Bölüm 7: Kilitlenme

İşletim Sistemleri

Sistem Modeli

- Sistem kaynaklardan oluşur.
- Kaynak türleri $R_1, R_2, \ldots, R_m$
  - CPU döngüleri, bellek alanı, I/O aygıtları gibi.
- Her kaynak tipi R_i, somut nesnelere (*instance*) W_i sahiptir.
- Her süreç bir kaynağı:
  - İster (request),
  - Kullanır (use),
  - Serbest (free) bırakır.

Kaynak Edinimi

- Kaynakları korumak için semafor kullanma. (a) Bir kaynak. (b) İki kaynak.

```
typedef int semaphore;
semaphore resource_1;

void process_A(void) {
    down(&resource_1);

    use_resource_1();

    up(&resource_1);
}
```

```
typedef int semaphore;
semaphore resource_1;
semaphore resource_2;
void process_A(void) {
    down(&resource_1);
    down(&resource_2);
    use_both_resources();
    up(&resource_2);
    up(&resource_1);
}
```

Kilitlenme İçermeyen Kod

```
typedef int semaphore;
semaphore resource_1;
semaphore resource_2;
void process_A(void) {
    down(&resource_1);
    down(&resource_2);
    use_both_resources();
    up(&resource_2);
    up(&resource_1);
}
```

```
void process_B(void) {
    down(&resource_1);
    down(&resource_2);
    use_both_resources();
    up(&resource_2);
    up(&resource_1);
}
```

Olası Kilitlenme İçeren Kod

```
typedef int semaphore;
semaphore resource_1;
semaphore resource_2;
void process_A(void) {
    down(&resource_1);
    down(&resource_2);
    use_both_resources();
    up(&resource_1);
    up(&resource_2);
}
```

```
void process_B(void) {
    down(&resource_1);
    down(&resource_2);
    use_both_resources();
    up(&resource_2);
    up(&resource_1);
}
```

Kilitlenme

- Bir dizi sürecin bloke olduğu durumdur.
  - Her süreç *bir kaynağı* tutar, ve
  - Başka bir süreç tarafından tutulan *bir kaynağı* bekler.
- Örneğin, iki tren aynı hatta ve bir ray var, trenlerin hiçbiri hareket edemiyor.
- Bazı kaynakları tutarken, diğer süreçler tarafından tutulan kaynakları bekleyen iki veya daha fazla süreç olduğunda ortaya çıkar.
- Örneğin,
  - **Süreç 1**, **Kaynak 1**'i tutuyor ve
  - **Süreç 2** tarafından tutulan **Kaynak 2**'yi bekliyor ve
  - **Süreç 2**, **Kaynak 1**'i bekliyor.

Kilitlenme Koşulları

- Karşılıklı dışlama (*mutual exclusion*)
  - Bir kaynağı aynı anda yalnızca bir iş parçacığı kullanabilir.
- Tut ve bekle durumu (*hold and wait*)
  - Bir kaynağı tutan iş parçacığı, diğerlerinin tuttuğu kaynakları bekliyor.
- Kesme yok (*no preemption*)
  - Kaynak, yalnızca onu tutan iş parçacığı tarafından serbest bırakılabilir.
- Döngüsel bekleme (*circular wait*)
  - T_0, T_1 tarafından tutulan bir kaynağı bekler,
  - T_1, T_2 tarafından tutulan bir kaynağı bekler, …

Kilitlenme Tespiti

- **Kaynak Tahsis (resource allocation) Çizgesi**: Süreçlerin ve kaynakların, kaynak tahsisini temsil eden yönlü kenarlara sahip çizge.
- **Döngüsel Tespit**: Kaynak tahsis çizgesinde bir döngünün bulunması.
- **Bekleme (wait-for) Çizgesi**: Kaynak tahsis çizgesine benzer, ancak yönlü kenarlar süreçler arasındaki bekleme ilişkilerini temsil eder.
- **Banker Algoritması**: Bir durumun güvenli mi, kilitlenmiş mi olduğunu belirlemek için, maksimum ihtiyaç hakkındaki bilgileri kullanan algoritma.

Kilitlenme Yönetim Teknikleri

- Tespit (*Deadlock Detection*)
- Kaçınma (*Deadlock Avoidance*)
- Engelleme (*Deadlock Prevention*)
- Kurtarma (*Deadlock Recovery*)

Tespit (Deadlock Detection)

- Kilitlenmeleri ele alan *reaktif* bir yaklaşımdır.
- Sistem, periyodik olarak kaynaklar ve süreçlerin durumunu inceler.
- Döngüsel bekleme veya potansiyel kilitlenmeleri belirler.
- Kaynak tahsis çizgesi veya beklemeye göre çizge algoritmaları kullanır.
- Kaynak tahsisi üzerinde önemli bir yük getirmez.
- Kilitlenmelerin tespitinde gecikmelere yol açabilir.

Kaçınma (Deadlock Avoidance)

- Kilitlenmelerin oluşmasını önceden engeller.
- *Proaktif* bir yaklaşımdır.
- Sistem, kaynak taleplerini değerlendirir ve kilitlenmeye yol açmayacaksa izin verir.
- Kilitlenme oluşmadığı için, kurtarma mekanizmalarına ihtiyaç duyulmaz.
- Tespit veya kurtarma yöntemlerinden daha verimli.
- Kaynak tahsisine kısıtlamalar getirebilir.

Engelleme (Deadlock Prevention)

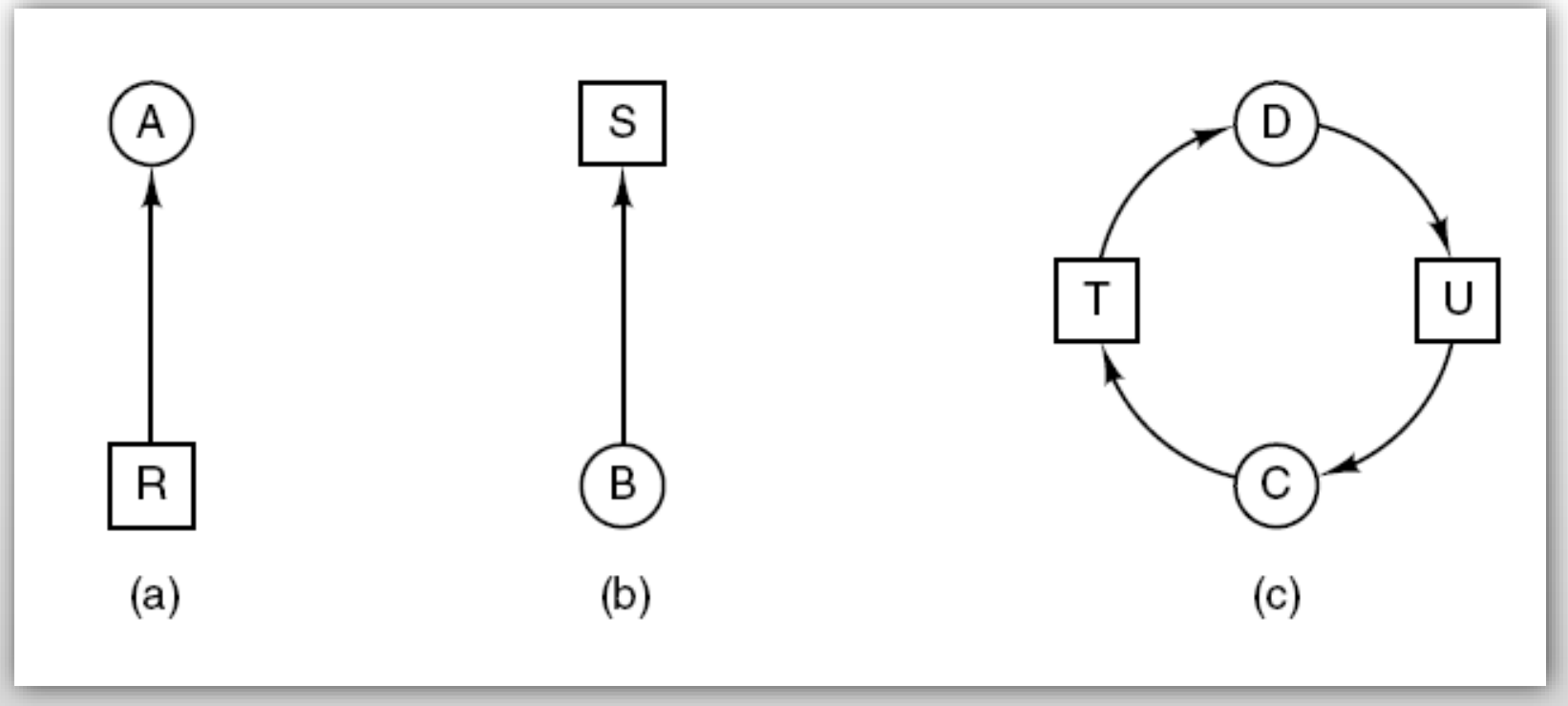
- Kilitlenme oluşumu için gerekli olan koşullardan bir veya daha fazlasını ortadan kaldırmaya odaklanır.
- *Proaktif* bir yaklaşımdır.
- Kilitlenme koşullarının oluşmasını önlemek için çeşitli teknikler kullanılır,
  - Süreçlerin belirli bir sırayla kaynak talep etmesi ve elinde tutması,
  - Süreç bekletilirken kaynakların geri alınması.
- Kaynak geri alma gibi bazı önlemler, ek karmaşıklık ve yük getirebilir.
- Karmaşık sistemlerde tüm kilitlenme koşullarını önlemek her zaman mümkün veya pratik olmayabilir.

Kurtarma (Deadlock Recovery)

- Bir kilitlenme tespit edildiğinde, süreçlerin durumunu geri alarak veya kaynakları geri alarak kilitlenmeyi kırar.
- Mevcut süreçler üzerinde önemli bir kesinti olmadan kilitlenmelerden kurtulmayı sağlar.

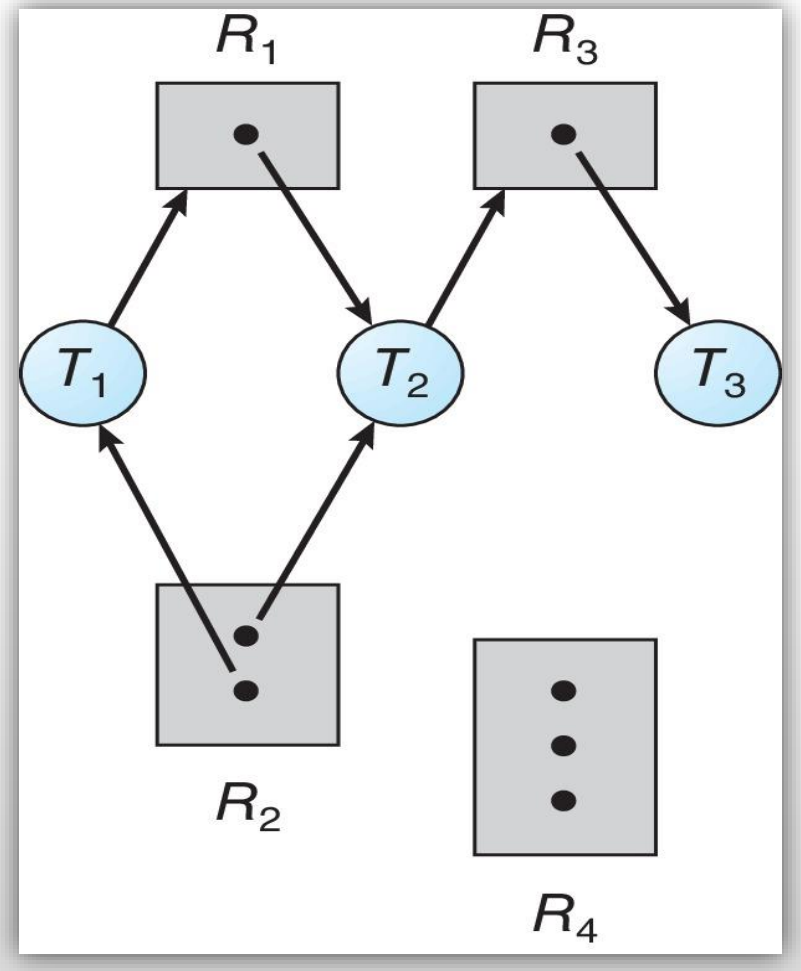
Kaynak Tahsis Çizgesi

- (a) Bir kaynağı tutma. (b) Bir kaynak isteme. (c) Kilitlenme.



Kaynak Tahsis Çizgesi

- Bir tane R1 kaynağı,
- İki tane R2 kaynağı,
- Bir tane R3 kaynağı,
- Üç tane R4 kaynağı,
- T1, bir R2 kaynağını tutar, ve
  - R1 kaynağını bekler.
- T2, R1 kaynağını,
  - bir R2 kaynağını tutar, ve
  - R3 kaynağını bekler.
- T3, R3 kaynağını tutar.

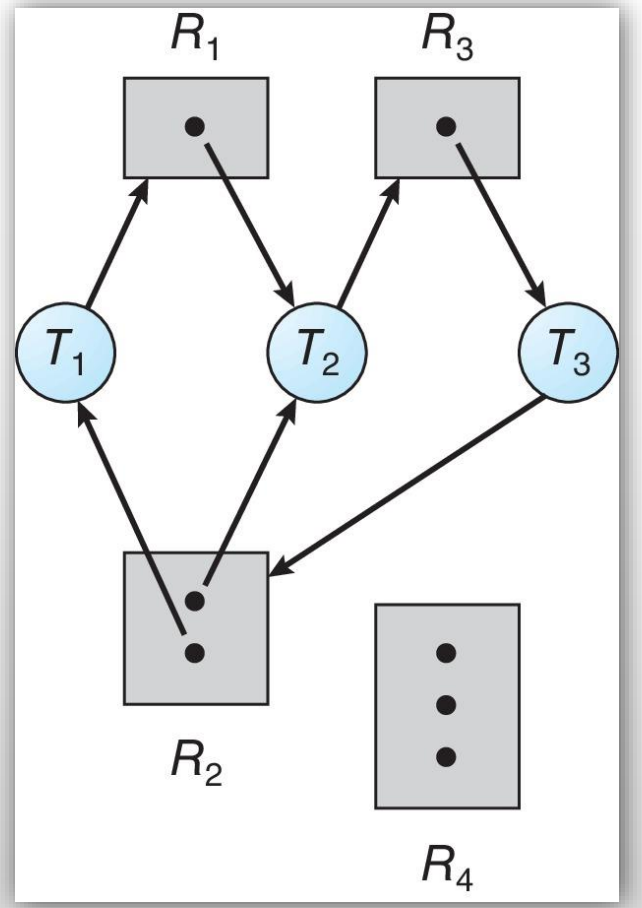


Kaynak Tahsis Çizgesi

- Çizge, döngü içermiyorsa,
  - kilitlenme **yok.**
- Döngü içeriyorsa,
  - o kaynak türünde yalnızca bir kaynak varsa, kilitlenme **var.**
  - o kaynak türünde birçok kaynak varsa, kilitlenme **olasılığı var.**

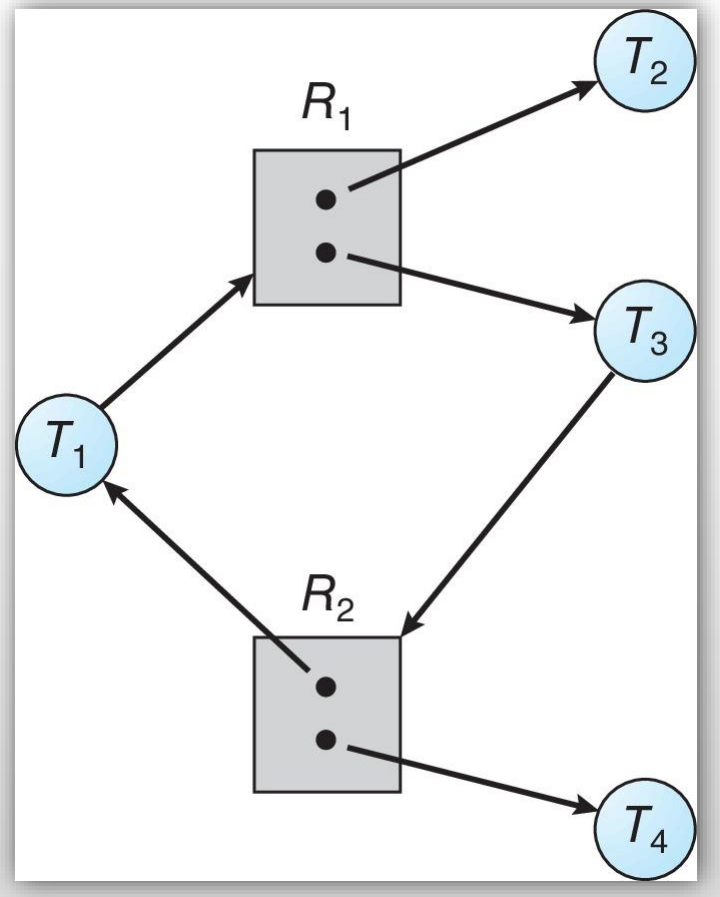
Kilitlenme İçeren Kaynak Tahsis Çizgesi

- .



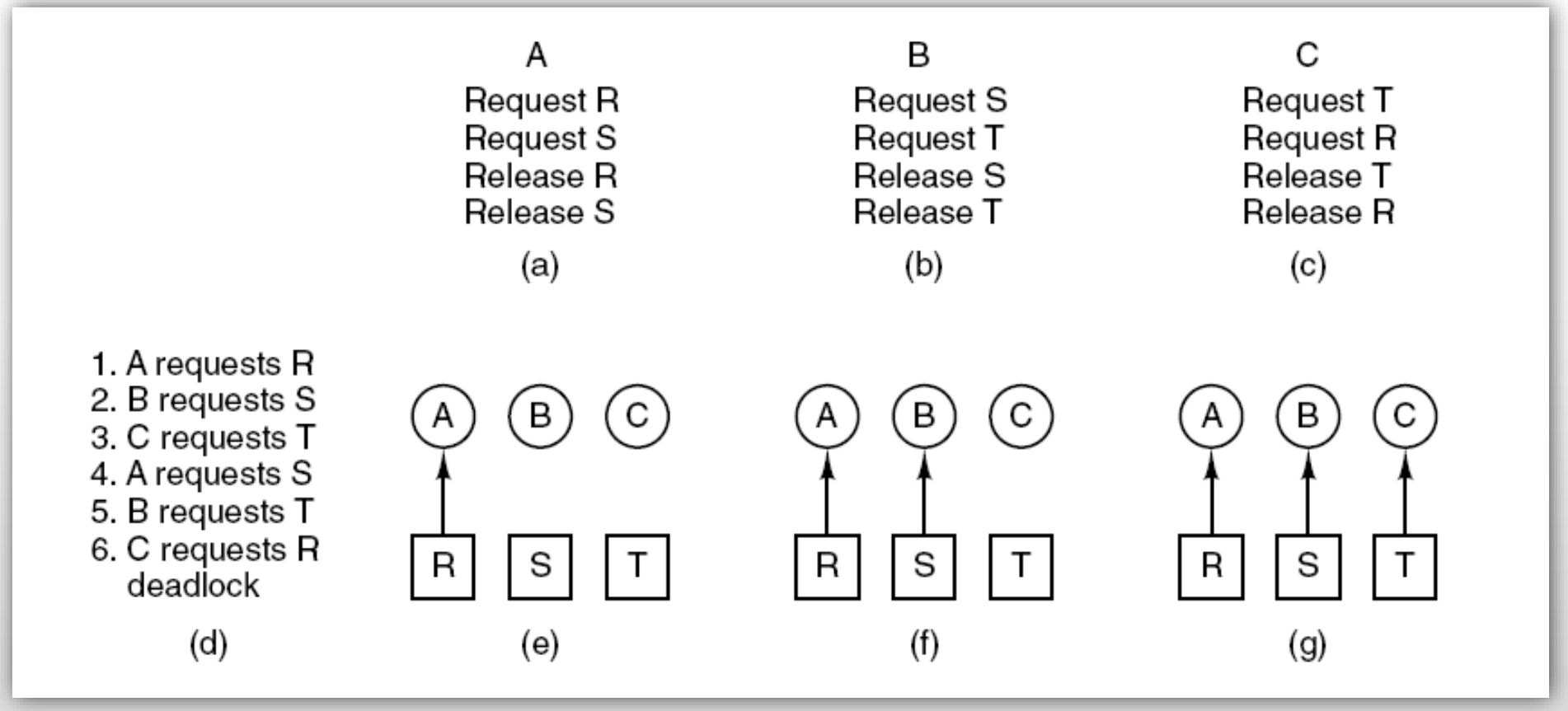
Kilitlenme İçermeyen Kaynak Tahsis Çizgesi

- .



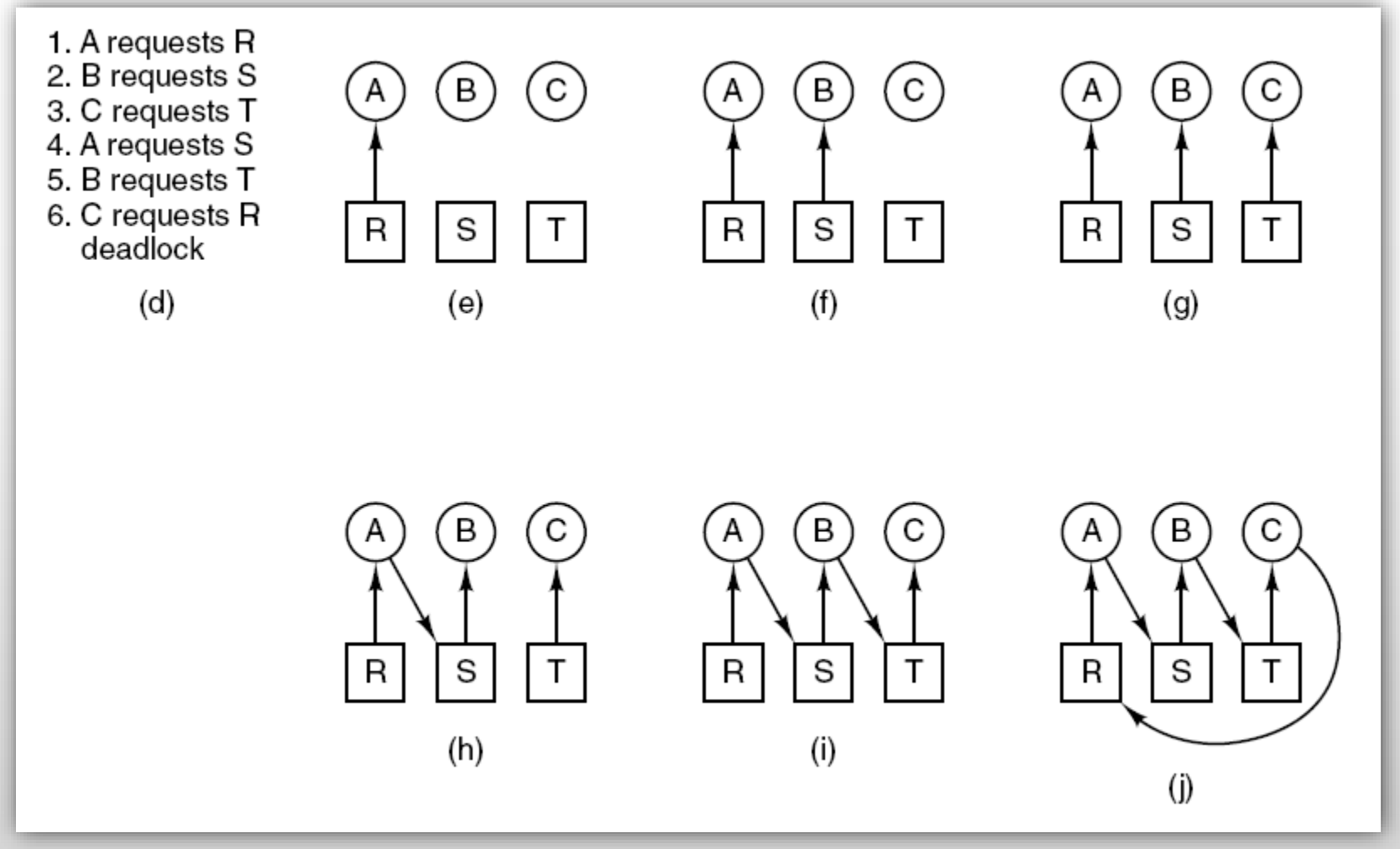
Kilitlenme Nasıl Oluşur?

- .

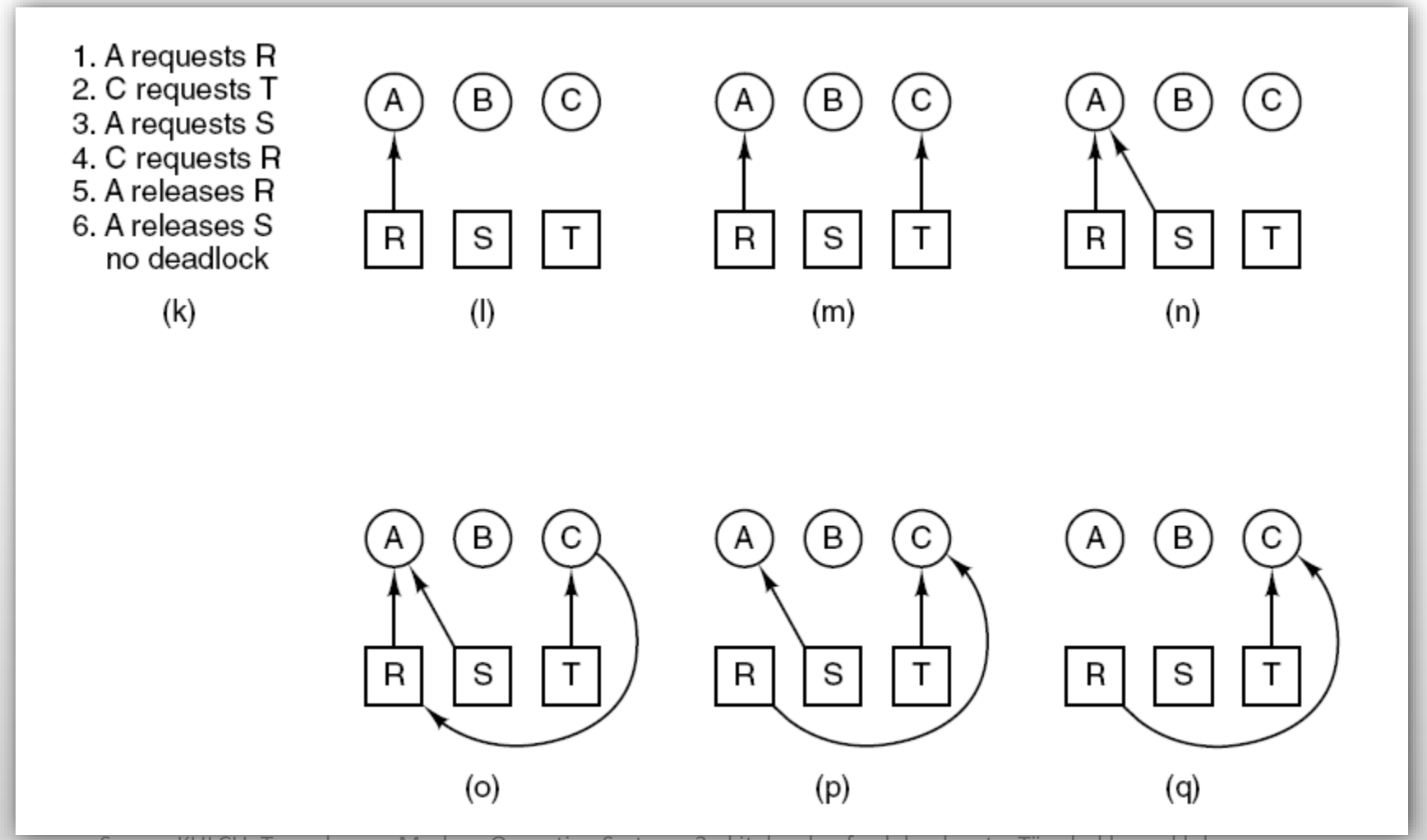


Kilitlenme Nasıl Oluşur?

- .



Kilitlenme Nasıl Önlenebilir?



Kilitlenmelerle Başa Çıkma Stratejileri

- Sorunu görmezden gel. ☺
- Sorunu tespit ve kurtarma.
- Kaynak tahsisi dikkatli yapılarak, dinamik olarak kilitlenmeden kaçınma.
  - en büyük kozumuz yakalanmamak ☺
- Gerekli dört koşuldan biri reddedilerek önlenir.
  - Karşılıklı dışlama (*mutual exclusion*)
  - Tut ve bekle durumu (*hold and wait*)
  - Kesme yok (*no preemption*)
  - Döngüsel bekleme (*circular wait*)

Devekuşu (Ostrich) Algoritması

- Deve kuşları, kafalarını toprağa gömüp sorunları görmezden gelir.
- Kilitlenmeleri tespit ve kaçınma arasında bir denge kurmayı amaçlar.

Devekuşu (Ostrich) Algoritması

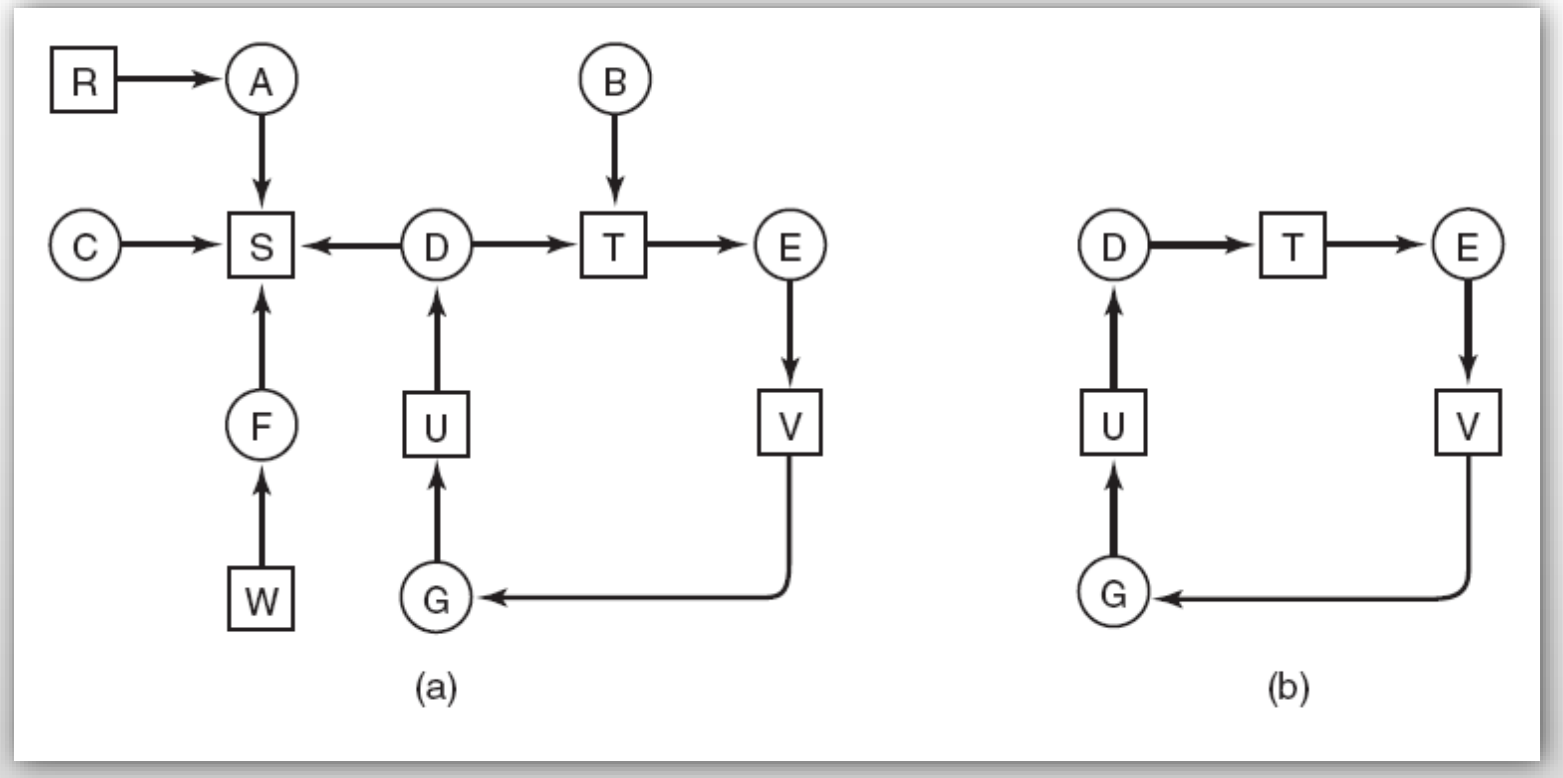
- Kilitlenmeleri geçici olarak yok sayar ve istekleri işlemeye devam eder.
- Kilitlenme meydana geldiğinde sistem, pasif bir moda girer.
  - Normal mod,
    - kilitlenme olup olmadığı kontrol edilmeden istekler işlenir.
  - Pasif mod,
    - kilitlenmeler periyodik kontrol edilir.
    - kilitlenme algılanırsa, kaynak serbest bırakılır, normal moda döner.
- Problem nadiren gerçekleşirse tercih edilebilir.
- Tespit ve önleme maliyetleri düşüktür.

Kilitlenme Tespiti

- Her bir süreç ve kaynak, çizgede bir *düğüm* ile temsil edilir.
- Süreç ile kaynak arasındaki ilişkiyi bir *kenar* temsil eder.
- Döngü tespiti için Derinlik Öncelikli Arama (DFS) gibi algoritmalar kullanır.
- Döngü varsa, kilitlenme vardır.
- Her sürece atanan kaynaklar bir veri yapısında tutulur.
- Süreçler kaynak alıp, serbest bıraktıkça kaynak tahsisi çizgesi güncellenir.
- Kilitlenme algılanırsa,
  - ilgili süreçlerden biri iptal edilir, veya
  - süreçler tarafından tutulan kaynakların bir kısmı serbest bırakılır.

Kilitlenme Tespiti – Her Türden Bir Kaynak

- (a) Bir kaynak çizgesi. (b) (a)'da tespit edilen bir döngü.

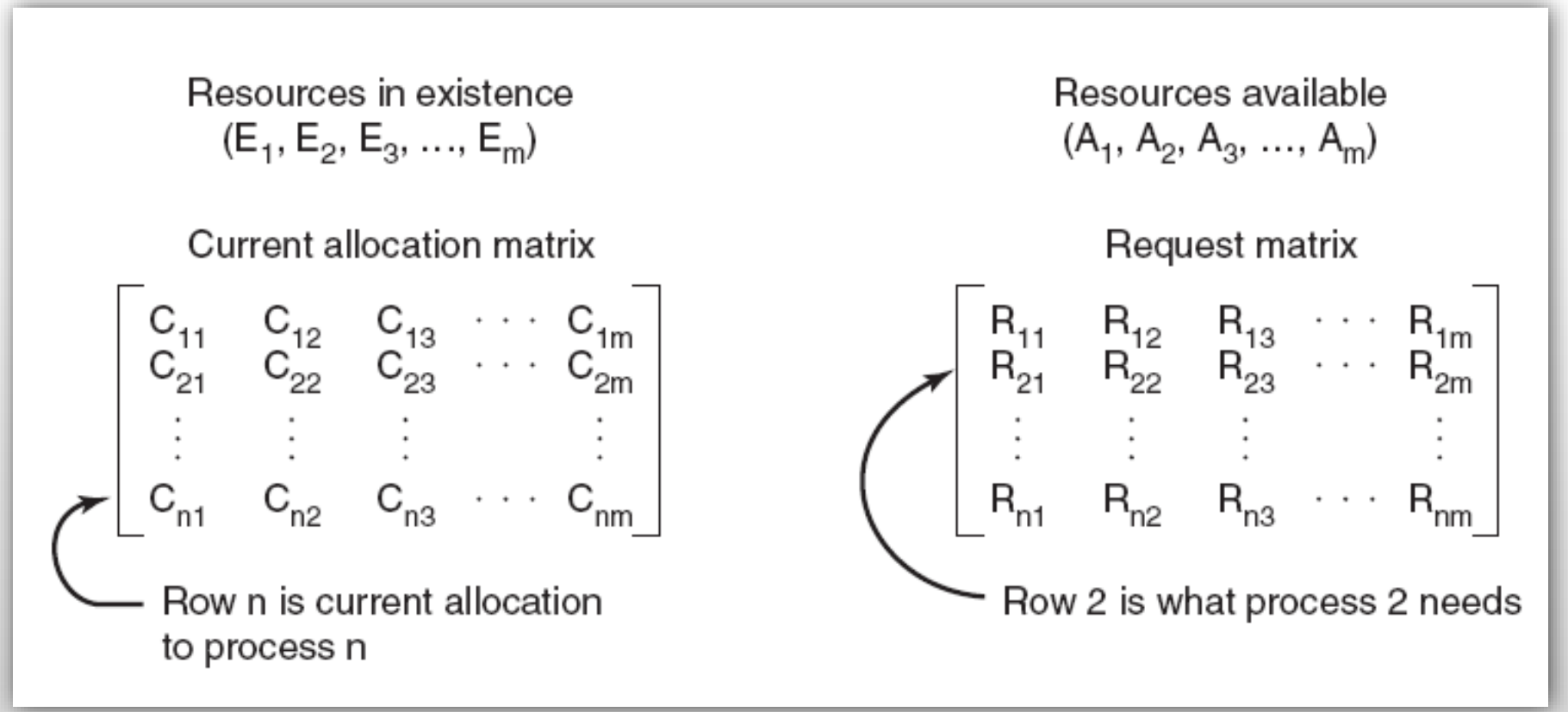


Kilitlenme Tespit Algoritması

1. Çizgedeki her bir *N* düğümü için, *N* düğümünden başlayarak;
2. *L*'yi boş liste olarak, tüm okları işaretlenmemiş olarak ata.
3. Mevcut düğümü L'nin sonuna ekle.
    1. Düğüm L'de iki kez varsa, çizge döngü içerir, algoritma sona erer.
4. Mevcut düğümden giden işaretlenmemiş ok,
    1. varsa, 5. adıma git;
    2. yoksa, 6. adıma git.
5. Oklardan birini seç, işaretle, okun gösterdiği düğüme git, 3. adıma git.
6. Başlangıç düğüme gelindi ise, çizge döngü içermez, algoritma sona erer.
7. Aksi takdirde düğümü listeden çıkar, önceki düğüme dön, 3. adıma git.

Kilitlenme Tespiti – Her Türden Çoklu Kaynak

- Kilitlenme tespit algoritması dört tane veri yapısı kullanır.



Kilitlenme Tespit Algoritması

C: *current allocation matrix* (var olan kaynakları tutar).

R: *request matrix* (istekte bulunulan kaynakları tutar).

A: *available resource count* (serbest olan kaynak sayısı).

1. **R** matrisinin, **i** satırında, <= **A**, işaretlenmemiş bir $\mathbf{P_i}$ süreci ara.
2. Bulunursa, **C**'nin **i**'inci satırına **A**'yı ekle, süreci işaretle, 1. adıma dön.
3. Böyle bir süreç yoksa, algoritma sona erer.

Kilitlenme Tespiti İçin Bir Örnek

- .

Tape drives Plotters Scanners CD Roms

E = (4 2 3 1)

Tape drives Plotters Scanners CD Roms

A = (2 1 0 0)

Current allocation matrix

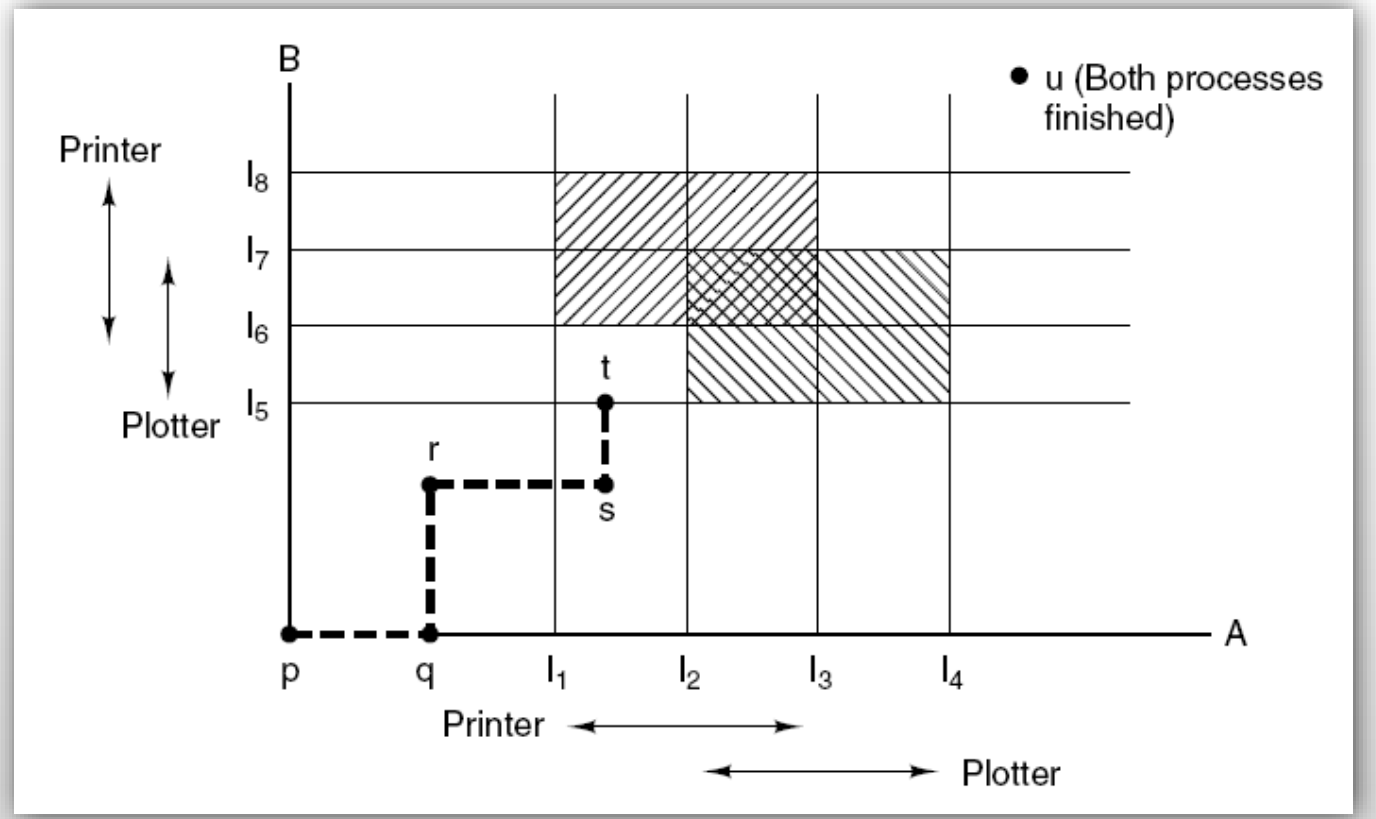
$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{bmatrix}$$

Request matrix

$$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix}$$

Kilitlenmeden Kaçınma

- İki süreç tarafından kullanılan iki kaynağın kullanım eğrisi (*trajectory*).



Güvenli ve Güvensiz Durumlar

- (a) tablosunda verilen durum güvenli,
- Süreçler B->C->A sırasında çalıştırılır.

	Has	Max
A	3	9
B	2	4
C	2	7

Free: 3

(a)

	Has	Max
A	3	9
B	4	4
C	2	7

Free: 1

(b)

	Has	Max
A	3	9
B	0	–
C	2	7

Free: 5

(c)

	Has	Max
A	3	9
B	0	–
C	7	7

Free: 0

(d)

	Has	Max
A	3	9
B	0	–
C	0	–

Free: 7

(e)

Güvenli ve Güvensiz Durumlar

- (b) tablosunda verilen durum güvensiz.

	Has	Max
A	3	9
B	2	4
C	2	7

Free: 3

(a)

	Has	Max
A	4	9
B	2	4
C	2	7

Free: 2

(b)

	Has	Max
A	4	9
B	4	4
C	2	7

Free: 0

(c)

	Has	Max
A	4	9
B	—	—
C	2	7

Free: 4

(d)

The Banker's Algoritması – Tek Kaynak

- Mevcut kaynakları ve her bir sürecin maksimum talebini takip eder.
- Sürecin talebi sistemi güvensiz bir durumda bırakmıyorsa, kaynaklar verilir.
- Sürecin talebi, güvensiz bir duruma yol açıyorsa, ertelenir.
- Tahsis ve kaynak ihtiyacını takip etmek için bir matris kullanır.
- Kilitlenmeye yol açmadan sonlanan süreç varsa durum güvenli kabul edilir.
- Algoritma,
  - tüm süreçler tamamlandığında veya
  - kilitlenme meydana geldiğinde sona erer.

The Banker's Algoritması – Tek Kaynak

- Üç kaynak tahsis durumu: (a) Güvenli. (b) Güvenli. (c) Güvensiz.

	Has	Max
A	0	6
B	0	5
C	0	4
D	0	7

Free: 10

(a)

	Has	Max
A	1	6
B	1	5
C	2	4
D	4	7

Free: 2

(b)

	Has	Max
A	1	6
B	2	5
C	2	4
D	4	7

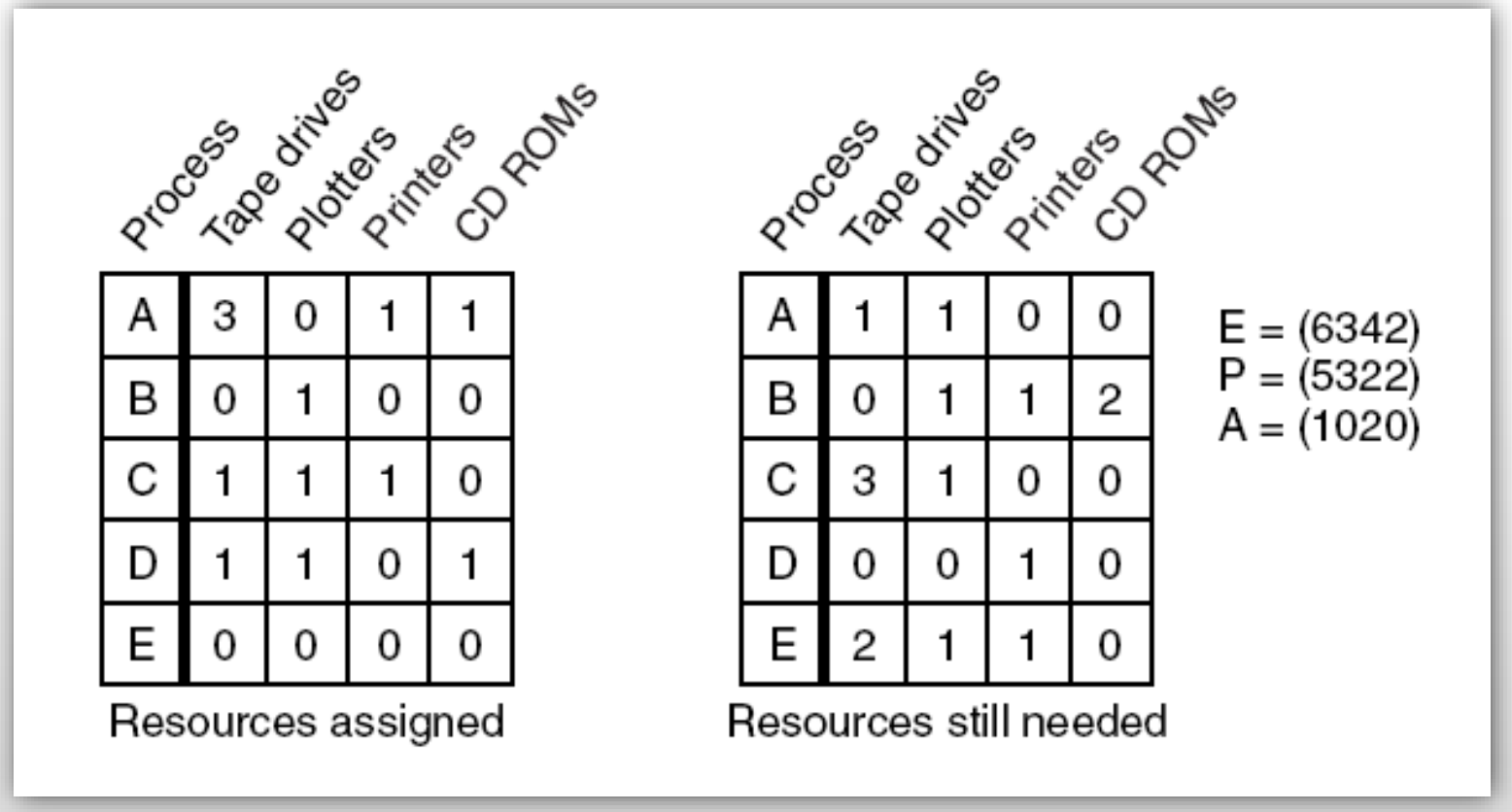
Free: 1

(c)

The Banker's Algoritması – Çoklu Kaynak

- Her sürece, bir dizi kaynak türü ve talep edilen maksimum miktarı atanır.
- Algoritma, her sürece tahsis edilen kaynakları ve mevcut kaynakları izler.
- Sürecin talebi,
  - yeterli miktarda kaynak varsa ve
  - kilitlenmeye neden olmuyorsa karşılanır.
- Algoritma, kaynak tahsisini yeterli kaynak sağlanana kadar erteleyebilir.
- Kaynak tahsisini izlemek ve kontrol için arka planda sürekli çalışır.

The Banker's Algoritması – Çoklu Kaynak



Process	Tape drives	Plotters	Printers	CD ROMs
A	3	0	1	1
B	0	1	0	0
C	1	1	1	0
D	1	1	0	1
E	0	0	0	0

Resources assigned

Process	Tape drives	Plotters	Printers	CD ROMs
A	1	1	0	0
B	0	1	1	2
C	3	1	0	0
D	0	0	1	0
E	2	1	1	0

Resources still needed

E = (6342)
P = (5322)
A = (1020)

The Banker’s Algoritması – Çoklu Kaynak

R: *request matrix* (istekte bulunulan kaynakları tutar).

A: *available resource count* (serbest olan kaynak sayısı).

1. **R** matrisinde, <= **A** kaynağa ihtiyaç duyan satır arar.
   - Böyle satır yoksa, hiçbir süreç tamamlanmayacak, sistem kilitlenecektir.
2. Seçili satırdaki süreci, işaretler, ve tüm kaynaklarını **A**'ya ekler.
3. İşaretlenmemiş ve talebi karşılanabilen süreç oldukça adımları tekrarlar.
4. Tüm süreçler işaretlendi ise kilitlenme yoktur.
5. Talebi karşılanmayan süreç varsa, kilitlenme vardır.

Kilitlenme Önleme

- **Karşılıklı dışlama** (*mutual exclusion*) koşulunu kaldırma,
  - Her seferinde yalnızca bir süreç kaynak isteyebilir ve tahsis edilebilir.
  - Kaynaklar sıralanır veya kaynağın birden çok örneğine izin verilir.
- **Tut ve bekle** (*hold and wait*) durumunu kaldırma,
  - Süreçlerin diğer süreçleri beklerken kaynakları tutması önlenir.
  - Süreç yürütülmeden önce ihtiyaç duyduğu tüm kaynakları talep eder.

Kilitlenme Önleme

- **Kesme yok** (*no preemption*) koşulunu kaldırmak,
  - Kaynak, yüksek öncelikli süreç tarafından önceden kullanılmamalı.
  - Süreç bitene kadar kaynakları tutmasına izin verilir.
  - Süreç bloke olursa kaynakları serbest bırakır.
- **Döngüsel bekleme** (*circular wait*) koşulunu kaldırmak,
  - Kaynak tahsis sıralaması tanımlar.
    - Kaynakları numaralandırarak veya
    - Kaynak hiyerarşisi kullanarak yapılabilir.

Döngüsel Bekleme Koşulunu Kaldırmak

- (a) Sayısal olarak sıralanmış kaynaklar. (b) Bir kaynak tahsis çizgesi.

Kilitlenme Önleme Yaklaşımları

Durum	Yaklaşım
Karşılıklı dışlama	Talepleri biriktir.
Tut ve bekle	Başlangıçta tüm kaynakları talep et.
Kesme yok	Kaynakları serbest bırak.
Döngüsel bekleme	Kaynakları sayısal olarak sırala.

Kilitlenmeden Kurtarma (recovery)

- Önleme (*prevention*): süreçlerin kaynak talep etmesini kısıtlar.
- Tespit (*detection*): kilitlenmeyi tespit etmek için sistemi izler.
- Geri alma (*rollback*): güvenli duruma geri dönmeyi sağlar.
- Sonlandırma (*termination*): süreci sonlandırır (*killing*).
- Çağrı üstünlüğü (*preemption*): kaynağı bir süreçten alıp başkasına verir.
- Kaynak tahsis ayarlaması (*setting allocated resource count*): sürece tahsis edilen kaynakların sayısının dinamik olarak ayarlanmasını içerir.

Diğer Sorunlar

- İki aşamalı kilitleme (*two phase locking*)
  - Yalnızca gerektiğinde kilit alınır ve en kısa sürede serbest bırakılır.
  - Veritabanı sistemlerinde kullanılan bir protokol.
- İletişim kilitlenmeleri (*communication deadlocks*)
  - Birden çok sürecin birbirinden yanıt beklediği sistemlerde oluşur.
- Canlı kilit (*live lock*)
  - Bir sürecin, diğerinin ileriye doğru bir adım atmasını beklediği durum.
- Açlık (*starvation*)
  - Bir sürecin sonlanmak için ihtiyaç duyduğu kaynakları,
    - elde etmesi engellendiğinde ortaya çıkar.

İki Aşamalı Kilitleme

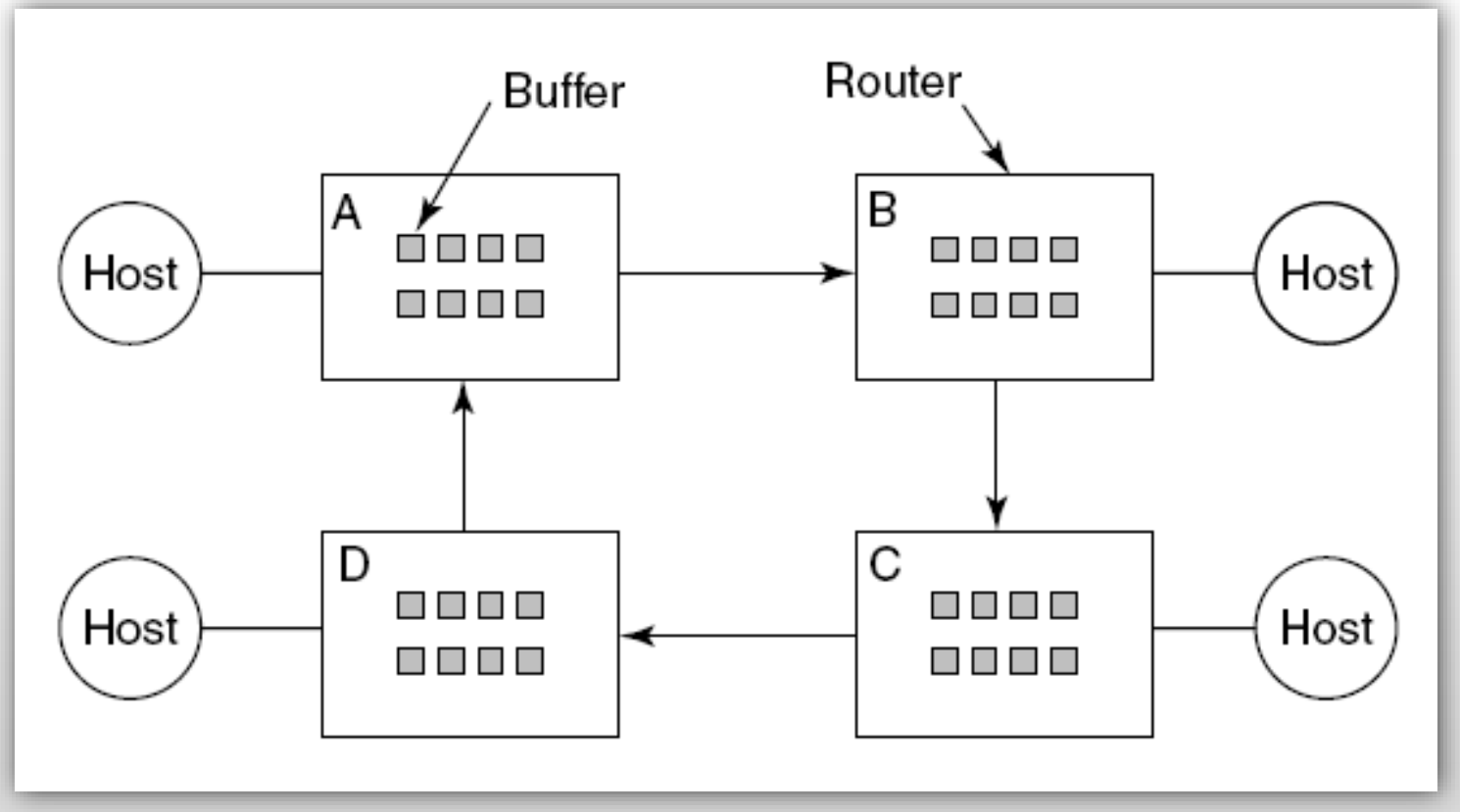
- İşletim sistemlerinde kilitlenmeleri önlemek veya yönetmek için kullanılan bir kilitleme protokolüdür.
- Bir süreç kaynak talep ettiğinde ve kaynakları serbest bıraktığında iki aşamalı bir süreci izler.
  - Büyüme (*growth*) evresi,
    - Süreç, kaynakları talep ederken kilitleri sırayla alır.
    - Kaynak talepleri kabul edilir, ancak serbest bırakılmaz.
  - Küçülme (*shrinkage*) evresi.
    - Süreç, kaynakları serbest bırakırken kilitleri sırayla bırakır.
    - Kaynaklar başka süreçler tarafından talep edilebilir hale gelir.

İletişim Kilitlenmeleri

- İletişim yollarında oluşan döngüler veya bekleme durumları nedeniyle meydana gelir.
- Bir süreç diğer bir sürece bir kaynak veya mesaj gönderirken,
  - Hedef süreçteki kaynakların kullanılabilir olmaması veya
  - Mesajın alınmaması sonucunda ortaya çıkar.
- Bu durumda,
  - Gönderen süreç beklemeye alınır ve
  - Kaynaklar serbest bırakılana veya mesaj alınana kadar süreç bekler.

İletişim Kilitlenmeleri

- Bir ağda kaynak kilitlenmesi. Tampon bellekler dolu.



Canlı Kilitlenme

- Süreçler birbirlerine yanıt verse de, gerçekte ilerleme kaydedilmez.
- Özellikle çoklu iş parçacıklı veya dağıtık sistemlerde karşılaşılır.
- Canlı kilitlenme,
  - İşlemlerin birbirlerine yanıt verme girişimleriyle başlar,
  - Ancak bu girişimler bir döngüye veya durumun kötüleşmesine yol açar.

Canlı Kilit

- Canlı Kilit'e (*livelock*) yol açabilecek meşgul bekleme (*busy waiting*).

```
void process_A(void) {
   enter_region(&resource_1);
   enter_region(&resource_2);
   use_both_resources();
   leave_region(&resource_2);
   leave_region(&resource_1);
}
```

```
void process_B(void) {
   enter_region(&resource_2);
   enter_region(&resource_1);
   use_both_resources();
   leave_region(&resource_1);
   leave_region(&resource_2);
 }
```

Açlık

- Açlık durumu, bazı süreçlerin sürekli olarak diğerlerinden öncelikli olarak kaynaklara erişmesi sonucunda oluşur.
- Bu durumda,
  - Bazı süreçler sürekli olarak kaynaklara erişirken,
  - Diğer süreçler sürekli olarak beklemek zorunda kalır ve kaynak talepleri reddedilir.

SON