Read-Log-Update

A Lightweight Synchronization Mechanism for Concurrent Programming

Alexander Matveev (MIT) Nir Shavit (MIT and TAU) Pascal Felber (UNINE) Patrick Marlier (UNINE)

Nian Liu
nianliu@sjtu.edu.cn

Concurrent Data Structure

- Multicore environment => Need concurrent data structure
- 2 key concerns for concurrent data structure:

Concurrent Data Structure

- Multicore environment => Need concurrent data structure
- 2 key concerns for concurrent data structure:
  - **Unsynchronized** traversals

    90% of the time is spent traversing data

Concurrent Data Structure

- Multicore environment => Need concurrent data structure
- 2 key concerns for concurrent data structure:
  - **Unsynchronized** traversals

    90% of the time is spent traversing data
  - **Multi-location** atomic updates

    Hide race conditions from programmers

Read-Copy Update

- Introduced by McKenney
- Linux kernel has 6,500+ RCU calls

  - **Unsynchronized** traversals
  - **Multi-location** atomic updates

Read-Copy Update

- Introduced by McKenney
- Linux kernel has 6,500+ RCU calls
- **No overhead** for readers

  - **Unsynchronized** traversals ✓
  - **Multi-location** atomic updates

Read-Copy Update

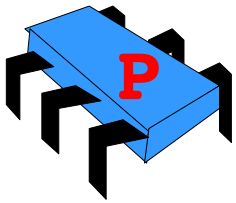
- Introduced by McKenney
- Linux kernel has 6,500+ RCU calls
- **No overhead** for readers
- Allow **only single pointer update**
  - **Unsynchronized** traversals ✓
  - **Multi-location** atomic updates ✗

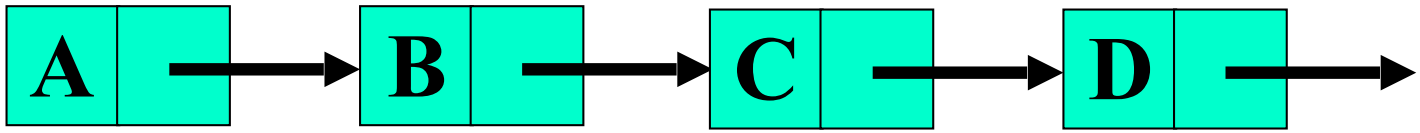
2 Key Technique in RCU

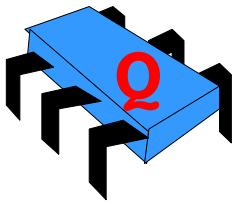
- RCU-Duplication
  - Duplicate objects & modify copies
  - Commit through updating a single pointer
- RCU-Epoch
  - Wait for all previous readers to finish
  - Safe to deallocate old objects

Read-Copy Update

Update C



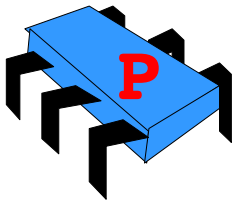




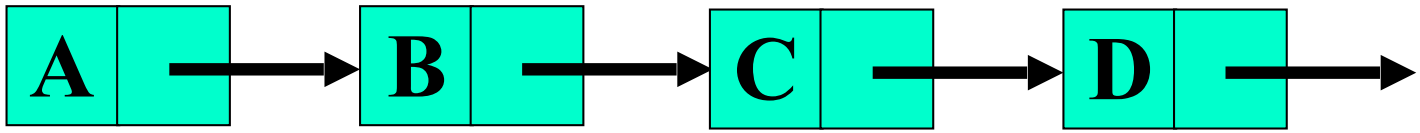
Lookup C

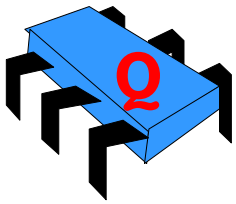
Read-Copy Update

Update C



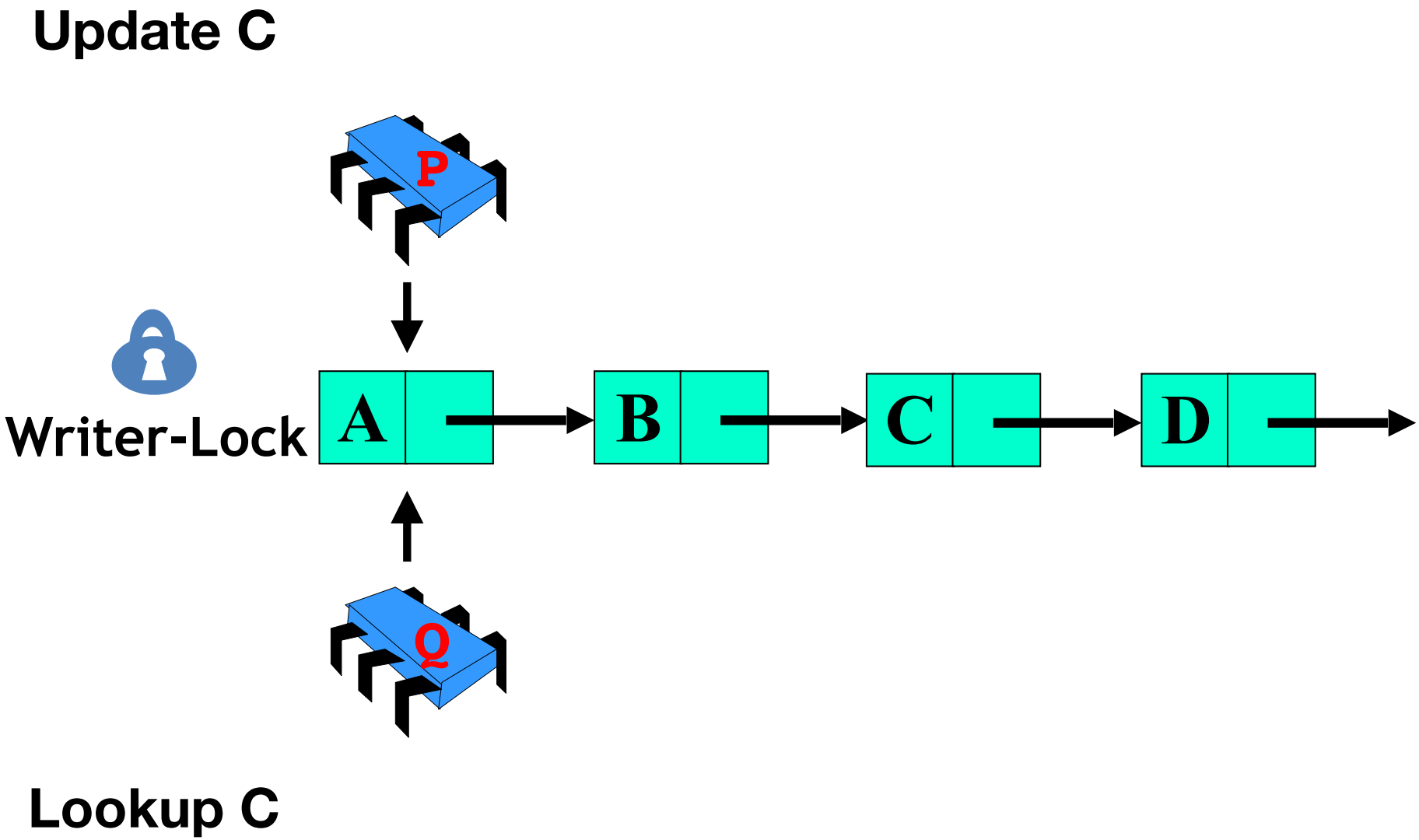
Writer-Lock





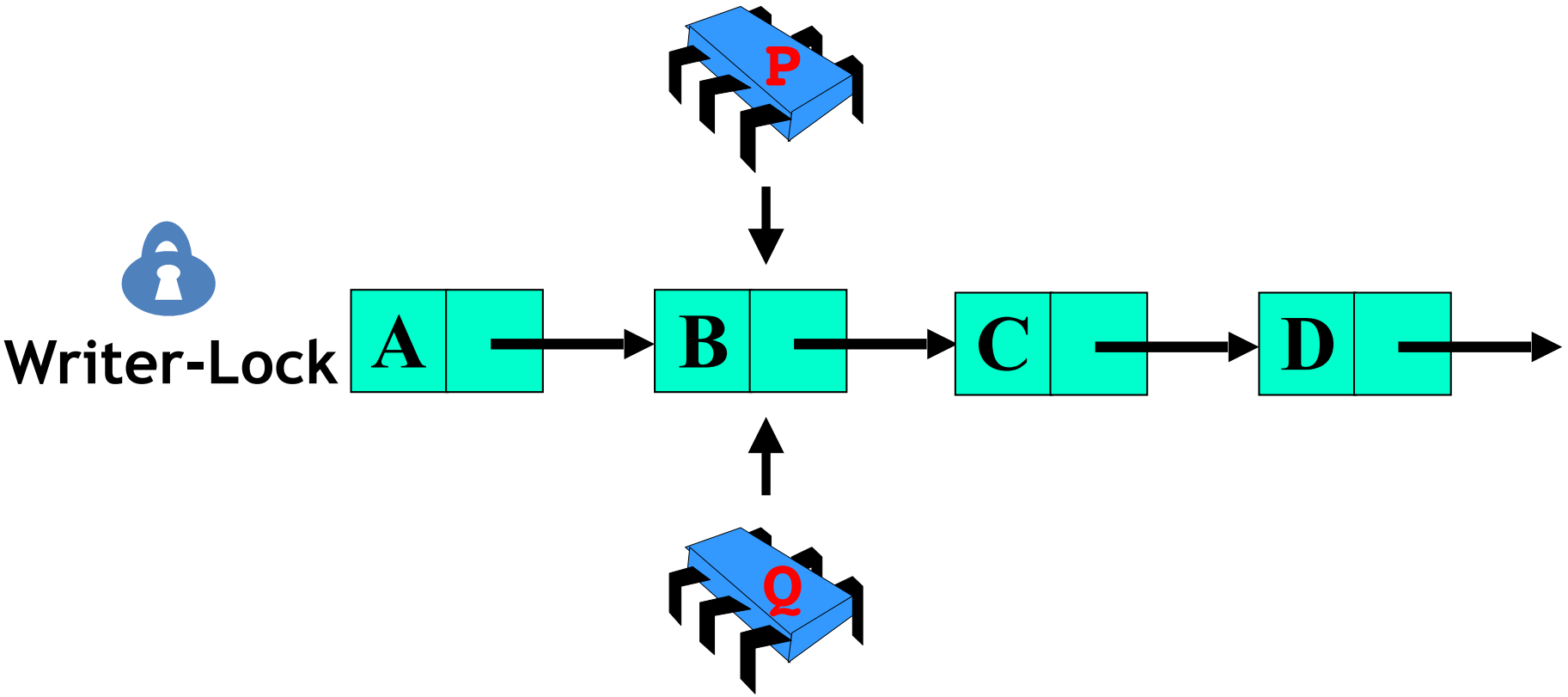
Lookup C

Read-Copy Update



Read-Copy Update

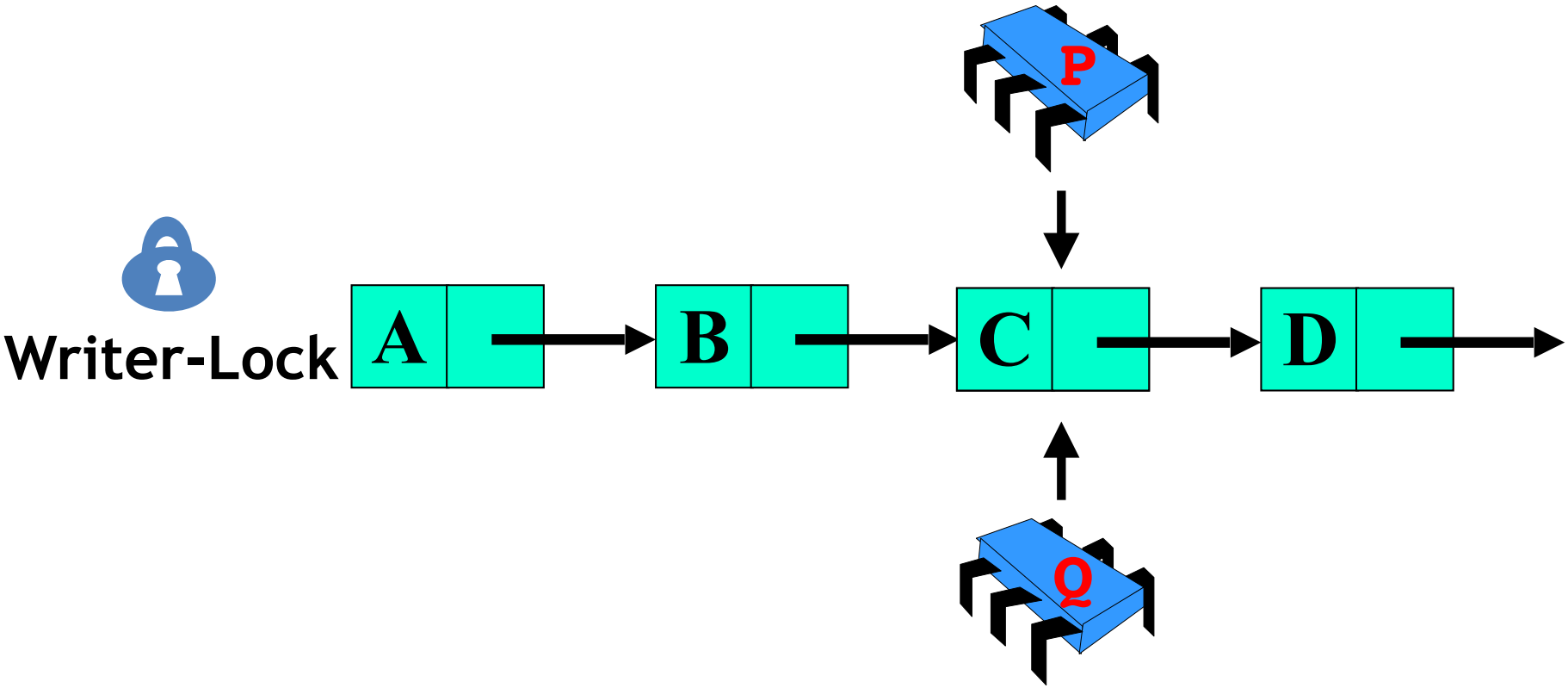
Update C



Lookup C

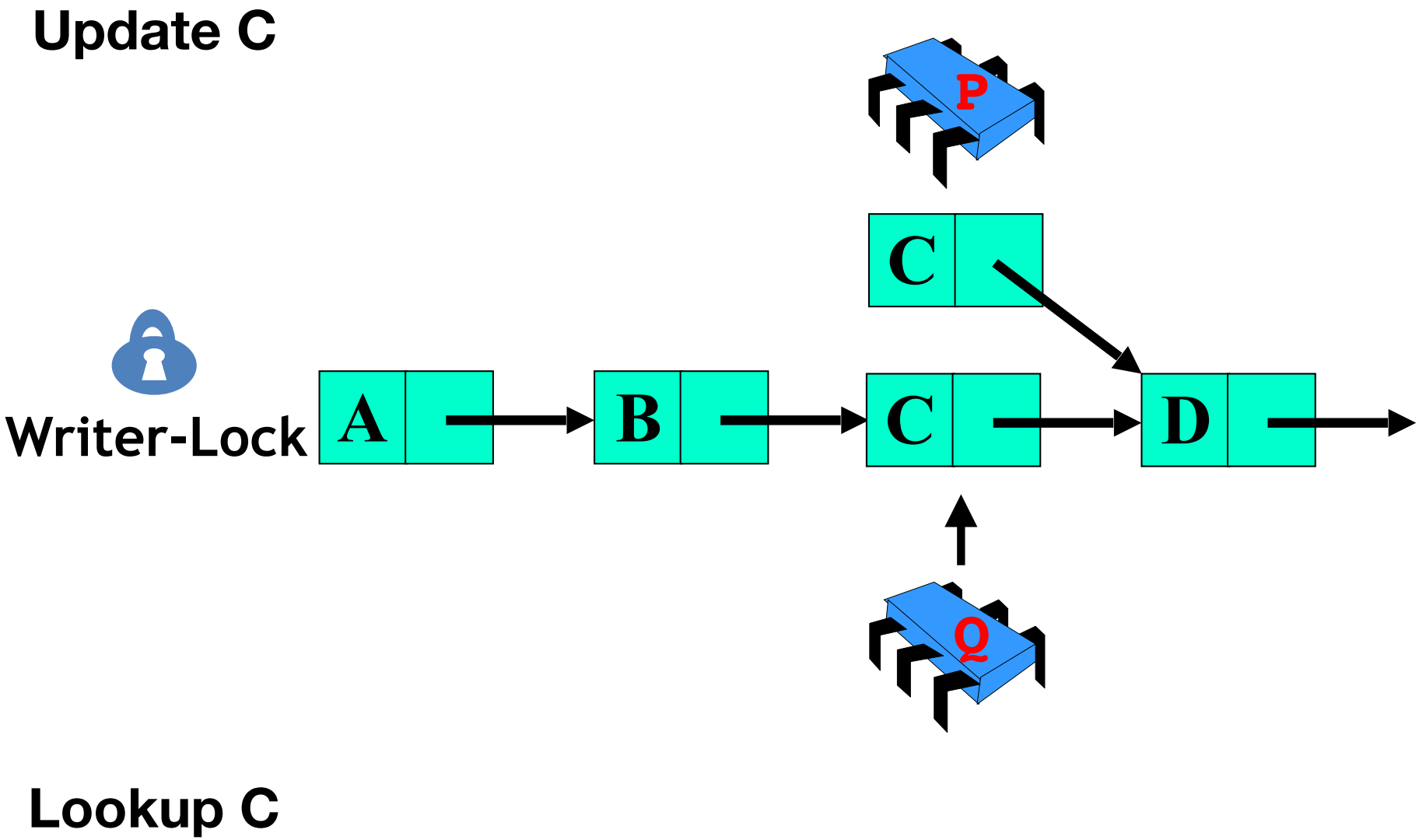
Read-Copy Update

Update C

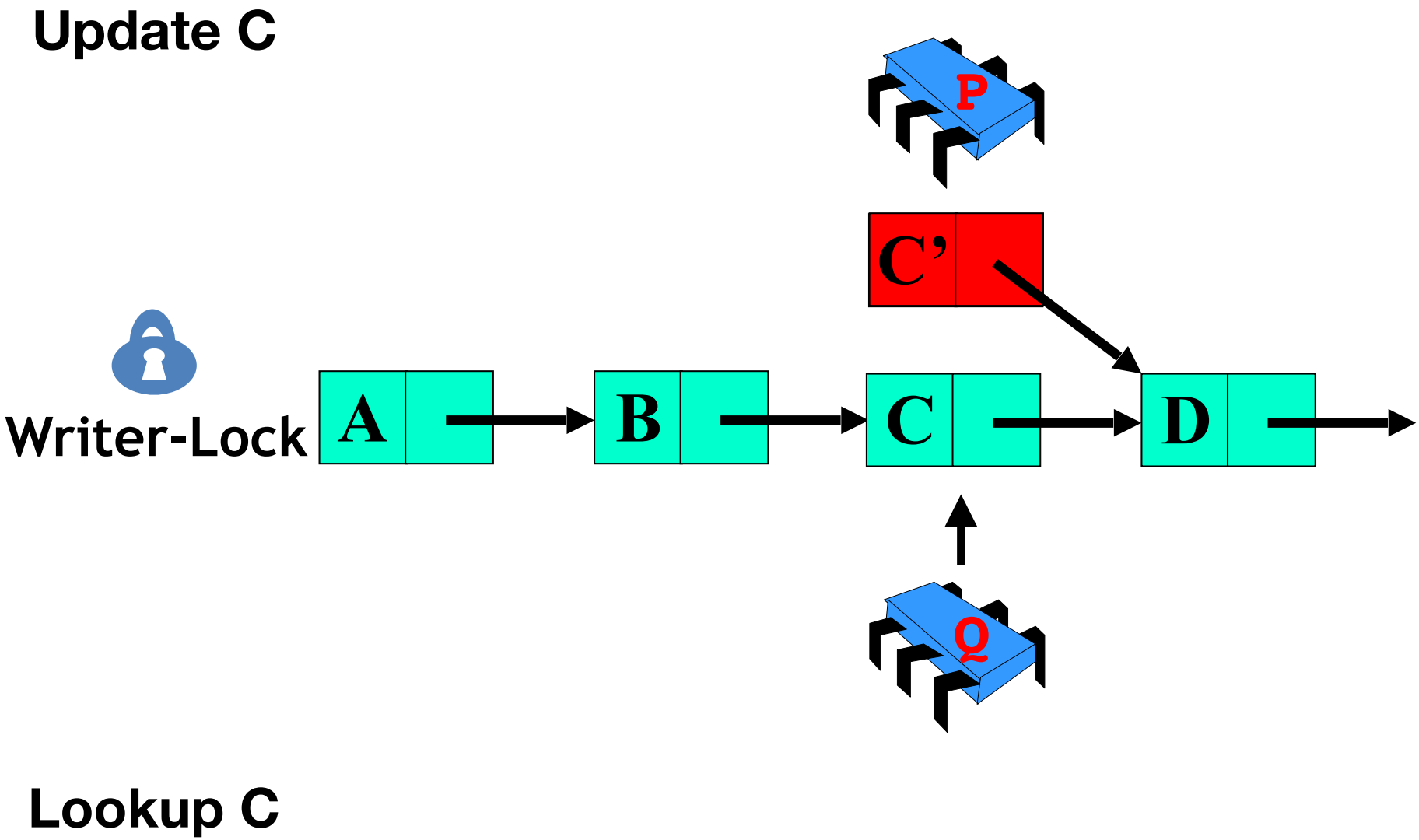


Lookup C

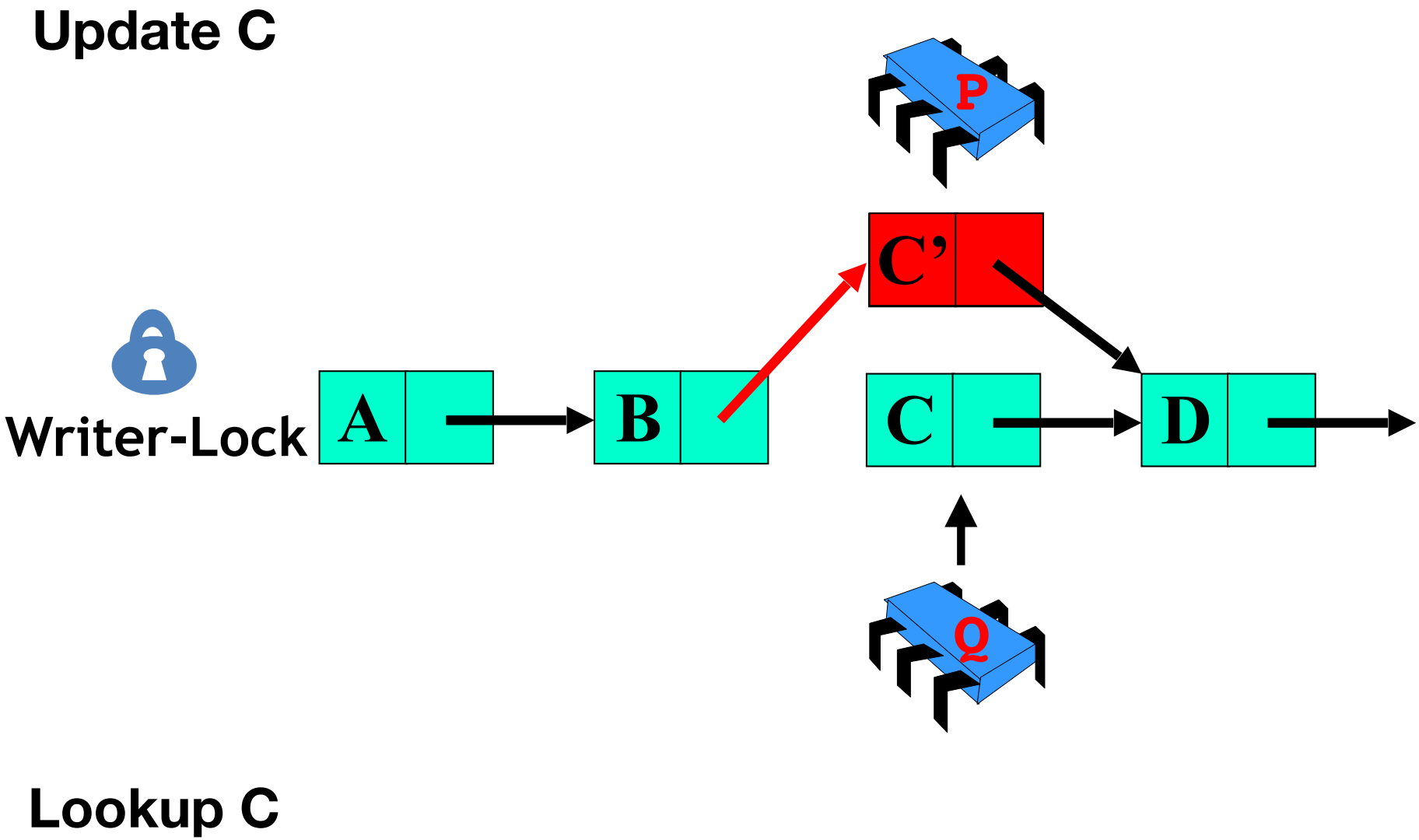
Read-Copy Update



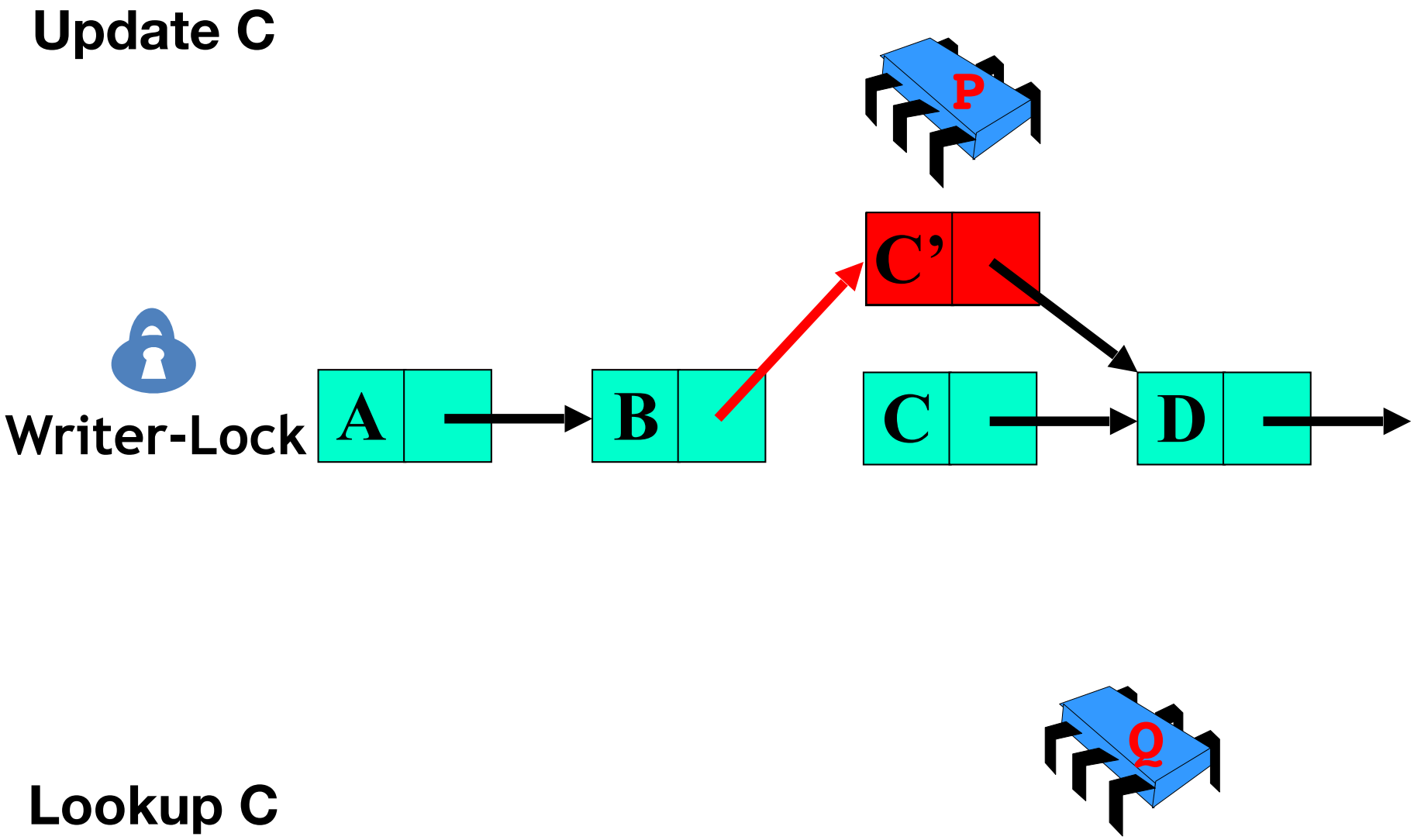
Read-Copy Update



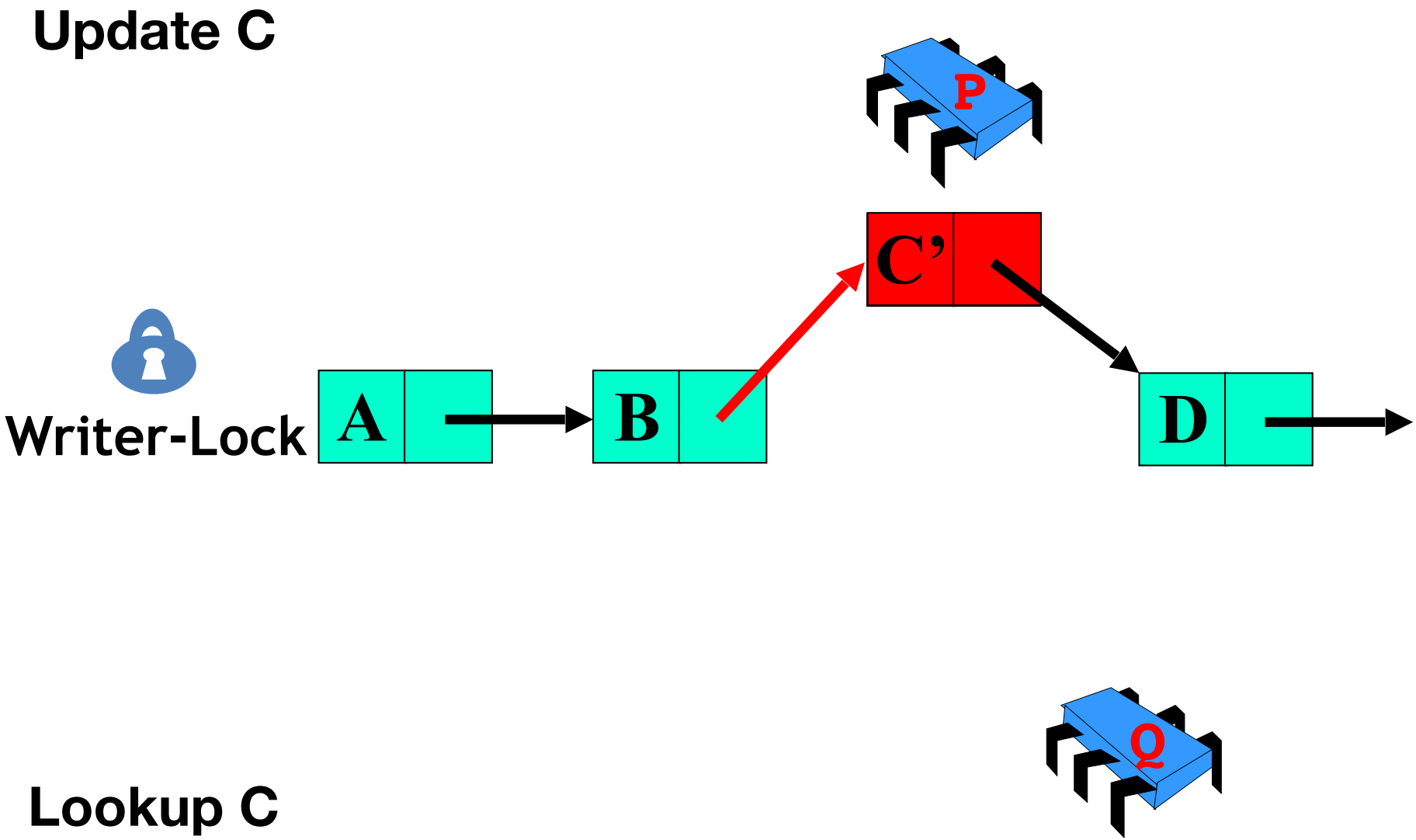
Read-Copy Update



Read-Copy Update

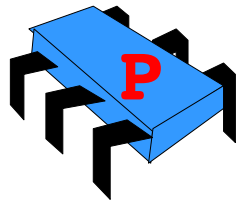


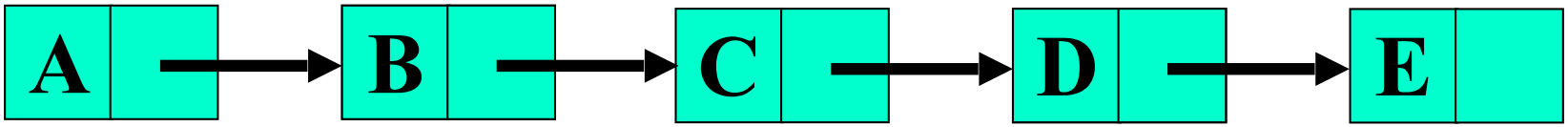
Read-Copy Update

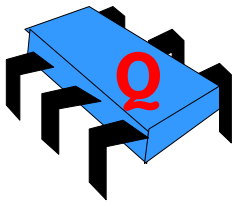


Problems with RCU

Update B & D simultaneous



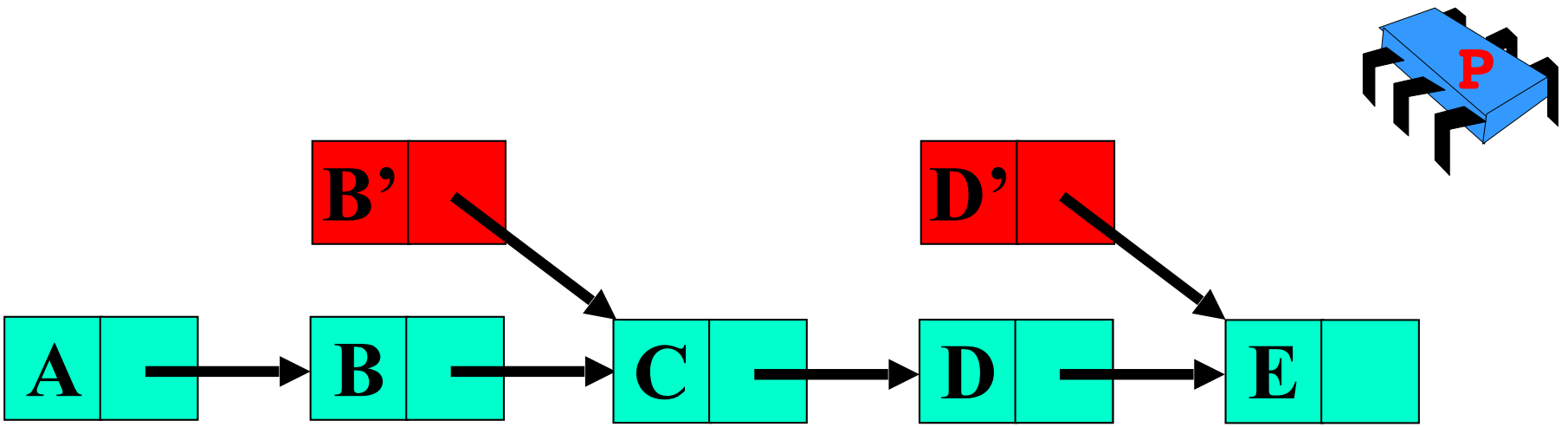


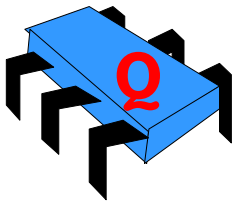


Lookup B & D

Problems with RCU

Update B & D simultaneous

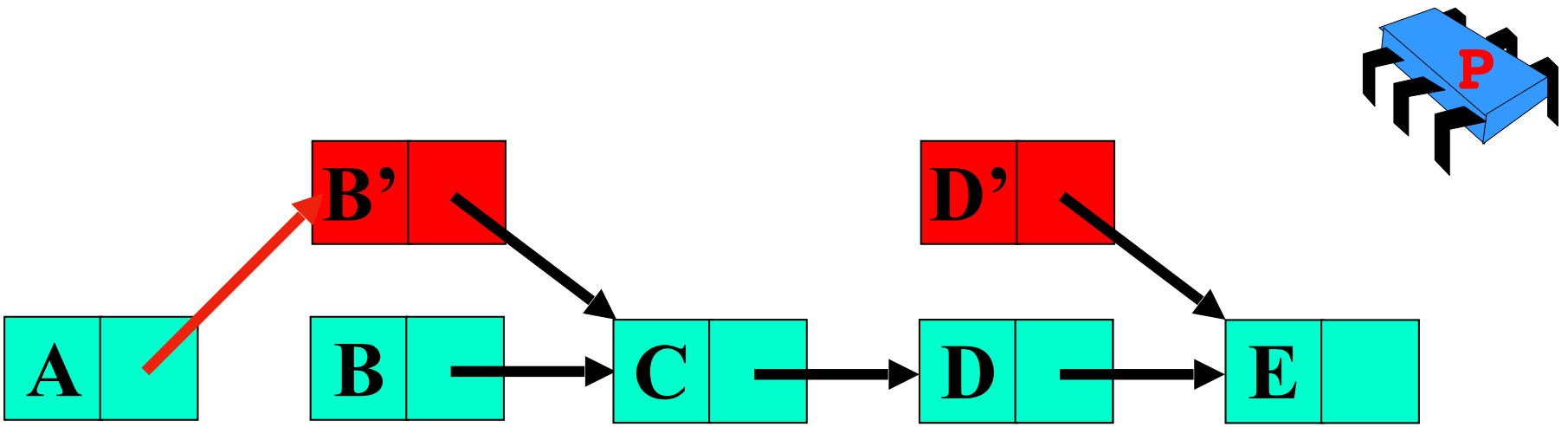


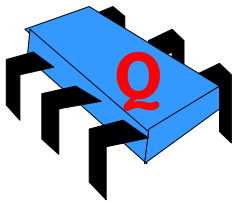


Lookup B & D

Problems with RCU

Update B & D simultaneous

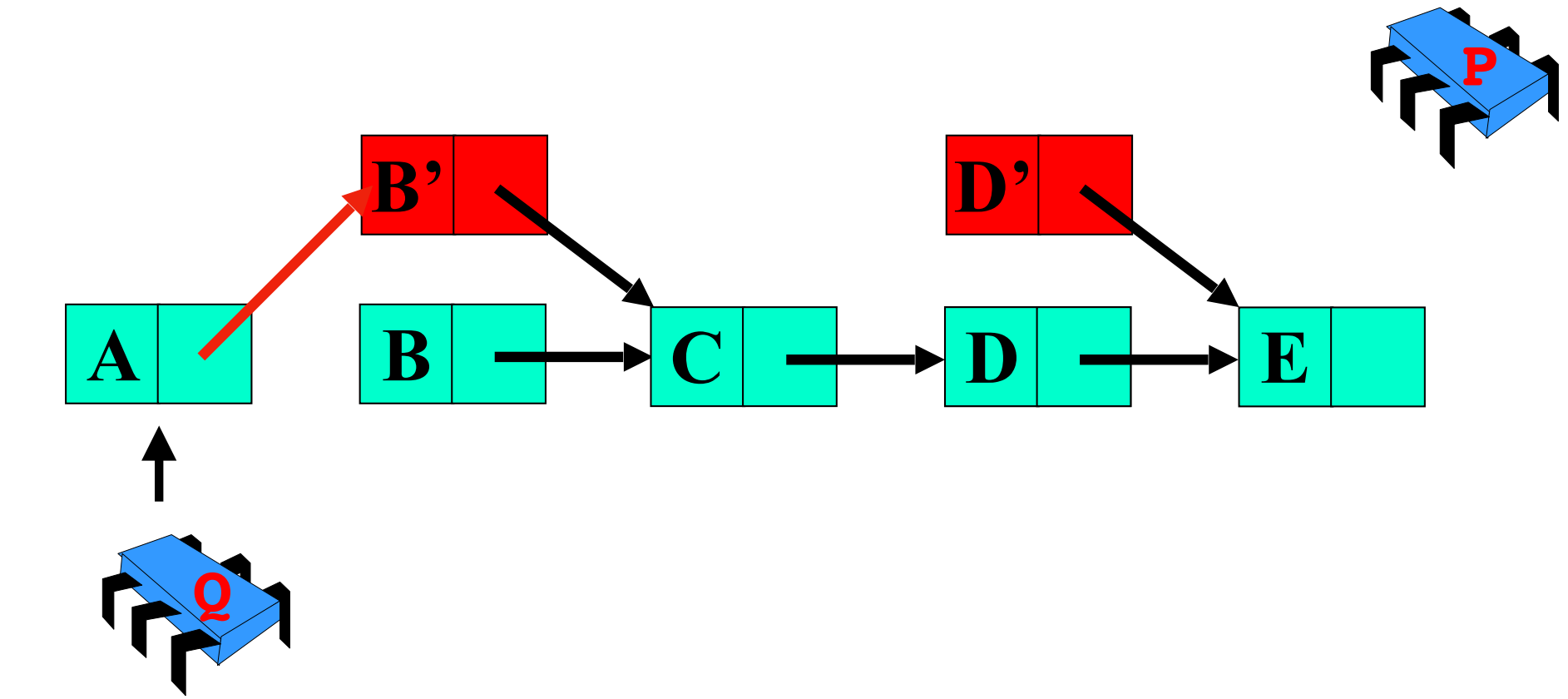




Lookup B & D

Problems with RCU

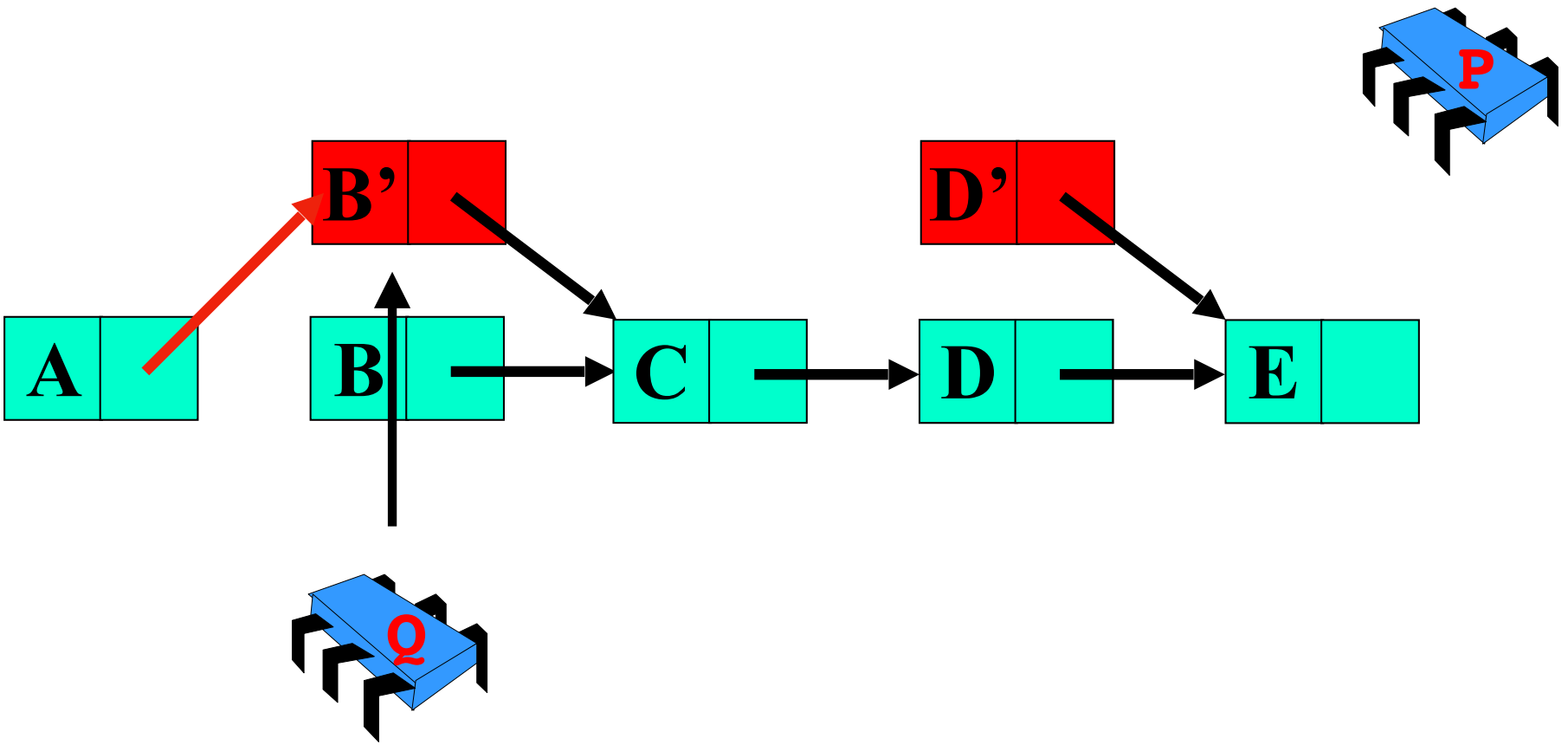
Update B & D simultaneous



Lookup B & D

Problems with RCU

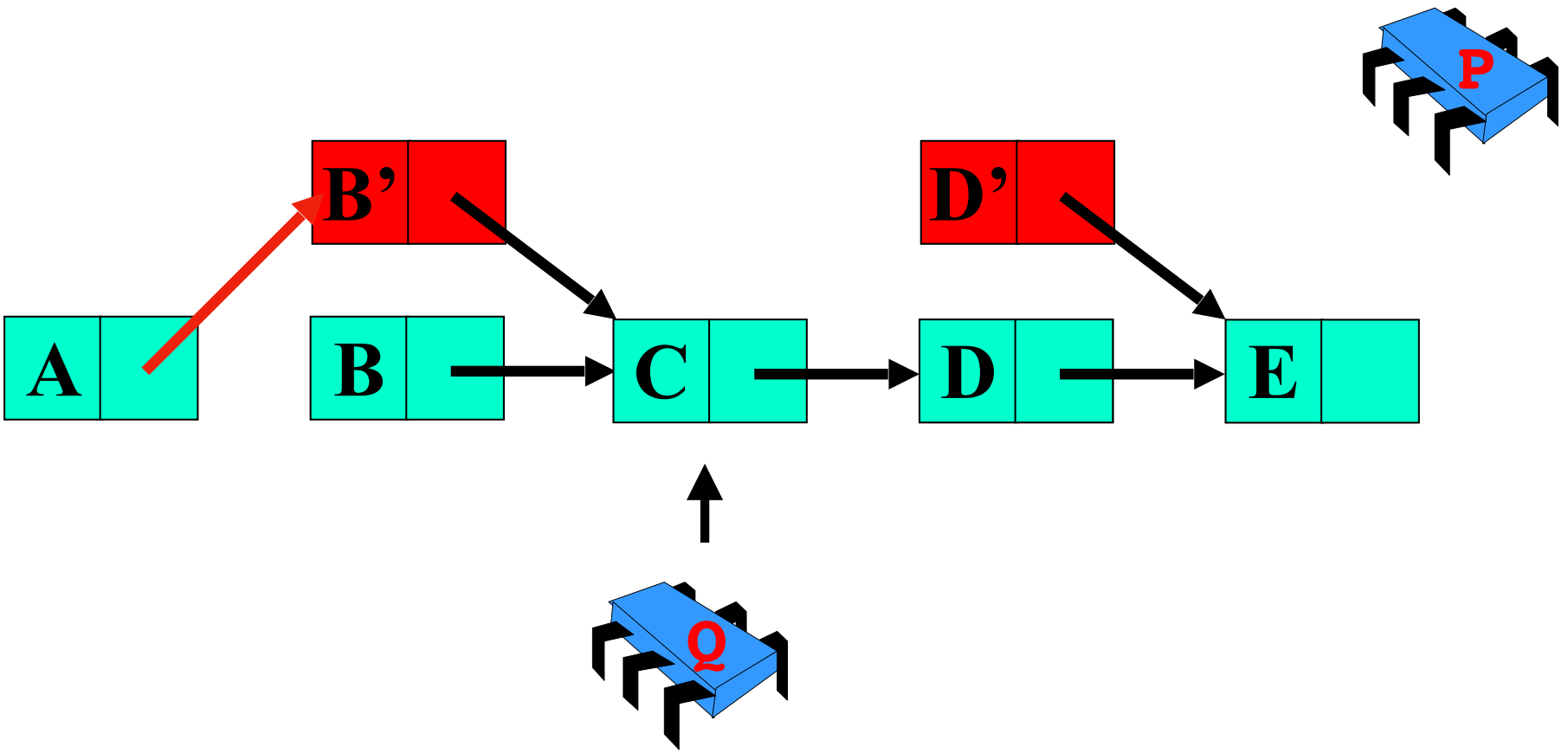
Update B & D simultaneous



Lookup B & D

Problems with RCU

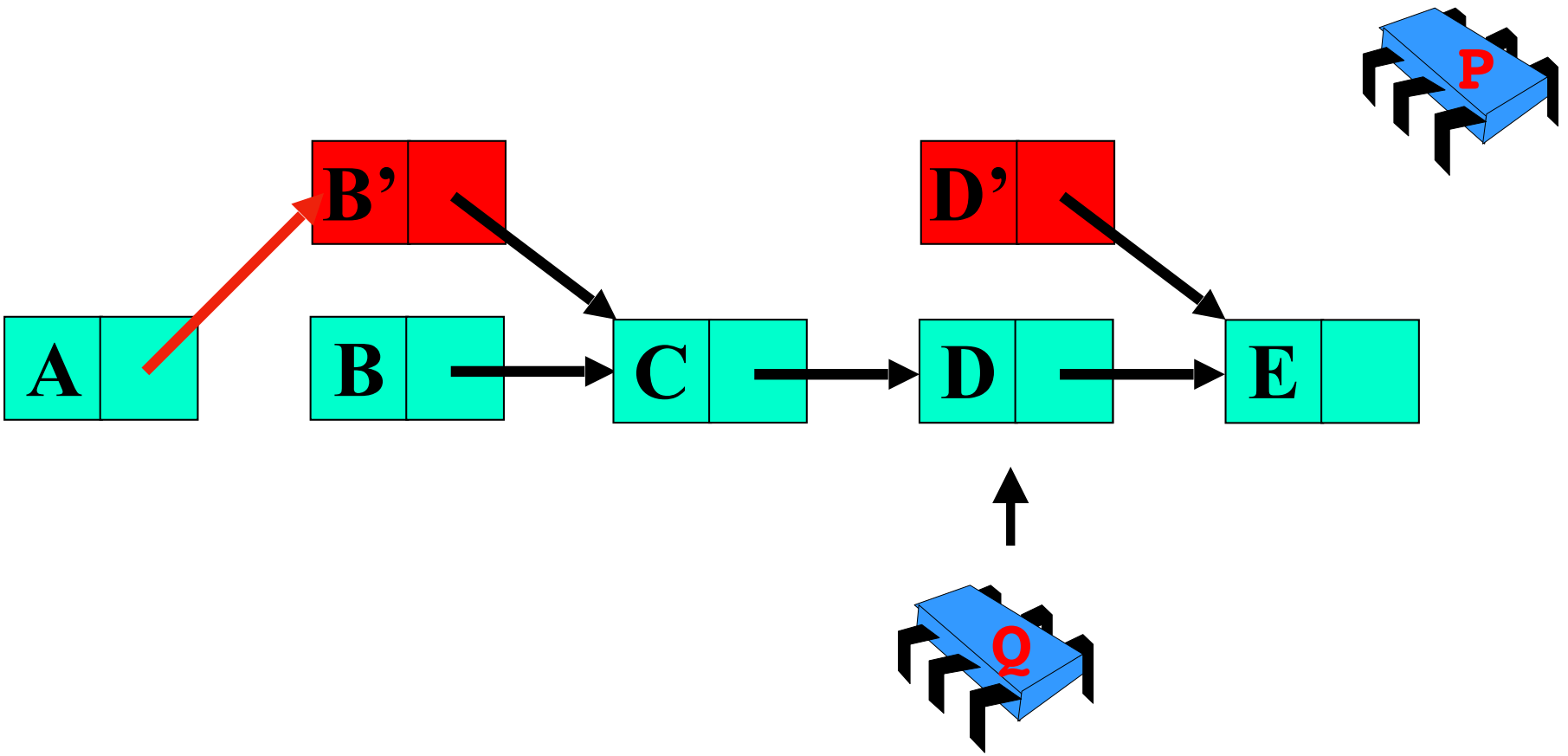
Update B & D simultaneous



Lookup B & D

Problems with RCU

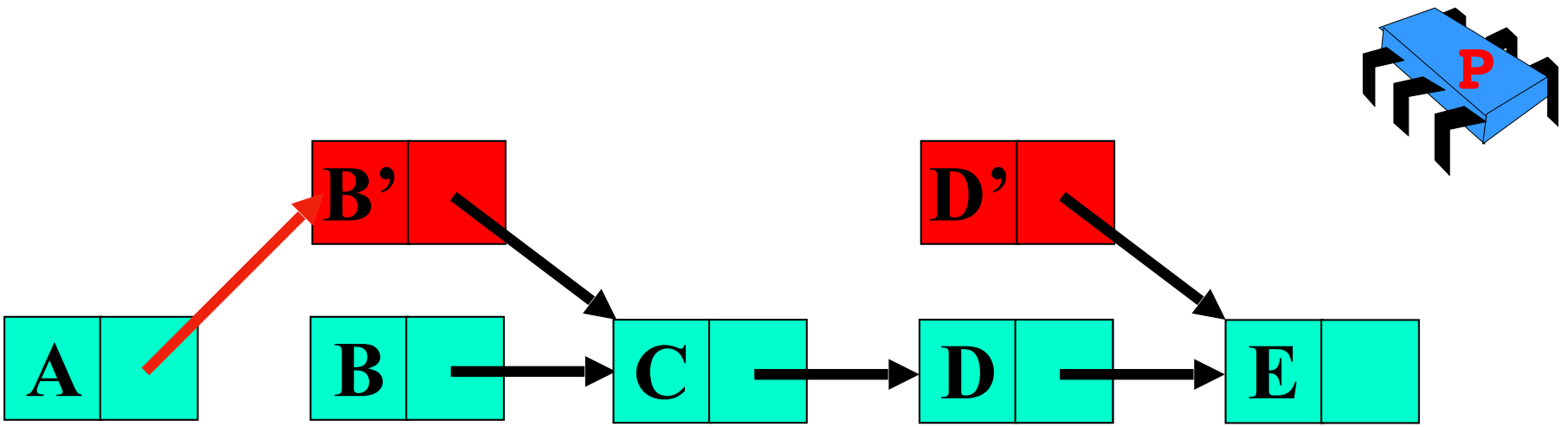
Update B & D simultaneous



Lookup B & D

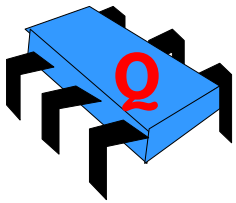
Problems with RCU

Update B & D simultaneous



Read B' and D

Inconsistent



Lookup B & D

Problems with RCU

- RCU is not capable of dealing with complex data structure

  Double link list / Graph / Resizable hash table

Problems with RCU

- RCU is not capable of dealing with complex data structure

  Double link list / Graph / Resizable hash table

- RCU does not provide concurrency among writers

  A **global writer lock** will become a bottleneck

Problems with RCU

- RCU is not capable of dealing with complex data structure

  Double link list / Graph / Resizable hash table

- RCU does not provide concurrency among writers

  A **global writer lock** will become a bottleneck

- RCU writers have to wait for all readers to complete

  Long epoch will **block** the writer

Introducing RLU

Introducing RLU

- RCU is not capable of dealing with complex data structure

  Update pointer => Commit log by updating **global clock**

- RCU does not provide concurrency among writers

- RCU writers have to wait for all readers to complete

Introducing RLU

- RCU is not capable of dealing with complex data structure

  Update pointer => Commit log by updating **global clock**

- RCU does not provide concurrency among writers

  Fine-grained Lock for each element

- RCU writers have to wait for all readers to complete

Introducing RLU

- RCU is not capable of dealing with complex data structure

  Update pointer => Commit log by updating **global clock**

- RCU does not provide concurrency among writers

  Fine-grained Lock for each element

- RCU writers have to wait for all readers to complete

  Deferral mechanism

Introducing RLU

- RCU is not capable of dealing with complex data structure

  Update pointer => Commit log by updating **global clock**

- RCU does not provide concurrency among writers

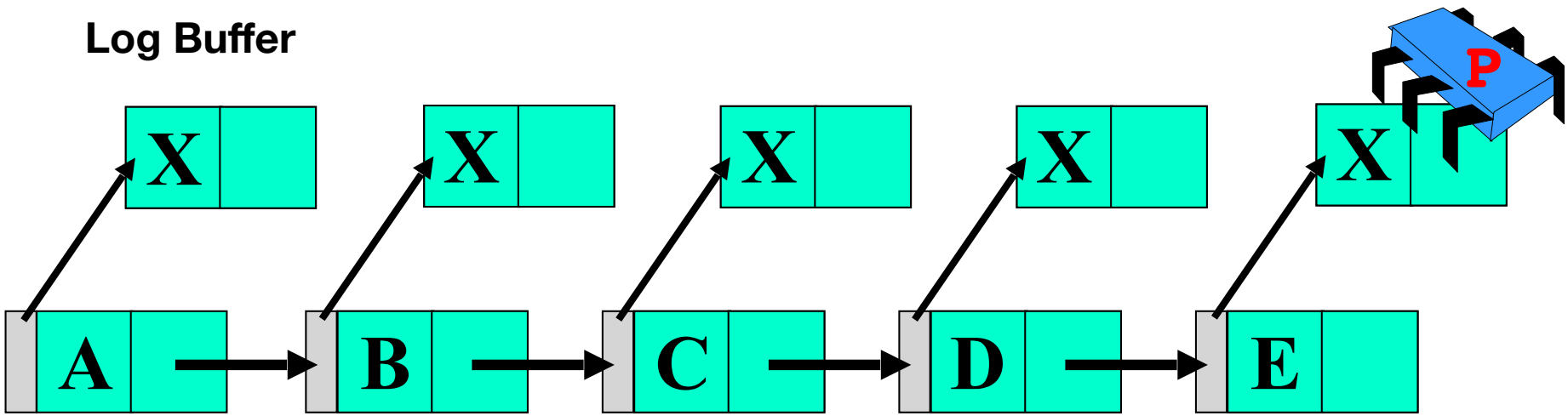
  Fine-grained Lock for each element

- RCU writers have to wait for all readers to complete

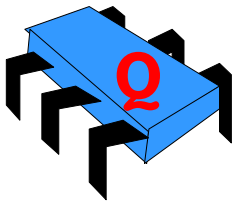
  Deferral mechanism

Read Log Update

Update B & D simultaneous



RLU header



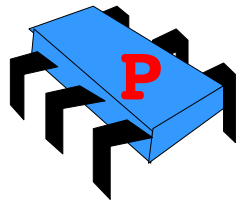
Lookup B & D

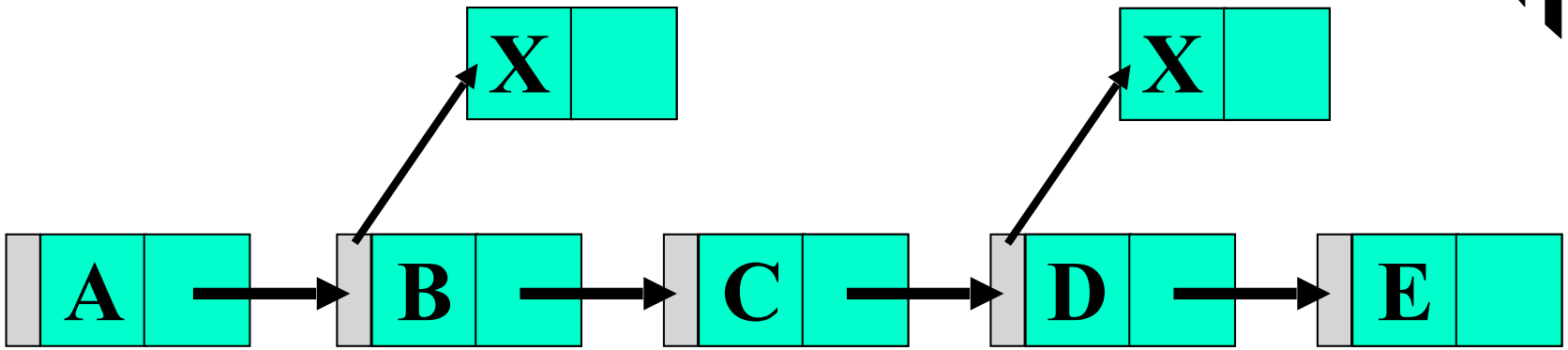
Read Log Update

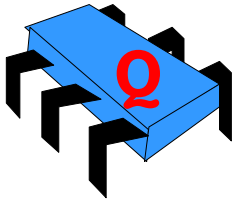
Update B & D simultaneous

Writer
Timestamp = ∞

Global
Timestamp = 22







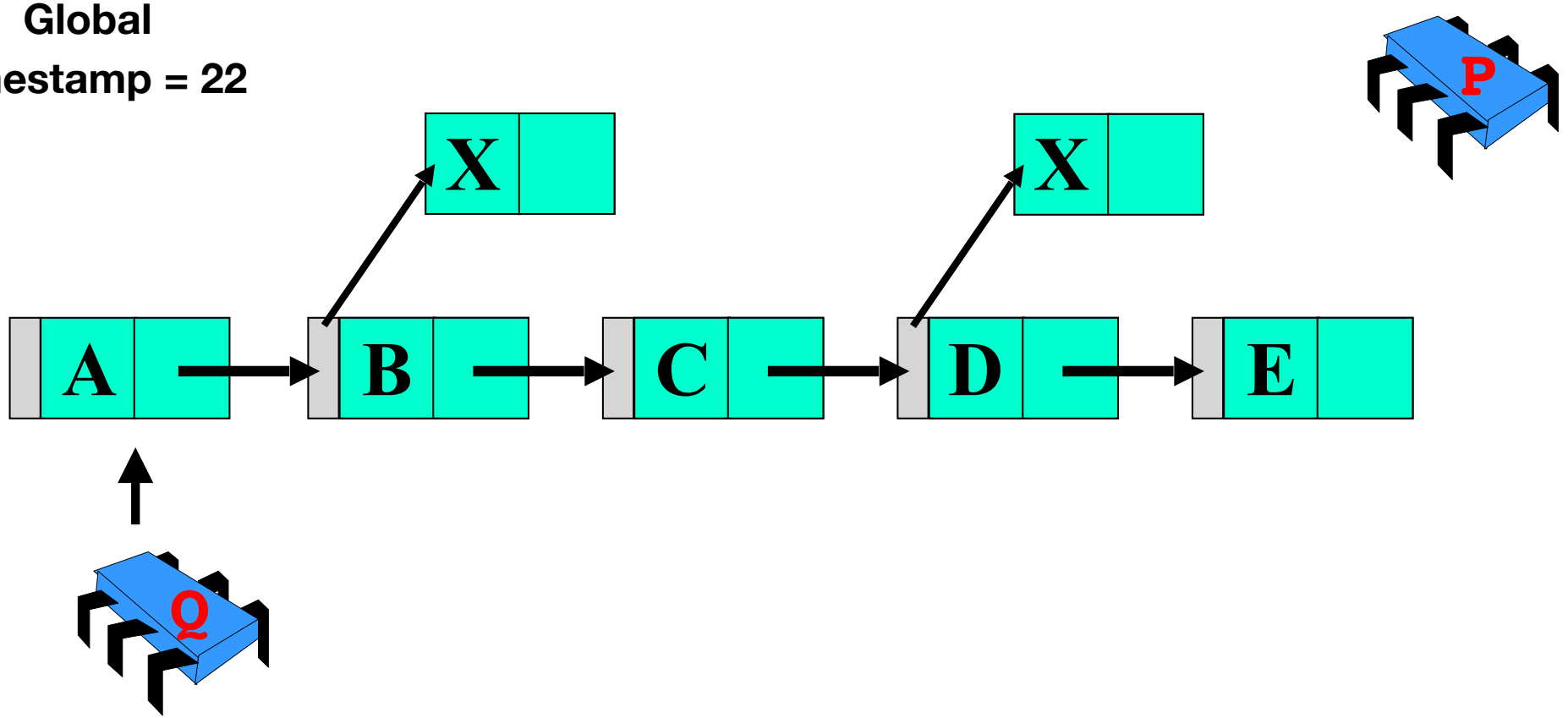
Lookup B & D

Read Log Update

Update B & D simultaneous

Writer
Timestamp = ∞

Global
Timestamp = 22



Lookup B & D

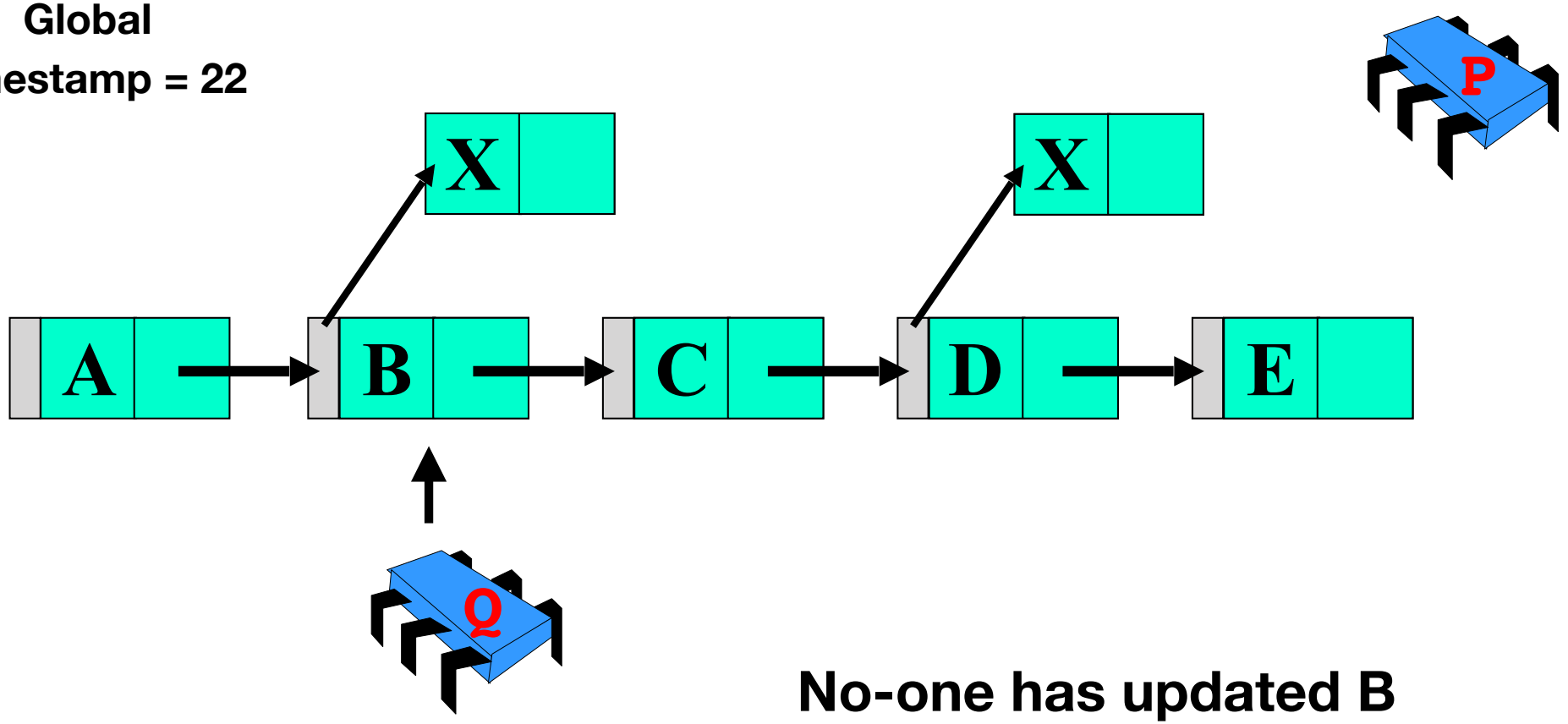
Local Timestamp = 22

Read Log Update

Update B & D simultaneous

Writer
Timestamp = ∞

Global
Timestamp = 22



No-one has updated B

Lookup B & D

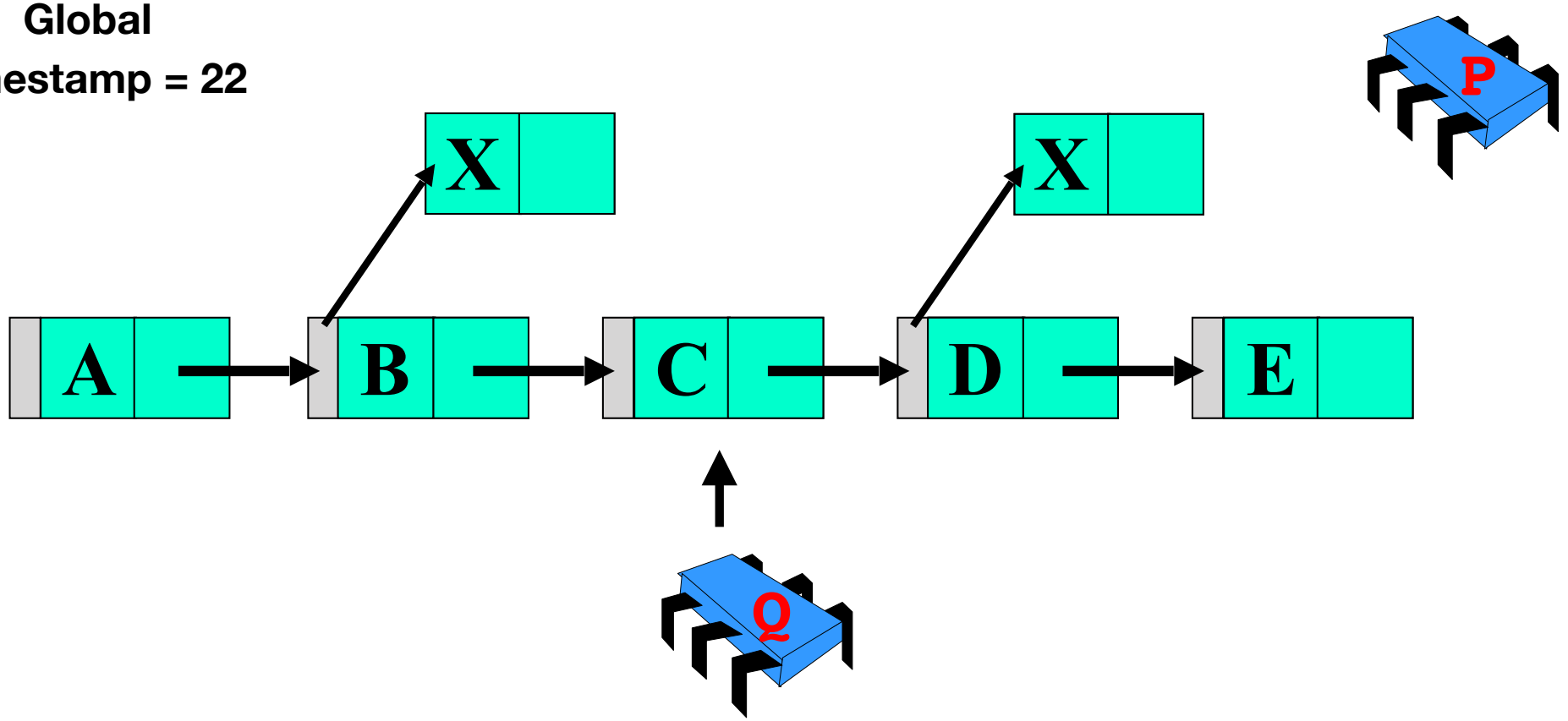
Local Timestamp = 22

Read Log Update

Update B & D simultaneous

Writer
Timestamp = ∞

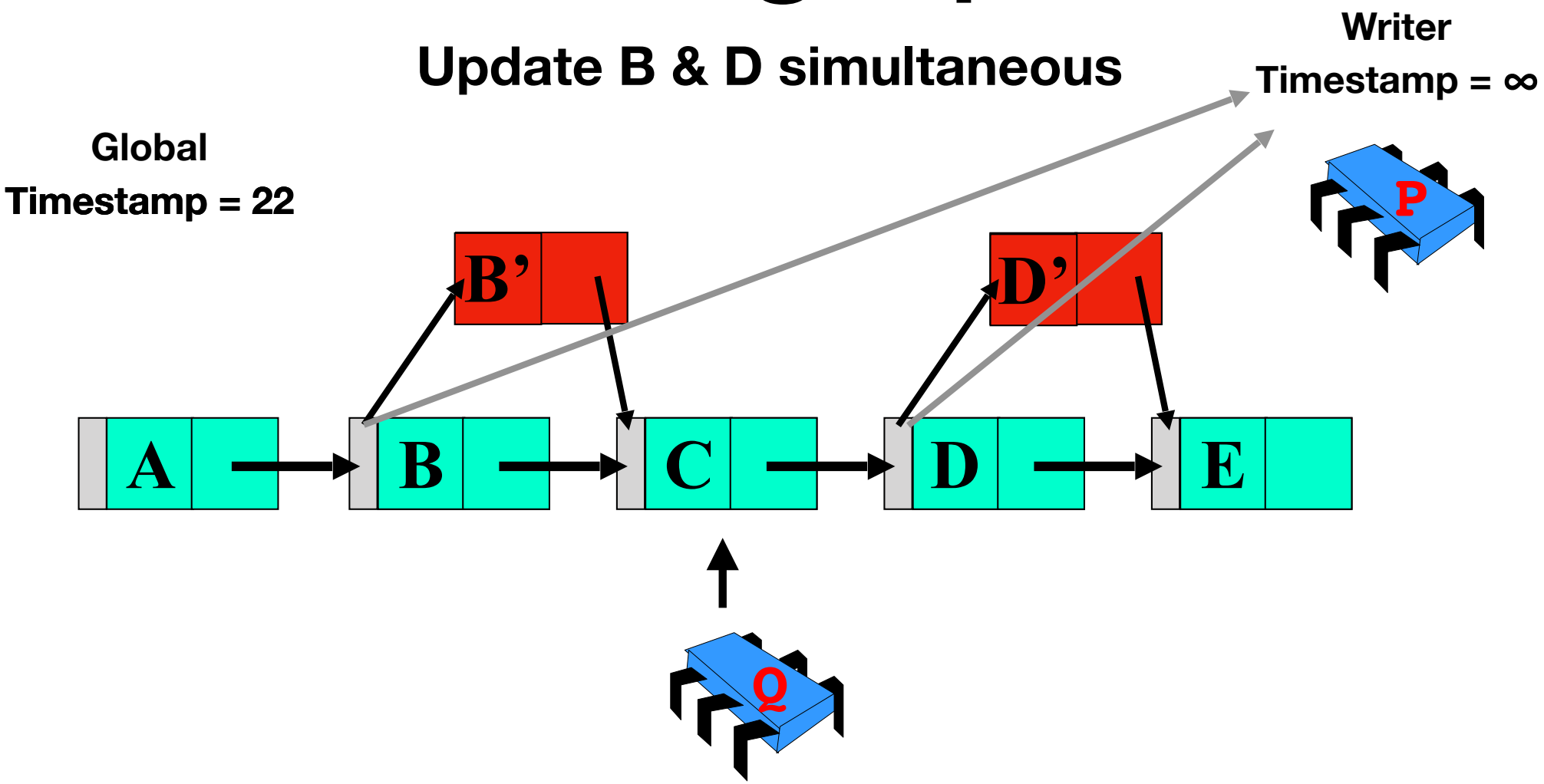
Global
Timestamp = 22



Lookup B & D

Local Timestamp = 22

Read Log Update



Read Log Update

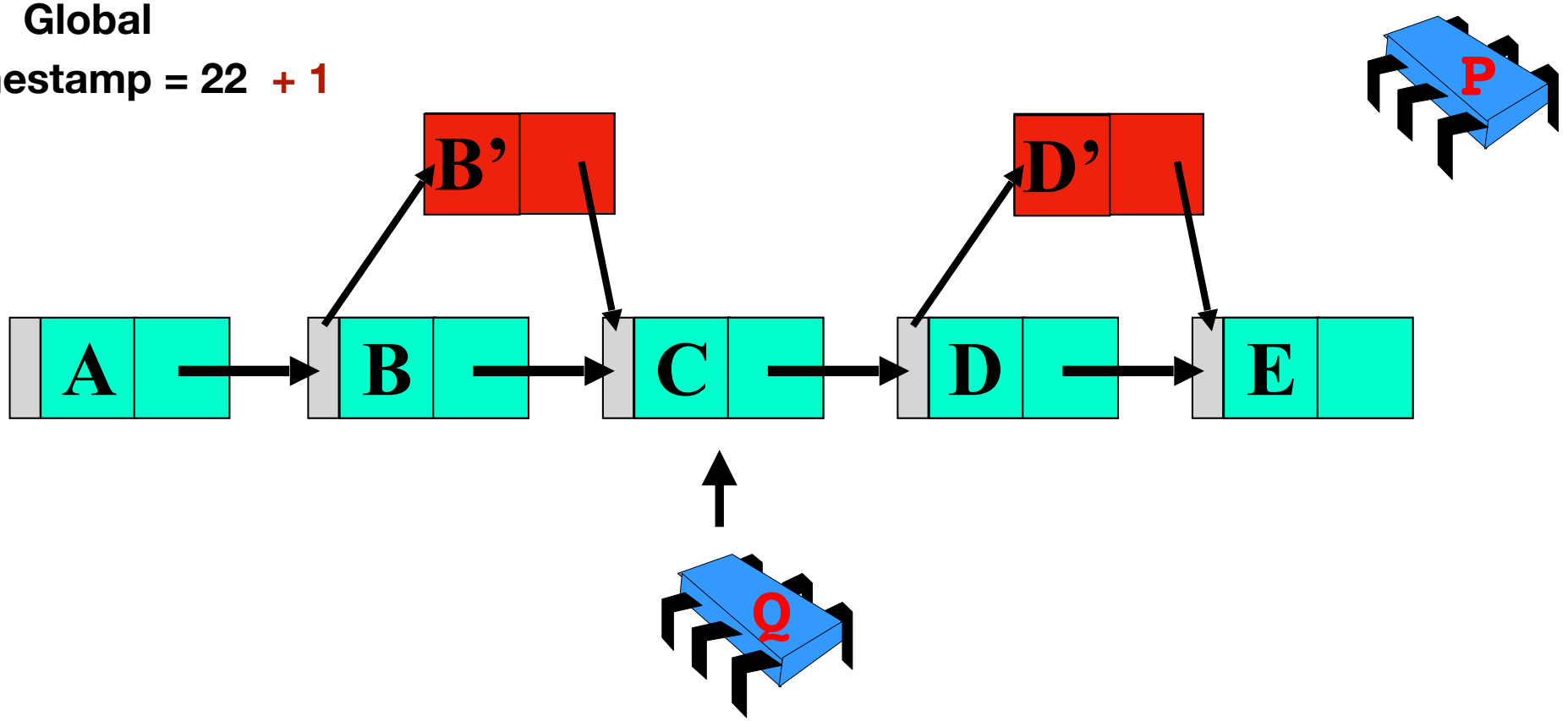
Update B & D simultaneous

Writer

Timestamp = 22 + 1

Global

Timestamp = 22 + 1



Lookup B & D

Local Timestamp = 22

Read Log Update

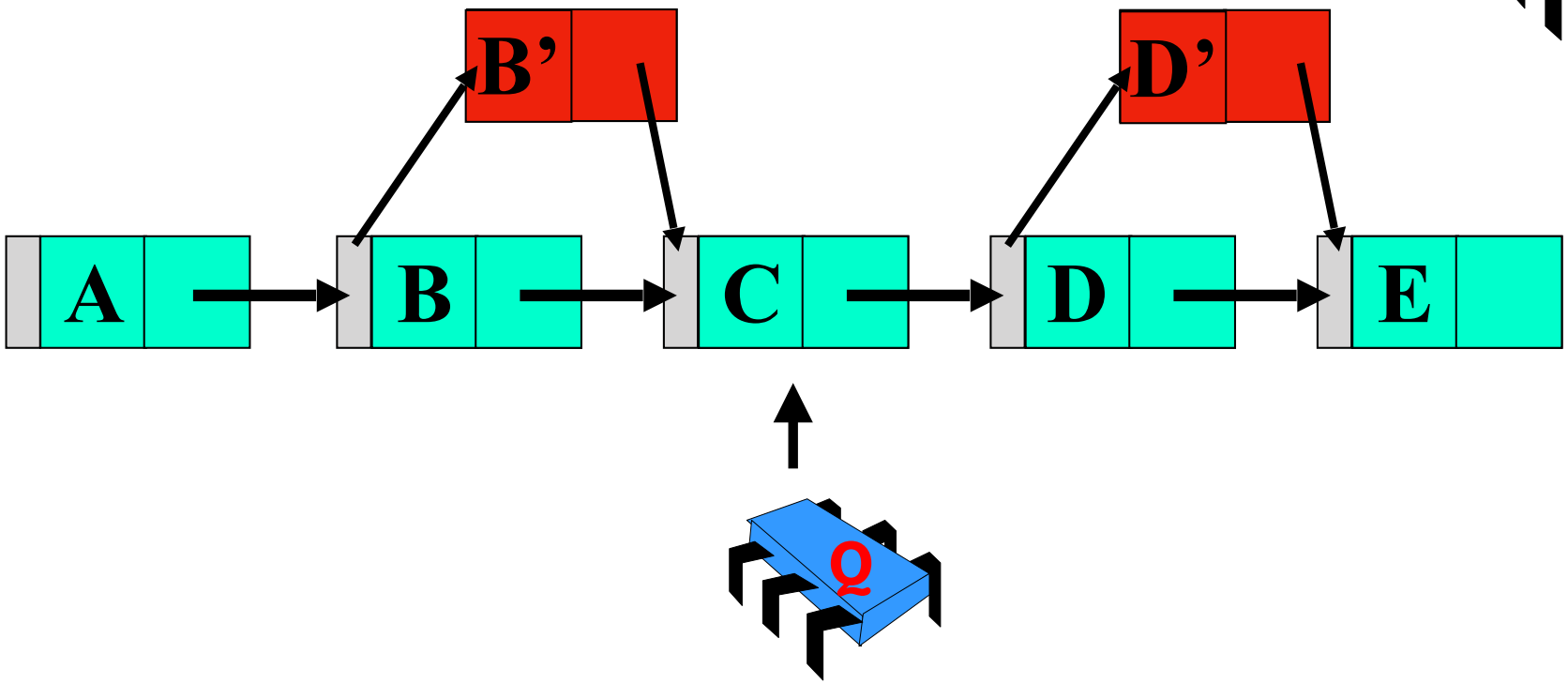
Update B & D simultaneous

Writer

Timestamp = 23

Global

Timestamp = 23



Lookup B & D

Local Timestamp = 22

Read Log Update

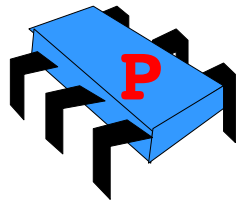
Update B & D simultaneous

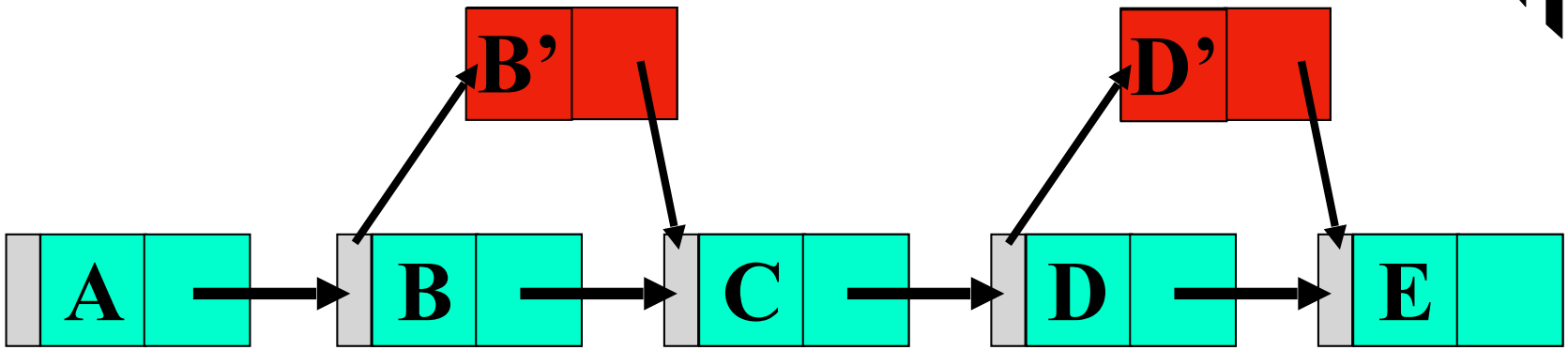
Writer

Timestamp = 23

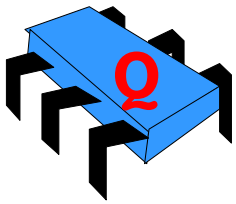
Global

Timestamp = 23





Anyone update D ?



Equal or smaller timestamp?

Lookup B & D

Local Timestamp = 22

Read Log Update

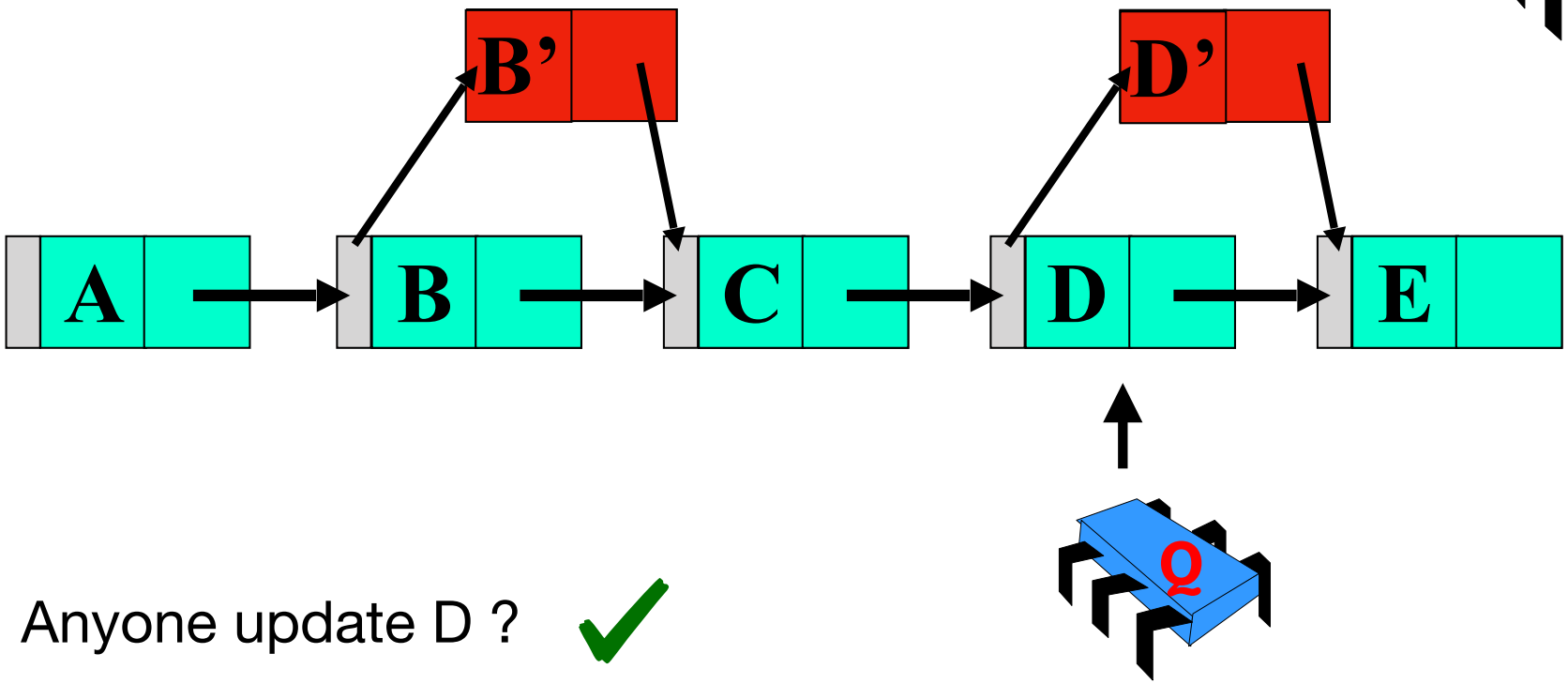
Update B & D simultaneous

Writer

Timestamp = 23

Global

Timestamp = 23



Anyone update D ? ✓

Equal or smaller timestamp? ✗

Lookup B & D

Local Timestamp = 22

Read Log Update

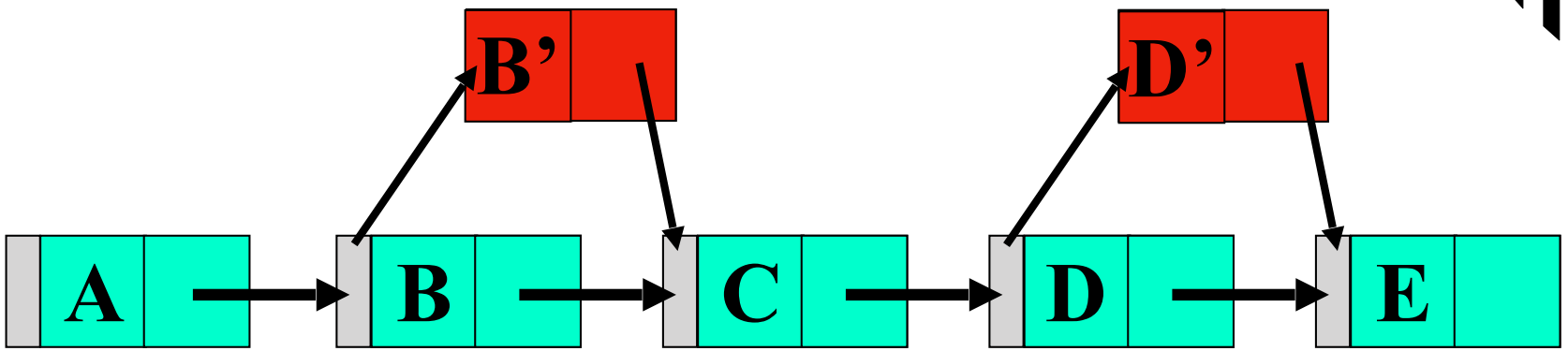
Update B & D simultaneous

Writer

Timestamp = 23

Global

Timestamp = 23



Read Log Update

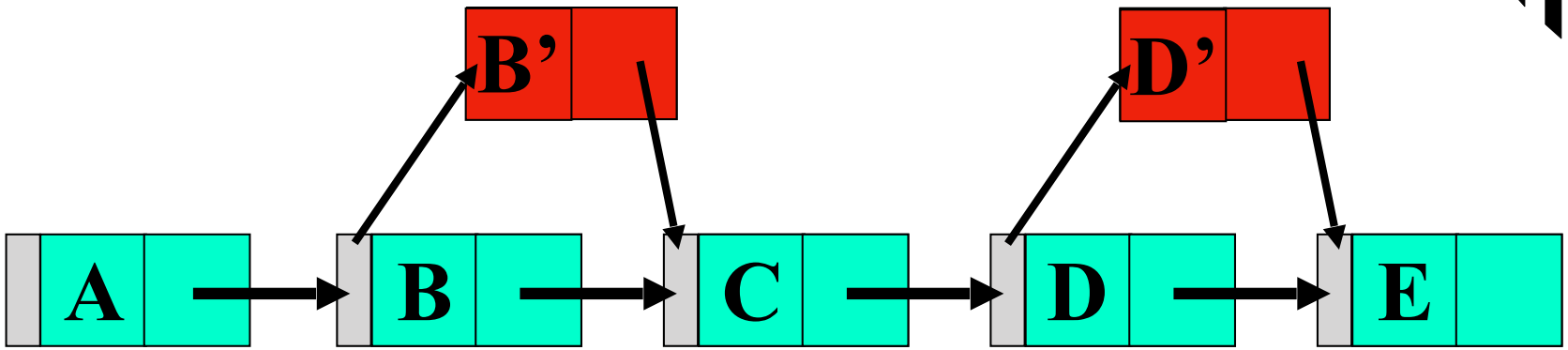
Update B & D simultaneous

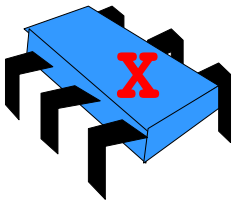
Writer

Timestamp = 23

Global

Timestamp = 23





Lookup B & D

Timestamp = 23

Anyone update B ? ✓

Equal or smaller timestamp? ✓

Read Log Update

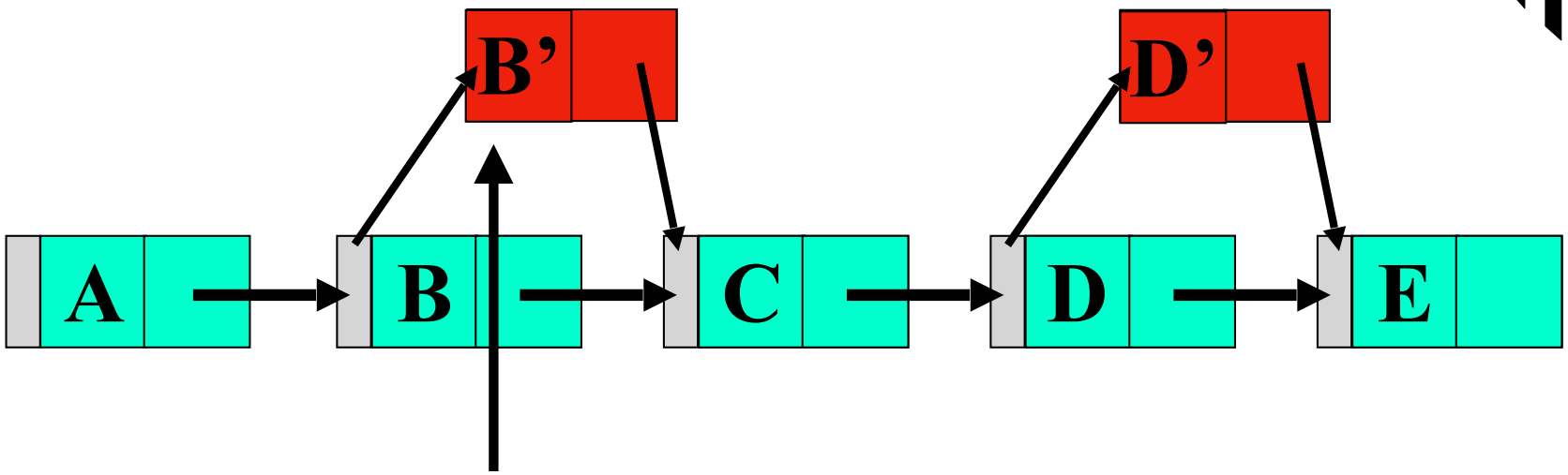
Update B & D simultaneous

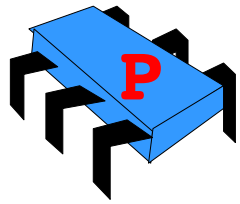
Writer

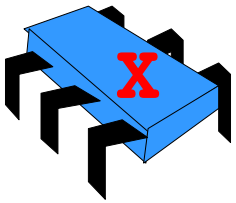
Timestamp = 23

Global

Timestamp = 23







Anyone update B ? ✓

Equal or smaller timestamp? ✓

Lookup B & D

Timestamp = 23

Read Log Update

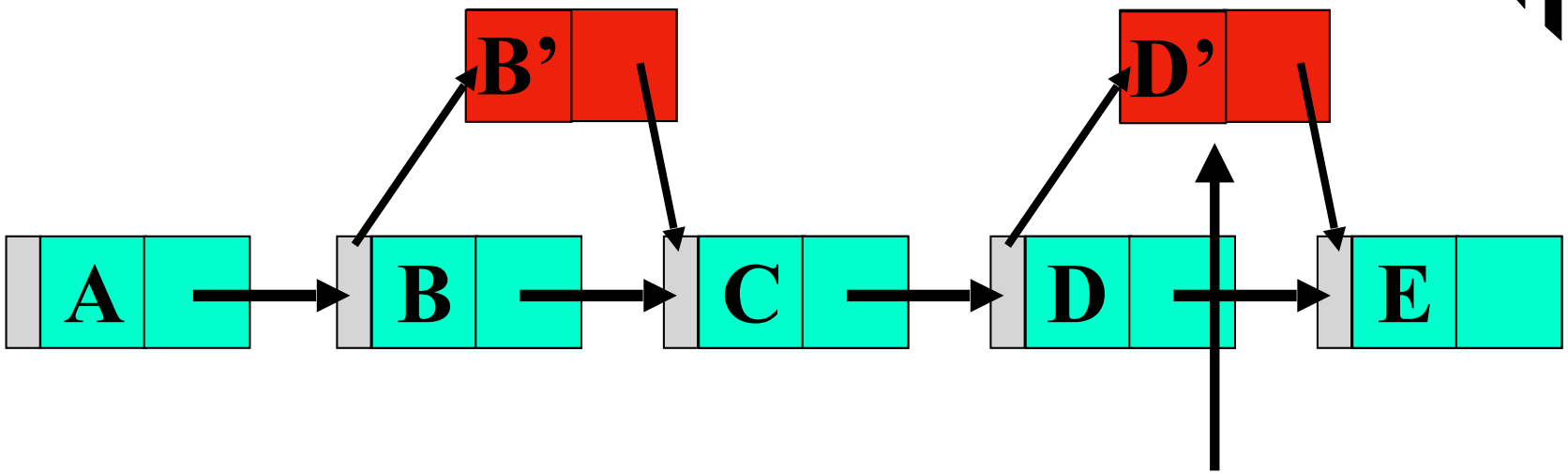
Update B & D simultaneous

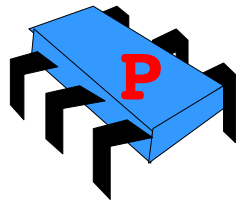
Writer

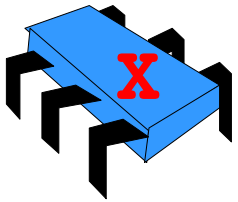
Timestamp = 23

Global

Timestamp = 23







Anyone update D ? ✓

Equal or smaller timestamp? ✓

Lookup B & D

Timestamp = 23

Read Log Update

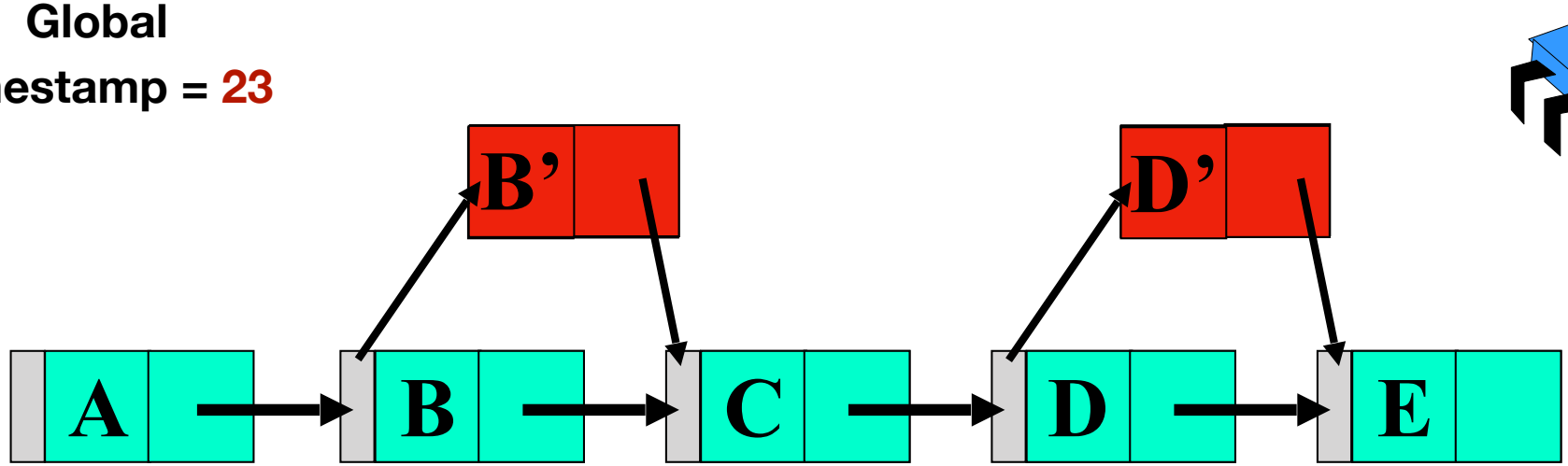
Update B & D simultaneous

Writer

Timestamp = 23

Global

Timestamp = 23



Wait until all previous readers finish their jobs

```
Readers' timestamp < Current writers’
```

Read Log Update

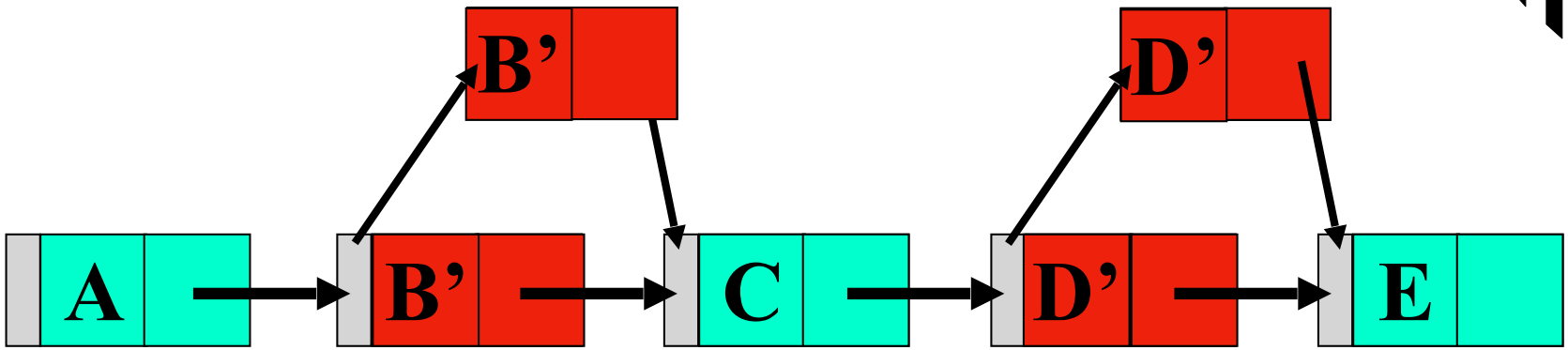
Update B & D simultaneous

Writer

Timestamp = 23

Global

Timestamp = 23



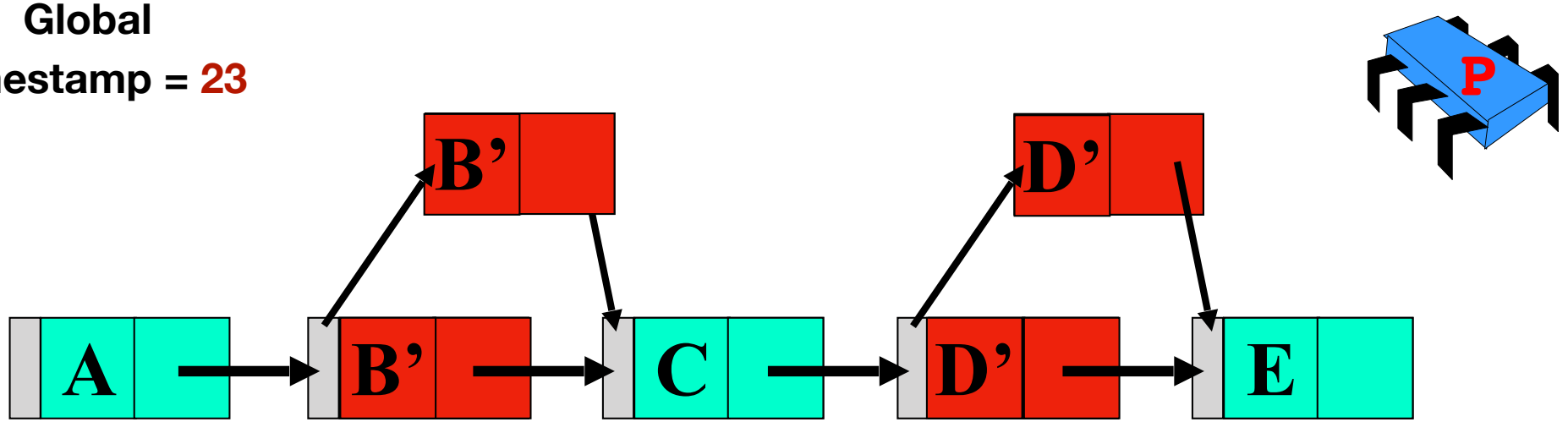
Write back logs when previous readers finish their jobs

Read Log Update

Update B & D simultaneous

Timestamp = ∞

Global Timestamp = 23

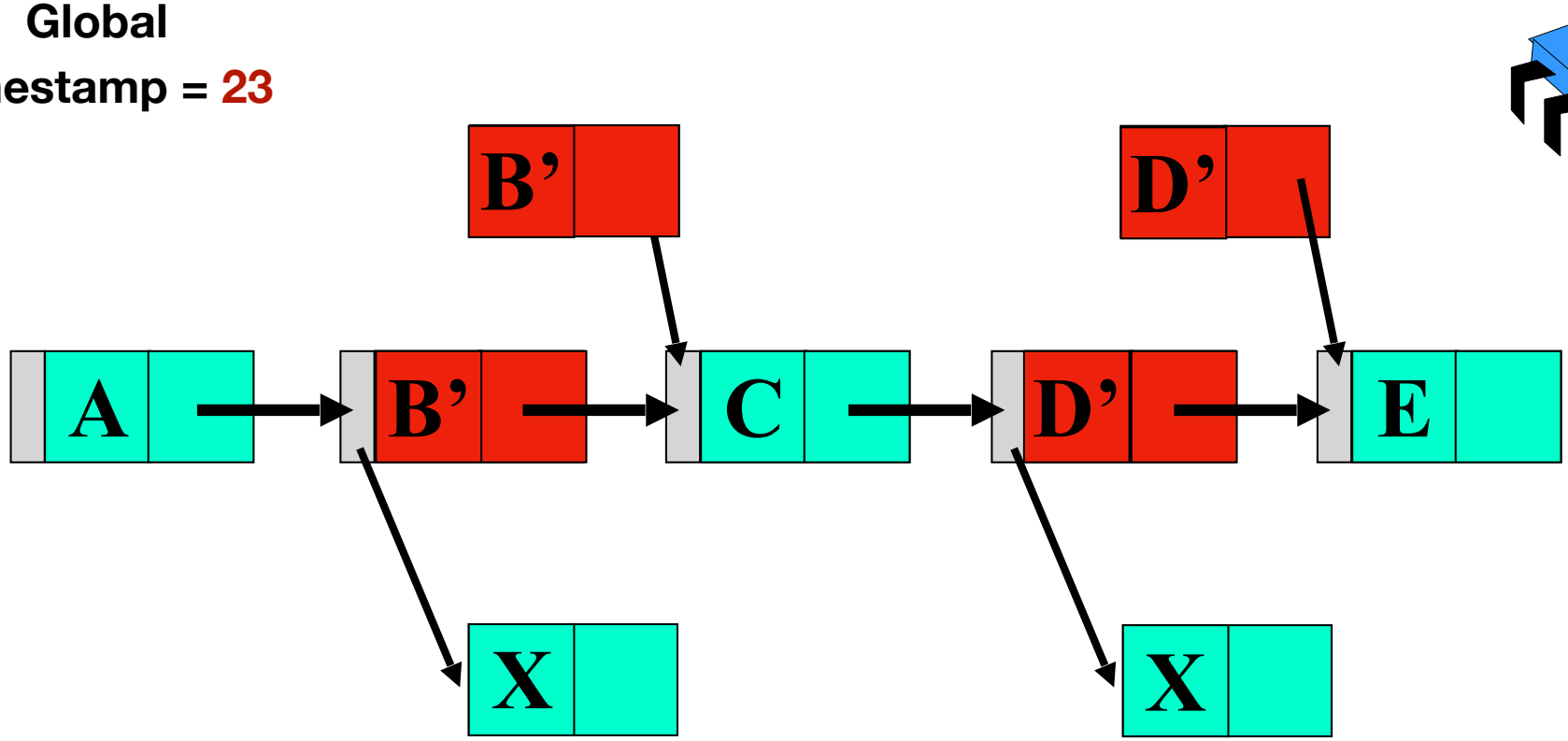


Reset write timestamp

Read Log Update

Update B & D simultaneous

Global
Timestamp = 23



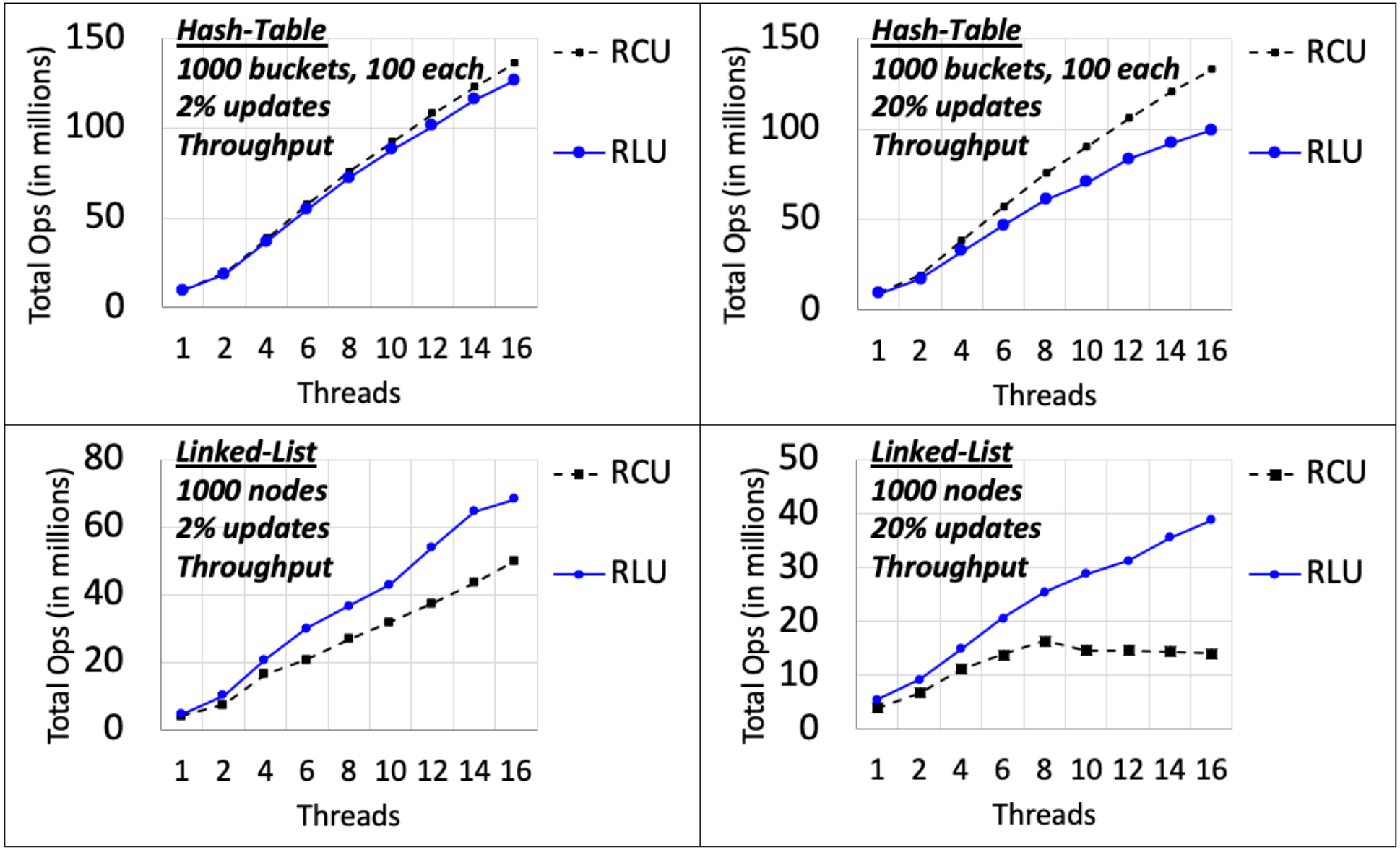
Swap to another log buffer

Most exciting part in RLU

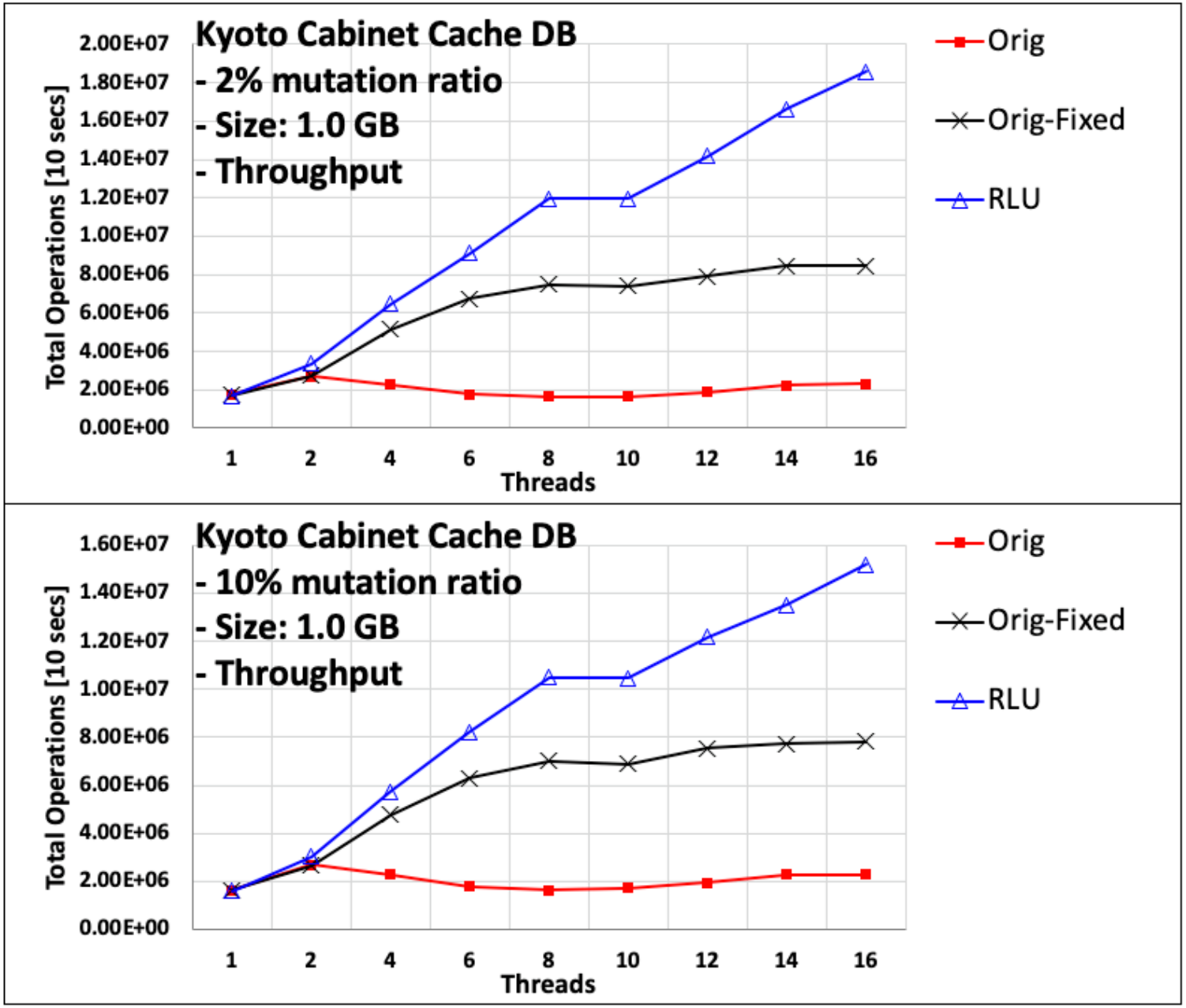
- 2 key concerns for concurrent data structure:
  - **Unsynchronized** traversals ✓
  - **Multi-location** atomic updates ✓
- Easier to be used in complicated applications
  - Double link list
  - Kyoto Cabinet

Evaluation

Microbenchmarks



Kyoto Cabinet



Conclusion

- Single pointer update => Global clock update
- Stronger semantic compared to RCU
- Significantly lower intellectual cost to the programmer
- No extra overhead on reader’s critical path
- Achieve similar (even better) performance with RCU

Thanks
Q & A