## 3/13 - Queue Locks, cont.

First remote lecture:

- This will be recorded - not on Panopto, but we will post the zoom recording
- Q & A at the end

Queue locks

- Recall: worked with two threads, but things went horribly wrong with 3
    - Mutual exclusion was failing
- What's going wrong?
    - Our model? The *algorithm!?*
- As it turns out, there's something wrong with Anderson's queue lock itself.
    - Violates mutual exclusion for with 3 threads but works for 4.
    - Question? Will it work for 5? 6? Is the issue odd/even?
    - Also breaks for 5 or 6 threads :(
    - Maybe the algorithm works for powers of 2?
        - If we run for 8 threads, we don't have a safety violation
    - Why are powers of 2 important here? (really things that divide 256)
        - There's an array of size 256 that represents the queue lock
            - When 'next' = 256, we wrap around to 0
        - The problem is that a thread sees they can go, but another thread is still in the critical section
- What did we show?
    - Anderson's Queue Lock works only when the space we have divides the number of threads we have
    - Notice - we only needed to model it for a small size (byte vs. int32/int64)
    - We can still discover the bug/precondition for it to work!
- This demonstrates what Spin is good for:
    - Very long traces (finding long/unbounded traces)
    - Small state (e.g. next in the queue lock)