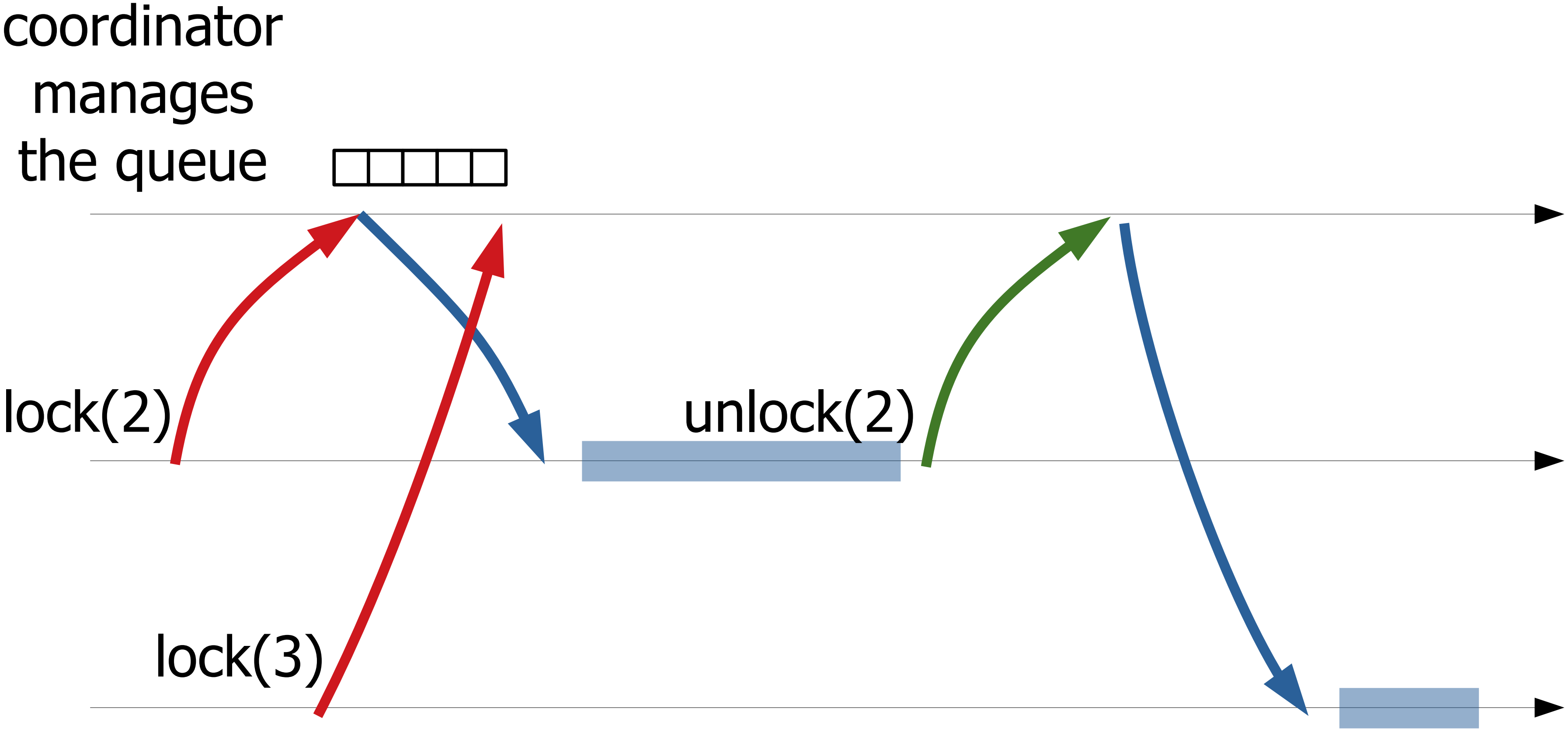
Distributed Mutual Exclusion

- Implement lock()/unlock() primitives in a distributed system
- Properties:
  - No two processes concurrently in the critical section
  - Some willing process eventually enters the critical section (weak fairness)
  - All willing processes eventually enter the critical section (strong fairness)

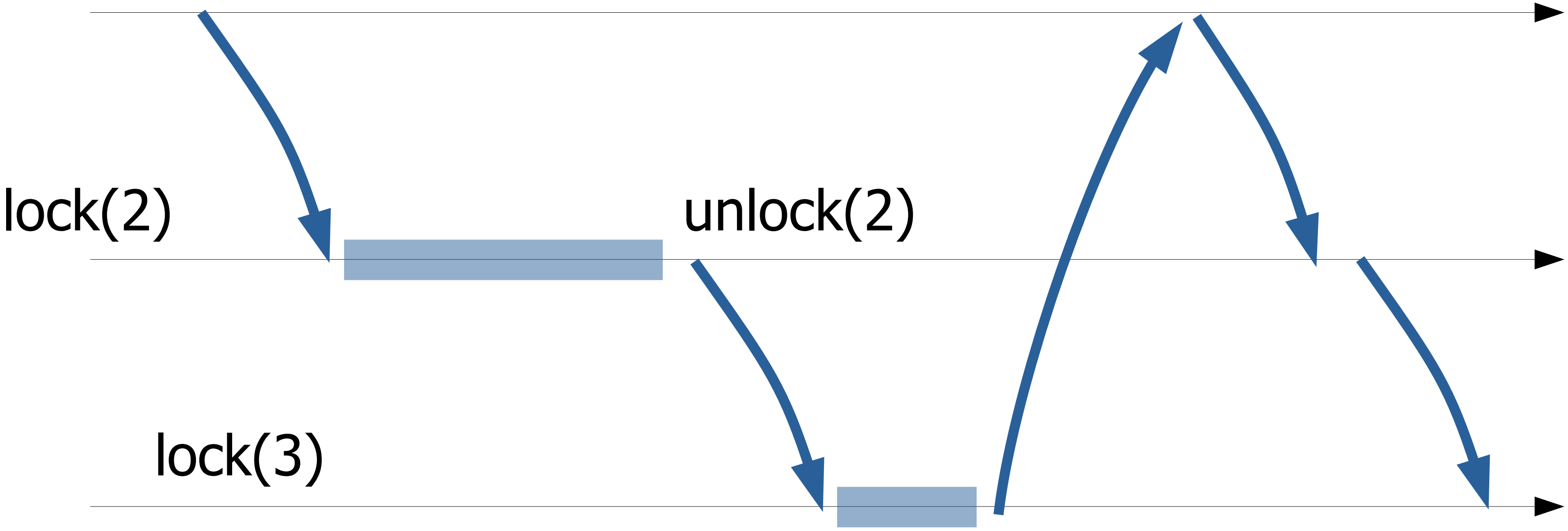
Centralized



- At least one round-trip to enter
- Coordinator handles all messages (bottleneck)

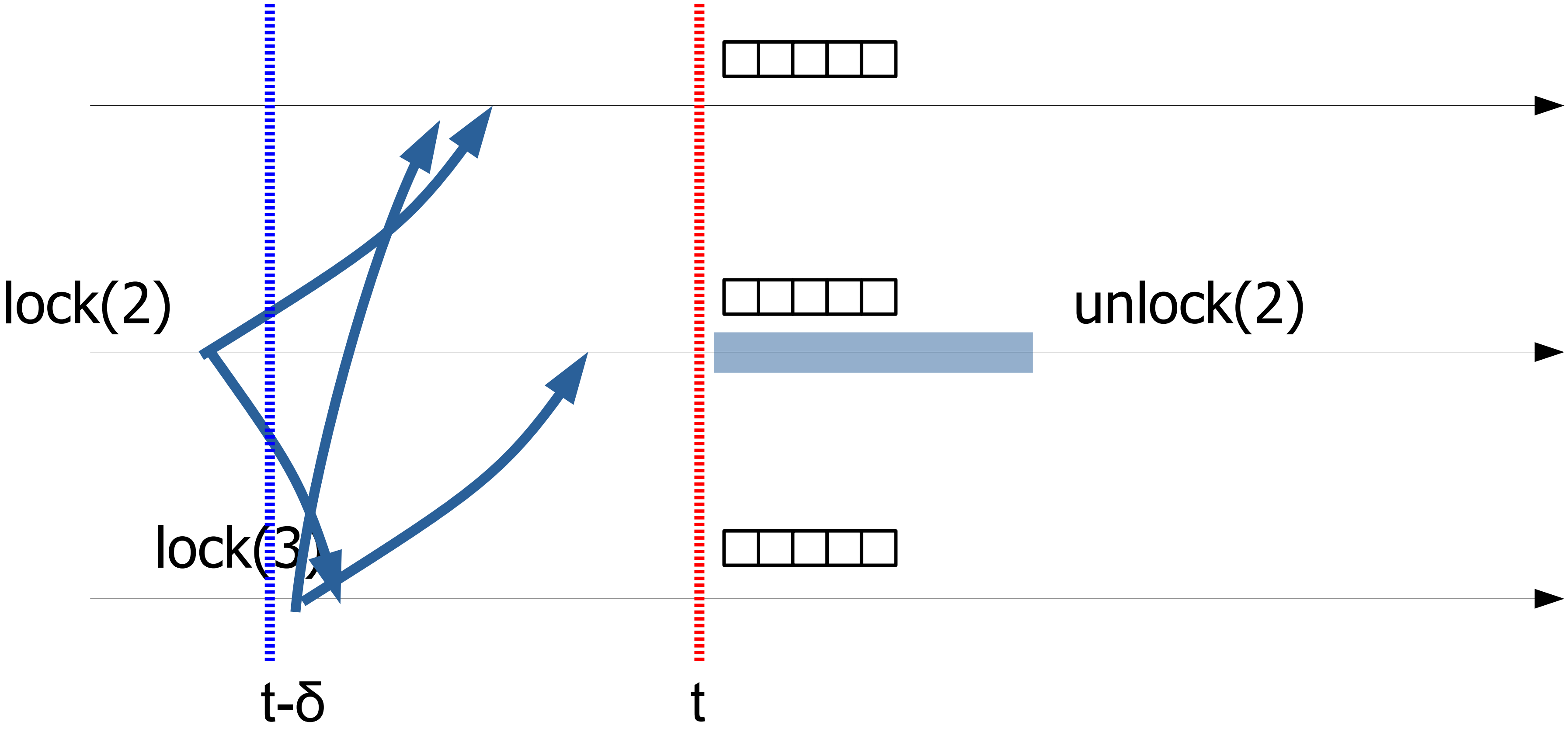
Ring

No explicit queue!



- N/2 hops to enter
- Distributed load, but not quiescent

Physical time

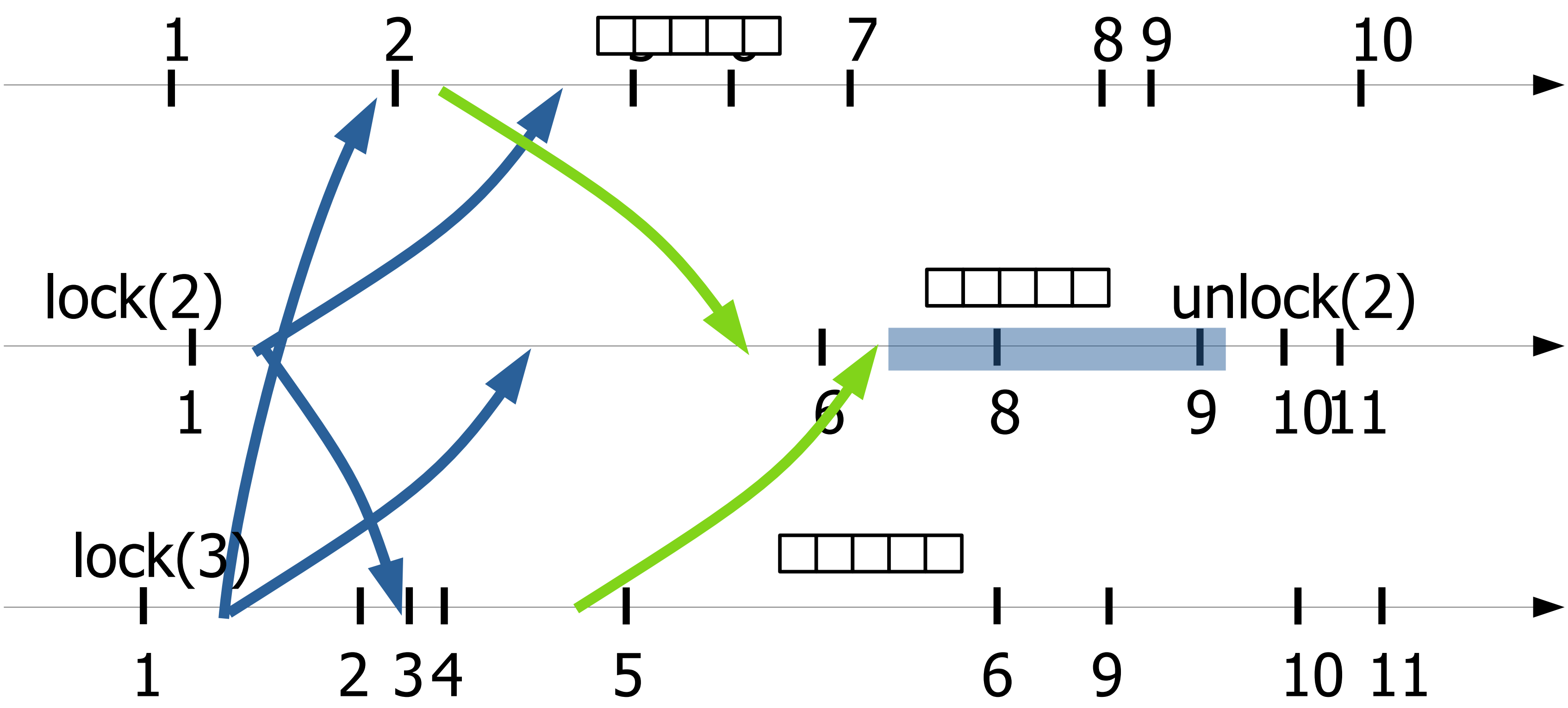


- δ delay to enter
- Distributed load, but synchronous

Physical time

- Algorithm:
  - At t, consider all requests up to t-δ
  - Order by timestamp, break ties by process id
- δ delay to enter
- Distributed load, but synchronous

Logical time



Logical time

- Algorithm:
  - ri[j] latest timestamp from j at i
  - Consider requests with t <= min(ri[j], for all j)
  - Order by timestamp, break ties by process id
- 1 hop to enter, if processes send messages frequently
- Distributed load, blocks if a process stops

Replicated state machine

- Note that all processes keep copies of the queue waiting for the lock
- Can be generalized for any data structure and deterministic computation
    - Replicated state machine