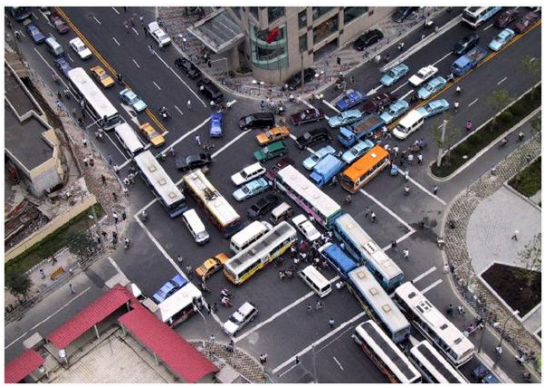
CSE 451: Operating Systems
Winter 2004

Module 8
Deadlock

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

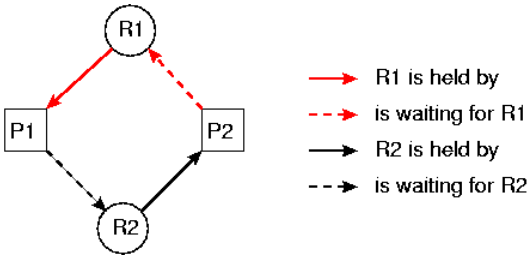


(Is Google the greatest, or what?)

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.

Resource graph

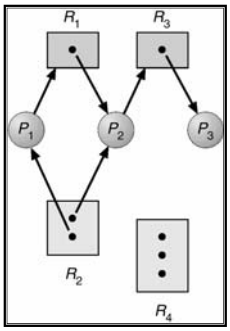


- A deadlock exists if there is an *irreducible cycle* in the resource graph (such as the one above)

Graph reduction

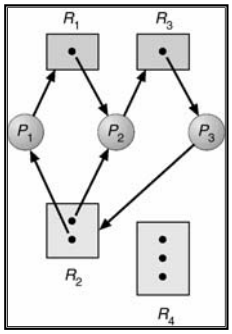
- A graph can be *reduced* by a thread if all of that thread's requests can be granted
  - in this case, the thread terminates – all resources are freed – all arcs (allocations) to it in the graph are deleted
- Miscellaneous theorems (Holt, Havender):
  - There are no deadlocked threads iff the graph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

Resource allocation graph with no cycle

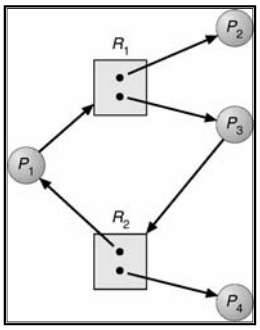


What would cause a deadlock?

Resource allocation graph with a deadlock



Resource allocation graph with a cycle but no deadlock



Approaches to deadlock

- Prevention – don't let deadlock occur
  1. each thread obtains all resources at the beginning; blocks until all are available
     - drawback?
  2. 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?
  3. 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?
     - example: a hammer and five screwdrivers

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?