  - Δ az ofszet számított értéke, míg a valódi értéke $O_{kj}(t_4)$ lenne
  - $P^{UC} = (P_{ik} - P_{ij})$ valamint $R^{UC} = (R_k - R_j)$: időzítésekben tapasztalható bizonytalanság: Példa: a tűzijáték hangja különböző időpontokban érkezik a terjedési idők különbsége miatt.
  - $\Delta O_{kj}(t_3,t_4)$ az ofszet változása annak felhasználása és számítása között
    - A gyorshajtós példában ez a következő módon jelentkezik. Az órák ofszetét este 22:10:00-kor mértük. Reggelre ez az ofszet változhat/változik, ez a ΔO_{kj}. Ezt a hibát a gyorshajtós példában nem vettük figyelembe! ΔO_{kj} a drift alapján számítható.
  - $\Delta O_j(t_3,t_2)$: ofszet változása az N_j node-on a referencia érkezési időpontjai (tehát t_3 és t_2) között.
    - Gyorshajtós példa: legyen τ az az idő, amennyivel Sz_2 később észleli a tűzijátékot, mint Sz_1. $\Delta O_j(t_3,t_2)$ az az idő, amennyit Sz_1 órája elcsúszik Sz_2 órájához képest a τ idő alatt. Mivel τ nagyon kicsi, így $\Delta O_k(t_3,t_2)$ itt gyakorlatilag elhanyagolható.
- A $\Delta O_{kj}(t_3,t_4)$ kompenzálható, ha több üzenetváltás alkalmával a driftet is becsüljük. Ilyenkor ugyanis $O_{kj}(t_4) = \Delta + \Delta O_{kj}(t_3,t_4) = \Delta + \rho_{kj} \cdot (t_4 - t_3)$ számítható.
- **Lényeges előny:**
  - **Az adatküldés és közeghozzáférés nem determinisztikus időzítése nem jelentkezik a hibában**

RBS multi-hop szinkronizáció

- A hálózatot olyan alhálózatokra osztjuk, melyeknek van közös pontjuk.
- N_A, N_B, N_C : broadcast források
- $N_1 \ldots N_9$: szinkronizálandó node-ok
- Közös node-ok: N_4, N_7, N_9
- A közös node-okon keresztül lehet szinkronizációt végezni.
- Példa: N_A és N_B szinkronizációs üzenetet küld T_A és T_B időpontokban. Az N_1 node T_A után 2sec-al érzékel egy objektumot (h_1^{E1}), míg az N_7 node T_B előtt 3sec-al érzékeli az objektumot (h_7^{E2}). Mennyi idő alatt jut el az objektum N_1-től N_7-ig, ha tudjuk, hogy N_4 az N_A broadcast üzenetet 10sec-el az N_B broadcast üzenete előtt kapta meg?





RBS multi-hop szinkronizáció

- Általánosabb formában multi-hop szinkronizáció a következő módon valósítható meg (szomszédos csoportokra):
- Legyen adott két csoport: $\{N_A, N_i, N_j\}$, $\{N_B, N_k, N_j\}$
  - Tegyük fel, hogy N_j node a két alhálózat közös pontja.
  - Legyen N_A és N_B két szomszédos alhálózatban a referenciaforrás.
  - N_A broadcast üzenete alapján meghatározható N_i és N_j közötti ofszet: $\Delta_{ij} = h_i - h_j$
  - N_B broadcast üzenete alapján meghatározható N_k és N_j közötti ofszet: $\Delta_{kj} = h_k - h_j$
  - A két ofszet különbségével meghatározható a két alhálózat N_i és N_k node-jai közötti ofszet:
    $\Delta_{ki} = \Delta_{kj} - \Delta_{ij} = (h_k - h_j) - (h_i - h_j) \approx h_k - h_i$ (azért $\approx$, mert a Δ-kat különböző időpontban mérjük)
  - Ez alapján: $h_k(t) \approx h_i(t) + \Delta_{ki}$. A referencia üzenetek közötti időelcsúszás hibát okoz.
  - Az algoritmus tetszőleges alhálózaton keresztül folytatódhat.
- A szinkronizációs folyamat során nem fontos dedikált node-nak küldenie a referencia üzenetet:
  - Például elképzelhető, hogy először N_A referencia üzenete alapján N_i és N_j szinkronizálódik egymáshoz, majd N_i referencia üzenete alapján N_A és N_j szinkronizálódik egymáshoz. Így már az alhálózat összes nodeja szinkronizált egymáshoz képest.

RBS és TPSN összehasonlítása

- Az adási időben bekövetkező bizonytalanság az RBS-nél nem jelentkezik, mivel itt egyazon üzenetről van szó mindkét node esetén. Ez platformtól függően viszonylag nagy értéket vehet fel.
  - Az RBS tehát alkalmas a közeghozzáférési idő kiküszöbölésére
  - Kihasználja azonban a vevők hasonló viselkedését (különböző eszközöknél gond lehet)
- A TPSN-nél a terjedési és fogadási időben jelentkező bizonytalanság (tehát $P^{UC} + R^{UC}$) ½ súllyal szerepel, mivel két üzenetváltást használ, így ezen hibák átlagolódnak és kisebb hibát okoznak, mint az RBS-nél.
- TPSN: páronkénti szinkronizáció → sok üzenetváltás
- RBS: broadcast üzenet → több node is szinkronizálódhat
  - Pl.: N_i üzenetét megkapja N_j (referencia) és N_k, N_l, N_m … :ez egyetlen üzenet. N_j is küldhet broadcast üzenetet, amit ha N_k, N_l, N_m … is megkapnak, akkor ezzel a két üzenettel egyszerre szinkronizáljuk N_j, N_k, N_l, N_m … node-okat