



Operating Systems

Deadlocks-Part1

Seyyed Ahmad Javadi

sajavadi@aut.ac.ir

Spring 2021

Outline

- Liveness
- System Model
- Deadlock Characterization
- Methods for Handling Deadlocks
- Deadlock Prevention
- Deadlock Avoidance

Chapter Objectives

- Illustrate how deadlock can occur when mutex locks are used
- Define the four necessary conditions that characterize deadlock
- Identify a deadlock situation in a resource allocation graph
- Evaluate the four different approaches for preventing deadlocks
- Apply the **banker's algorithm** for deadlock avoidance

* مفهوم واسع *

Liveness

- Processes may have to wait indefinitely while trying to acquire a synchronization tool such as a mutex lock or semaphore.

لiveness *
* حیثیت را نهادن از تمهیل کردن آن جلوگیری کن!

- Waiting indefinitely violates the progress and bounded-waiting criteria.

* حیثیت، محدودیت و پیشرفت را نهادن
* liveness, bounded, progress

Liveness

- Liveness refers to a set of properties that a system must satisfy to ensure processes make progress.
- Indefinite waiting is an example of a liveness failure.

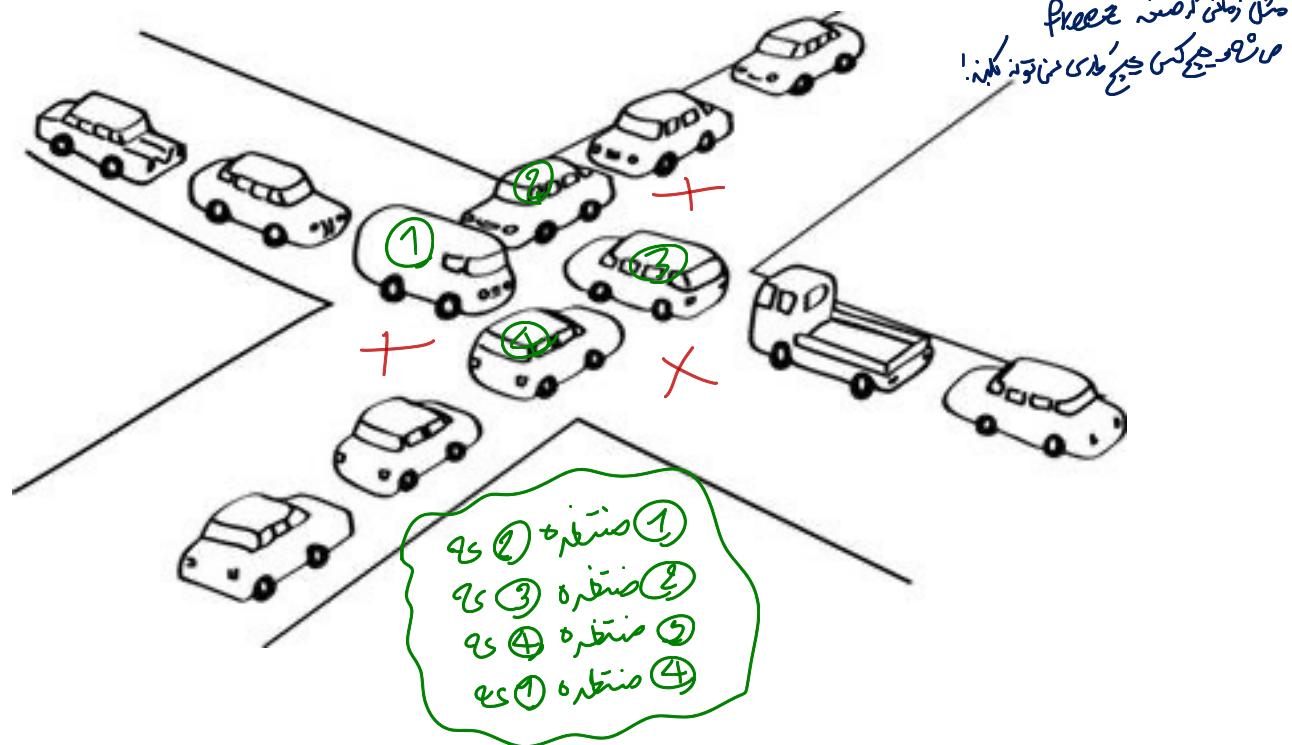
Deadlock

* two or more processes are waiting indefinitely for an event

این روند نمیخواهد!

that can be caused by only one of the waiting processes.

بعنده خود این event را بخواهند اما این پردازشها که قبلاً شروع شده اند آن event را اخذ نمیکنند



Liveness (cont.)

(١) binary glo Semaphore

- Let S and Q be two semaphores initialized to 1

دو تاصنیف کریں کہ سیمافور محفوظ
و ۱ کی ترتیب میں

P_0

P_1



wait(S);



wait(Q);



wait(Q);



wait(S);

P_0 must wait until P_1 executes signal(Q).

...

...

P_1 is waiting until P_0 execute signal(S)

signal(S);

signal(Q);

signal(Q);

signal(S);



کہ دریں deadlock ایجاد ہے و
جھاتتے ہیں صندل جاندے کہ
لiveness توہنی توہنی

Since these signal() operations will never be executed, P_0 and

P_1 are **deadlocked**.

* لینجا ما راجح ایکل وقوع deadlock

صحت مکنی ! حقاً کہ دیس بینوا ایکل وقوع deadlock ہے وجود دار نہ سایر کی وجہ پر صحتی ایسی !

Other Forms of Deadlock

اسم حالت فشای ددگری

liveness failure

■ Starvation – indefinite blocking

- A process may never be removed from the semaphore queue in which it is suspended.

لiveness حفظ حیووند!

■ Priority Inversion – Scheduling problem when lower-priority process holds a lock needed by higher-priority process.

- We do not cover this. (در حد تعریف)

فرعن کنیت کی بروزه در حال اجراست و نیز
دو lock کرده! نیز بروزه کی درگیری
با اعلیه است بالاتر صورت آید و خوب طبق الگوریتم که طریع
کرده قبیل مارک و CPU به این بروزه خود را می‌خواهد.
حالا بروزه جریه برای اجرا شنای با این semaphore داره که
کوچک است وlock کرده بود! (wait در روی این صراف زن !)
حالا بروزه جریه با این که اولویت پر کرده باشد سرمه رون امکانی ندارد!

(خلاص اون میعنی کاملاً خراب و انتقام را نمی‌دم)

System Model

- System consists of resources
- Resource types R_1, R_2, \dots, R_m *النوعات المادية*
- CPU cycles, memory space, I/O devices
- Each resource type R_i has $\underline{W_i}$ instances. *(كل نوع مادي يحتوي على W_i نسخ)*
- Each process utilizes a resource as follows:
 - ① • request
 - ② • use
 - ③ • release

Deadlock with Semaphores

- Data:
 - A semaphore **S1** initialized to 1
 - A semaphore **S2** initialized to 1

- Two processes P1 and P2

- **P1:**

wait (s1)

wait (s2) → *البيانات متاحة
وسيتم إصدارها*

- **P2:**

wait (s2)

wait (s1) → *بيانات متاحة
وسيتم إصدارها*

Deadlock Characterization

Deadlock can arise if four conditions hold simultaneously.

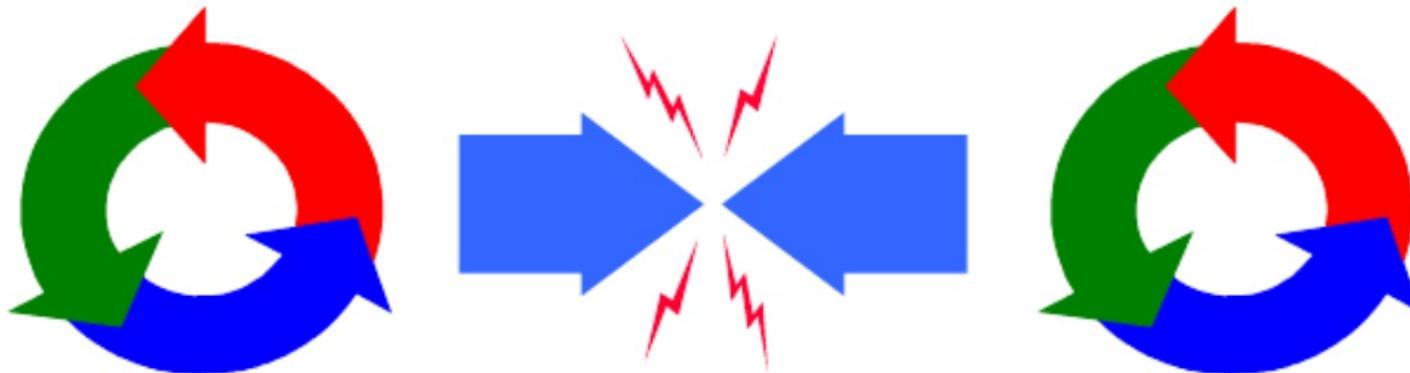
1. Mutual exclusion
2. Hold and wait
3. No preemption
4. Circular wait

true عکس میتواند باشد
*! میتواند دادلکت نباشد

1-Mutual Exclusion

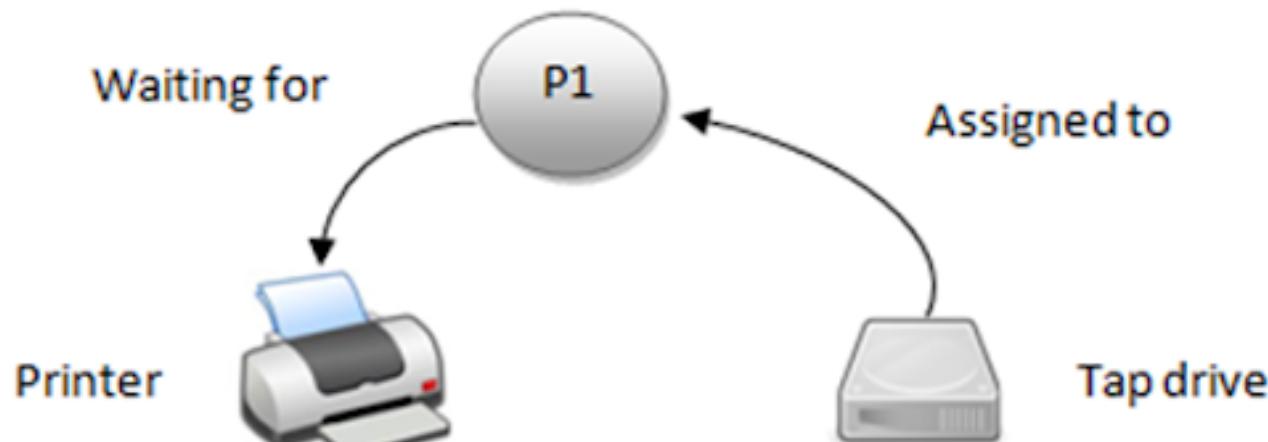
- Only one process at a time can use a resource.

* حونَّ لِلرِّازمِ نَبْوَكَ كَيْفَيْتُ مُسْتَعْدِيْنَ لَهُ دَلْيَيْنَ دَرْجَيْنَ دَرْجَيْنَ



2-Hold and Wait

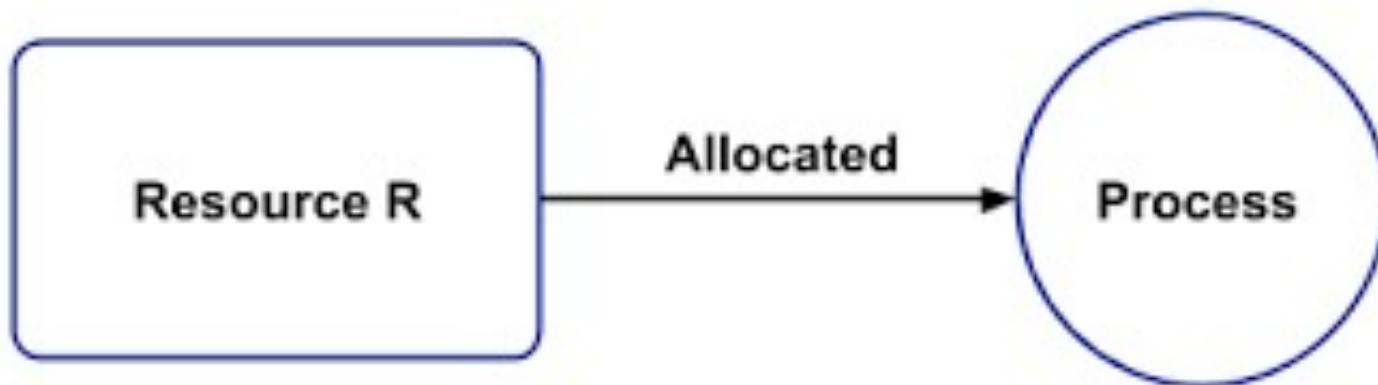
- A process holding at least one resource is waiting to acquire additional resources held by other processes.



3-No Preemption

- A resource can be released only **voluntarily** by the process holding it, after that process has completed its task.

هی وچ کل صنع او دلم آی سبک داره تارماهه نه طریق با این منع سفرموده ازش گذاشتیم!

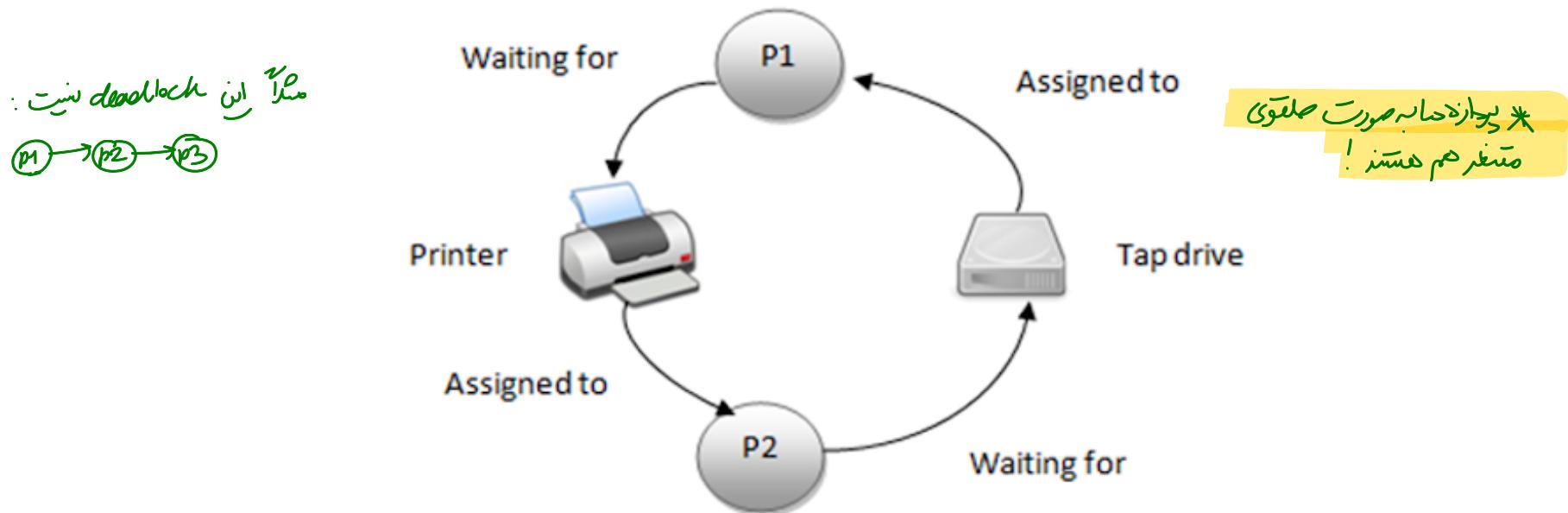


4-Circular Wait



مَوْقِعُ الْجُنُوبِ !
بَاسِتَ بَعْدَ !

- There exists a set $\{P_0, P_1, \dots, P_n\}$ of waiting processes such that:
 - P_0 is waiting for a resource that is held by P_1 ,
 - P_1 is waiting for a resource that is held by P_2 , ...,
 - P_{n-1} is waiting for a resource that is held by P_n ,
 - and P_n is waiting for a resource that is held by P_0 .



Resource-Allocation Graph

یعنی گراف بین مواردی که در سیستم اجرا شوند
و خروجی این موارد!

A set of vertices V and a set of edges E .

- V is partitioned into two types:

- $P = \{P_1, P_2, \dots, P_n\}$,
 - ▶ The set consisting of all the processes in the system.
- $R = \{R_1, R_2, \dots, R_m\}$,
 - ▶ The set consisting of all resource types in the system.
... printers locks

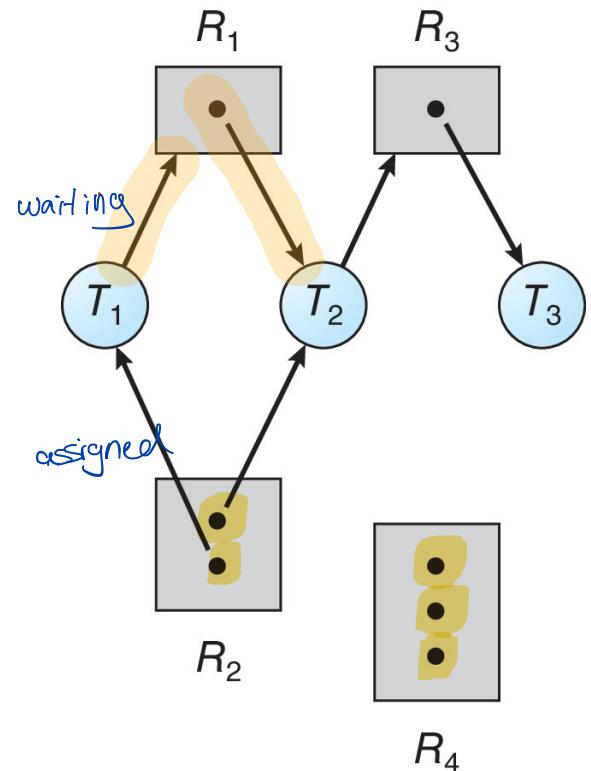
Resource-Allocation Graph

A set of vertices V and a set of edges E .

- **request edge** – directed edge $P_i \rightarrow R_j$
برگ صنیع ارگین پروژه
عن لئی پروژے کام مکن
کیا ورنے اونو رام خواهد
Resource type
- **assignment edge** – directed edge $R_j \rightarrow P_i$
ادئی صنیع بسای پروژه
ان صنیع بارن پروژے امضاون
تائی نه است.

Resource Allocation Graph Example

- One instance of R1
- Two instances of R2
- One instance of R3
- Three instances of R4



Resource Allocation Graph Example

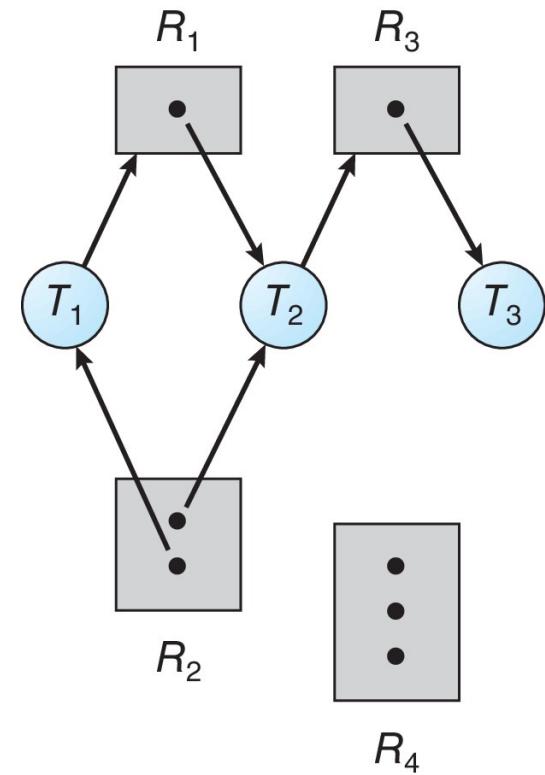
- T1 holds one instance of R2 and is waiting

for an instance of R1

- T2 holds one instance of R1, one instance of R2, and is waiting for an instance of R3.

- T3 is holding one instance of R3

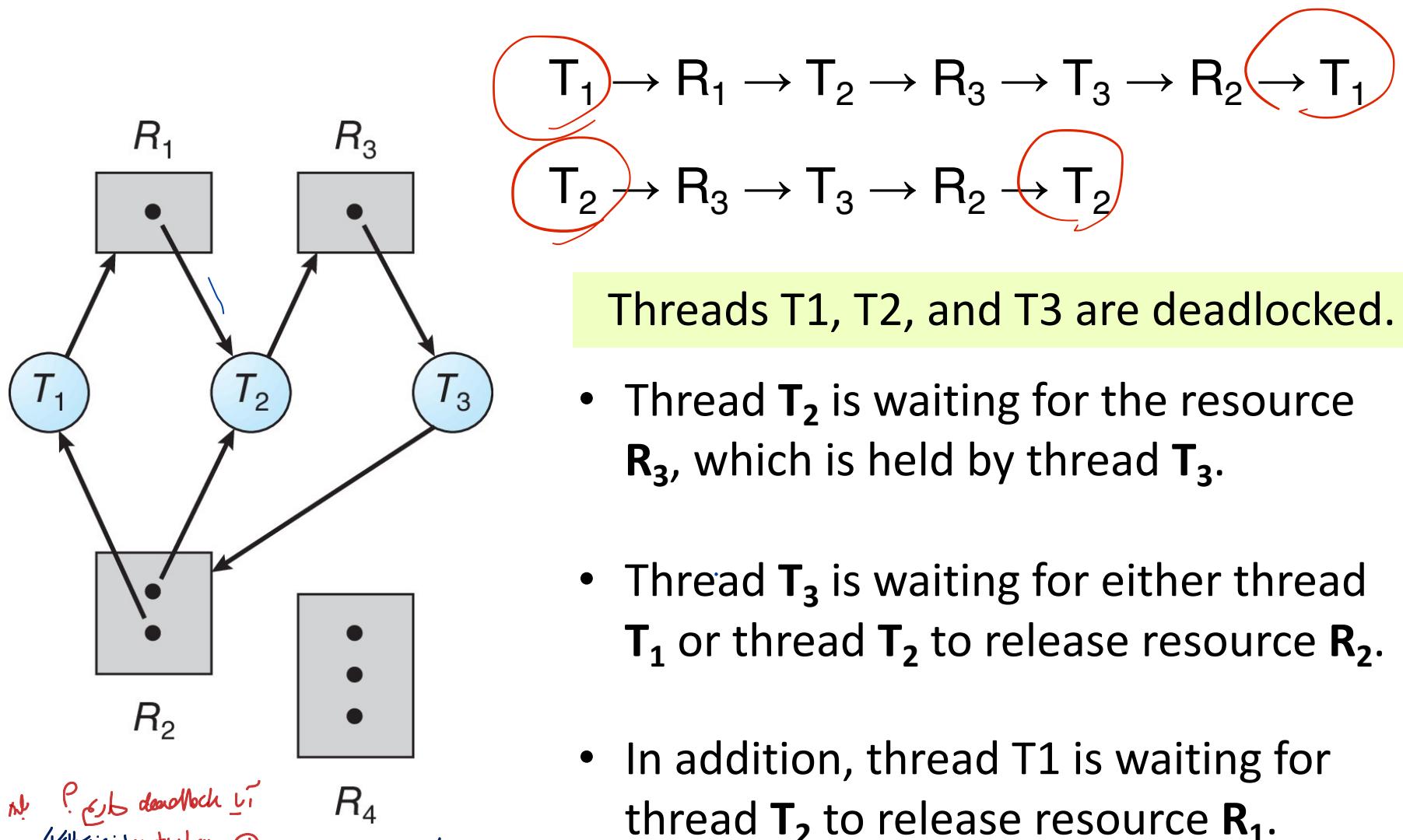
Waiting X



(↗ ↘) 8 "The ! is to be used to indicate that the lock is held by the process" (مُدَّعِيَةً)

✓ Mutual exclusion
✗ Hold and Wait
✓ No preemtive
✗ Circular

Resource Allocation Graph with a Deadlock

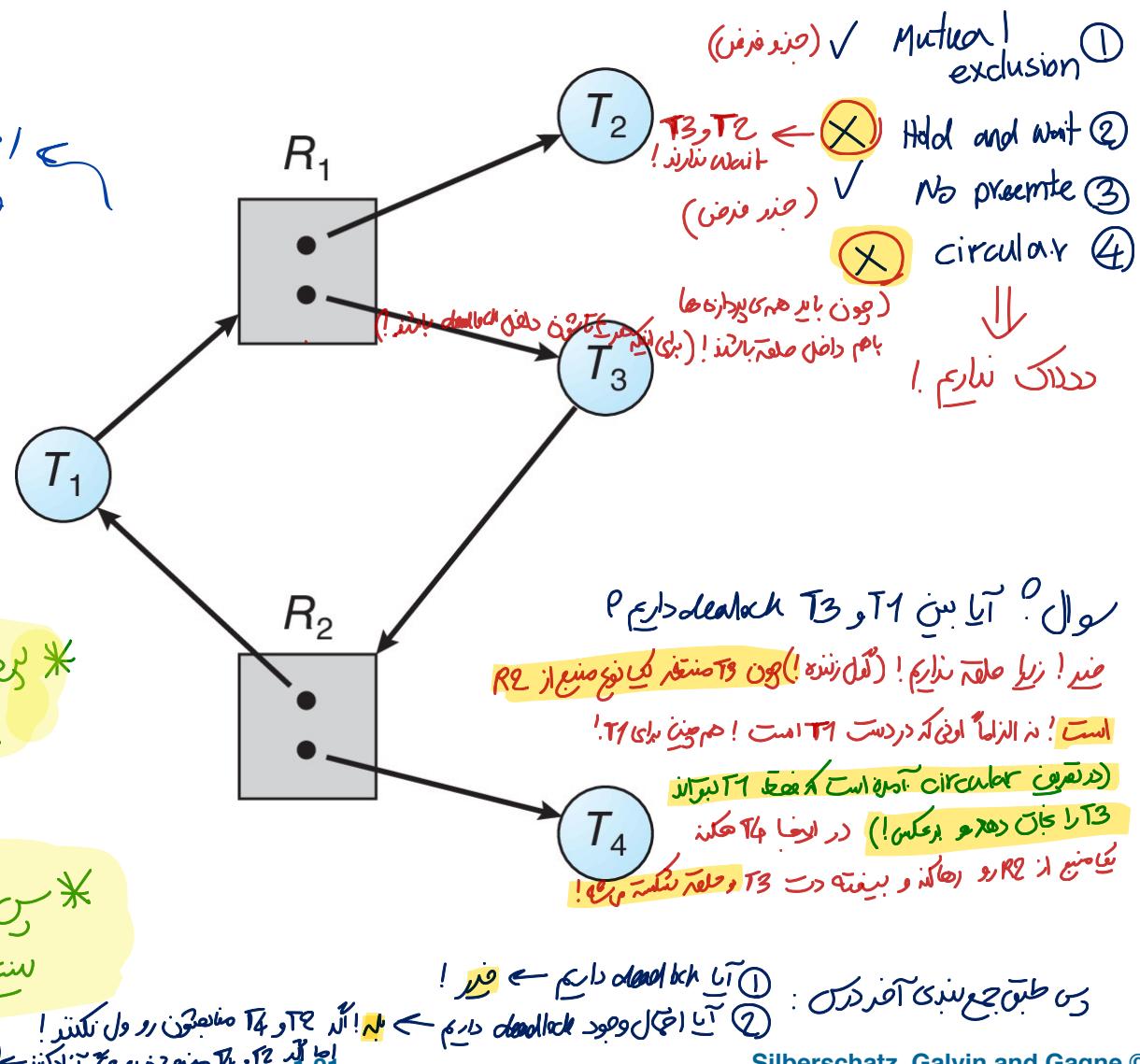


مکالمہ
P. کے deadlock لی
(V) mutual ex ①
(V) Hold and wait ②
(V) No pre-empt ③
Circular ④

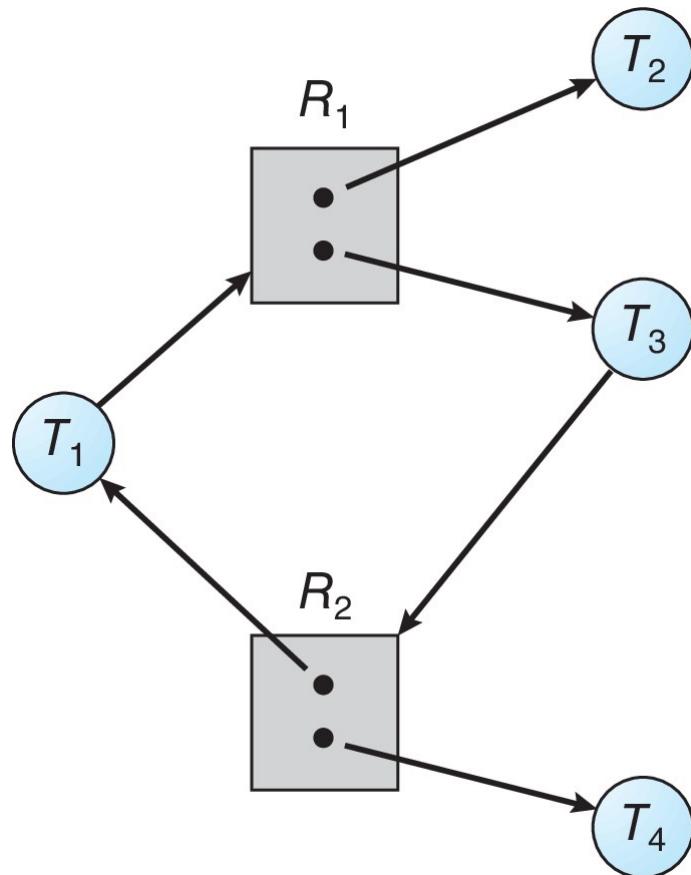
Operating System Concepts – 10th Edition

Is there a Deadlock?

کے اچھاں ددراک چونہاں ہست!
ولی ددراک نواریم!



Graph with a Cycle But no Deadlock



$T_1 \rightarrow R_1 \rightarrow T_3 \rightarrow R_2 \rightarrow T_1$

There is no deadlock. Observe that thread T4 **may release** its instance of resource type R2. That resource can then be allocated to T3, **breaking the cycle.**

(*) (*) !
! no circular wait مکانیزم خروجی نه ممکن *

Basic Facts

میں اچھے سنتے ہوں!

- If graph contains no cycles \Rightarrow no deadlock

اگر کوئی طریق نہ ہو

لے کر قلعہ میں دلکشی!

- If graph contains a cycle \Rightarrow

اگر چھوڑ برس نہ ہوں گے (عذاب!)

- if only one instance per resource type, then deadlock
- if several instances per resource type, possibility of deadlock

بڑا ہوتا سایل ہوا رکھوں : ① چھپ کر مکمل طریق
 ② اچل کر قلعہ میں دلکشی

if the graph contains no cycles \rightarrow no thread in the system is deadlocked

If the graph does contain a cycle \rightarrow a deadlock may or may not exist.

