# Lecture 9.3
# Fast Mutual Exclusion

EN 600.320/420

Instructor: Randal Burns

26 February 2018


Department of Computer Science, *Johns Hopkins University*

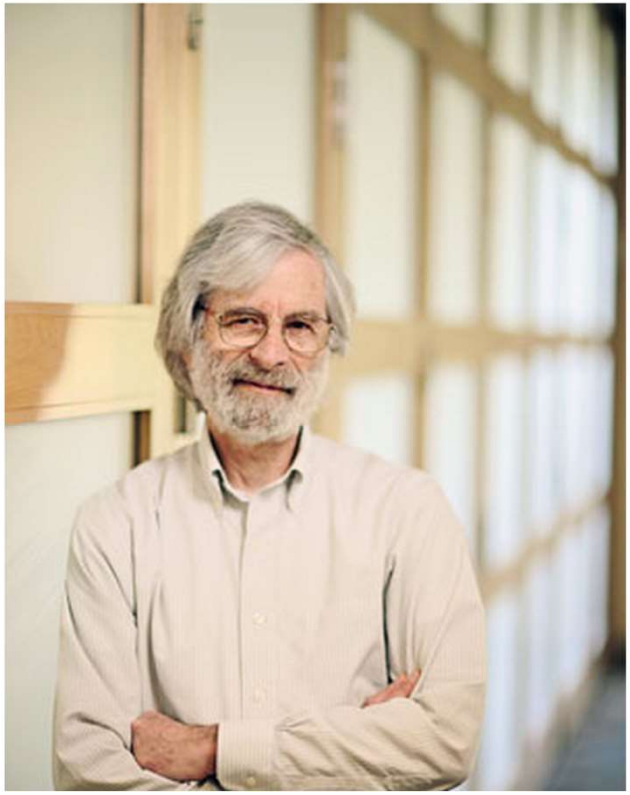
# Fast Mutual Exclusion

- Now let's scale this to *n* processes

## Leslie Lamport Receives Turing Award

Microsoft Research
March 18, 2014 6:00 AM PT

Like 1.4k Tweet 1,285



Leslie Lamport first began dabbling in computers while he was still in high school. Nothing too unusual about that—until you consider that this was in the mid-1950s. Lamport was attending the Bronx High School of Science in New York, and he and a friend used to scrounge around, looking for discarded vacuum tubes to build a digital circuit.

The path to greatness begins with baby steps, and for Lamport, a principal researcher with Microsoft Research, that teenage curiosity has yet to be quenched. Over the ensuing decades, he has become a veritable legend in computing circles. His work in the theory of distributed computing is foundational. His 1978 paper *Time, Clocks, and the Ordering of Events in a Distributed System*

Publications

- *Time, Clocks, and the Ordering of Events in a Distributed System*
- *The Maintenance of Duplicate Databases*
- *A New Solution of Dijkstra's Concurrent Programming Problem*
- *The Byzantine Generals Problem*
- *The Part-Time Parliament*
- *Paxos Made Simple*
- *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*

Awards

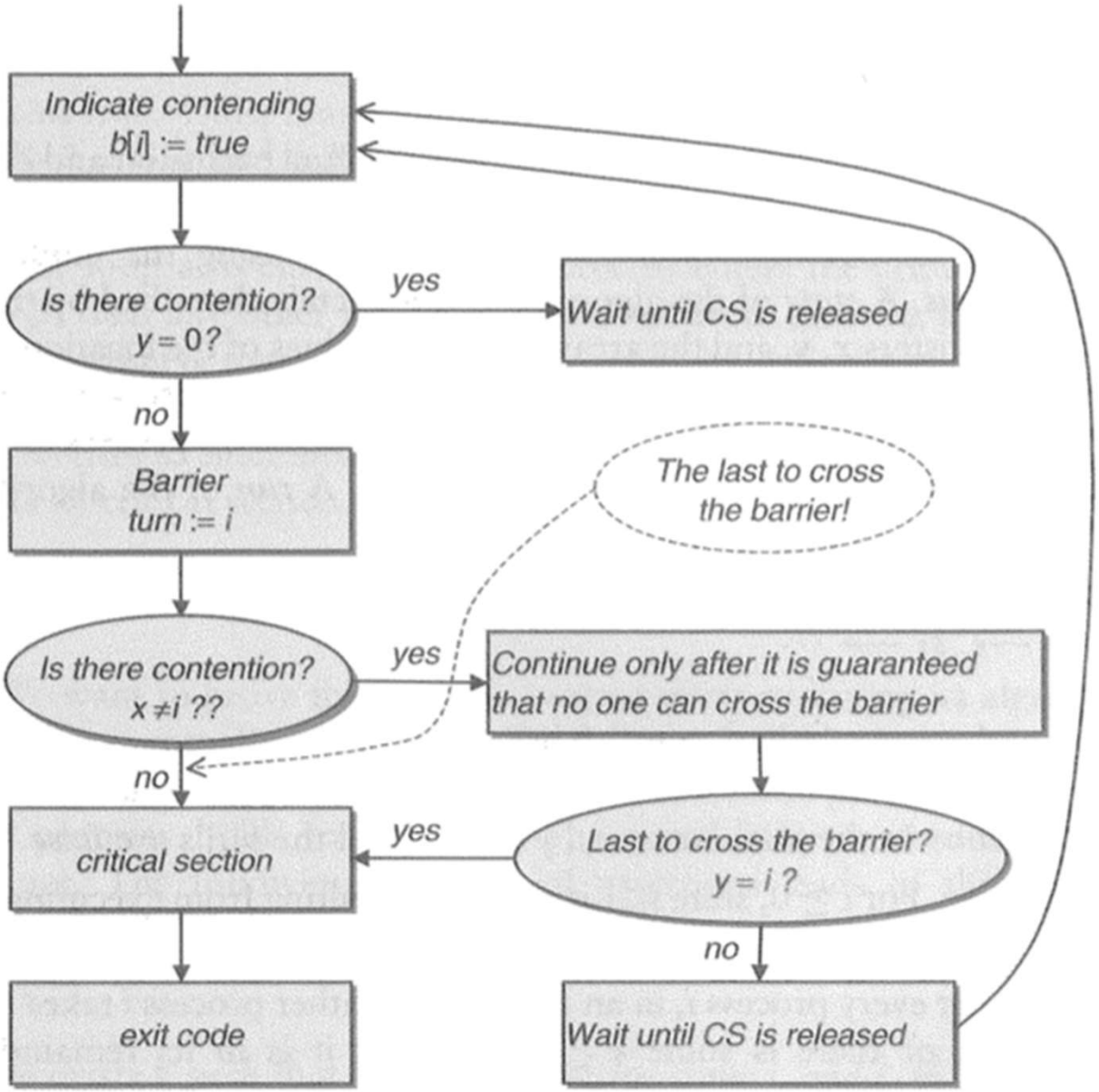
- A.M. Turing Award

# Fast Mutual Exclusion

- Lamport 1987

```
1  start:  b[i] := true;
2          x := i;
3          if y ≠ 0 then b[i] := false;
4                        await y = 0;
5                        goto start fi;
6          y := i;
7          if x ≠ i then b[i] := false;
8                        for j := 1 to n do await ¬b[j] od;
9                        if y ≠ i then await y = 0;
10                                     goto start fi fi;
11         critical section;
12         y := 0;
13         b[i] := false
```

# Fast Mutual Exclusion




# Fast MutEx Properties

- Mutual exclusion/deadlock freedom
- Contention free overhead=  7 accesses
- **Starvation is possible!** (of any process)
  - Unbounded wait times

# Practical Concerns

- None of the algorithms are used in practice
  - Too simple
  - H/W support
- But demonstrate fundamental tradeoffs
  - Space (# shared registers), speed, fairness (bounded waiting)
- All of these algorithms rely on *atomic* registers
  - Not available in distributed memory machines, which leads to whole new families of protocols