Operációs rendszerek

Holtpont

Holtpont (deadlock) fogalma

- A folyamatok egy csoportja olyan eseményre vár, amelyet egy másik, ugyancsak várakozó folyamat tud előidézni.
- Esemény:
  - tipikusan erőforrás felszabadulása.

Kiéheztetés és a holtpont különböző fogalmak!

Rendszer modell I.

- Véges számú és típusú erőforrást kell felosztani az ezekre igényt tartó folyamatok között.
- Az erőforrások osztályokba (típusokba) sorolhatók, melyek az osztályon belül csereszabatosak.
- Erőforrás osztályok (típusok):
  - egyetlen erőforráspéldány van – egypéldányos erőforrások (pl.: rajzgép, nyomtató),
  - több példány áll rendelkezésre – többpéldányos erőforrások: használati értékükben azonosak (pl.: memória),
  - az azonos osztályokba tartozó erőforrások közül egy igénylő folyamat bármelyiket igénybe veheti.

Rendszer modell I.

- Az erőforrások használati módja lehet:
  - osztottan használható erőforrások:
    - állapota menthető és visszaállítható (preemtable), és így elvehetők egy folyamattól, és később visszaadhatók úgy, hogy a folyamat zökkenőmentesen folytatódhasson - ütemezéssel, látszólagos párhuzamos használat szimulálható (pl.: CPU, memória),
  - kizárólagosan használható erőforrások:
    - állapota nem menthető (non-preemptable),
    - nem vehetők el a folyamattól anélkül, hogy a folyamatot visszaküldenénk egy korábbi állapotába - ezen erőforrások (pl.: nyomtató) egészen addig tartoznak egy folyamathoz, amíg az le nem mond róluk.
- A rendszer állapotának leírása:
  - erőforrás-használati gráffal (resource allocation graph).

Holtpont példa

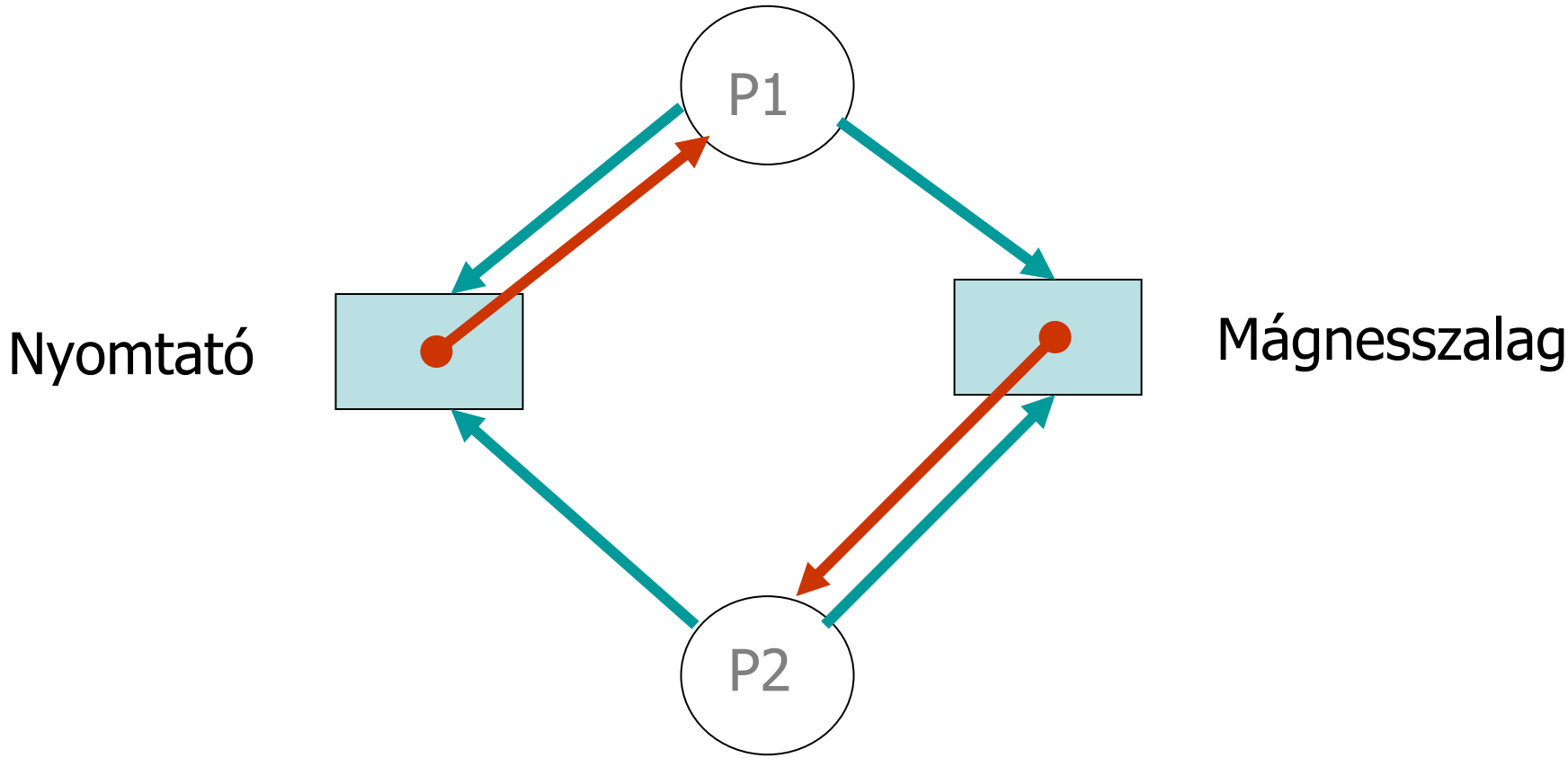
P1 folyamat

```
...
Lefoglal(M)
<mágnesszalag használata>
Lefoglal(NY)
<nyomtató és mágnesszalag együttes
 használata>
Felszabadít(M)
<nyomtató használata>
Felszabadít(NY)
...
```

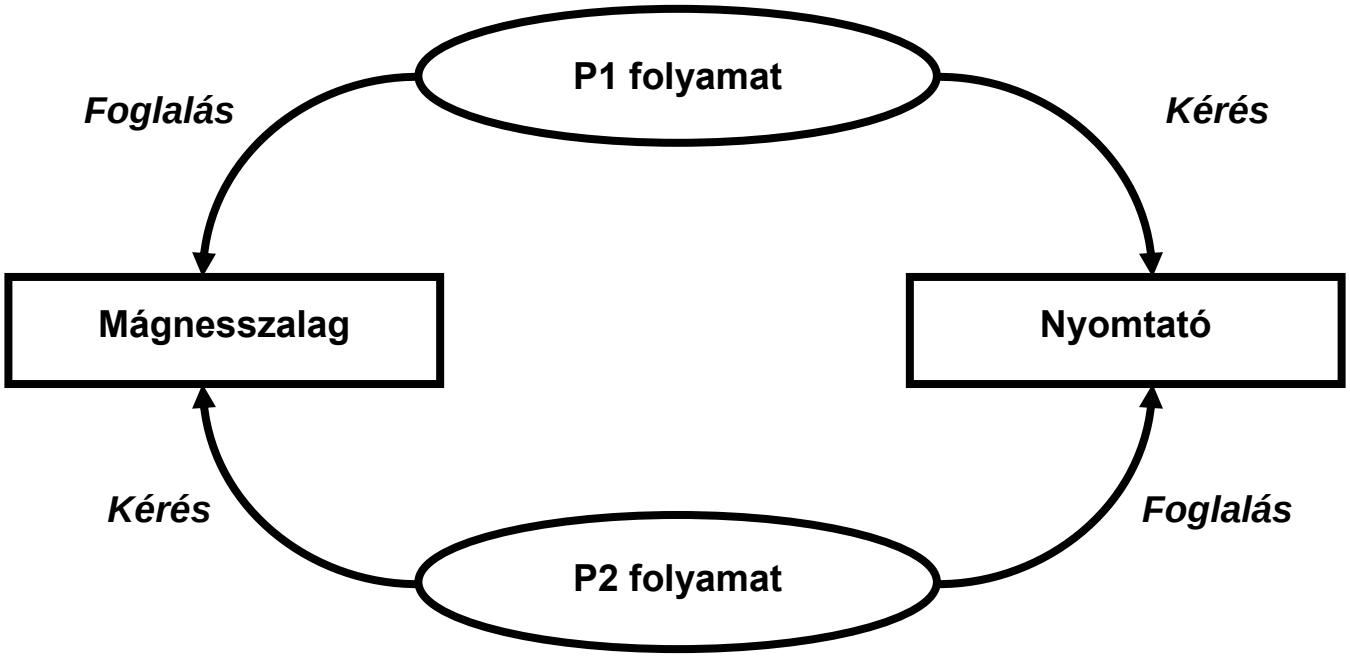
P2 folyamat

```
...
Lefoglal(NY)
<nyomtató használata>
Lefoglal(M)
<nyomtató és mágnesszalag együttes
 használata>
Felszabadít(NY)
<mágnesszalag használata>
Felszabadít(M)
...
```

Holtpont kialakulása



Egyszerű holtpont szituáció



Más esetek – nincs holtpont

Nem alakul ki holtpont:

– ha a folyamatok úgy futnak le, hogy valamelyik folyamatnak sikerül mindkét erőforrást lefoglalni, és ezeket előbb utóbb fel is szabadítja (pl., ha a folyamatok egyetlen művelettel kérnék el mindkét erőforrást),

– ha mindkét folyamat azonos sorrendben kérné el a két erőforrást.

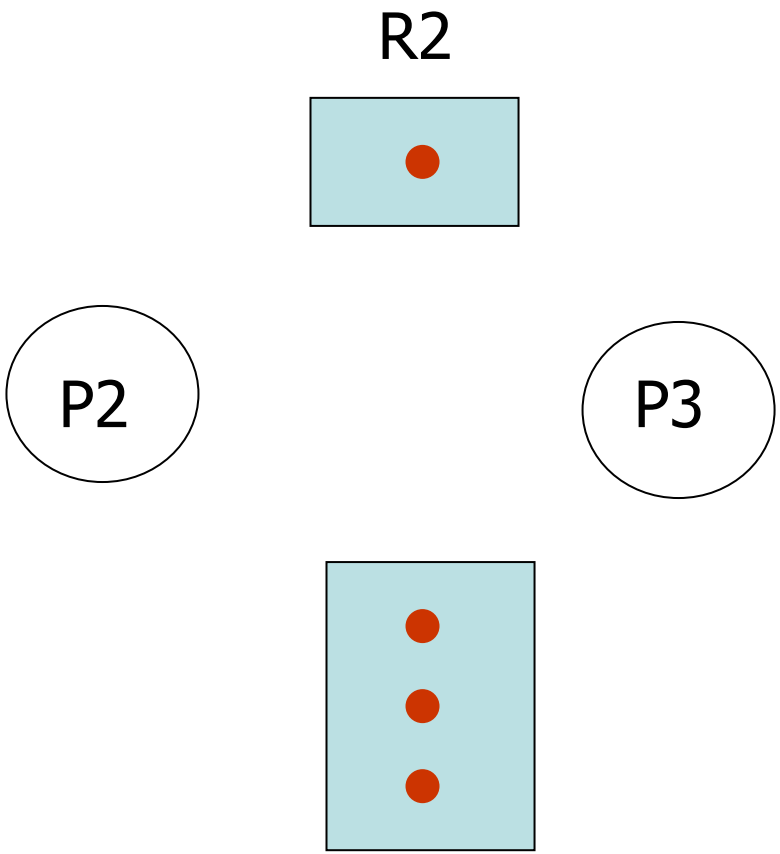
A holtpont kialakulásának valószínűsége annál nagyobb, minél hosszabb a folyamatok azon szakasza, amikor még csak az egyik erőforrást birtokolják.

- Az esemény definíció általános, ugyanis esemény nemcsak erőforrás felszabadulása lehet, hanem tetszőleges más valami is, amire egy folyamat várakozni tud.
- A rendszerben lehetnek futó, élő folyamatok a holtpontban levők mellett, tehát nem biztos, hogy a befagyás teljes.
- Nem biztos, hogy a holtpont a folyamatok minden együttfutásakor kialakul, az esetek jelentős részében igen kis valószínűséggel alakul ki. A jelenség tehát nehezen reprodukálható, alattomos hibaforrás.
- A holtpont egyrészt azon funkciók kiesését okozza, amelyeket a befagyott folyamatok látnak el, másrészt csökkenti a rendszer teljesítőképességét, hiszen a befagyott folyamatok által lekötött erőforrásokhoz a többi folyamat sem tud hozzájutni.

Erőforrás-használati gráf

- Erőforrások.
- Csereszabatos erőforrások csoportja.
- Folyamatok.
- Erőforrás igénylés.
- Erőforrás foglalás.

Erőforrás-használati gráf



Erőforrás-használati gráf

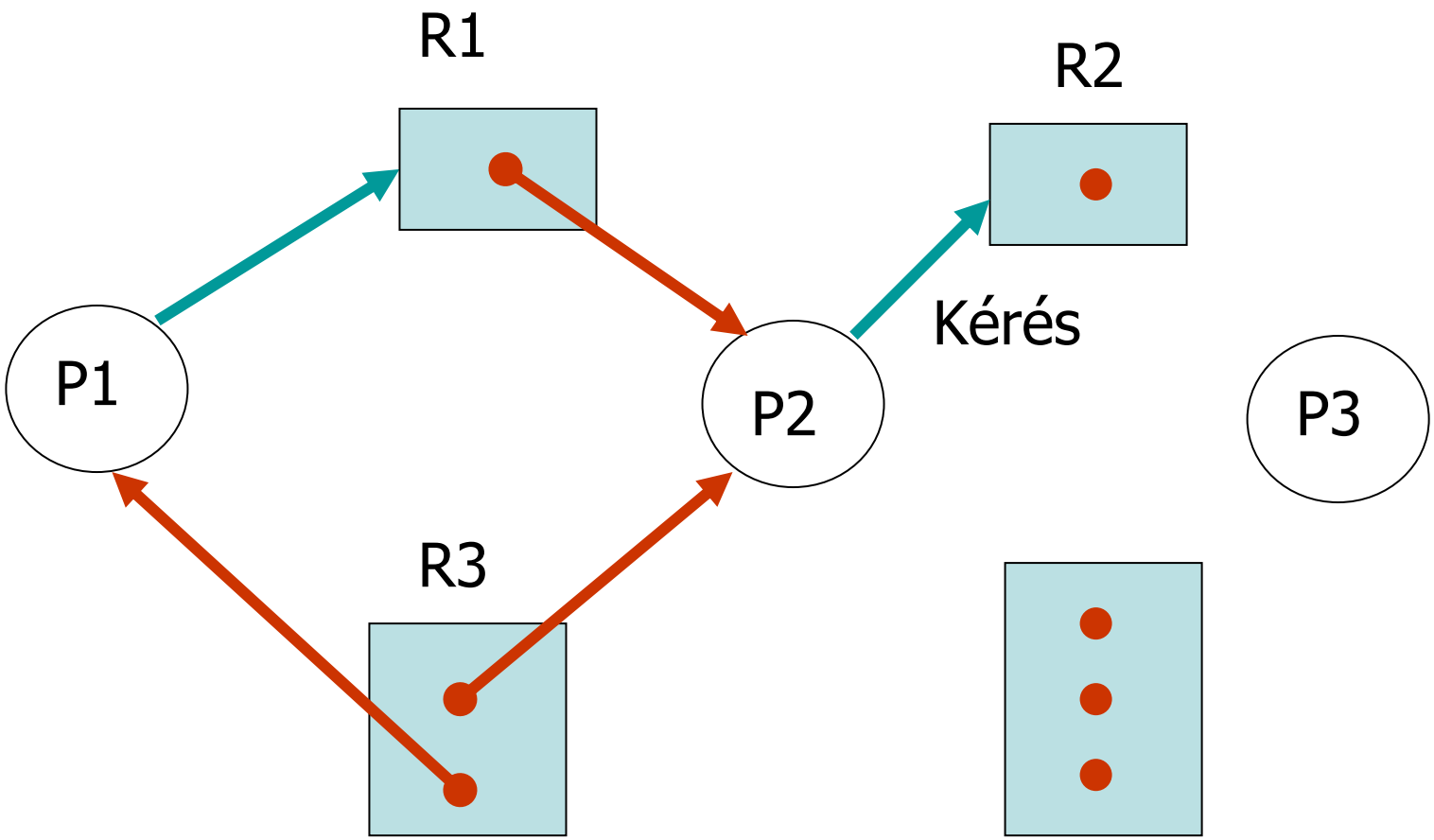
Két csomópont:

- a Pi csomópontok konkrét folyamatokat jelölnek,
- az Ri csomópontok erőforrás osztályokhoz tartoznak.

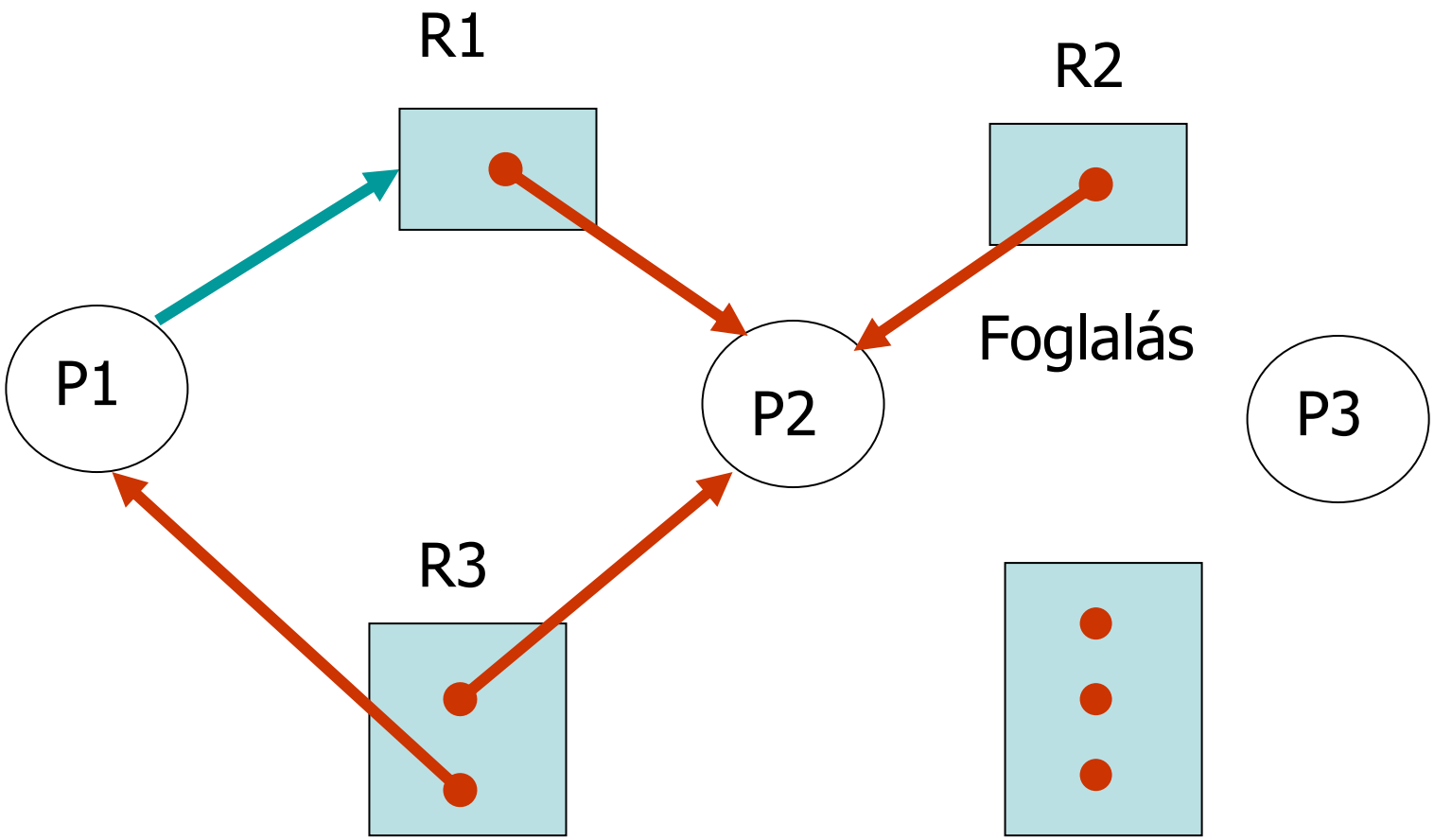
Két él:

- kérés él: a Pi → Rj átmenetek erőforrás igénylést jelölnek: Pi már kérte Rj-t, de még nem kapta meg,
- foglalás él: az Ri → Pj átmenetek erőforrás használatot kódolnak: Pj birtokolja Ri-t, megkapta, de még nem szabadította fel.

Erőforrás-használati gráf



Erőforrás-használati gráf

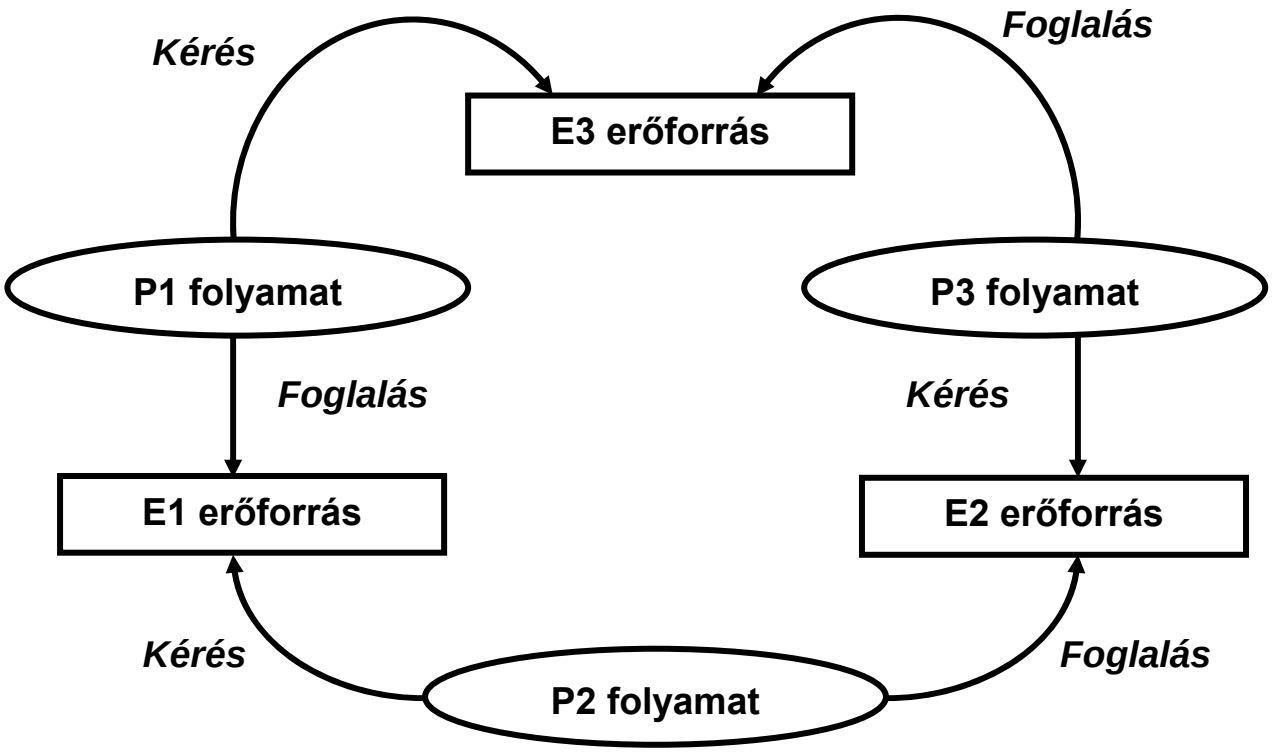


Erőforrás-használati gráf

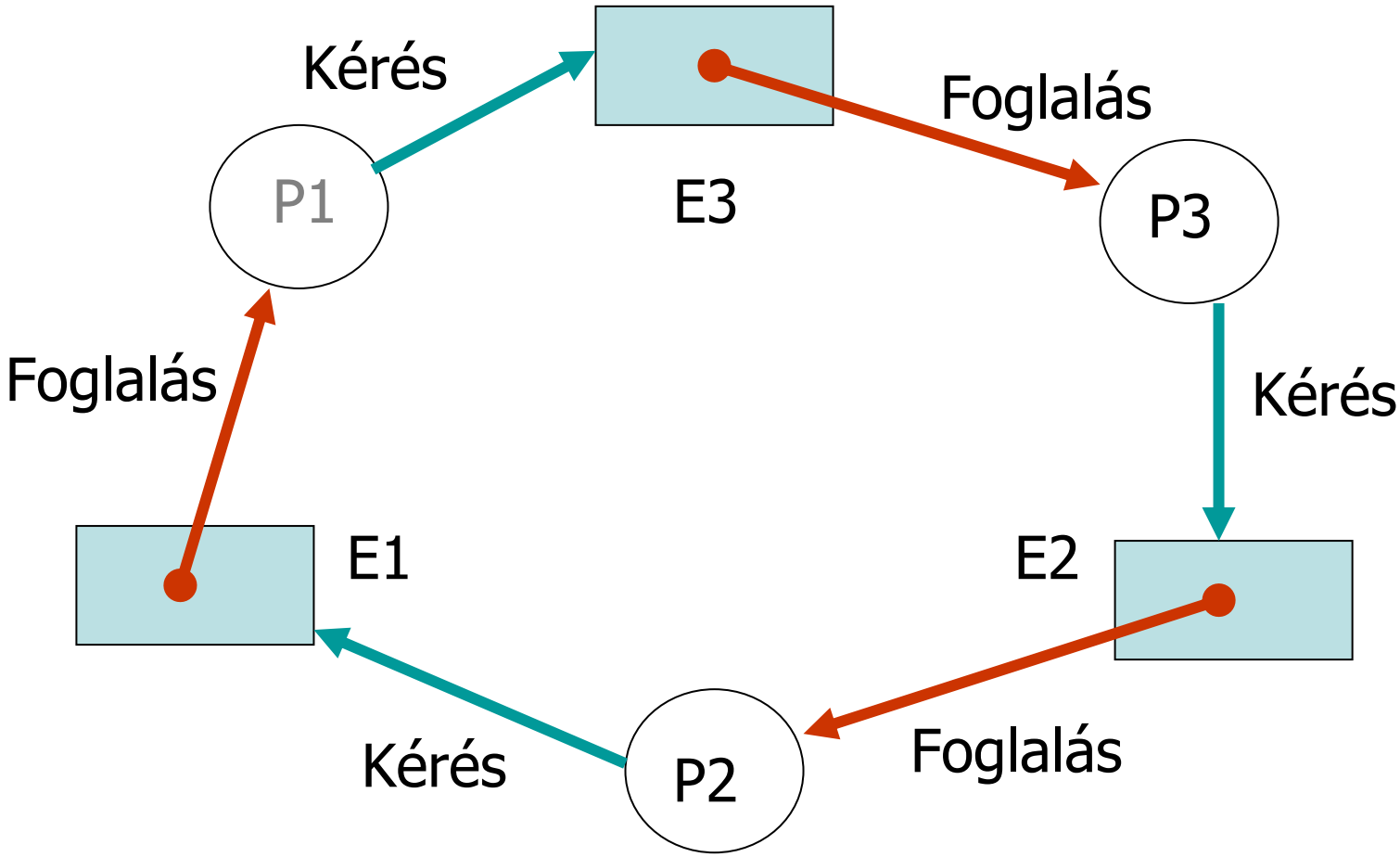
Többpéldányos erőforrások jelölése:

- a példányok megfelelő számú ponttal jelezve a téglalapon belül,
- a folyamat konkrét példányt birtokol, de típust kér:
  - a foglalási él a konkrét példányt jelképző pontból indítva,
  - kérési él csak a téglalap határáig vezetve jelölve.

Holtpont három folyamattal



Erőforrás-használati gráf



Rendszer modell II.

Erőforrások használatba vételének lépései:

- Igénylés (rendszerhívással):
  - ha az igény nem teljesíthető, a folyamat várakozik.
- Az erőforrás kizárólagos használata:
  - a folyamat kizárólagosan használja az adott erőforrást.
- Az erőforrás felszabadítása:
  - az adott folyamat felszabadítja az erőforrást, amelyet ezután igénybe vehet egy másik, arra váró folyamat,
  - valamelyik várakozó továbbengedése.

A holtpont kialakulásának feltételei I.

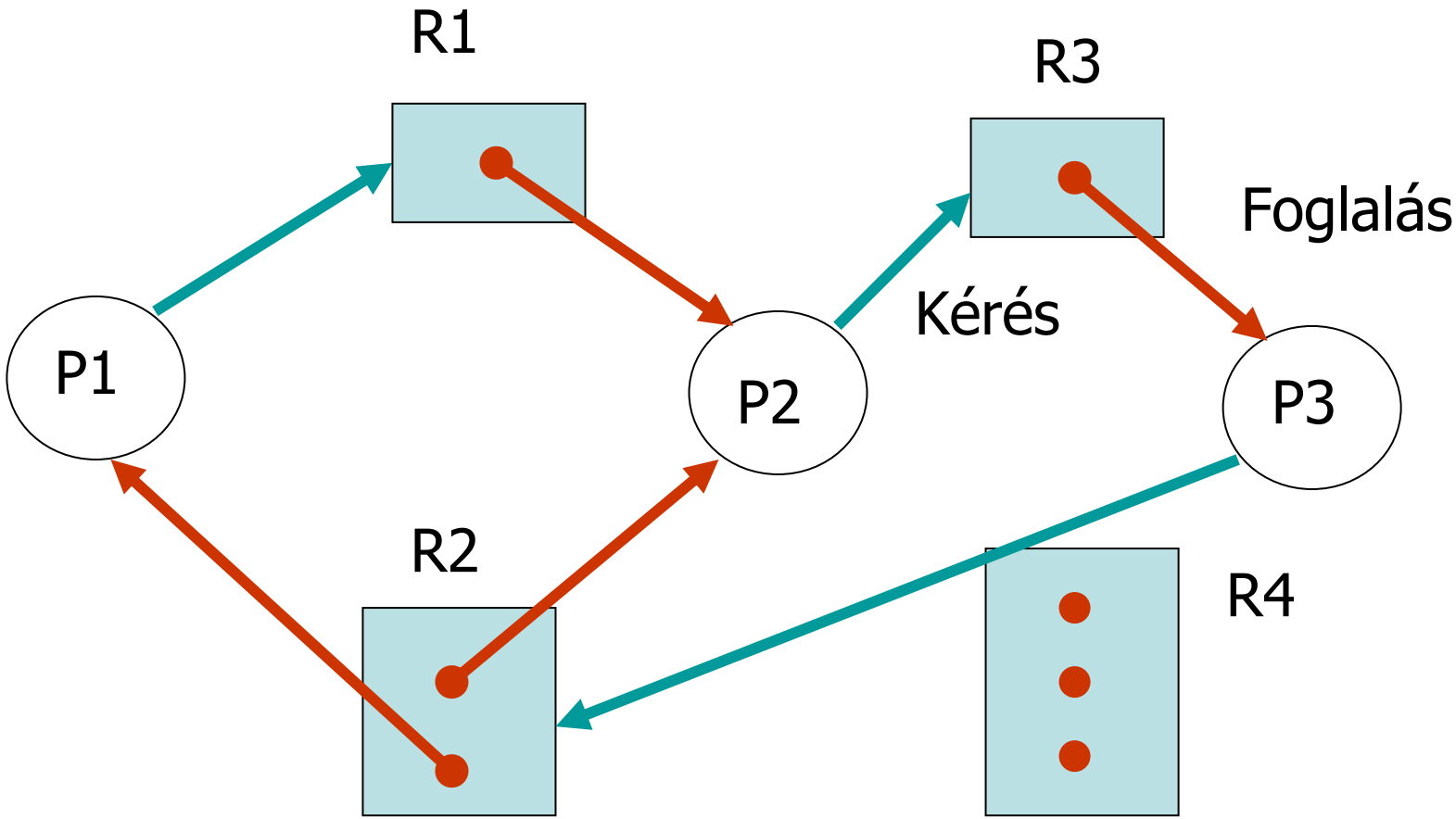
A holtpont kialakulásának *szükséges* feltételei:

- kölcsönös kizárás (erőforrás használat),
- foglalva várakozás,
- nem elvehető erőforrások,
- körkörös várakozás.

A feltételeknek egyszerre kell teljesülniük, vagyis ha legalább az egyik feltétel nem teljesül nem alakulhat ki holtpont!!!

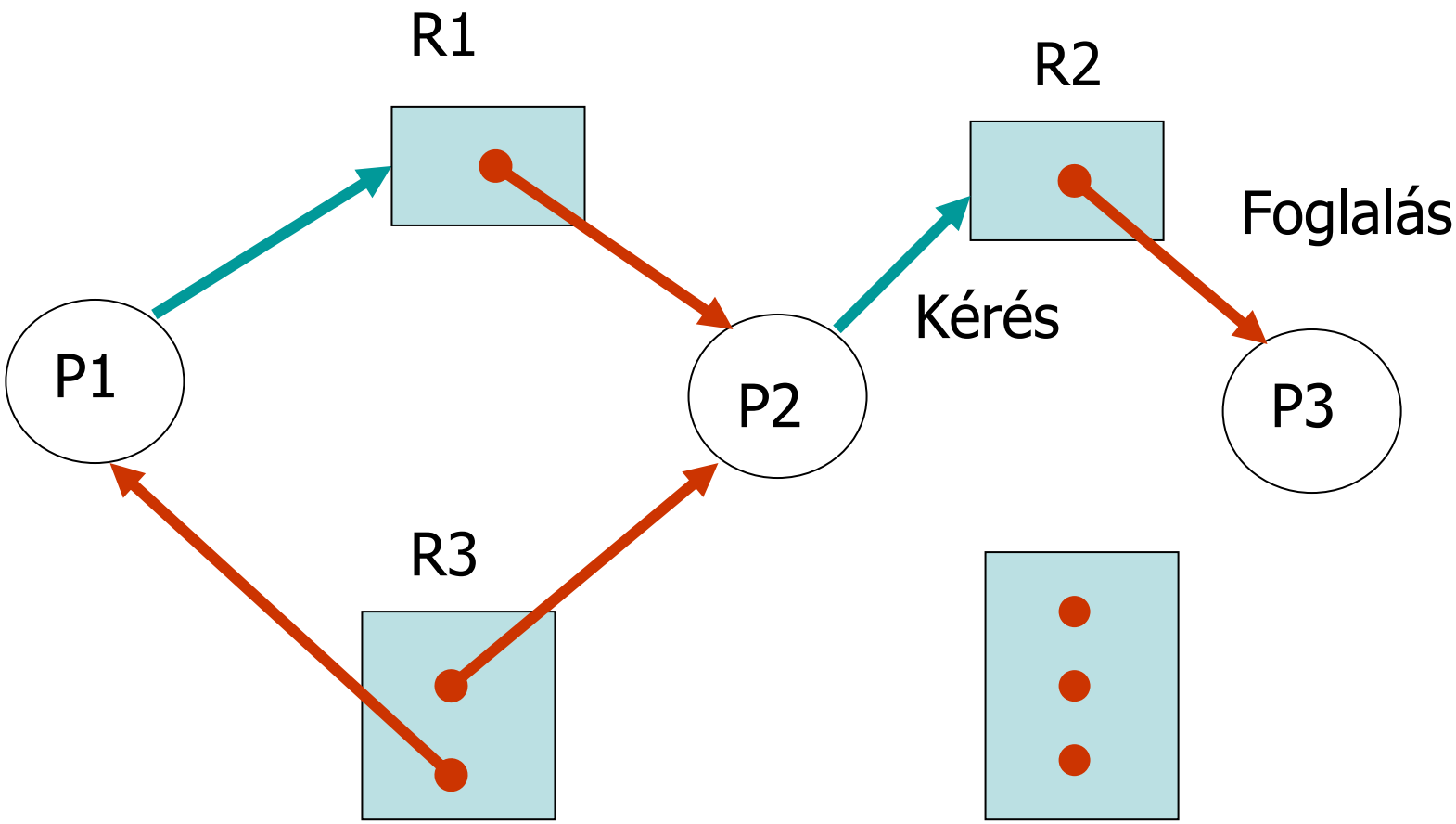
- Kölcsönös kizárás: legyenek olyan erőforrások a rendszerben, amelyeket a folyamatok csak kizárólagosan használhatnak.
- Foglalva várakozás: legyen olyan folyamat, amelyik lefoglalva tart erőforrásokat, miközben más erőforrásokra várakozik.
- Nem elvehető erőforrások: nincs erőszakos erőforrás-elvétel, vagyis minden folyamat addig birtokolja a megkapott erőforrásokat, amíg saját maga fel nem szabadítja azokat.
- Körkörös várakozás: a folyamatok közül kiválasztható olyan {P0, P1,...,Pn} folyamat részhalmaz mint rendezett sorozat, amelyben P0 egy P1 által lefoglalt, P1 egy P2 által lefoglalt,...., Pn egy P0 által lefoglalt erőforrásra vár.

Holtpont

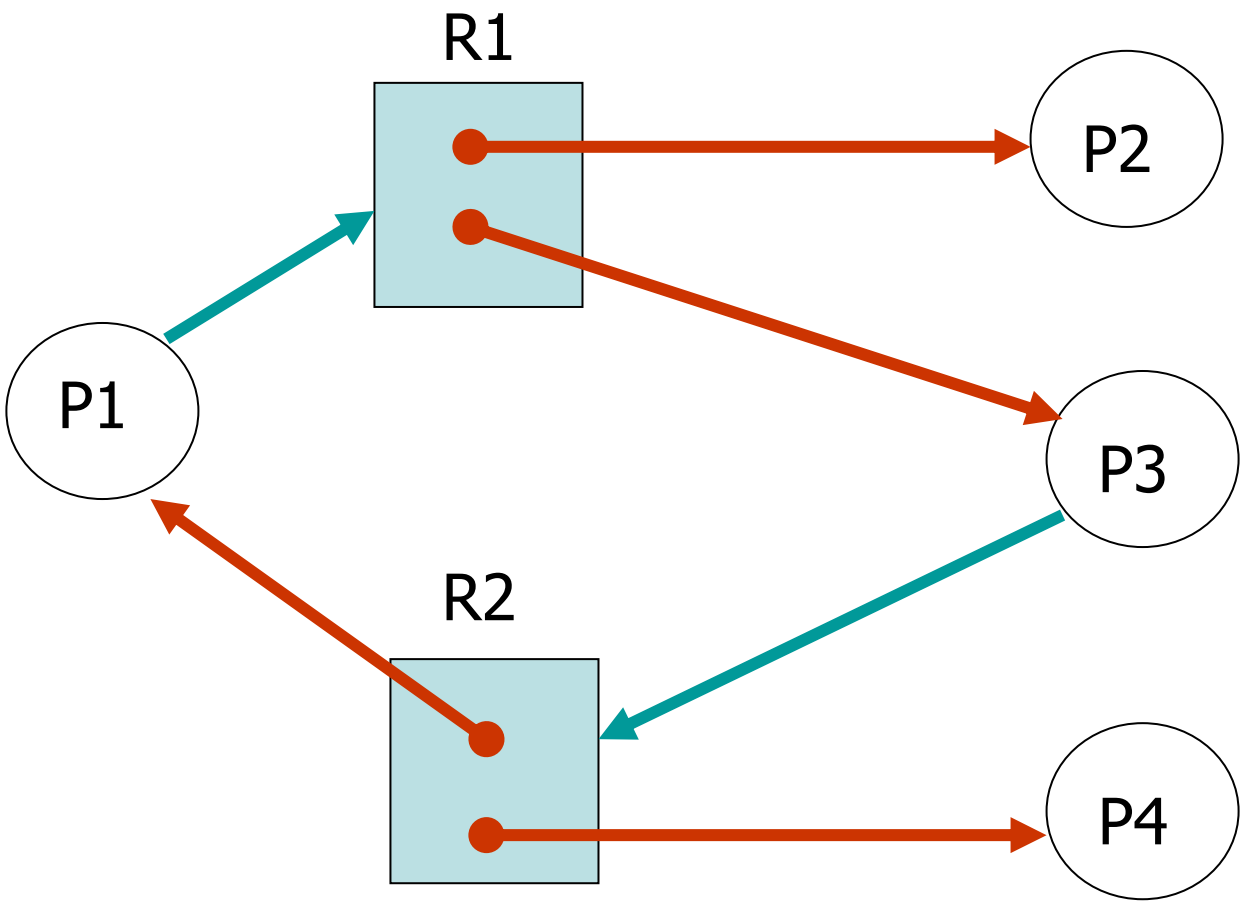


Kettő hurok: P1-R1-P2-R3-P3-R2-P1 és P2-R3-P3-R2-P2

Nincs holtpont



Nincs holtpont



A holtpont kialakulásának feltételei II.

- A négy szükséges feltétel egyben elégséges feltétel is, ha minden erőforrás osztályban csak egyetlen erőforrás van.
- Általában az erőforrás-használati gráfban a hurok szükséges feltétele a holtpontnak. Amennyiben valamennyi erőforrás osztály csak egyetlen erőforrást tartalmaz, a hurok elégséges feltétel is egyben.
- Ha egy EFG nem tartalmaz hurkot, akkor a rendszer nem lehet holtponti állapotban.
- Ha legalább egy hurkot tartalmaz, akkor a rendszer lehet, hogy holtponti állapotban van, de az is lehet, hogy nincs.

Holtpont kezelési stratégiák

- Erőforrás használati szabályokkal biztosítani, hogy holtpont ne alakuljon ki:
  - holtpont megelőzés (deadlock prevention), ez statikus védekezés,
  - holtpont elkerülés (deadlock avoidance), ez dinamikus védekezés.
- Csak a holtpont kialakulásánál avatkozunk be:
  - holtpont felismerés (deadlock recognition),
  - holtpont felszámolása (deadlock recovery).

A holtpont megelőzése I.

A holtpont kialakulásának valamelyik szükséges feltételének kizárása:

- A kölcsönös kizárást nem lehet kiküszöbölni:
  - nem megosztható erőforrások (printer) esetében,
  - a megosztható erőforrások nem követelik meg a kizárást, így nem lehet közük a holtponthoz (read-only file-okhoz – egyidejűleg, várakozás nélkül tudnak hozzáférni a folyamatok).

A holtpont megelőzése II.

A holtpont kialakulásának valamelyik szükséges feltételének kizárása:

- *a* foglalva várakozás kizárása:
  egy folyamat csak akkor kérhet új erőforrást, ha nem tart lefoglalva másikat. Alternatívák:
  - futás elején lefoglalja az összes erőforrást, s csak akkor futhat, ha valamennyi a rendelkezésére áll,
  - erőforrás-foglalás előtt a foglalt erőforrásokat felszabadítja, vagyis a folyamat csak akkor igényelhet ki egy erőforrást, ha már más erőforrást nem használ.

Hátrányok: nagy a kiéhezés veszélye, az erőforrások kihasználtsága rossz.

A holtpont megelőzése III.

Nem elvehető erőforrások kezelése:

- erőforrások elvétele egyes folyamatoktól,
- ha egy folyamatnak van nem kielégíthető erőforrás igénye, az OPR elveheti tőle az összes igényelt erőforrást, vagy
- ha egy folyamatnak olyan erőforrásra van szüksége, amely pillanatnyilag foglalt, az OPR megnézi, hogy ez a foglaltság egy szintén várakozó folyamattól ered-e. Ha igen, az erőforrás e várakozótól vétetik el az éppen igénylő javára. Ha nincs ilyen várakozó folyamat, egyszerűen az igénylő folyamat is várakozni kezd.

Hátrányok: az erőforrások elvétele egy folyamattól sokszor azzal jár, hogy annak futási eredményei is elvesznek. Itt is előfordulhat a kiéhezés.

A holtpont megelőzése IV.

Körkörös várakozás kizárása:

- erőforrások megszámozása, a rendszer összes erőforrása megszámozható egy növekvő számsorozattal,
- erőforrások csak sorrendben igényelhetők.

Ekkor tehát:

- biztosítjuk, hogy a folyamatok csak növekvő sorrendben igényelnek erőforrásokat, vagy
- vagy biztosítjuk, hogy egy folyamat csak akkor igényelhet ki egy erőforrást, ha nem használ egy annál magasabb sorszámút.

Hátrány: a sorszámozás általában önkényes lesz, s nem felel meg a változékony gyakorlati igényeknek.

A holtpont elkerülése

- Dinamikus védekezés: a rendszer a foglalási-kérési szekvenciák elemzése alapján dönt, hogy mely folyamatoknak adjon erőforrást, illetve melyeket késztessen várakozásra.
- Az erőforrások óvatos allokálásával elkerülhető a holtpont kialakulása.
- Megvalósítás feltétele:
  a folyamatok erőforrás-igényéről kiegészítő információval kell rendelkezni:
  - a folyamatok erőforrás-osztályonkénti maximális igénye.

Más szóval: a holtpont kialakulásának feltételei megmaradnak ugyan, de az erőforrások allokálása "óvatosan történik". Ez a módszer akkor használható, ha:

- biztos, hogy valamennyi olyan folyamat, amelynek erőforrás igényei maximálisan ki vannak elégítve, véges idő alatt lefut, s ezután felszabadulnak az általa lefoglalt erőforrások,
- valamennyi folyamatról előre ismert, hogy milyen erőforrás típusból maximálisan mennyit igényel egyidejűleg lefoglalva.

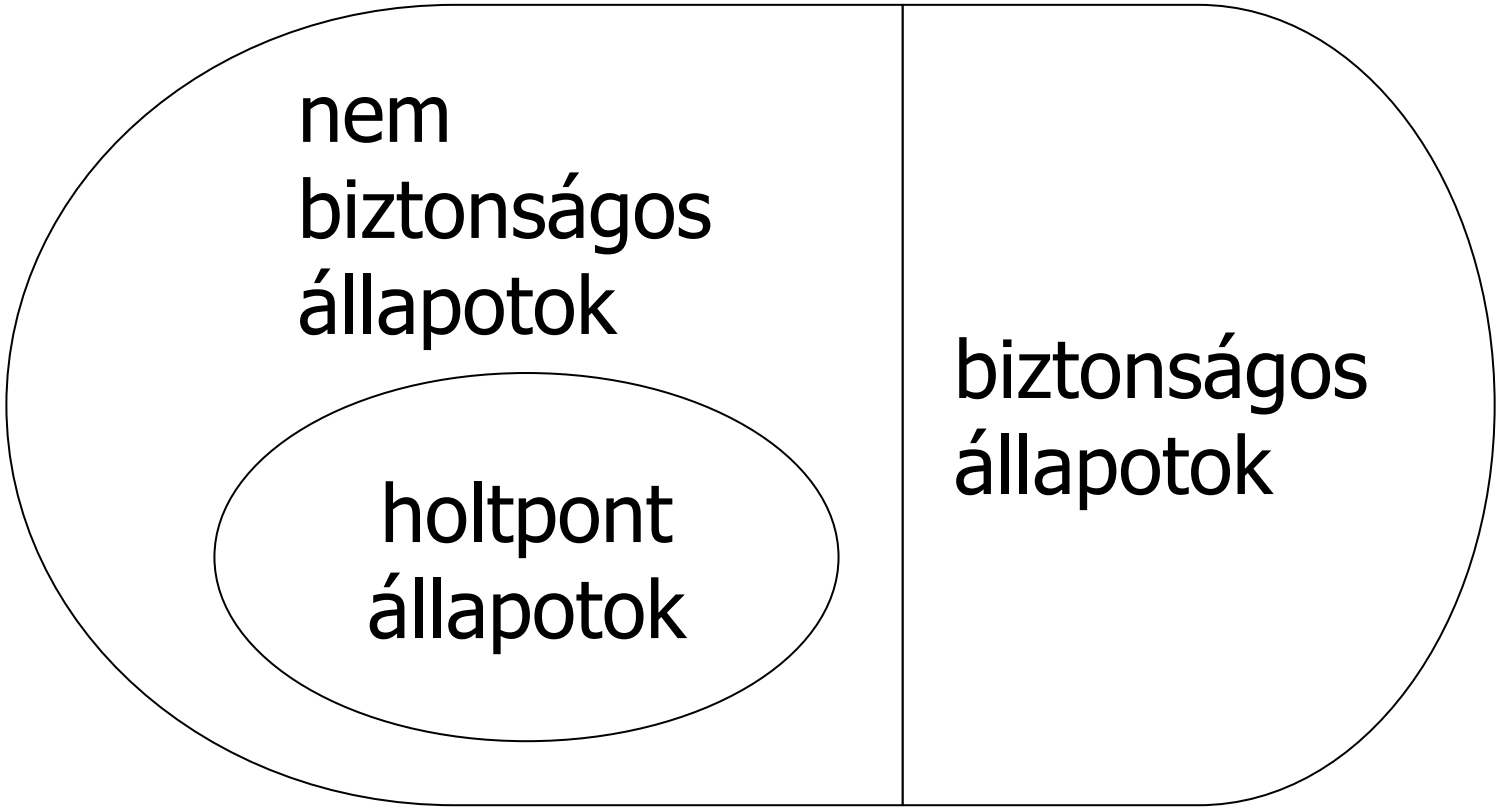
Rendszer modell

- Feltételezés:
  A folyamat erőforrás igényének kielégítése után véges időn belül visszaadja az összes erőforrást.

A holtpont elkerülésének módszere

- Az erőforrás igényt csak akkor teljesítjük, ha az így kialakult rendszer biztonságos állapotban marad.
- Biztonságos állapot:
  Található olyan folyamat végrehajtási sorrend, hogy az aktuálisan futó folyamat maximális igénye kielégíthető.
- Megvalósítás:
  Bankár algoritmus.

Biztonságos és holtpont állapotok viszonya



- Biztonságos állapotban nem alakul ki holtpont.
- Egy nem biztonságos állapot még nem garantálja holtpont kialakulását. A holtpontra vezető állapotok csupán a nem biztonságos állapotok halmazának részhalmazai.
- Egy stratégiai lehetőség a holtpont elkerülésére, ha csak biztonságos állapot létrejöttét engedjük meg (banker algorithm).

Holtpont elkerülése

- Bankár algoritmus:
  - csak azokat a folyamatokat engedi futni, amelyek erőforrásigénye kielégíthető, ha az összes erőforrást igényli amire szüksége van.

Bankár algoritmus

- A bankok működésének az analógiájára épül.
- A bankvilágban egy bank soha ne kölcsönözhesse ki az összes pénzkészletét.
- Amikor egy új folyamat lép a rendszerbe, annak meg kell adni minden erőforrástípusból az általa maximálisan kérhető darabszámot, ami nem lehet több, mint a rendszer erőforrásainak a teljes száma.
- Amikor a folyamat kéri a konkrét erőforrások egy halmazát, akkor az OPR meghatározza, hogy a kért erőforrások allokálása biztonságos állapotban hagyja-e a rendszert:
  - ha igen, akkor a folyamat megkapja az erőforrásokat,
  - ha nem, várakozni kényszerül.
- Nem preemptív: egyik folyamat sem szoríthatja ki a másikat

A holtpont felismerése I.

- Az erőforrás-használati gráf elemzése alapján.
- Gráf redukciós algoritmus:
  - Minden olyan folyamat kiválasztása, amelynek a jelenlegi igényei a szabad erőforrásokkal kielégíthetők.
  - A folyamat által lefoglalt erőforrásokat visszaadjuk (optimista algoritmus).
  - Újra keresünk kielégíthető folyamatot, ha nincs ilyen, de maradtak kielégítetlen folyamatok, akkor holtpont van.

A holtpont felismerése II.

- Hátrány:
  - az algoritmus viszonylag lassú.
- Mikor fusson?
  - Minden erőforrás igény teljesítésekor.
  - Meghatározott időközönként.

A holtpont felszámolása I.

- Folyamatok megszűntetésével:
  - Minden holtpontban lévő folyamatot megszűntetünk.
  - Csak néhány folyamatot szűntetünk meg.
  - A megszüntetendő folyamat kiválasztásához paraméterek pl.:
    - hány holtpont hurokban szerepel,
    - mekkora a prioritása,
    - mennyi ideje fut (várhatóan mennyi ideig futna még),
    - milyen erőforrásokat tart lefoglalva,
    - milyen erőforrásokat igényelne még.

A holtpont felszámolása II.

- Erőforrások elvételével - preemptív jellegű megoldás.
  Probléma: melyik folyamattól és milyen erőforrásokat vegyük el.
- Megvalósítás szempontjai:
  - kiválasztás és a
  - visszaállítás: a futási eredmény megőrzéséhez szükséges ellenőrzési pontok (checkpoint) definiálása és a visszaállítás (rollback) használata.

A holtpont felszámolása - a megvalósítás szempontjai

Kiválasztás:

- Fontos az elvételek sorrendjének meghatározása. Az elvételek a folyamat megszűnését is eredményezhetik.
- Költségtényező: a holtponti folyamat által lekötött erőforrások száma, valamint az általa addig elhasznált futási idő mértéke.

A holtpont felszámolása - a megvalósítás szempontjai

Visszaállítás:

- A folyamattól elvesszük az erőforrást, ilyenkor a folyamat nem tudja tovább fenntartani normál működését.
- Lehetőség: a folyamat menetében visszalépünk egy korábbi biztonságos állapothoz, és újraindítjuk attól az állapottól. A biztonságos állapot megállapítása nehéz, ezért legegyszerűbb a teljes újrakezdés. A folyamatot abortáljuk és újraindítjuk.

A holtpont felszámolása - a megvalósítás szempontjai

Visszaállítás a gyakorlatban:

- a folyamat szabályos időközönként közli az OPR-rel a futás közbeni állapotait, amelyeket a rendszer elment,
- futás közbeni ellenőrzési pontok (checkpoint) létrehozása,
- holtpont esetén az erőforrás elvétel után a rendszer a folyamatot egy alkalmasnak ítélt ellenőrzési ponttól tudja később újraindítani,
- visszaállítás: egy megfelelőnek ítélt ellenőrzési ponthoz való visszatérés és onnan történő újraindítás.

Kombinált holtpont-kezelő stratégiák

- Különböző erőforrás-típusokhoz más-más stratégiát használhatunk, pl.:
  - Belső rendszer-erőforrások (pl. rendszertáblák): sorrendi foglalás.
  - Menthető állapotú erőforrások (pl. központi tár): erőforrás elvétele.
  - Kötegelt erőforrások (pl. file-ok): előre megadott erőforrás szükséglet, holtpont elkerülés.
  - Tárcsere terület a háttértáron: előzetes foglalás.

Megszakítások kezelése

- Megszakítások kezelése: megszakítás kiszolgálásainak lépései.
- Alapvető módszerek:
  - kiszolgáló rutin (nincs környezetváltás),
  - kiszolgáló folyamat (mindig van környezetváltás).
- Köztes megoldás, kiszolgáló rutin, a várakozó folyamatok átütemezésének kezdeményezése.

Bankár algoritmus - példa

Példa 1. - Bankár algoritmus

Aktuális állapot:

Összesen 12 db eroforrás

A és B folyamatok

	Foglal	Max. igény	Várható igény
A	4	6	2
B	4	11	7
Szabad	4		

Várakozó „C” folyamat - BEENGEDHETO ?

C	2	8	6

Tegyük föl, hogy beengedjük...

	Foglal	Max. igény	Várható igény
A	4	6	2
B	4	11	7
C	2	8	6
Szabad	2		

Az „A” folyamat lefuthat, mivel várható igénye nem nagyobb a szabad eroforrások számánál

Az „A” lefutott, foglalt eroforrásai felszabadultak…

	Foglal	Max. igény	Várható igény
B	4	11	7
C	2	8	6
Szabad	6		

Az „C” folyamat lefuthat, mivel várható igénye nem nagyobb a szabad eroforrások számánál

Az „C” lefutott, foglalt eroforrásai felszabadultak...

	Foglal	Max. igény	Várható igény
B	4	11	7
Szabad	8		

Az „B” folyamat lefuthat, mivel várható igénye nem nagyobb a szabad eroforrások számánál

Az állapot a „C” beengedése után is biztonságos

ENGEDJÜK BE !

Példa 2. - Bankár algoritmus

Aktuális állapot:

Összesen 12 db eroforrás

A és B folyamatok

	Foglal	Max. igény	Várható igény
A	4	6	2
B	4	11	7
Szabad	4		

Várakozó „C” folyamat - BEENGEDHETO ?

C	2	9	7

Tegyük föl, hogy beengedjük...

	Foglal	Max. igény	Várható igény
A	4	6	2
B	4	11	7
C	2	9	7
Szabad	2		

Az „A” folyamat lefuthat, mivel várható igénye nem nagyobb a szabad eroforrások számánál

Az „A” lefutott, foglalt eroforrásai felszabadultak…

	Foglal	Max. igény	Várható igény
B	4	11	7
C	2	9	7
Szabad	6		

Az állapot „C” beengedése után

NEM BIZTONSÁGOS

Bankár algoritmus

1. Felírjuk a folyamatok által maximálisan igényelt eroforrások számait tartalmazó mátrixot (MAX. IGÉNY)
2. Felírjuk a folyamatok által lefoglalva tartott eroforrások számait tartalmazó mátrixot (FOGLAL)
3. A MAX. IGÉNYbol FOGLALt kivonva kapjuk a még kielégítetlen igényeket leíró mátrixot (IGÉNY)
4. Eroforrás fajtánként összeadjuk a lefoglalva tartott eroforrások számait, majd ezeket kivonva az egyes eroforrások összdarabszámából kapjuk a pillanatnyilag rendelkezésre álló eroforrás készletet (KÉSZLET)

Bankár algoritmus

5. Megnézzük, hogy a KÉSZLETbol kielégítheto-e valamelyik folyamat igénye
6a. Ha nincs ilyen folyamat, a rendszer NEM BIZTONSÁGOS állapotban van
6b. Ha van ilyen folyamat, kielégítjük igényét
7. A kiválasztott folyamat a lefutása után felszabadítja az összes általa használt eroforrást, azaz KÉSZLET új értékét megkapjuk, ha elozo értékéhez hozzáadjuk a folyamat által eredetileg lefoglalva tartott eroforrások számát (FOGLAL megfelelo sorát)
8. Visszamegyünk az 5. lépésre

Bankár algoritmus több erőforrás esetén

Egy rendszerben az alábbi eroforrások vannak:

E1: 10 darab E2: 5 darab E3: 7 darab

A rendszerben 5 folyamat van: F1, F2, F3, F4, F5

Biztonságos-e holtpontmentesség szempontjából a következo állapot?

	MAX. IGÉNY			FOGLAL		
	E1	E2	E3	E1	E2	E3
F1	7	5	3	0	1	0
F2	3	2	2	3	0	2
F3	9	0	2	3	0	2
F4	2	2	2	2	1	1
F5	4	3	3	0	0	2

Igény meghatározás

[IGÉNY] = [MAX.IGÉNY] - [FOGLAL]

	MAX. IGÉNY		
	E1	E2	E3
F1	7	5	3
F2	3	2	2
F3	9	0	2
F4	2	2	2
F5	4	3	3

	FOGLAL		
	E1	E2	E3
F1	0	1	0
F2	3	0	2
F3	3	0	2
F4	2	1	1
F5	0	0	2

	IGÉNY		
	E1	E2	E3
F1	7	4	3
F2	0	2	0
F3	6	0	0
F4	0	1	1
F5	4	3	1

Készlet meghatározás

(minden eroforrásra)

? EROFORRÁS

- ? FOGLAL

KÉSZLET

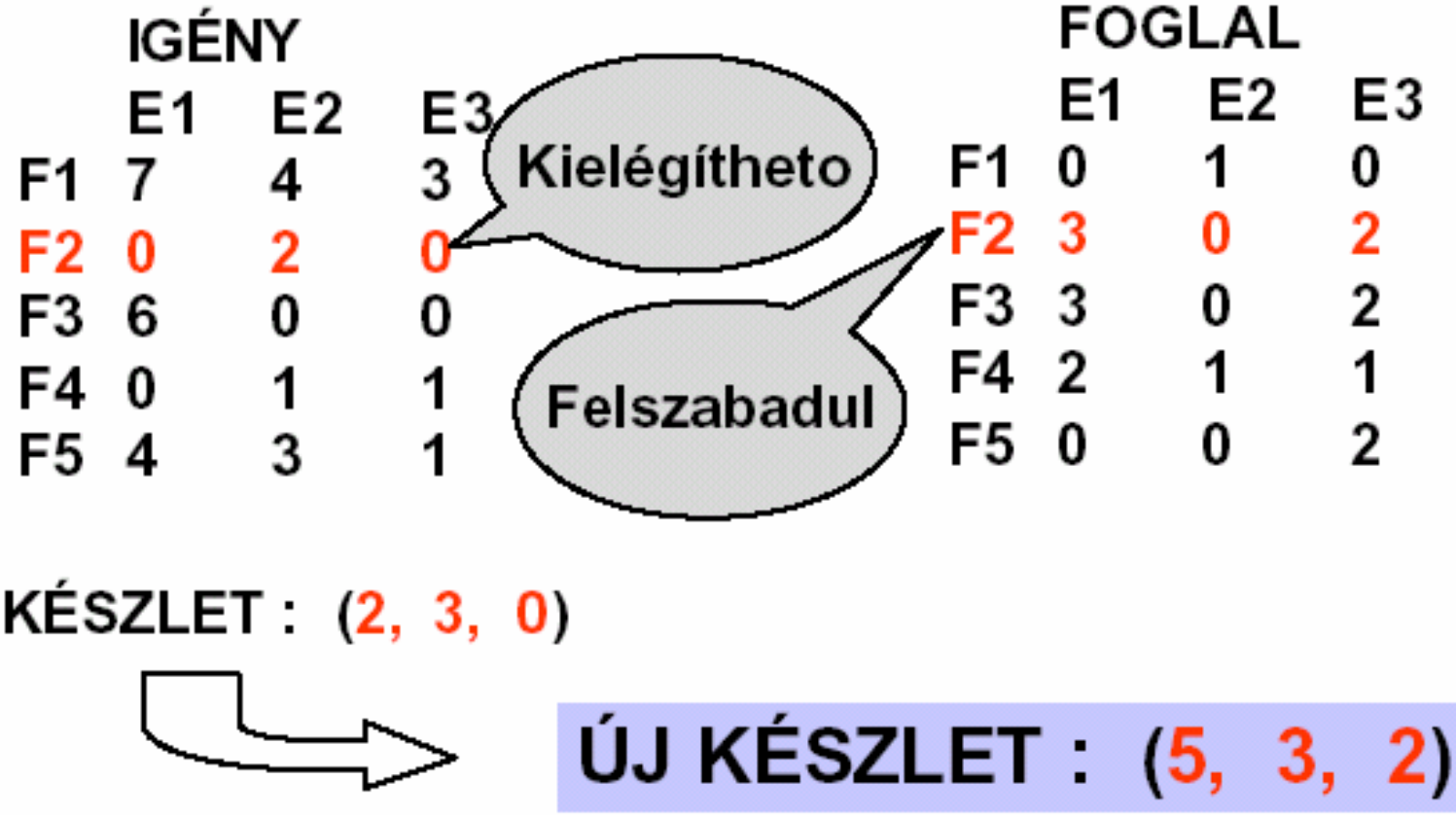
FOGLAL

	E1	E2	E3
F1	0	1	0
F2	3	0	2
F3	3	0	2
F4	2	1	1
F5	0	0	2
? FOGLAL	8	2	7

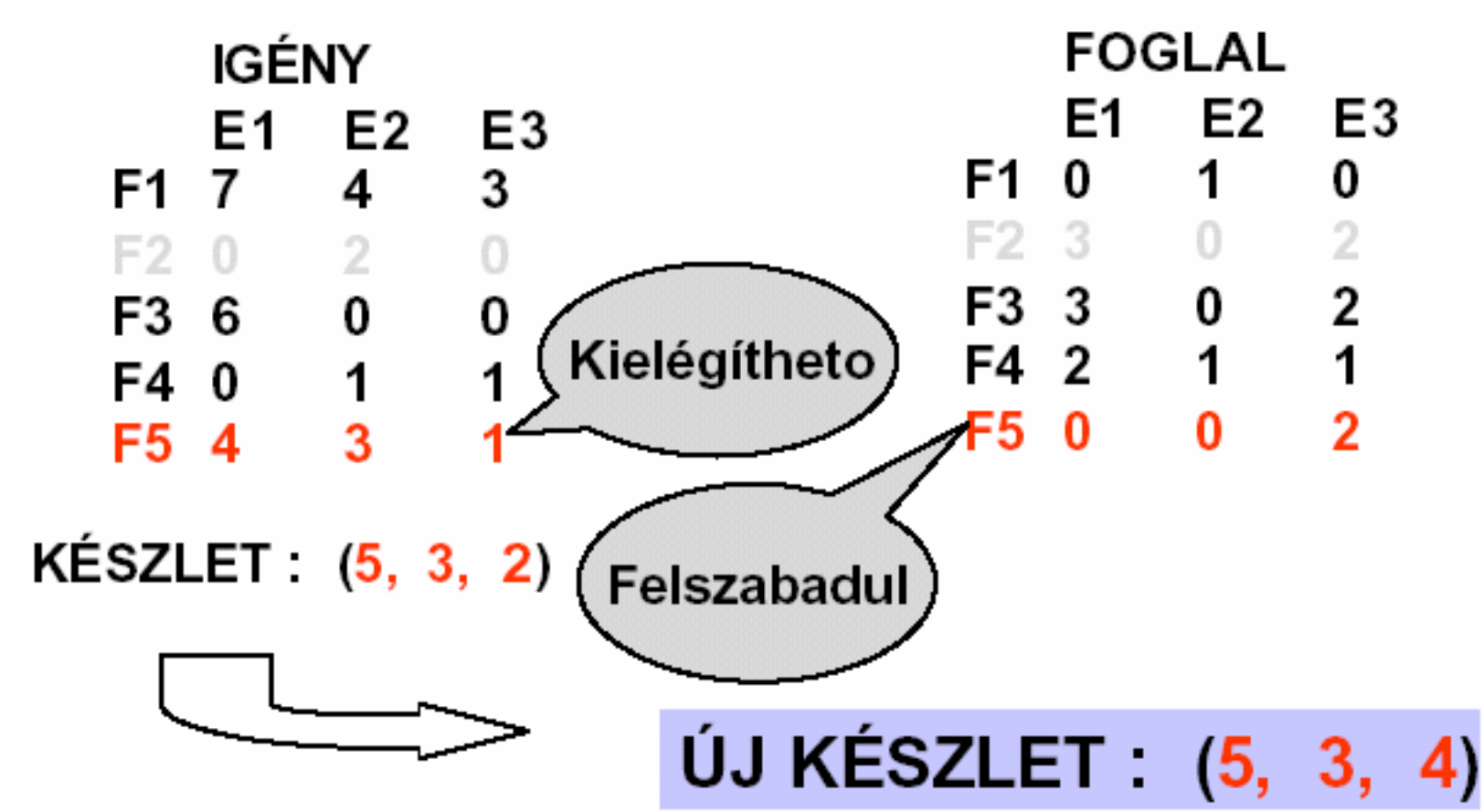
E1: 10 darab
E2: 5 darab
E3: 7 darab

	E1	E2	E3
? EROFORRÁS	10	5	7
? FOGLAL	8	2	7
KÉSZLET	2	3	0

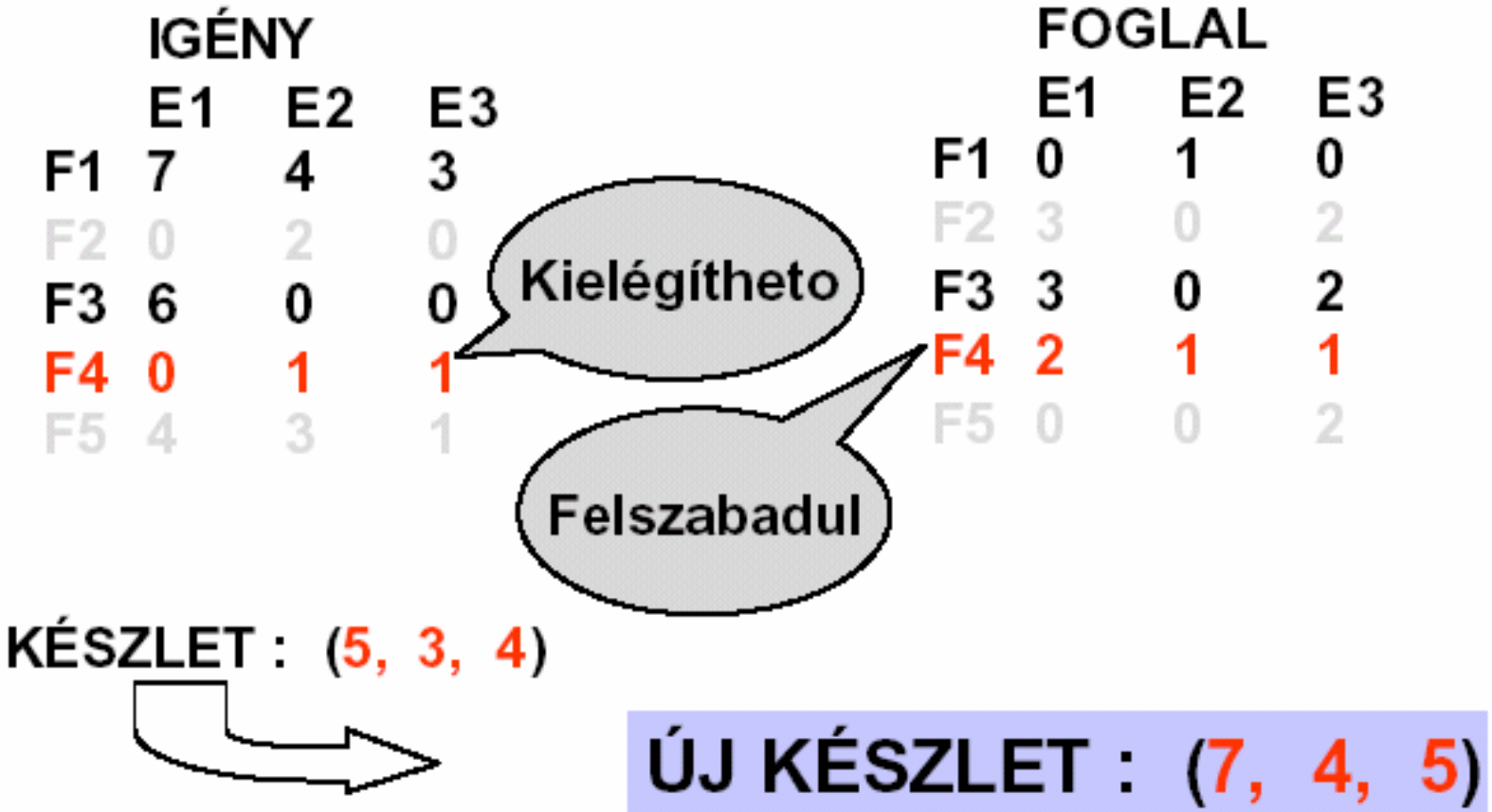
Igény kielégítés - 1. lépés



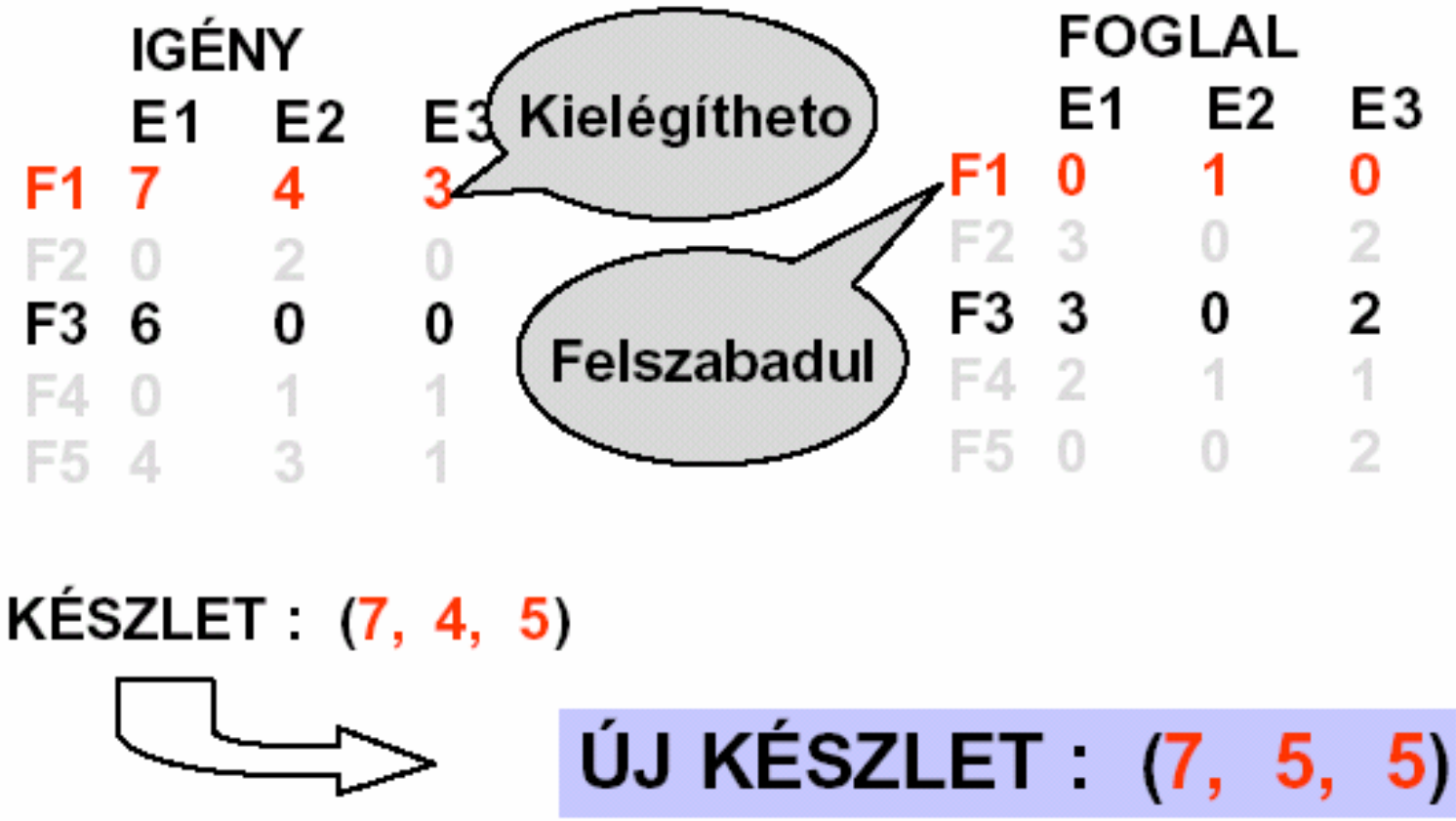
Igény kielégítés - 2. lépés



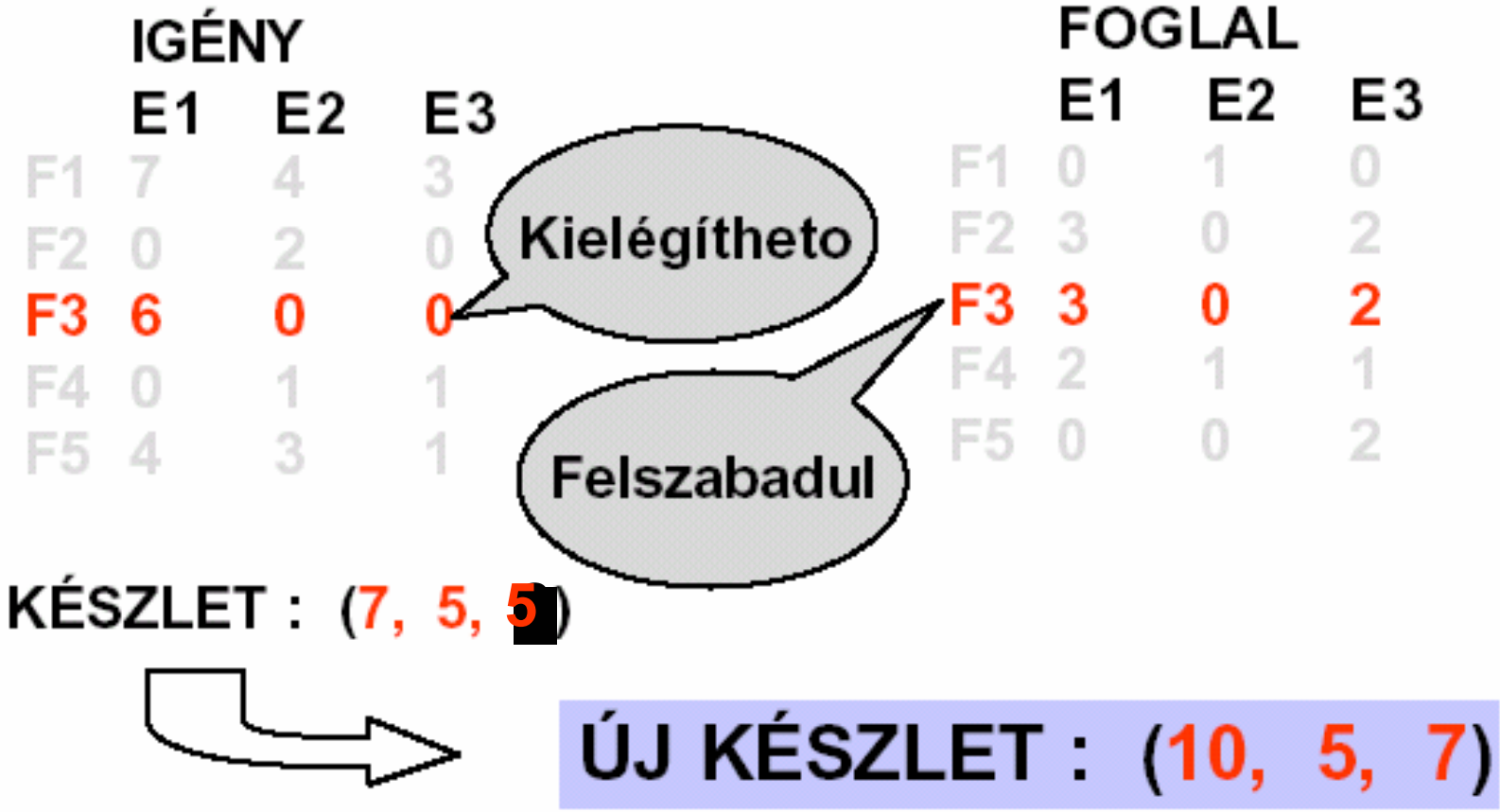
Igény kielégítés - 3. lépés



Igény kielégítés - 4. lépés



Igény kielégítés - 5. lépés



Biztonságos!

Vagyis találtunk legalább egy (ebben a példában több is van) sorrendet, amelyben a folyamatok eroforrás igénye kielégítheto:

F2, F5, F4, F1, F3

Tehát a rendszer

BIZTONSÁGOS ÁLLAPOTBAN van.

Bankár algoritmus

	MAX				VAN	*10*	*8*	*5*	*13*						
	E1	**E2**	**E3**	**E4**	**AKTUÁLIS**	**E1**	**E2**	**E3**	**E4**		**E1**	**E2**	**E3**	**E4**	
F1	5	2	4	3		1	2	1	3		4	0	3	0	HAMIS
F2	7	5	2	7		2	2	2	3		5	3	0	4	HAMIS
F3	2	3	1	2		1	3	0	2		1	0	1	0	IGAZ
F4	3	6	5	2		3	1	1	1		0	5	4	1	HAMIS
					SUM	**7**	**8**	**4**	**9**						
					REST	**3**	**0**	**1**	**4**						
F1	5	2	4	3		1	2	1	3		4	0	3	0	HAMIS
F2	7	5	2	7		2	2	2	3		5	3	0	4	HAMIS
F3	2	3	1	2		0	0	0	0		2	3	1	2	IGAZ
F4	3	6	5	2		3	1	1	1		0	5	4	1	HAMIS
					SUM	**6**	**5**	**4**	**7**						
					REST	**4**	**3**	**1**	**6**	Nincs BIZTONSÁGOS állapotban!					