

CSE 451: Operating Systems Spring 2005

Module 8 Deadlock

Ed Lazowska
lazowska@cs.washington.edu
Allen Center 570



(Is Google the greatest, or what?)

4/17/2005

© 2005 Gribble, Lazowska, Levy

2

Definition

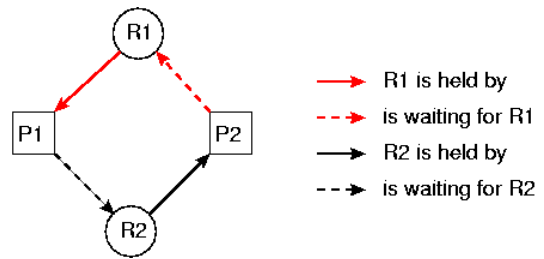
- A thread is **deadlocked** when it's waiting for an event that can never occur
 - I'm waiting for you to clear the intersection, so I can proceed
 - but you can't move until he moves, and he can't move until she moves, and she can't move until I move
 - thread A is in critical section 1, waiting for access to critical section 2; thread B is in critical section 2, waiting for access to critical section 1
 - I'm trying to book a vacation package to Tahiti – air transportation, ground transportation, hotel, side-trips. It's all-or-nothing – one high-level transaction – with the four databases locked in that order. You're trying to do the same thing in the opposite order.

4/17/2005

© 2005 Gribble, Lazowska, Levy

3

Resource graph



- R1 is held by
- - - is waiting for R1
- R2 is held by
- - - is waiting for R2

- A deadlock exists if there is an *irreducible cycle* in the resource graph (such as the one above)

4/17/2005

© 2005 Gribble, Lazowska, Levy

4

Graph reduction

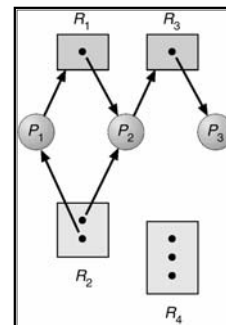
- A graph can be *reduced* by a thread if all of that thread's requests can be granted
 - in this case, the thread eventually will terminate – all resources are freed – all arcs (allocations) to it in the graph are deleted
- Miscellaneous theorems (Holt, Havender):
 - There are no deadlocked threads iff the graph is completely reducible
 - The order of reductions is irrelevant
- (Detail: resources with multiple units)

4/17/2005

© 2005 Gribble, Lazowska, Levy

5

Resource allocation graph with no cycle

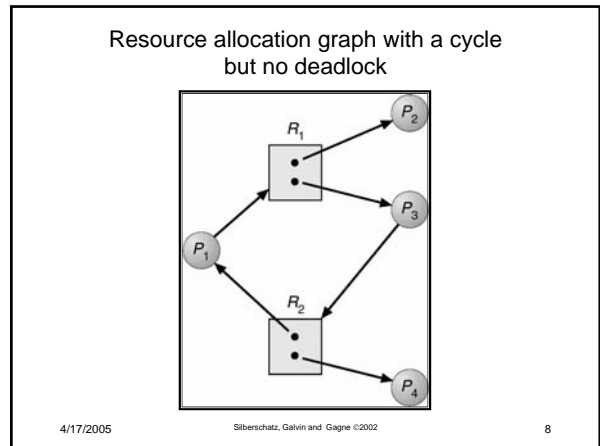
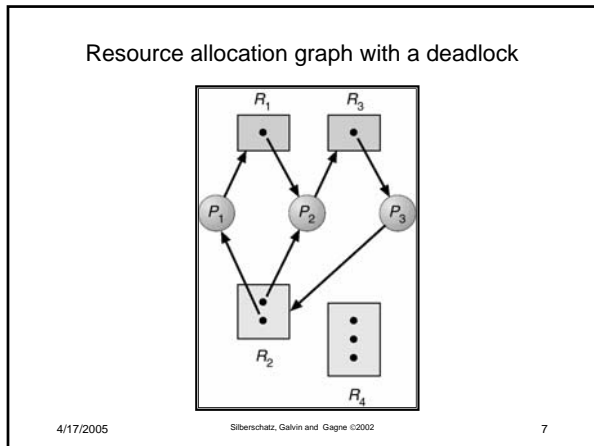


What would cause a deadlock?

4/17/2005

Silberschatz, Galvin and Gagne ©2002

6



- ### Approaches to deadlock
- Prevention – don't let deadlock occur
 - each thread obtains all resources at the beginning; blocks until all are available
 - drawback?
 - resources are numbered; each thread obtains them in sequence (which means acquiring some before they are actually needed)
 - why does this work?
 - pros and cons?
 - each thread states its maximum claim for every resource type; system runs the Banker's algorithm at each allocation request
 - if I were to allocate you that resource, and then everyone were to request their maximum claim for every resource, would there be a deadlock?
 - how do I tell if there would be a deadlock?
- 4/17/2005 © 2005 Gribble, Lazowska, Levy 9

- ### Approaches (cont'd.)
- Detection and correction
 - every once in a while, check to see if there's a deadlock
 - how?
 - if so, eliminate it
 - how?
- 4/17/2005 © 2005 Gribble, Lazowska, Levy 10

- ### Banker's Algorithm example
- When a request is made
 - pretend you granted it
 - pretend all other legal requests were made
 - can the graph be reduced?
 - if so, allocate the requested resource
 - if not, block the thread
- 4/17/2005 © 2005 Gribble, Lazowska, Levy 11

