7. előadás

Elnevezési rendszerek

3. rész

Szinkronizálás

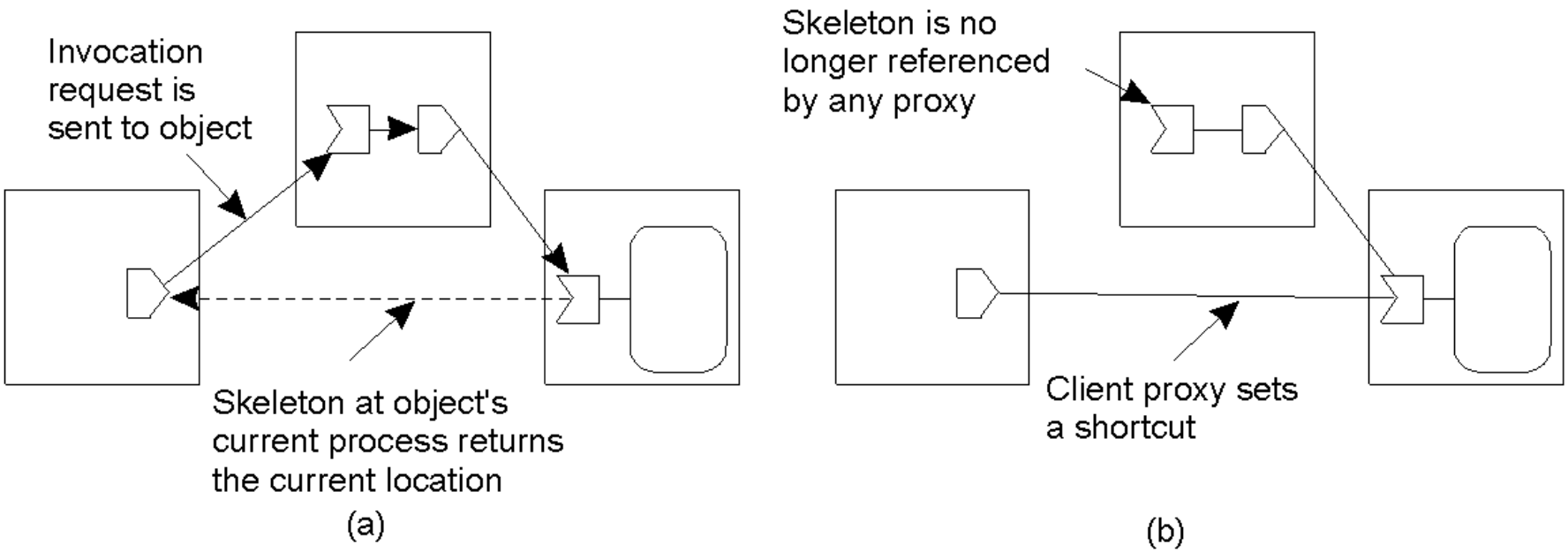
1. rész

Elnevezési rendszerek

3. rész

A nem hivatkozott entitások eltávolítása

Motiváció

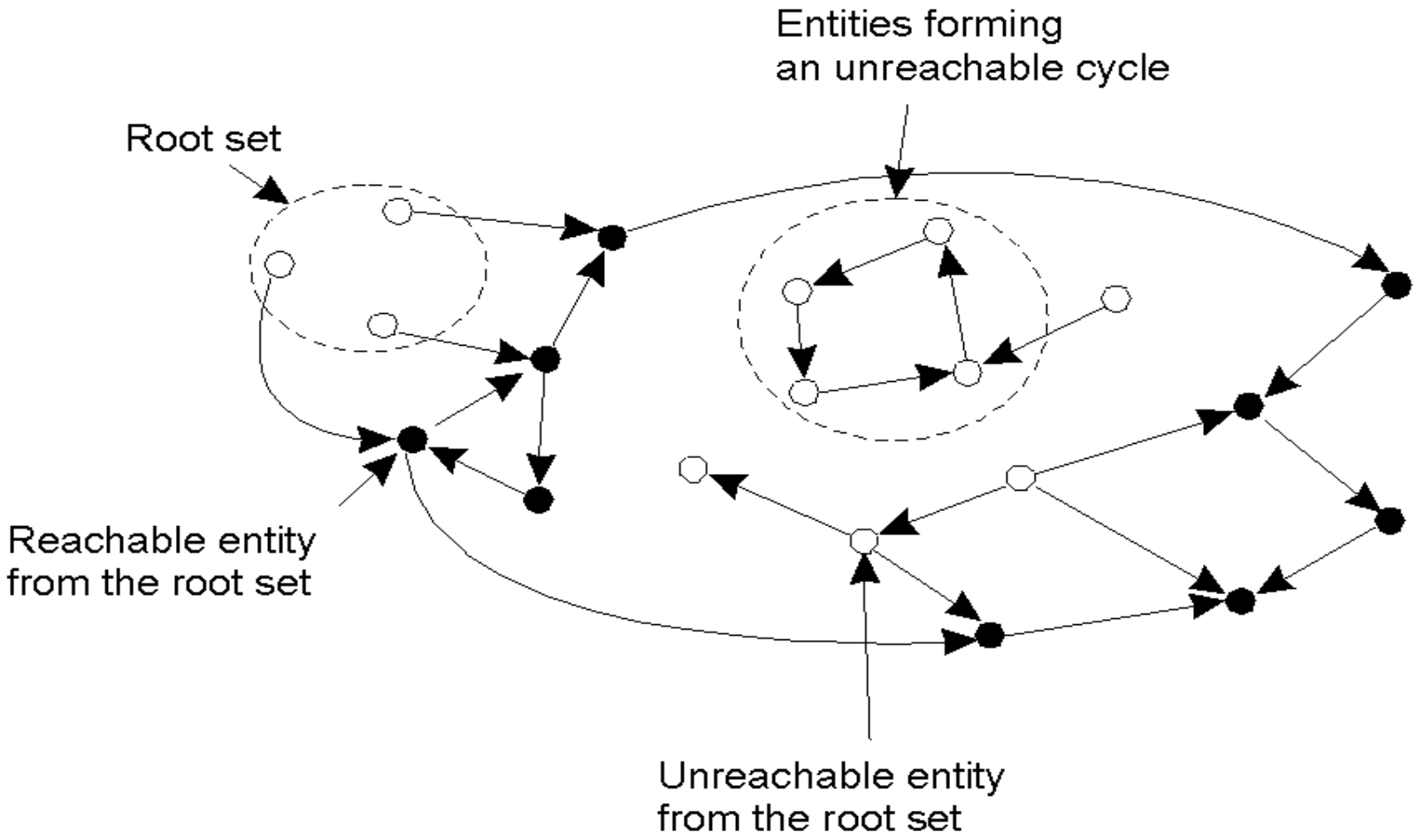


A nem elérhető hivatkozást el kell távolítani

=>

elosztott szemétgyűjtő

A nem hivatkozott objektumok problémája

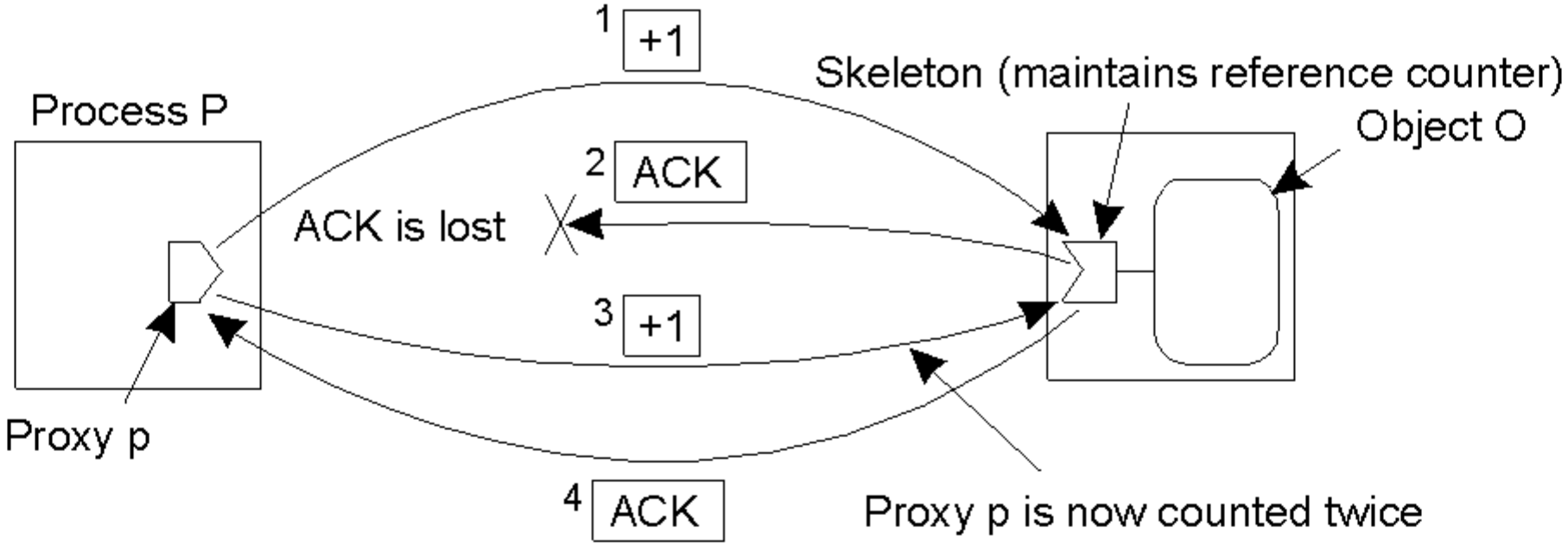


Példa az egymásra hivatkozó objektumokat ábrázoló gráfra.

Egyszerű hivatkozásszámlálás

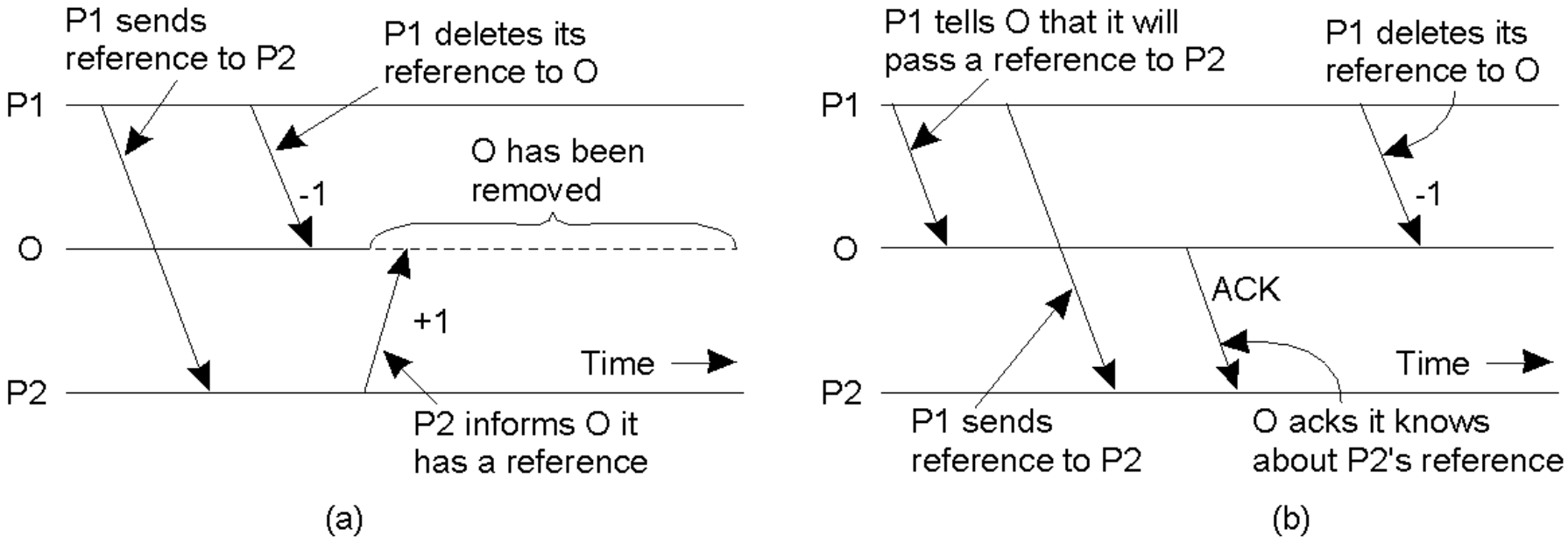
- Objektumra mutató hivatkozások megszámolása
- Hivatkozás létrehozásakor növeljük
- Hivatkozás eltávolításakor csökkentjük
- Ha a számláló 0, az objektum törölhető

Probléma az egyszerű hivatkozásszámlálással (1)



A hivatkozásszámláló helyes értékének beállítási problémája megbízhatatlan kommunikáció esetén.

Probléma az egyszerű hivatkozásszámlálással (2)



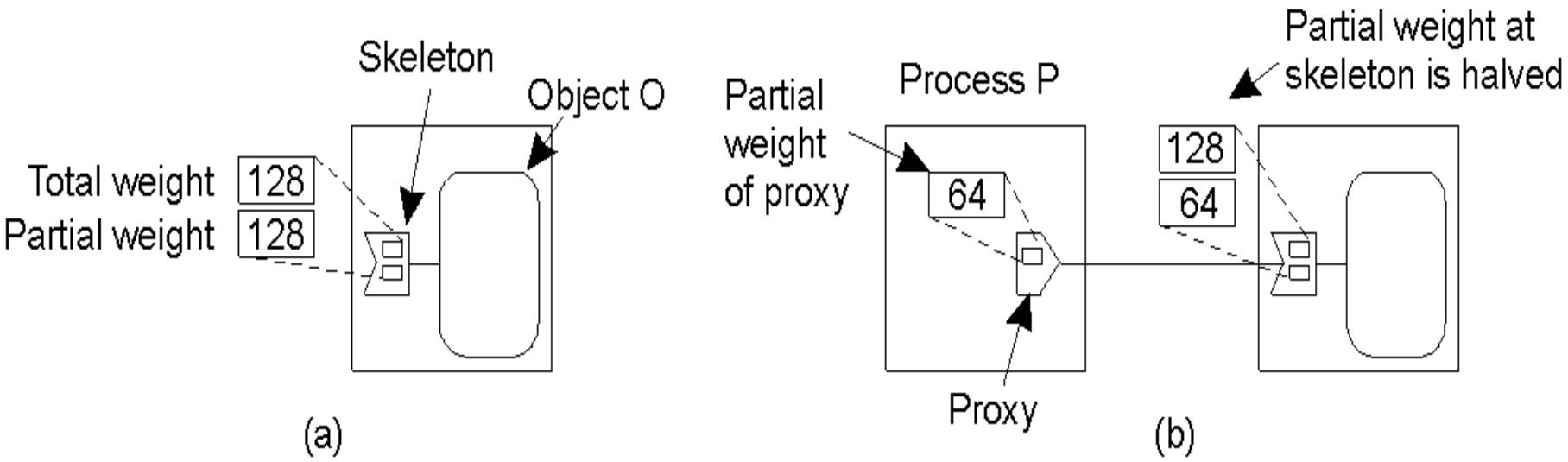
a) A hivatkozás átmásolása egy másik folyamatnak és a hivatkozásszámláló elkésett növelése

b) A megoldás

Fejlettebb hivatkozásszámlálás (1)

- Súlyozott hivatkozásszámlálás – csak csökkentés
- Minden objektumnak előre meghatározott teljes súlya van. <-> részleges súly
- Új hivatkozáskor, másoláskor a súly felét átadjuk
- Töréskor: az objektum a teljes súlyát a törlendő részleges súlyával csökkenti
- Ha a teljes súly 0, az objektum törölhető

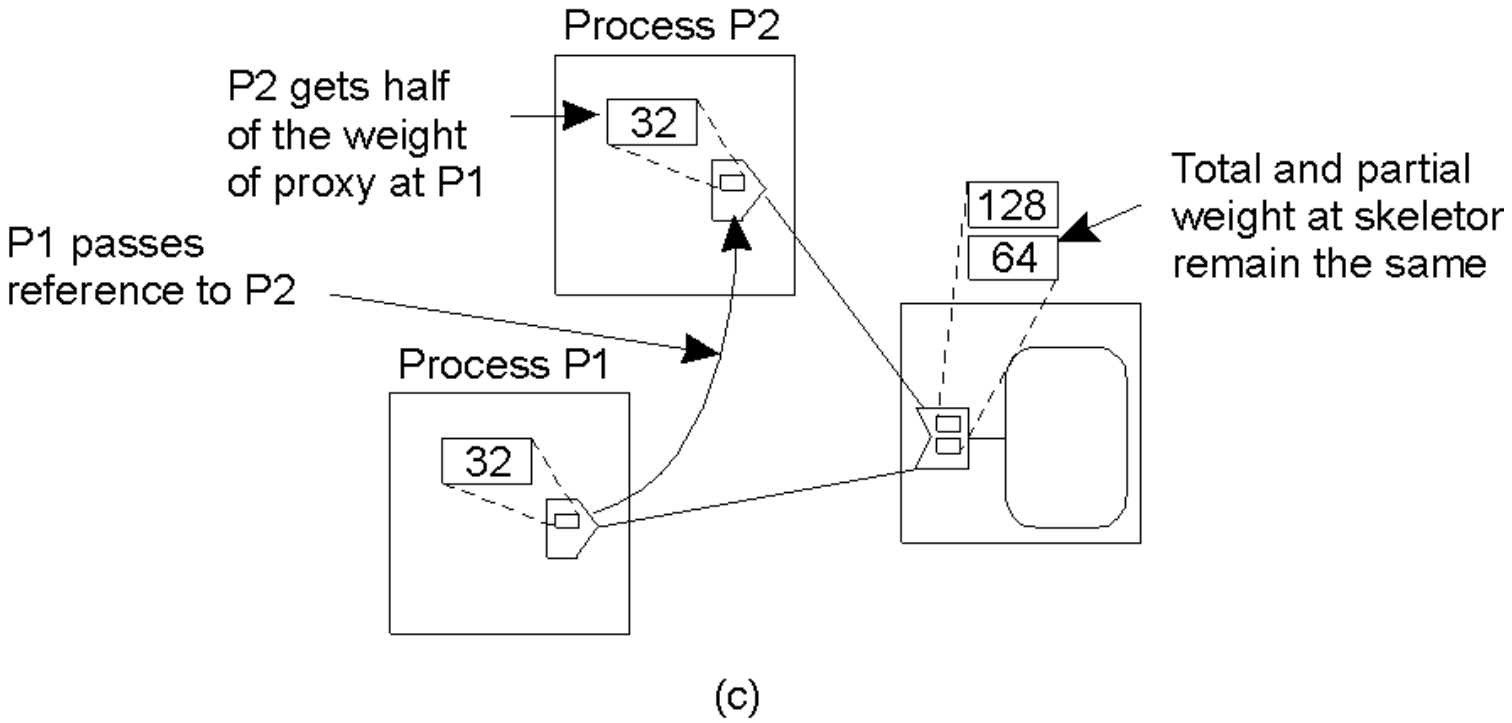
Fejlettebb hivatkozásszámlálás (2)



a) A súlyok kezdeti hozzárendelése súlyozott hivatkozásnál.

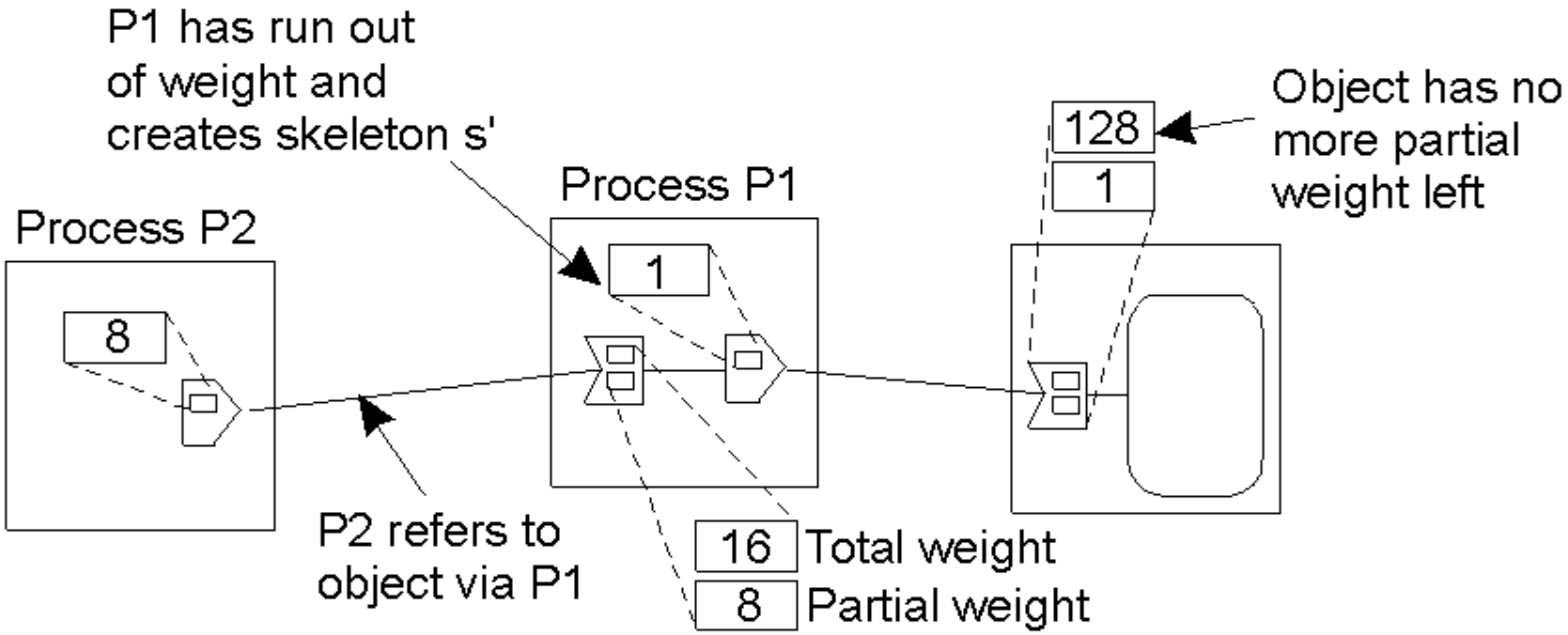
b) Súlyok hozzárendelése az új hivatkozás létrehozásakor.

Fejlettebb hivatkozásszámlálás (3)



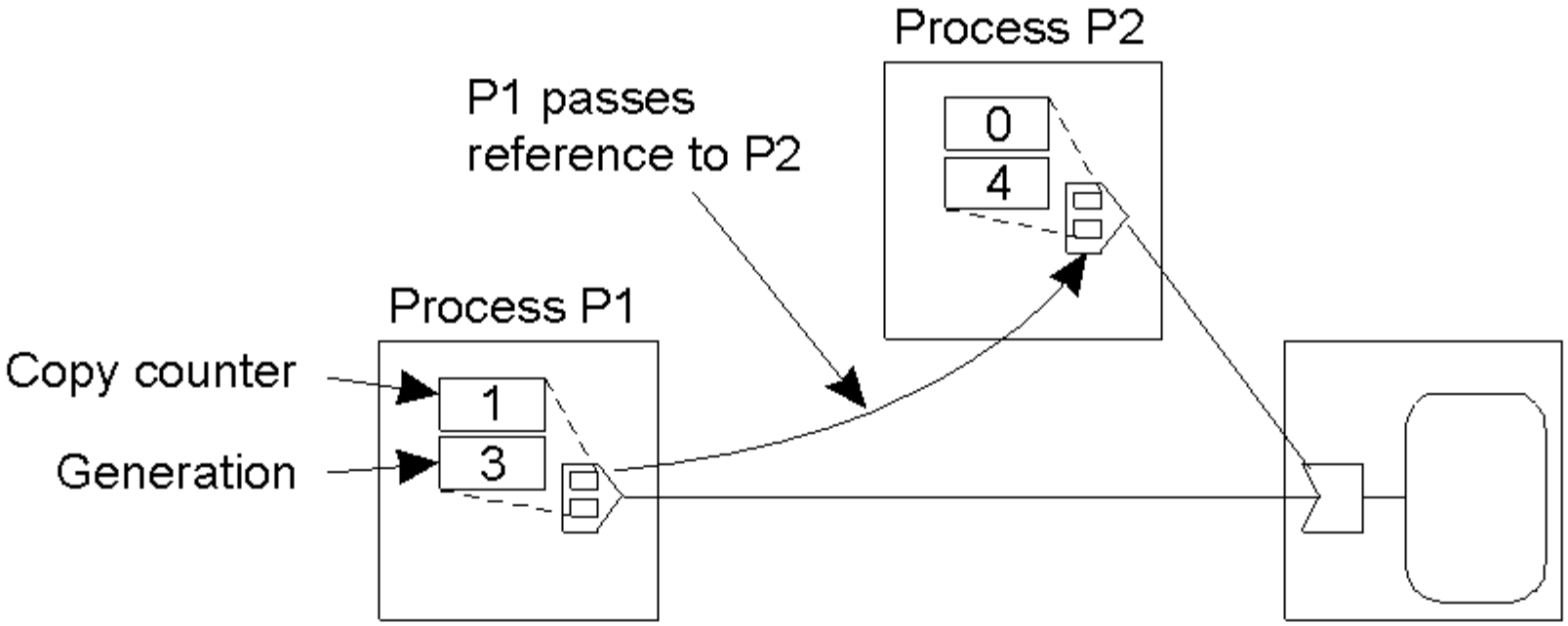
c) Súlyok hozzárendelése a hivatkozás másolásánál.

Fejlettebb hivatkozásszámlálás (4)



Indirekció alkalmazása, amikor egy hivatkozás részleges súlya eléri az egyet.

Fejlettebb hivatkozásszámlálás (5)



Távoli hivatkozás létrehozása és átmásolása nemzedéki hivatkozásszámlálás esetén.

Hivatkozáslista

- A váz nyilvántartja a rá hivatkozó helyetteseket (mutató)
- Létező elem hozzáadása, nem létező törlése idempotens művelet
- Létrehozáskor az új elküldi azonosítóját a váznak
- Másoláskor az új értesíti a vázat

- Pl. Java RMI

Elérhetetlen entitások azonosítása (1)

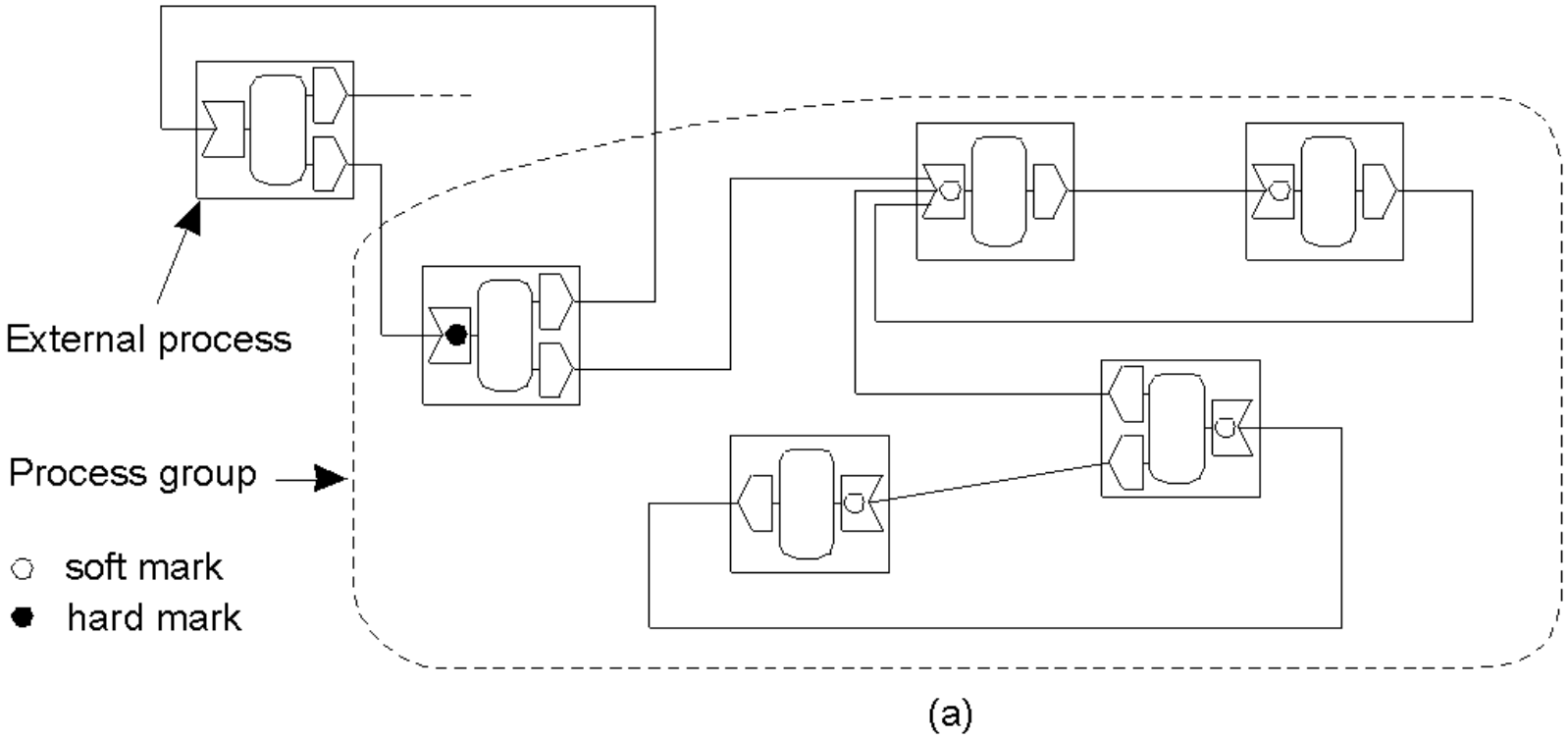
- Egyszerű nyomkövetés az elosztott rendszerben
  - egyprocesszoros rendszerekben
  - jelöl és takarít (mark and sweep)
  - a jelölőszakasz a gyökérkészletből indulva megjelöli az entitásokat
    - fehér – minden entitás kezdetben
    - szürke – ami elérhető, de még nem dolgoztuk fel (a folyamat előrehaladása közben)
    - fekete – ami elérhető a gyökérből (a jelölő szakasz végére)
  - a takarítószakasz törli a meg nem jelölteket

Elérhetetlen entitások azonosítása (2)

- Csoporton belüli nyomkövetés
  - vázak megjelölése
  - a jelölések kiterjesztése a vázakról a helyettesekre
  - a jelölések kiterjesztése a helyettesekről a vázakra
  - stabilizálás az előző két lépés megismétlésével
  - szemét eltávolítása

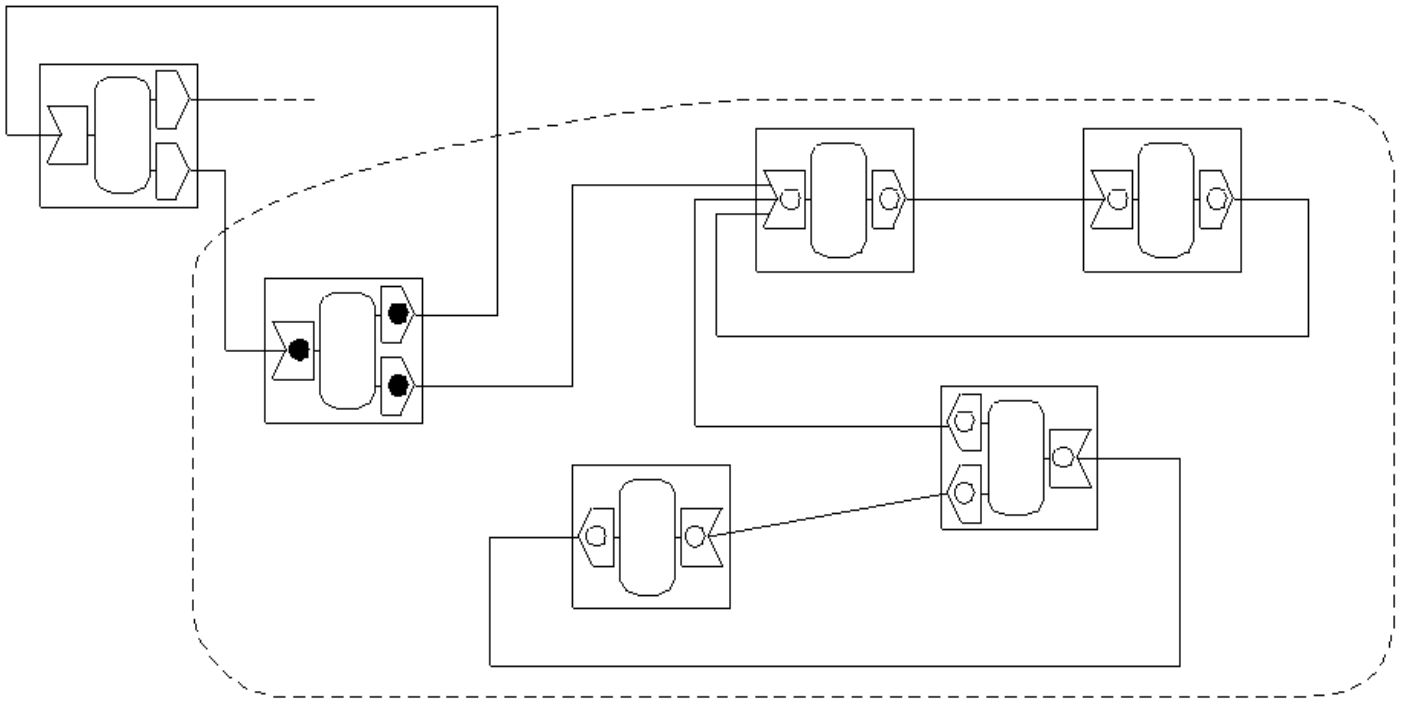
  - váz lehet: puha / kemény
  - a helyettes lehet: puha / kemény / semmilyen

Csoporton belüli nyomkövetés (1)



A vázak kezdeti jelölése.

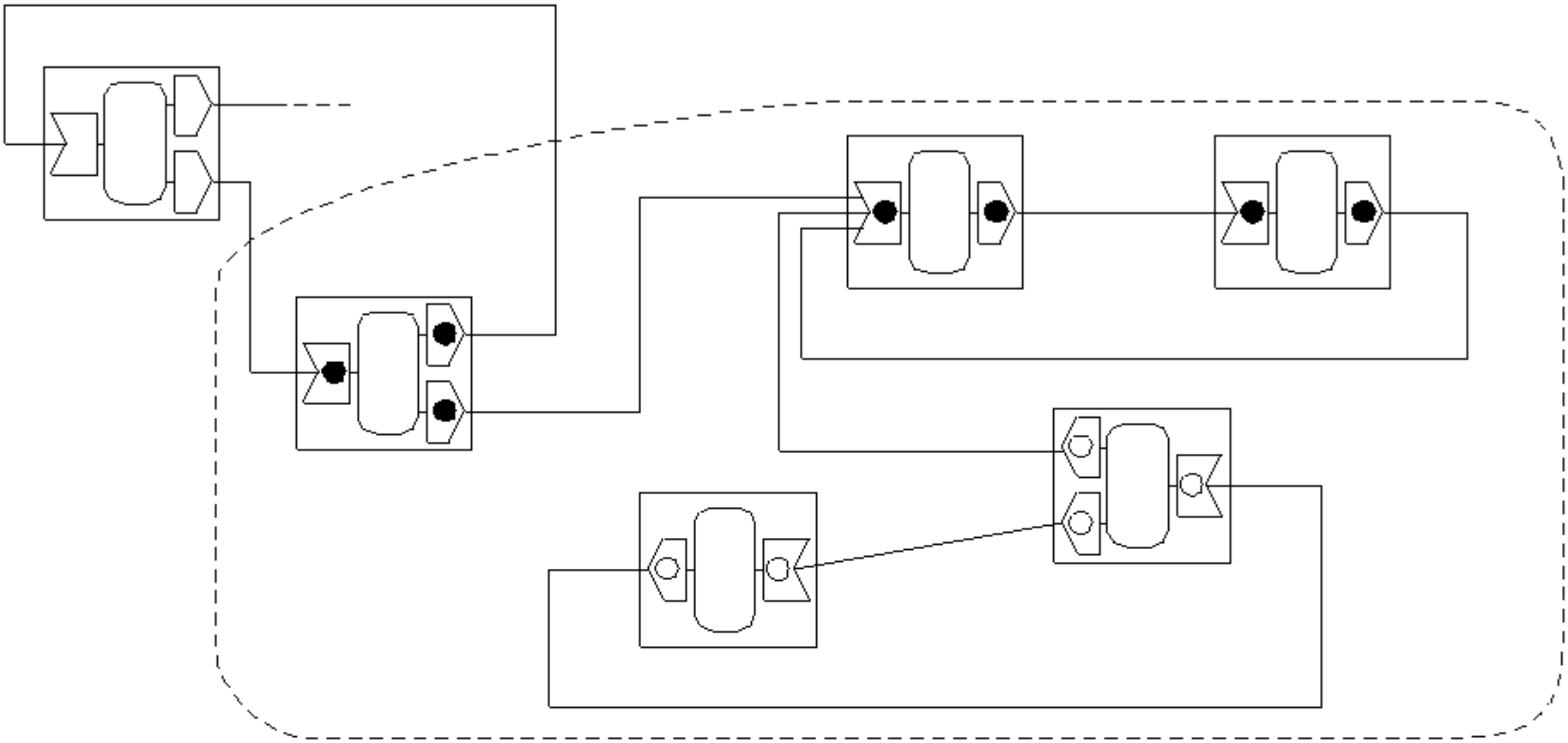
Csoporton belüli nyomkövetés (2)



(b)

A folyamatok helyi jelölésmásolásának befejezése utáni állapot.

Csoporton belüli nyomkövetés (3)



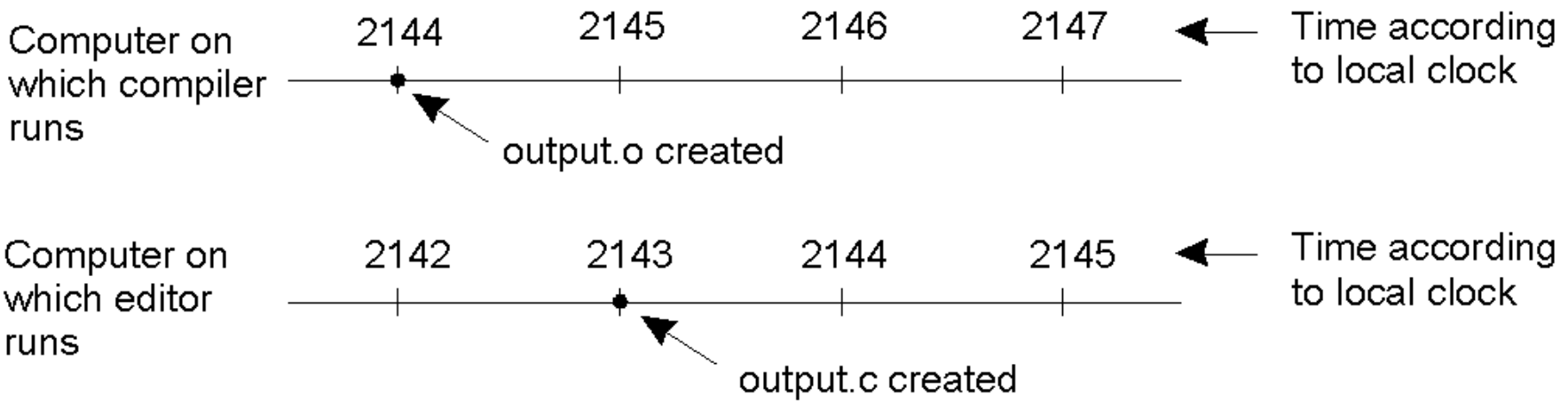
(c)

Végső jelölések.

Szinkronizálás

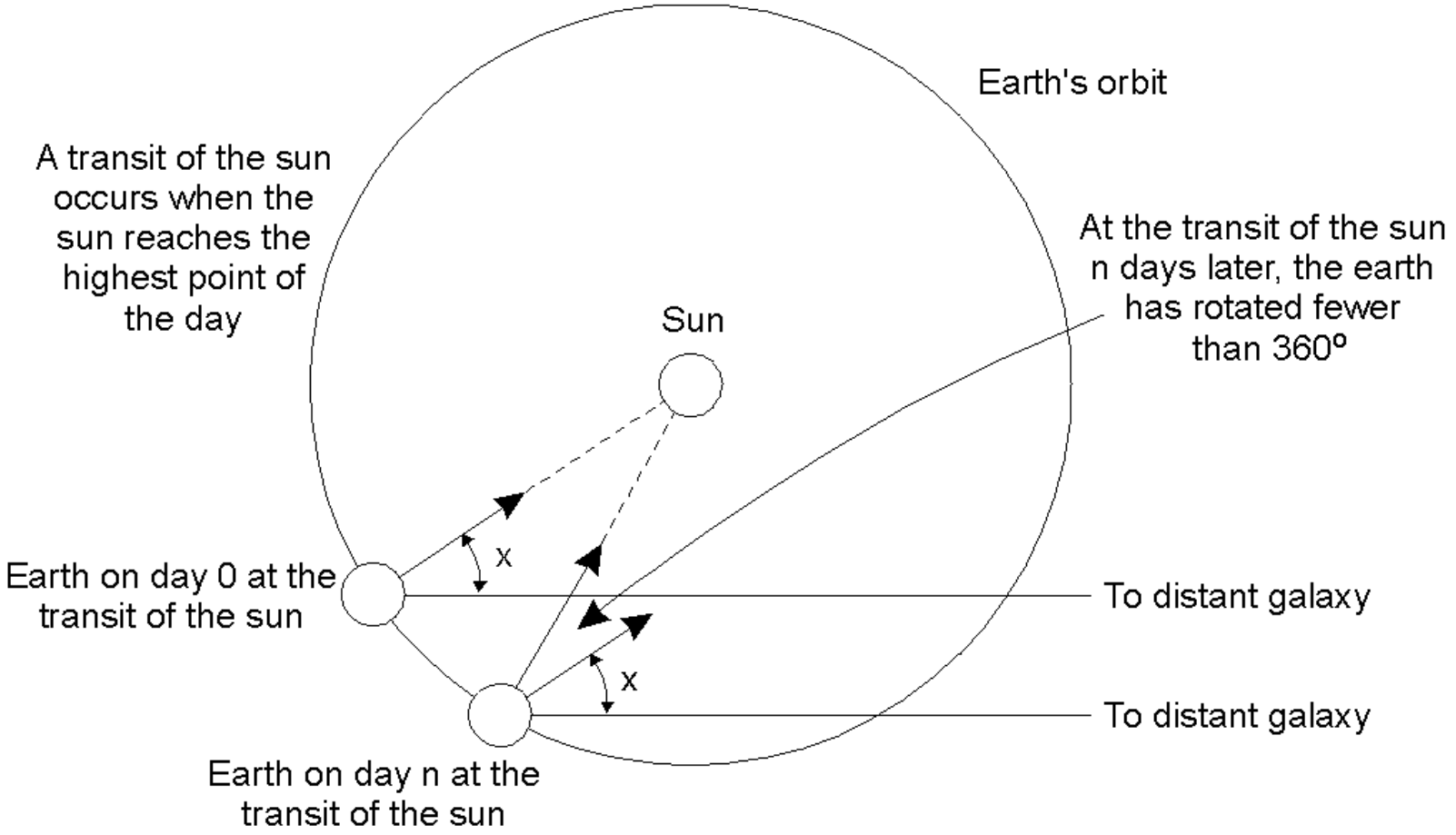
1. rész

Az órák szinkronizálása



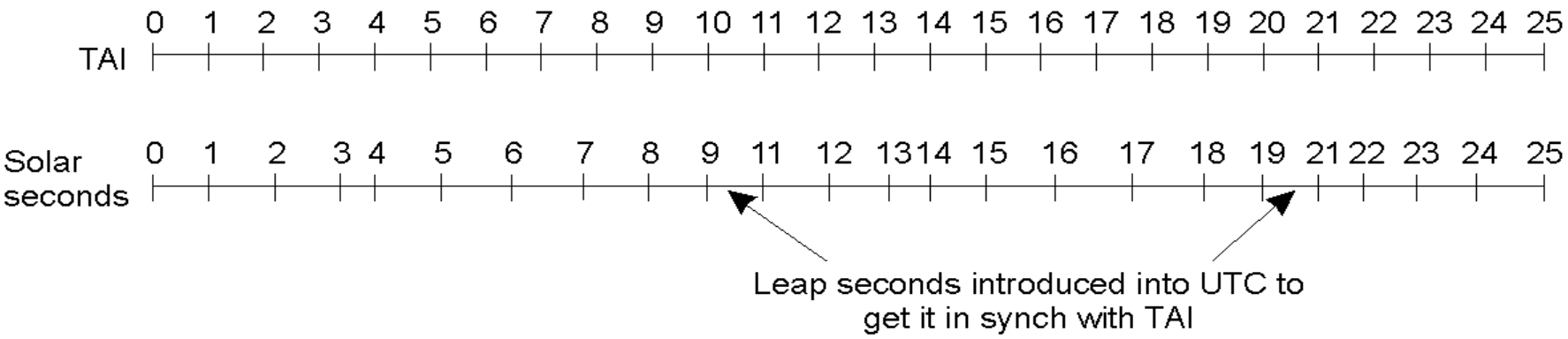
Ha mindegyik gép a saját óráját használja, akkor az adott esemény után történt másik eseményhez az elsőnél korábbi idő társulhat.

Fizikai órák (1)



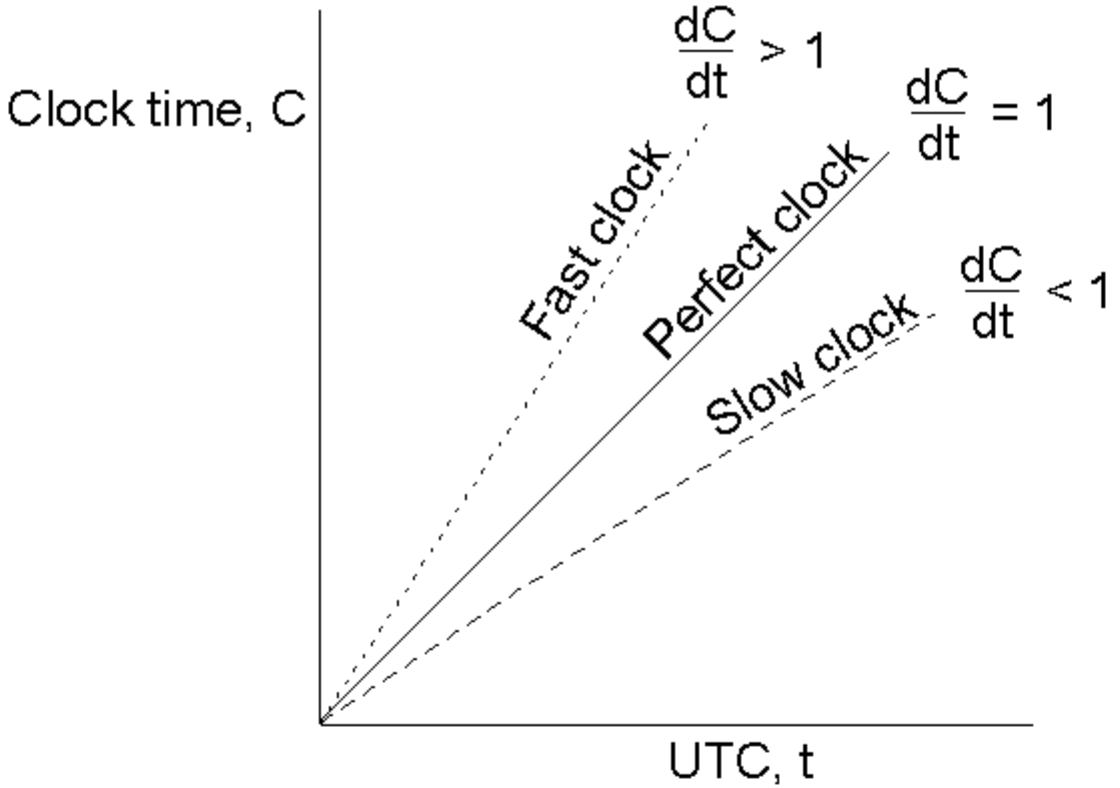
A csillagidő és a szoláris idő összefüggése.

Fizikai órák (2)



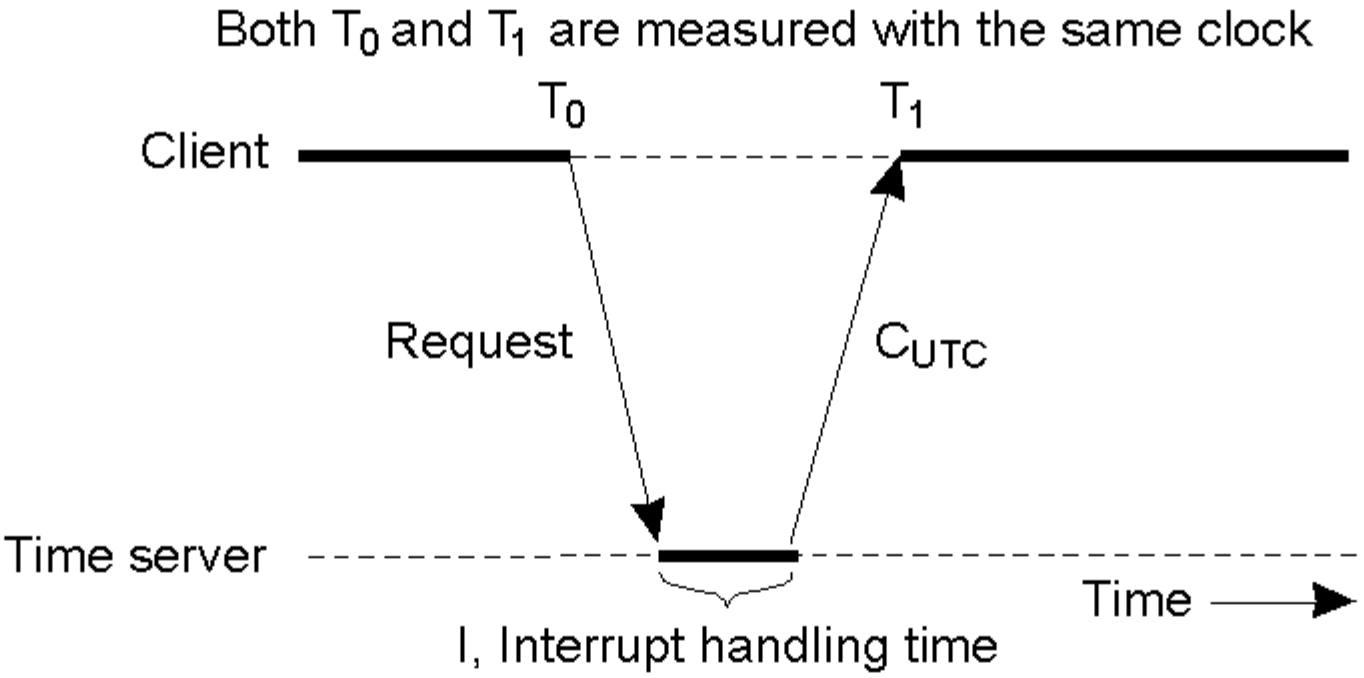
A TAI-másodperc a szolárissal ellentétben azonos hosszúságú. Szökőmásodperceket kell használnunk arra, hogy a TAI a Nappal szinkronban tartható legyen.

Óraszinkronizáló algoritmusok



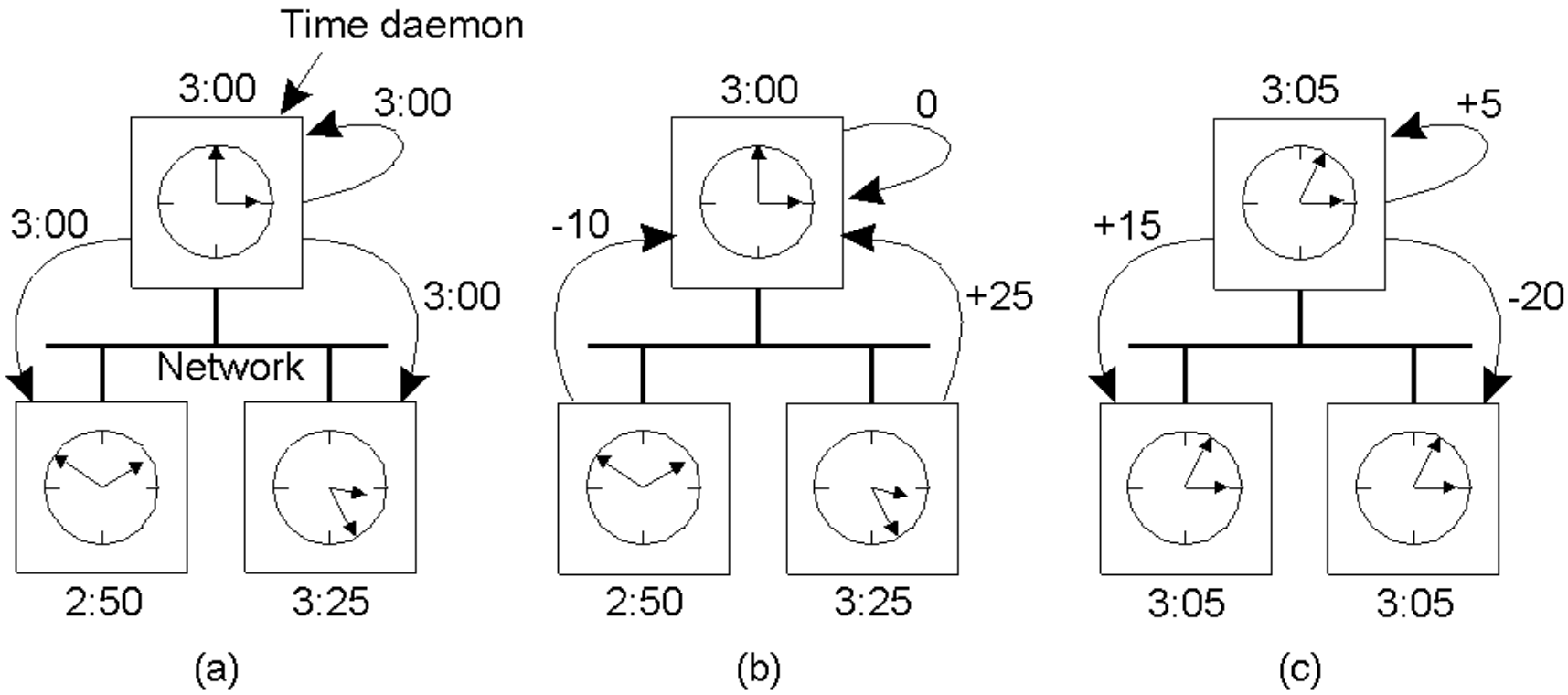
Az óra által mutatott idő és az UTC összefüggése az óra eltérése esetén.

Cristian algoritmusa



A pontos idő lekérdezése az időszerverről.

A Berkeley-algoritmus



a) Az időmon lekérdezi a többi géptől az általuk nyilvántartott pontos időt

b) A gépek válaszolnak

c) Az időmon mindegyiket utasítja, hogyan állítsa be az óráját

Átlagoló algoritmusok

- Decentralizált algoritmus
- Újraszinkronizálási időszakok
  - minden gép kihirdeti a saját idejét
  - elindít egy időmérőt
  - a beérkezett hírdetéseket összegyűjti

Többszörözött külső időforrások

- Több UTC-forrás vevő a rendszerben
- Ezek rendszeresen kihirdetik az idejüket

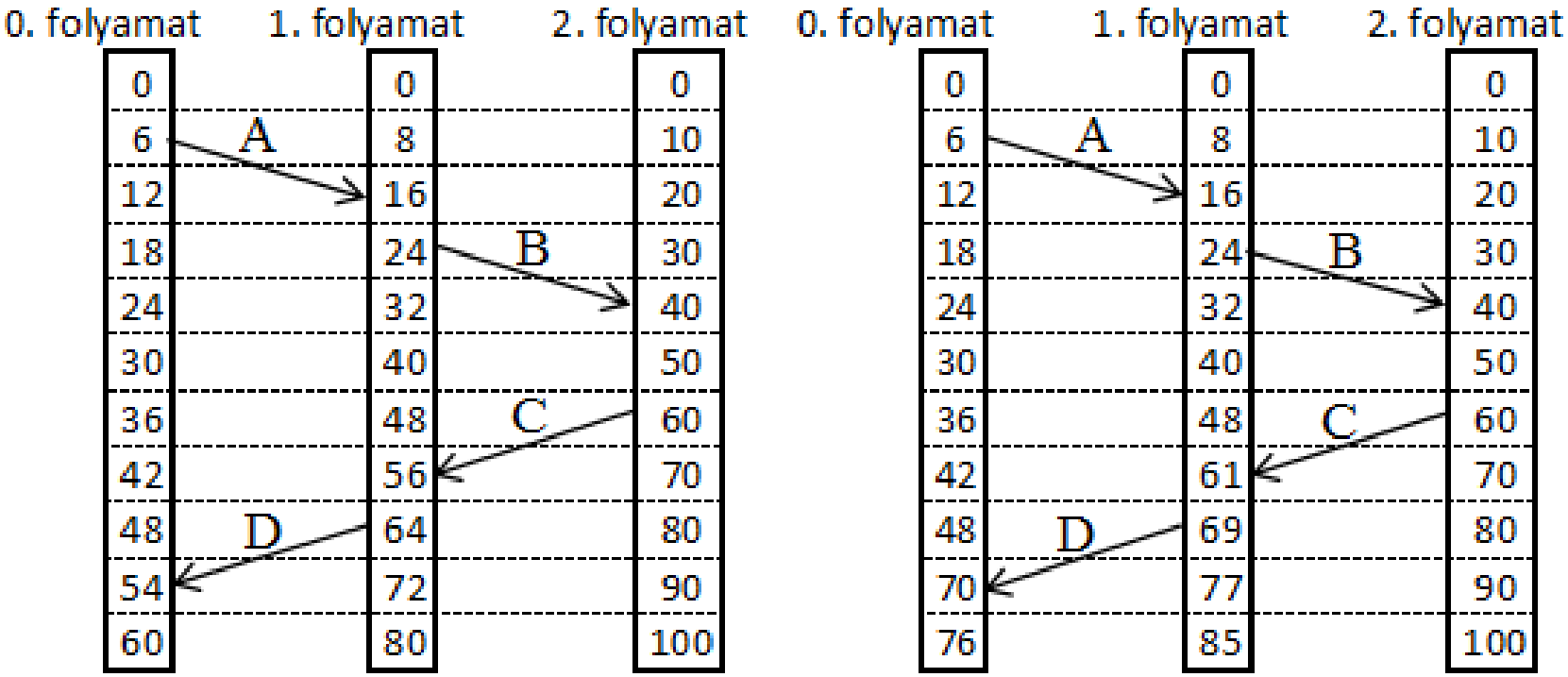
Szinkronizált órák használata

- Többször küldött üzenetek figyelmen kívül hagyására
  - hogy ne kelljen a végtelenségig tárolni

Logikai órák

- Elegendő, ha az összes gép órája ugyanazt az időt mutatja
- Az események sorrendje a fontos

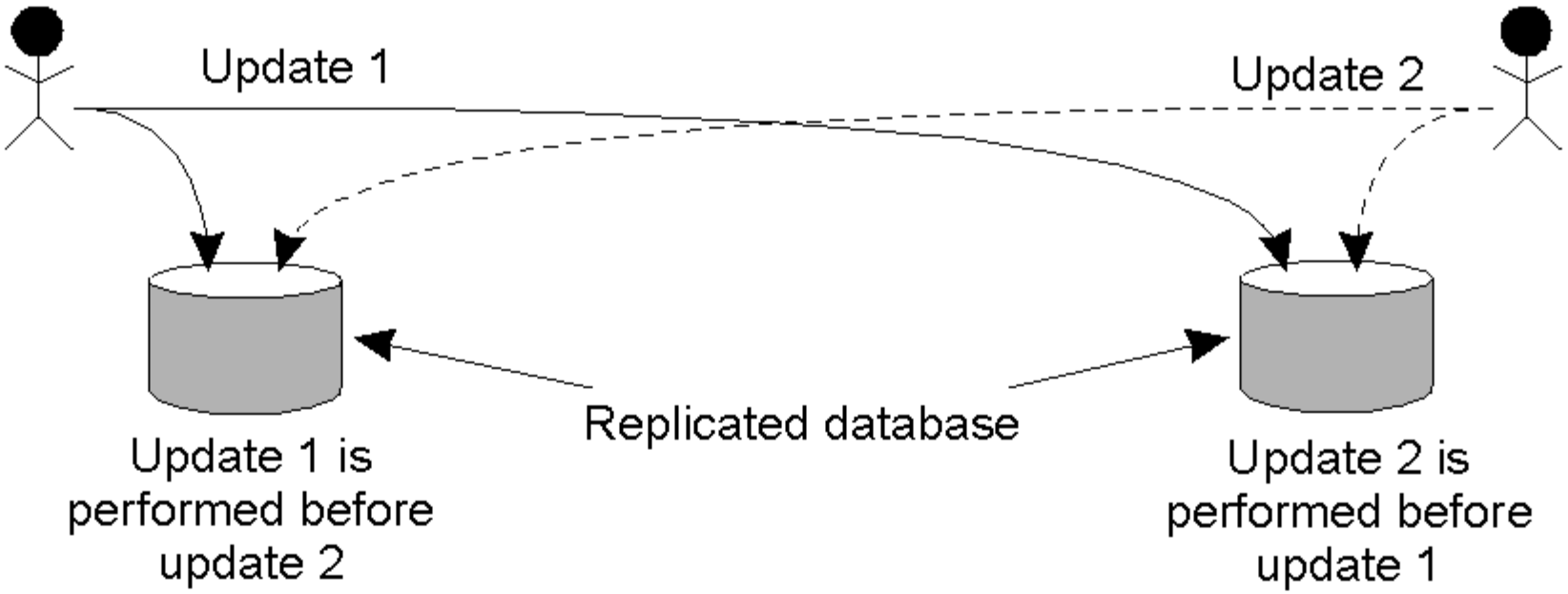
Lamport időbélyege



a). Három folyamat, mindegyik saját órával. Az órák különböző sebességgel futnak.

b). Lamport algoritmusa az órák összehangolására.

Példa – pontosan sorbarendezett csoportcímzés



A többszörözött adatbázis frissítése és az inkonzisztens állapot kialakulása.

Időbélyeg vektor (1)

- ‘C(a)’ és ‘C(b)’ összehasonlítása semmit sem mond ‘a’ és ‘b’ kapcsolatáról
- Lamport időbélyegei nem foglalkoznak az oksági viszonyokkal

- Oksági viszony – időbélyeg-vektorok alkalmazása
- Ha ‘VT(a)’ < ‘VT(b)’ => ‘a’ ‘b’ előzménye

Időbélyeg vektor (2)

- P_i folyamat V_i vektorának tulajdonságai:
  1. $V_i[i]$ a P_i folyamatban eddig bekövetkezett események száma
  2. Ha $V_i[j] = k$, akkor P_i tudja, hogy a P_j folyamatban eddig k esemény történt

- Okozatilag összefüggő üzenetek küldése
- r csak akkor kézbesíti az üzenetet, ha:
  1. $vt(r)[j] = V_k[j] + 1$
  2. $vt(r)[i] <= V_k[i]$ teljesül valamennyi i<>j