Incorrect Systems: It's not the Problem, It's the Solution

Christoph Kirsch
Universität Salzburg

Austrian Computer Science Day, Vienna, June 2012

Software

Software/
Hardware

Hardware

Software

Software/
Hardware

Hardware

Krishna Palem
Rice

Software

Software/
Hardware

Probabilistic or
Approximate
Computing

Krishna Palem
Rice

Software

Software/
Hardware

Rakesh Kumar
UIUC

Probabilistic or
Approximate
Computing

Krishna Palem
Rice

Software

Stochastic Processors

Rakesh Kumar
UIUC

Probabilistic or Approximate Computing

Krishna Palem
Rice

Software

Martin Rinard
MIT

Stochastic Processors

Rakesh Kumar
UIUC

Probabilistic or Approximate Computing

Krishna Palem
Rice

Program Transformation

Martin Rinard
MIT

Stochastic Processors

Rakesh Kumar
UIUC

Probabilistic or Approximate Computing

Krishna Palem
Rice

Program Transformation

1. memory leaks
2. addressing errors
3. infinite loops

Stochastic Processors

Rakesh Kumar
UIUC

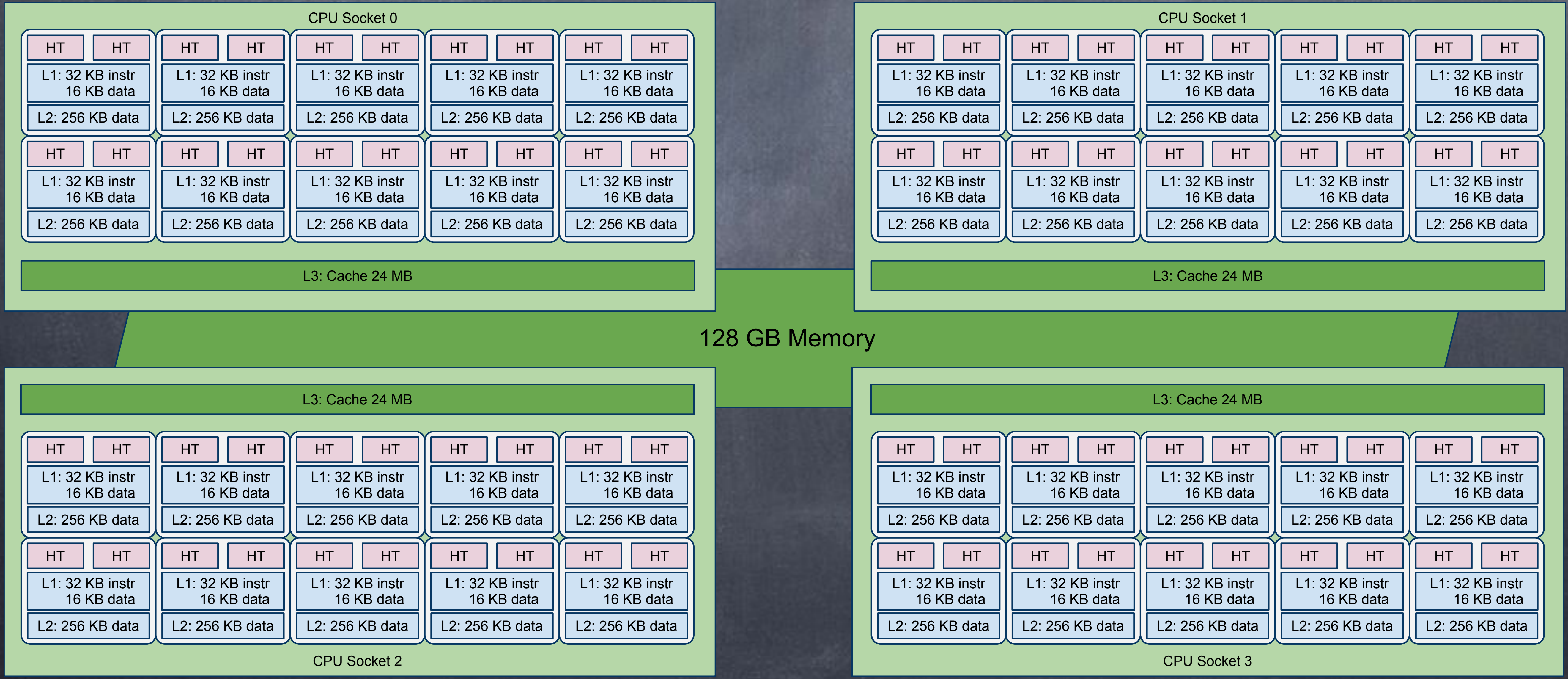
Probabilistic or Approximate Computing

Krishna Palem
Rice

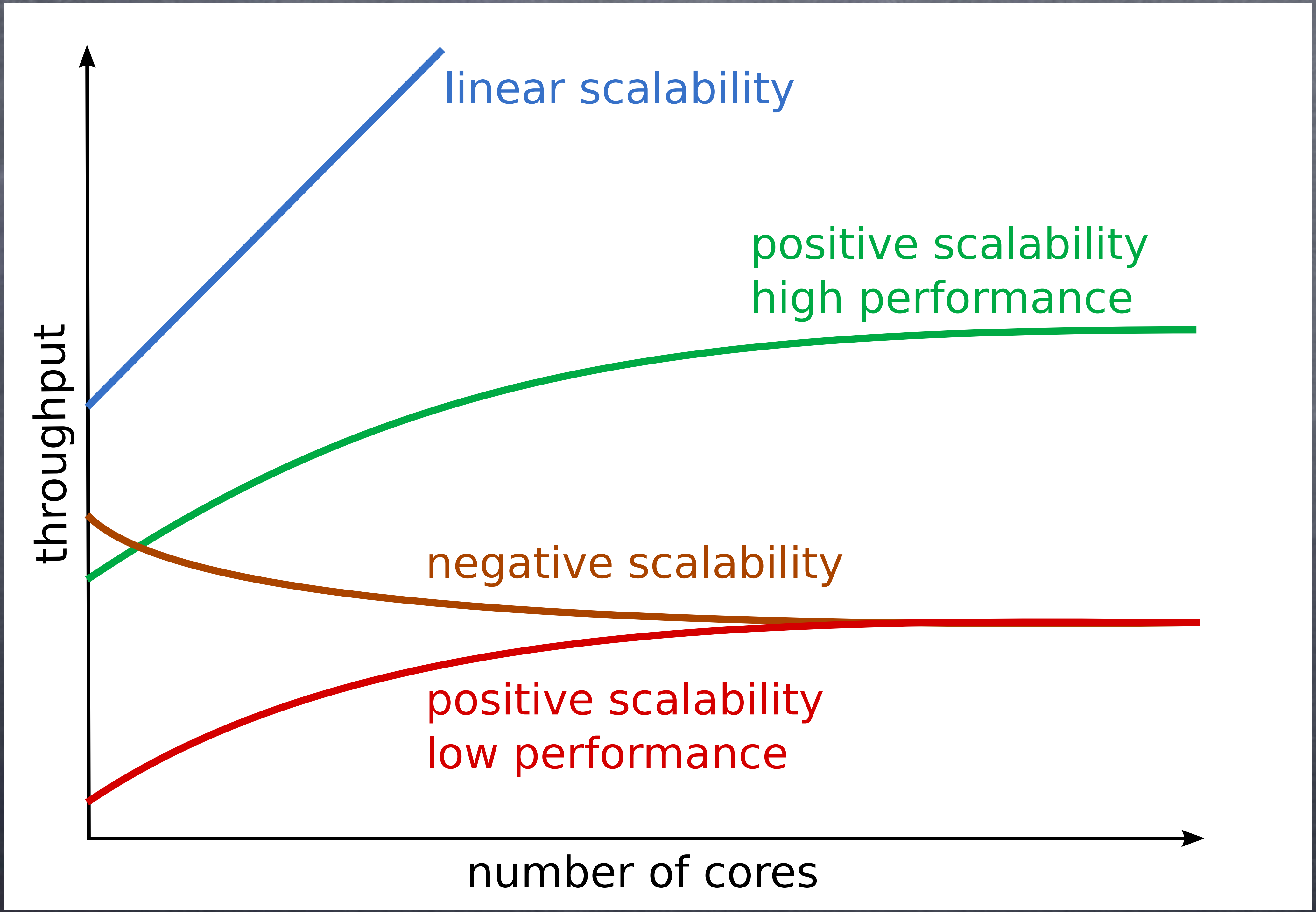
Metrics of Correctness in Systems Engineering

Joint work w/ A. Haas,
M. Lippautz, H. Payer,
H. Röck, A. Sokolova and
our collaborators at ISTA
T. Henzinger, A. Sezgin

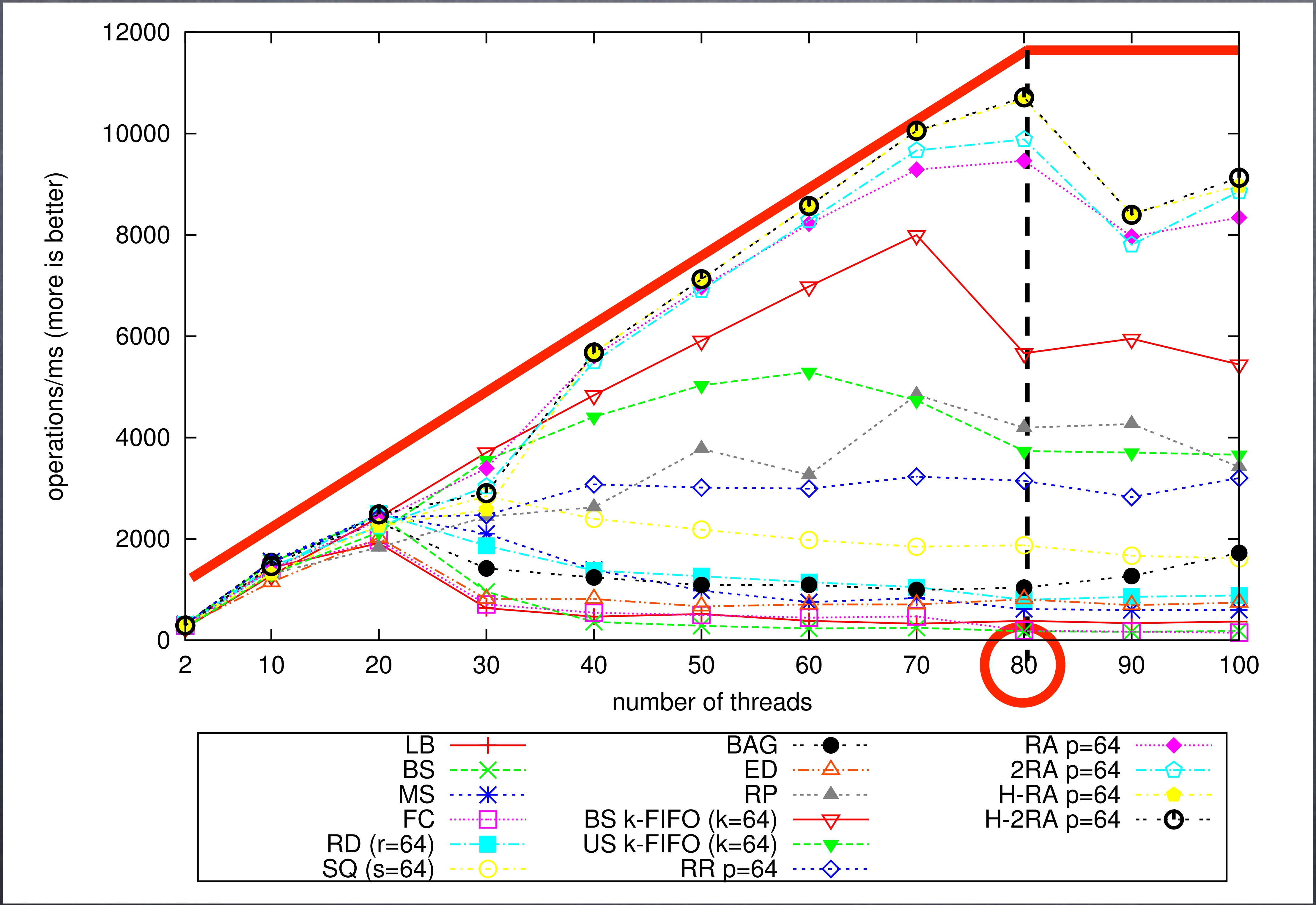
4 processors x 10 cores x 2 hardware threads = 80 hardware threads



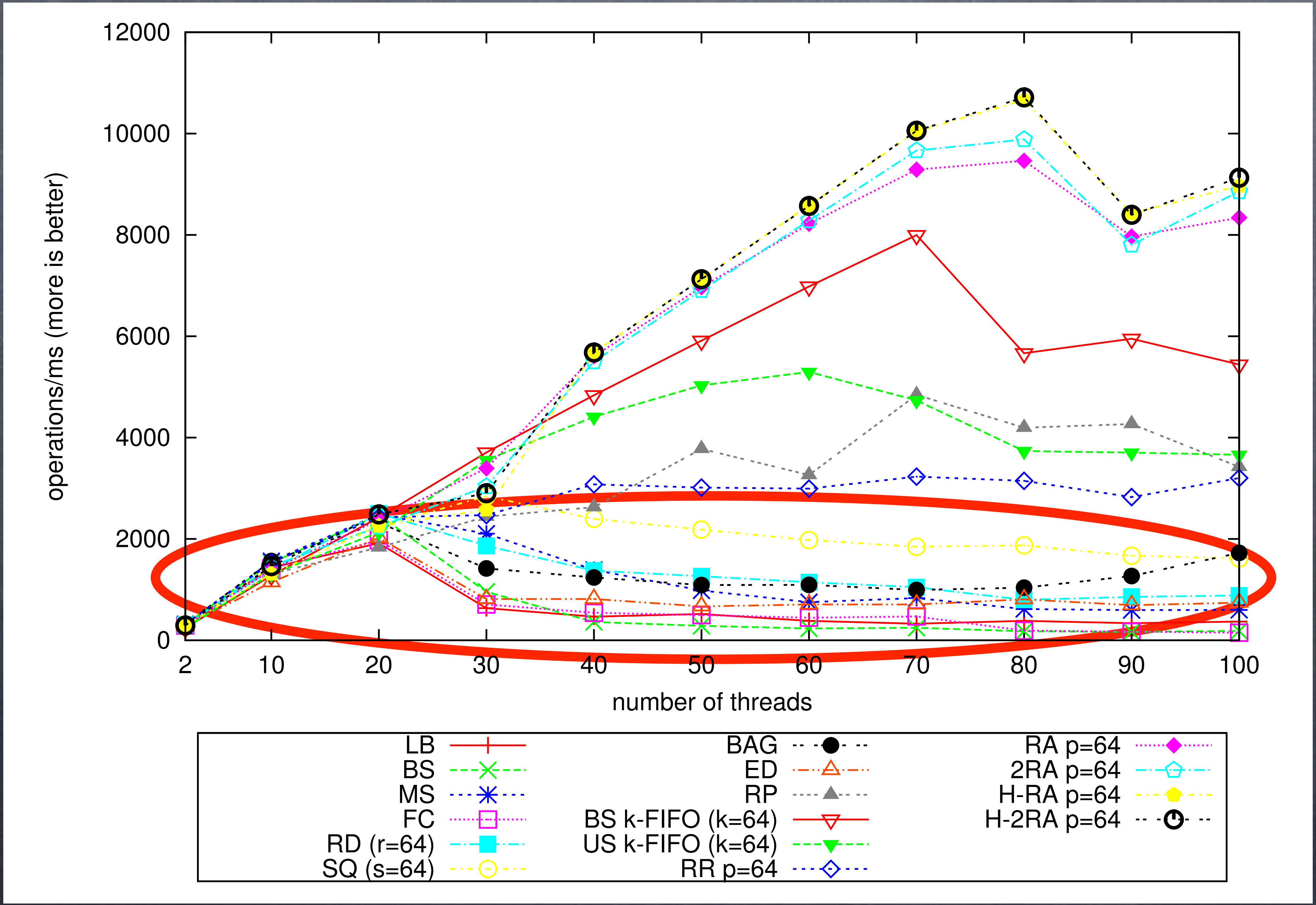
Performance & Scalability



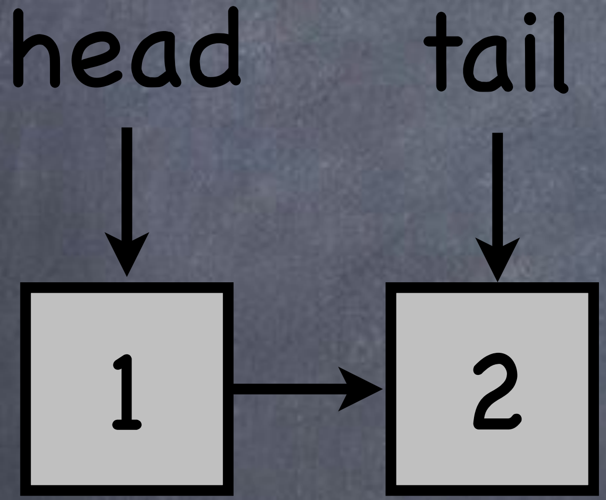
Ideal 80-Thread Performance



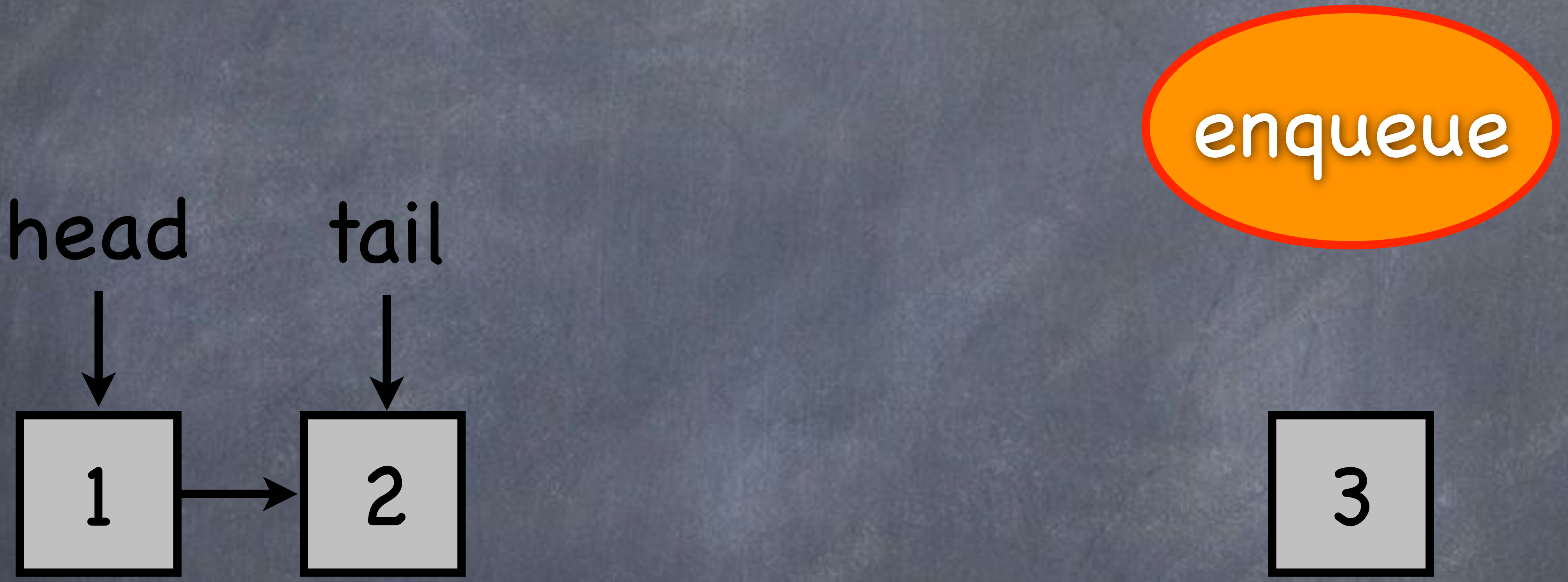
Regular FIFO Queues



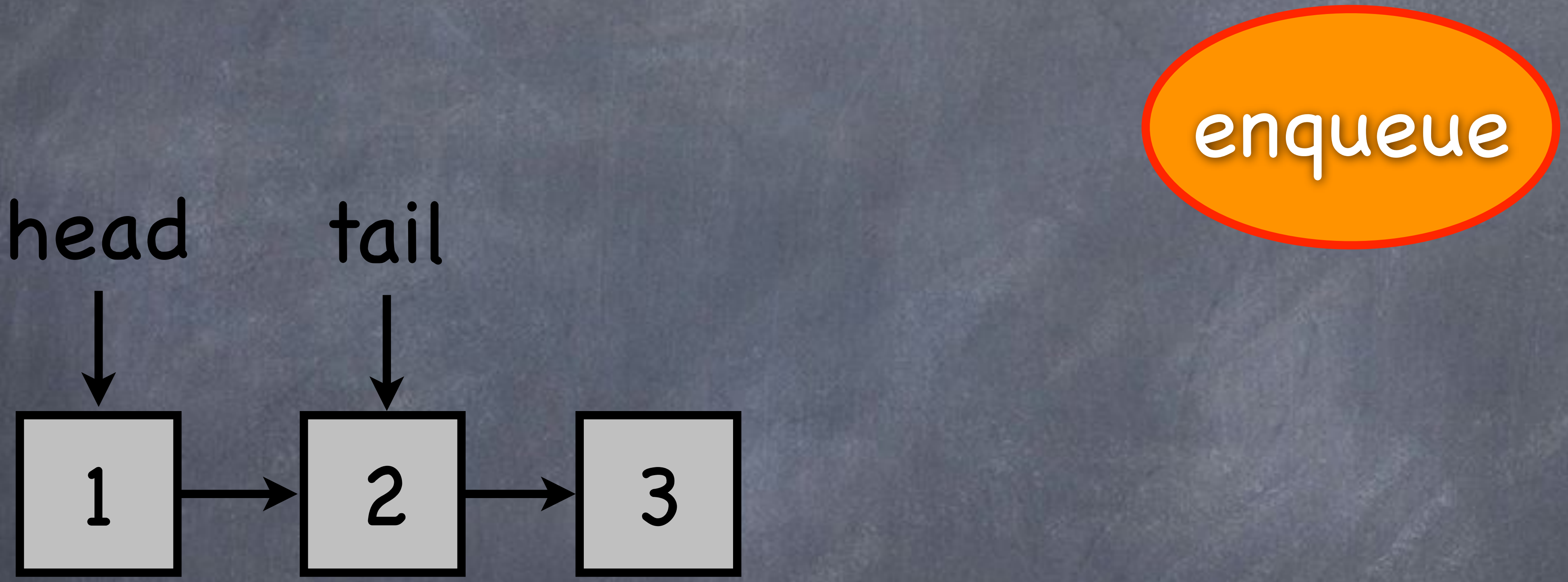
Concurrent First-in-First-out (FIFO) Queue



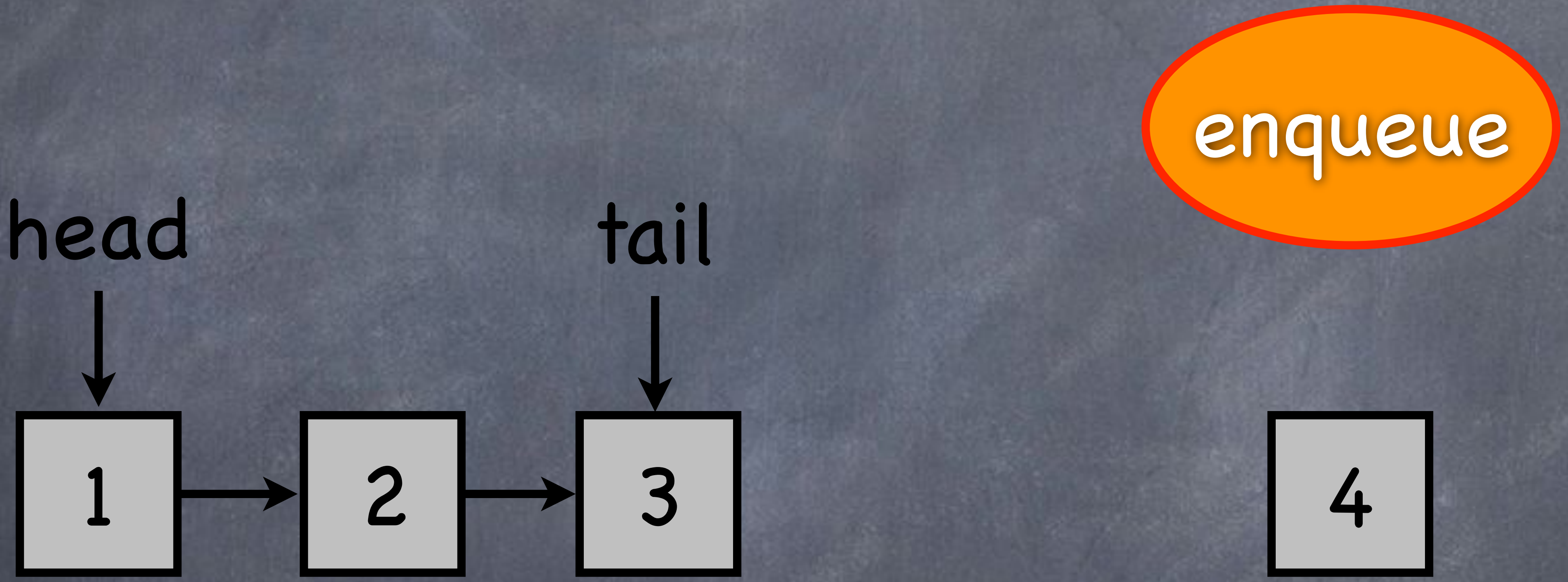
Concurrent First-in-First-out (FIFO) Queue



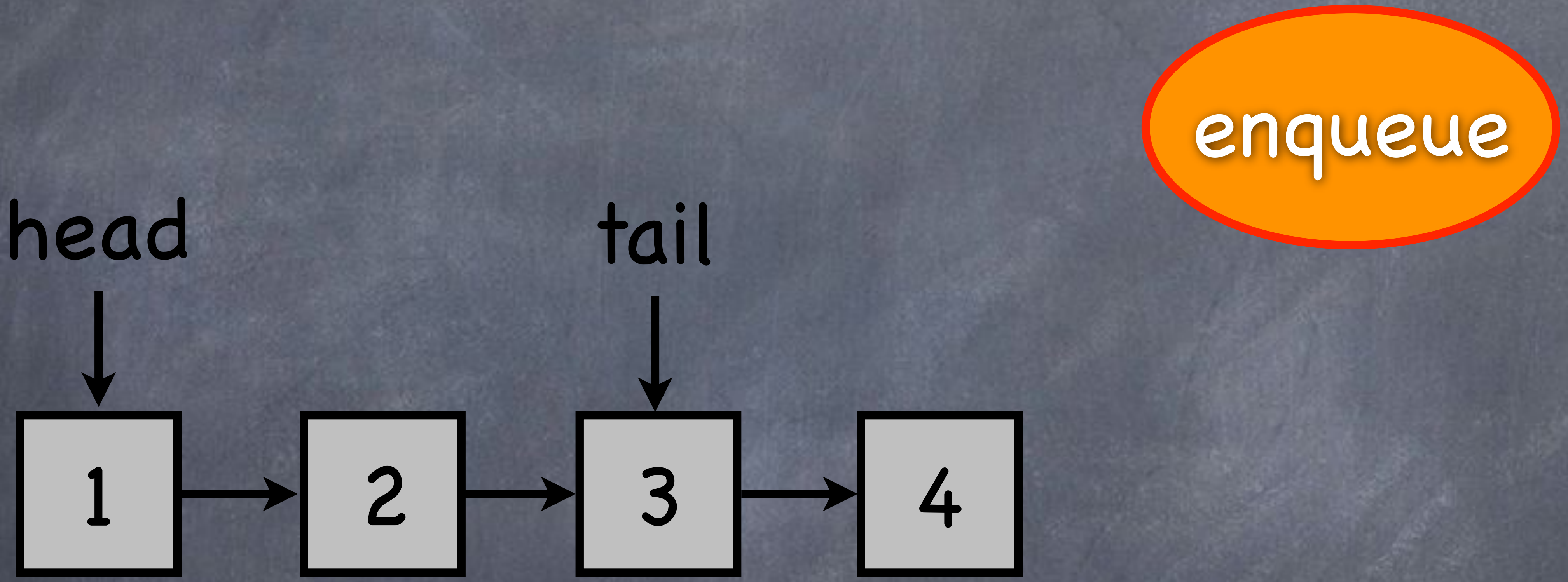
Concurrent First-in-First-out (FIFO) Queue



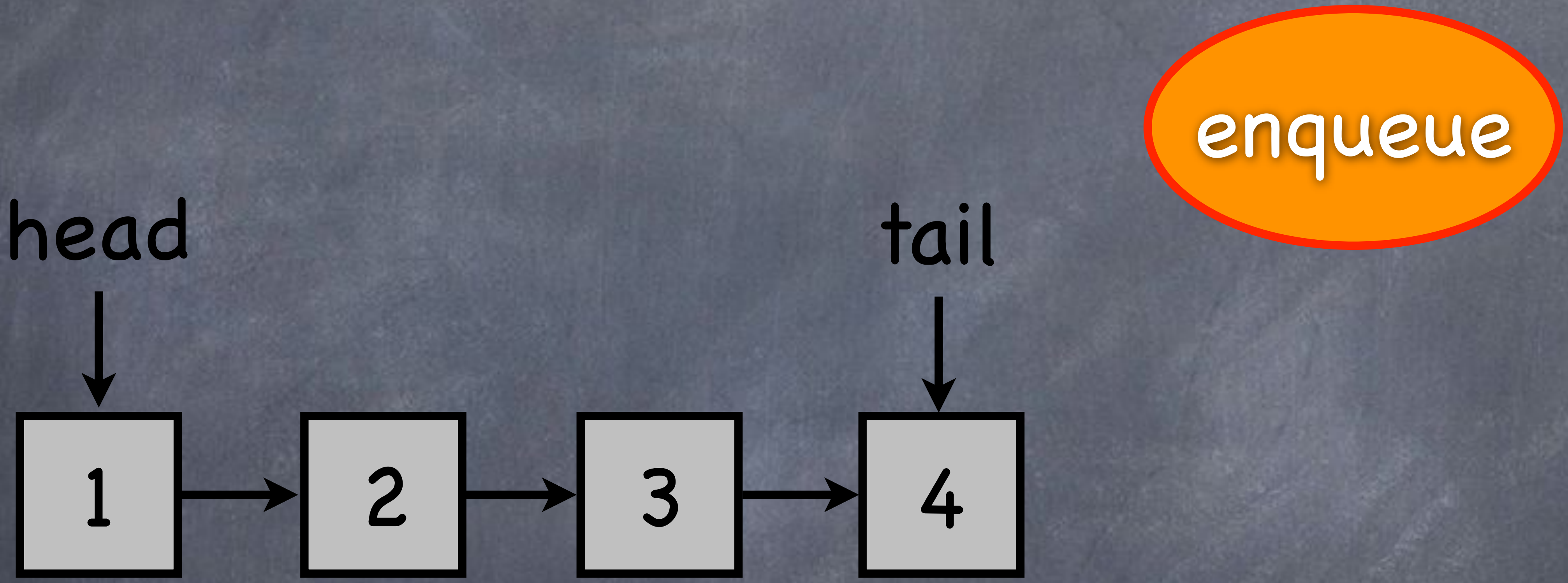
Concurrent First-in-First-out (FIFO) Queue



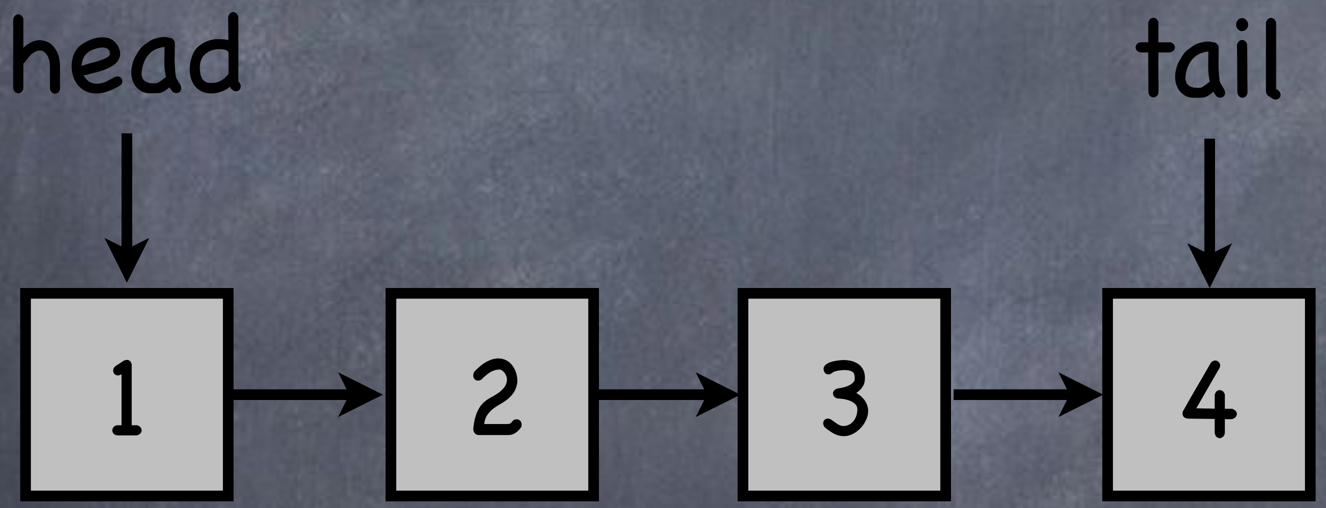
Concurrent First-in-First-out (FIFO) Queue



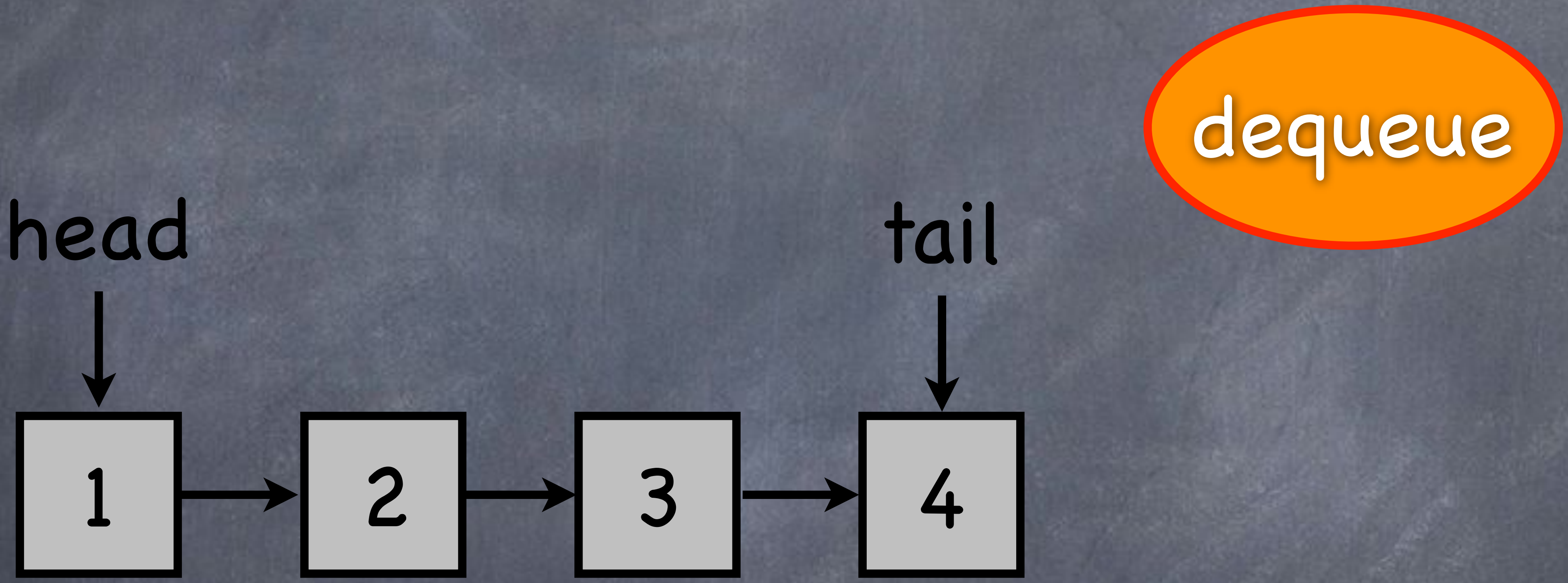
Concurrent First-in-First-out (FIFO) Queue



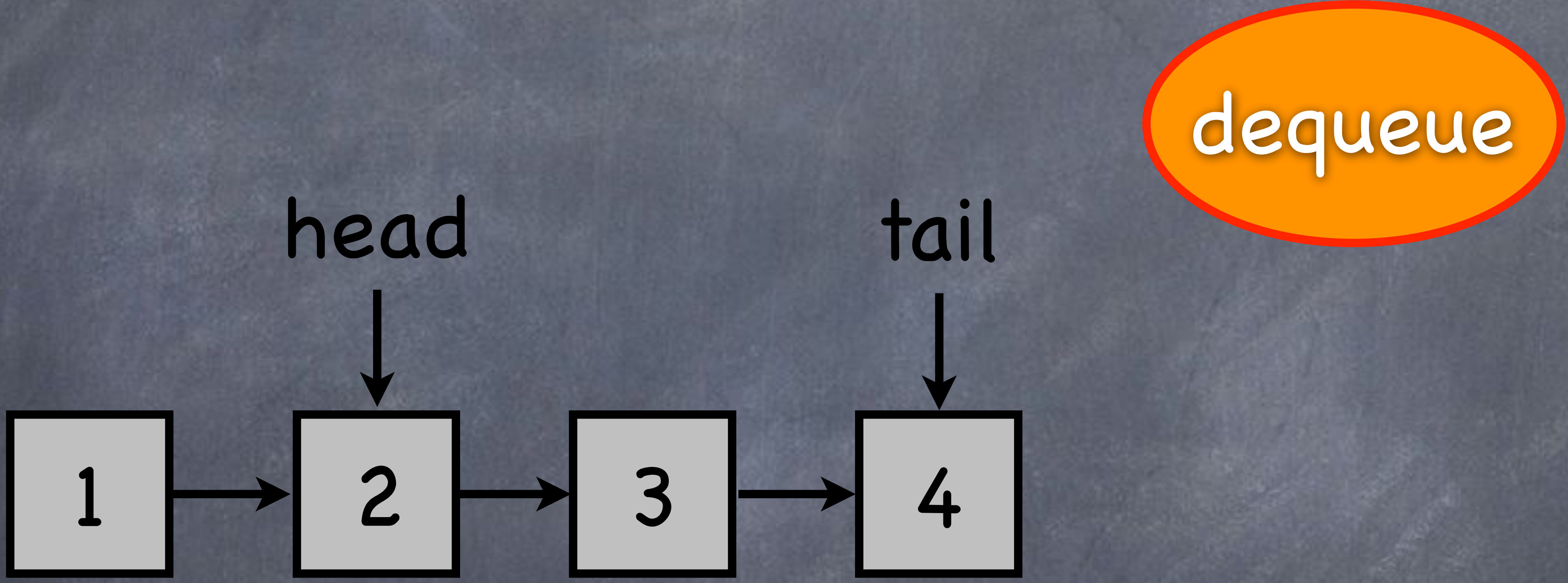
Concurrent First-in-First-out (FIFO) Queue



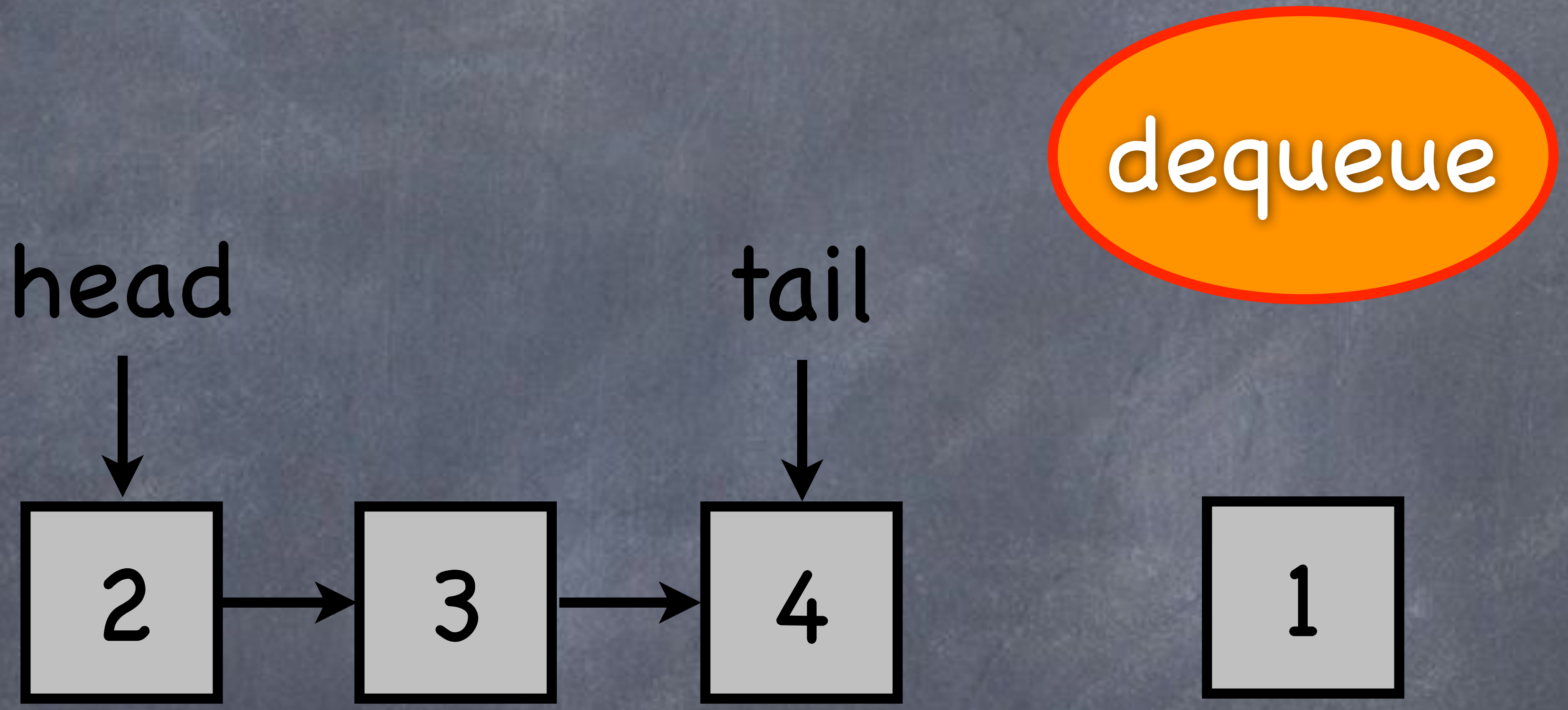
Concurrent First-in-First-out (FIFO) Queue



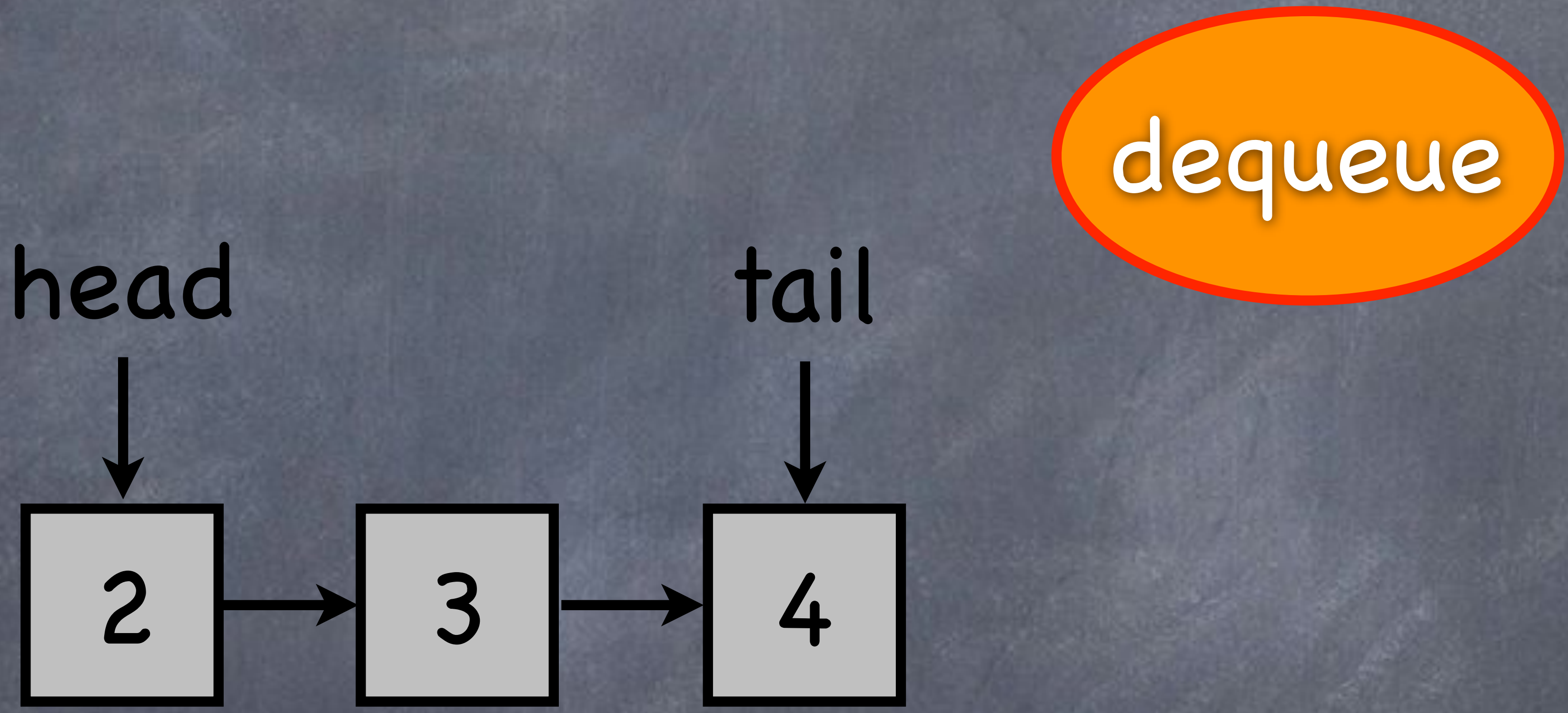
Concurrent First-in-First-out (FIFO) Queue



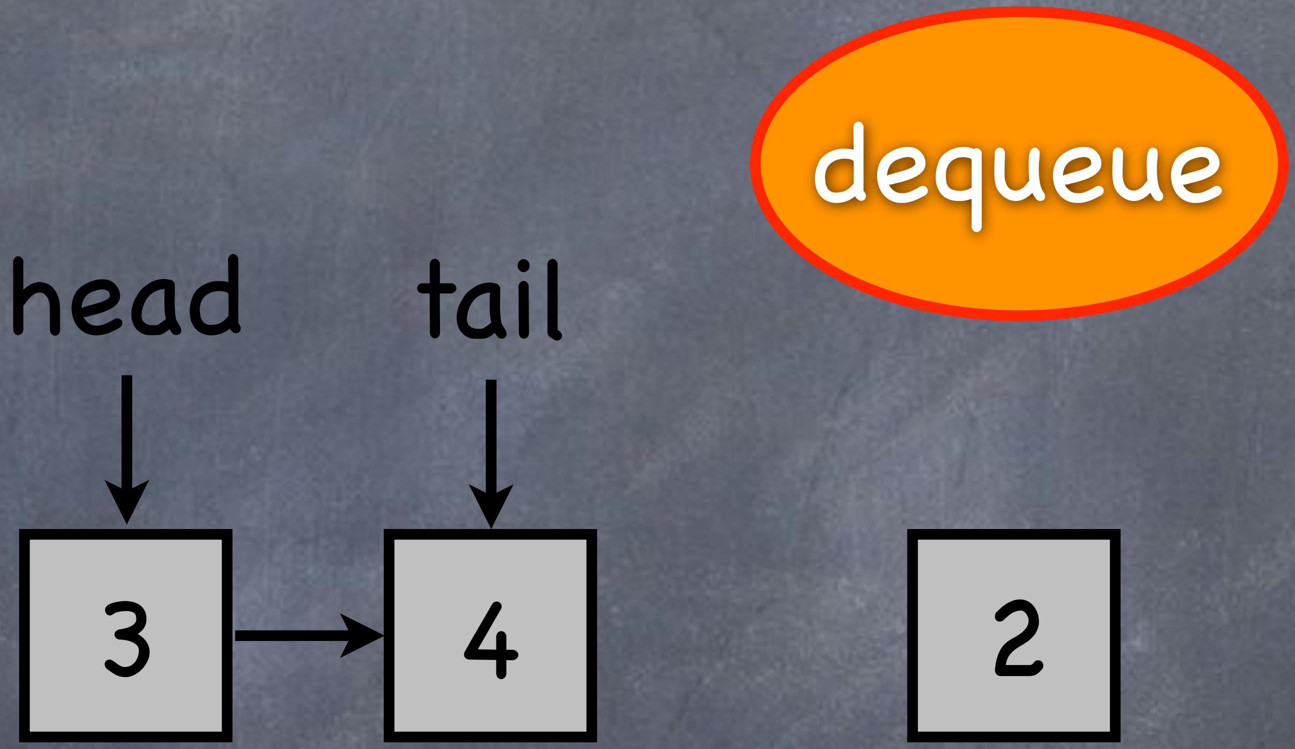
Concurrent First-in-First-out (FIFO) Queue



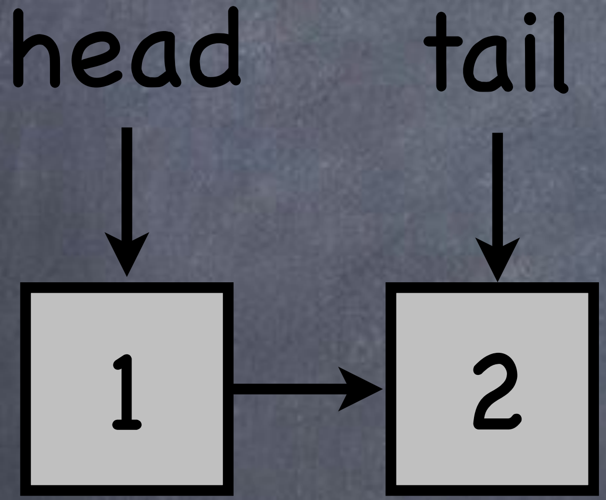
Concurrent First-in-First-out (FIFO) Queue



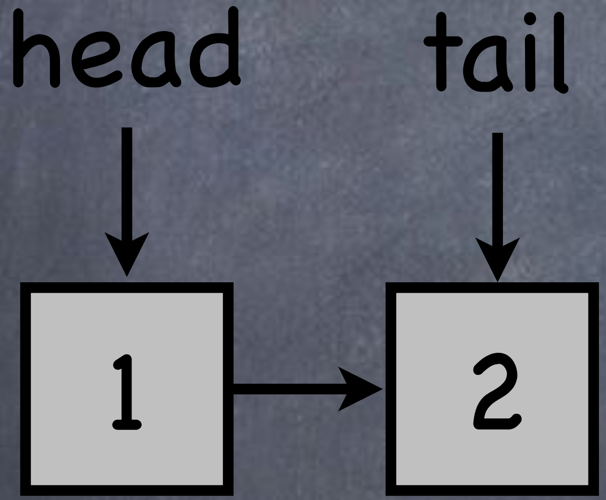
Concurrent First-in-First-out (FIFO) Queue



Concurrent First-in-First-out (FIFO) Queue

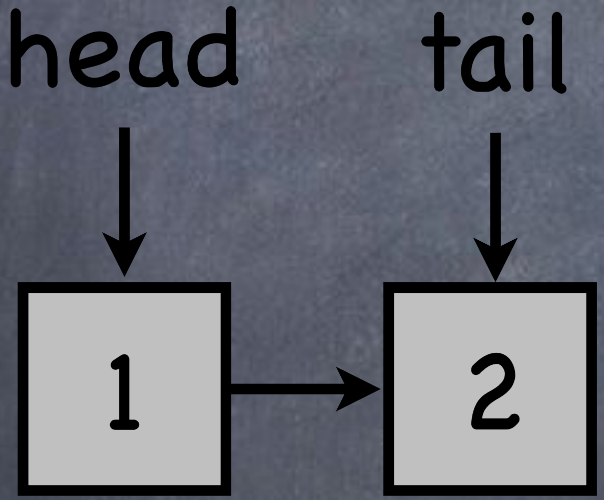


Concurrent First-in-First-out (FIFO) Queue



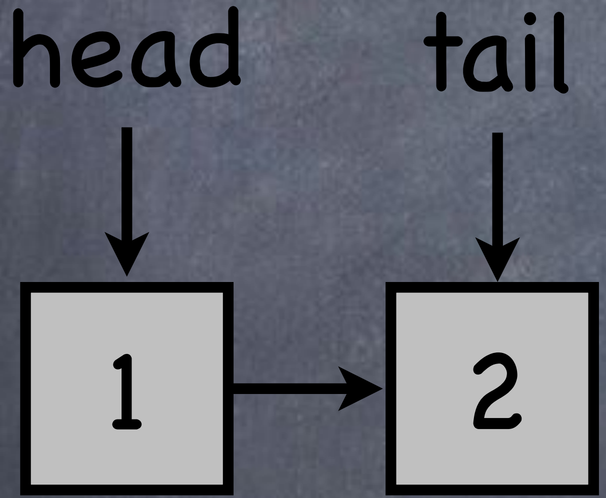
-> 1 lock

Concurrent First-in-First-out (FIFO) Queue



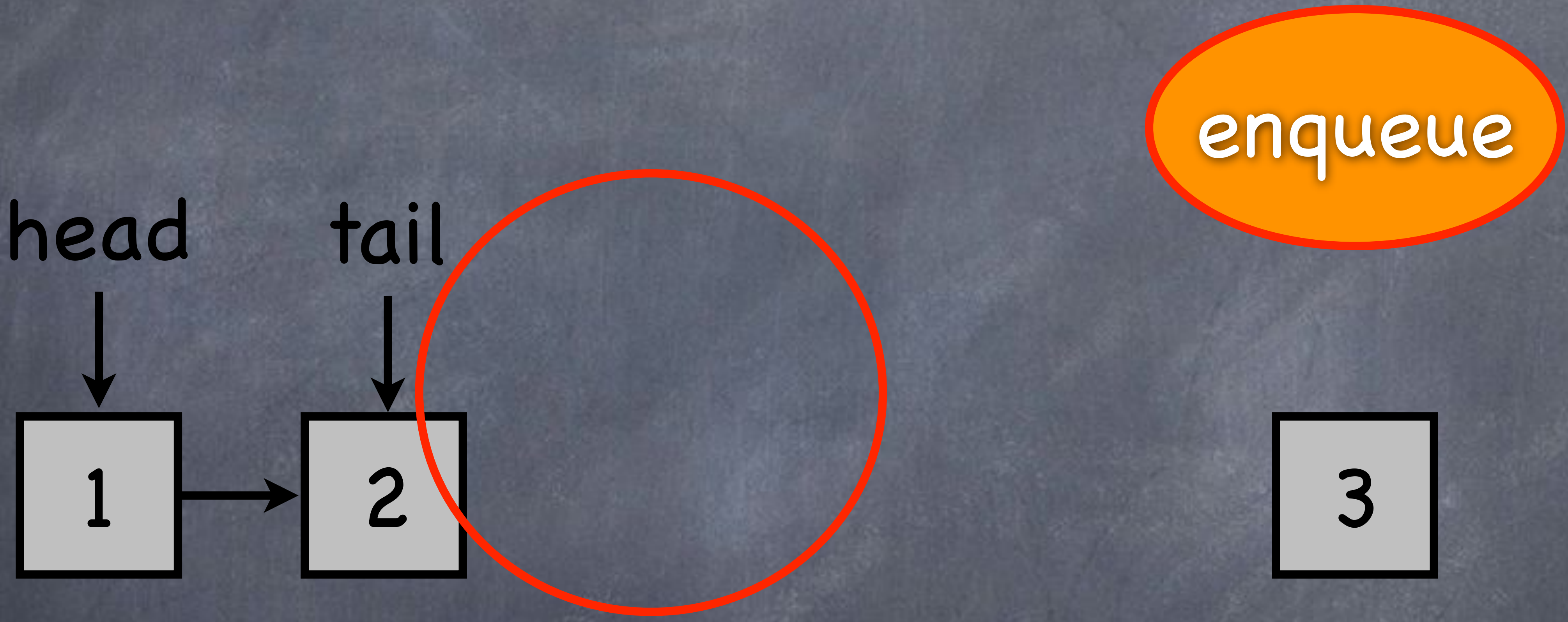
-> 1 lock -> 2 locks

Concurrent First-in-First-out (FIFO) Queue



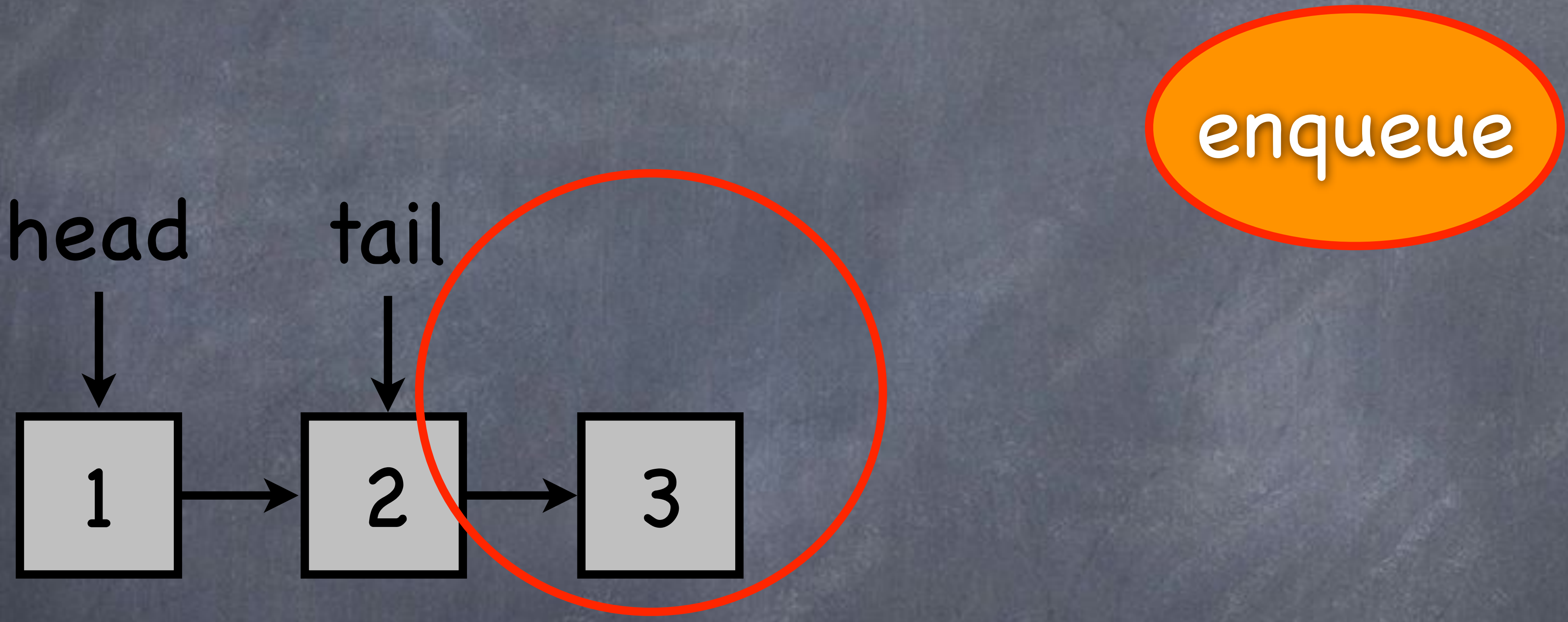
-> 1 lock -> 2 locks -> 0 locks

Concurrent First-in-First-out (FIFO) Queue



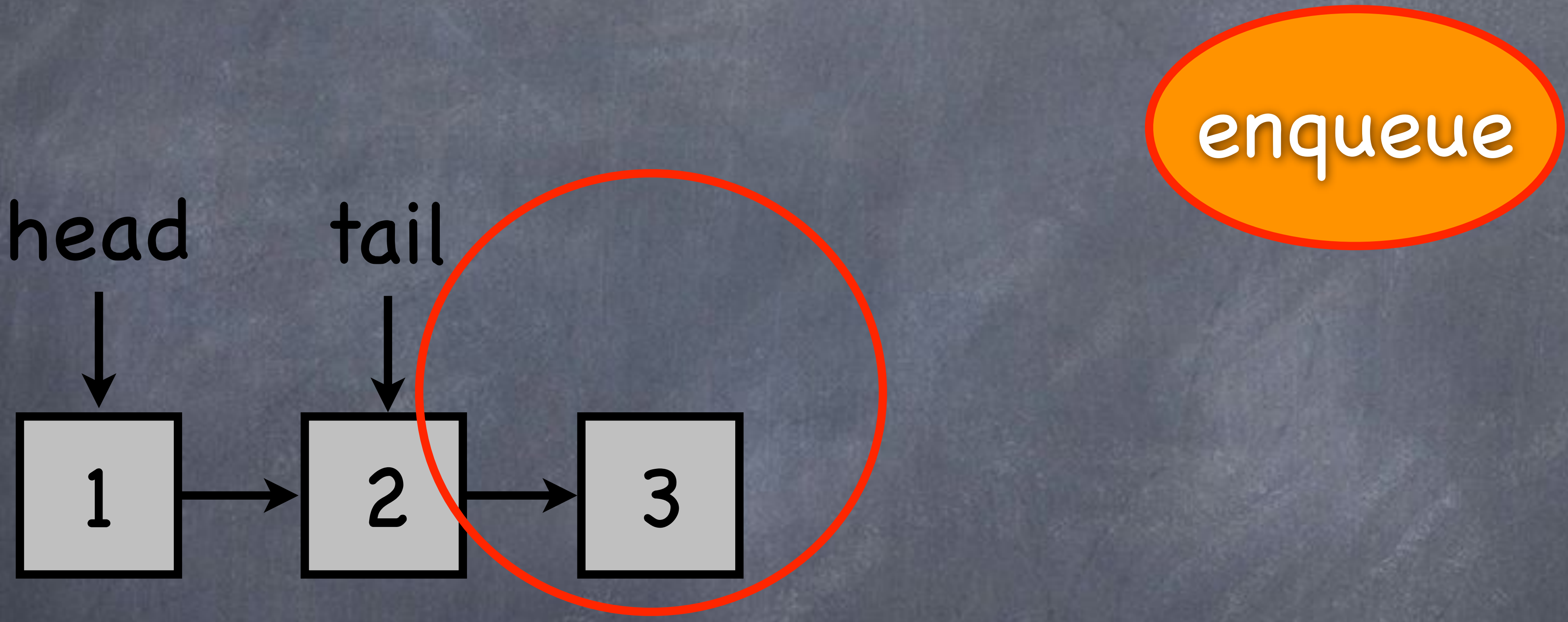
-> 1 lock -> 2 locks -> 0 locks

Concurrent First-in-First-out (FIFO) Queue



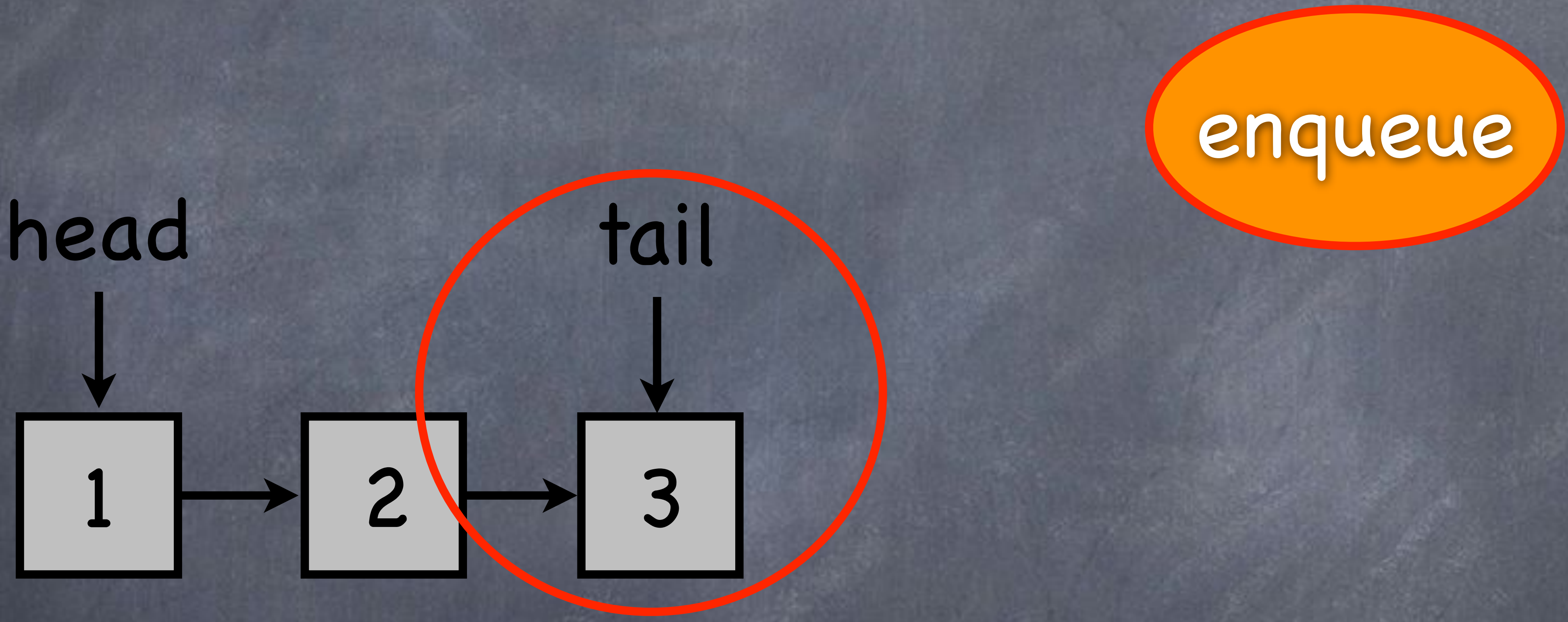
-> 1 lock -> 2 locks -> 0 locks

Concurrent First-in-First-out (FIFO) Queue



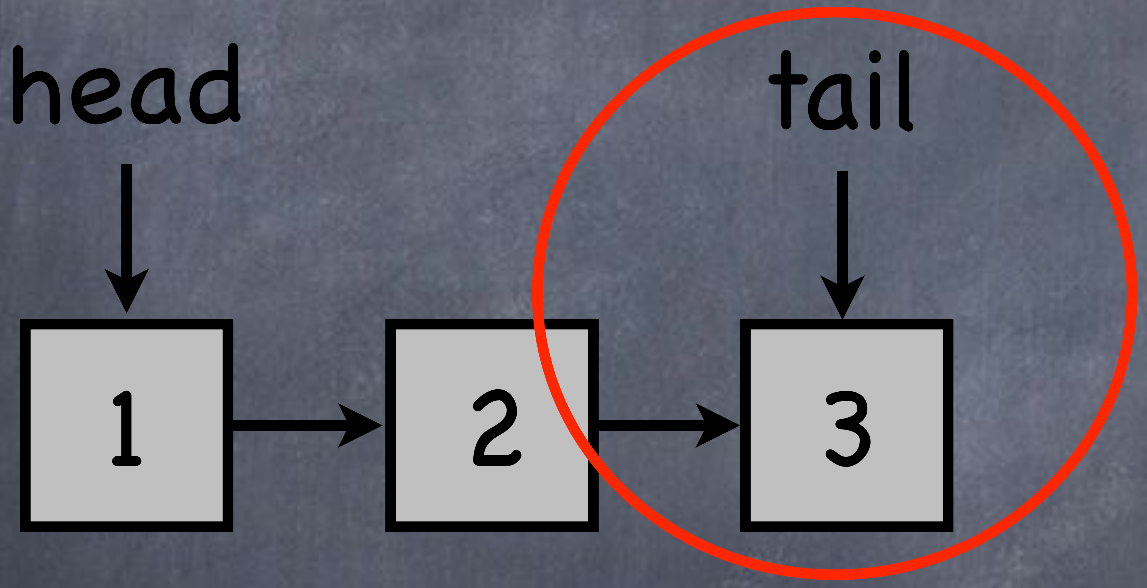
-> 1 lock -> 2 locks -> 0 locks -> compare & swap

Concurrent First-in-First-out (FIFO) Queue



-> 1 lock -> 2 locks -> 0 locks -> compare & swap

Concurrent First-in-First-out (FIFO) Queue

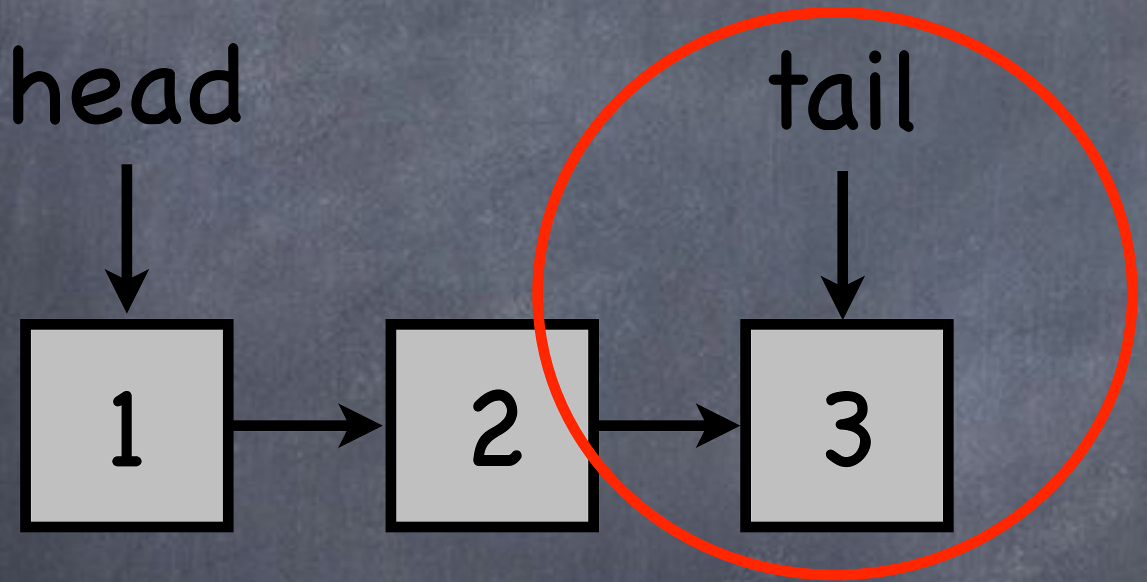


enqueue

[Michael,Scott'96]

-> 1 lock -> 2 locks -> 0 locks -> compare & swap

Concurrent First-in-First-out (FIFO) Queue

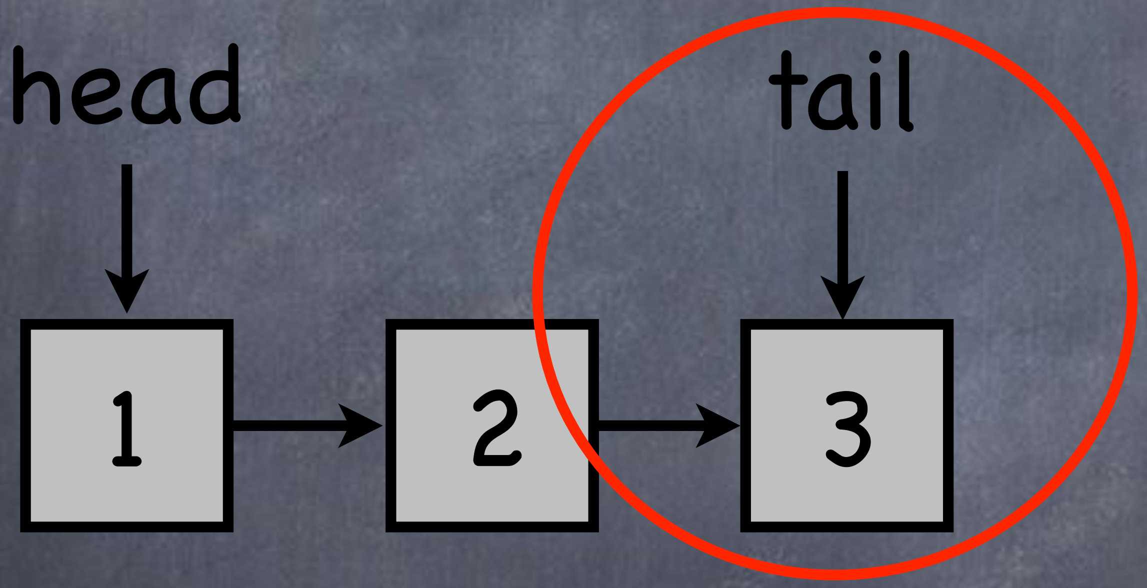


enqueue

[Michael,Scott'96]

-> 1 lock -> 2 locks -> 0 locks -> compare & swap
-> lock-based vs. lock-free vs. wait-free?

Concurrent First-in-First-out (FIFO) Queue

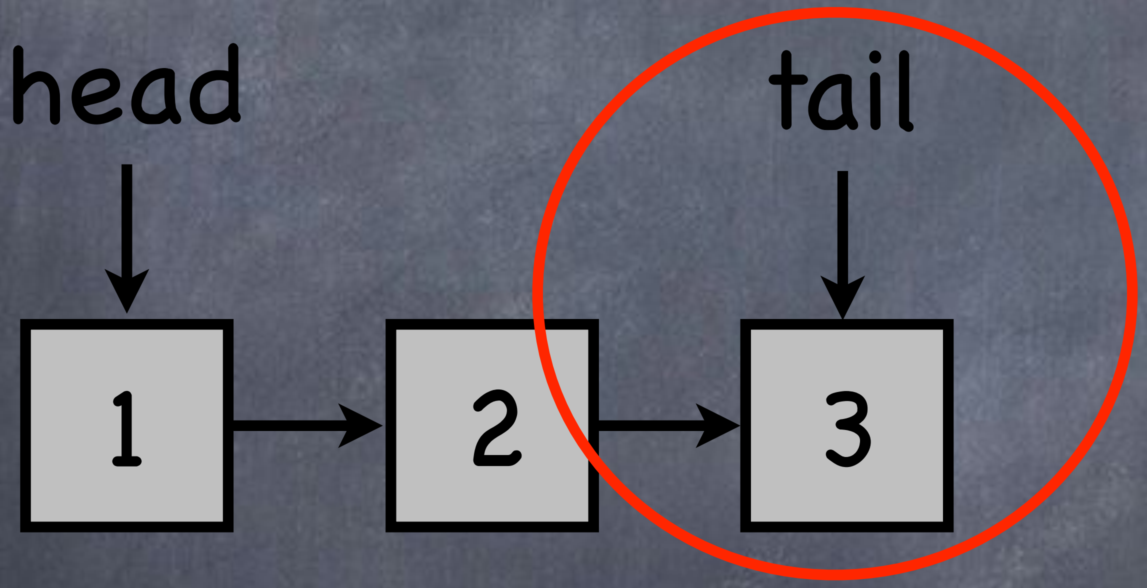


enqueue

[Michael,Scott'96]

-> 1 lock -> 2 locks -> 0 locks -> compare & swap
-> lock-based vs. lock-free vs. wait-free?
-> memory contention on head and tail pointers!

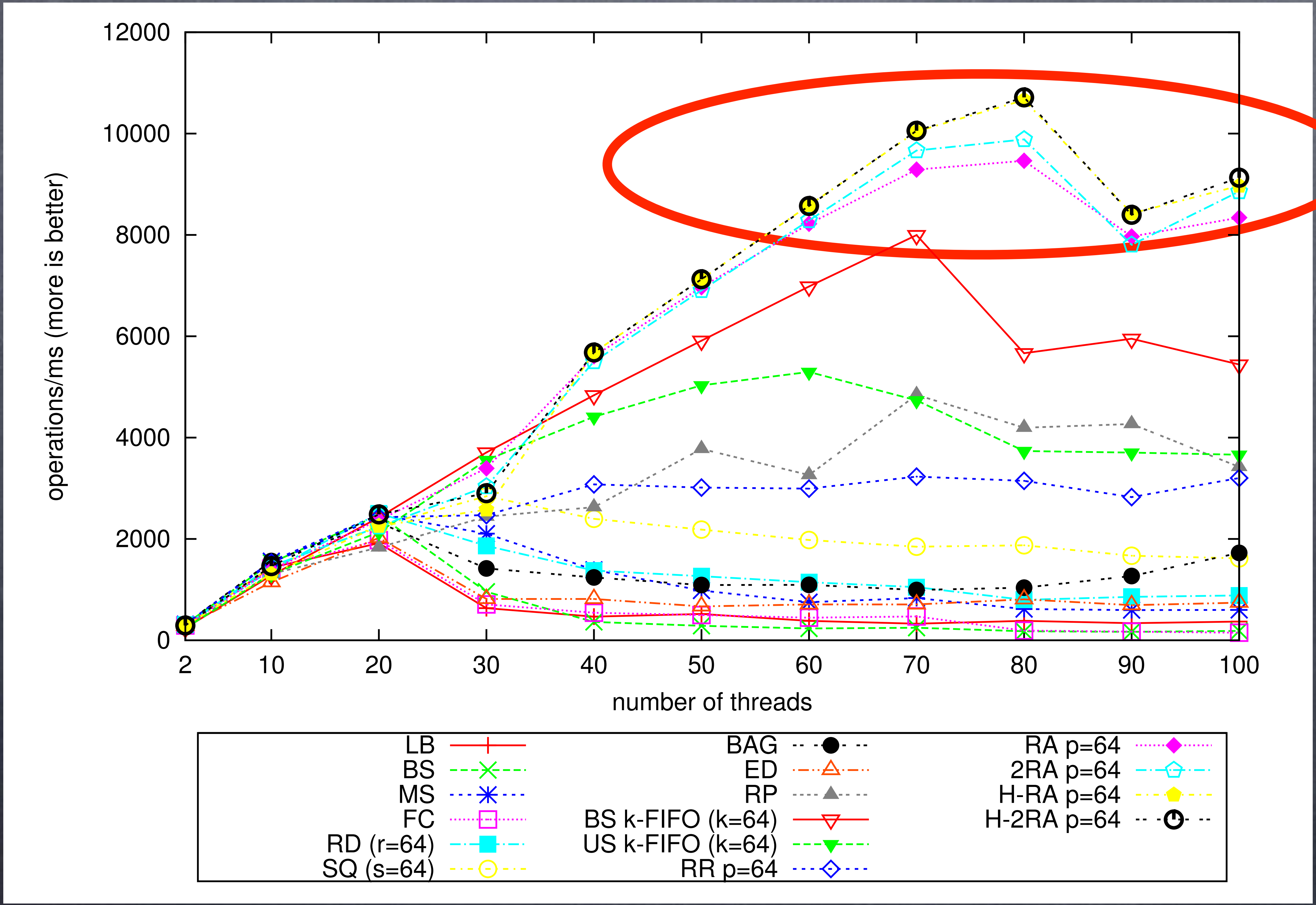
Concurrent First-in-First-out (FIFO) Queue



[Michael,Scott'96]

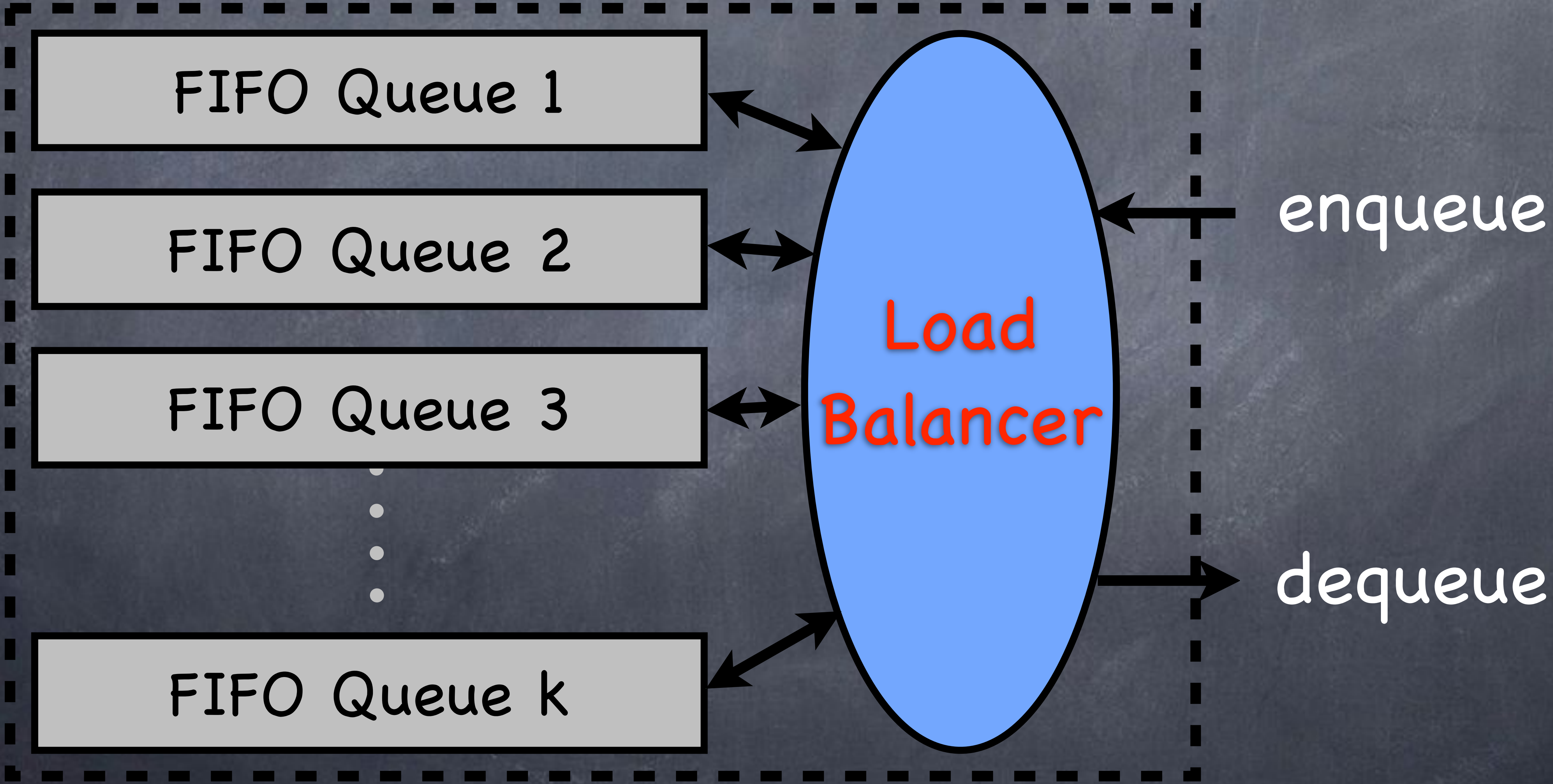
-> 1 lock -> 2 locks -> 0 locks -> compare & swap
-> lock-based vs. lock-free vs. wait-free?
-> memory contention on head and tail pointers!
-> and on next pointers!

Distributed Queues

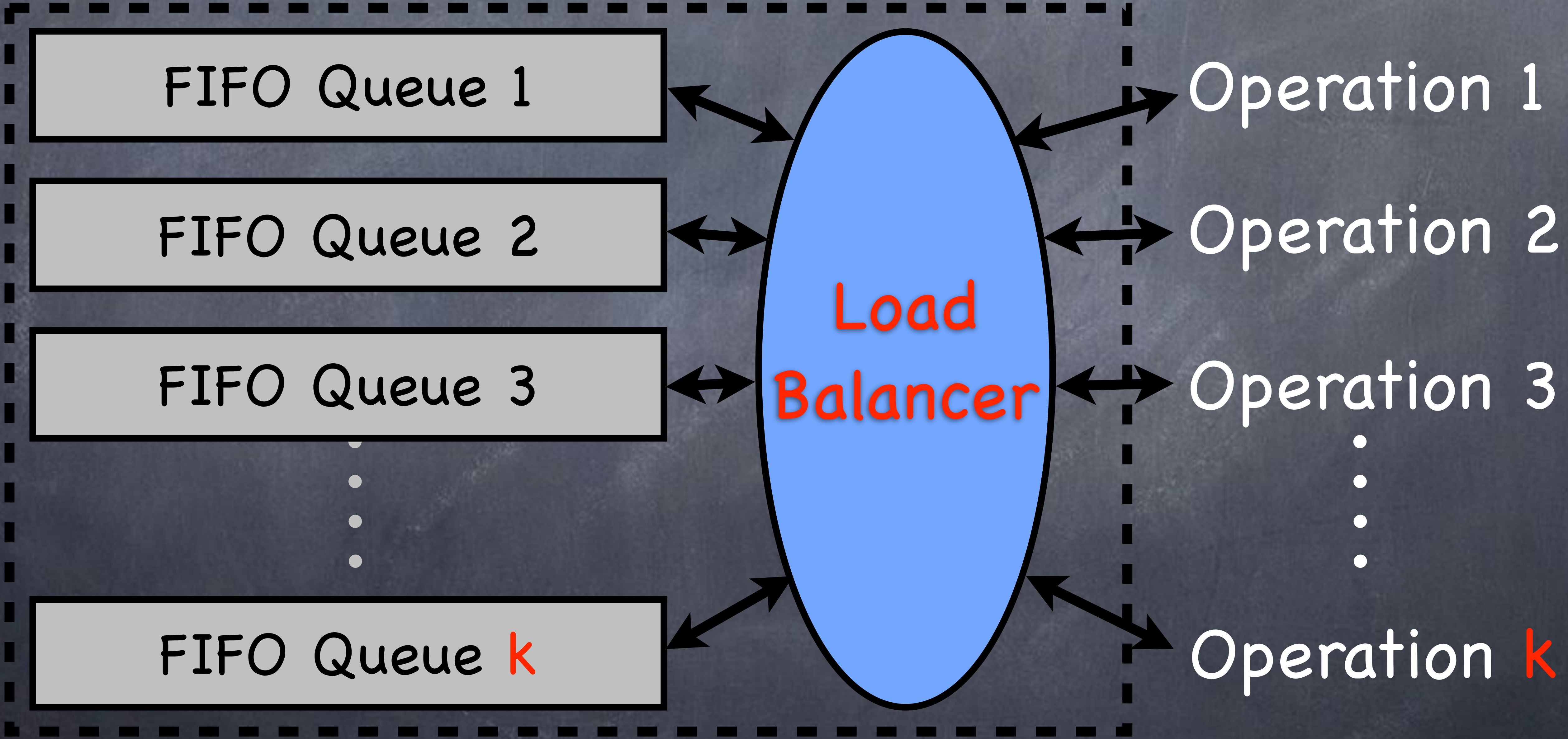


Distributed Queues

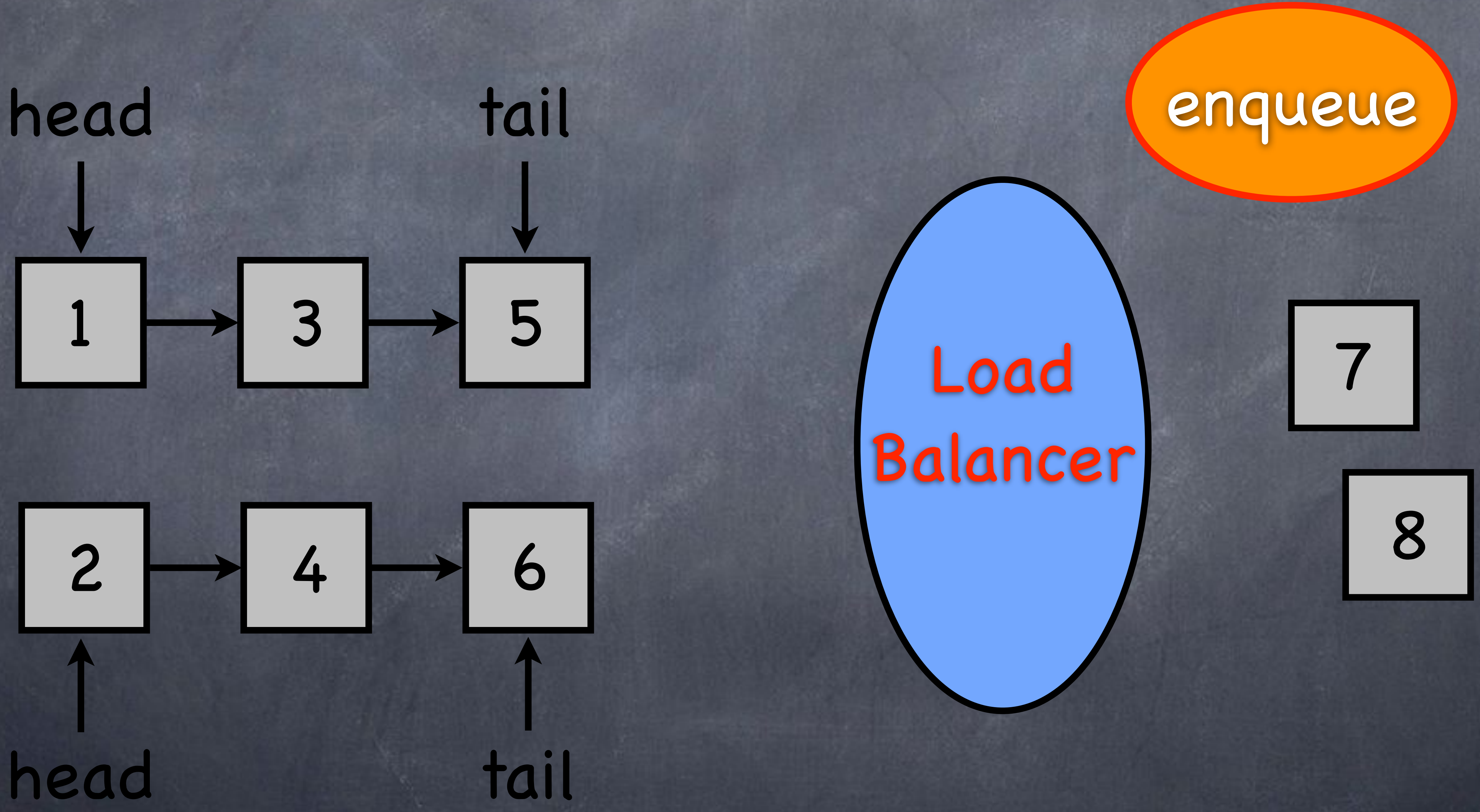
[_,Payer,Röck,Sokolova'12]



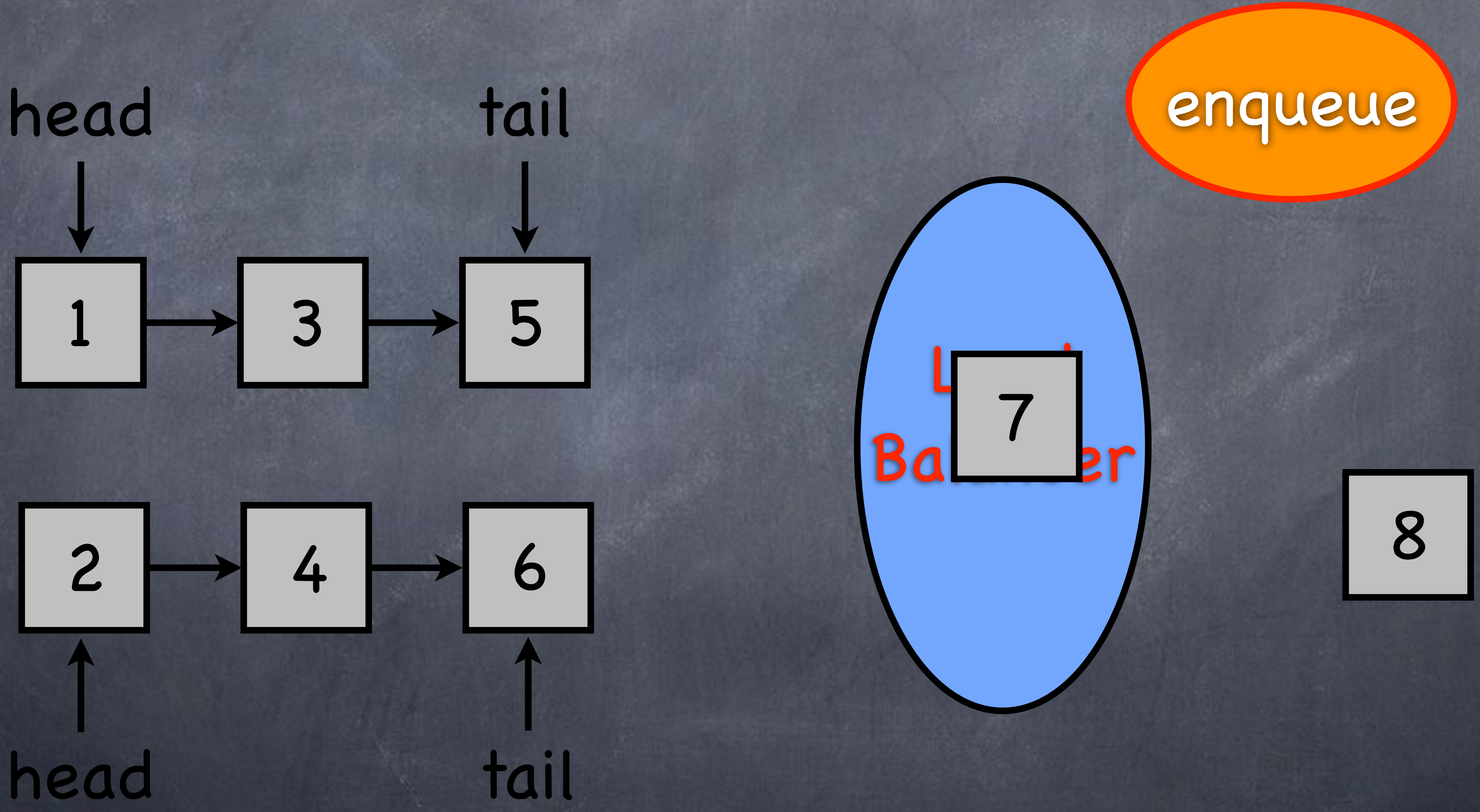
Up to k Parallel Enqueues and k Parallel Dequeues



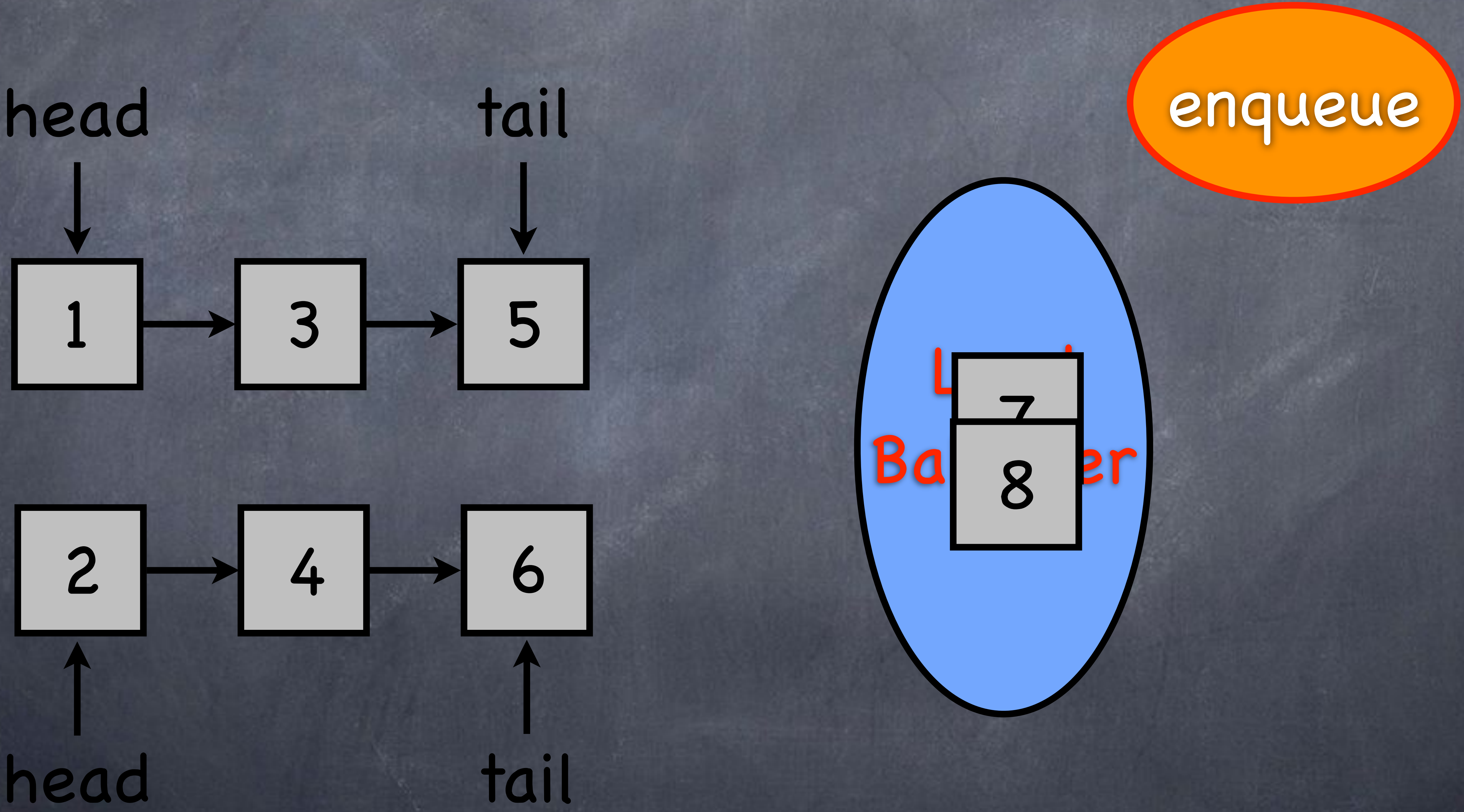
Load Balancing



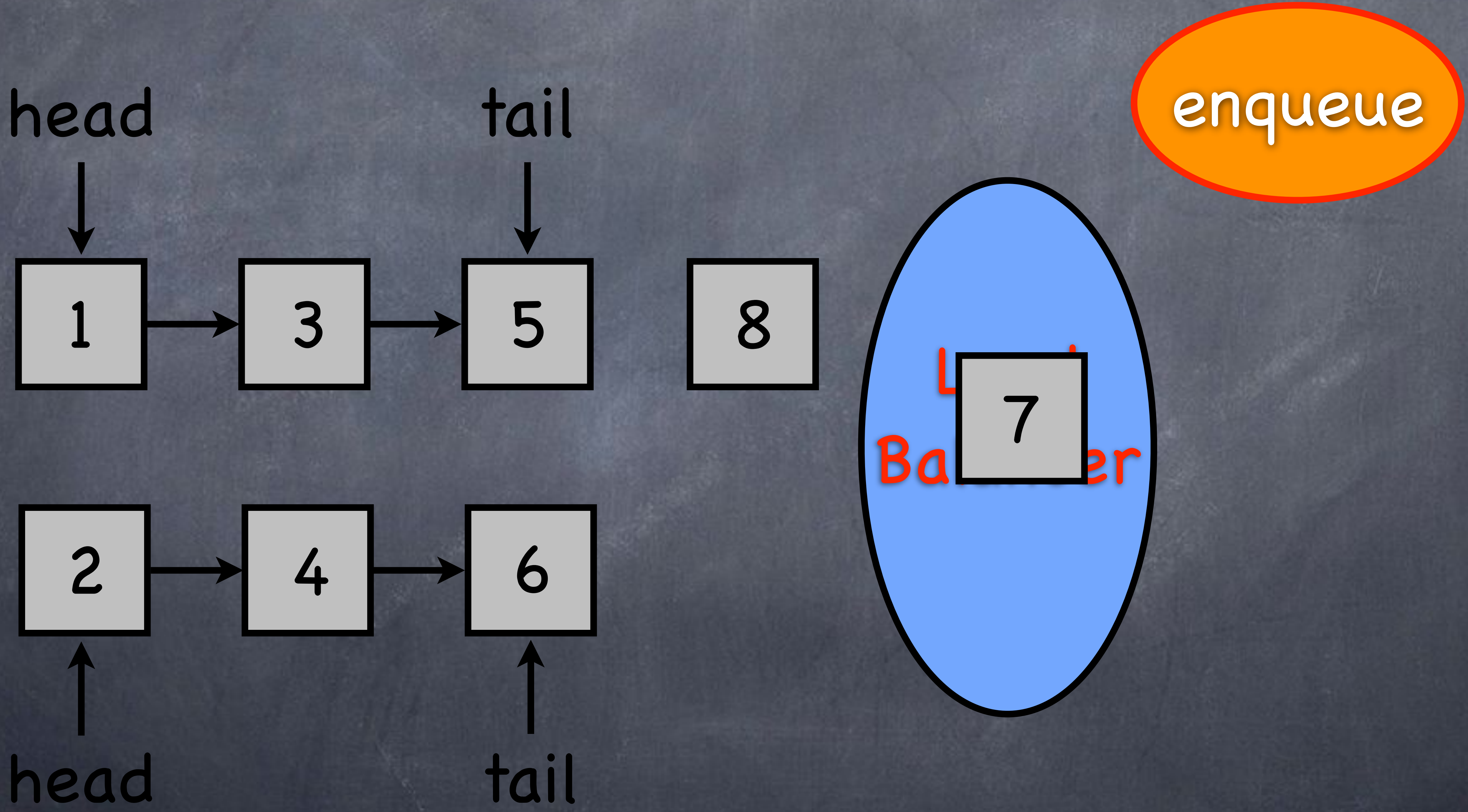
Load Balancing



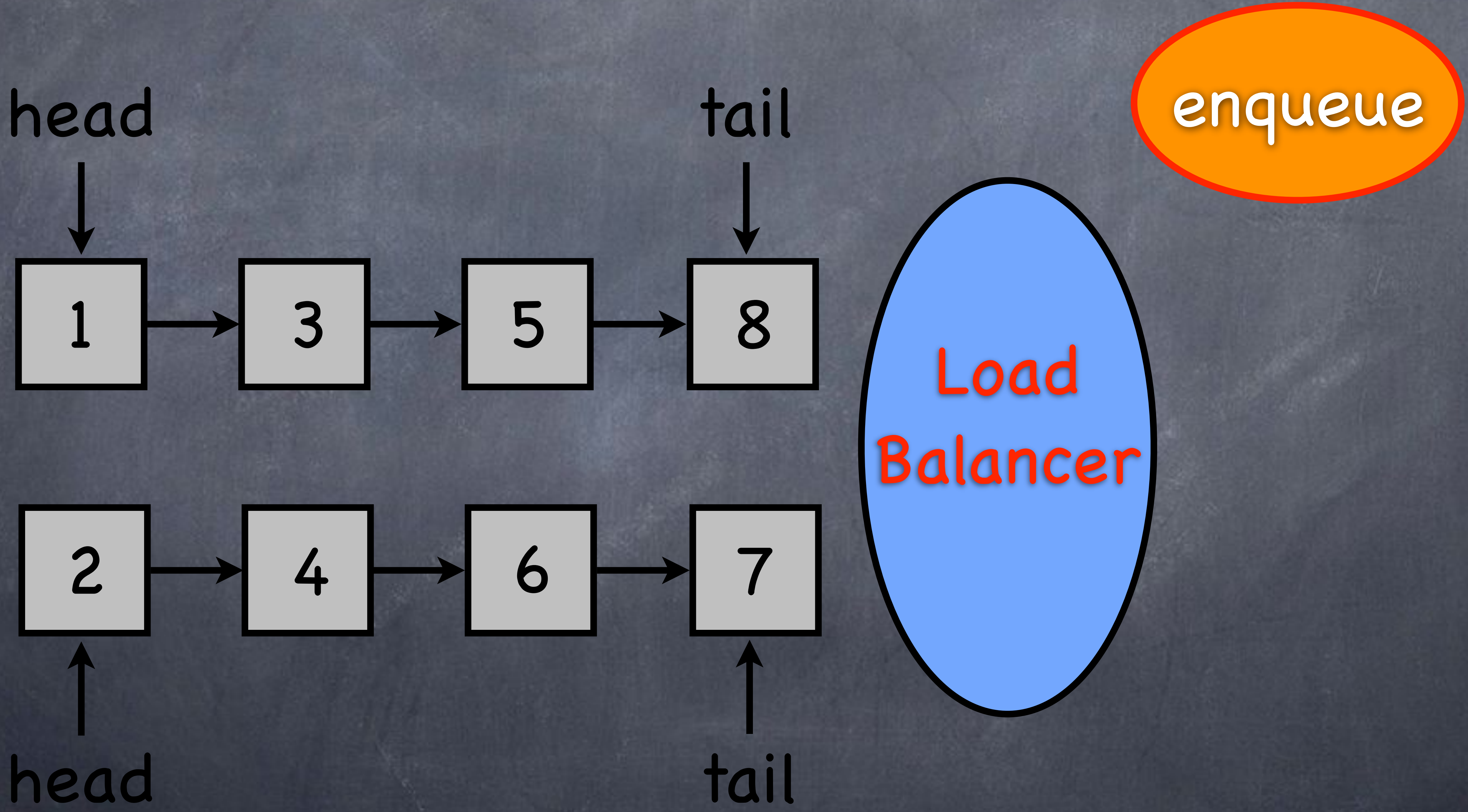
Load Balancing



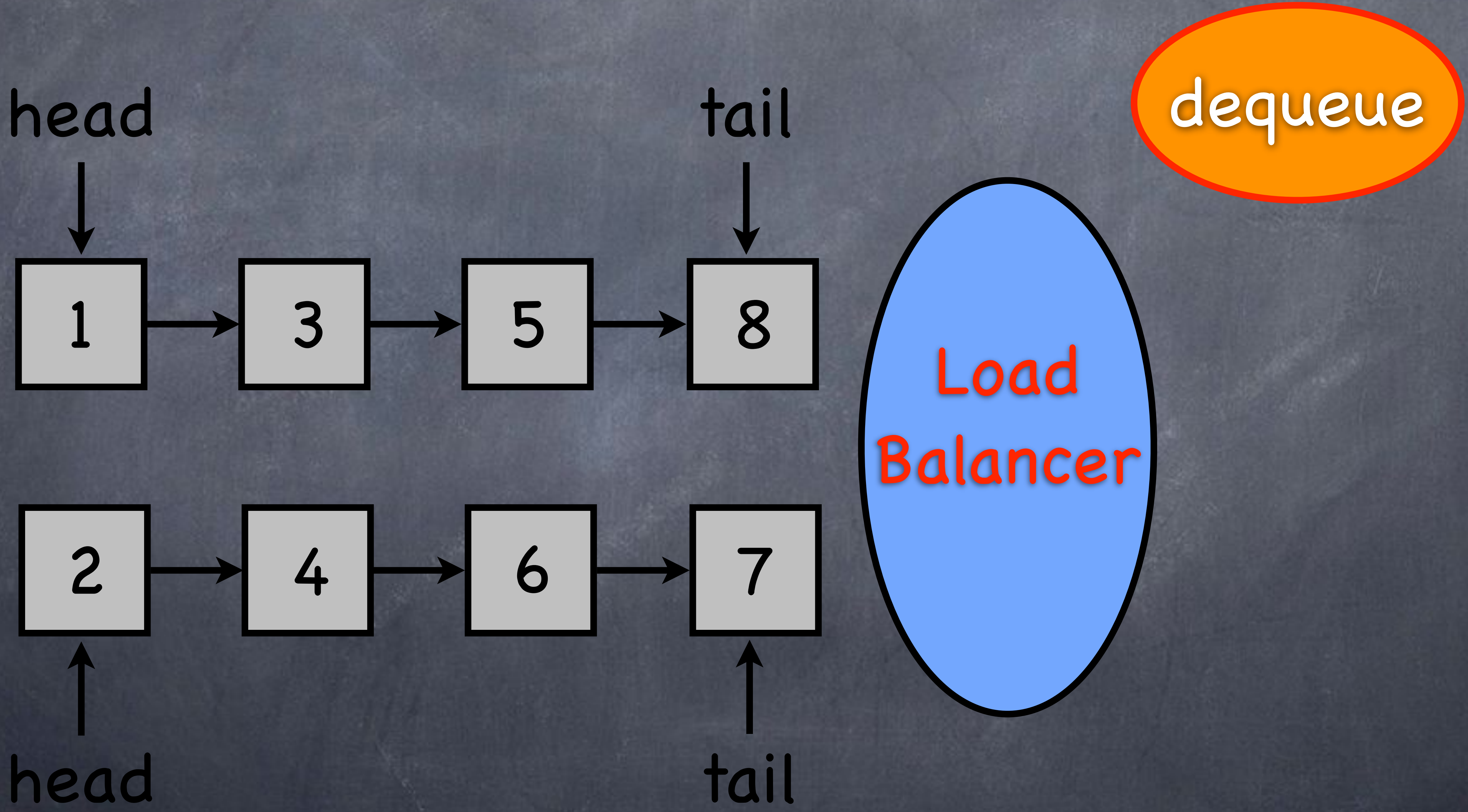
Load Balancing



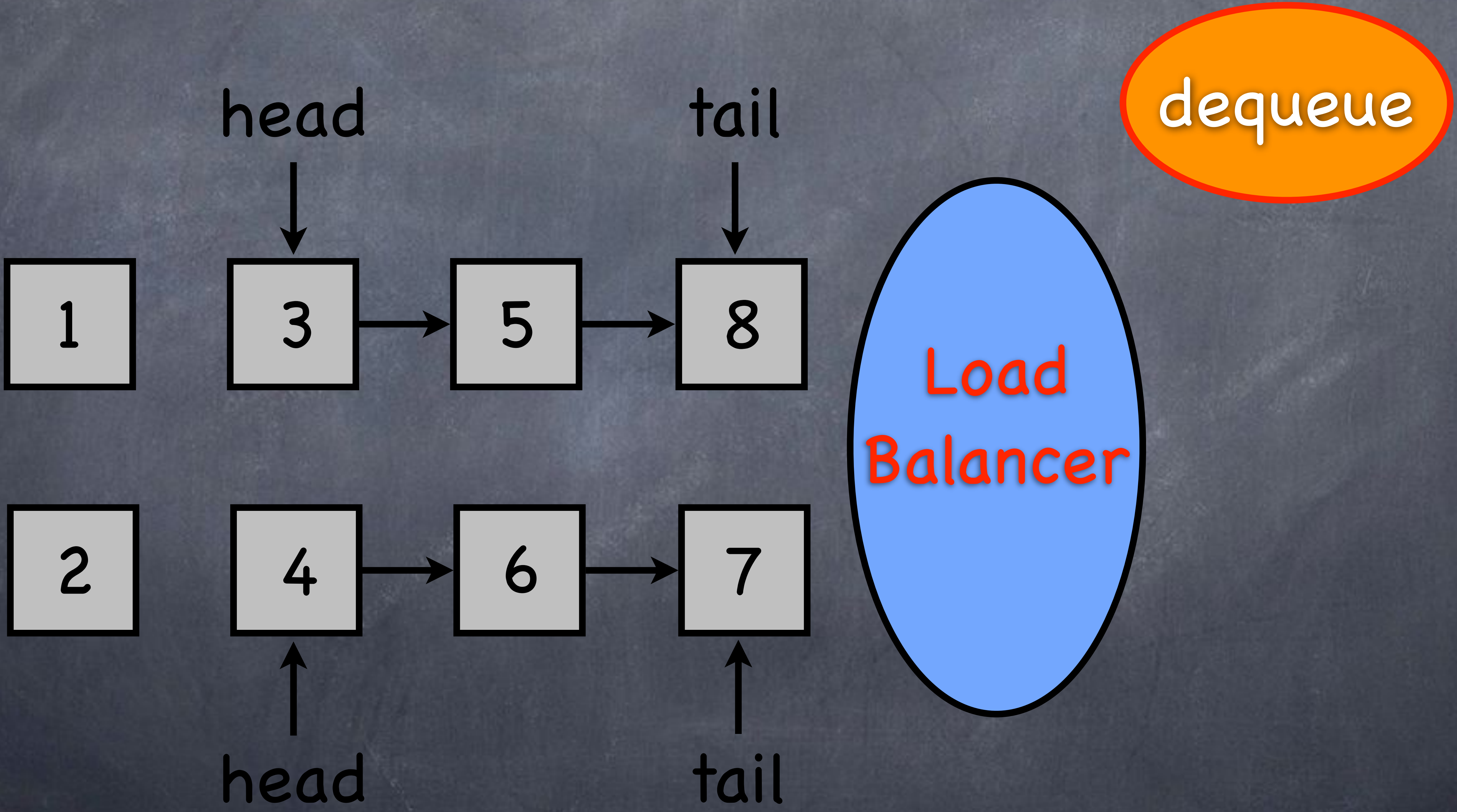
Load Balancing



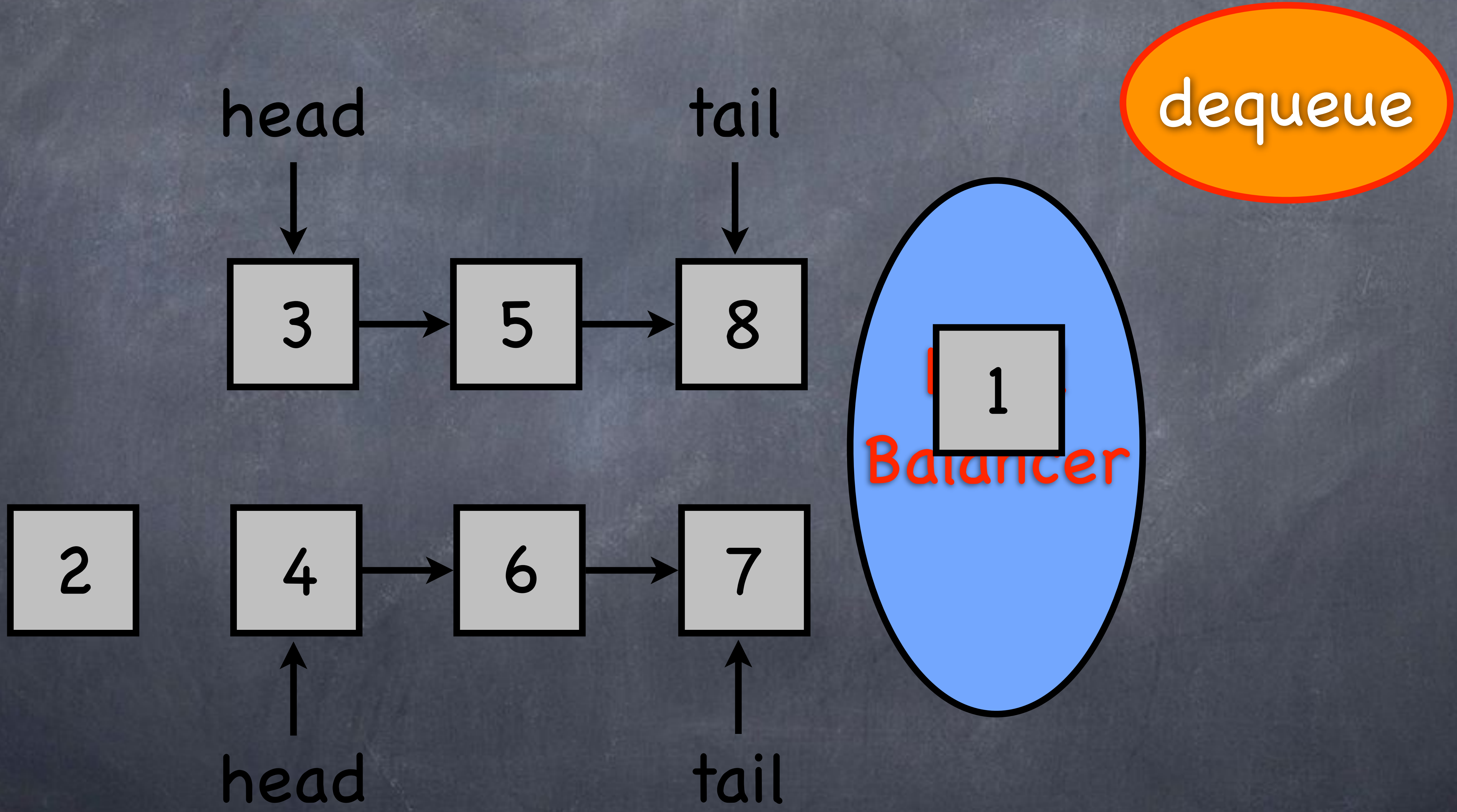
Load Balancing



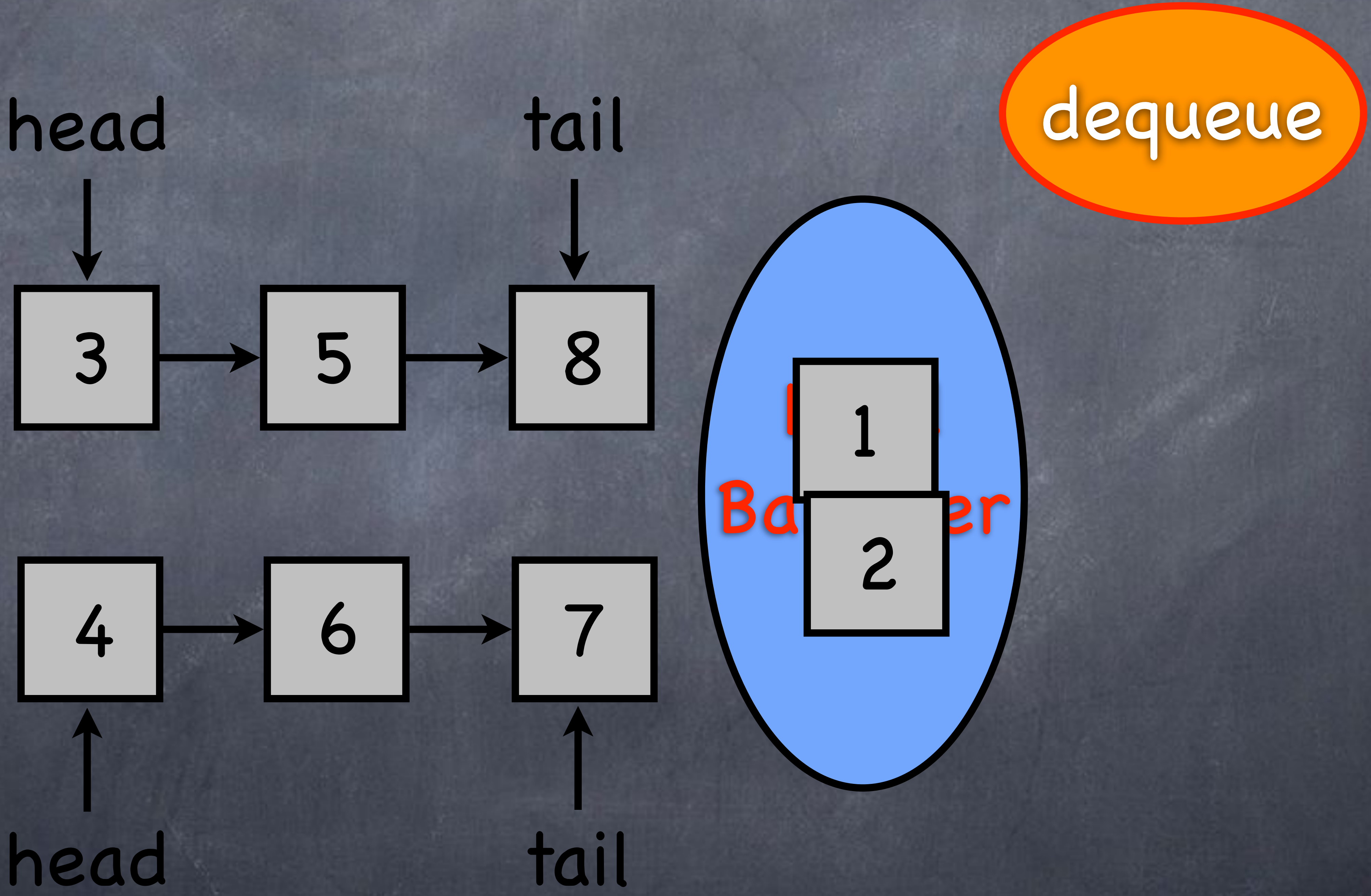
Load Balancing



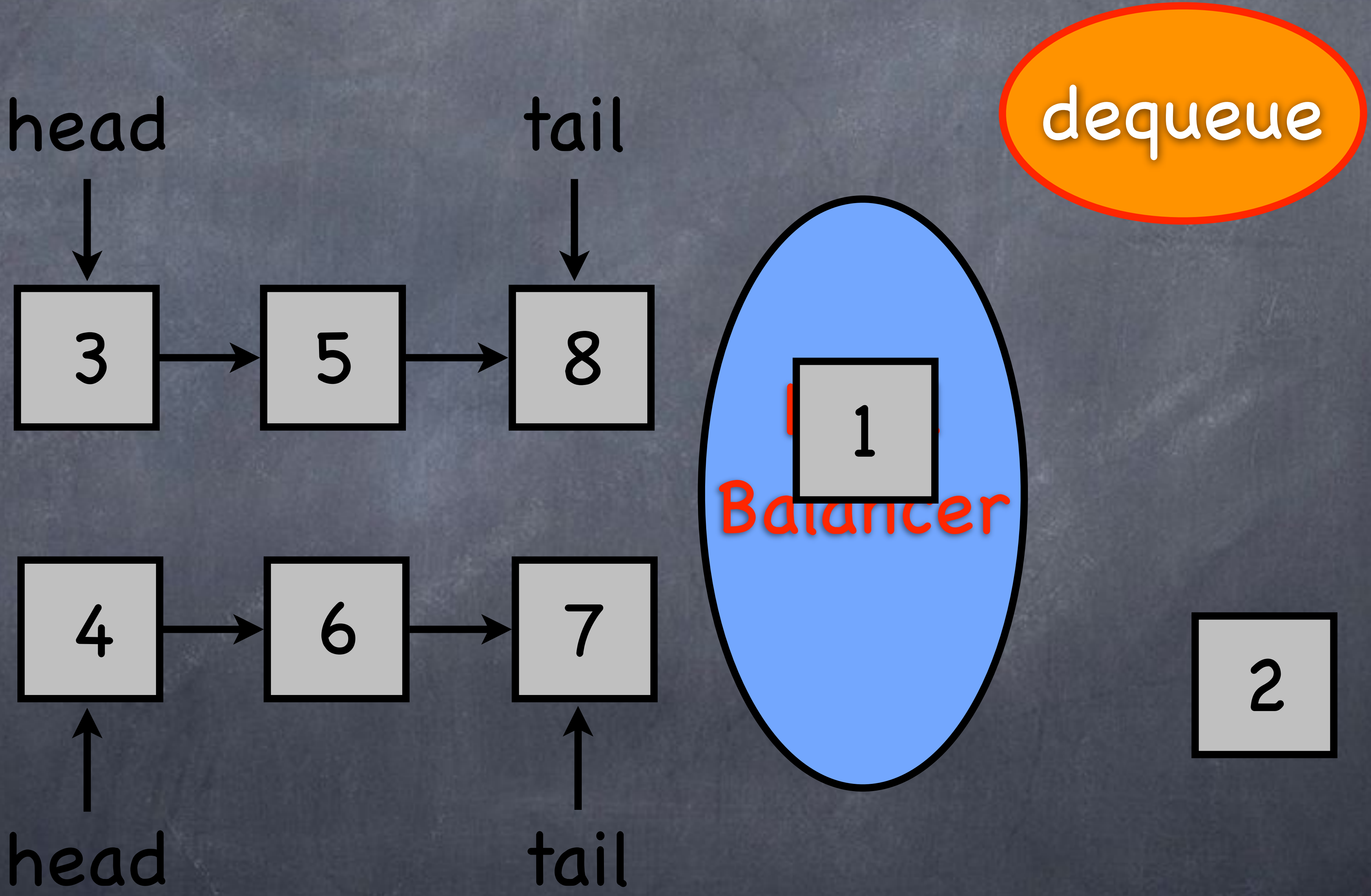
Load Balancing



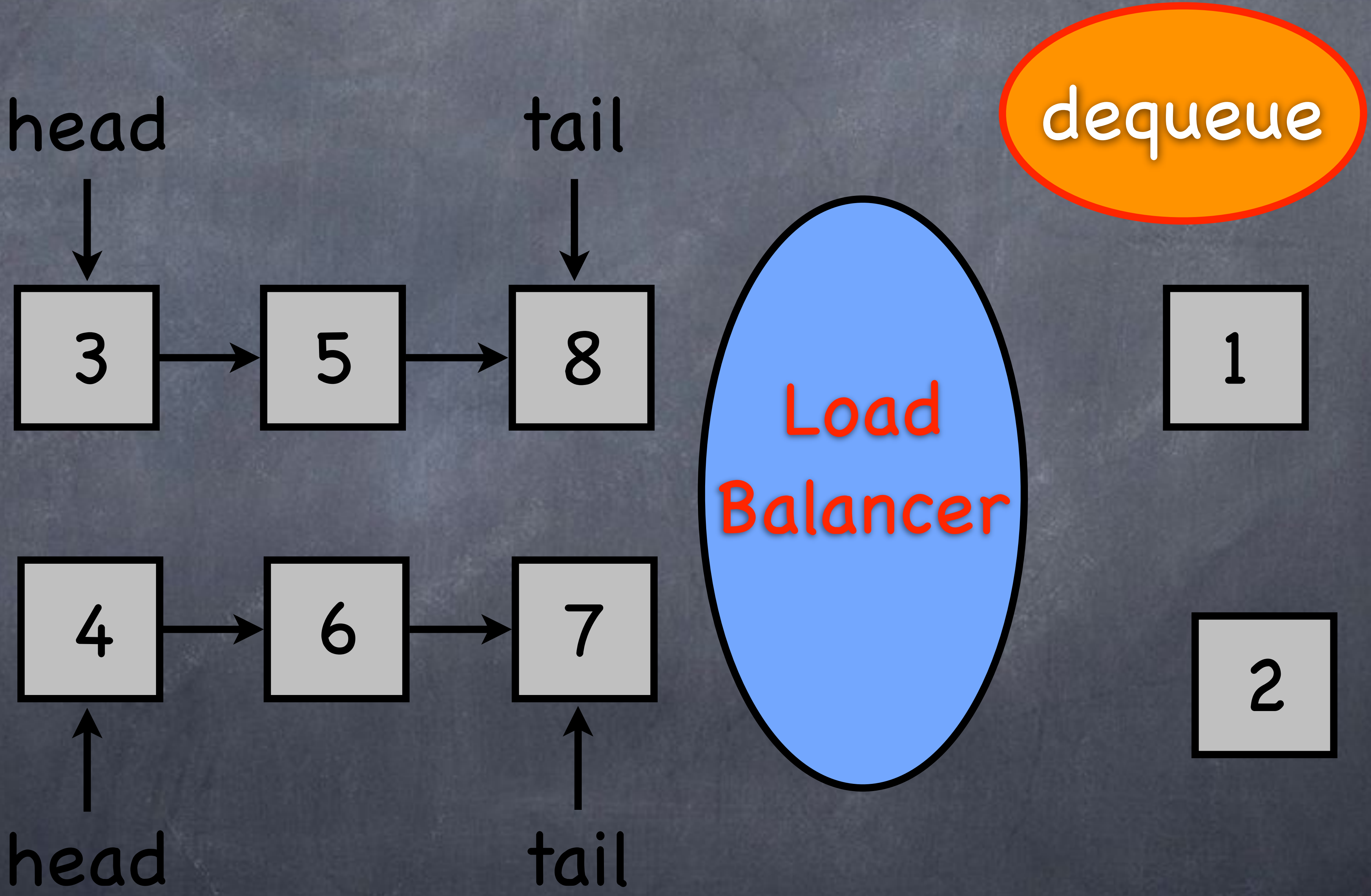
Load Balancing



Load Balancing

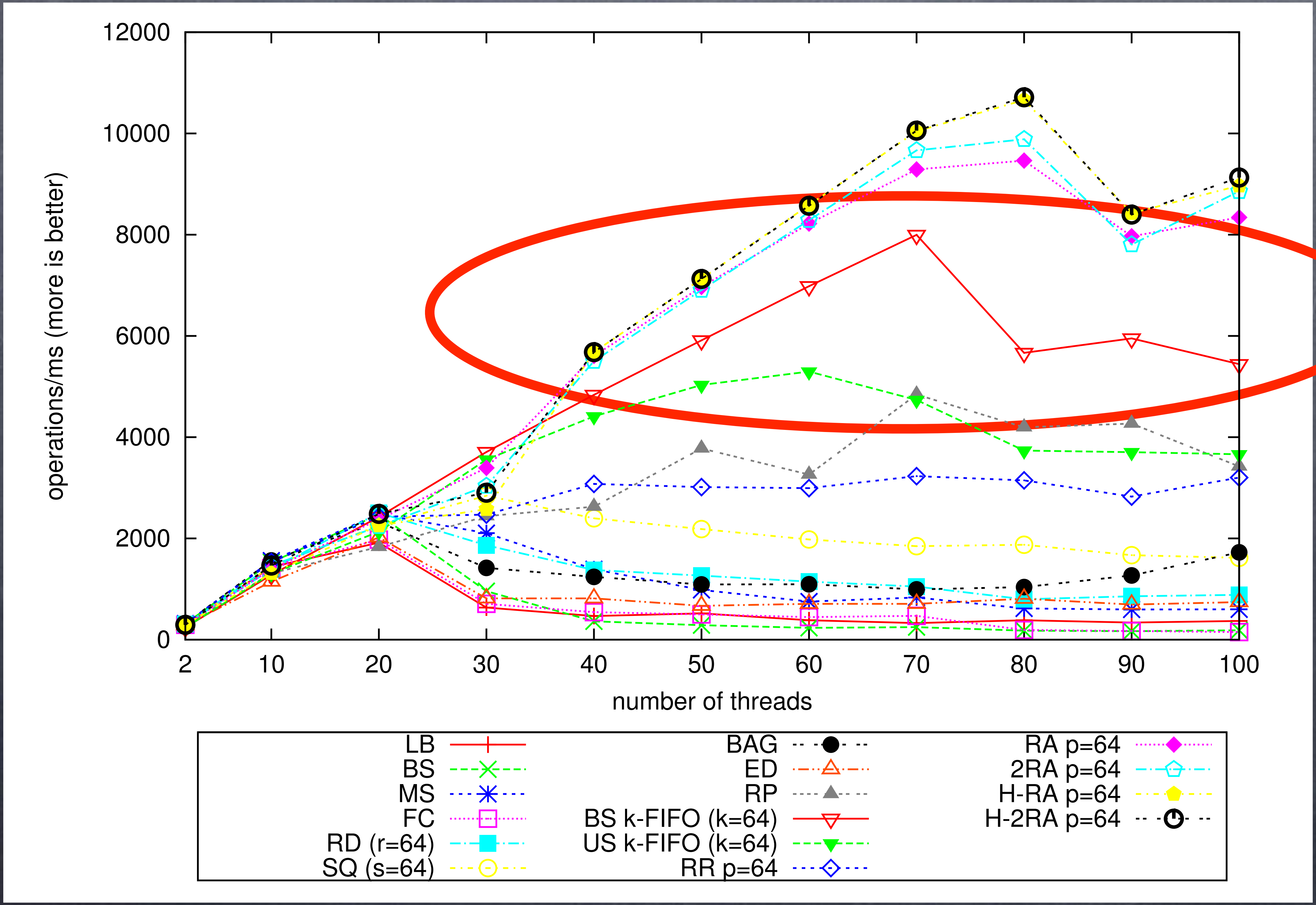


Load Balancing



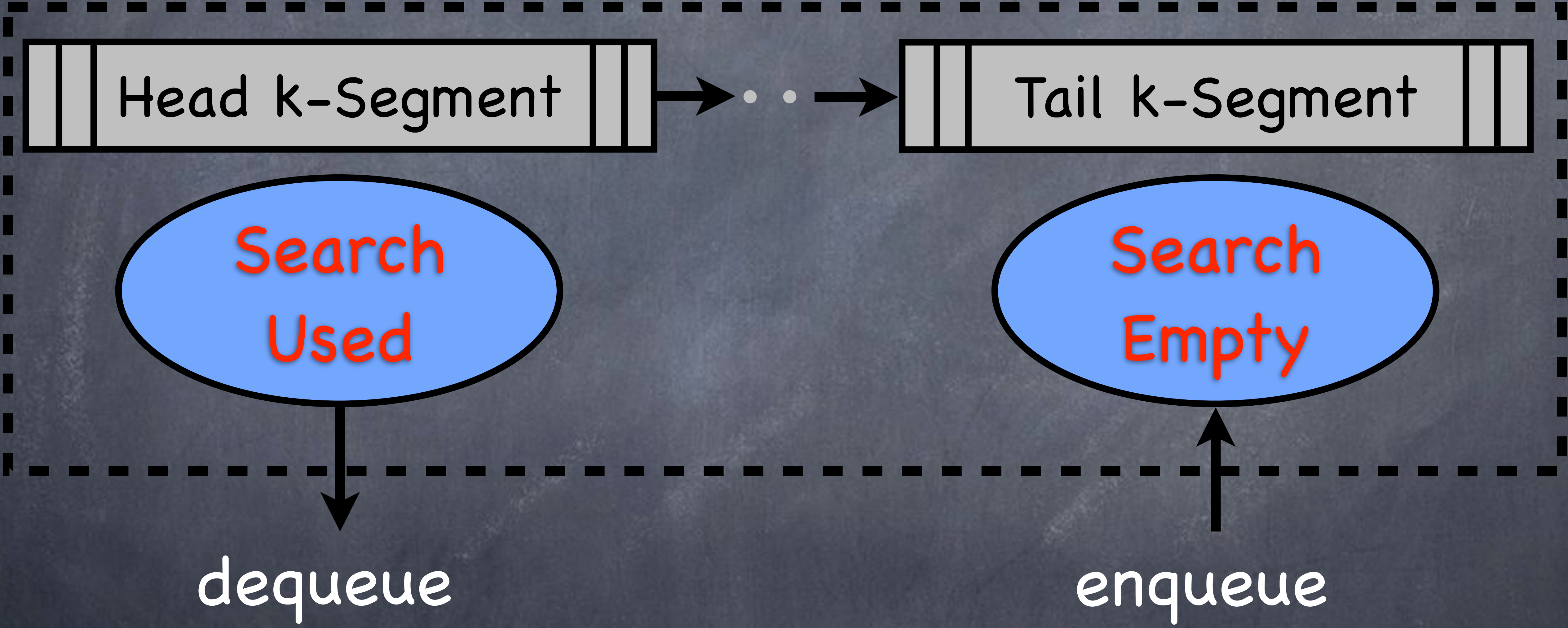
Emptiness Check?

Segmented Queues



Segmented Queues

[Afek,Korland,Yanovsky'10],[_,Lippautz,Payer'12]



Emptiness Check?

Concurrent k-FIFO Queue

- with a k-FIFO queue elements may be returned out-of-FIFO order up to k

Concurrent k-FIFO Queue

- with a k-FIFO queue elements may be returned out-of-FIFO order up to k
- the oldest element is returned after at most k+1 dequeue operations that may return elements not younger than k (or return nothing)

Concurrent k-FIFO Queue

- with a k-FIFO queue elements may be returned out-of-FIFO order up to k
- the oldest element is returned after at most k+1 dequeue operations that may return elements not younger than k (or return nothing)
- starvation-free for finite k

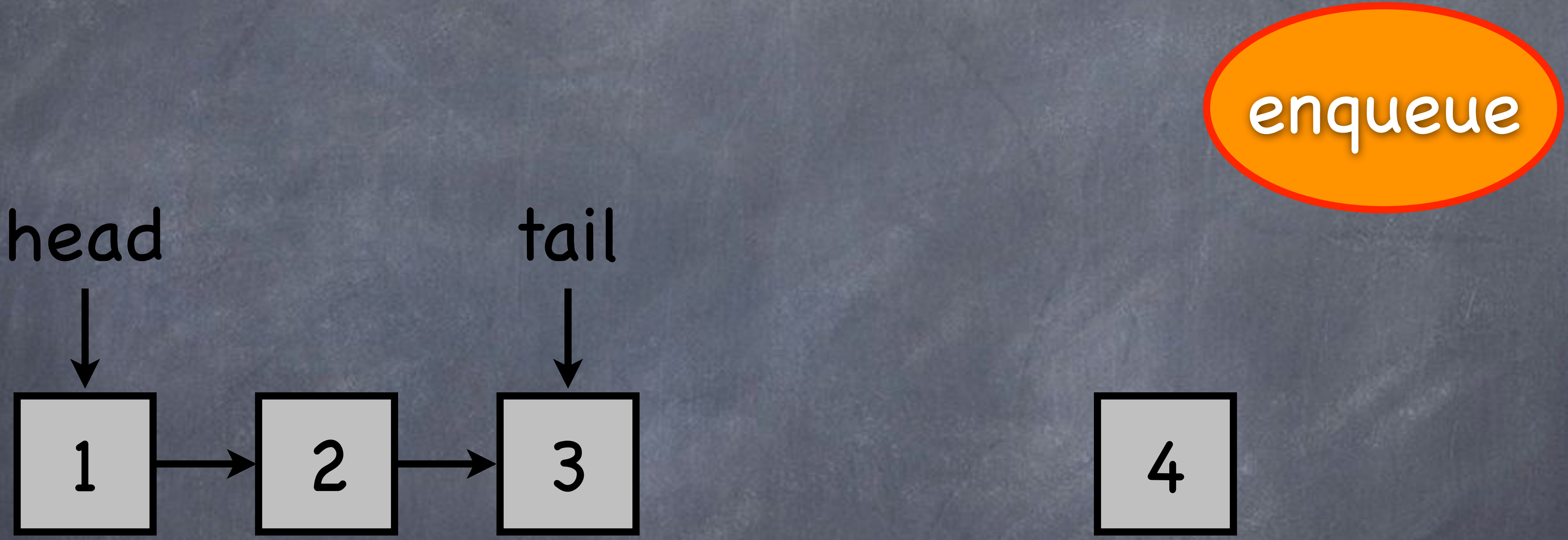
Concurrent k-FIFO Queue

- with a k-FIFO queue elements may be returned out-of-FIFO order up to k
- the oldest element is returned after at most k+1 dequeue operations that may return elements not younger than k (or return nothing)
- starvation-free for finite k
- 0-FIFO queue = regular FIFO queue

Concurrent k-FIFO Queue

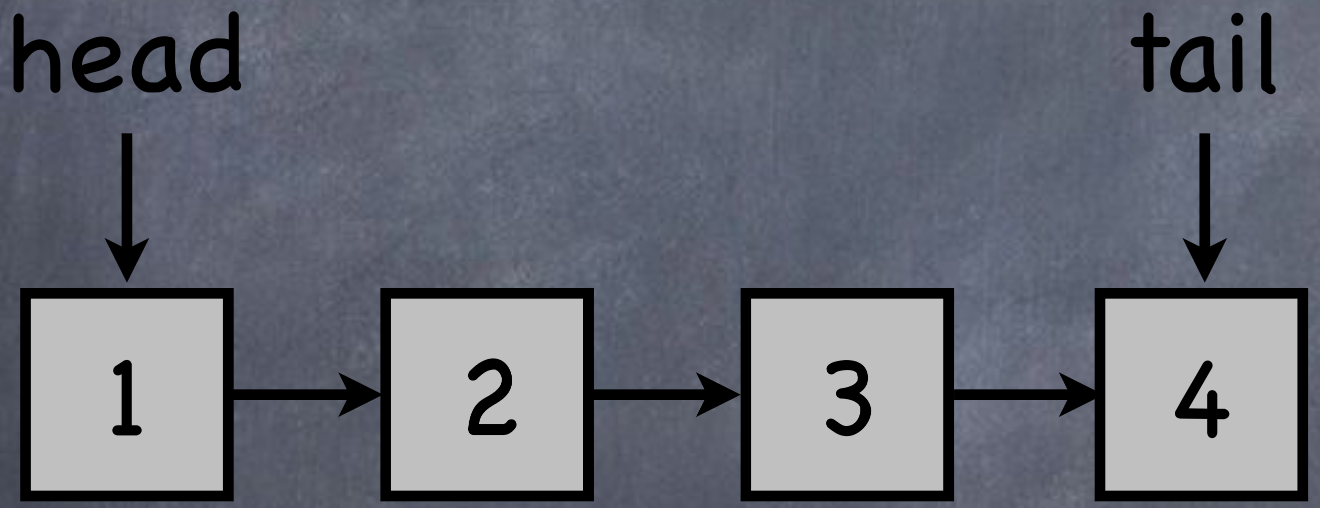
- with a k-FIFO queue elements may be returned out-of-FIFO order up to k
- the oldest element is returned after at most k+1 dequeue operations that may return elements not younger than k (or return nothing)
- starvation-free for finite k
- 0-FIFO queue = regular FIFO queue
- bigger k -> better performance, scalability?

Concurrent 2-FIFO Queue (k=2)

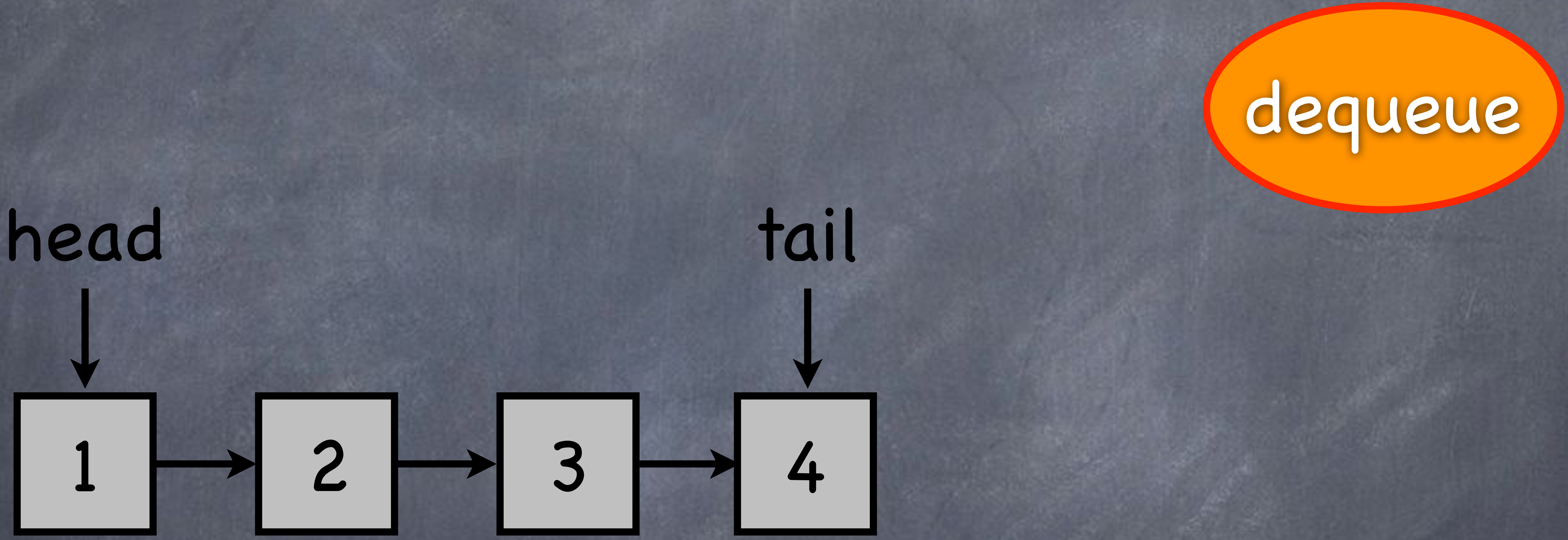


Concurrent 2-FIFO Queue (k=2)

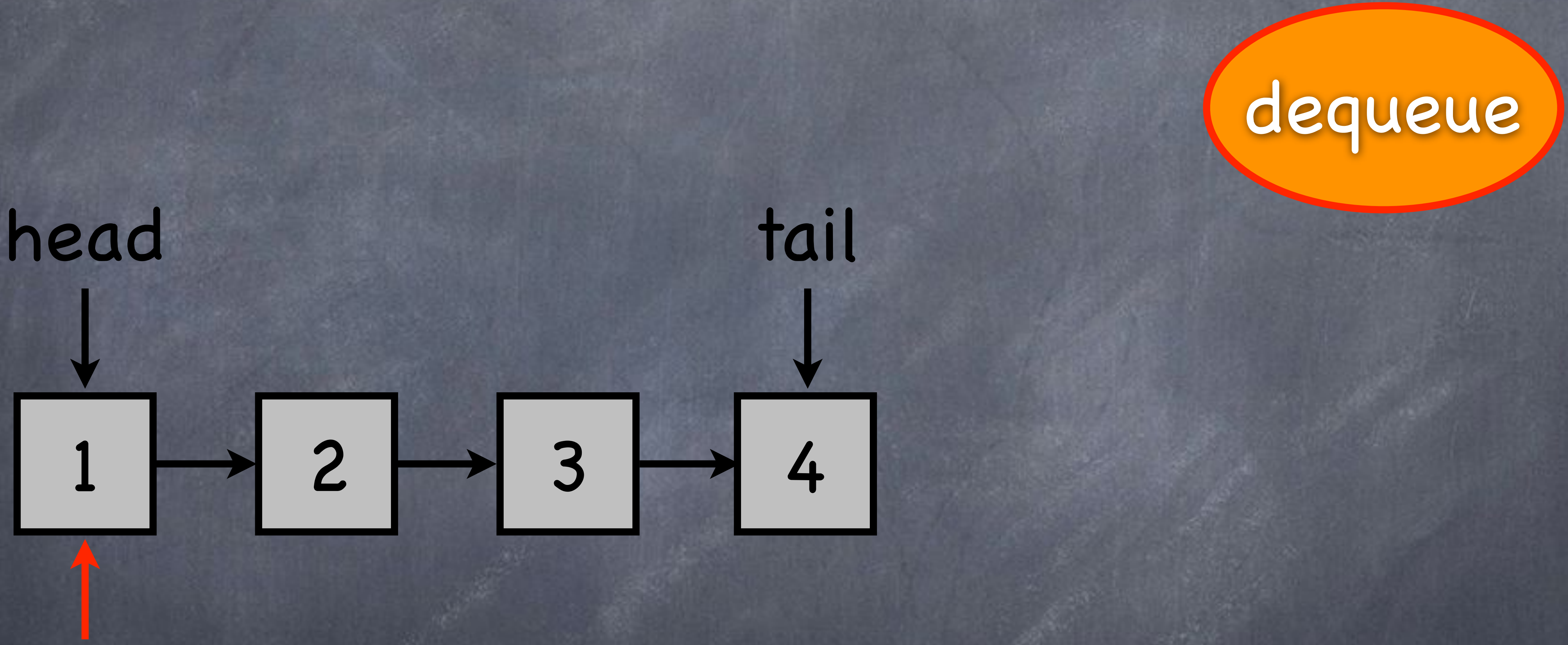
enqueue



Concurrent 2-FIFO Queue (k=2)

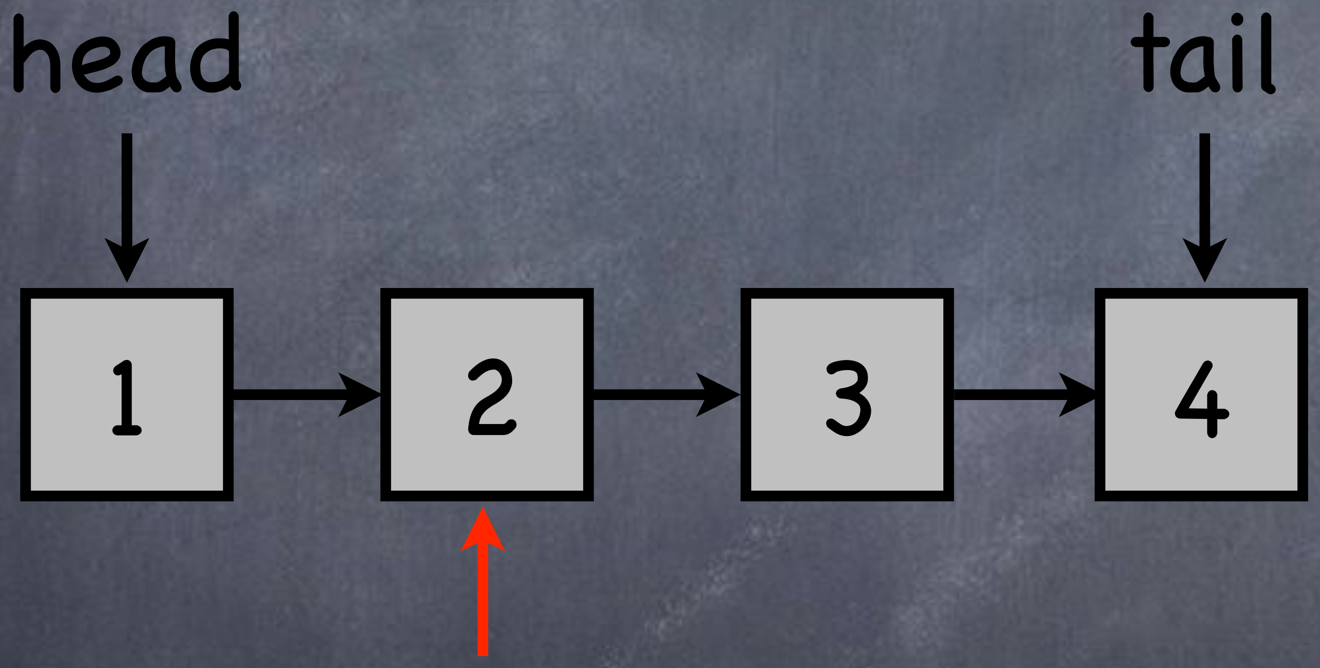


Concurrent 2-FIFO Queue (k=2)

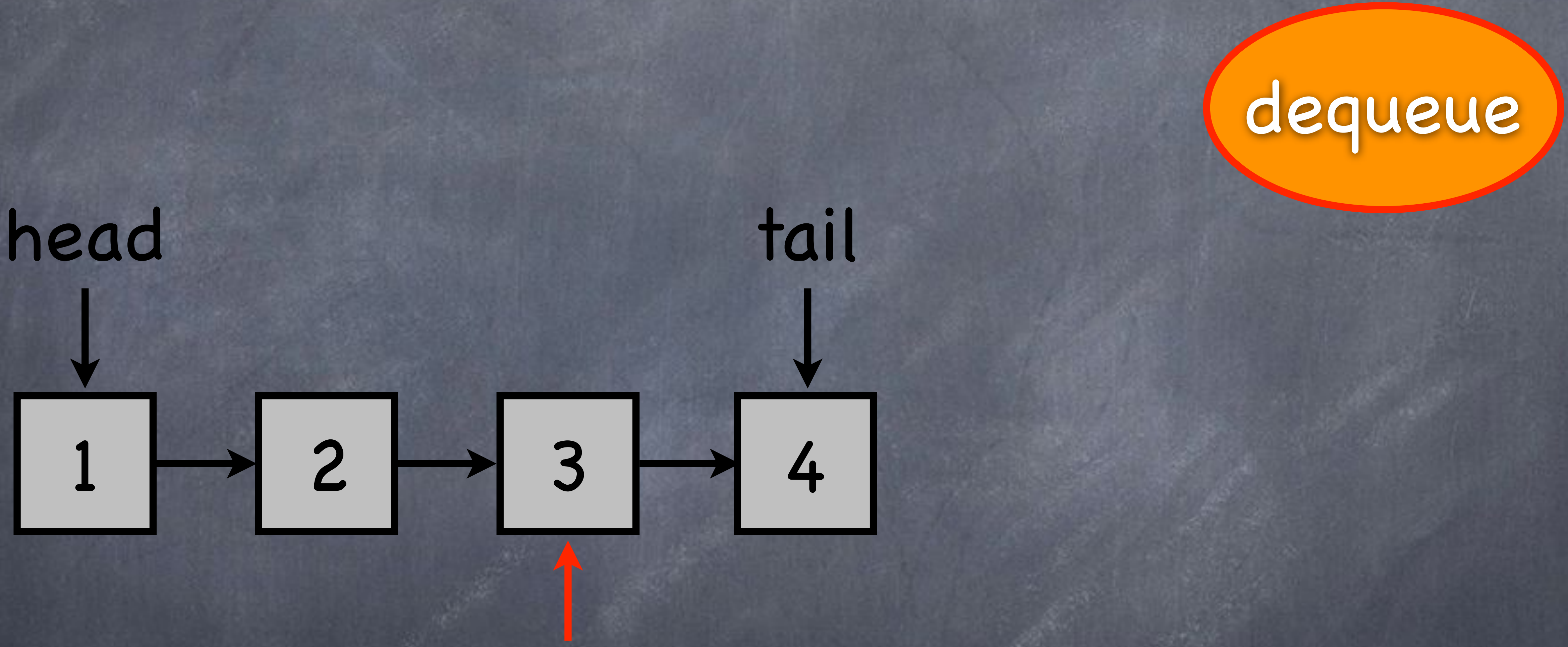


Concurrent 2-FIFO Queue (k=2)

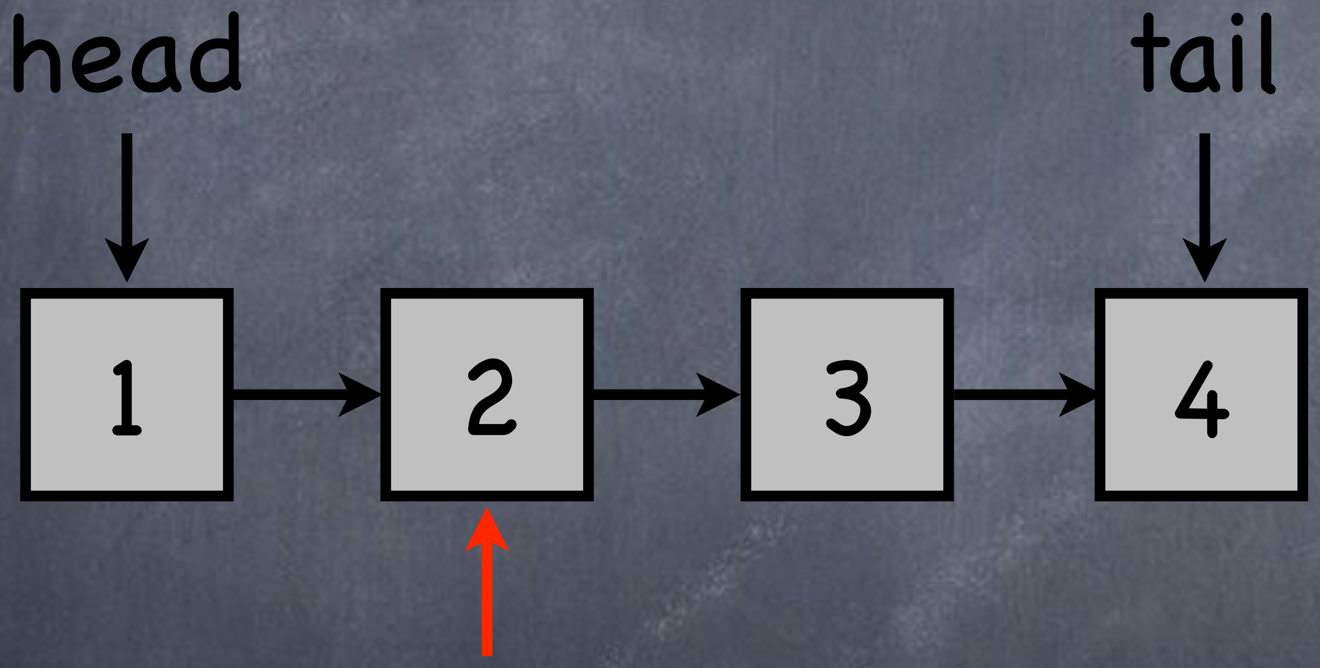
dequeue



Concurrent 2-FIFO Queue (k=2)

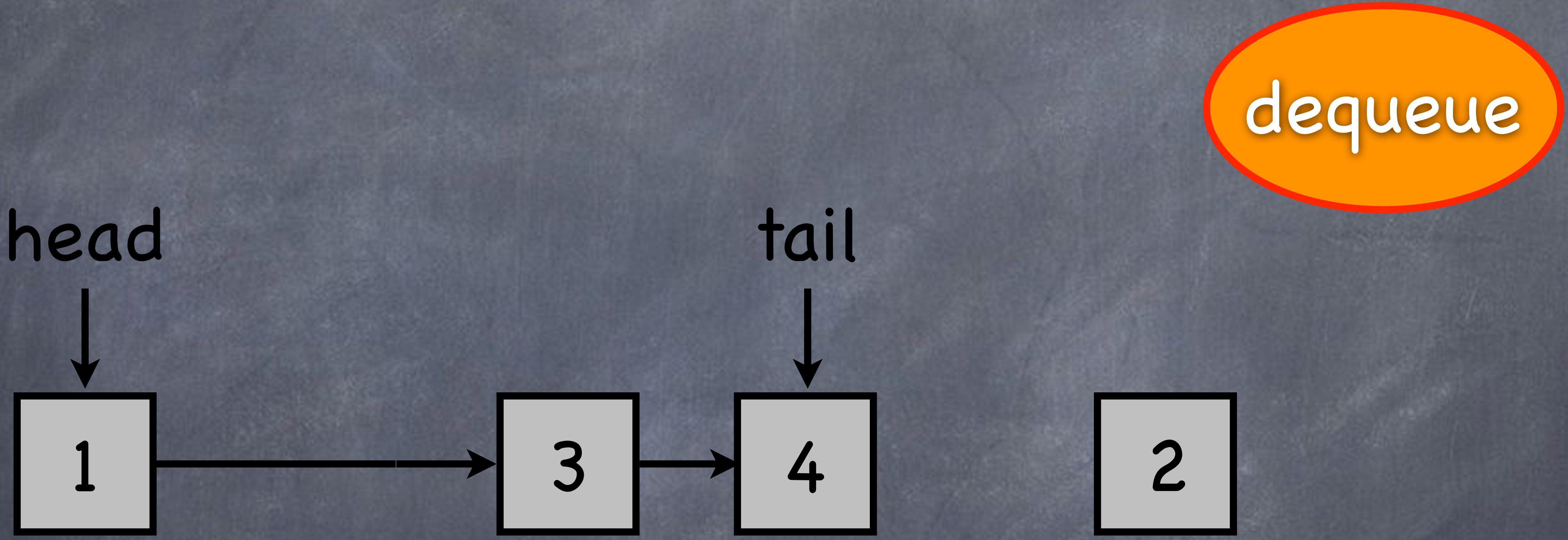


Concurrent 2-FIFO Queue (k=2)

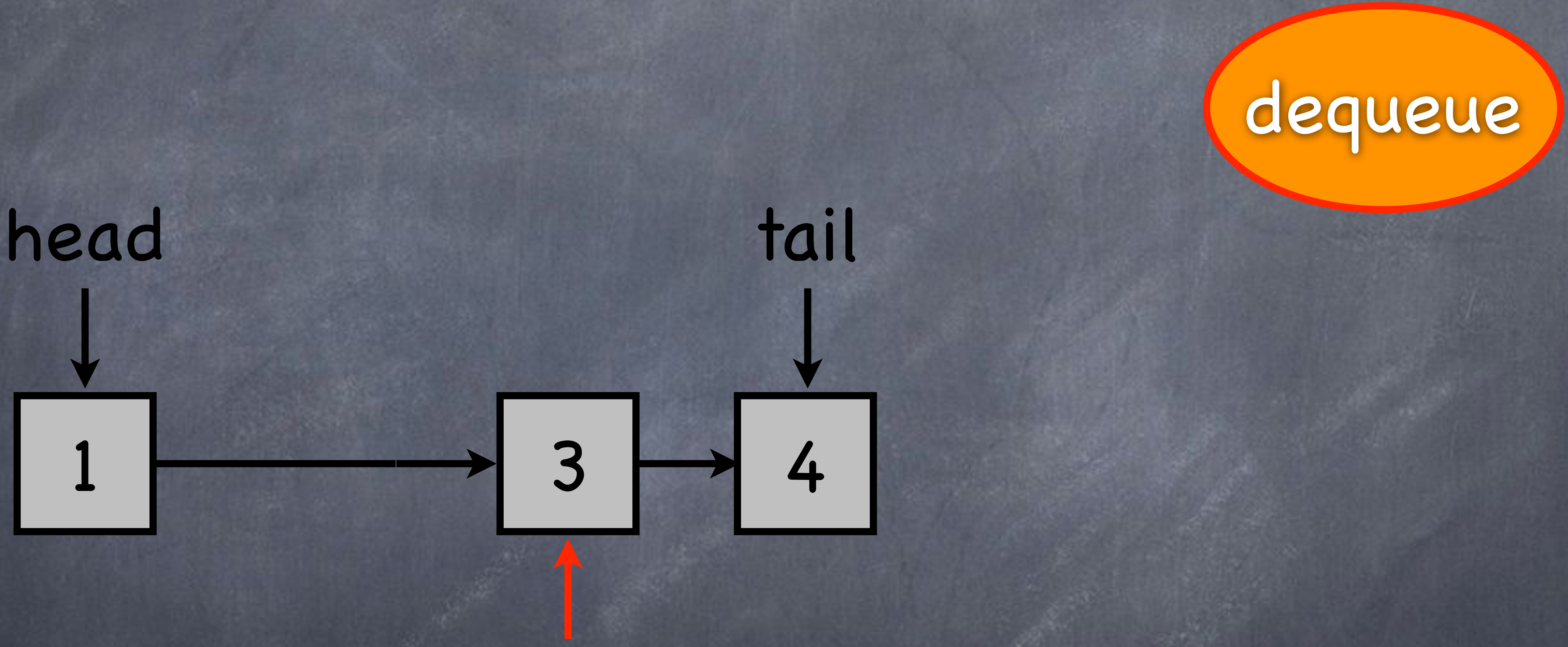


dequeue

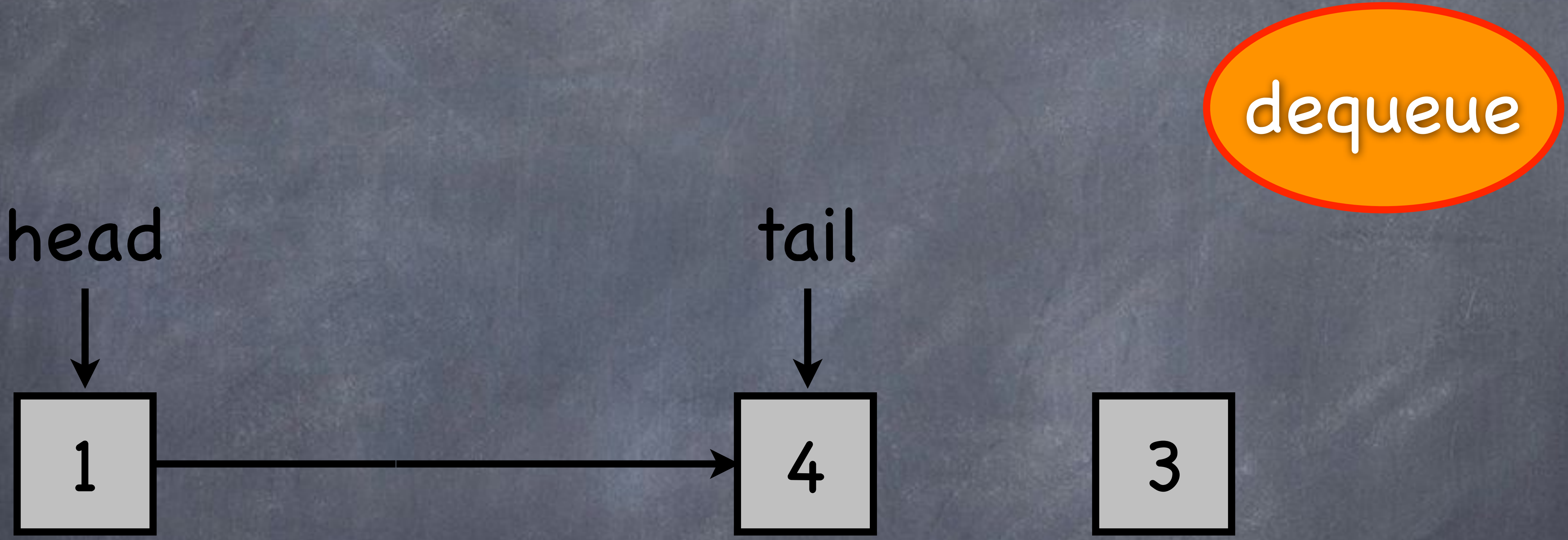
Concurrent 2-FIFO Queue (k=2)



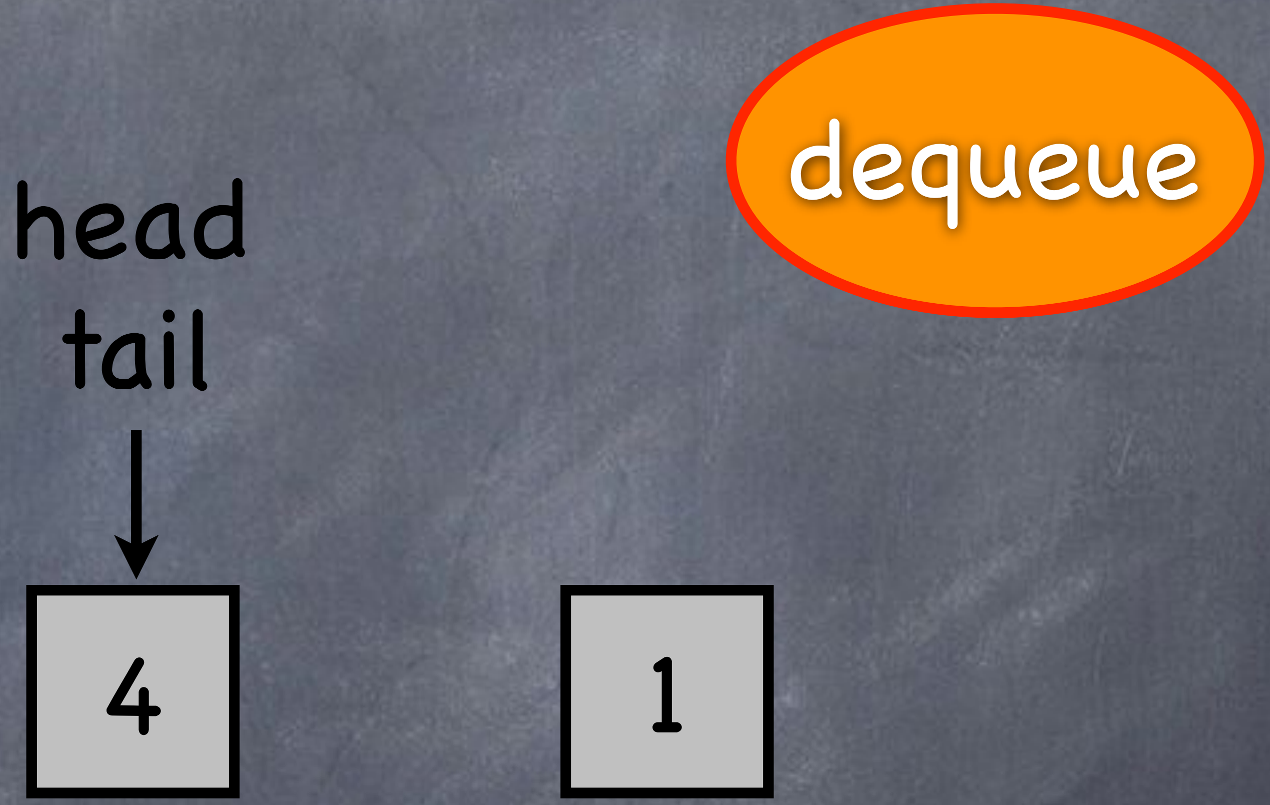
Concurrent 2-FIFO Queue (k=2)



Concurrent 2-FIFO Queue (k=2)



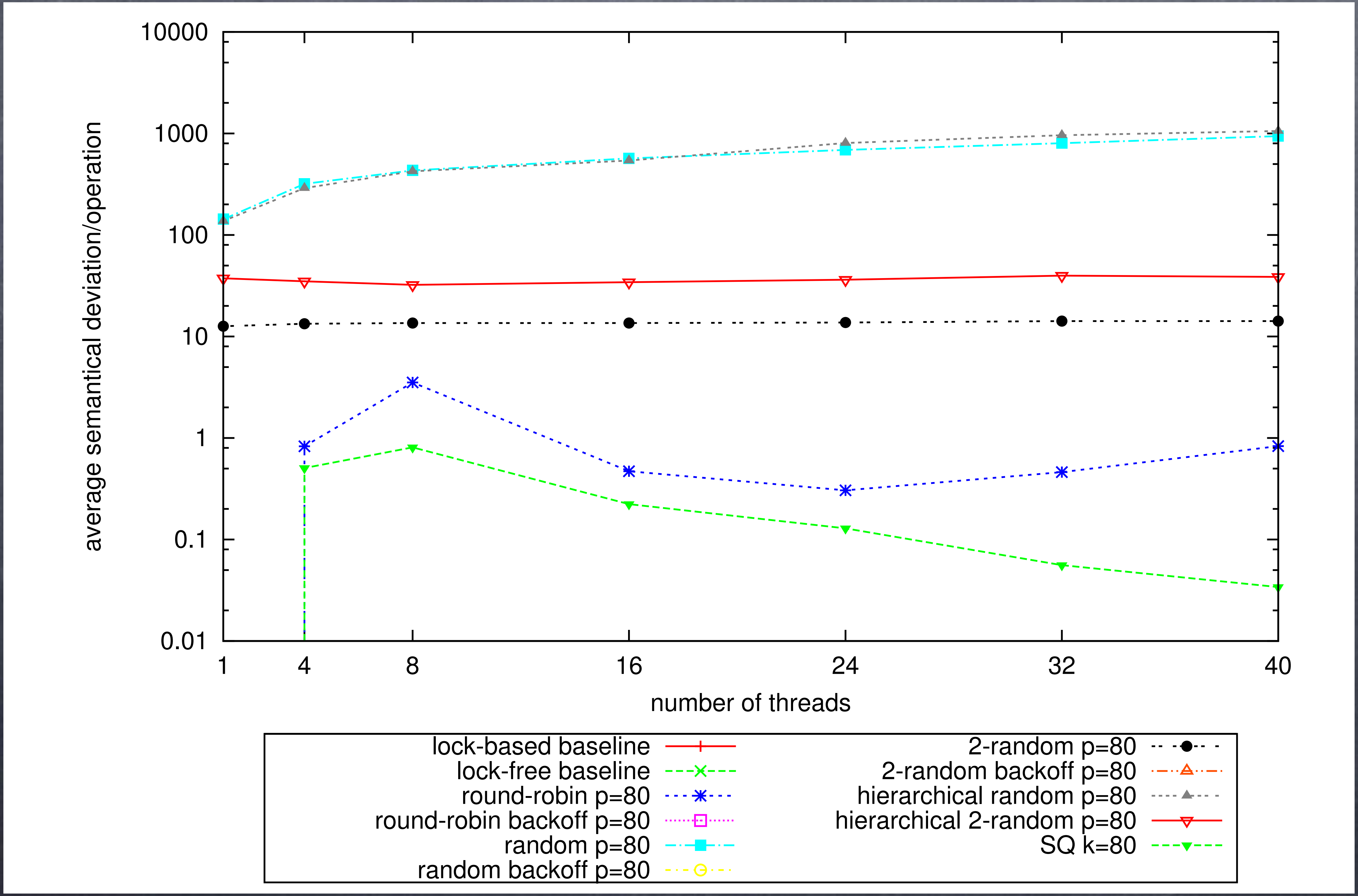
Concurrent 2-FIFO Queue (k=2)



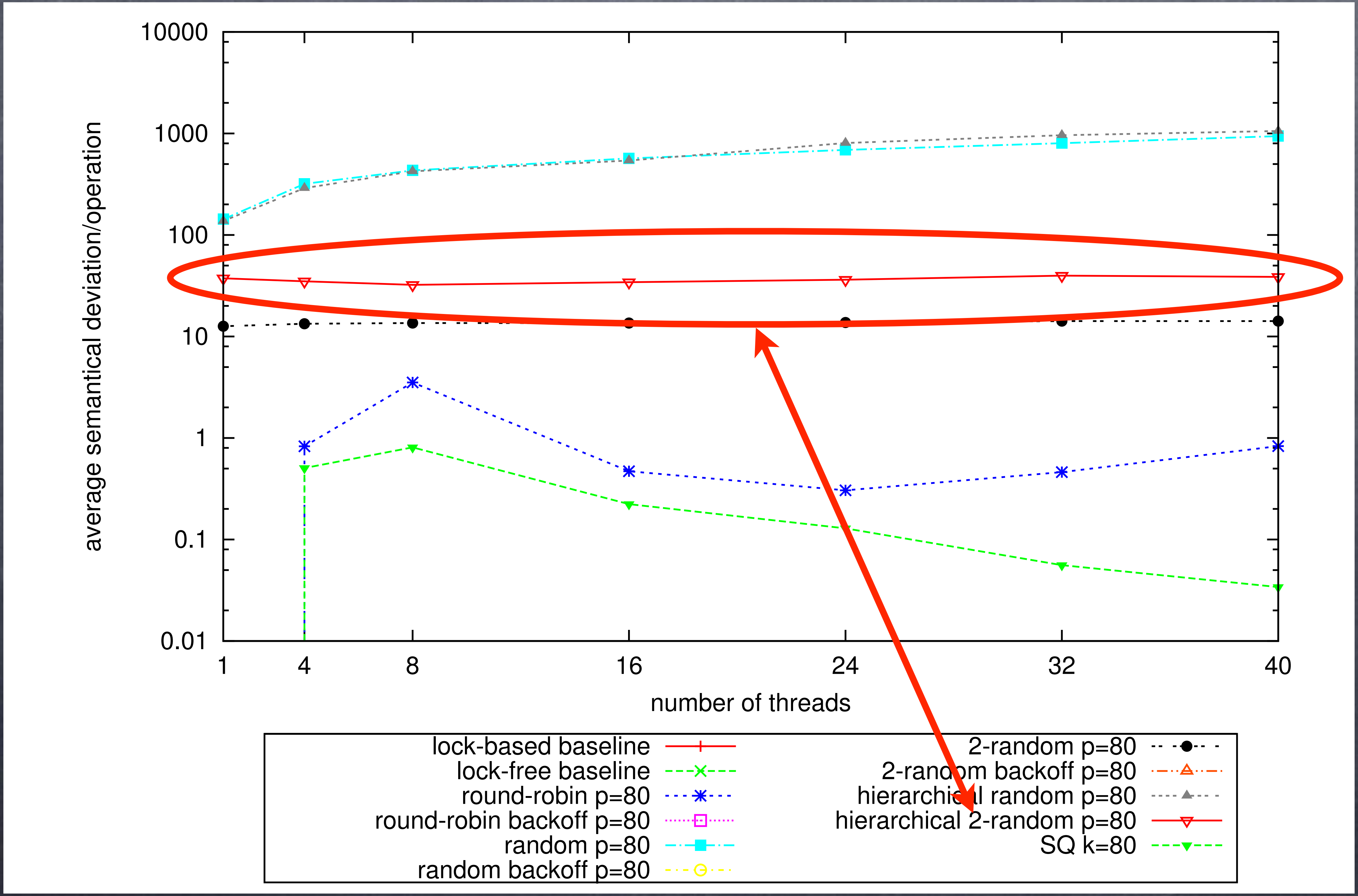
We call k
the <u>worst-case semantical deviation</u> (WCSD) of
a k-FIFO queue from
a regular FIFO queue

The actual semantical deviation (ASD) is the semantical deviation of a k-FIFO queue when applied to a given workload

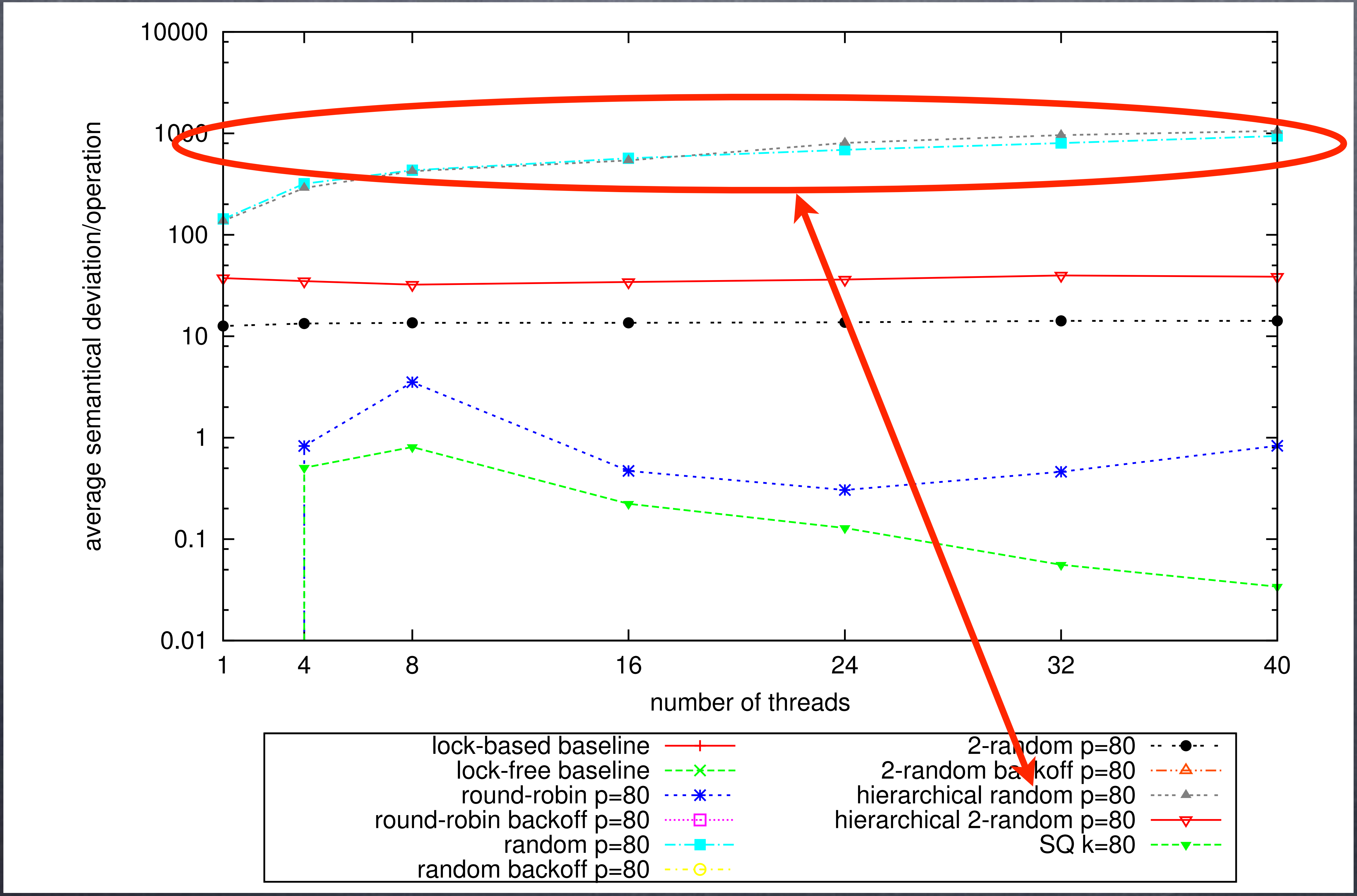
Actual Semantical Deviation



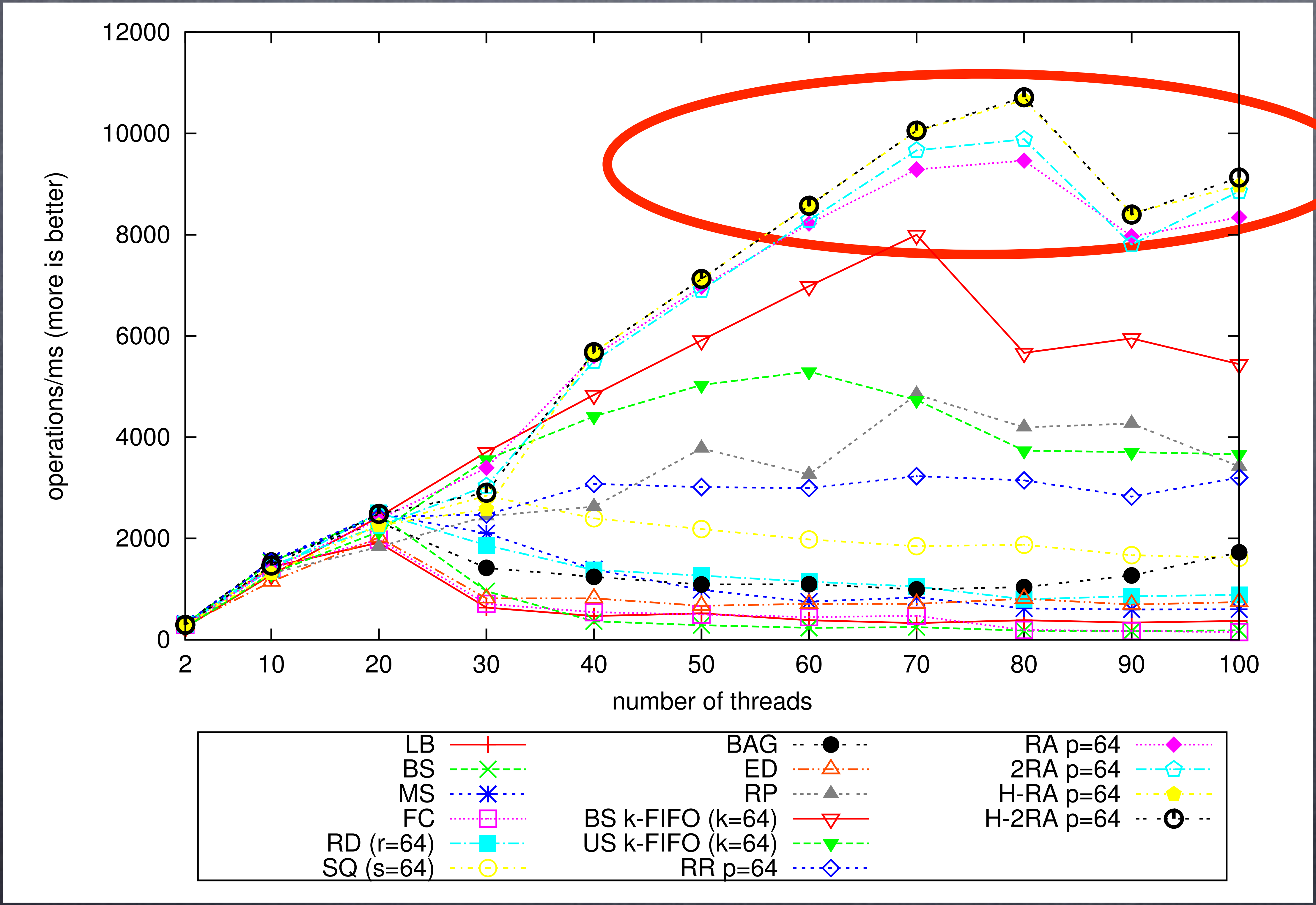
Here k may be around 40 on average: best tradeoff



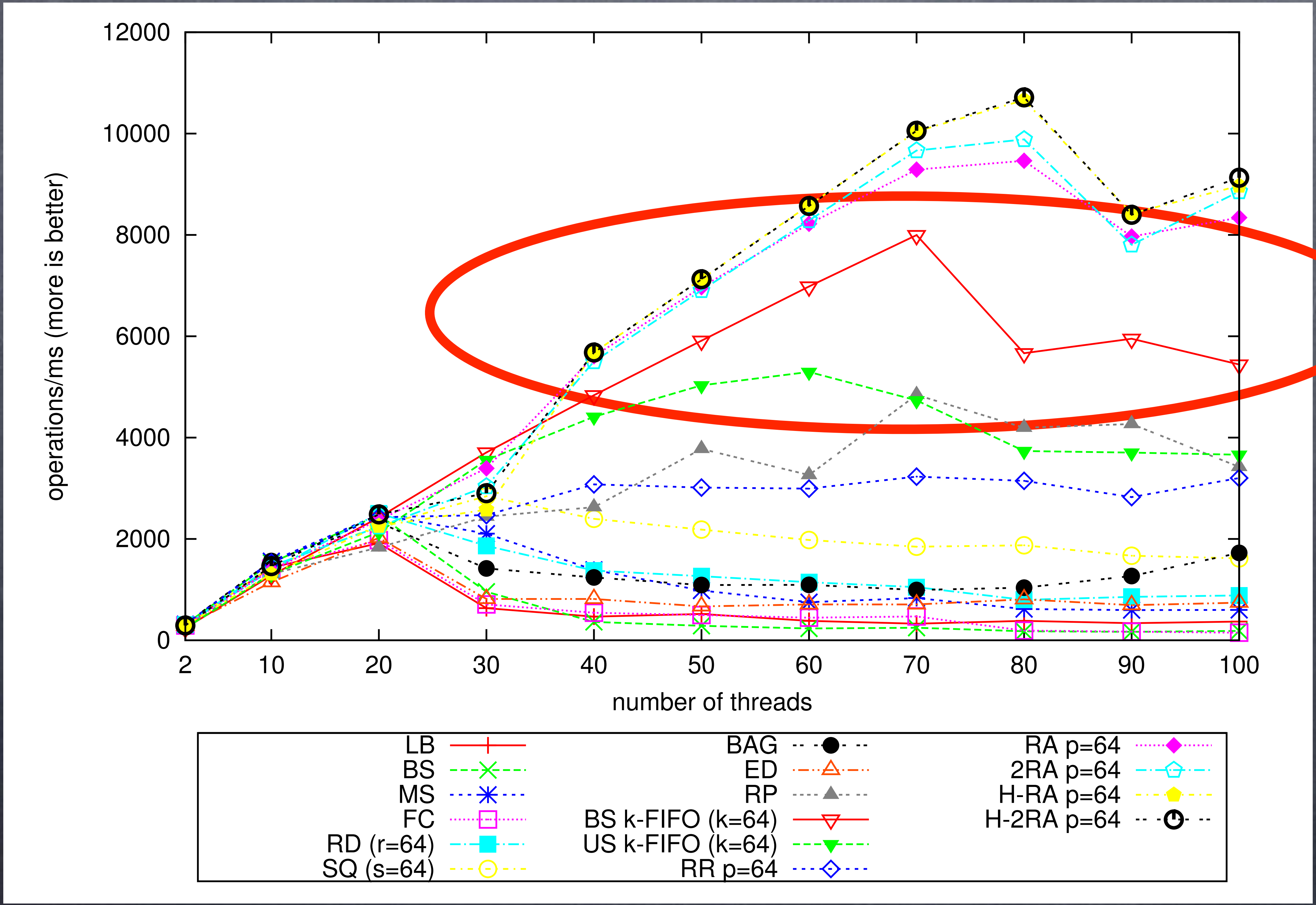
Whereas here k is one order of magnitude bigger w/o gain



Random vs. d-Random



Segmented Queues



Back to Correctness?

Questions?