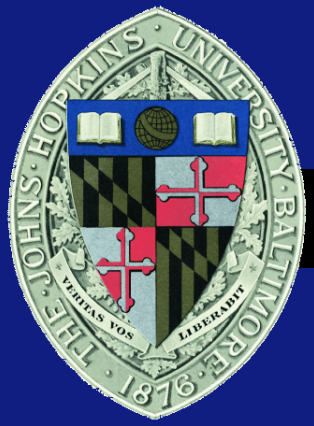
Lecture 13.3
Barriers and Deadlock

EN 600.320/420

Instructor: Randal Burns

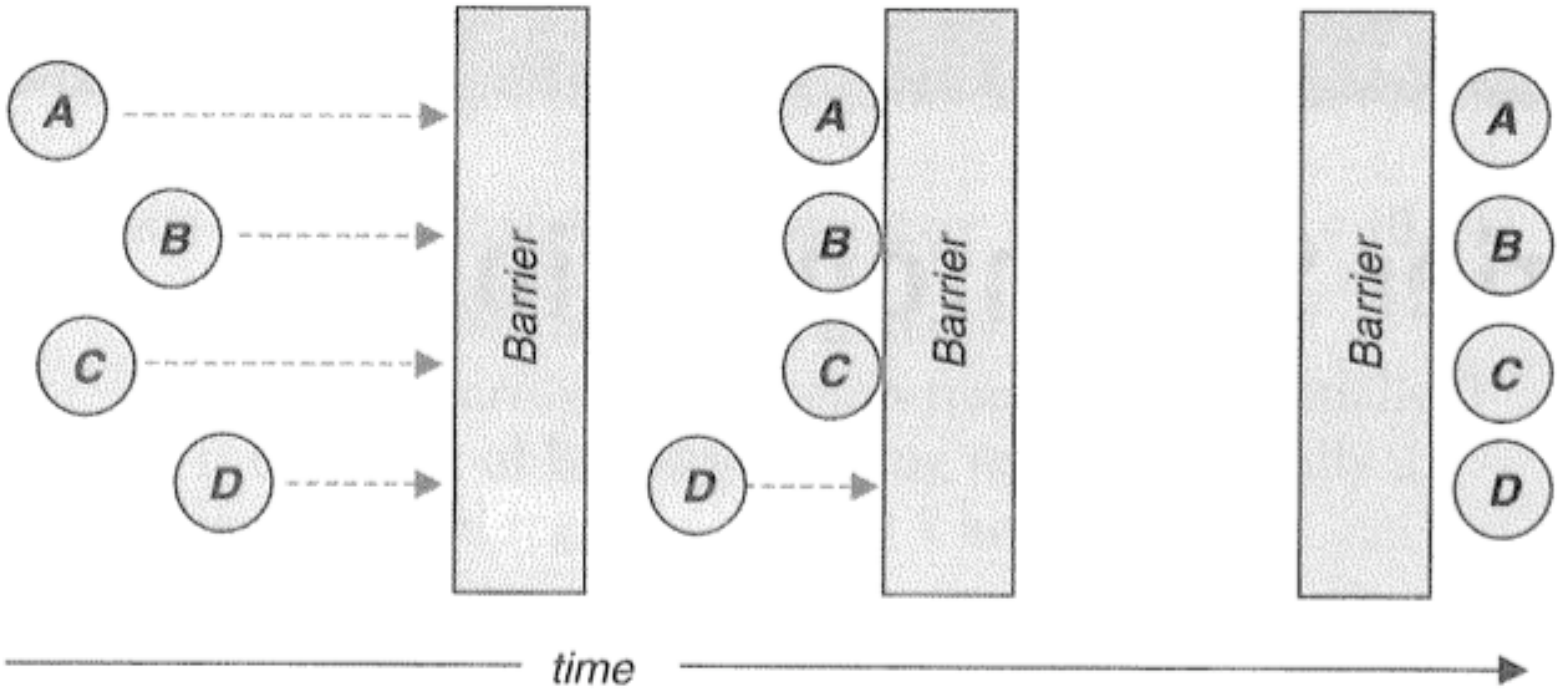
15 March 2017



Department of Computer Science, *Johns Hopkins University*

Barriers

- Allows a “synchronous” algorithm to run on asynchronous hardware



Simple Barrier

- Built on an atomic counter and atomic bits

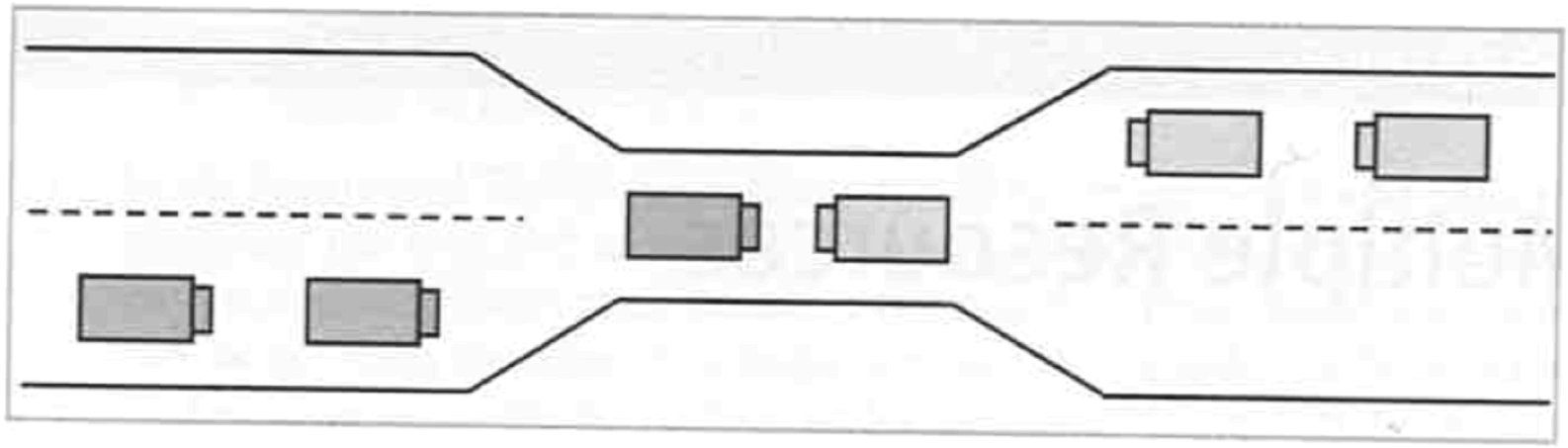
```
shared   counter: atomic counter ranges over {0, ..., n}, initially 0
         go: atomic bit, initial value is immaterial
local    local.go: a bit, initial value is immaterial

1 local.go := go                          /* remembers current value */
2 counter := counter + 1
                                  /* atomically increment the counter */
3 if counter = n then             /* last to arrive to the barrier */
4    counter := 0                           /* reset the barrier */
5    go := 1 - go                                 /* notify all */
6 else await(local.go ≠ go) fi            /* not the last to arrive */
```

Multiple Resources

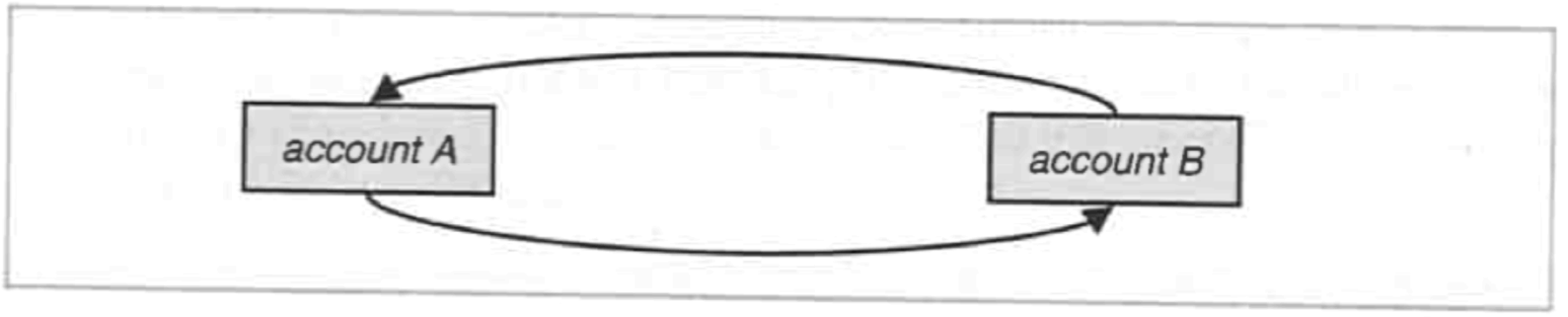
- To now, we have talked about deadlock freedom for mutual exclusive access to a single resource

- With multiple resources, we get deadlock even with deadlock-free access to each resource

Deadlock



(b)



(a)

Figure 7.1 Examples of deadlocks. (*a*) Transferring money between two accounts. (*b*) Bridge crossing: the resources are the two entrances.

Deadlock

- *Def'n:* A set of processes are deadlocked if each process in the set is waiting for an event that only another process in the set can cause

- Requirements (all four must hold simultaneously)
  - Mutual exclusion
  - Hold and wait
  - No preemption
  - Circular wait

- We'll do more on deadlock in MPI