

Optimizing Tail Latency in the Light-Tailed $M/G/k$

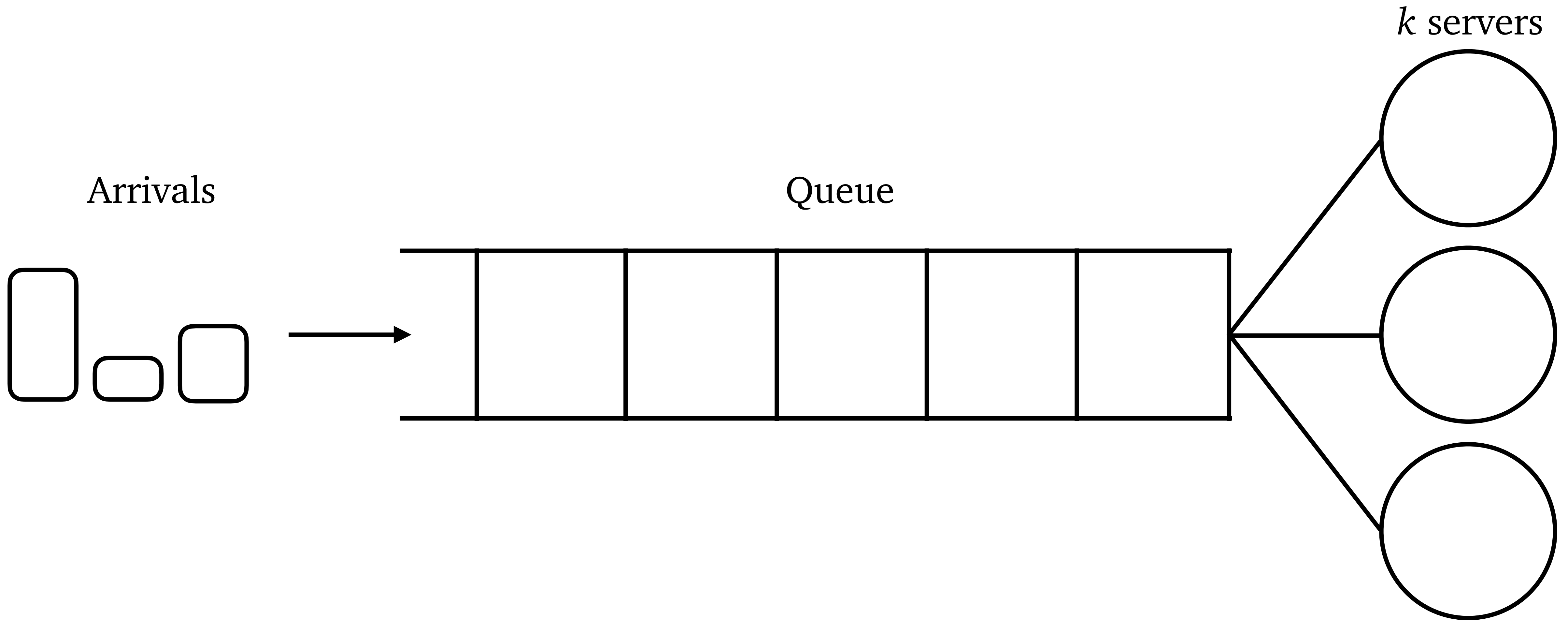
George Yu
Cornell ORIE

Amit Harlev
Cornell CAM

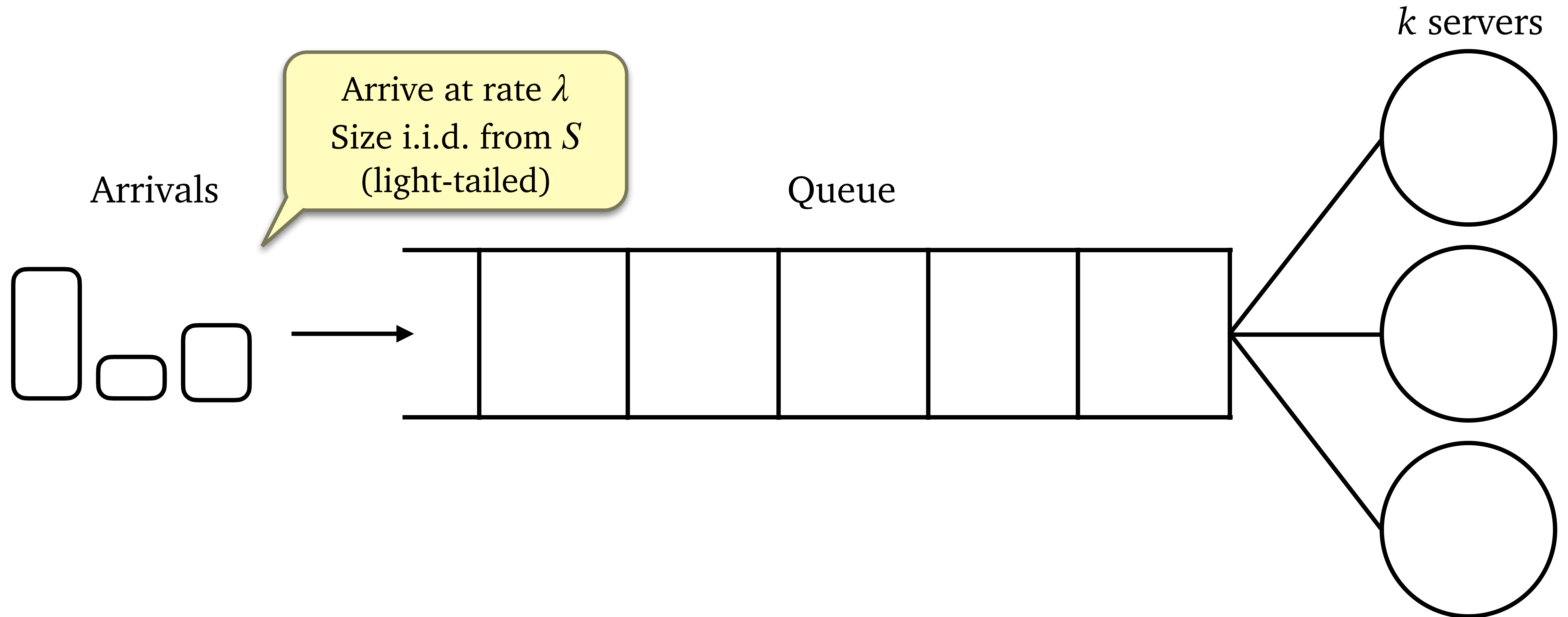
Reevu Adakroy
Cornell University

Ziv Scully
Cornell ORIE

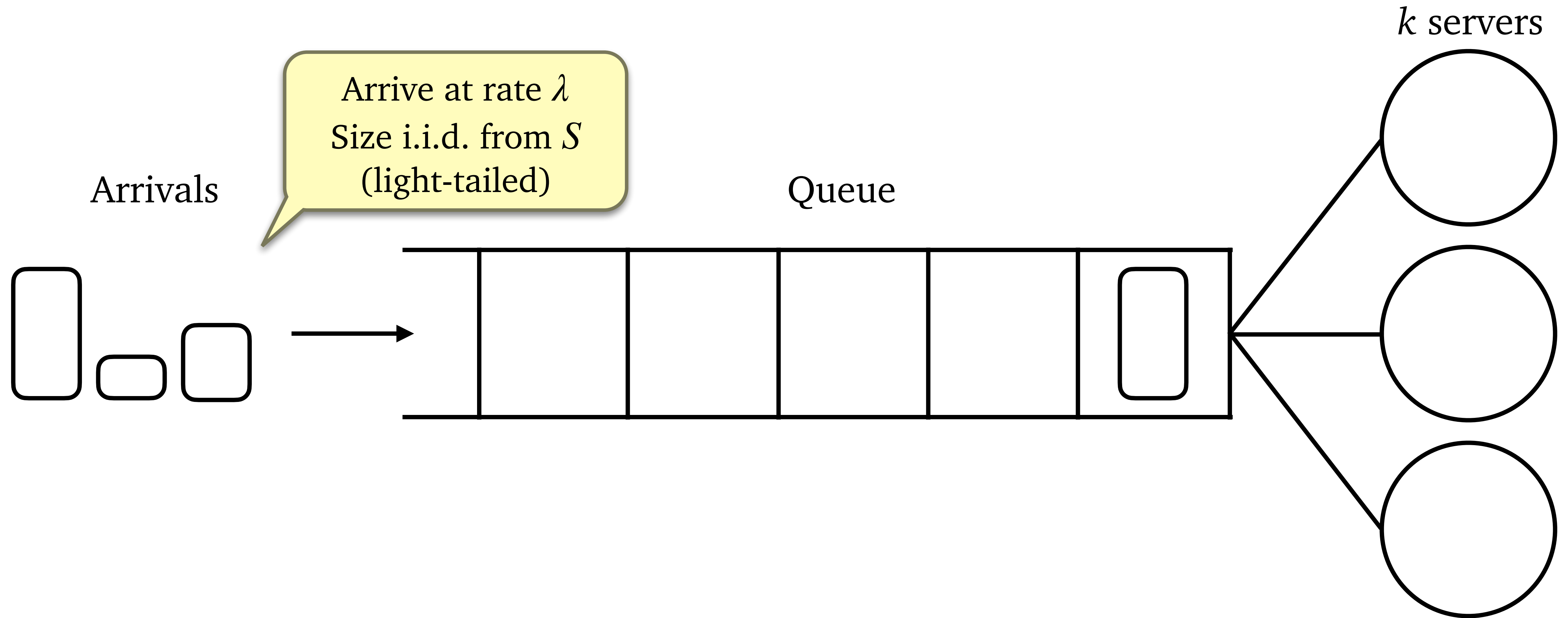
The light-tailed $M/G/k$



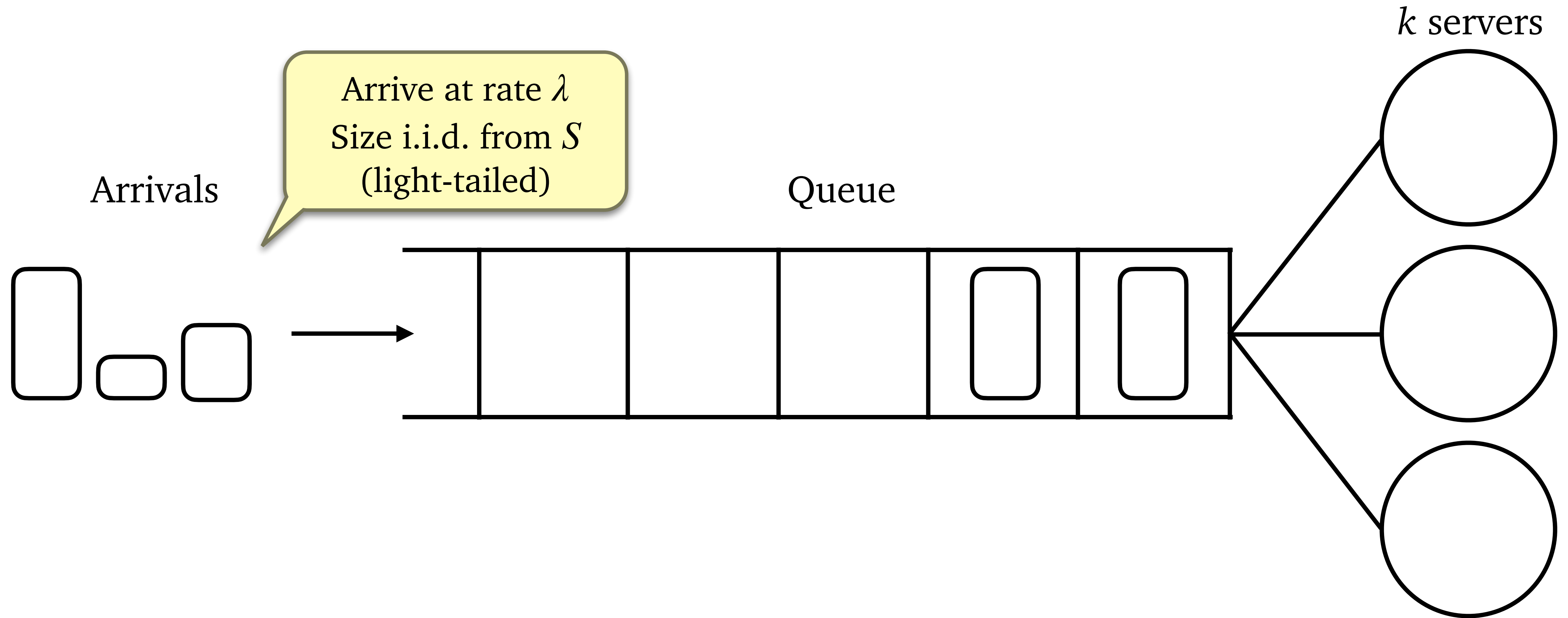
The light-tailed M/G/k



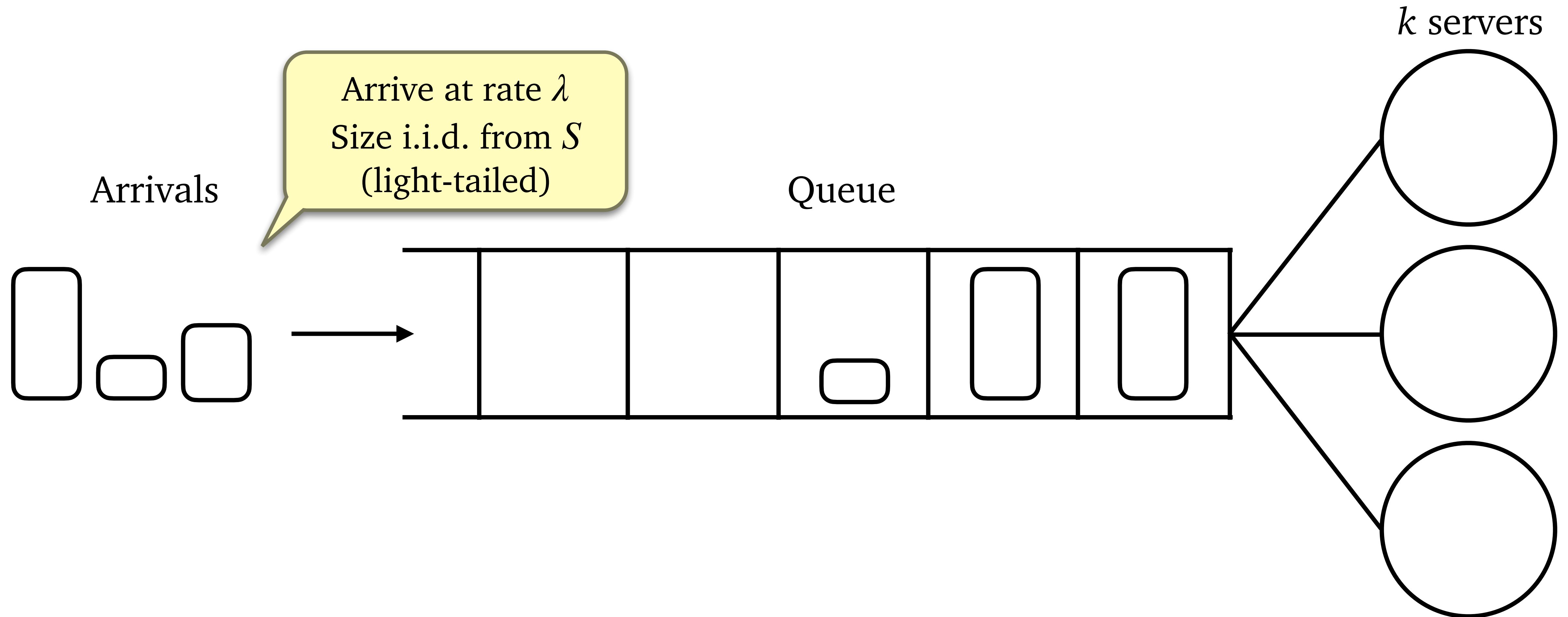
The light-tailed $M/G/k$



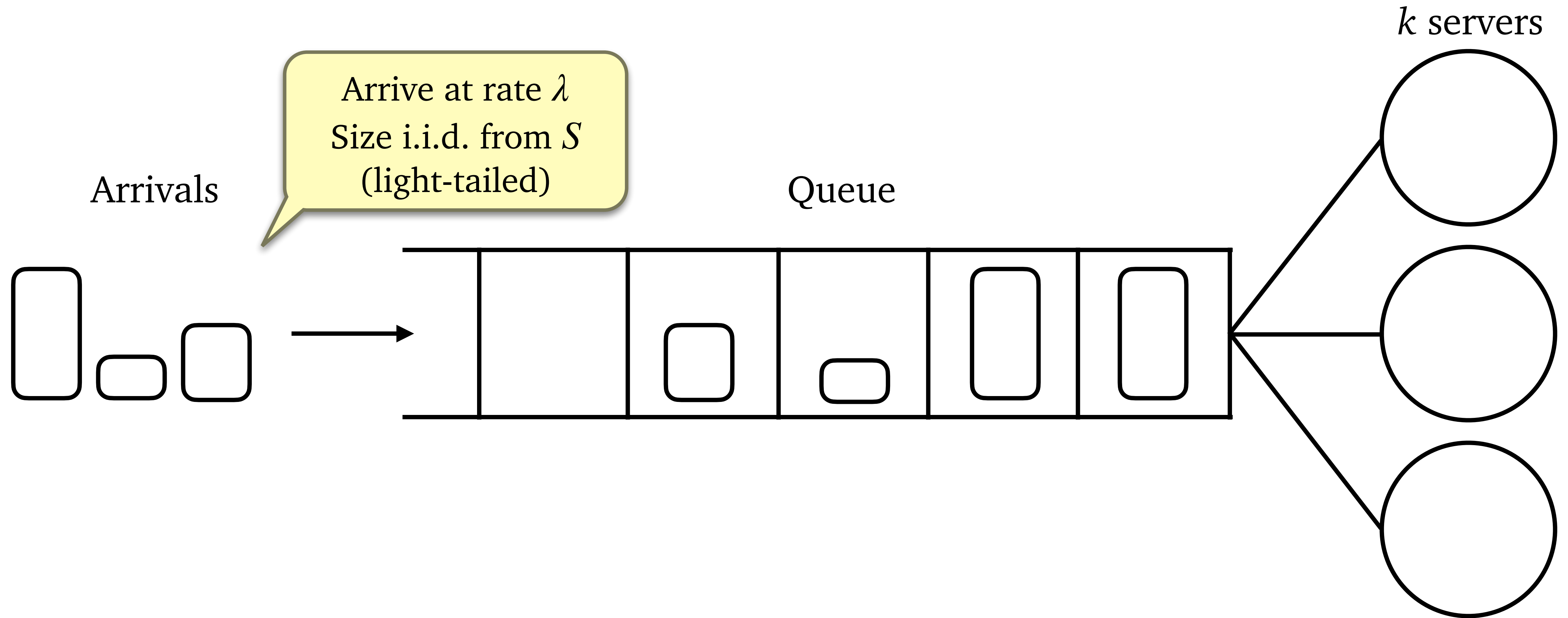
The light-tailed M/G/k



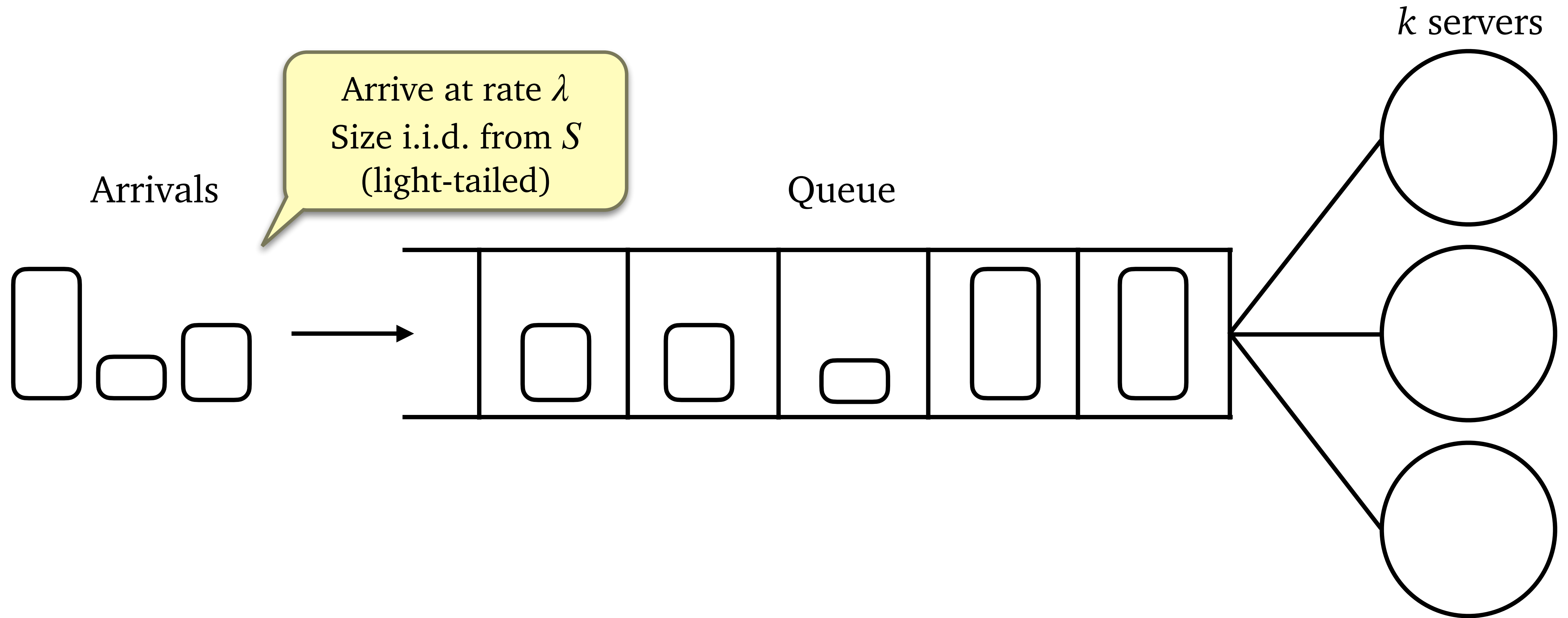
The light-tailed $M/G/k$



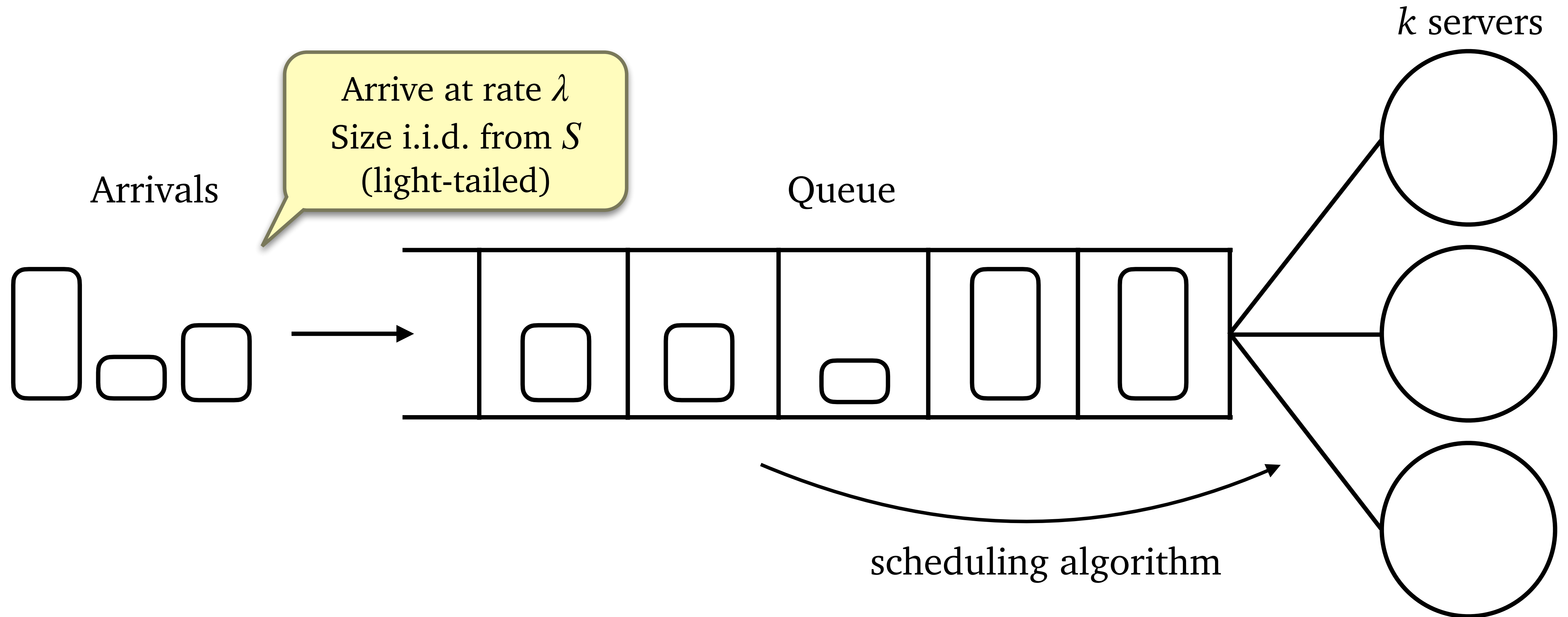
The light-tailed M/G/k



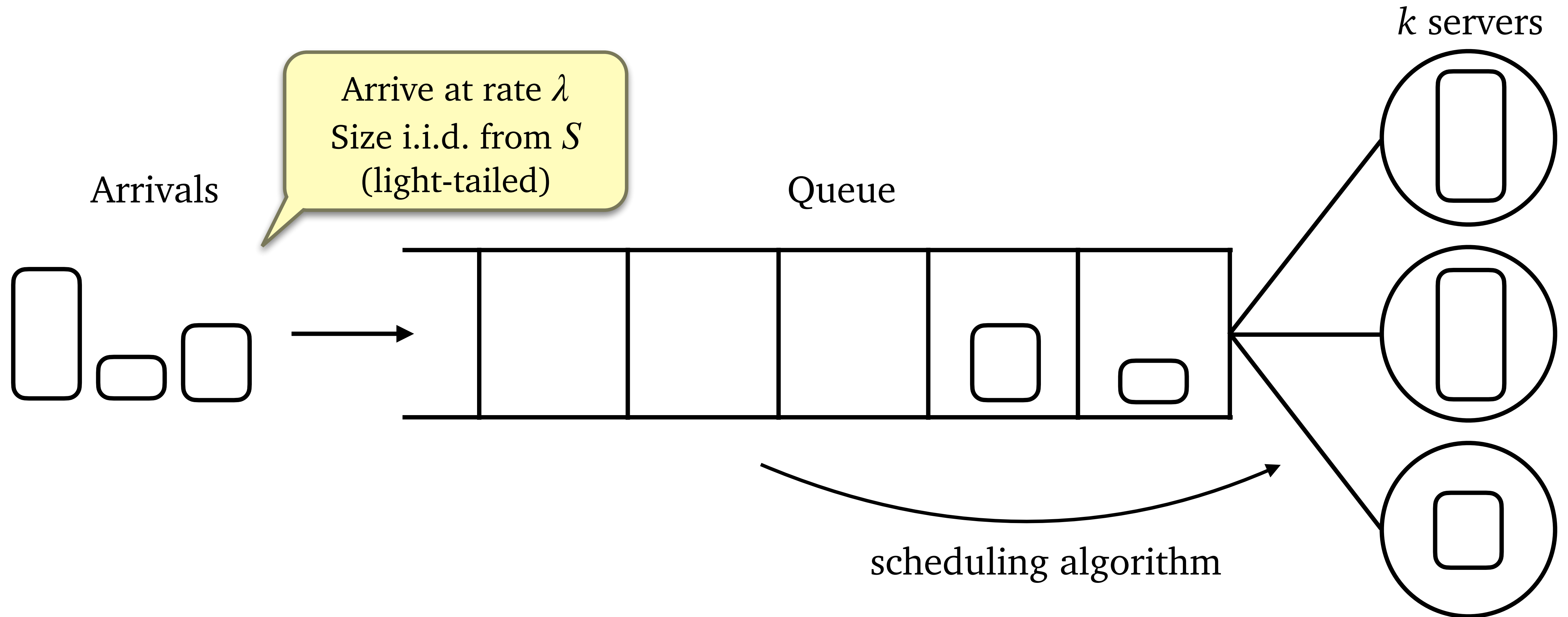
The light-tailed $M/G/k$



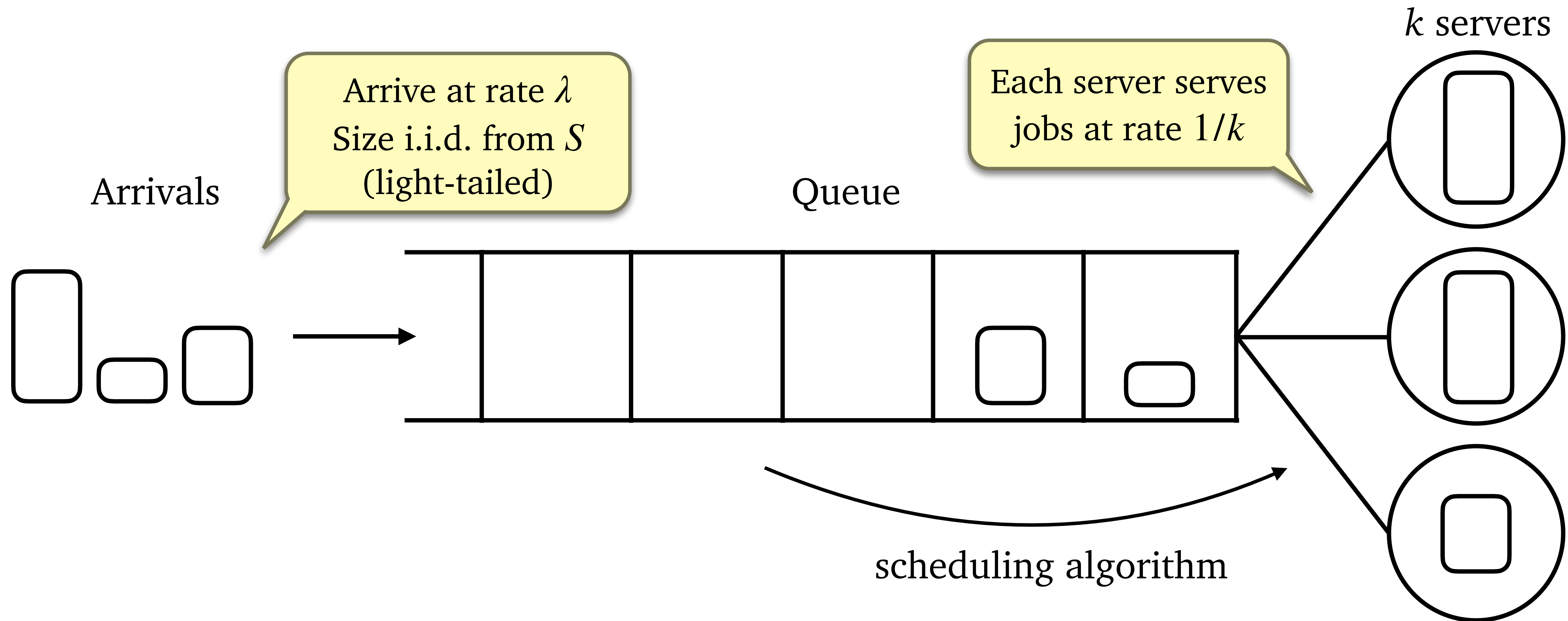
The light-tailed M/G/k



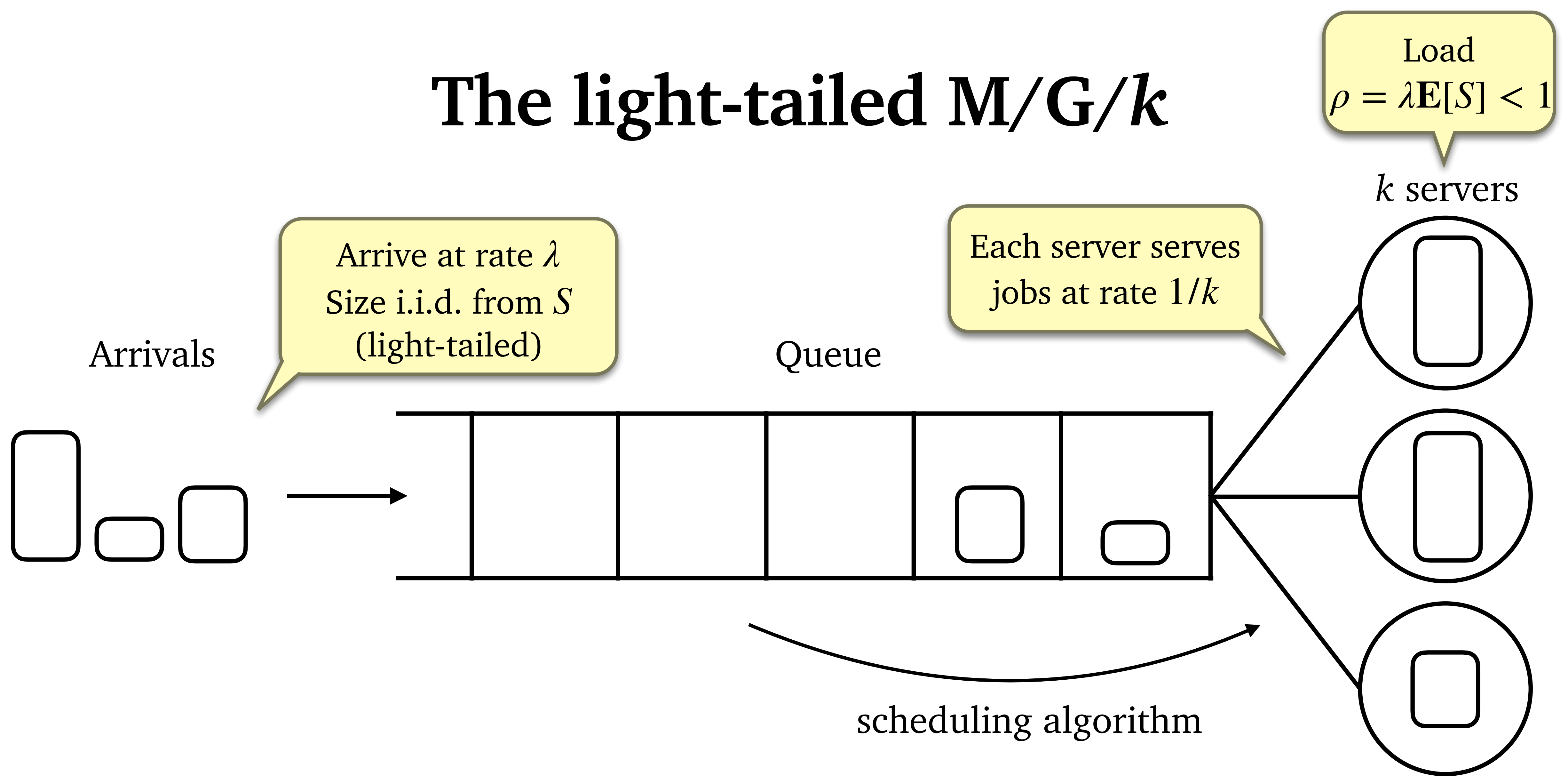
The light-tailed M/G/k



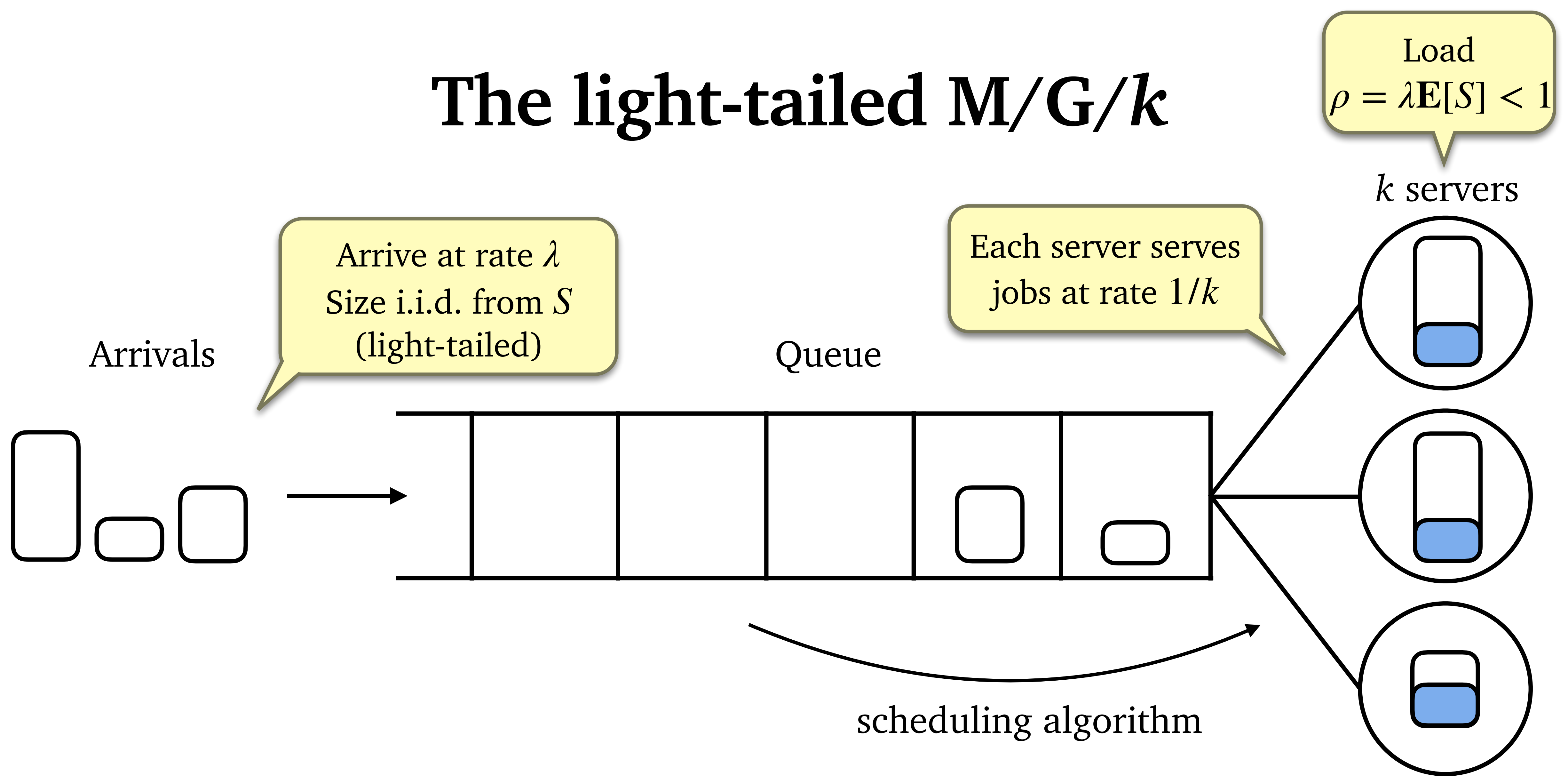
The light-tailed M/G/k



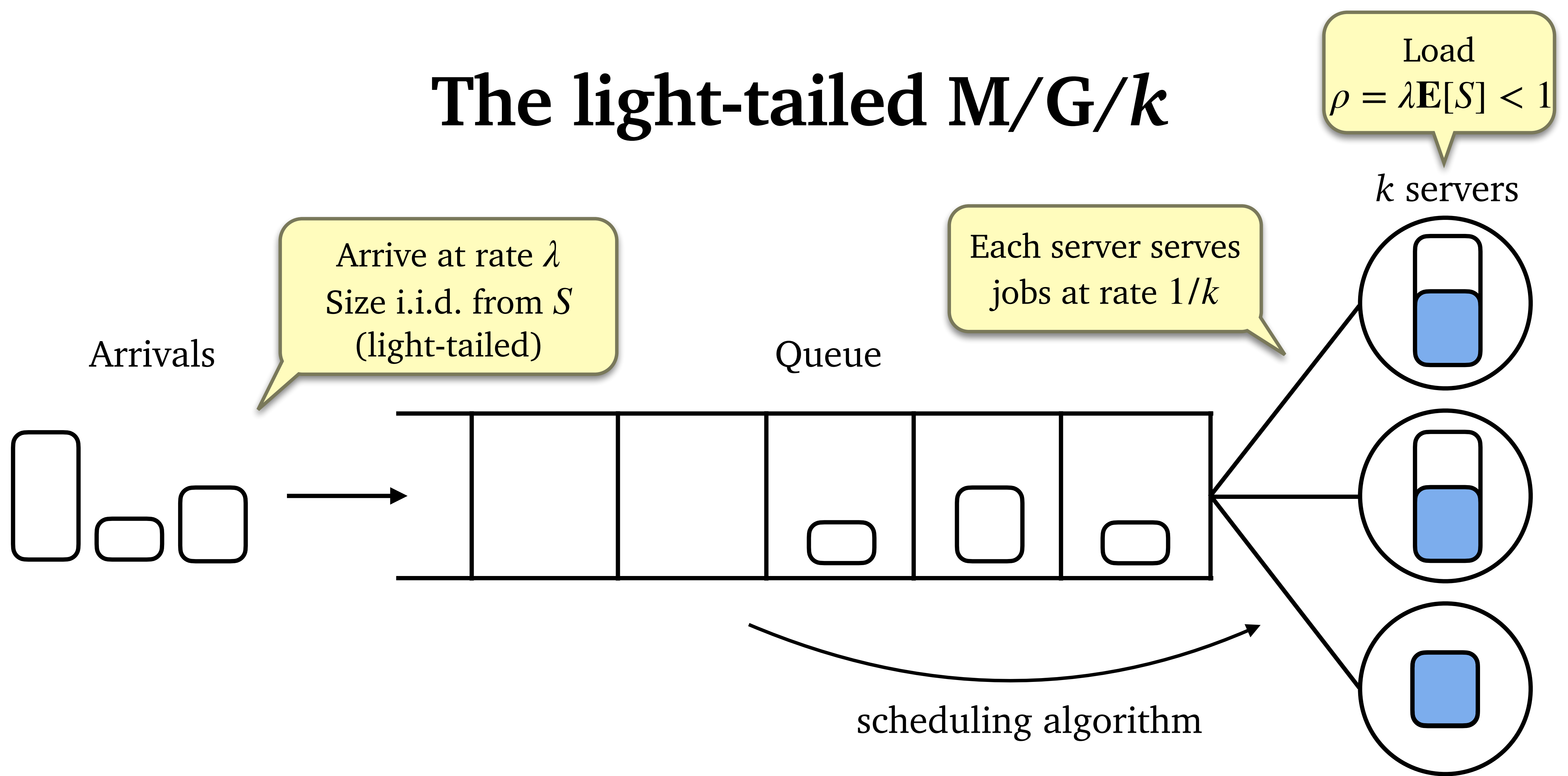
The light-tailed M/G/k



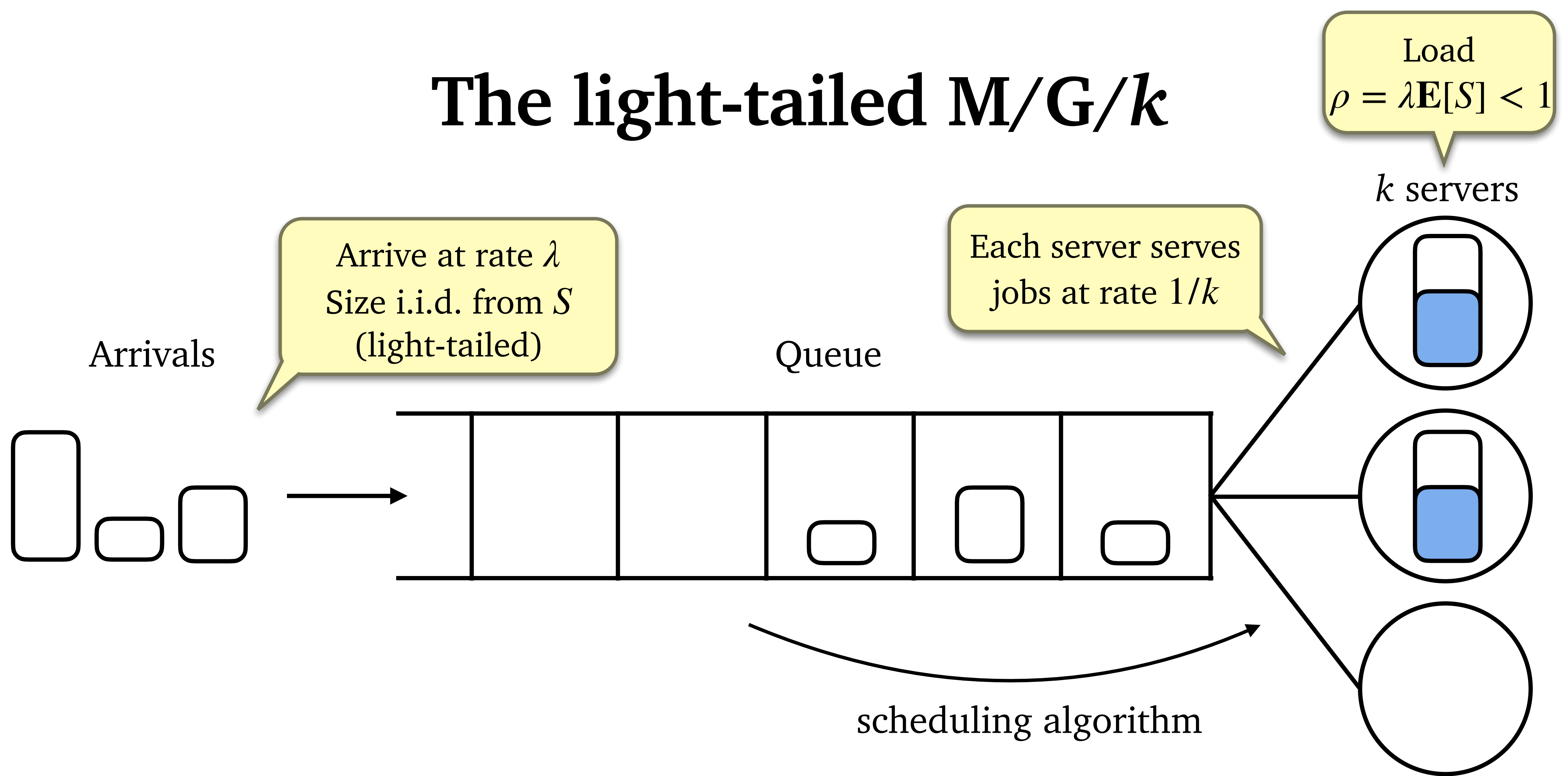
The light-tailed M/G/k



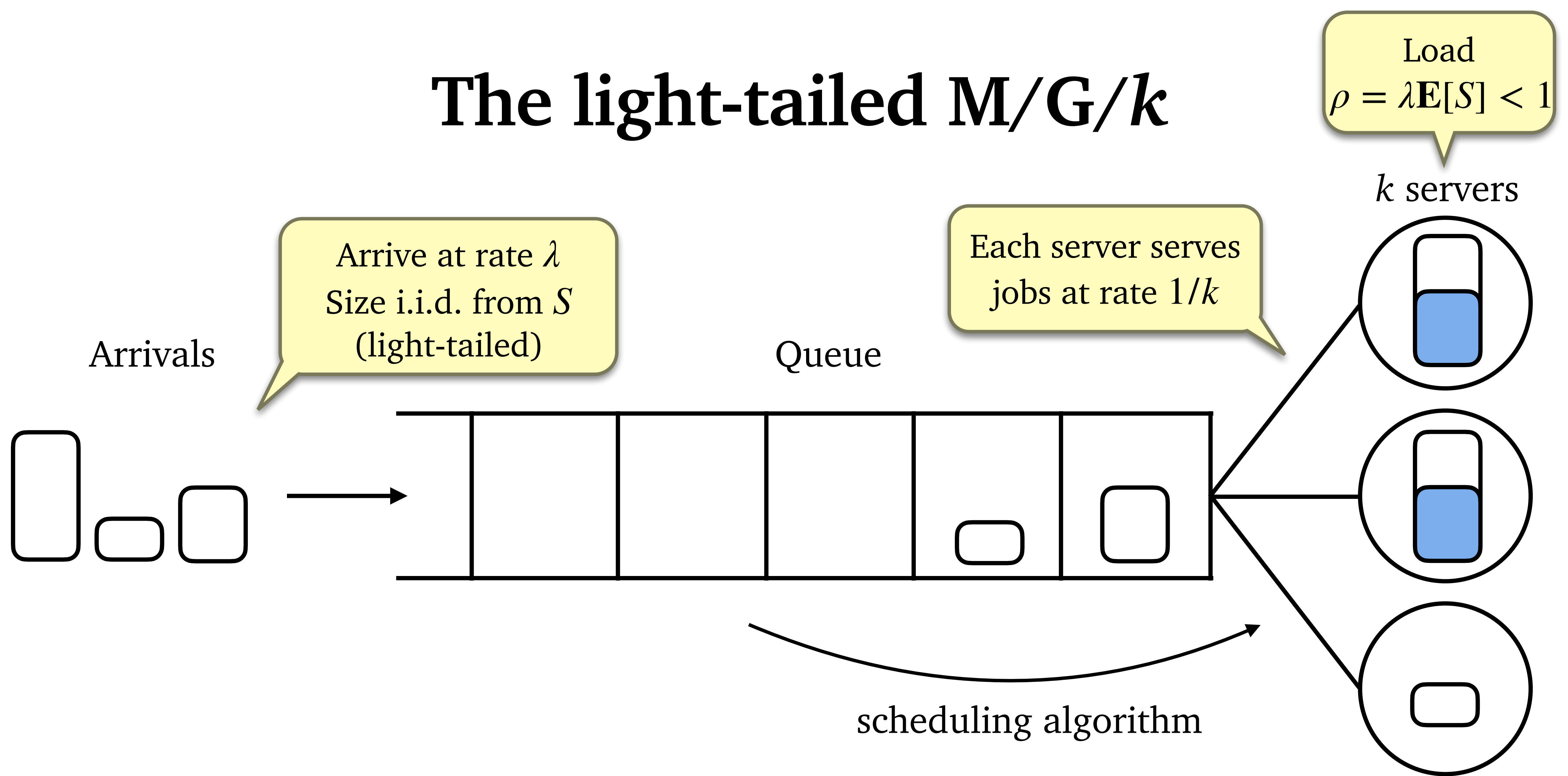
The light-tailed M/G/k



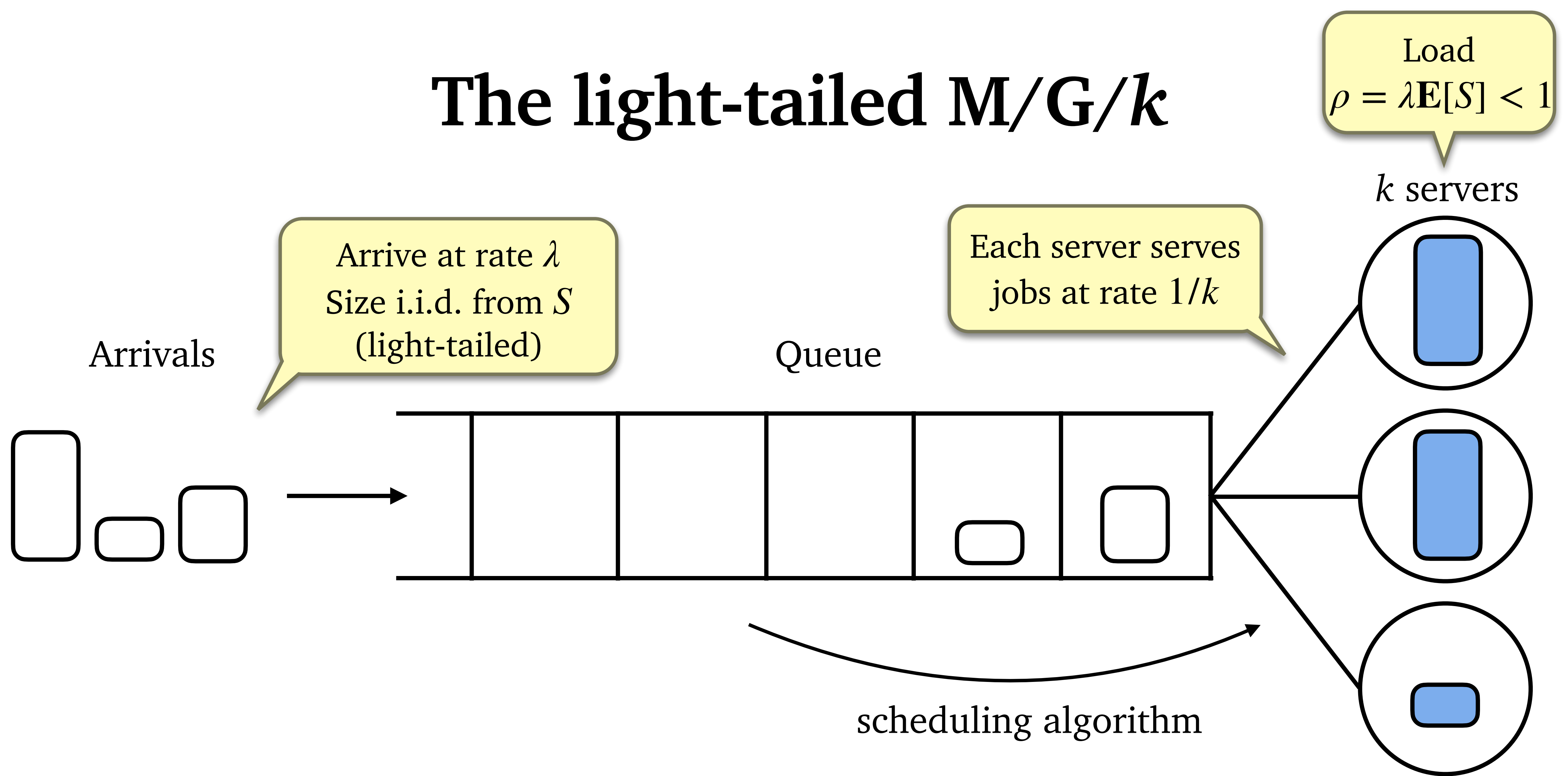
The light-tailed M/G/k



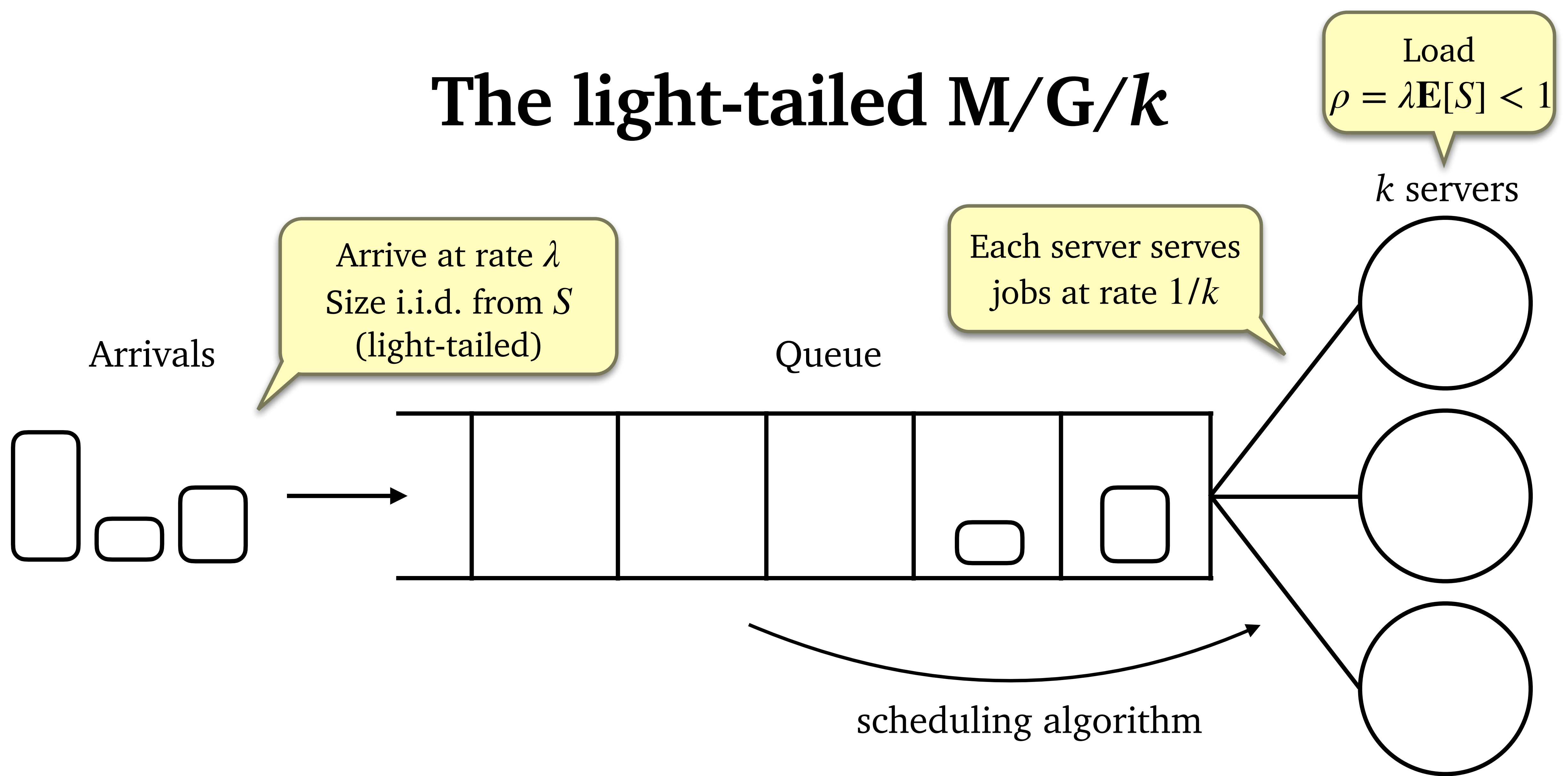
The light-tailed M/G/k



The light-tailed M/G/k



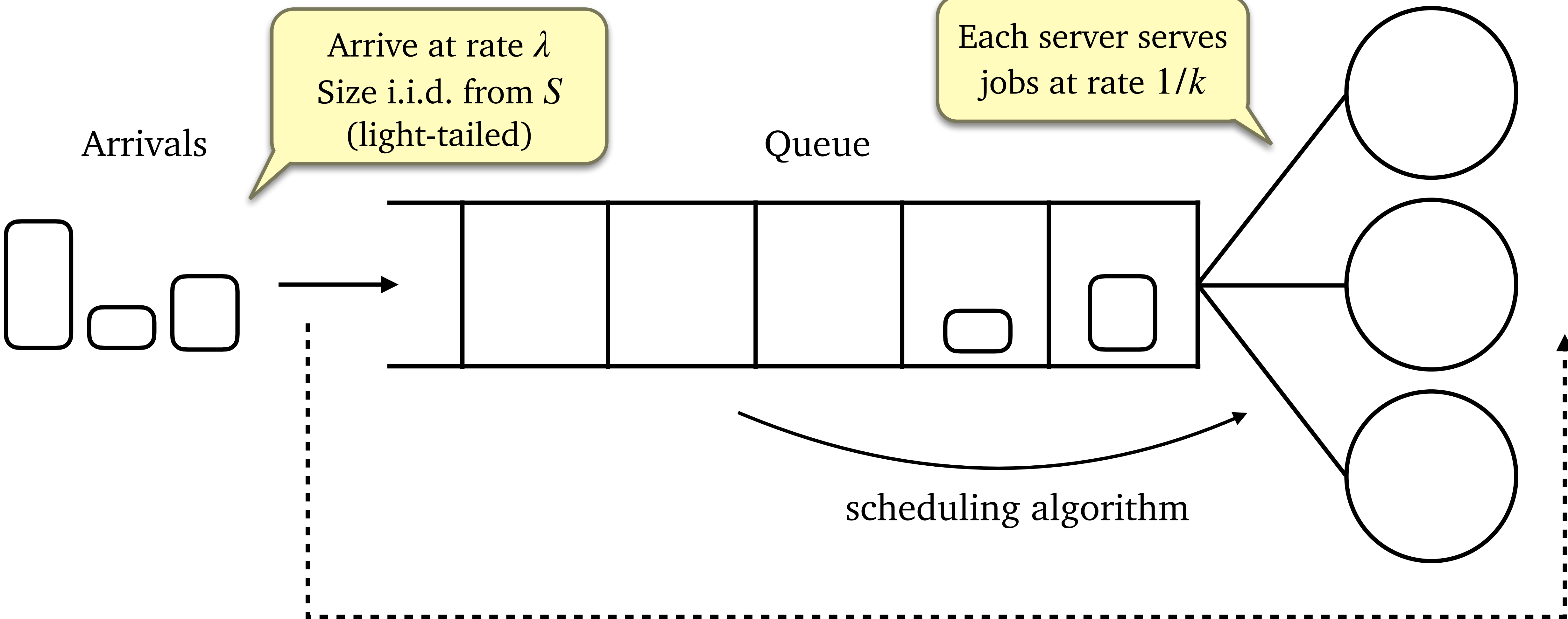
The light-tailed M/G/k



The light-tailed M/G/k

Load
 $\rho = \lambda \mathbb{E}[S] < 1$

k servers



The light-tailed M/G/k

Load
 $\rho = \lambda \mathbb{E}[S] < 1$

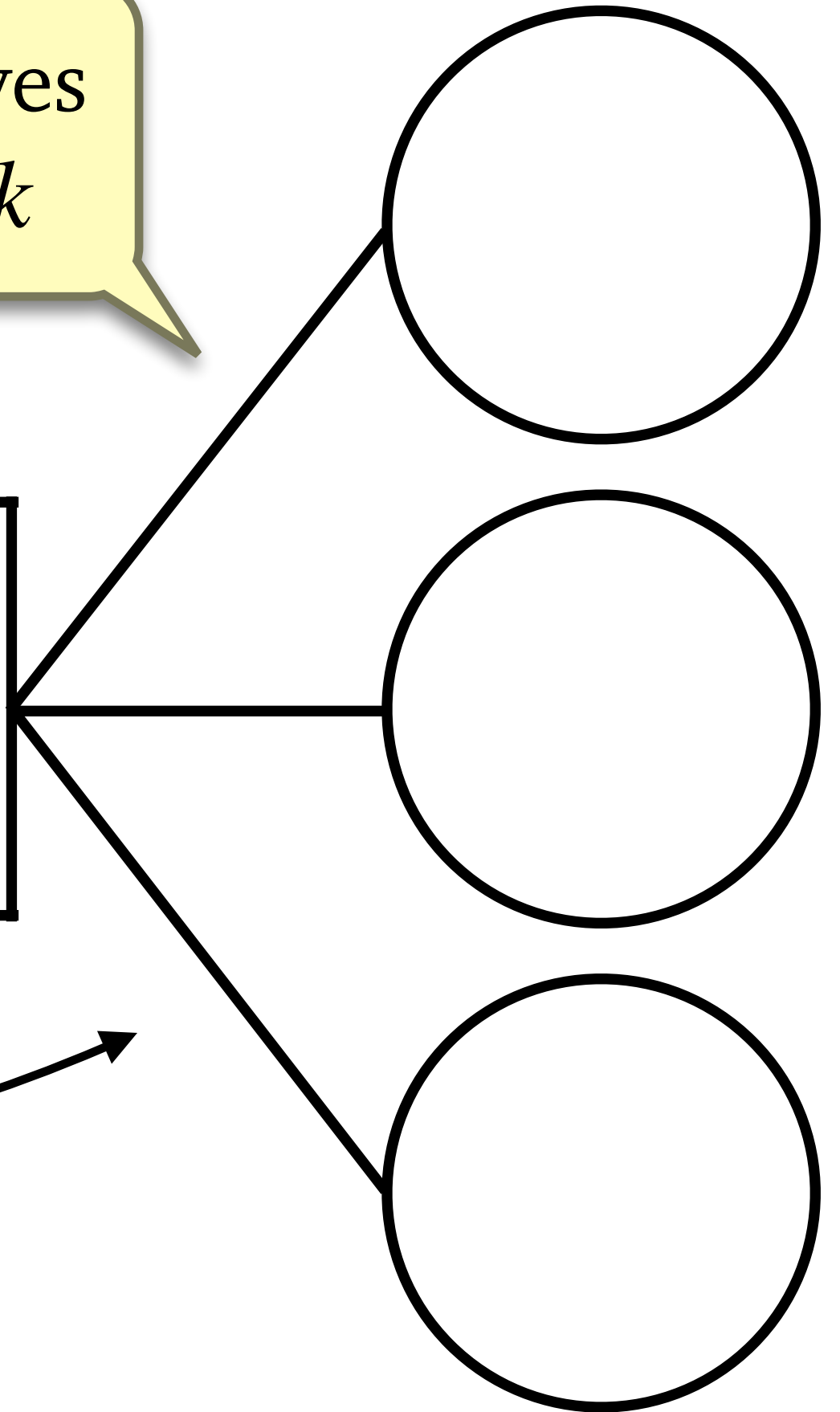
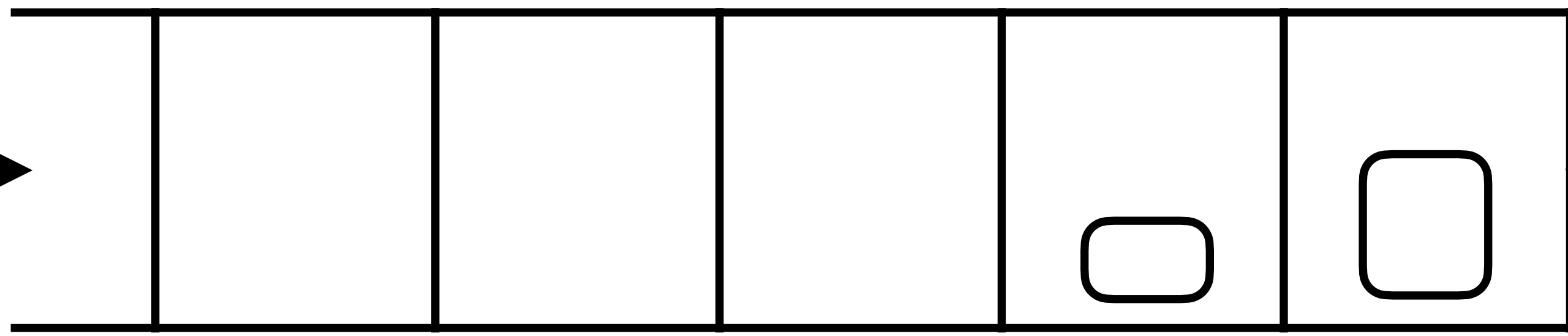
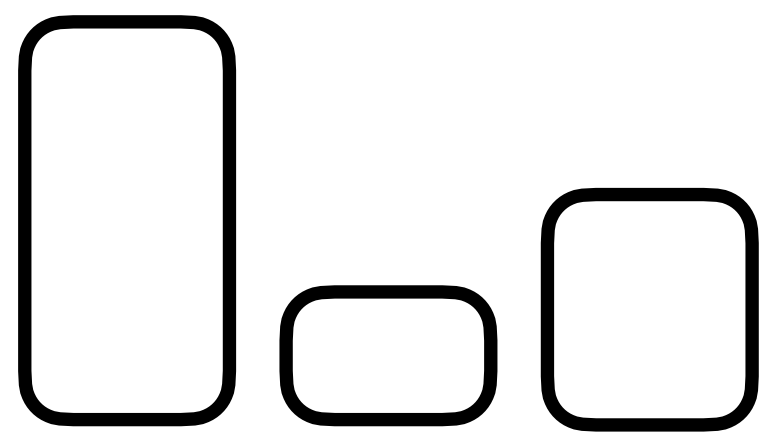
k servers

Each server serves jobs at rate $1/k$

Arrive at rate λ
Size i.i.d. from S
(light-tailed)

Arrivals

Queue



Which scheduling algorithm should we choose?

scheduling algorithm

$T =$ Response time

Which scheduling algorithm should we choose?

Setting \ Objective			

Which scheduling algorithm should we choose?

Setting \ Objective			
$\min \mathbf{E}[T]$			

Which scheduling algorithm should we choose?

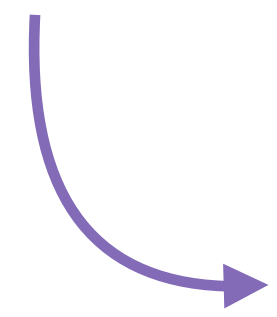
Setting Objective	single-server ($k = 1$)		
$\min \mathbf{E}[T]$			

Which scheduling algorithm should we choose?

Setting Objective	single-server ($k = 1$)		
$\min \mathbf{E}[T]$	SRPT (shortest remaining processing time)		

Which scheduling algorithm should we choose?

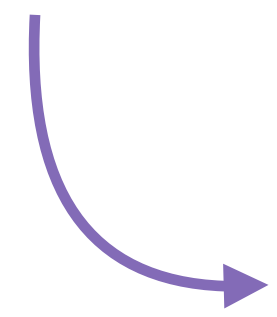
In practice



Setting \ Objective	single-server ($k = 1$)		
$\min \mathbf{E}[T]$	SRPT (shortest remaining processing time)		
$\min \mathbf{P}[T > t]$ “for all large t ”			

Which scheduling algorithm should we choose?

In practice



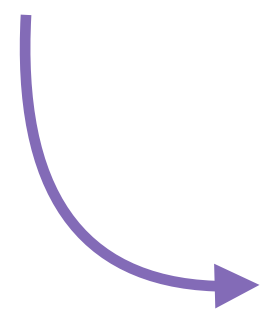
Setting \ Objective	single-server ($k = 1$)		
$\min \mathbf{E}[T]$	SRPT (shortest remaining processing time)		
$\min \mathbf{P}[T > t]$ “for all large t ”	γ -Boost		
	[Yu & Scully 2024]		

Which scheduling algorithm should we choose?

Setting \ Objective	single-server ($k = 1$)	multi-server (light traffic, $\rho \ll 1$)	
$\min \mathbf{E}[T]$	SRPT (shortest remaining processing time)		
$\min \mathbf{P}[T > t]$ “for all large t ”	γ -Boost		

[Yu & Scully 2024]

In practice

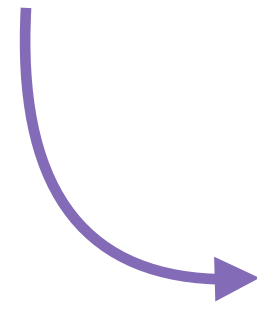


Which scheduling algorithm should we choose?

Hard!

Setting \ Objective	single-server ($k = 1$)	multi-server (light traffic, $\rho \ll 1$)	
$\min \mathbf{E}[T]$	SRPT (shortest remaining processing time)		
$\min \mathbf{P}[T > t]$ “for all large t ”	γ -Boost		

In practice



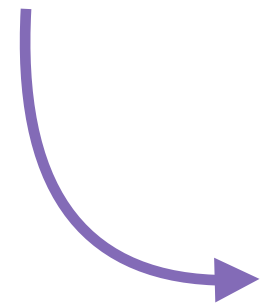
[Yu & Scully 2024]

Which scheduling algorithm should we choose?

Hard!

Setting \ Objective	single-server ($k = 1$)	multi-server (light traffic, $\rho \ll 1$)	multi-server (heavy traffic, $\rho \rightarrow 1$)
$\min \mathbf{E}[T]$	SRPT (shortest remaining processing time)		
$\min \mathbf{P}[T > t]$ “for all large t ”	γ -Boost		

In practice



[Yu & Scully 2024]

Which scheduling algorithm should we choose?

Setting / Objective	single-server ($k = 1$)	multi-server (light traffic, $\rho \ll 1$)	multi-server (heavy traffic, $\rho \rightarrow 1$)
$\min \mathbf{E}[T]$	SRPT (shortest remaining processing time)		
$\min \mathbf{P}[T > t]$ “for all large t ”	γ -Boost		

Hard! (pointing to multi-server light traffic)

Hope ideas carry over (arc from multi-server heavy traffic to multi-server light traffic)

In practice (pointing to $\min \mathbf{P}[T > t]$)

[Yu & Scully 2024] (pointing to γ -Boost)

Which scheduling algorithm should we choose?

Setting \ Objective	single-server ($k = 1$)	multi-server (light traffic, $\rho \ll 1$)	multi-server (heavy traffic, $\rho \rightarrow 1$)
$\min \mathbf{E}[T]$	SRPT (shortest remaining processing time)		SRPT
$\min \mathbf{P}[T > t]$ "for all large t "	γ -Boost		

Hard! (yellow callout pointing to multi-server light traffic)

Hope ideas carry over (purple arrow from multi-server light traffic to multi-server heavy traffic)

[Grosf et al. 2018] (grey callout pointing to SRPT in heavy traffic)

[Yu & Scully 2024] (grey callout pointing to γ -Boost)

In practice (purple arrow pointing to the $\min \mathbf{P}[T > t]$ objective)

Which scheduling algorithm should we choose?

Setting	single-server ($k = 1$)	multi-server (light traffic, $\rho \ll 1$)	multi-server (heavy traffic, $\rho \rightarrow 1$)
Objective			
$\min \mathbf{E}[T]$	SRPT (shortest remaining processing time)	SRPT*	SRPT
$\min \mathbf{P}[T > t]$ “for all large t ”	γ -Boost		

Empirically

Hard!

Hope ideas carry over

[Grosf et al. 2018]

[Yu & Scully 2024]

In practice

*: You can do a bit better, stay tuned for the next talk!

Which scheduling algorithm should we choose?

Setting	single-server ($k = 1$)	multi-server (light traffic, $\rho \ll 1$)	multi-server (heavy traffic, $\rho \rightarrow 1$)
Objective			
$\min \mathbf{E}[T]$	SRPT (shortest remaining processing time)	SRPT*	SRPT
$\min \mathbf{P}[T > t]$ “for all large t ”	γ -Boost	?	?

Hard! (pointing to multi-server light traffic)

Hope ideas carry over (arc from multi-server light traffic to multi-server heavy traffic)

Empirically (pointing to SRPT in single-server)

[Grosf et al. 2018] (pointing to SRPT in multi-server heavy traffic)

In practice (pointing to γ -Boost)

[Yu & Scully 2024] (pointing to γ -Boost)

*: You can do a bit better, stay tuned for the next talk!

Which scheduling algorithm should we choose?

Setting	single-server ($k = 1$)	multi-server (light traffic, $\rho \ll 1$)	multi-server (heavy traffic, $\rho \rightarrow 1$)
Objective			
$\min \mathbf{E}[T]$	SRPT (shortest remaining processing time)	SRPT*	SRPT
$\min \mathbf{P}[T > t]$ “for all large t ”	γ -Boost	?	? γ -Boost

Hard! (pointing to multi-server light traffic)

Hope ideas carry over (arc from multi-server light to heavy traffic)

[Grosf et al. 2018] (pointing to SRPT in heavy traffic)

[Yu & Scully 2024] (pointing to γ -Boost in single-server)

In practice (pointing to $\min \mathbf{P}[T > t]$)

*: You can do a bit better, stay tuned for the next talk!

Which scheduling algorithm should we choose?

Setting	single-server ($k = 1$)	multi-server (light traffic, $\rho \ll 1$)	multi-server (heavy traffic, $\rho \rightarrow 1$)
Objective			
$\min \mathbf{E}[T]$	SRPT (shortest remaining processing time)	SRPT*	SRPT
$\min \mathbf{P}[T > t]$ “for all large t ”	γ -Boost	? γ -Boost	? γ -Boost

Hard!

Hope ideas carry over

Empirically

[Grosf et al. 2018]

In practice

[Yu & Scully 2024]

*: You can do a bit better, stay tuned for the next talk!

This talk

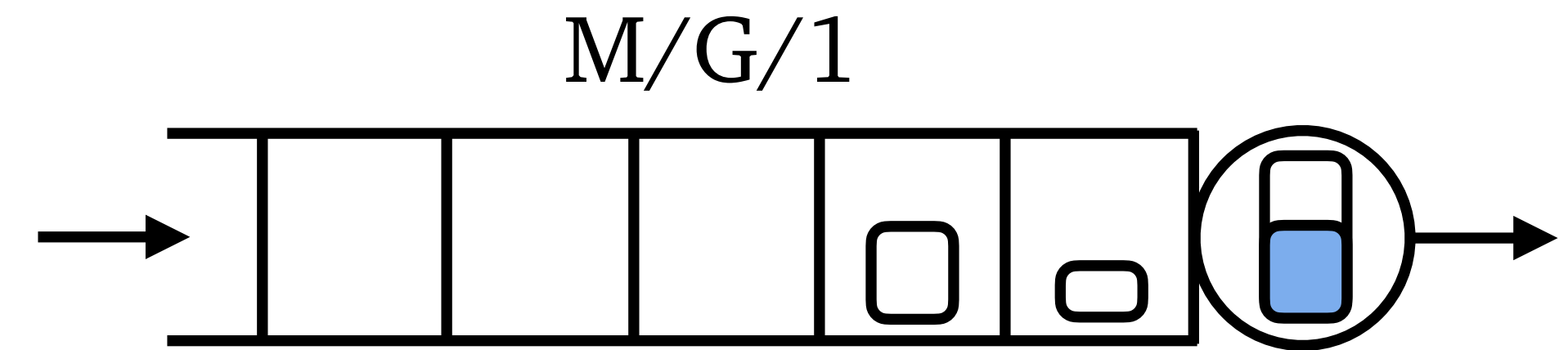
This talk



What do we mean by $\min \mathbf{P}[T > t]$ “for all large t ”?



What is the γ -Boost policy?



This talk



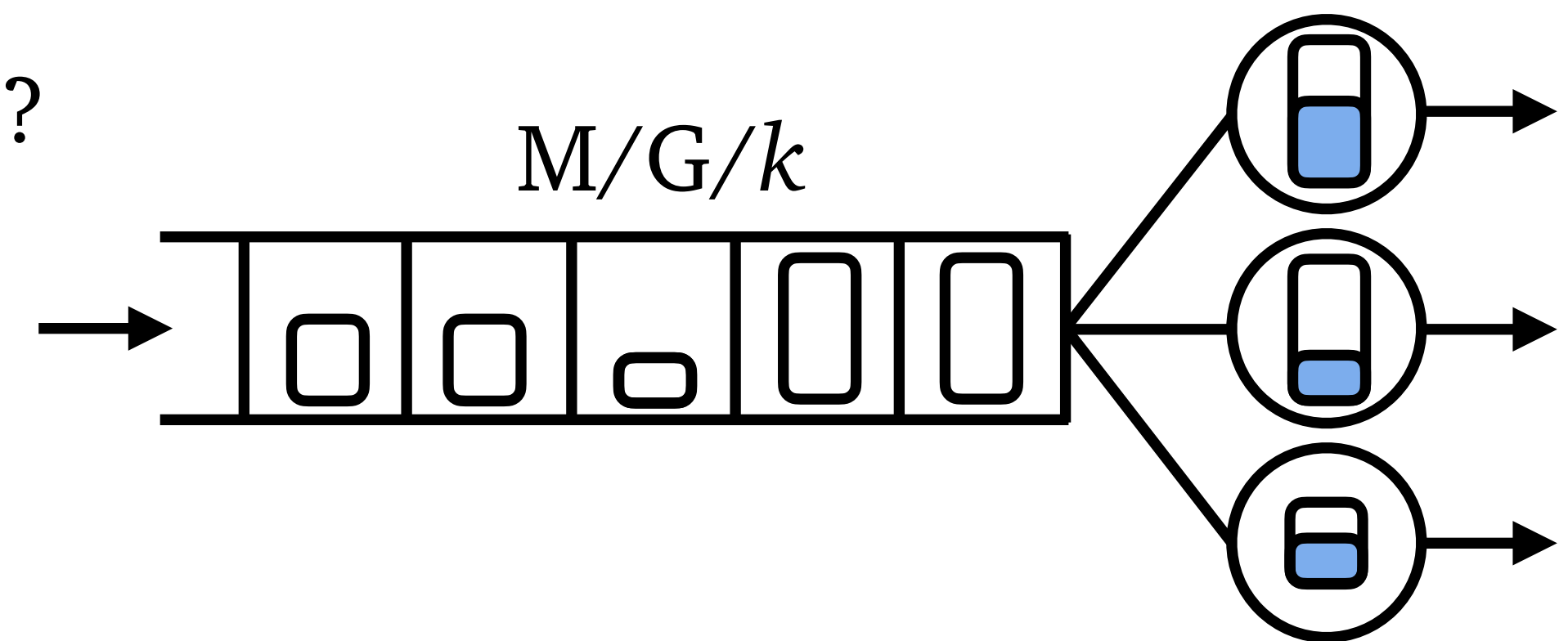
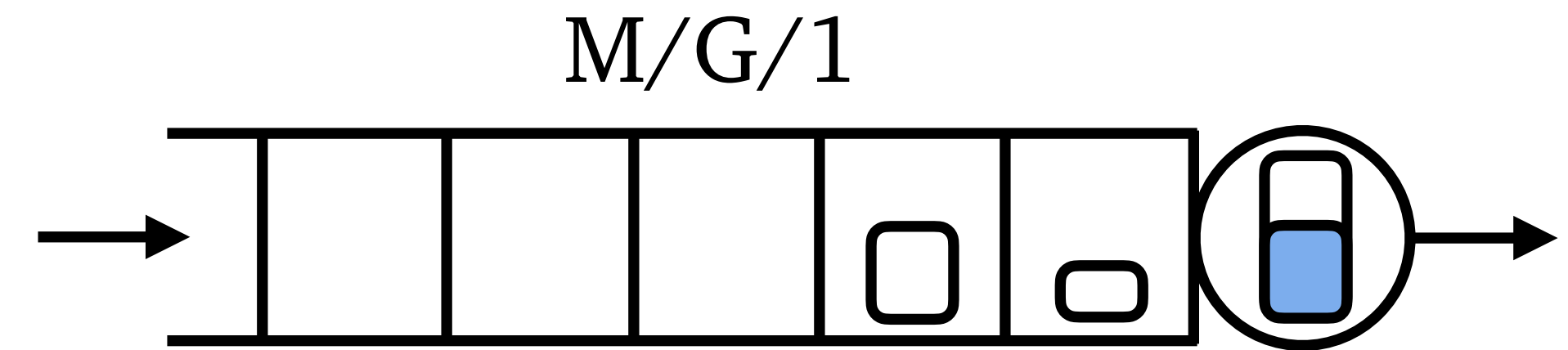
What do we mean by $\min \mathbf{P}[T > t]$ “for all large t ”?



What is the γ -Boost policy?



Is γ -Boost tail optimal in the $M/G/k$ in heavy traffic?



This talk



What do we mean by $\min \mathbf{P}[T > t]$ “for all large t ”?



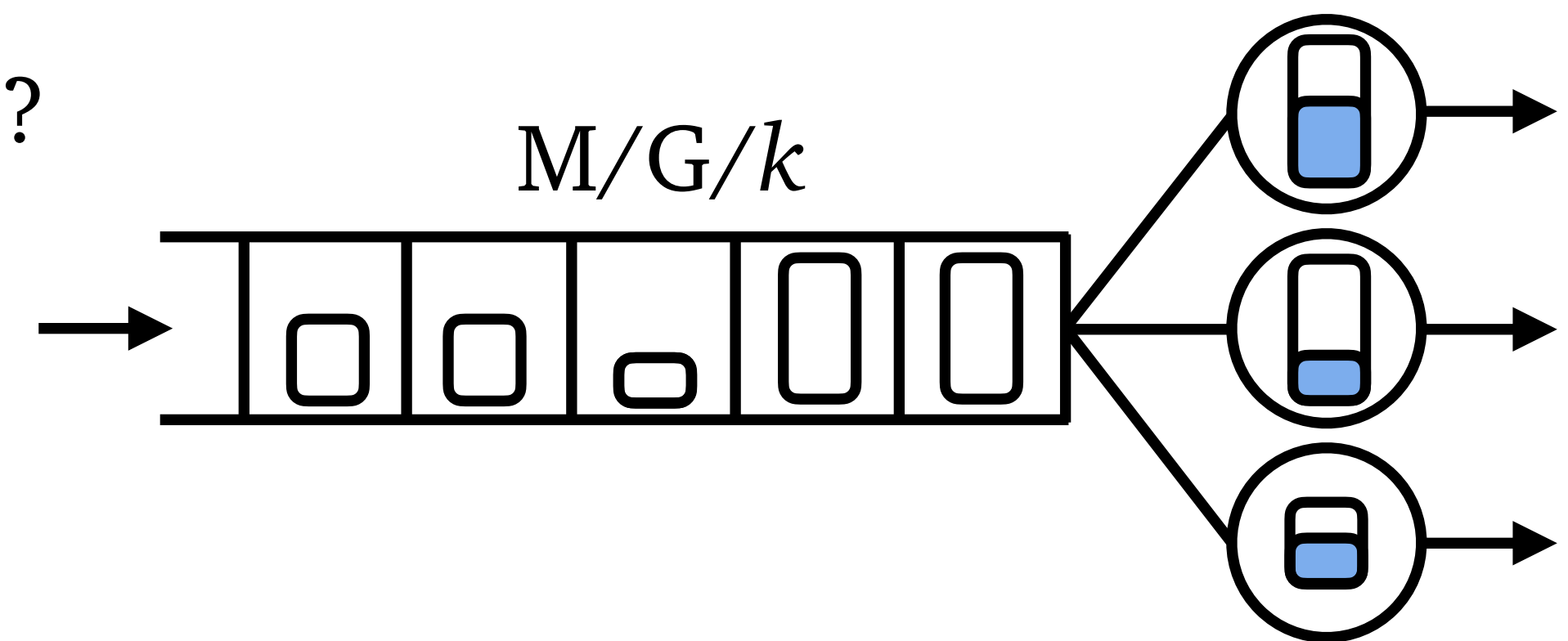
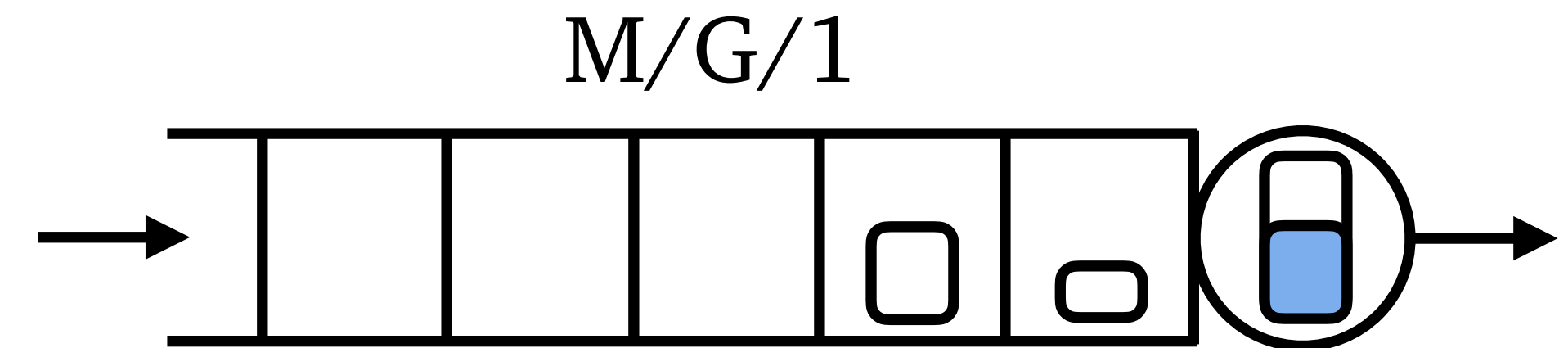
What is the γ -Boost policy?



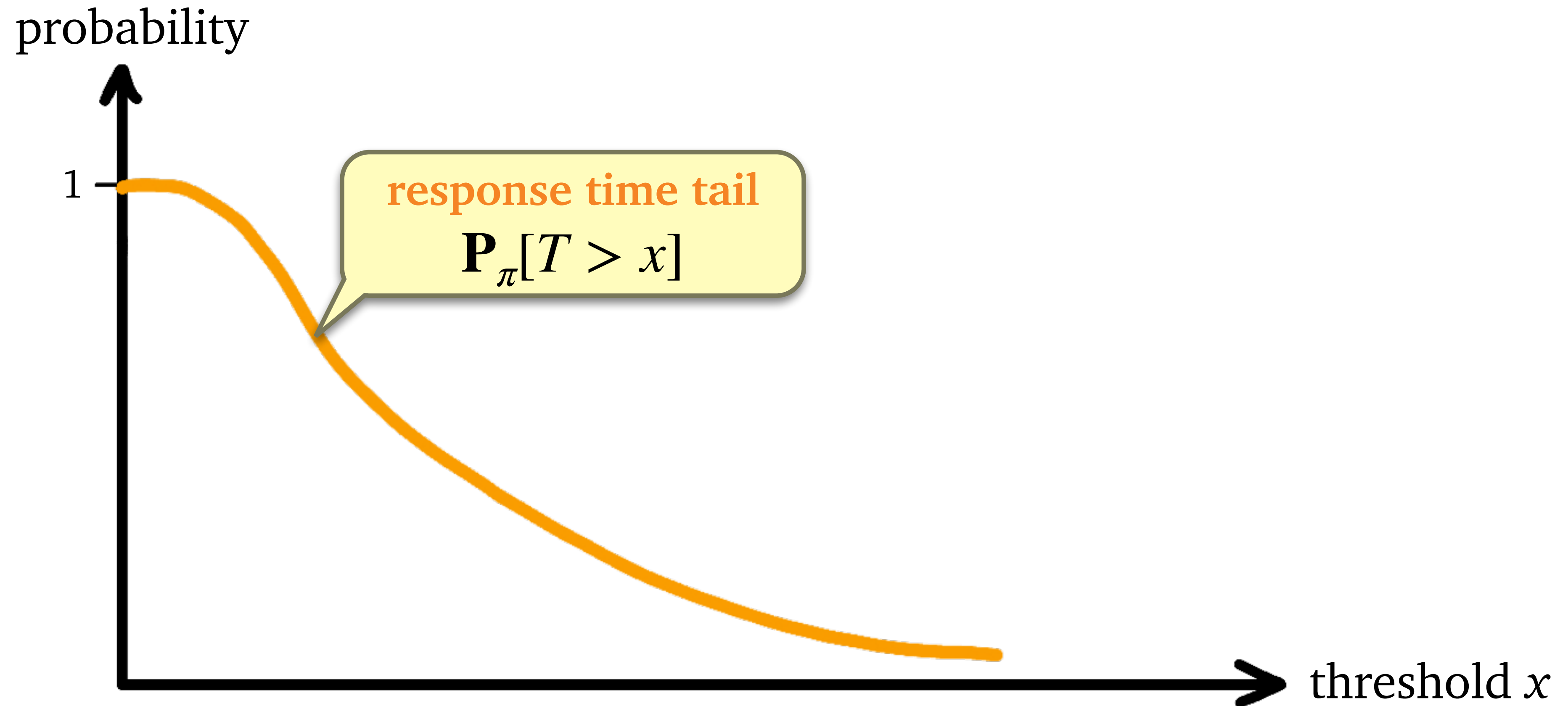
Is γ -Boost tail optimal in the $M/G/k$ in heavy traffic?



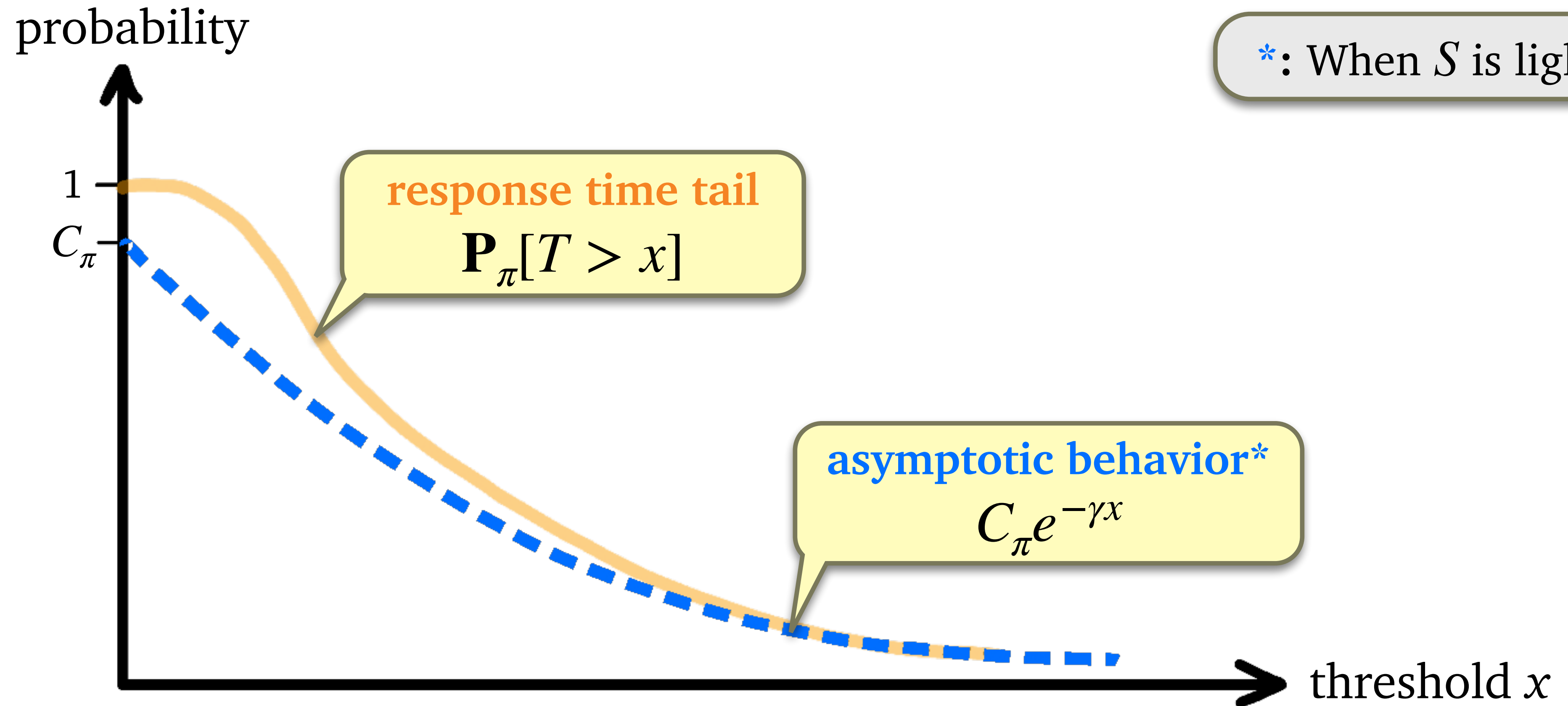
Does γ -Boost have good empirical tail performance in the $M/G/k$ in lighter traffic?



What do we mean by $\min \mathbf{P}[T > t]$ “for all large t ”?

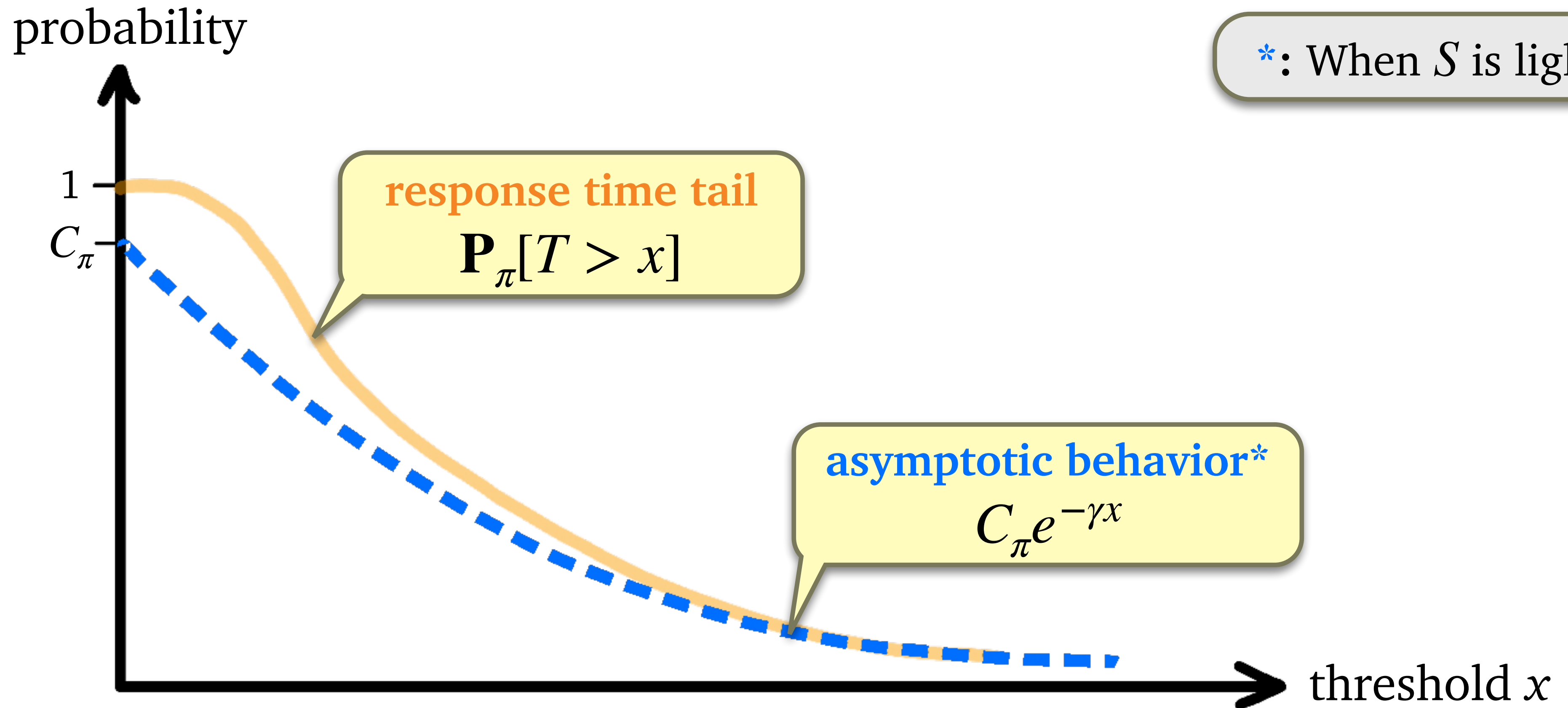


What do we mean by $\min \mathbf{P}[T > t]$ “for all large t ”?



$C_\pi = \text{tail constant of } \pi$

What do we mean by $\min \mathbf{P}[T > t]$ “for all large t ”?



$C_\pi = \text{tail constant of } \pi$

Goal: find π that minimizes C_π

What is the γ -Boost policy?

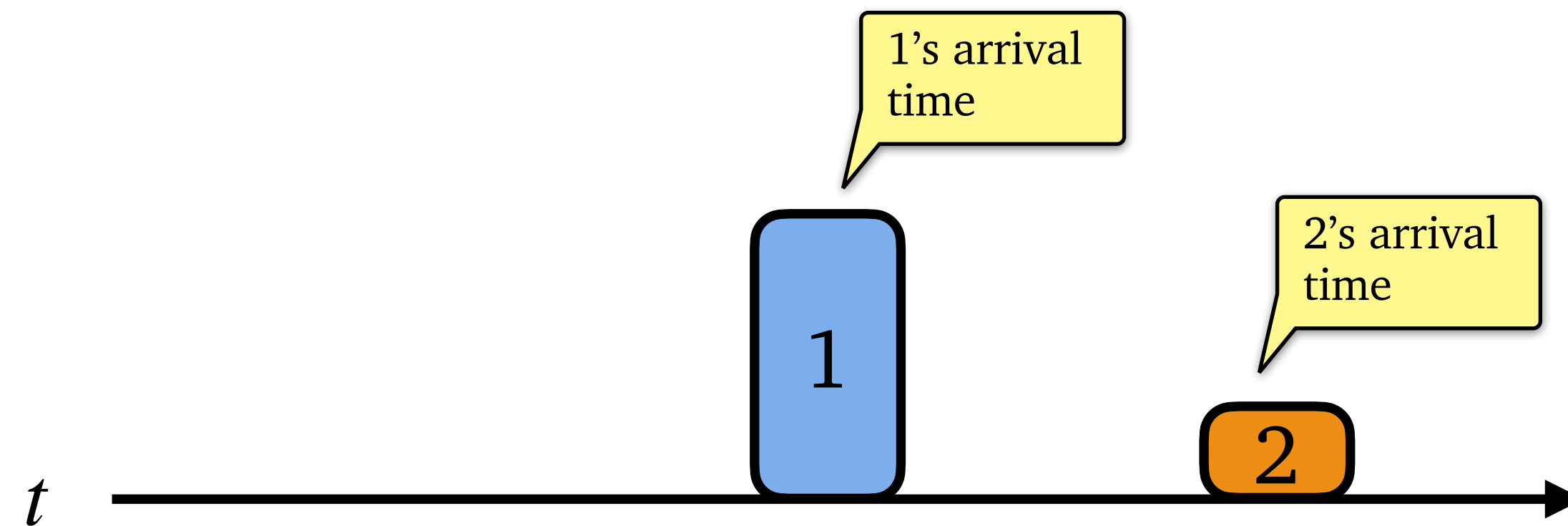
What is the γ -Boost policy?

γ -Boost is *like FCFS*, but gives each job of size s a *boost of $b(s)$ to its arrival time*.

$$b(s) = \frac{1}{\gamma} \log \left(\frac{1}{1 - e^{-\gamma s}} \right)$$

What is the γ -Boost policy?

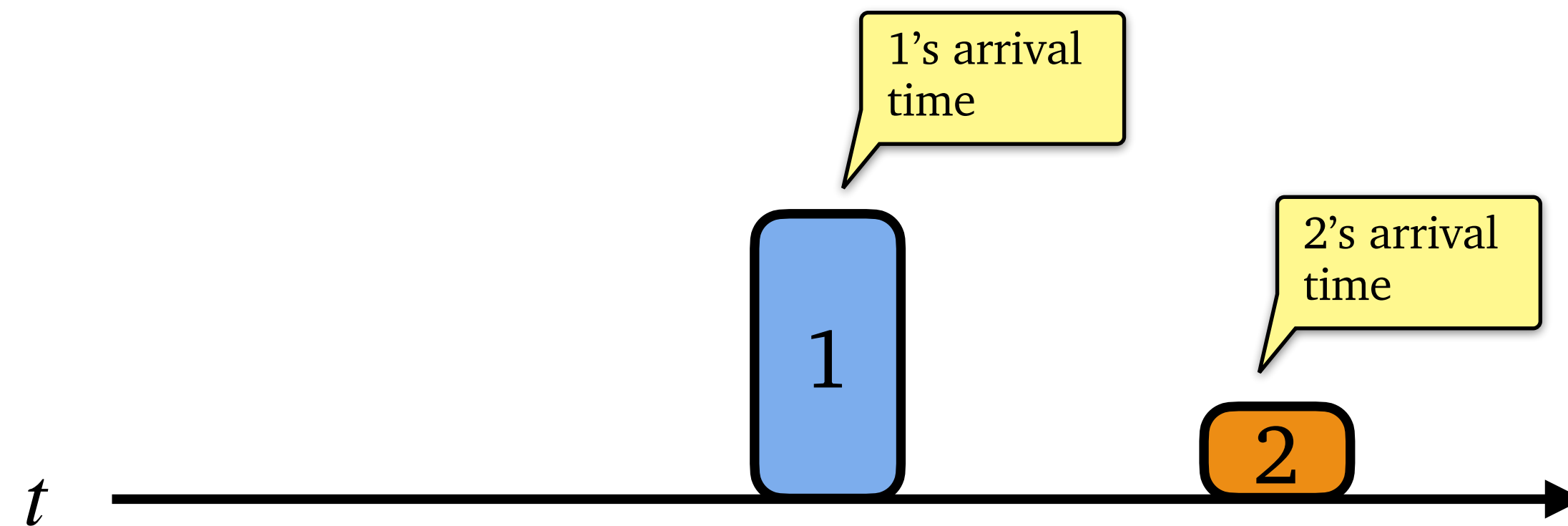
γ -Boost is like *FCFS*, but gives each job of size s a *boost* of $b(s)$ to its arrival time.



$$b(s) = \frac{1}{\gamma} \log \left(\frac{1}{1 - e^{-\gamma s}} \right)$$

What is the γ -Boost policy?

γ -Boost is like *FCFS*, but gives each job of size s a *boost of $b(s)$ to its arrival time*.

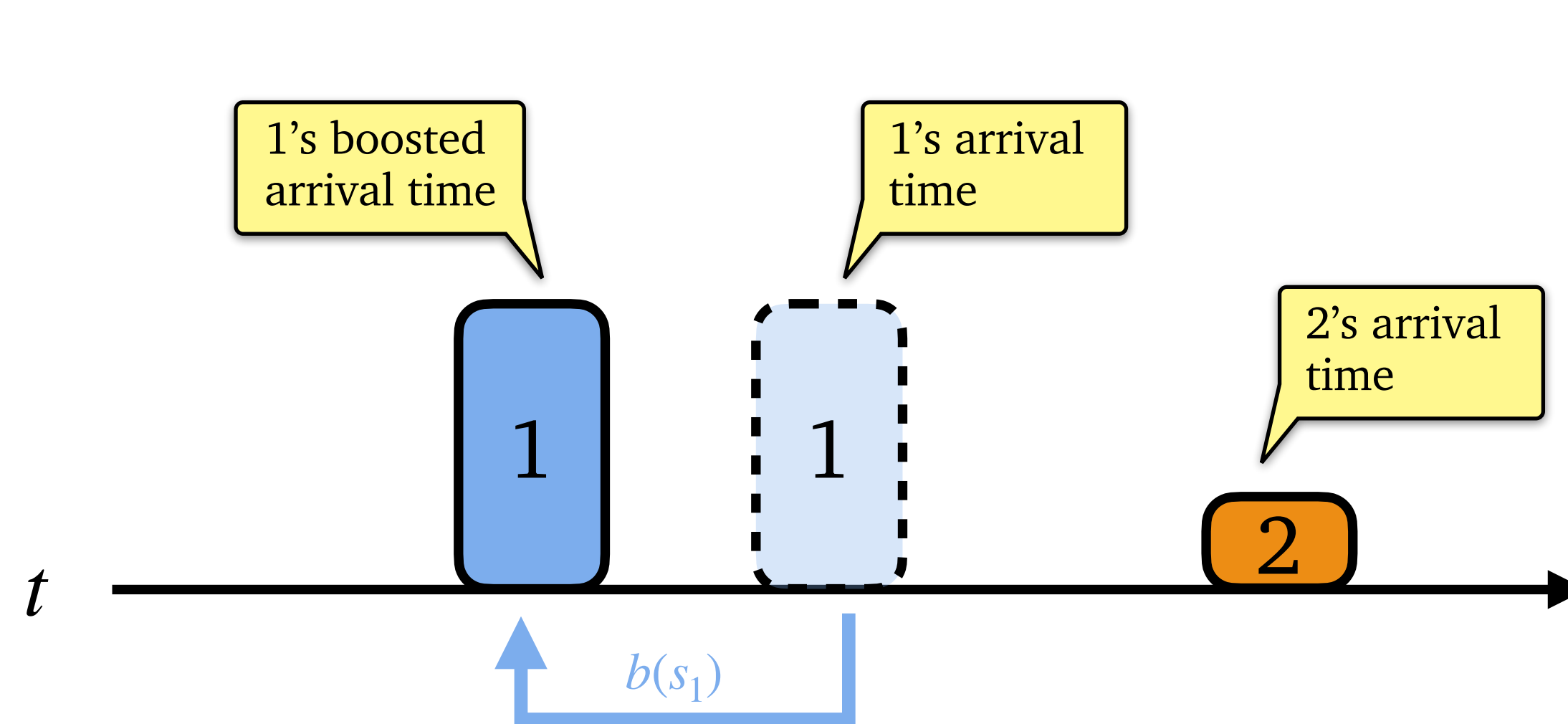


$$b(s) = \frac{1}{\gamma} \log \left(\frac{1}{1 - e^{-\gamma s}} \right)$$

FCFS: serves job 1 then job 2.

What is the γ -Boost policy?

γ -Boost is like *FCFS*, but gives each job of size s a *boost* of $b(s)$ to its arrival time.

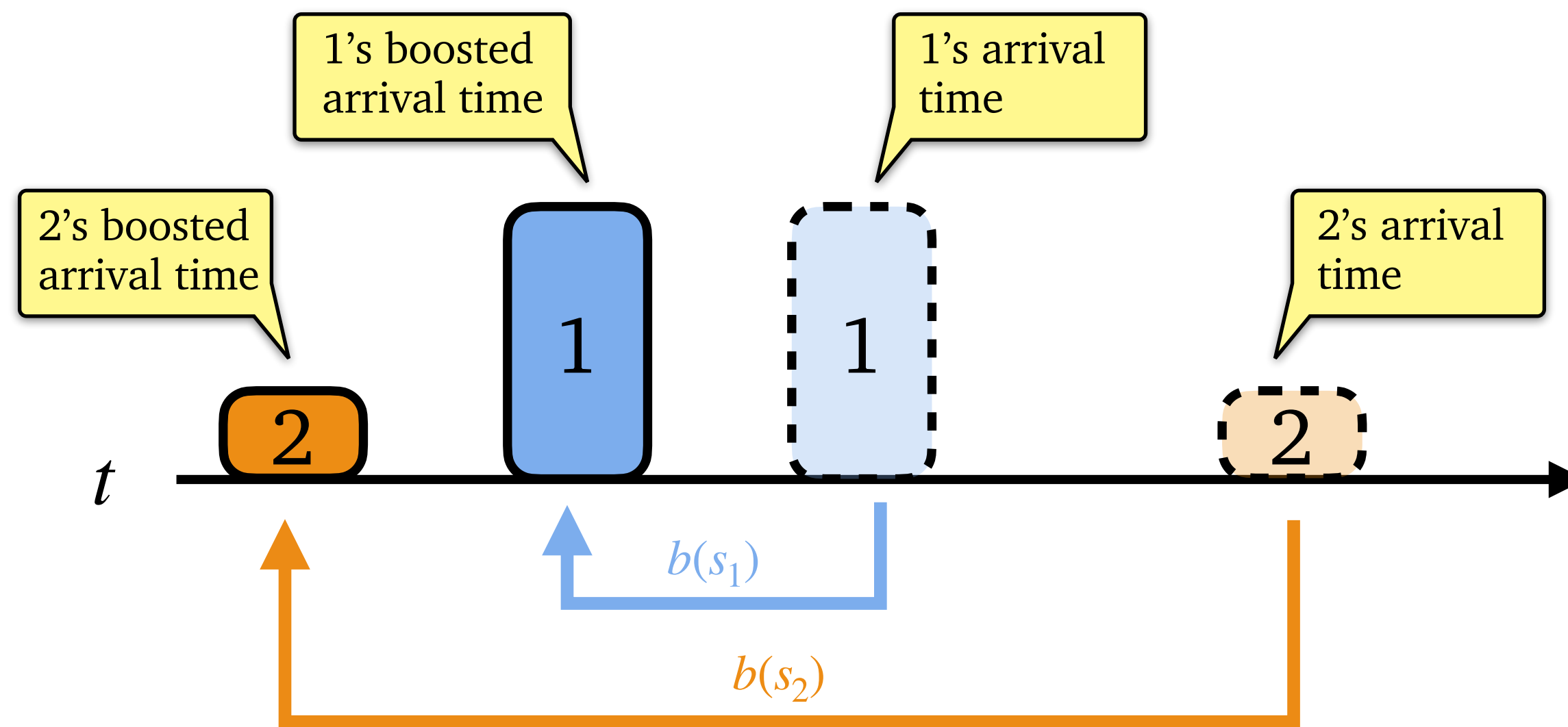


$$b(s) = \frac{1}{\gamma} \log \left(\frac{1}{1 - e^{-\gamma s}} \right)$$

FCFS: serves job 1 then job 2.

What is the γ -Boost policy?

γ -Boost is like *FCFS*, but gives each job of size s a *boost* of $b(s)$ to its arrival time.

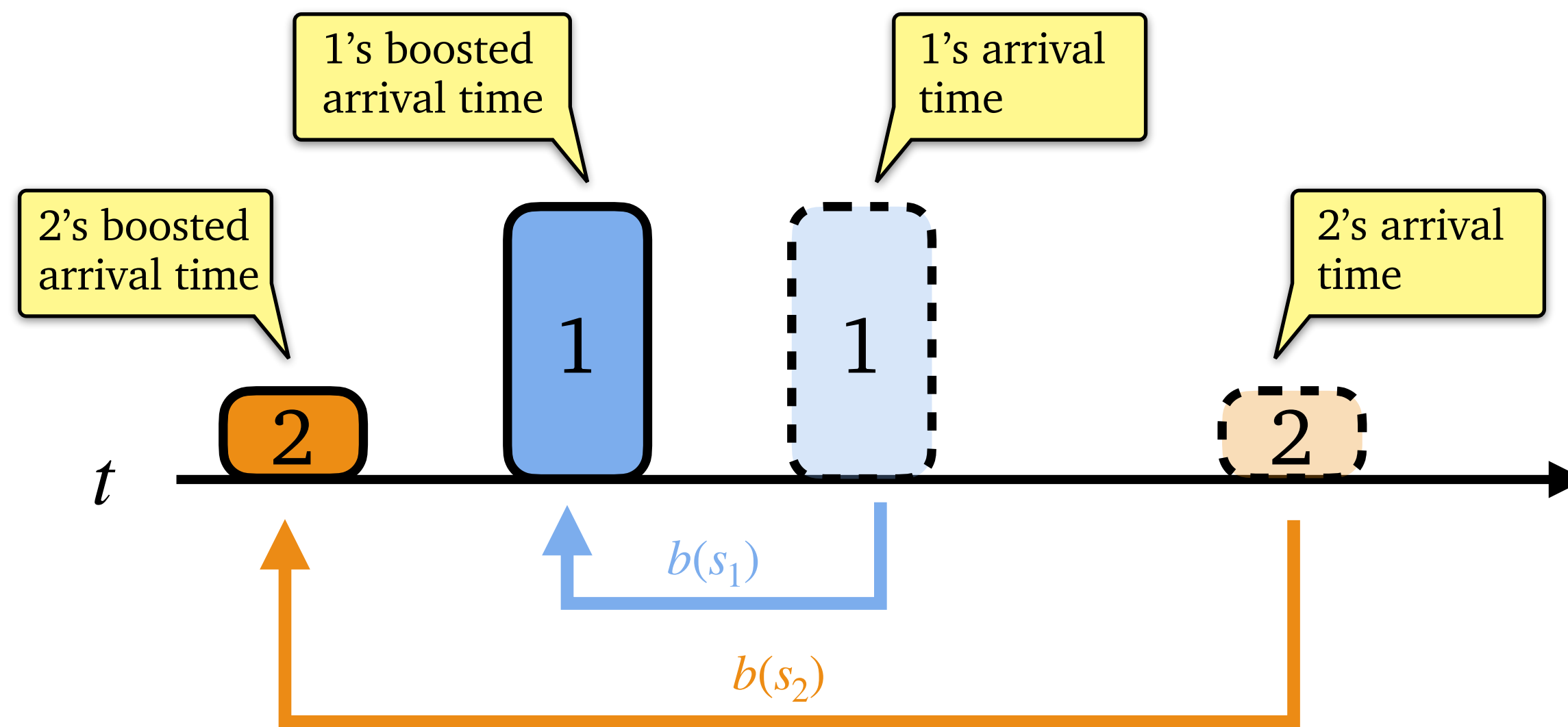


$$b(s) = \frac{1}{\gamma} \log \left(\frac{1}{1 - e^{-\gamma s}} \right)$$

FCFS: serves job 1 then job 2.

What is the γ -Boost policy?

γ -Boost is like *FCFS*, but gives each job of size s a *boost* of $b(s)$ to its arrival time.



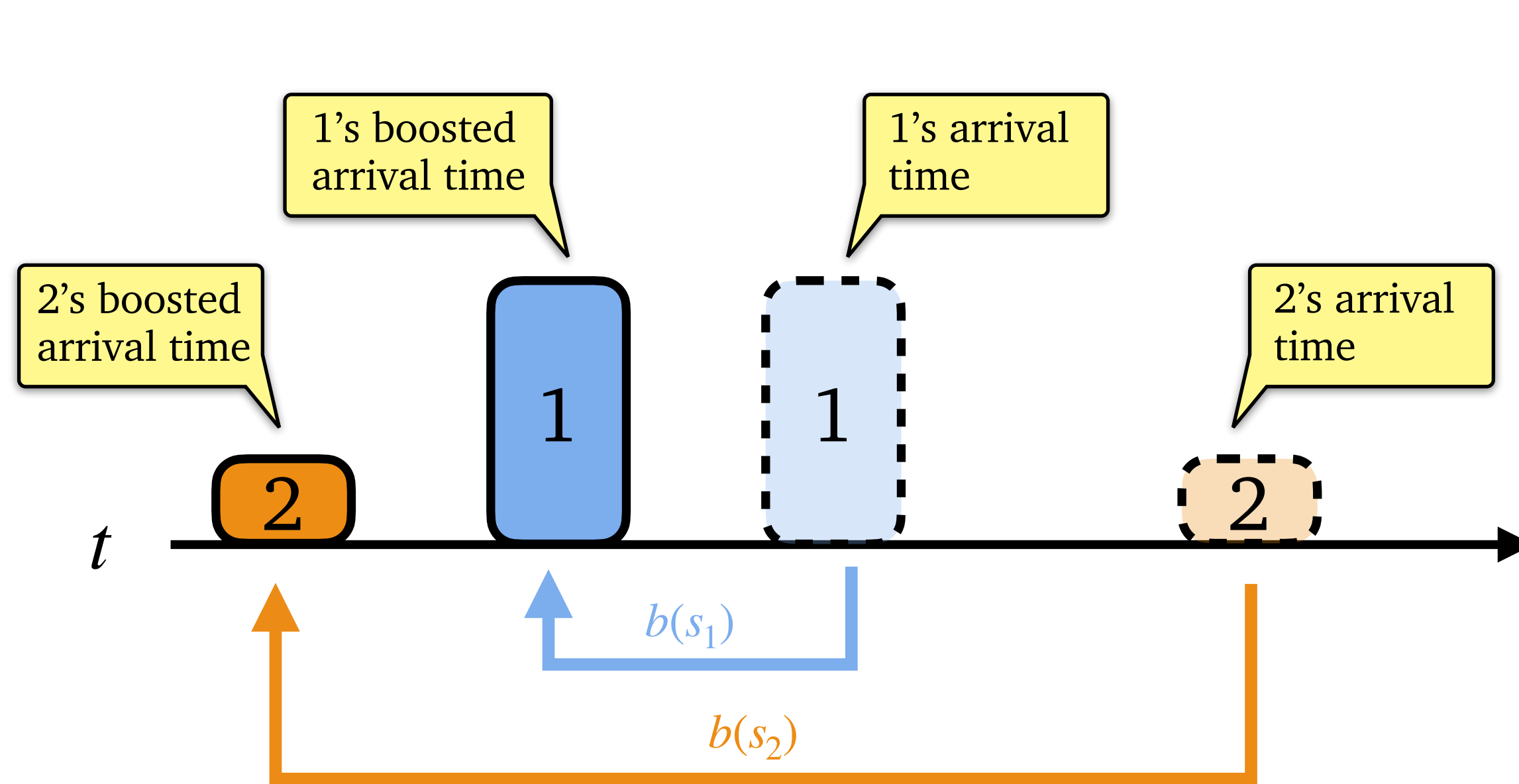
$$b(s) = \frac{1}{\gamma} \log \left(\frac{1}{1 - e^{-\gamma s}} \right)$$

FCFS: serves job 1 then job 2.

γ -Boost: serves job 2 then job 1.

What is the γ -Boost policy?

γ -Boost is like FCFS, but gives each job of size s a boost of $b(s)$ to its arrival time.



$$b(s) = \frac{1}{\gamma} \log \left(\frac{1}{1 - e^{-\gamma s}} \right)$$

FCFS: serves job 1 then job 2.

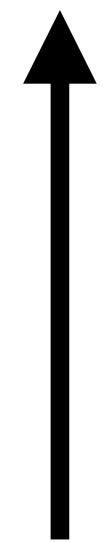
γ -Boost: serves job 2 then job 1.

Theorem: in the M/G/1, $C_{\gamma\text{-Boost}} = \inf_{\pi} C_{\pi}$ [Yu & Scully 2024]

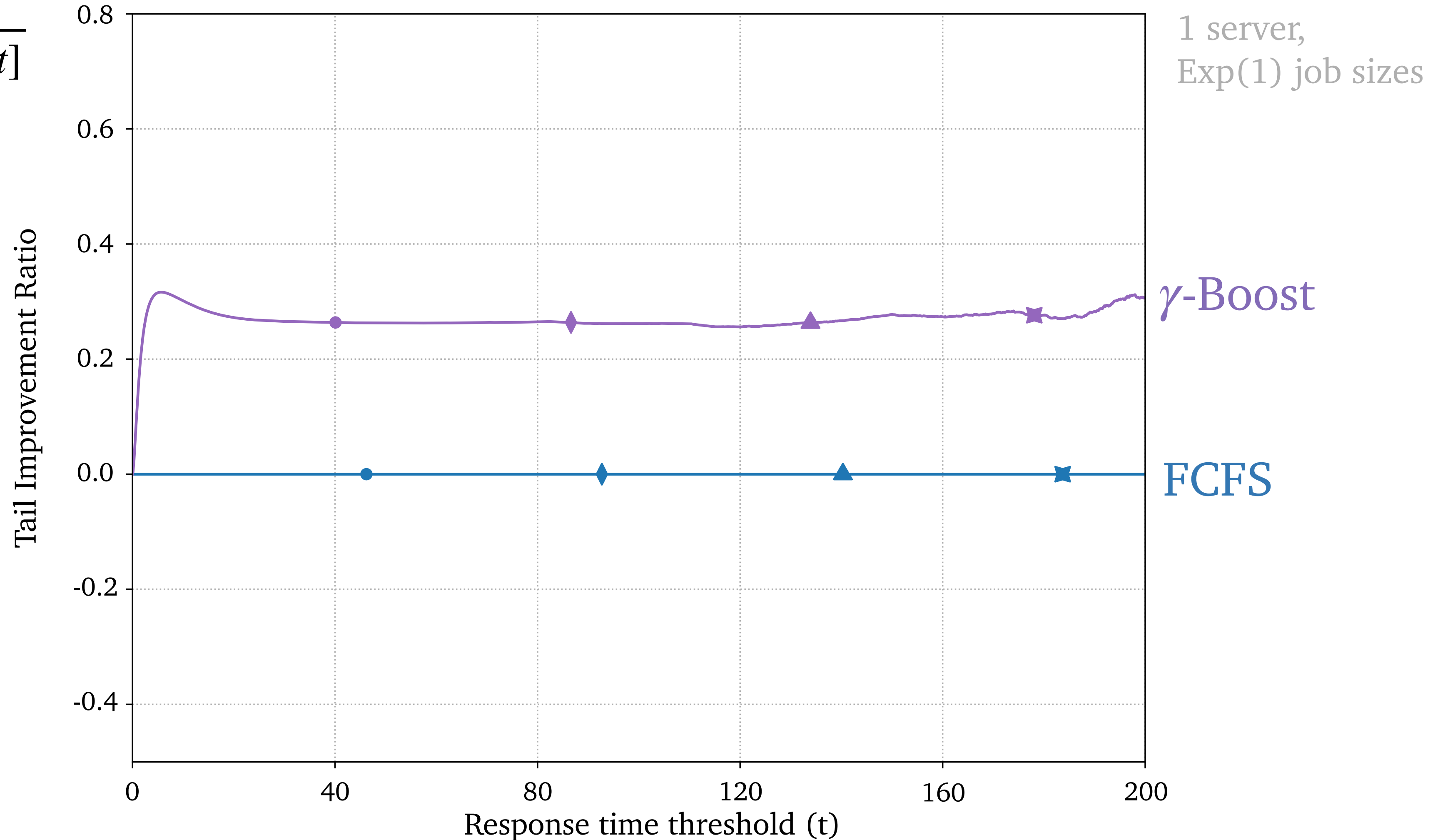
γ -Boost's performance in the M/G/1

Load: 0.95

$$1 - \frac{\mathbf{P}[T_{\pi} > t]}{\mathbf{P}[T_{\text{FCFS}} > t]}$$



(Higher is better)



γ -Boost

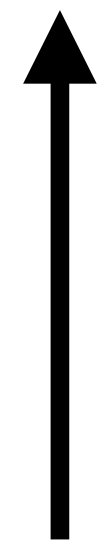
FCFS

- 90th percentile
- ◆ 99th percentile
- ▲ 99.9th percentile
- ✱ 99.99th percentile
- ★ 99.999th percentile

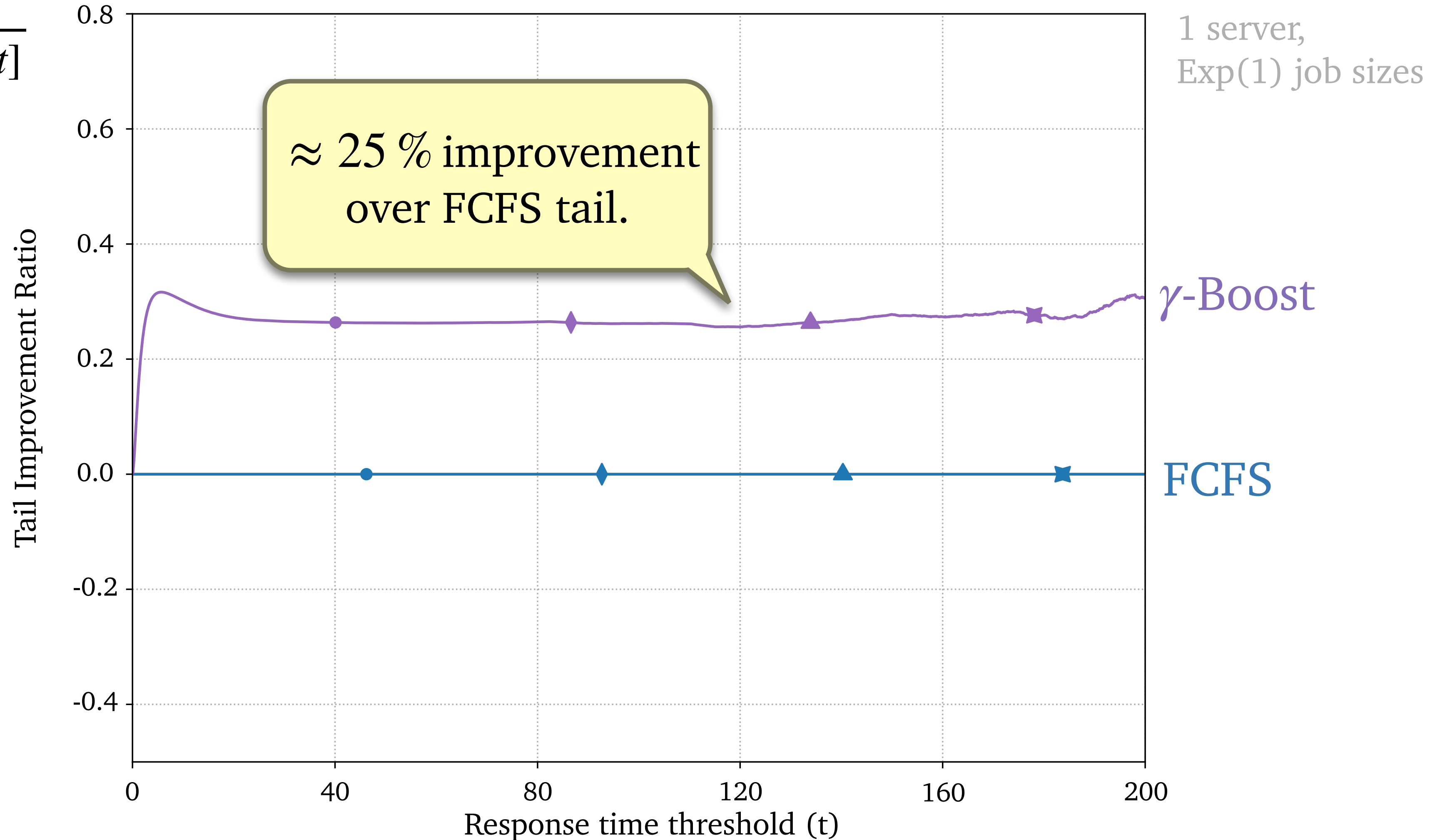
γ -Boost's performance in the M/G/1

Load: 0.95

$$1 - \frac{\mathbf{P}[T_{\pi} > t]}{\mathbf{P}[T_{\text{FCFS}} > t]}$$



(Higher is better)



- 90th percentile
- ◆ 99th percentile
- ▲ 99.9th percentile
- ✱ 99.99th percentile
- ★ 99.999th percentile

This talk



What do we mean by $\min \mathbf{P}[T > t]$ “for all large t ”?



What is the γ -Boost policy?



Is γ -Boost tail optimal in the M/G/k in heavy traffic?

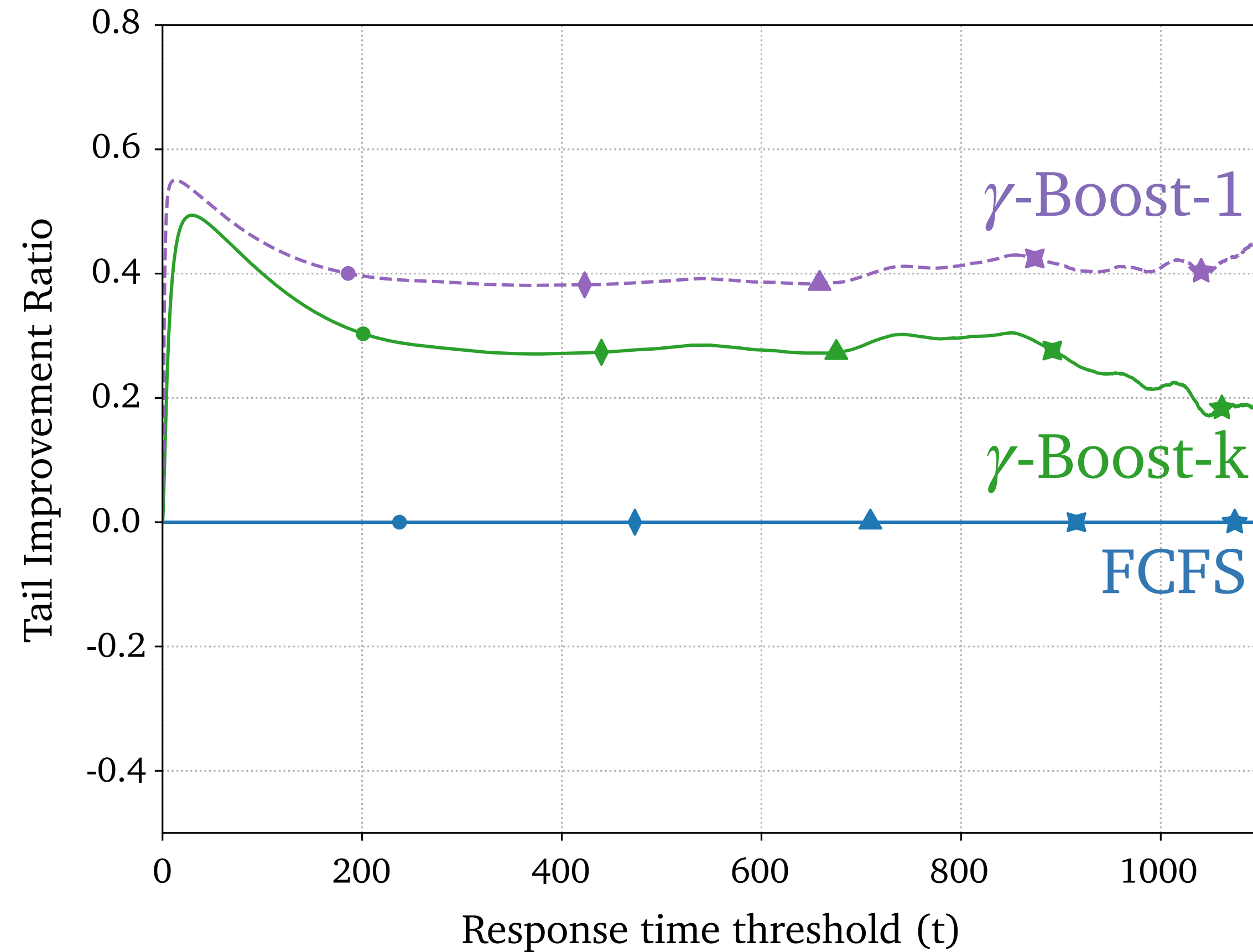


Does γ -Boost have good empirical tail performance in the M/G/k in lighter traffic?

γ -Boost-k performance in heavy traffic

γ -Boost-k performance in heavy traffic

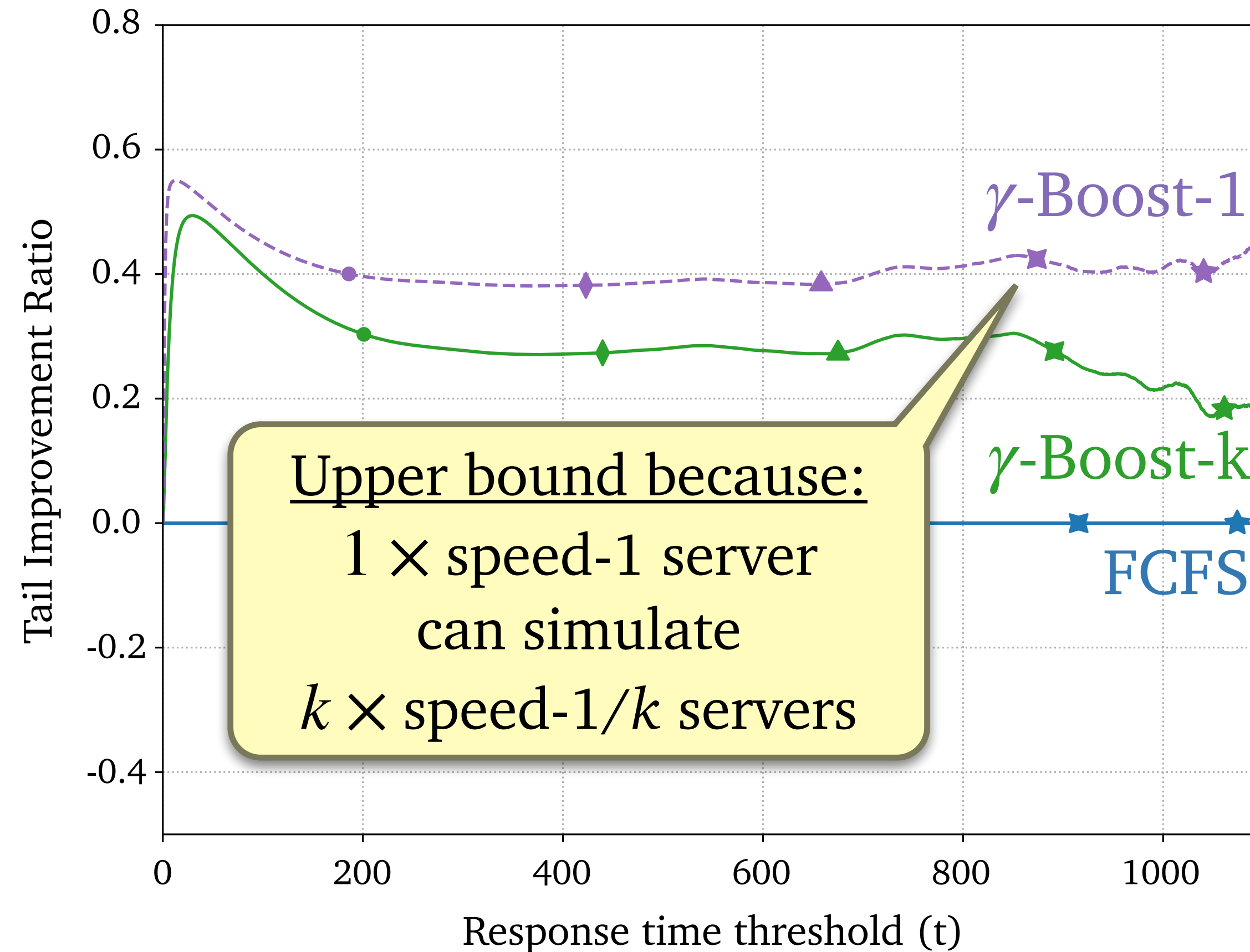
Load: 0.99



• 90th percentile ♦ 99th percentile ▲ 99.9th percentile ✦ 99.99th percentile ★ 99.999th percentile

γ -Boost- k performance in heavy traffic

Load: 0.99

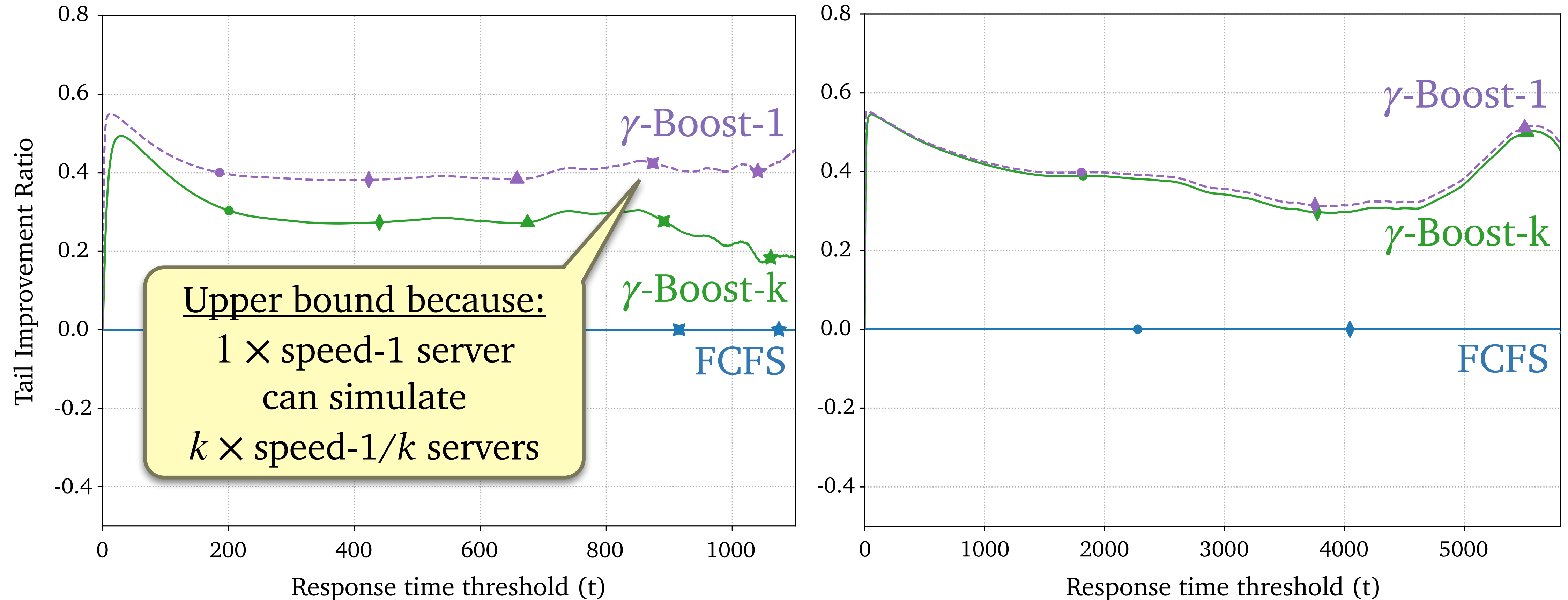


• 90th percentile ♦ 99th percentile ▲ 99.9th percentile ✦ 99.99th percentile ★ 99.999th percentile

γ -Boost-k performance in heavy traffic

Load: 0.99

Load: 0.999

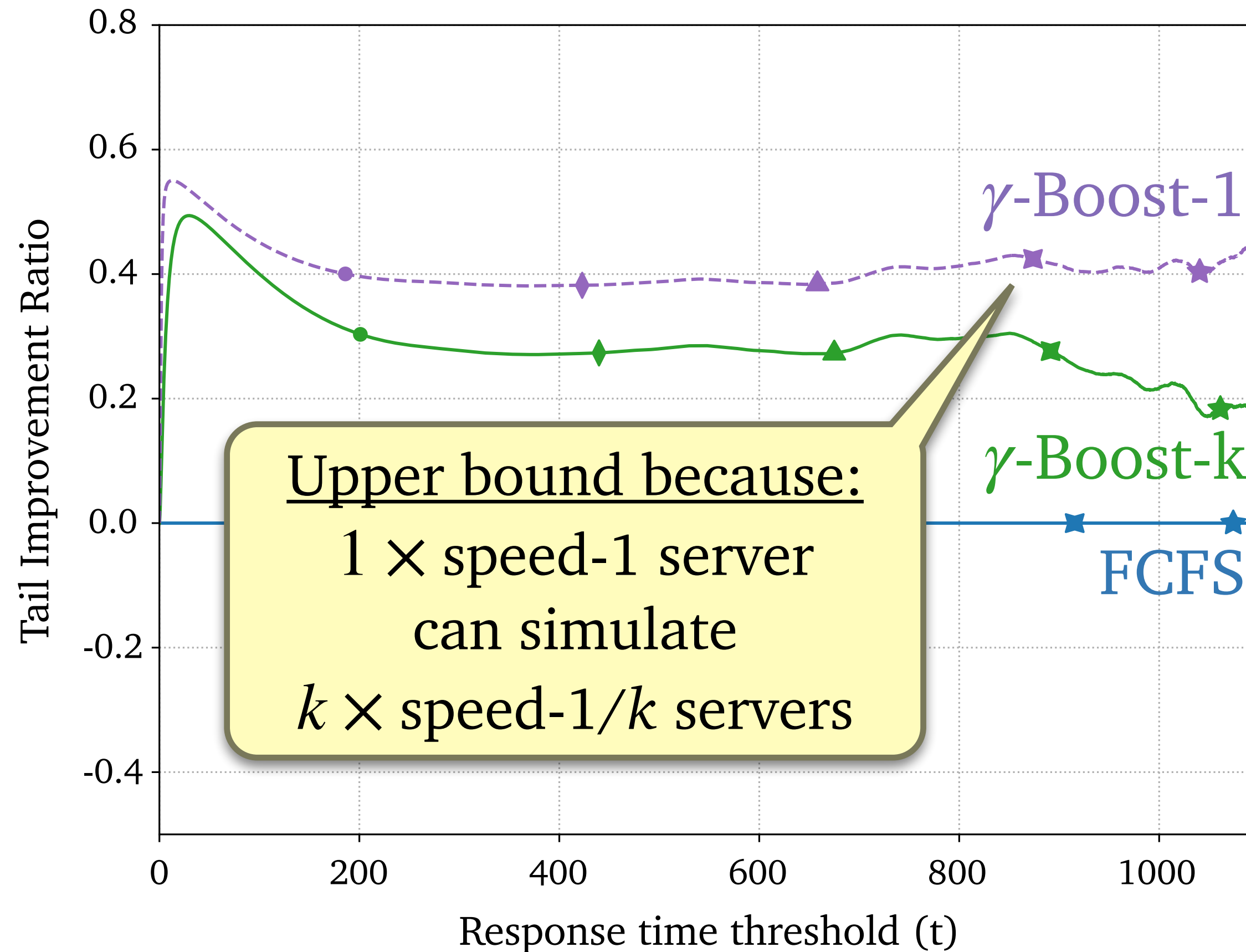


• 90th percentile ◆ 99th percentile ▲ 99.9th percentile ✦ 99.99th percentile ★ 99.999th percentile

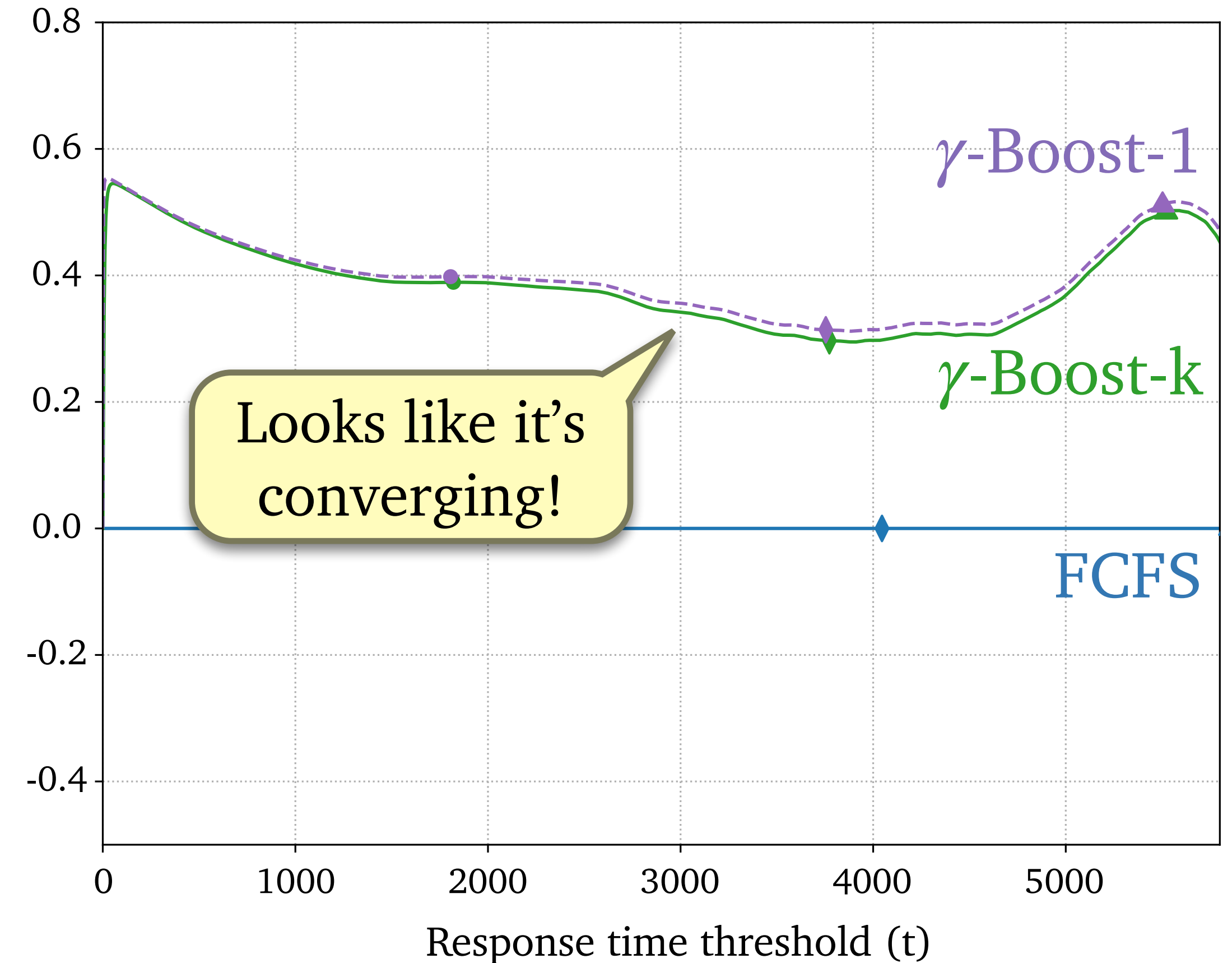
10 servers,
Exp(1) job sizes

γ -Boost- k performance in heavy traffic

Load: 0.99



Load: 0.999



90th percentile
 99th percentile
 99.9th percentile
 99.99th percentile
 99.999th percentile



Theorem: $\lim_{\rho \rightarrow 1} C_{\gamma\text{-Boost-}k} = C_{\gamma\text{-Boost-}1}$



Theorem: $\lim_{\rho \rightarrow 1} C_{\gamma\text{-Boost-}k} = C_{\gamma\text{-Boost-}1}$

Main tool: Work Decomposition Law

$$W_{M/G/k} = W_{M/G/1} + W_{\text{extra}}$$



Theorem: $\lim_{\rho \rightarrow 1} C_{\gamma\text{-Boost-}k} = C_{\gamma\text{-Boost-}1}$

Main tool: Work Decomposition Law

$$W_{M/G/k} = W_{M/G/1} + W_{\text{extra}}$$

Independent



Theorem: $\lim_{\rho \rightarrow 1} C_{\gamma\text{-Boost-}k} = C_{\gamma\text{-Boost-}1}$

Main tool: Work Decomposition Law

$$W_{M/G/k} = W_{M/G/1} + W_{\text{extra}}$$

Depends on π

Independent

Depends on π



Theorem: $\lim_{\rho \rightarrow 1} C_{\gamma\text{-Boost-}k} = C_{\gamma\text{-Boost-}1}$

Main tool: Work Decomposition Law

$$W_{M/G/k} = W_{M/G/1} + W_{\text{extra}}$$

Depends on π

Independent

Depends on π



Lemma: $\mathbf{P}[W_{\text{extra}} > t] \leq \min\left(1, \frac{k\rho}{1-\rho} \mathbf{P}[kS_{\text{excess}} > t]\right)$



Theorem: $\lim_{\rho \rightarrow 1} C_{\gamma\text{-Boost-}k} = C_{\gamma\text{-Boost-}1}$

Main tool: Work Decomposition Law

$$W_{M/G/k} = W_{M/G/1} + W_{\text{extra}}$$

Depends on π

Independent

Depends on π



Lemma: $\mathbf{P}[W_{\text{extra}} > t] \leq \min\left(1, \frac{k\rho}{1-\rho} \mathbf{P}[kS_{\text{excess}} > t]\right)$

For all work-conserving policies!



Theorem: $\lim_{\rho \rightarrow 1} C_{\gamma\text{-Boost-}k} = C_{\gamma\text{-Boost-}1}$

Main tool: Work Decomposition Law

$$W_{M/G/k} = W_{M/G/1} + W_{\text{extra}}$$

Depends on π

Independent

Depends on π



Lemma: $\mathbf{P}[W_{\text{extra}} > t] \leq \min\left(1, \frac{k\rho}{1-\rho} \mathbf{P}[kS_{\text{excess}} > t]\right)$



George's Lemma

For all work-conserving policies!

This talk



What do we mean by $\min \mathbf{P}[T > t]$ “for all large t ”?



What is the γ -Boost policy?



Is γ -Boost tail optimal in the M/G/k in heavy traffic?



Does γ -Boost have good empirical tail performance in the M/G/k in lighter traffic?

This talk



What do we mean by $\min \mathbf{P}[T > t]$ “for all large t ”?



What is the γ -Boost policy?



Is γ -Boost tail optimal in the M/G/k in heavy traffic?

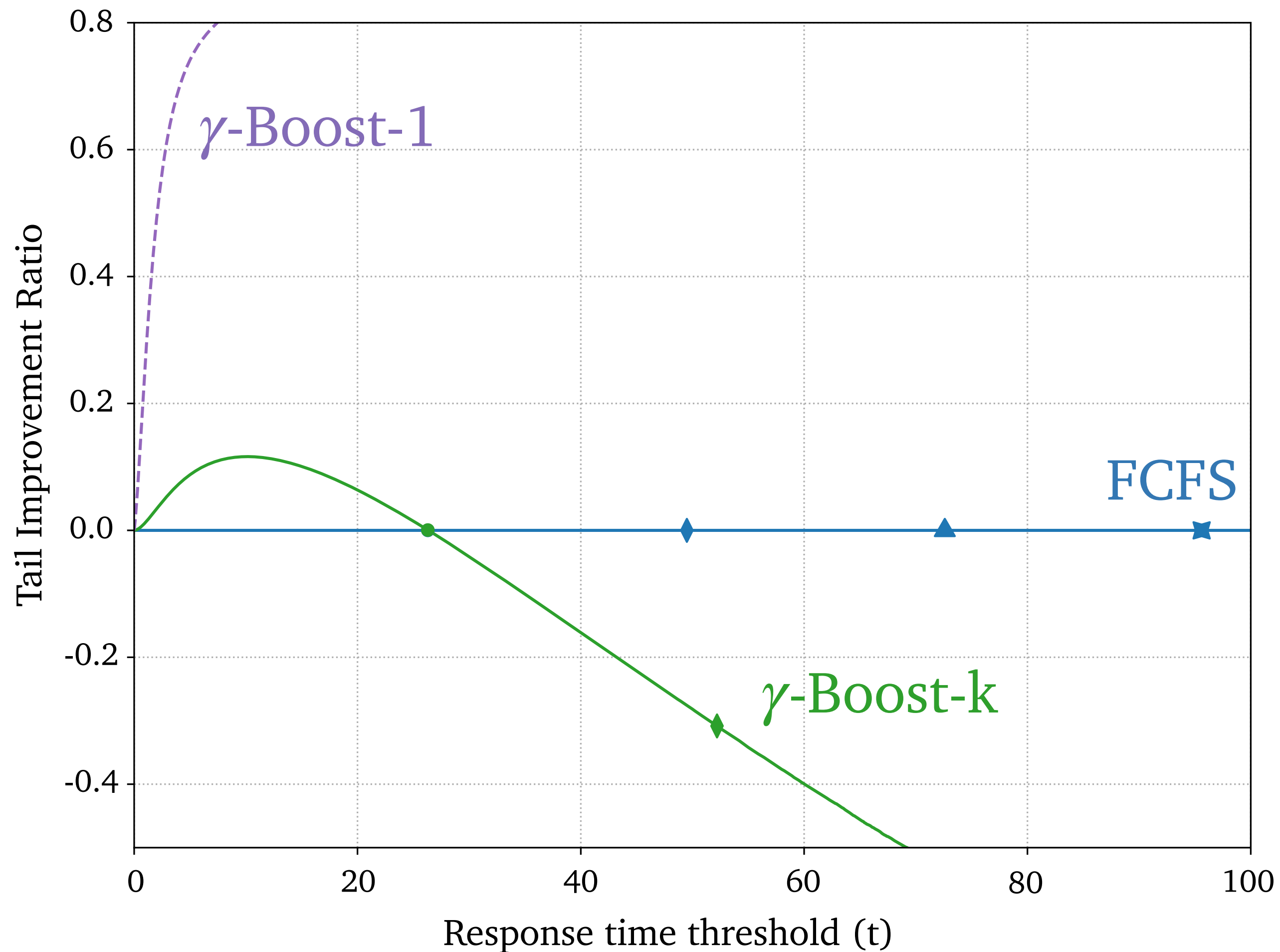
Yes!



Does γ -Boost have good empirical tail performance in the M/G/k in lighter traffic?

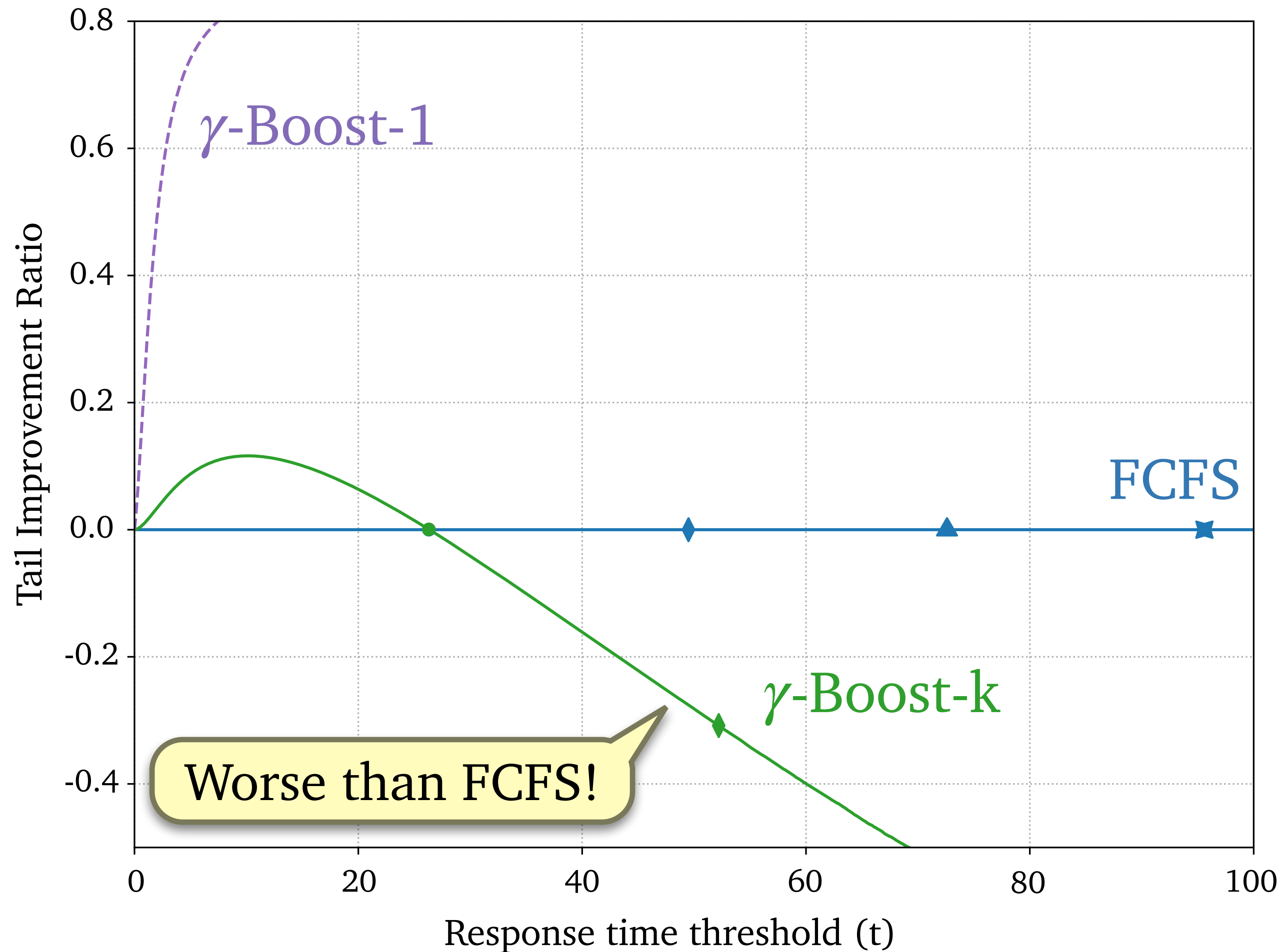
γ -Boost performance in lighter traffic

Load: 0.8



γ -Boost performance in lighter traffic

Load: 0.8



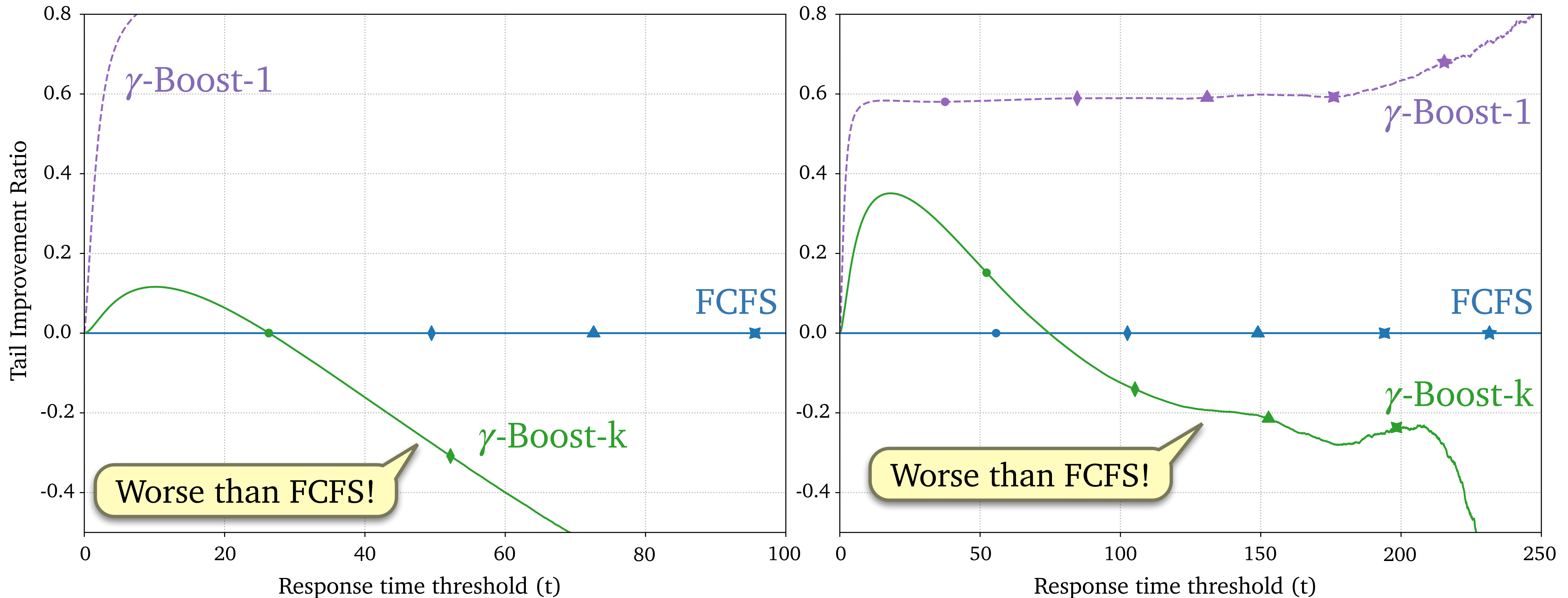
Worse than FCFS!

• 90th percentile ◆ 99th percentile ▲ 99.9th percentile ✦ 99.99th percentile ★ 99.999th percentile

γ -Boost performance in lighter traffic

Load: 0.8

Load: 0.95



This talk



What do we mean by $\min \mathbf{P}[T > t]$ “for all large t ”?



What is the γ -Boost policy?



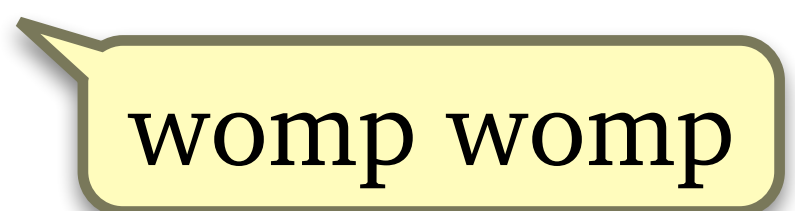
Is γ -Boost tail optimal in the M/G/k in heavy traffic?

Yes!



Does γ -Boost have good empirical tail performance in the M/G/k in lighter traffic?

No..



This talk



What do we mean by $\min \mathbf{P}[T > t]$ “for all large t ”?



What is the γ -Boost policy?



Is γ -Boost tail optimal in the M/G/k in heavy traffic?

Yes!



Does γ -Boost have *balance* in the M/G/k in lighter traffic?

What should we do instead?

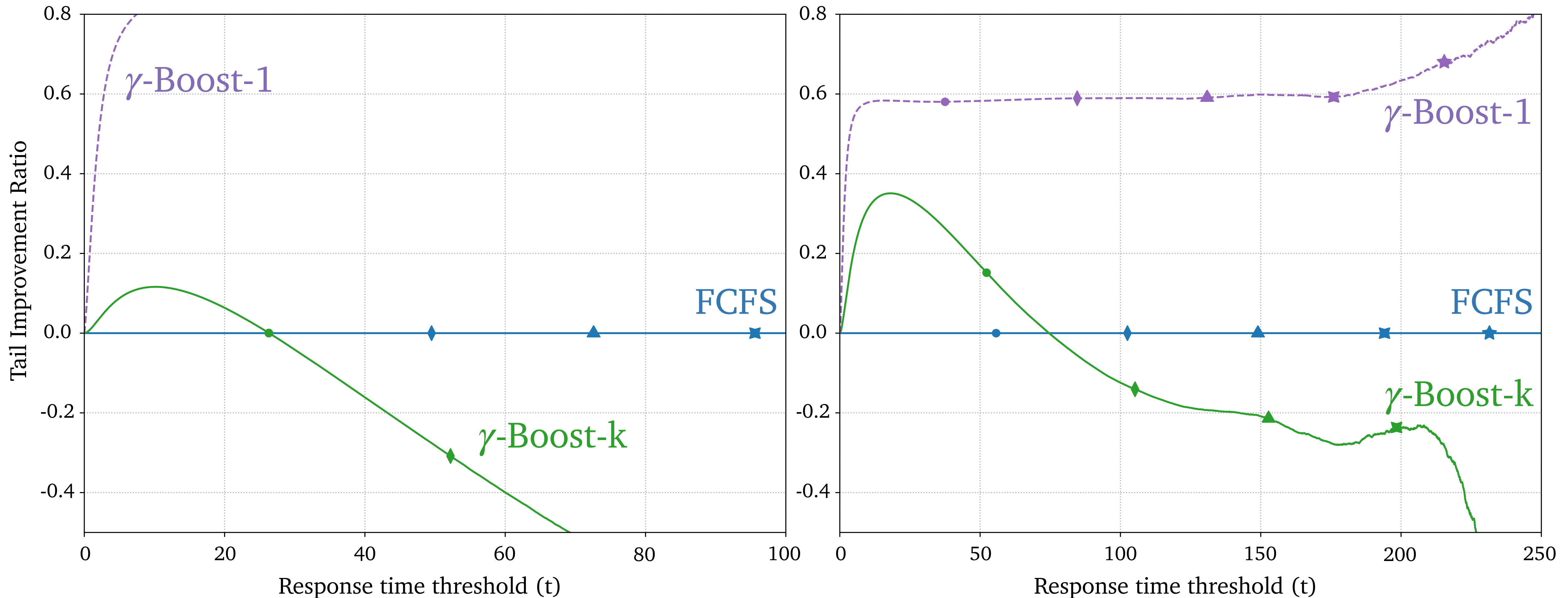
No..

womp womp

γ -Boost performance in lighter traffic

Load: 0.8

Load: 0.95



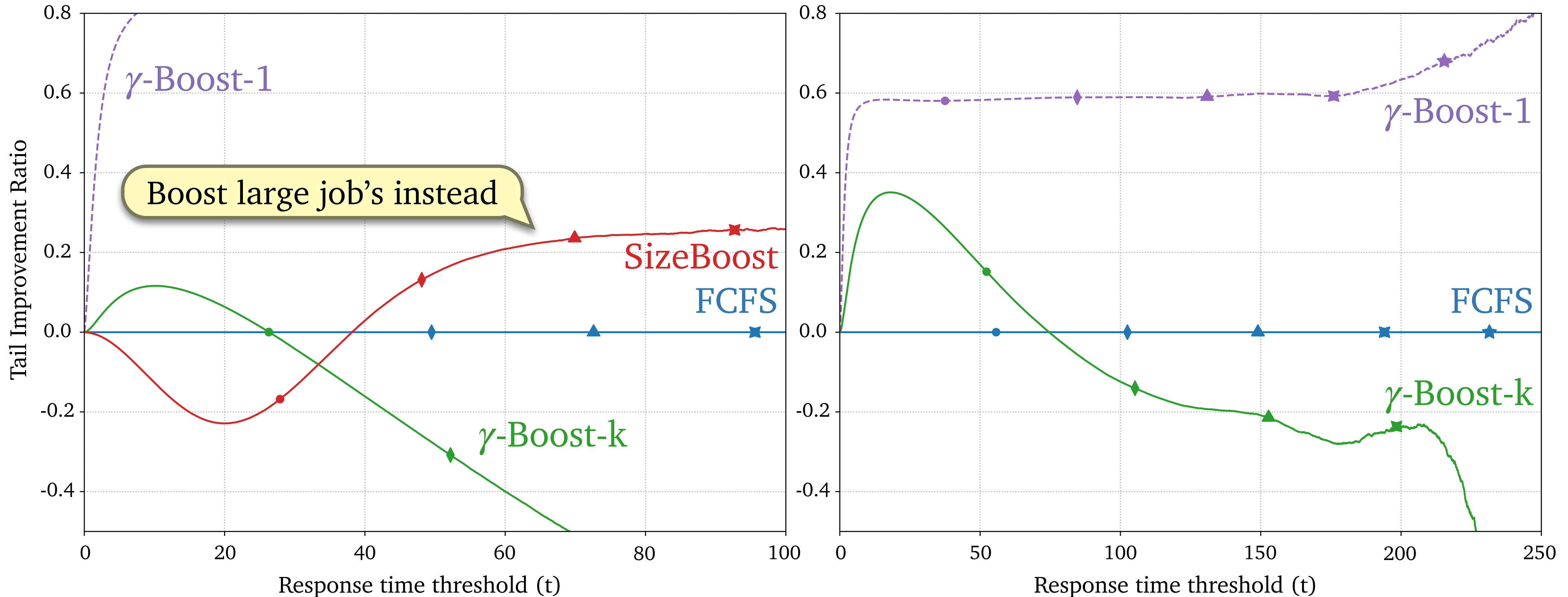
10 servers,
Exp(1) job sizes



γ -Boost performance in lighter traffic

Load: 0.8

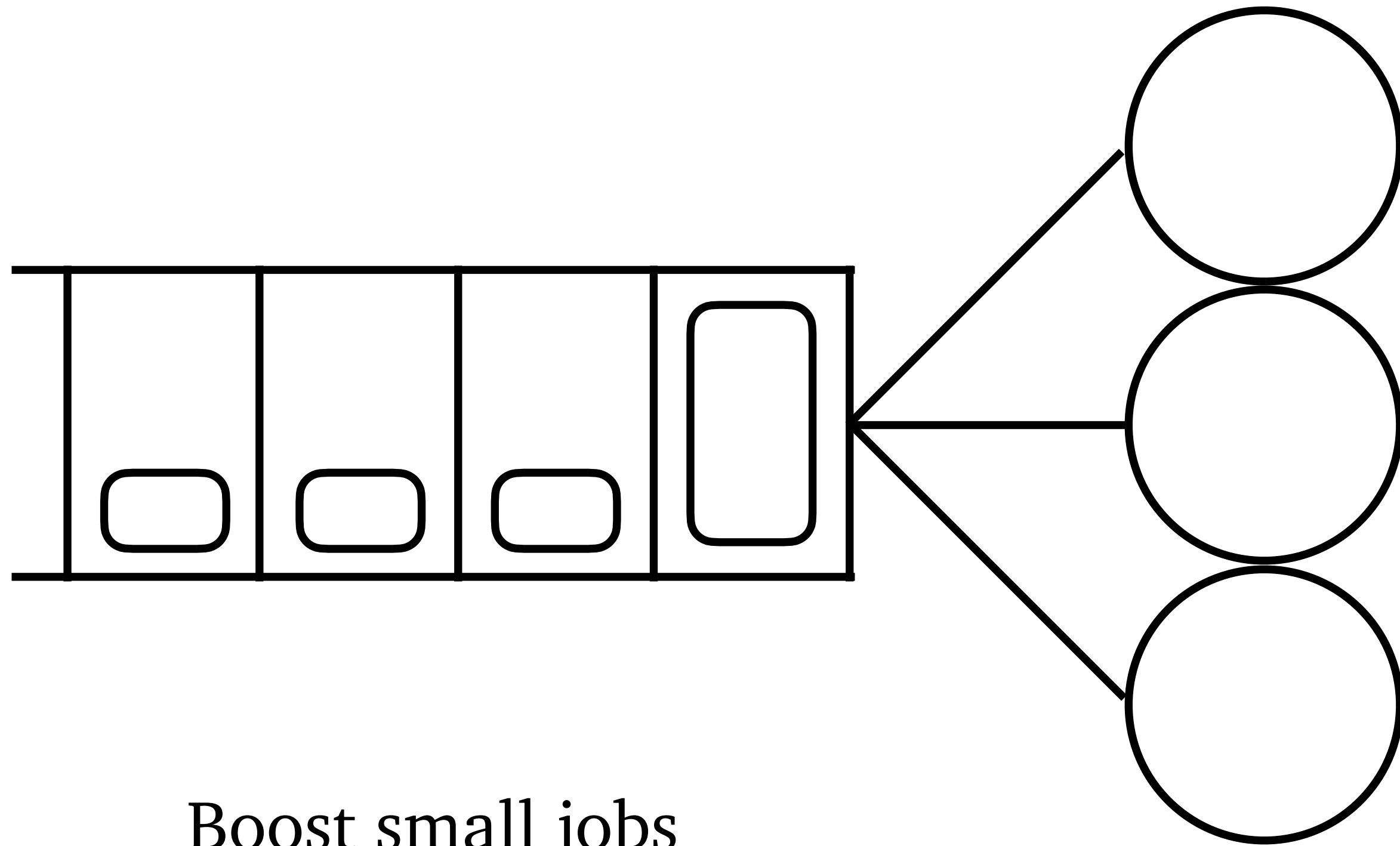
Load: 0.95



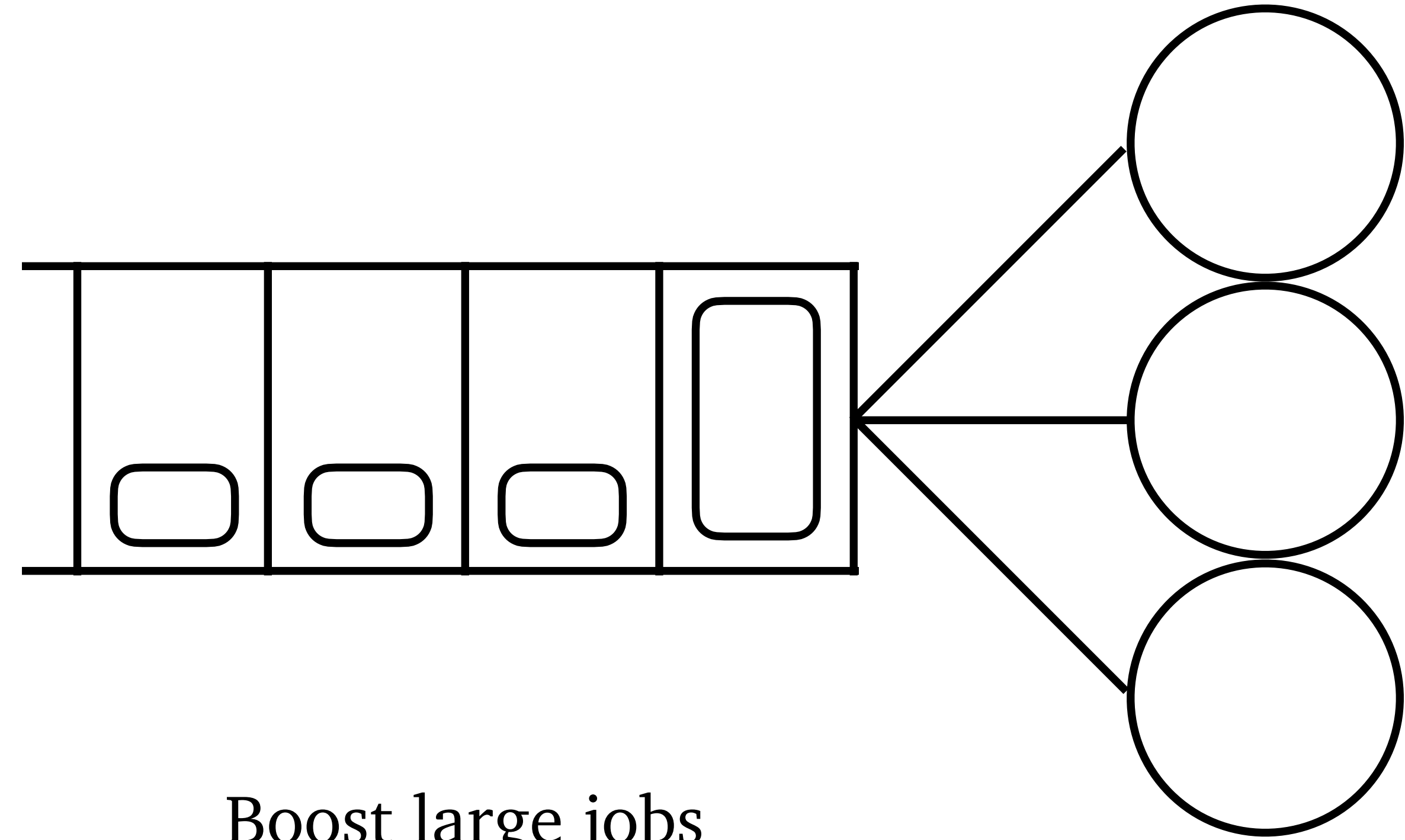
10 servers,
Exp(1) job sizes



Why might SizeBoost make sense at low load?

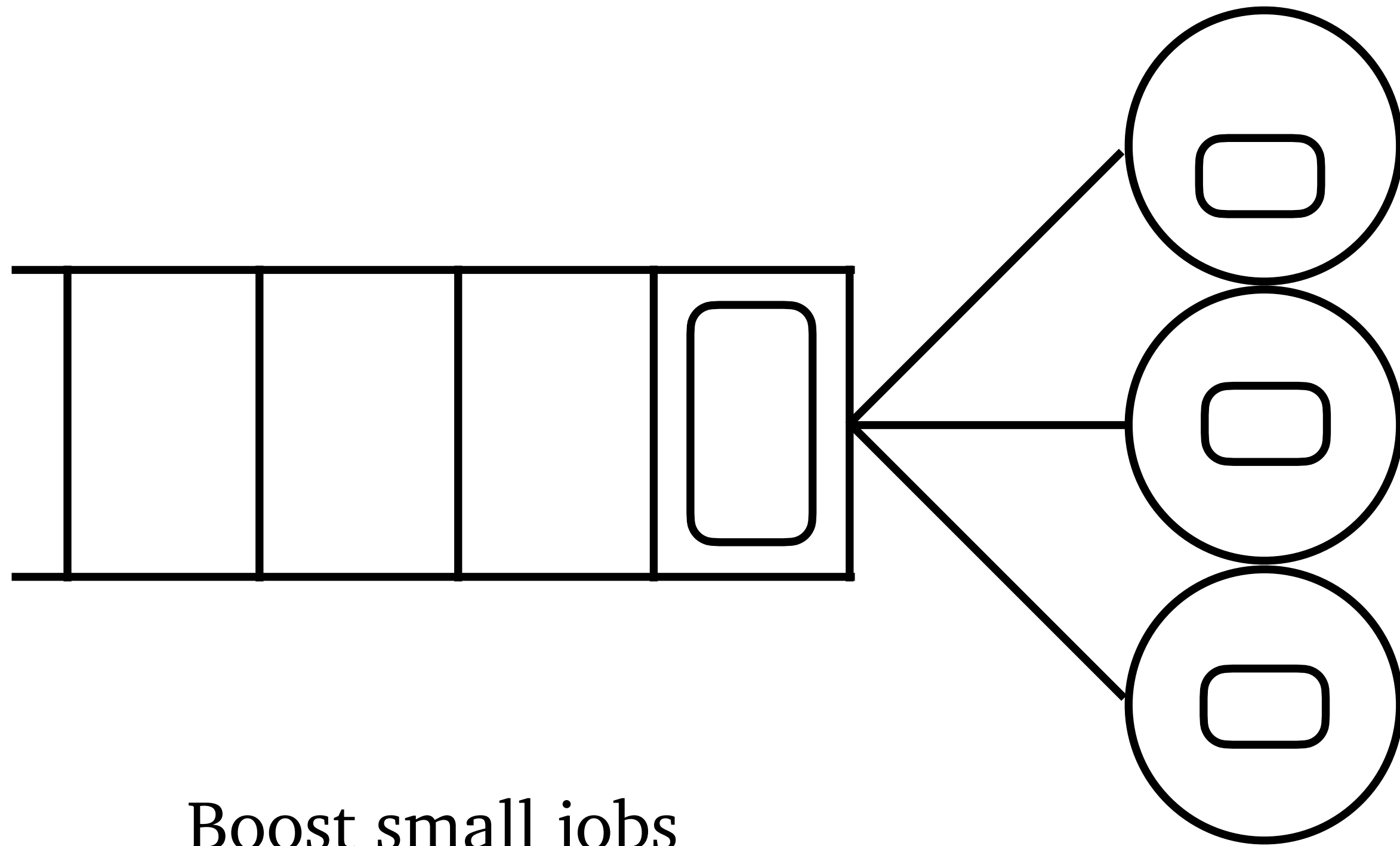


Boost small jobs

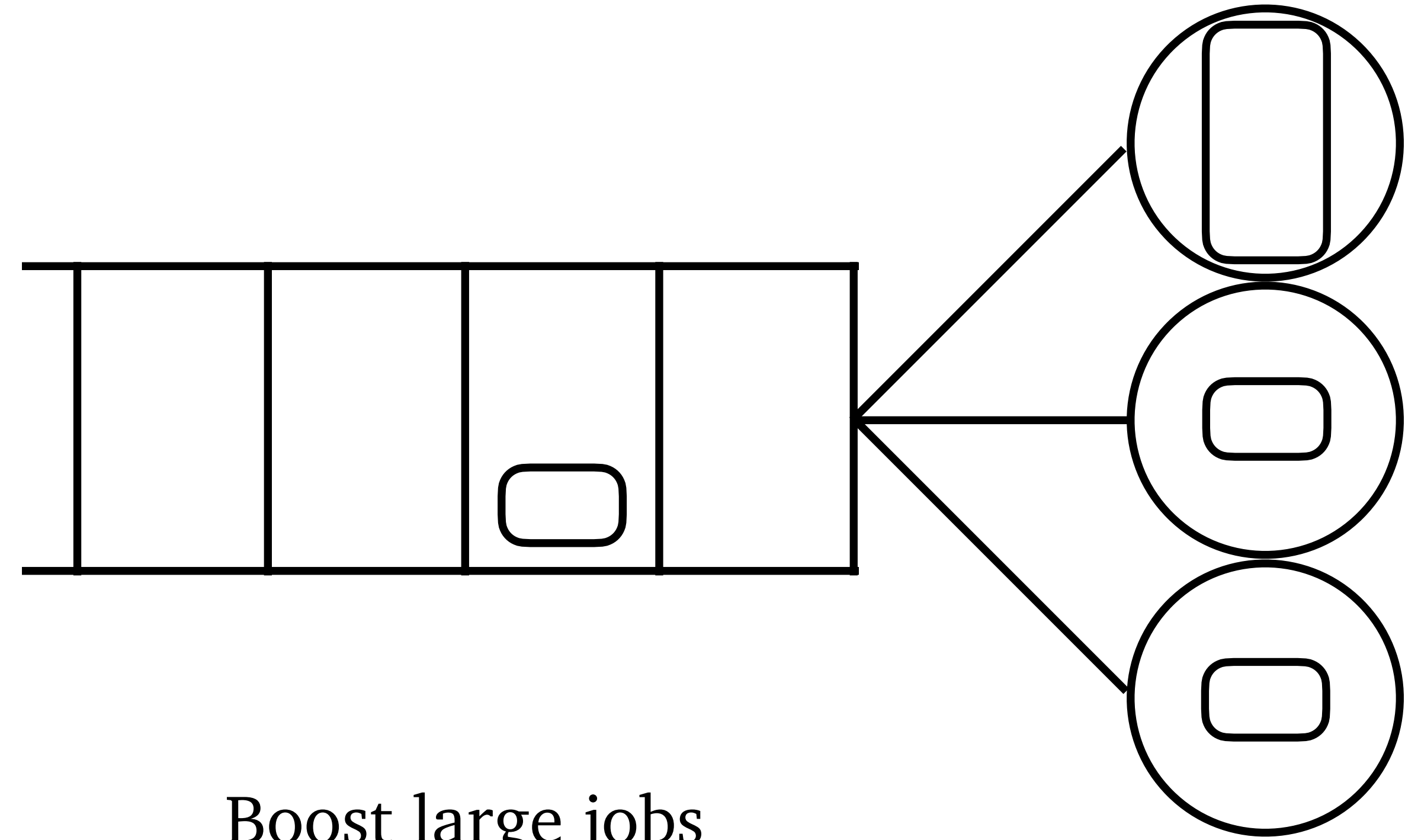


Boost large jobs

Why might SizeBoost make sense at low load?

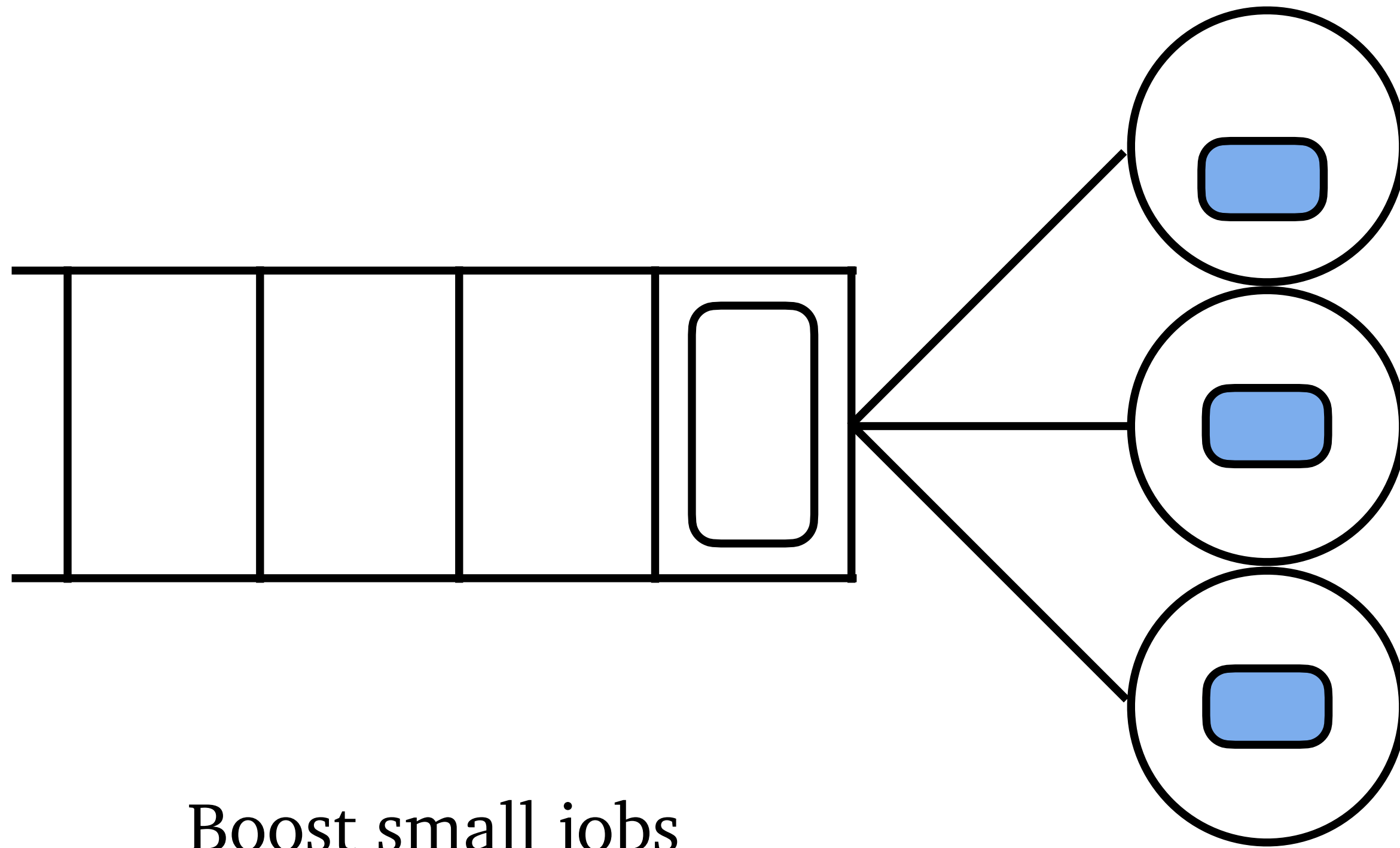


Boost small jobs

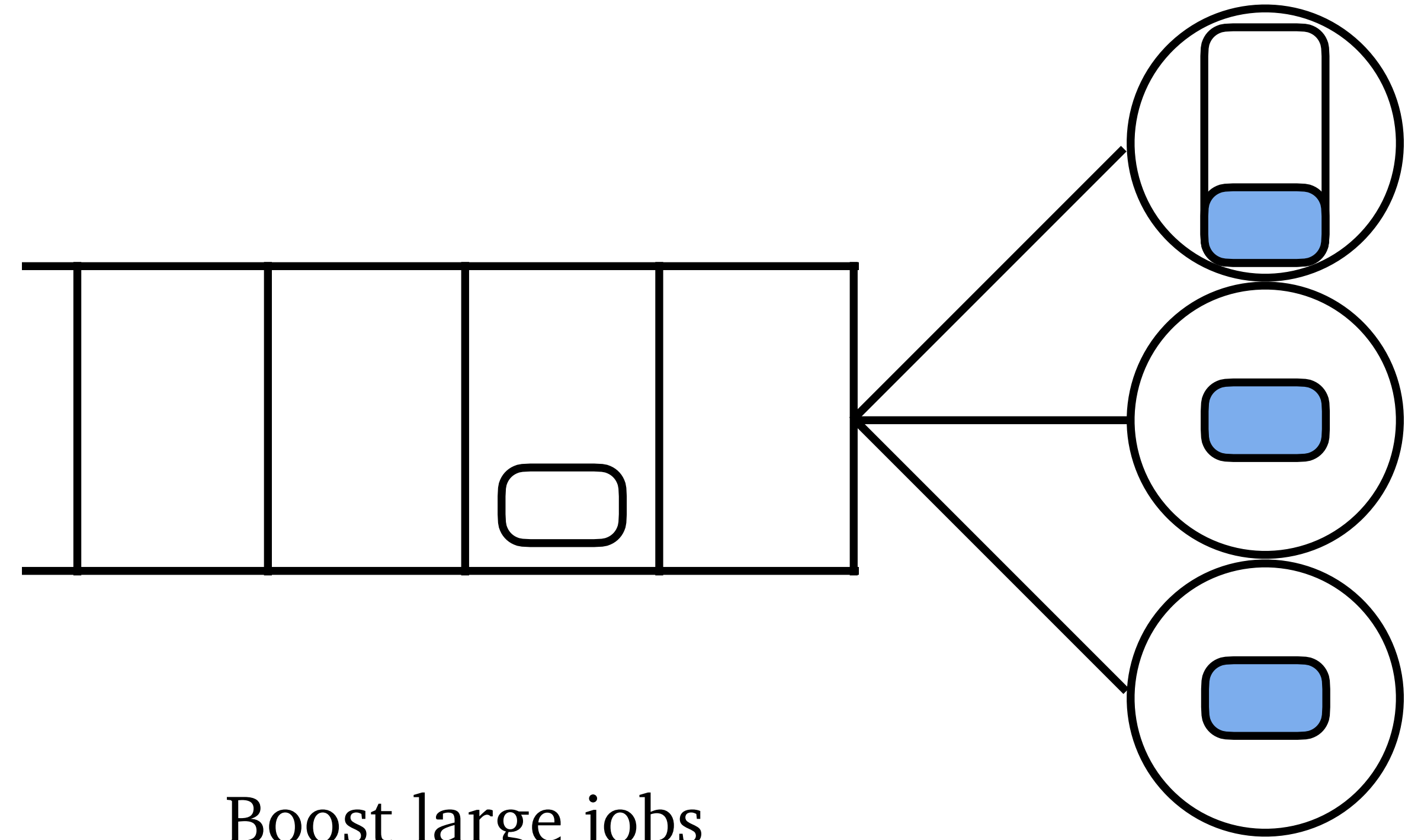


Boost large jobs

Why might SizeBoost make sense at low load?

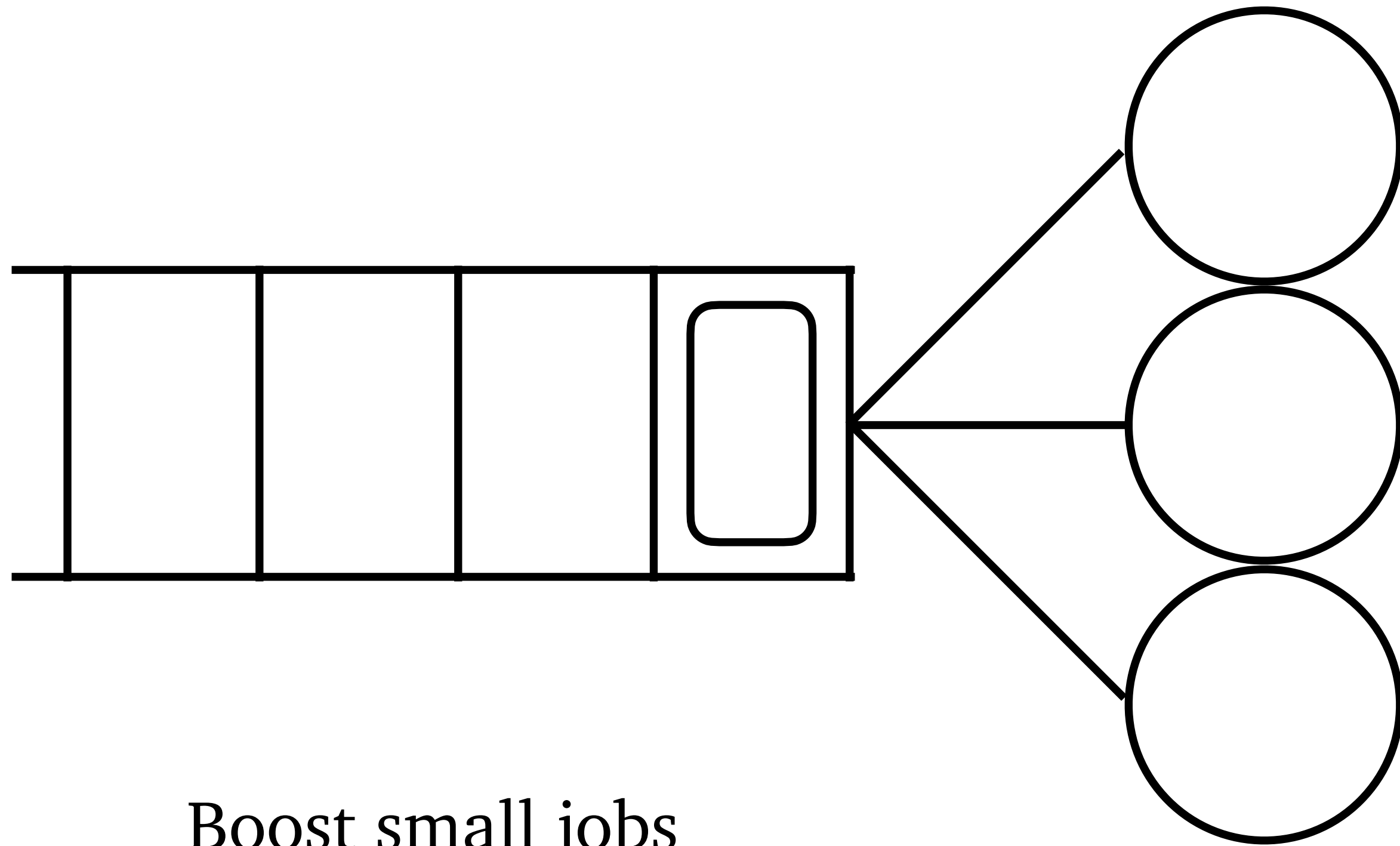


Boost small jobs

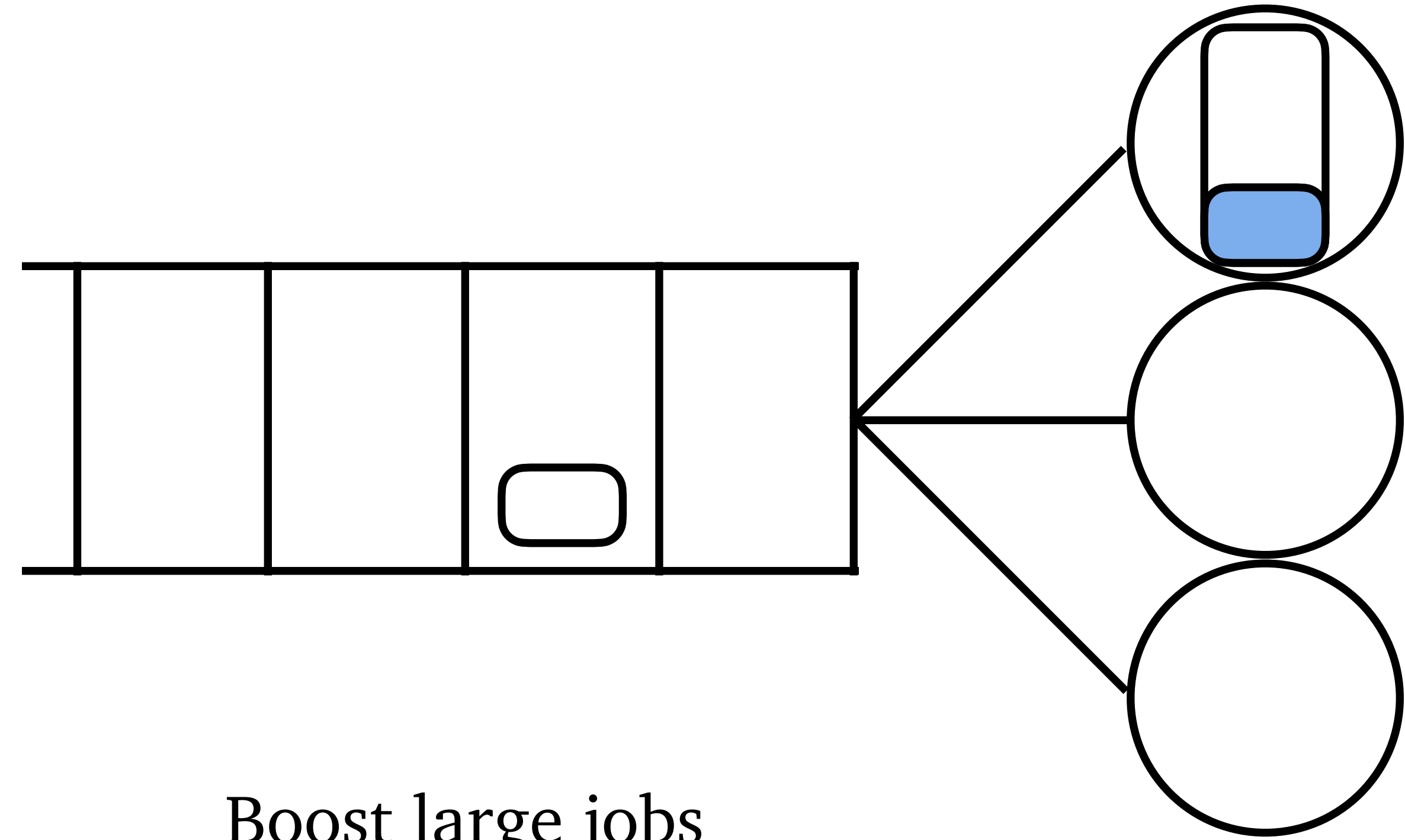


Boost large jobs

Why might SizeBoost make sense at low load?

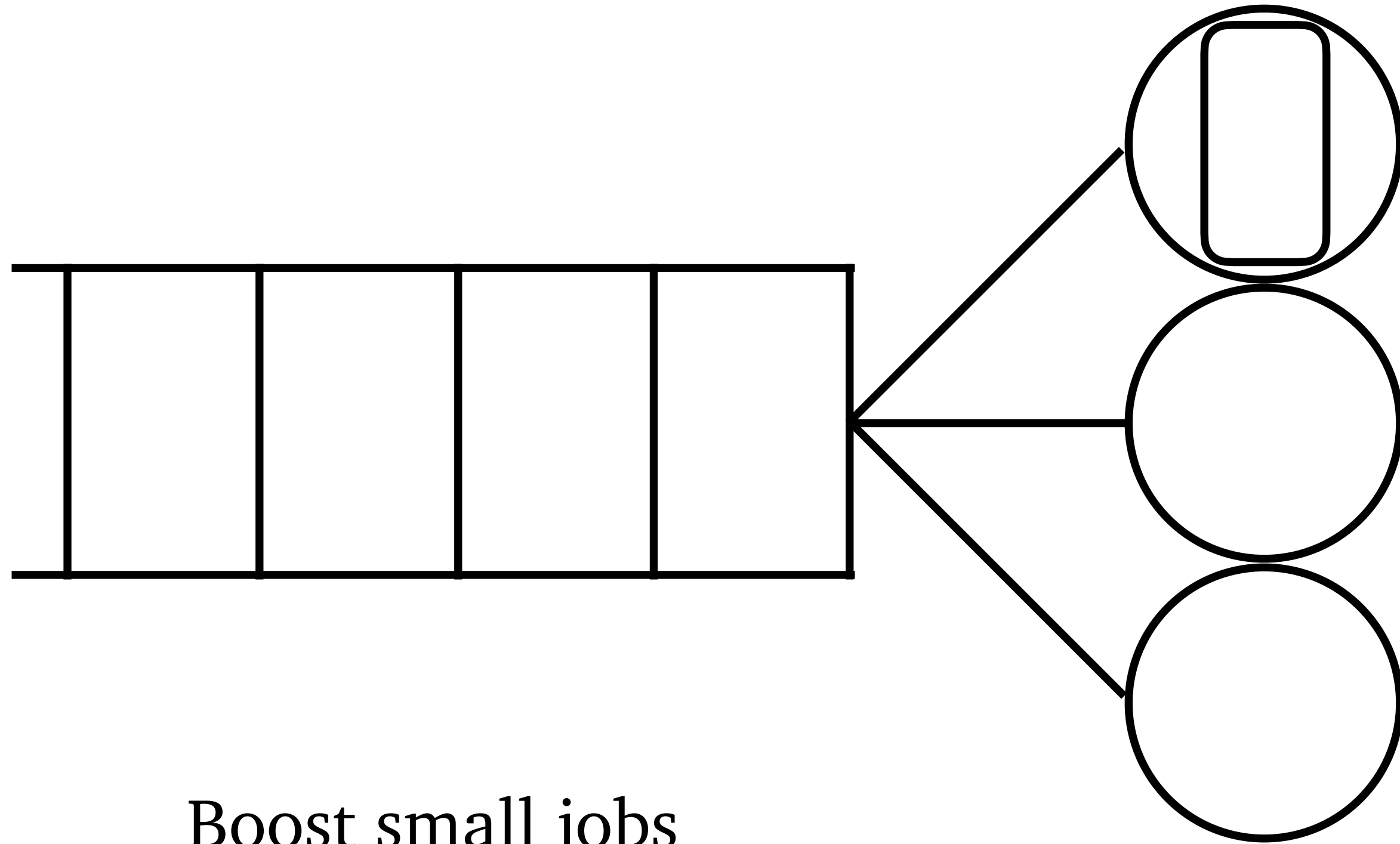


Boost small jobs

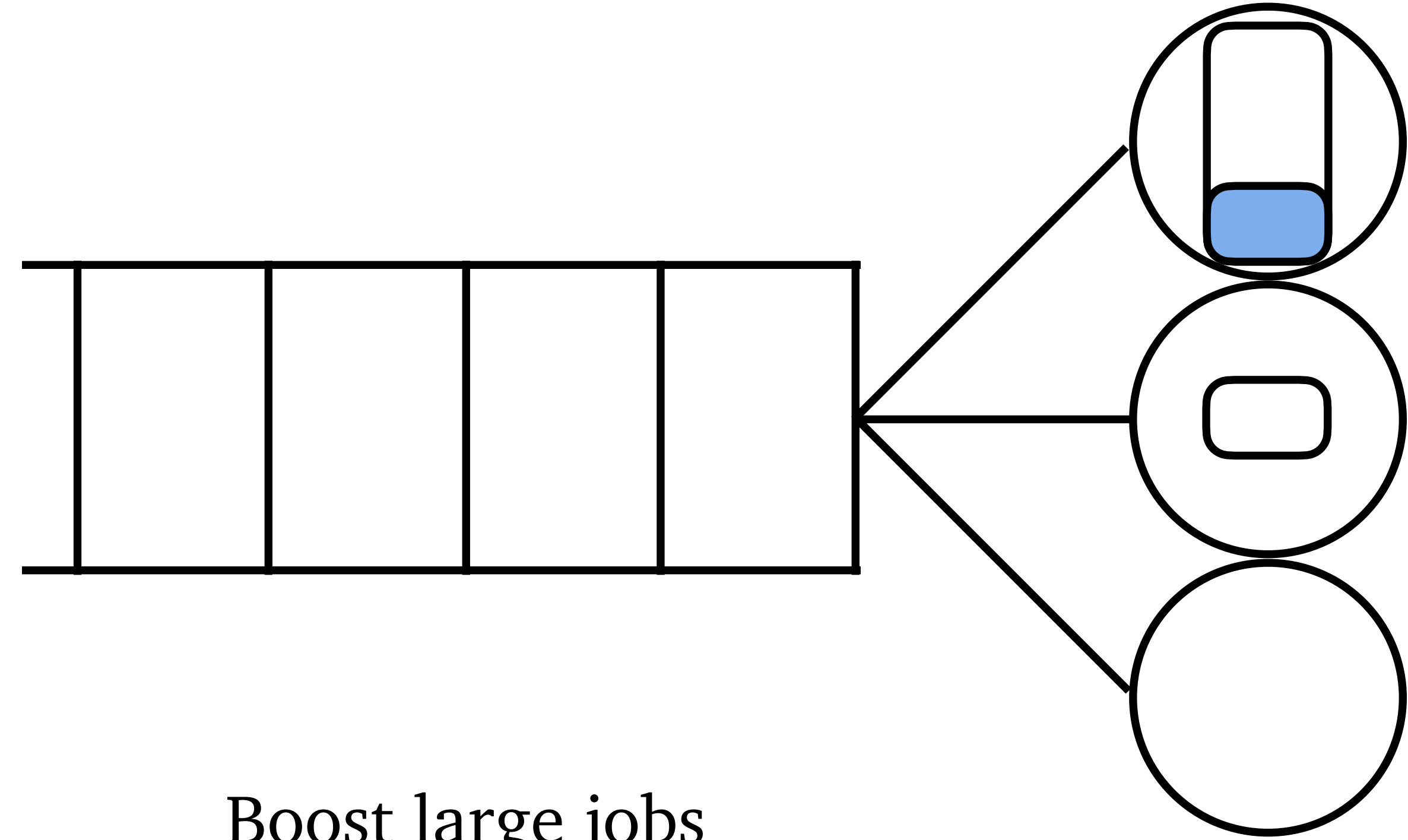


Boost large jobs

Why might SizeBoost make sense at low load?

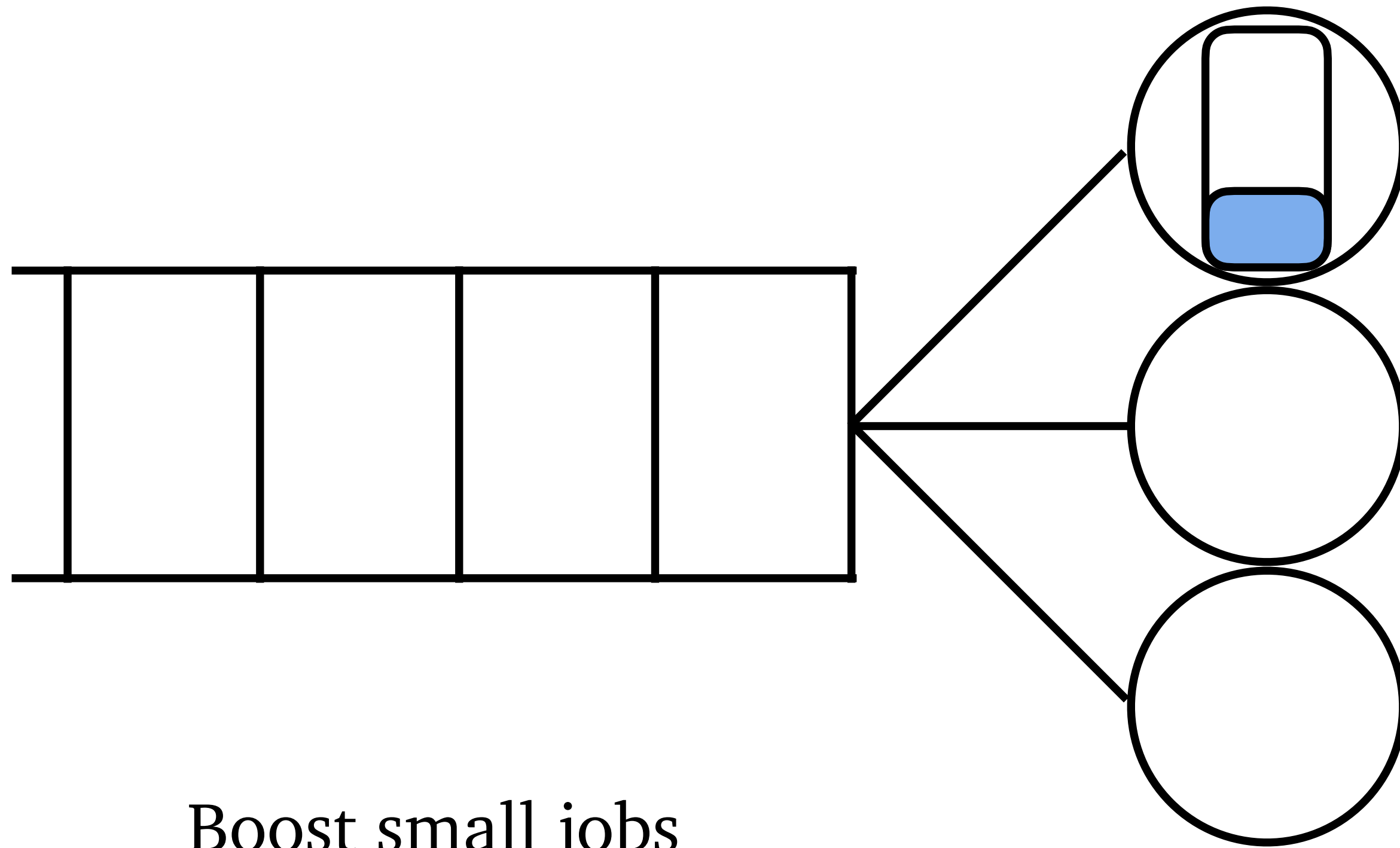


Boost small jobs

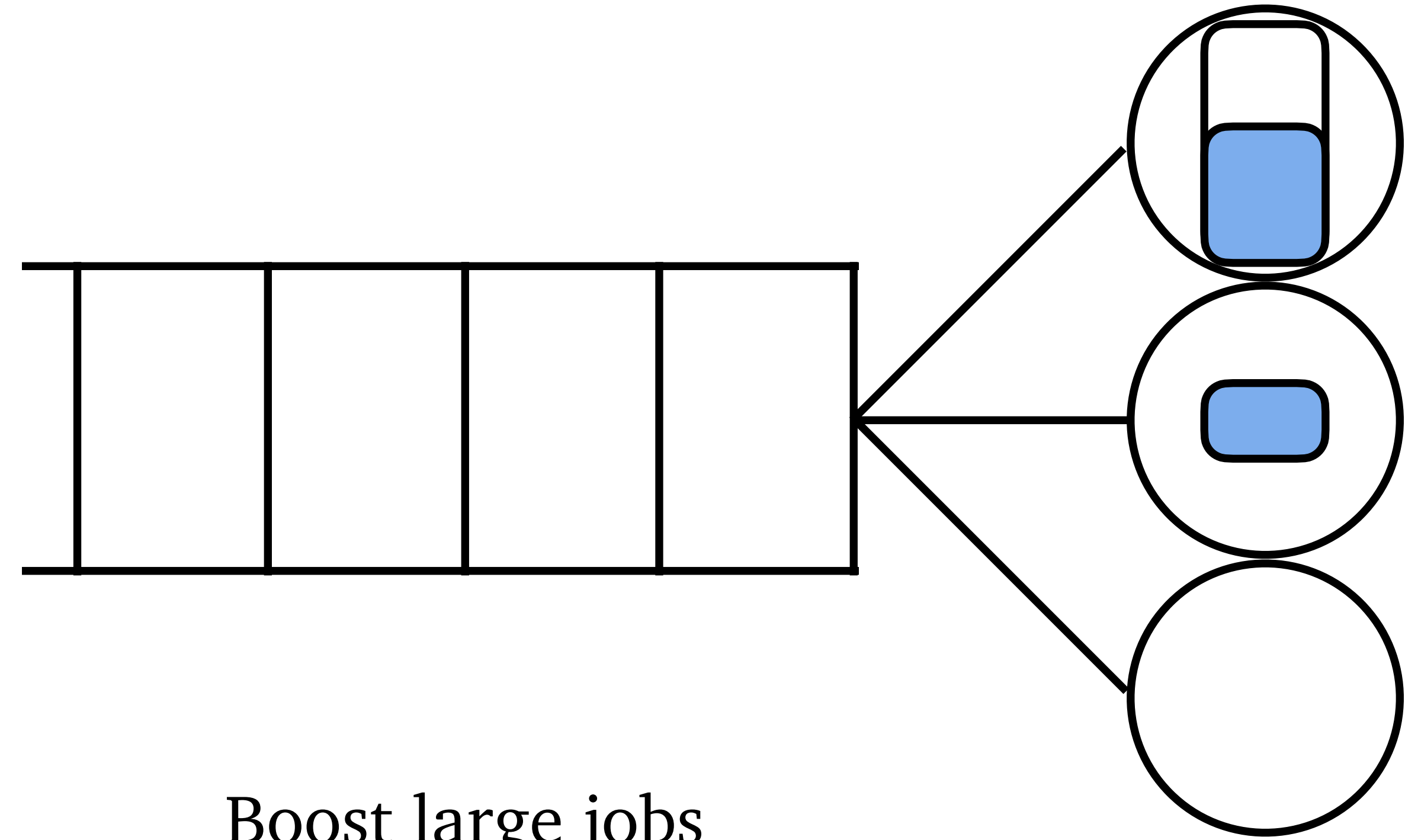


Boost large jobs

Why might SizeBoost make sense at low load?

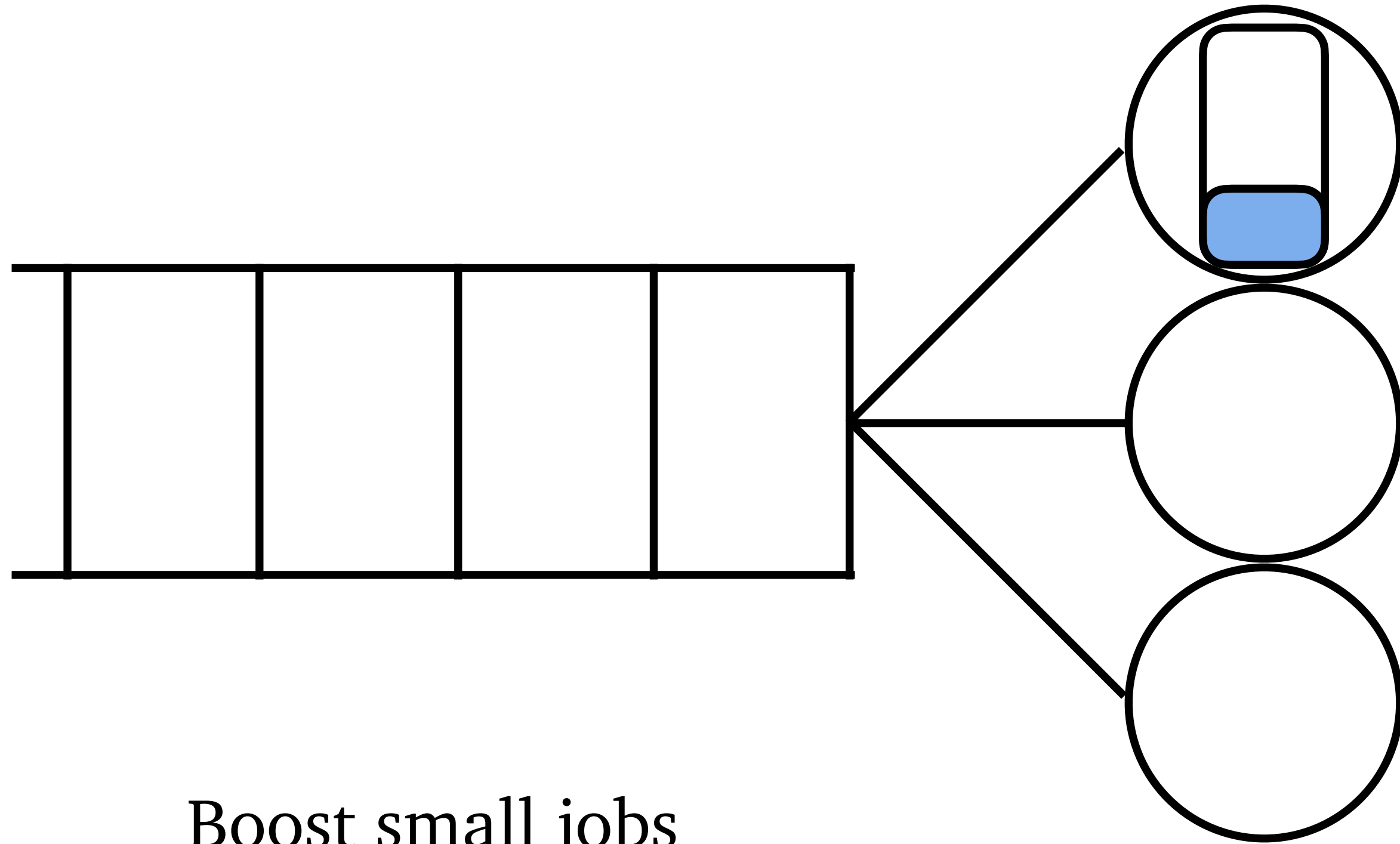


Boost small jobs

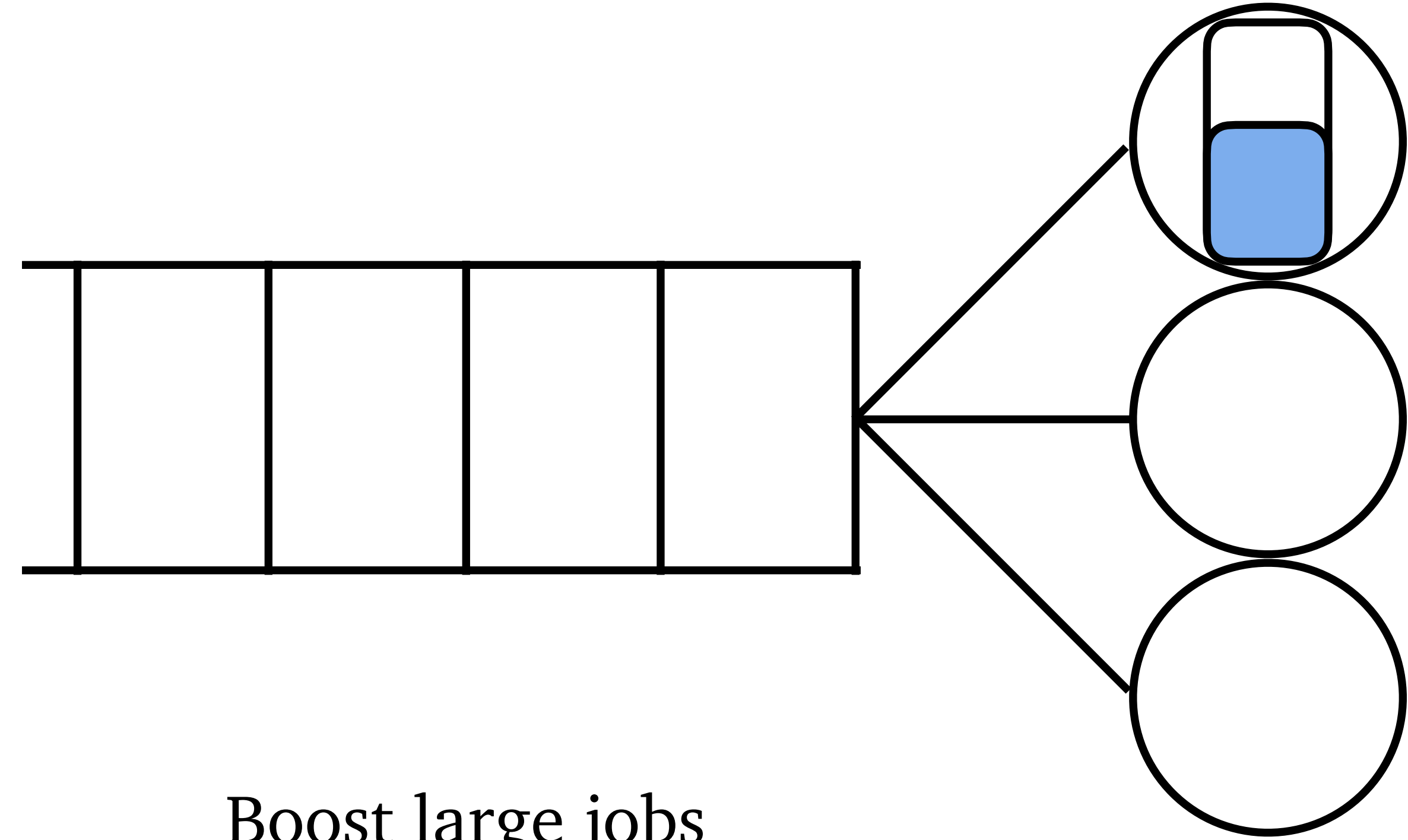


Boost large jobs

Why might SizeBoost make sense at low load?

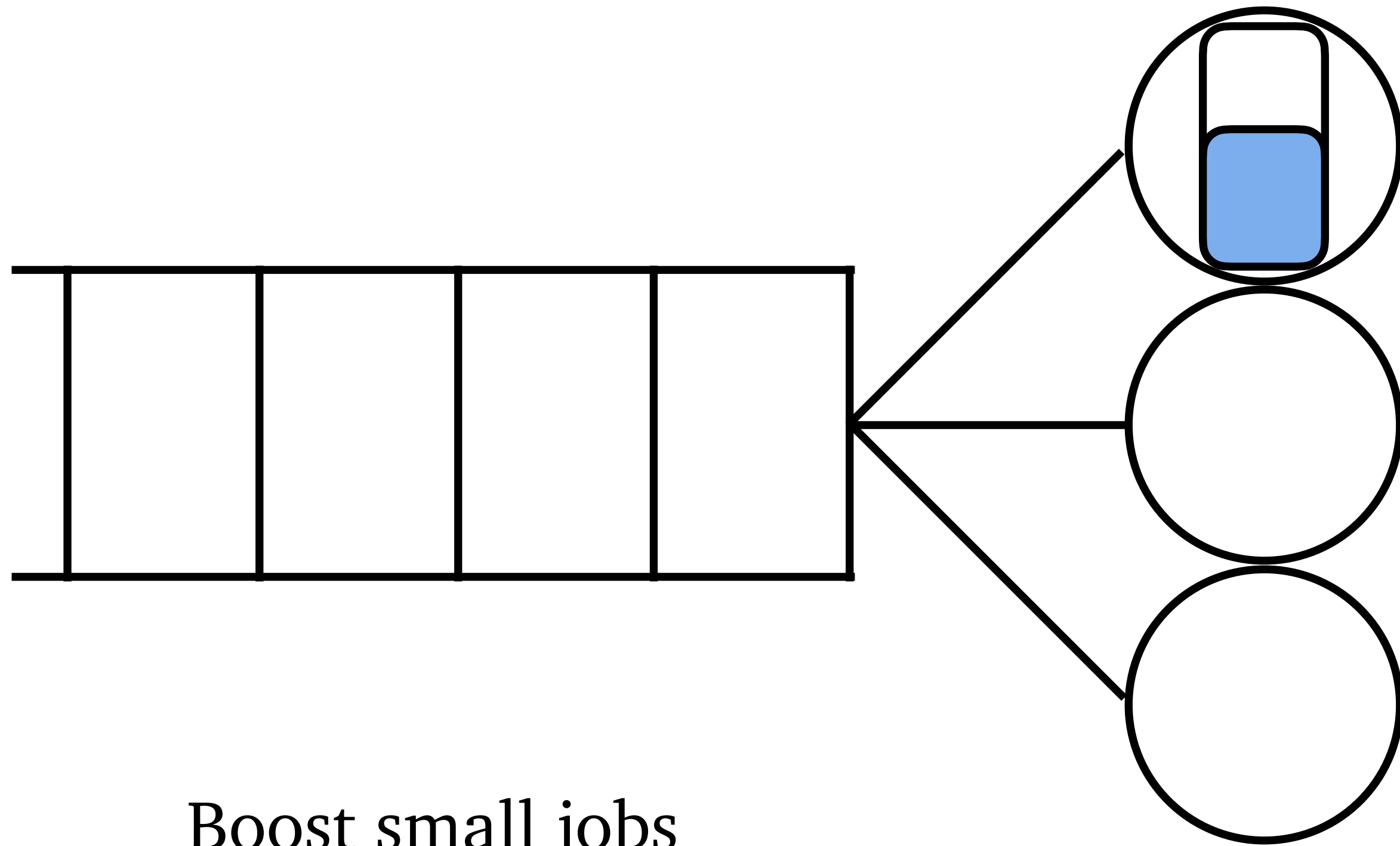


Boost small jobs

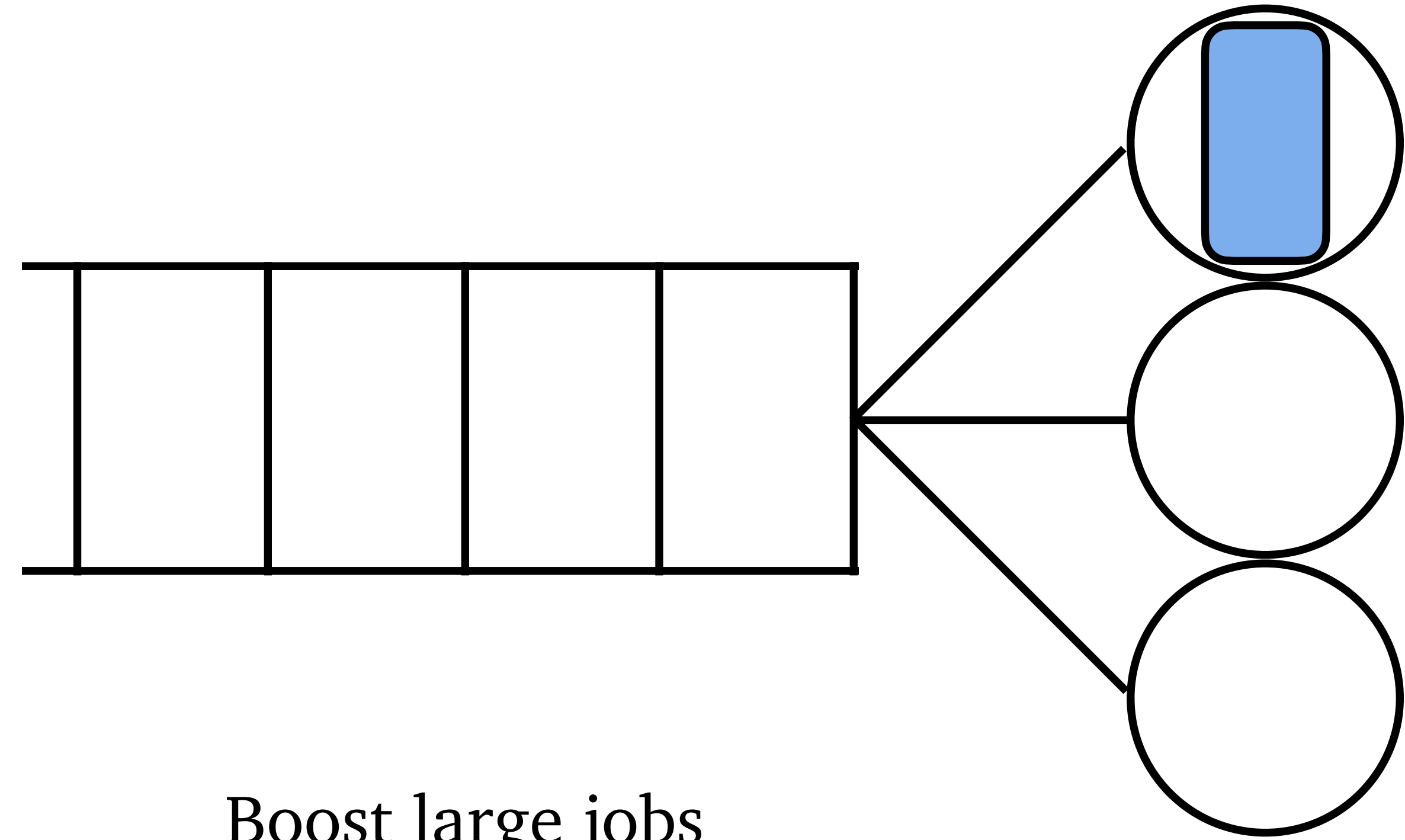


Boost large jobs

Why might SizeBoost make sense at low load?

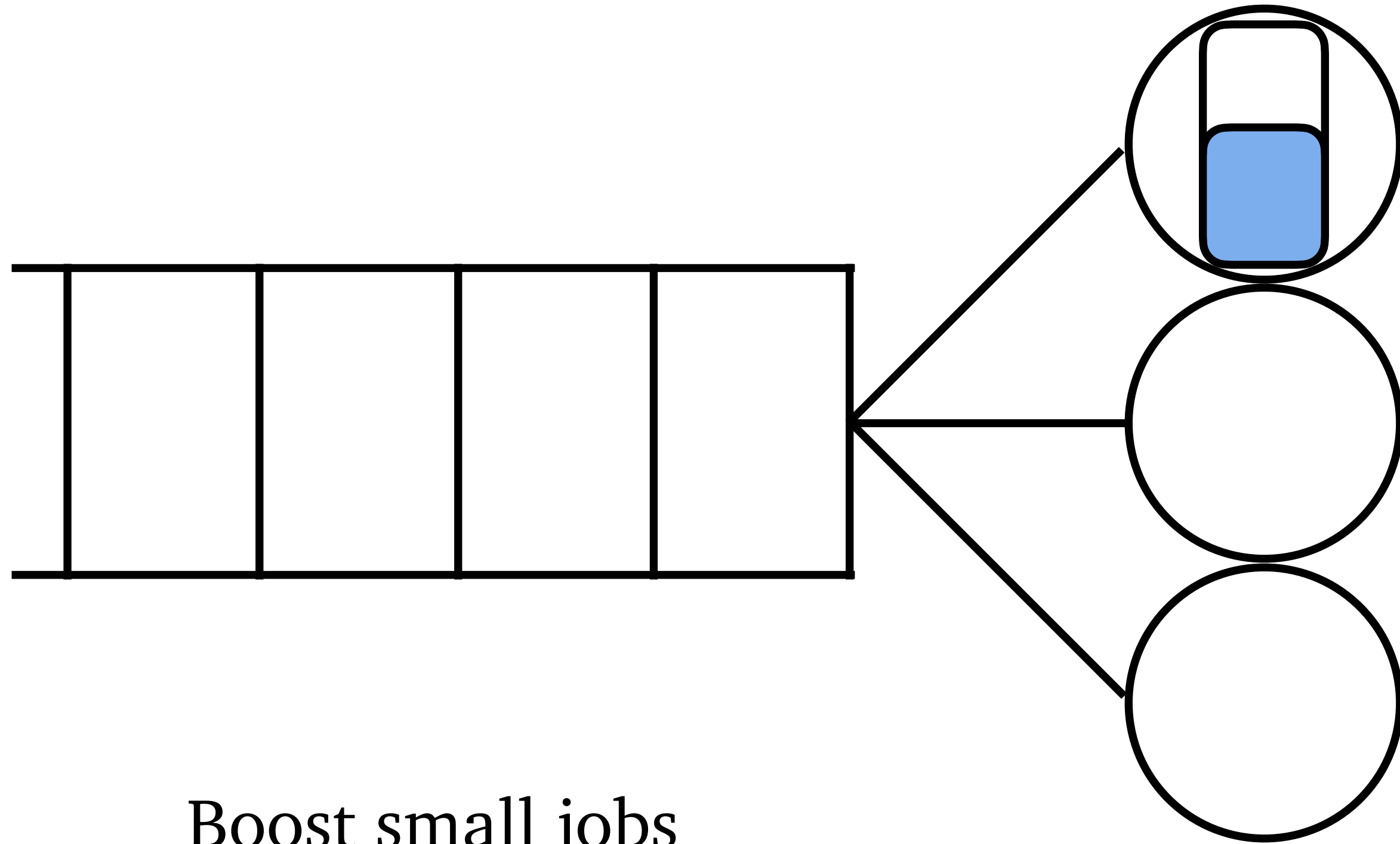


Boost small jobs

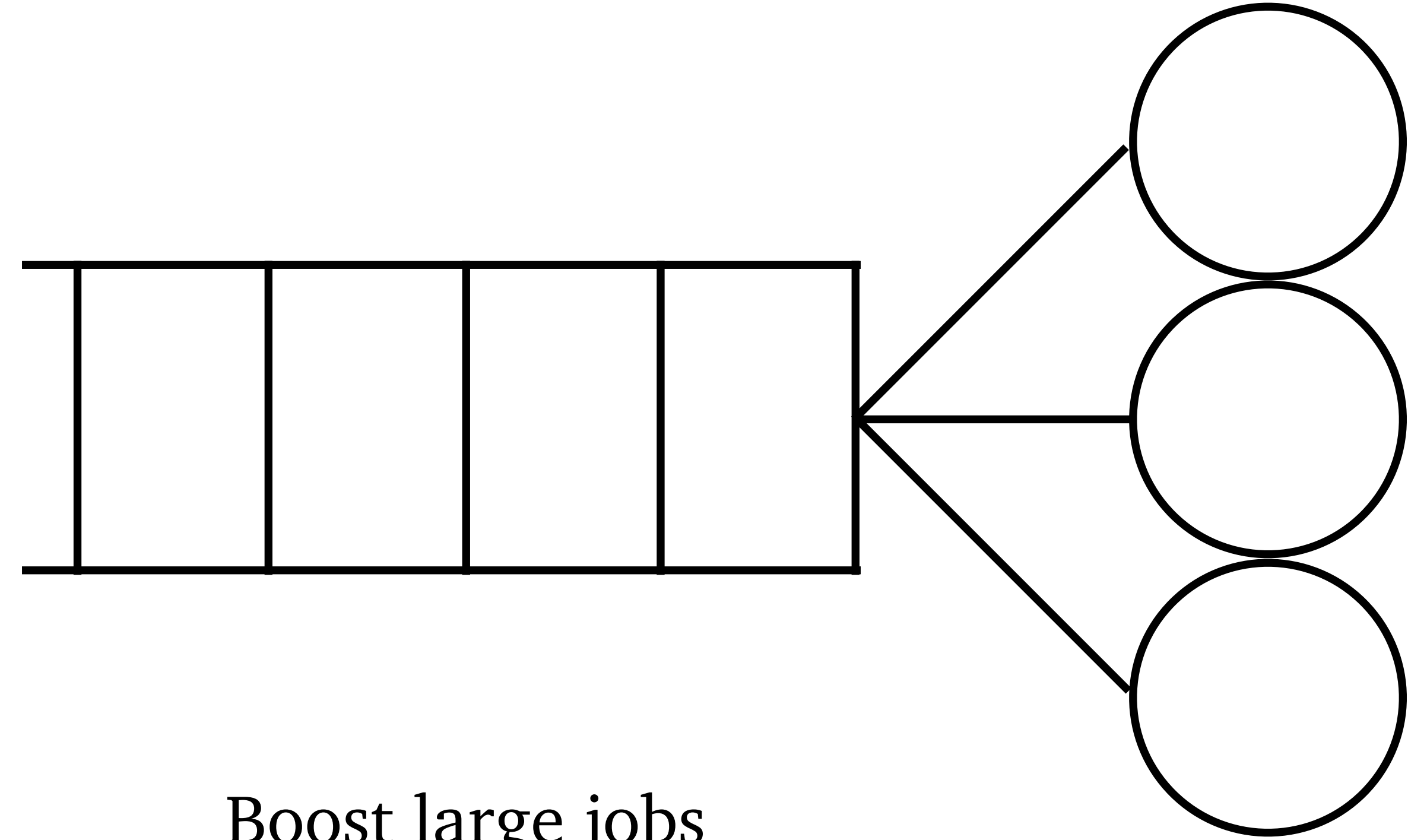


Boost large jobs

Why might SizeBoost make sense at low load?

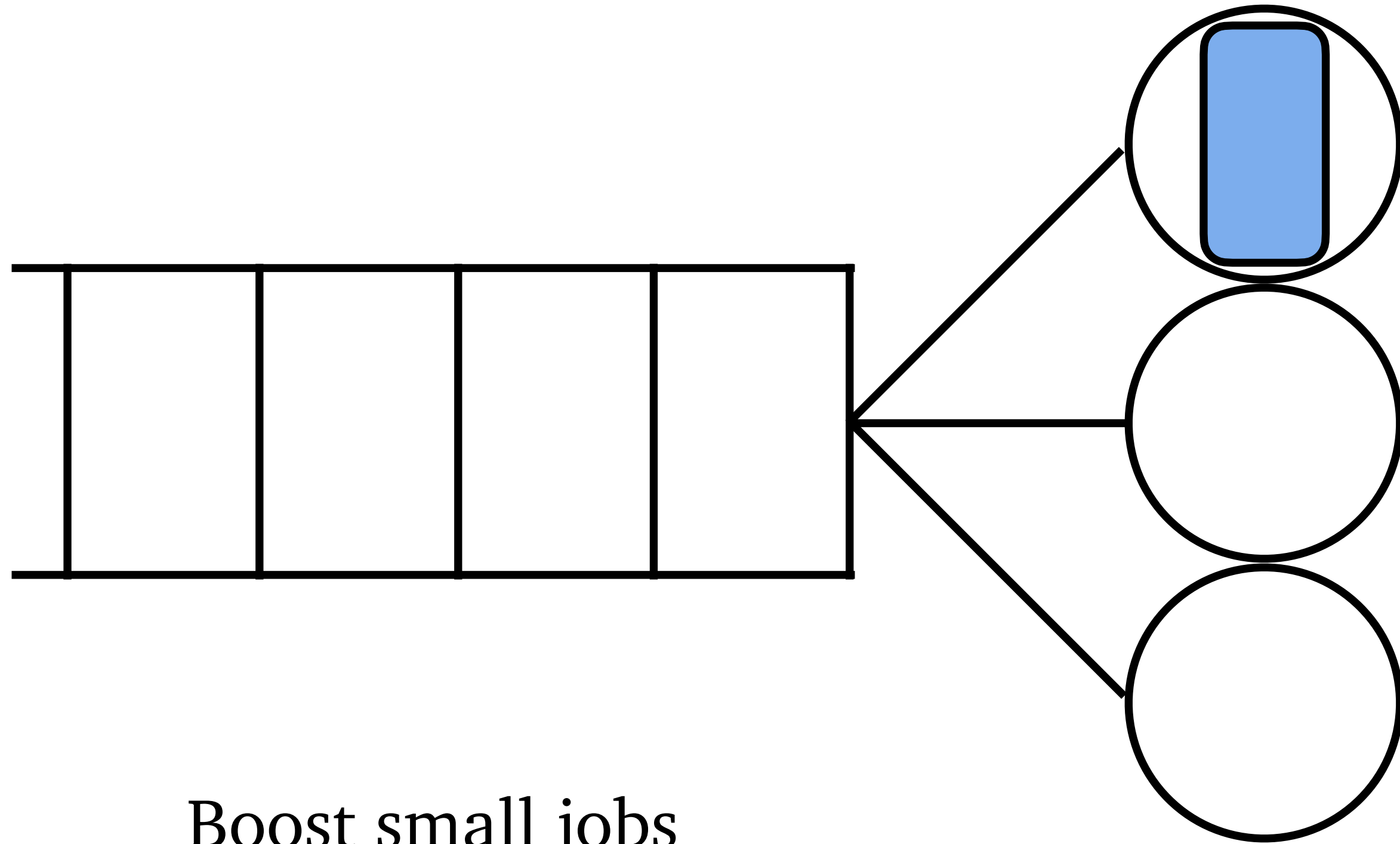


Boost small jobs

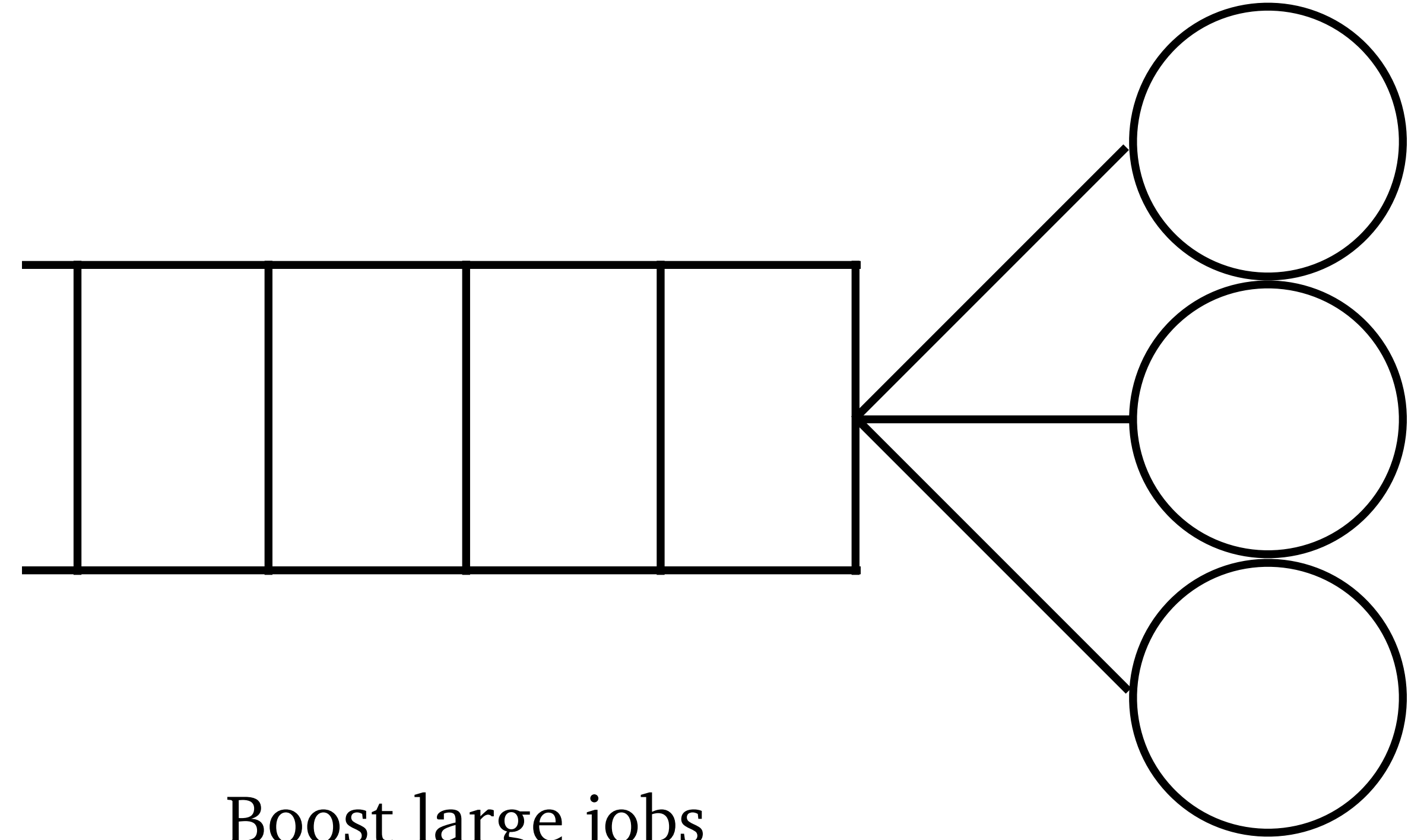


Boost large jobs

Why might SizeBoost make sense at low load?

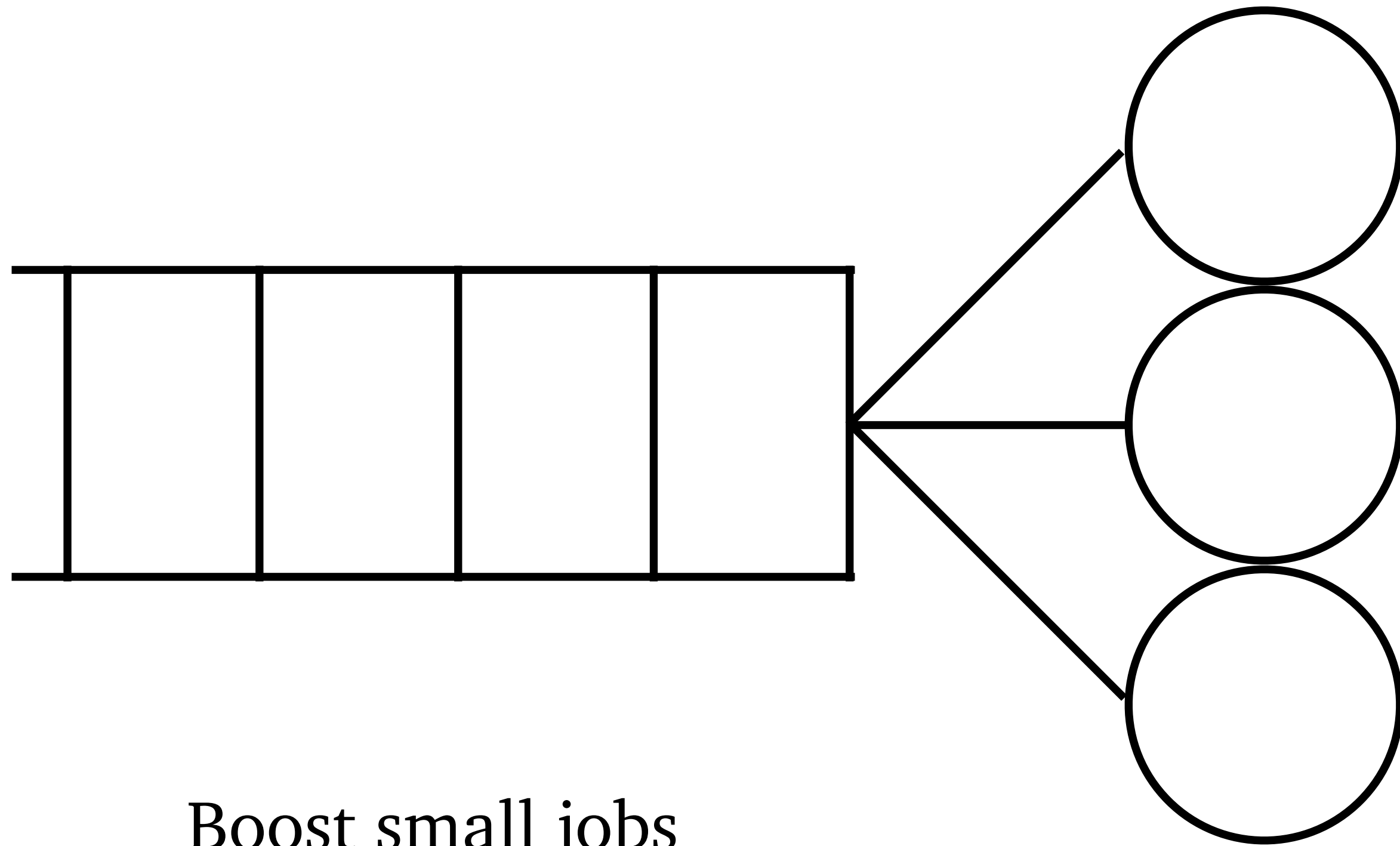


Boost small jobs

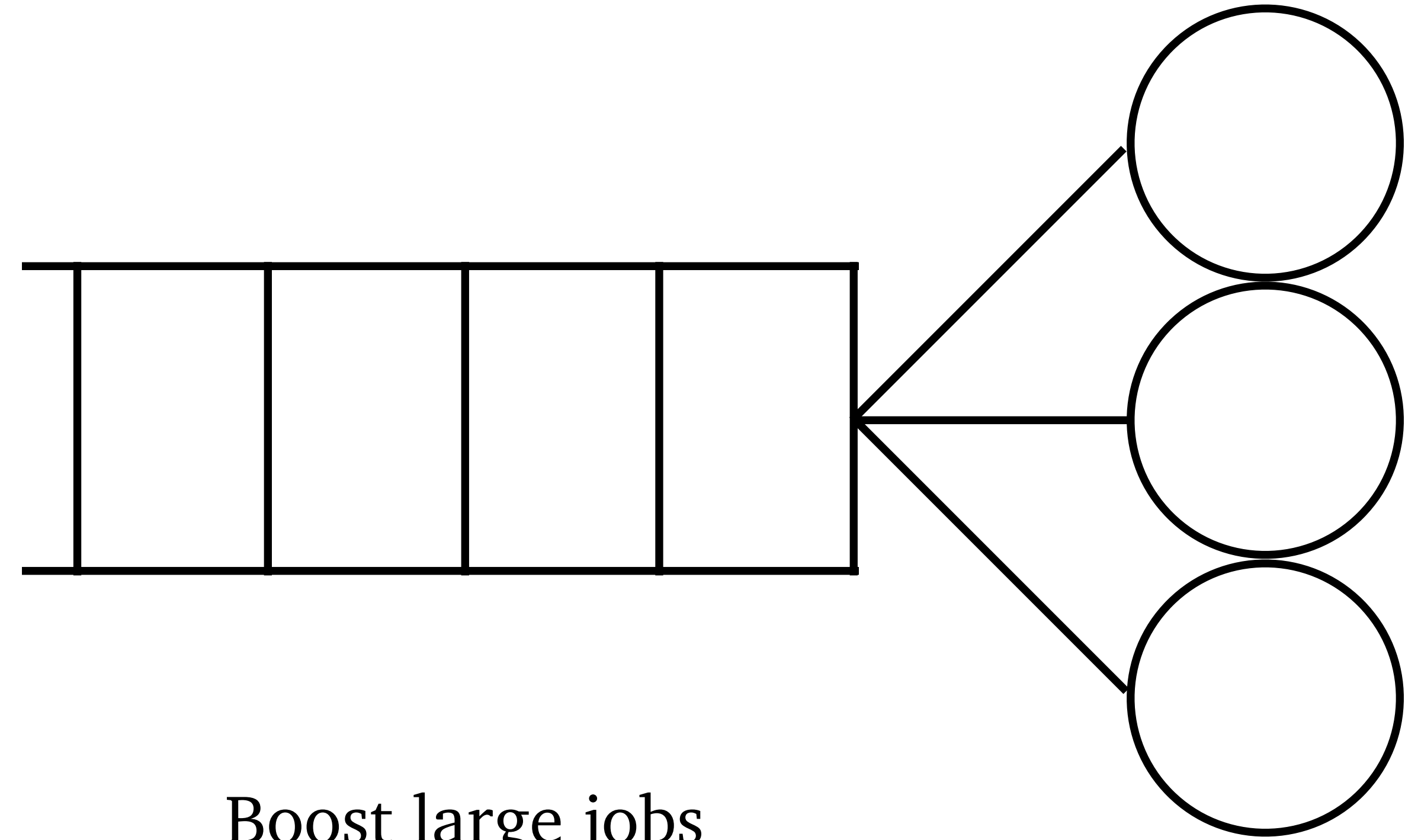


Boost large jobs

Why might SizeBoost make sense at low load?



Boost small jobs

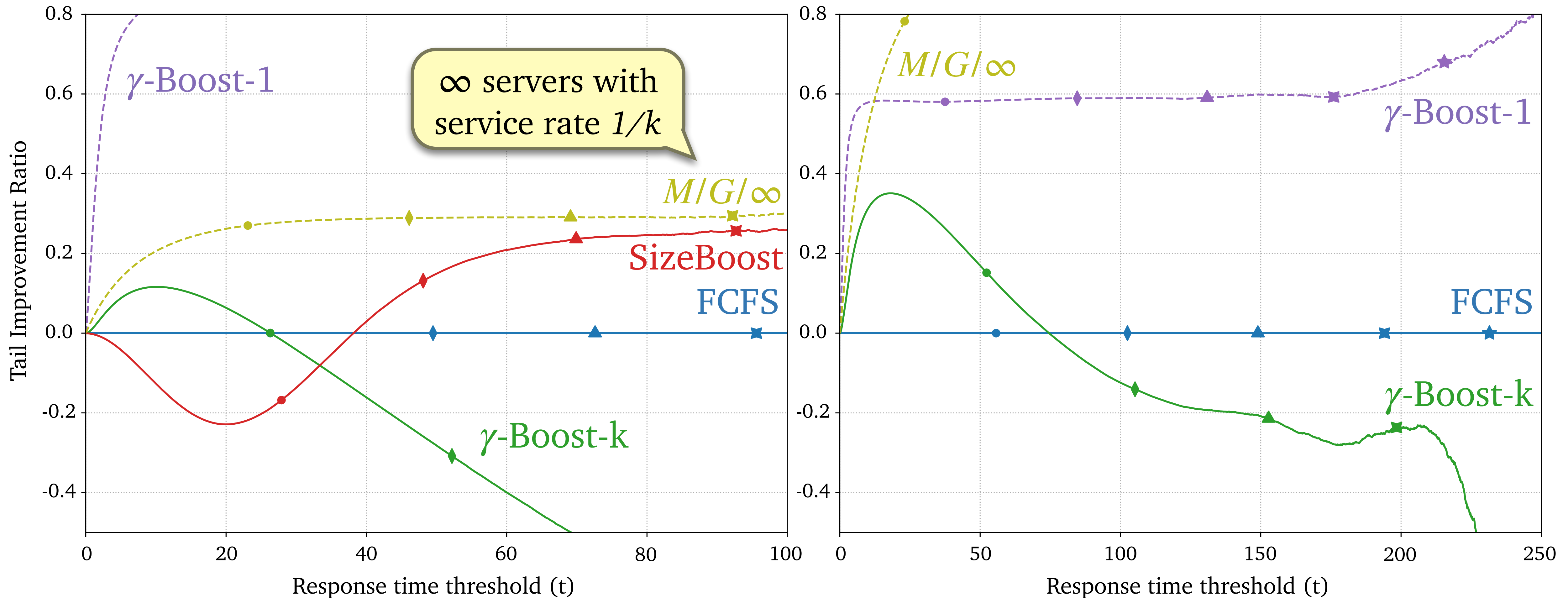


Boost large jobs

γ -Boost performance in lighter traffic

Load: 0.8

Load: 0.95



∞ servers with service rate $1/k$

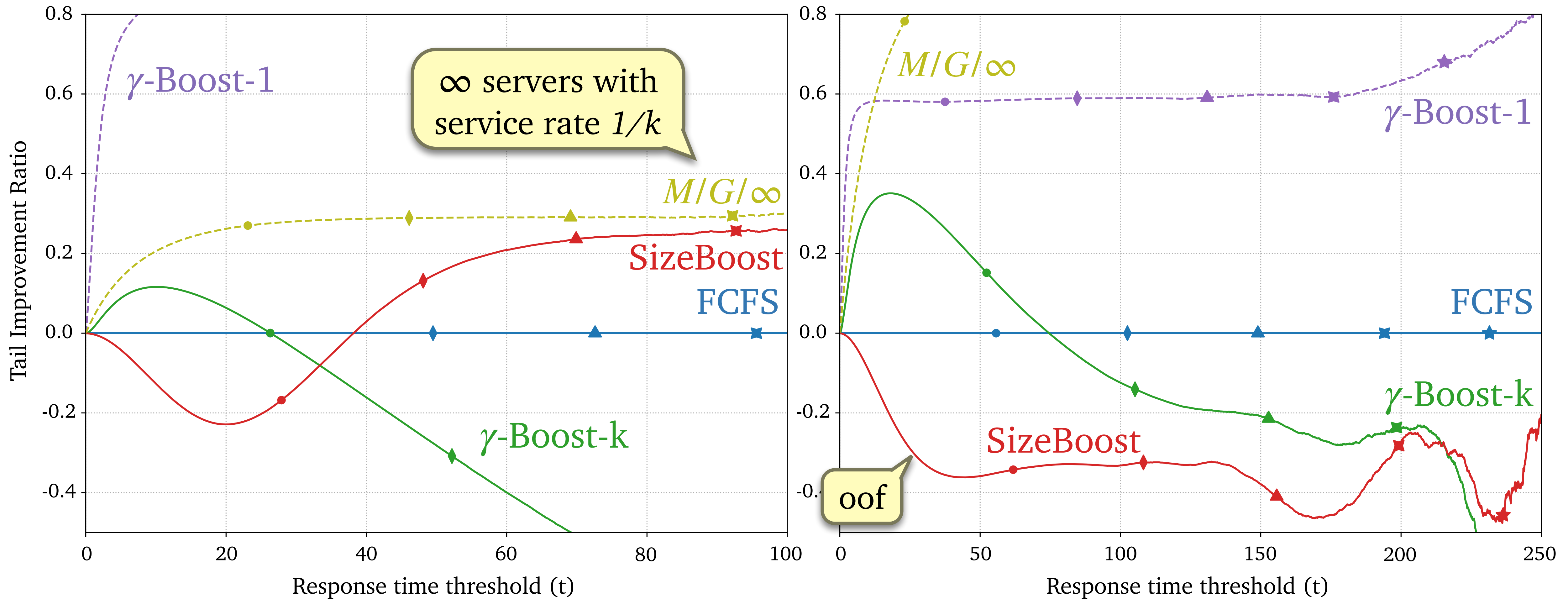
10 servers, Exp(1) job sizes

● 90th percentile
 ◆ 99th percentile
 ▲ 99.9th percentile
 ✱ 99.99th percentile
 ★ 99.999th percentile

γ -Boost performance in lighter traffic

Load: 0.8

Load: 0.95



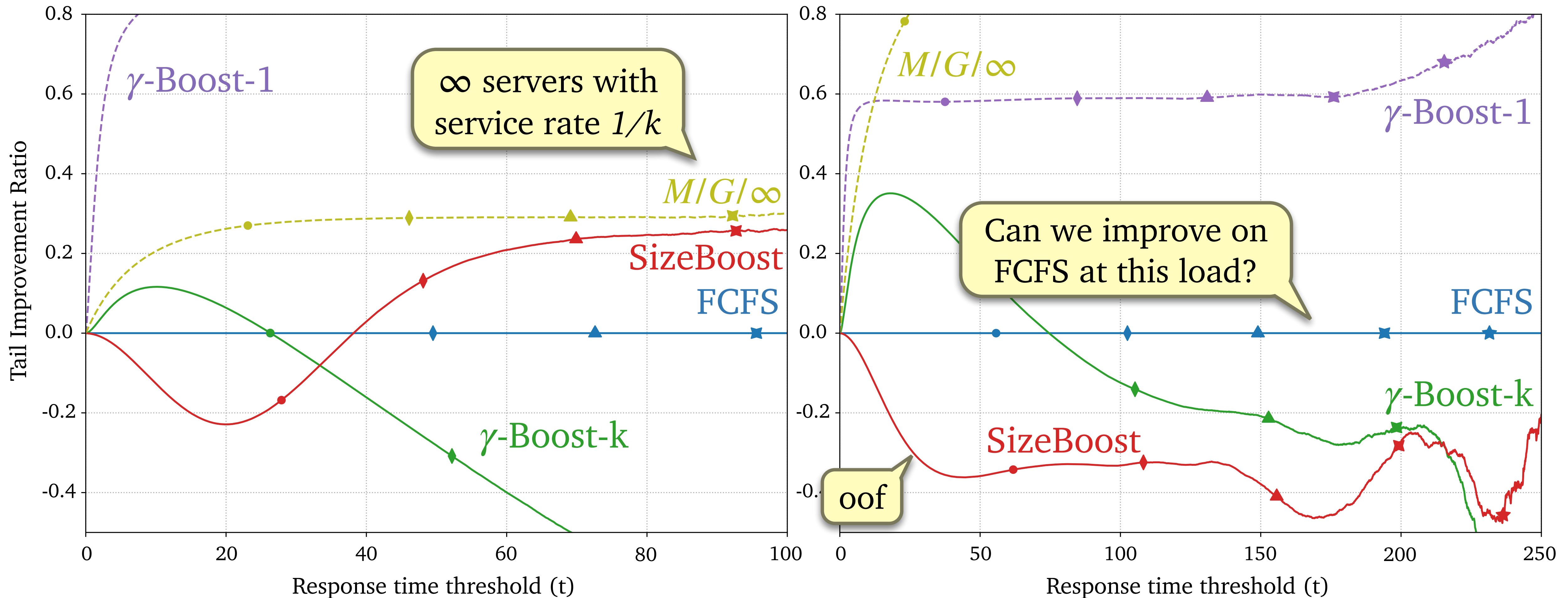
10 servers,
Exp(1) job sizes

90th percentile
 99th percentile
 99.9th percentile
 99.99th percentile
 99.999th percentile
 18

γ -Boost performance in lighter traffic

Load: 0.8

Load: 0.95



∞ servers with service rate $1/k$

Can we improve on FCFS at this load?

oof

- 90th percentile
- ◆ 99th percentile
- ▲ 99.9th percentile
- 99.99th percentile
- ★ 99.999th percentile

Where does γ -Boost come from?

Where does γ -Boost come from?

Queueing Problem

$$\text{minimize } C_\pi = \lim_{t \rightarrow \infty} e^{\gamma t} \mathbf{P}[T_\pi > t] = \lim_{\theta \rightarrow \gamma} \frac{\gamma - \theta}{\gamma} \mathbf{E}[e^{\theta T_\pi}]$$

(by final value theorem
for Laplace transforms)

Where does γ -Boost come from?

Queueing Problem

$$\text{minimize } C_\pi = \lim_{t \rightarrow \infty} e^{\gamma t} \mathbf{P}[T_\pi > t] = \lim_{\theta \rightarrow \gamma} \underbrace{\frac{\gamma - \theta}{\gamma} \mathbf{E}[e^{\theta T_\pi}]}_{\text{"0} \cdot \mathbf{E}[e^{\gamma T_\pi}]}$$

(by final value theorem
for Laplace transforms)

Where does γ -Boost come from?

Queueing Problem

$$\text{minimize } C_\pi = \lim_{t \rightarrow \infty} e^{\gamma t} \mathbf{P}[T_\pi > t] = \lim_{\theta \rightarrow \gamma} \underbrace{\frac{\gamma - \theta}{\gamma} \mathbf{E}[e^{\theta T_\pi}]}_{\text{"0} \cdot \mathbf{E}[e^{\gamma T_\pi}]}$$

(by final value theorem
for Laplace transforms)

Batch Problem

$$\text{minimize } \frac{1}{n} \sum_{i=1}^n e^{\gamma t_i}$$

minimized by γ -Boost in M/G/1

Where does γ -Boost come from?

Queueing Problem

$$\text{minimize } C_\pi = \lim_{t \rightarrow \infty} e^{\gamma t} \mathbf{P}[T_\pi > t] = \lim_{\theta \rightarrow \gamma} \underbrace{\frac{\gamma - \theta}{\gamma} \mathbf{E}[e^{\theta T_\pi}]}_{\text{"0} \cdot \mathbf{E}[e^{\gamma T_\pi}]}$$

(by final value theorem for Laplace transforms)

Batch Problem

$$\text{minimize } \frac{1}{n} \sum_{i=1}^n e^{\gamma t_i}$$

minimized by γ -Boost in M/G/1

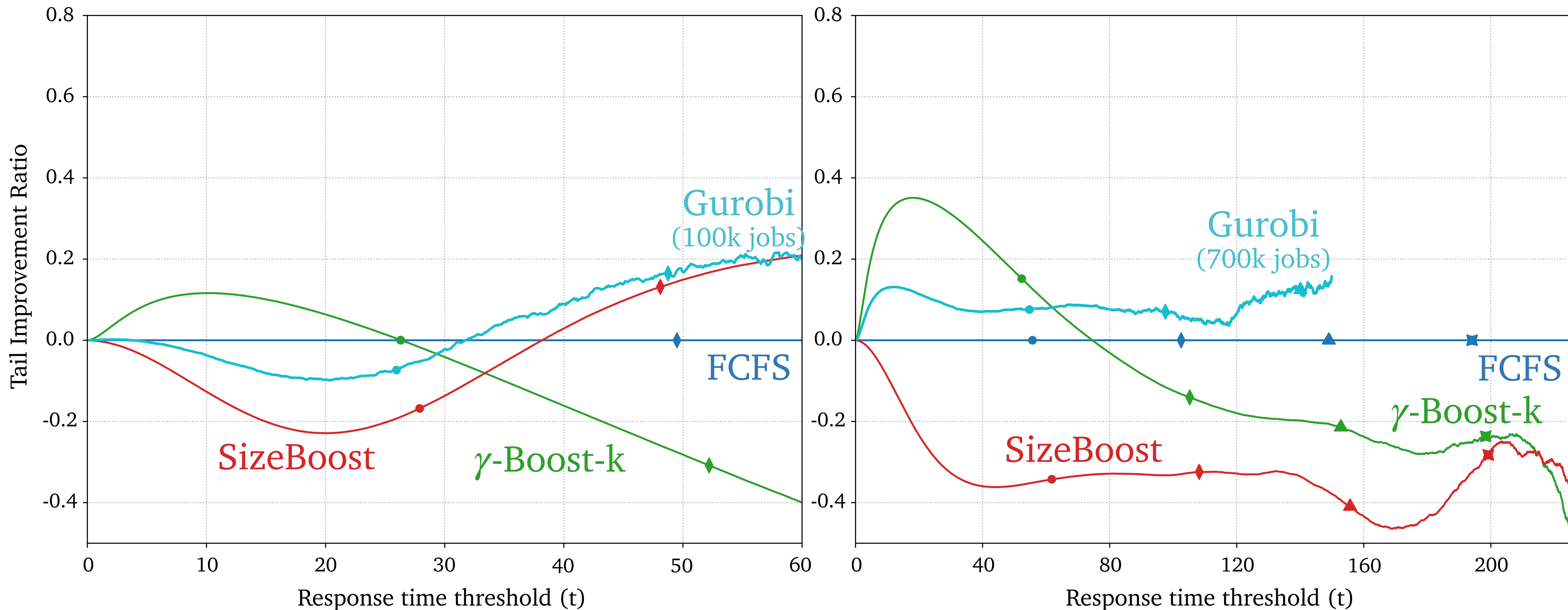


Numerically optimize $\frac{1}{n} \sum_{i=1}^n e^{\gamma t_i}$
in M/G/k after each arrival.

Numerically optimizing the batch problem in the M/G/k

Load: 0.8

Load: 0.95



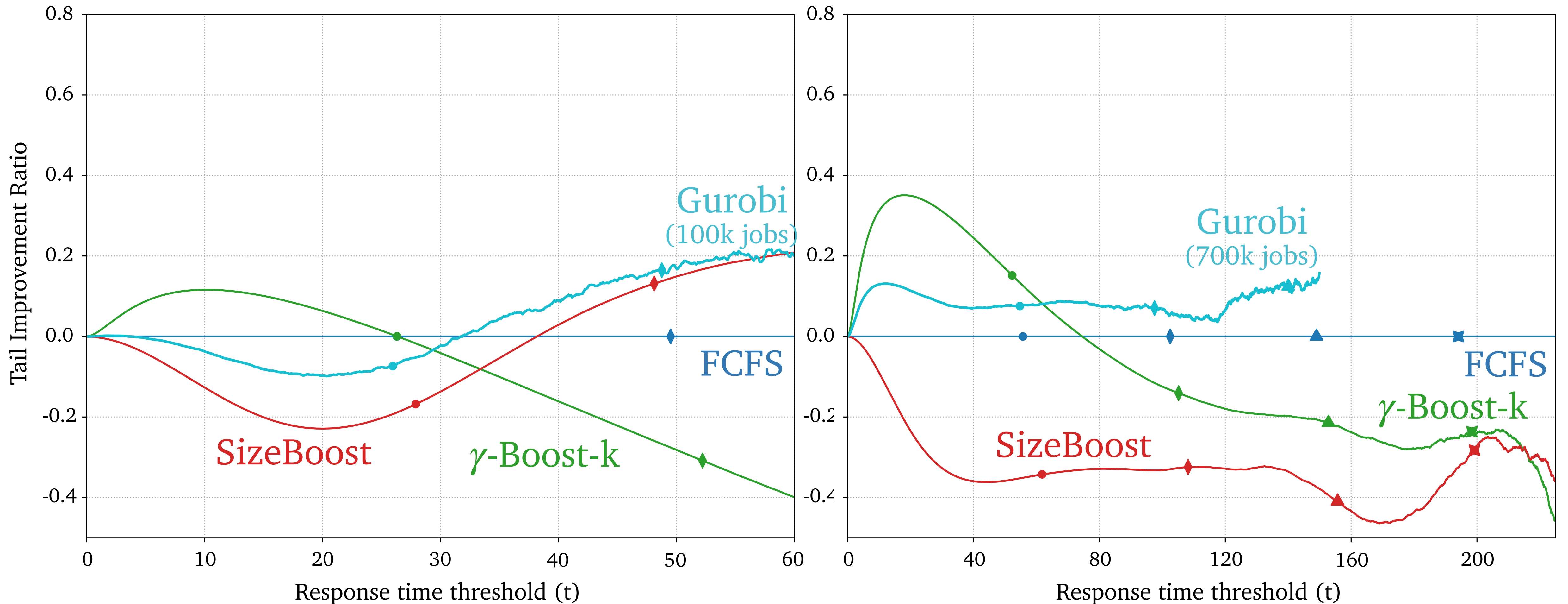
10 servers,
Exp(1) job sizes

● 90th percentile ◆ 99th percentile ▲ 99.9th percentile ✦ 99.99th percentile ★ 99.999th percentile 20

One week before submission...

Load: 0.8

Load: 0.95



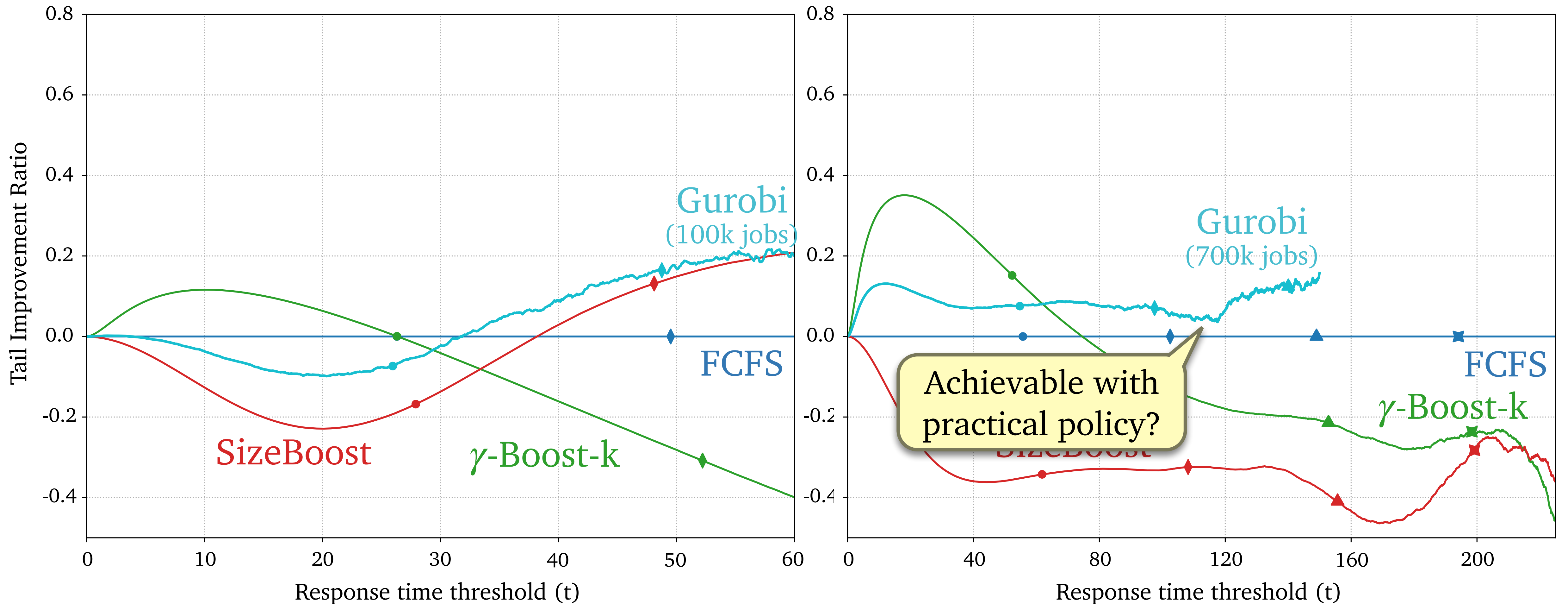
10 servers,
Exp(1) job sizes



One week before submission...

Load: 0.8

Load: 0.95



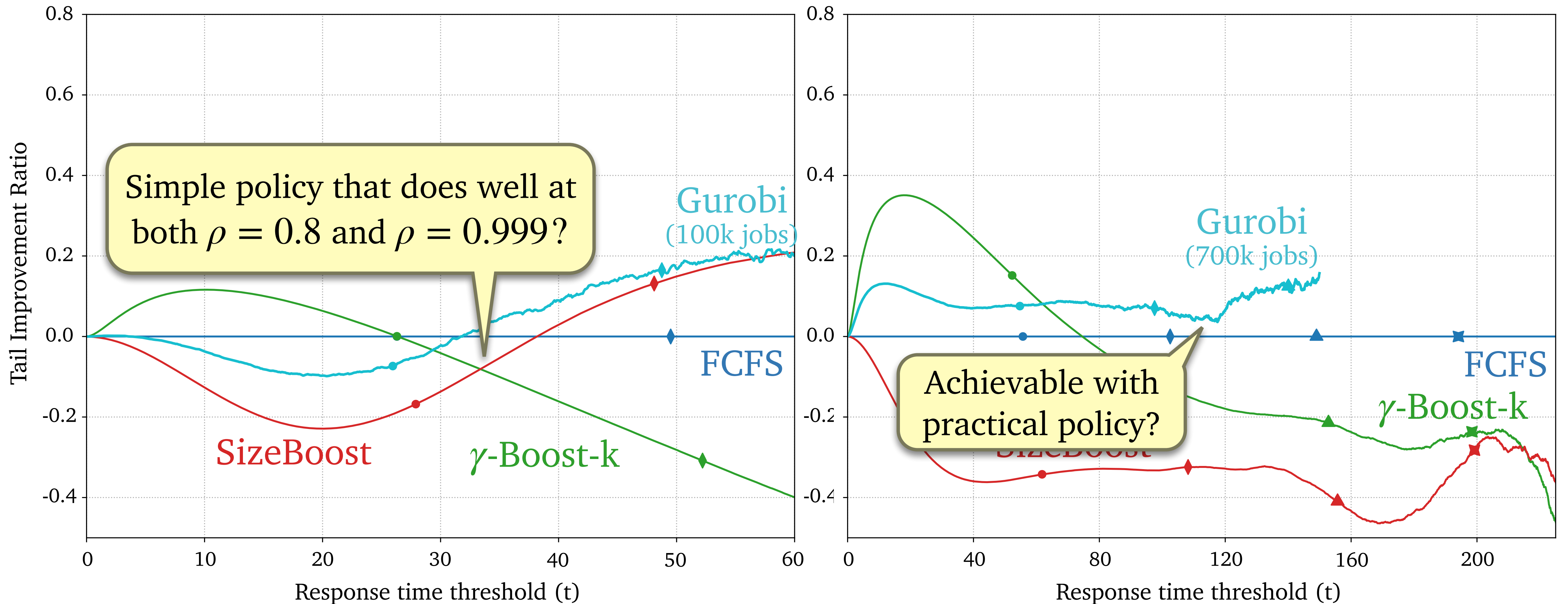
10 servers,
Exp(1) job sizes



One week before submission...

Load: 0.8

Load: 0.95



Simple policy that does well at both $\rho = 0.8$ and $\rho = 0.999$?

Achievable with practical policy?

• 90th percentile ◆ 99th percentile ▲ 99.9th percentile ✦ 99.99th percentile ★ 99.999th percentile

10 servers, Exp(1) job sizes

Two days before submission...

Two days before submission...



Dr. Ralph Scully
(Professor at Harvard Med)

Two days before submission...



Dr. Ralph Scully
(Professor at Harvard Med)

a.k.a. Ziv's dad

Two days before submission...

*Is there some way to combine
 γ -Boost and SizeBoost?*



Dr. Ralph Scully
(Professor at Harvard Med)

a.k.a. Ziv's dad

Two days before submission...

*Is there some way to combine
 γ -Boost and SizeBoost?*



Dr. Ralph Scully
(Professor at Harvard Med)

a.k.a. Ziv's dad

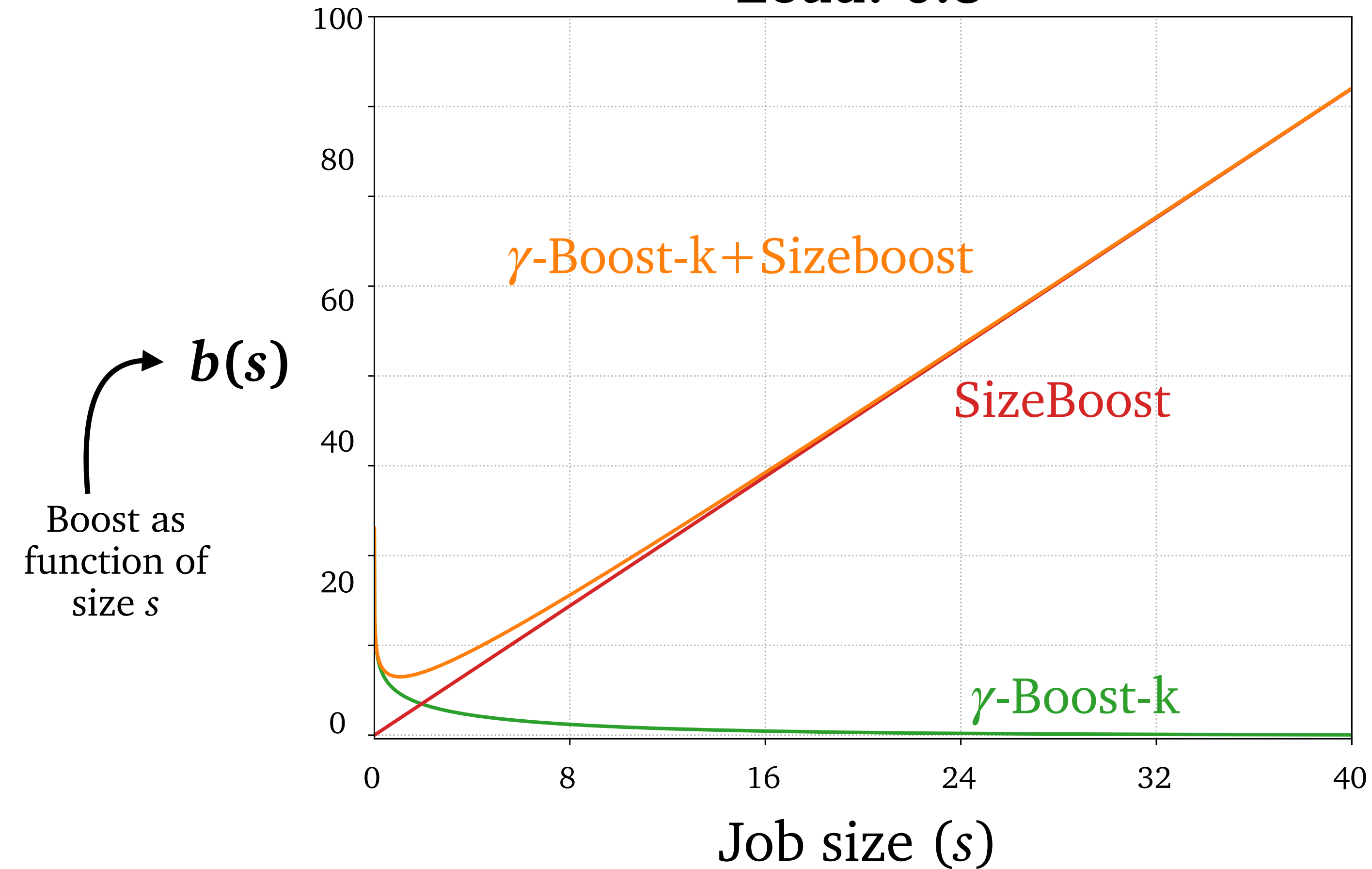


Add the boosts from
both policies together!

Why does adding the boosts make any sense?

Why does adding the boosts make any sense?

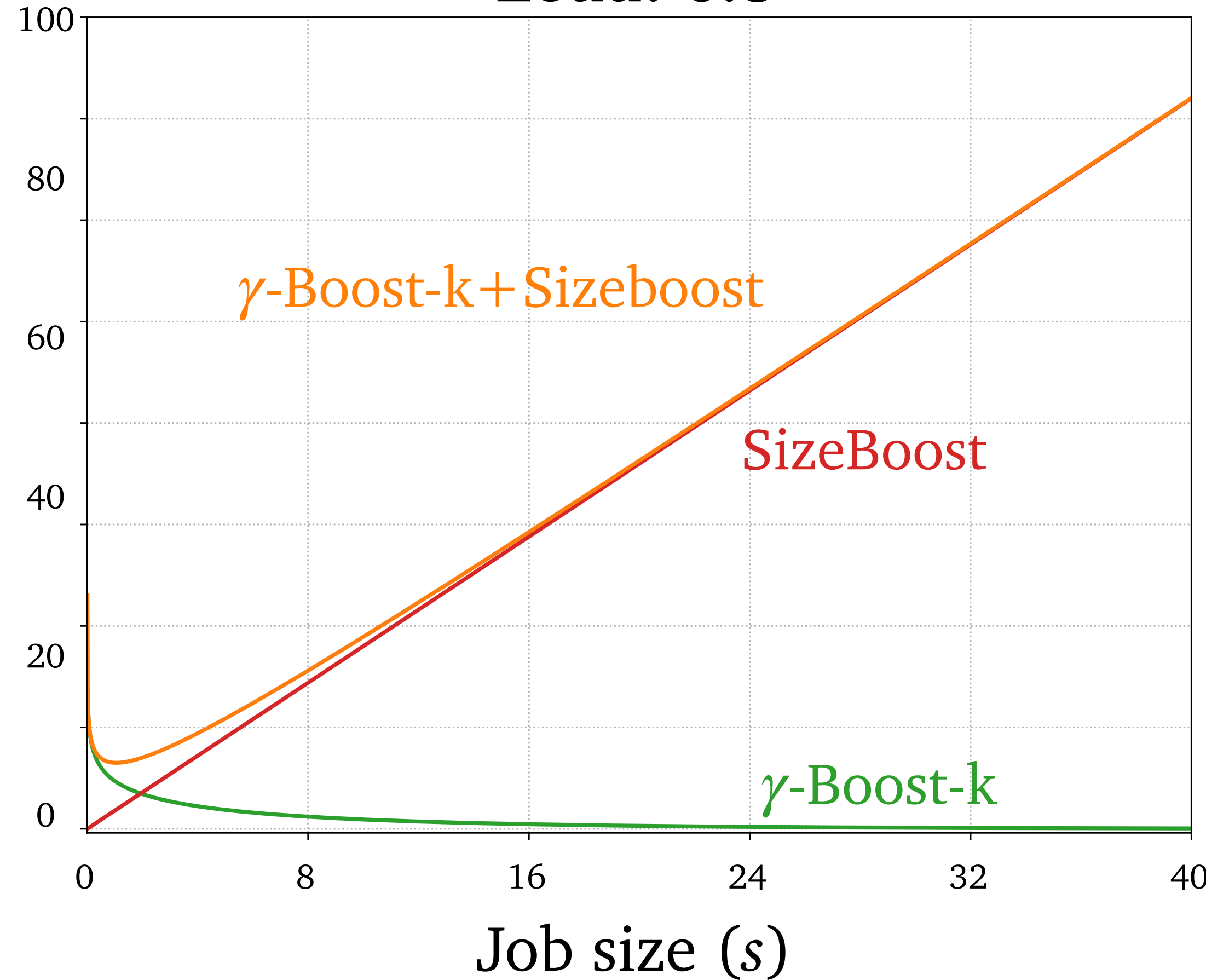
Load: 0.8



Boost as function of size s $b(s)$

Why does adding the boosts make any sense?

Load: 0.8



Load: 0.999



Boost as function of size s $b(s)$

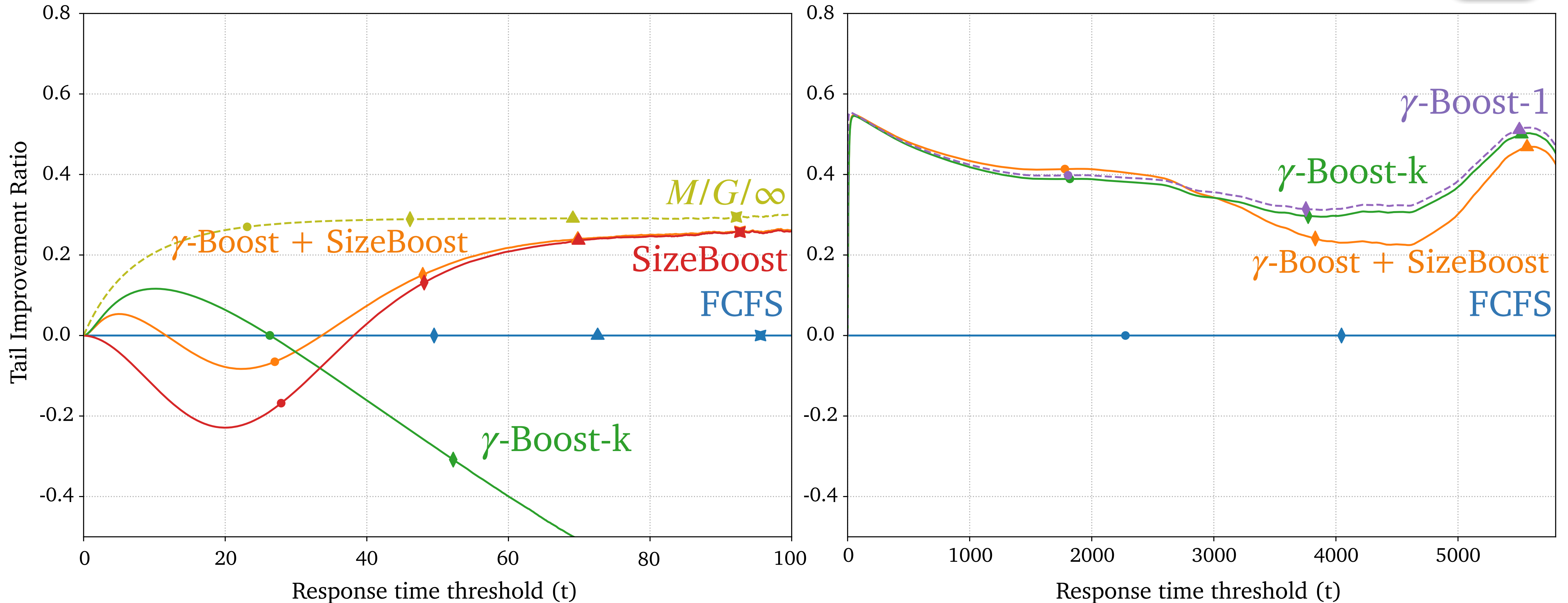
Is this policy good at both $\rho = 0.8$ and $\rho = 0.999$?

Is this policy good at both $\rho = 0.8$ and $\rho = 0.999$?

Yes!

Load: 0.8

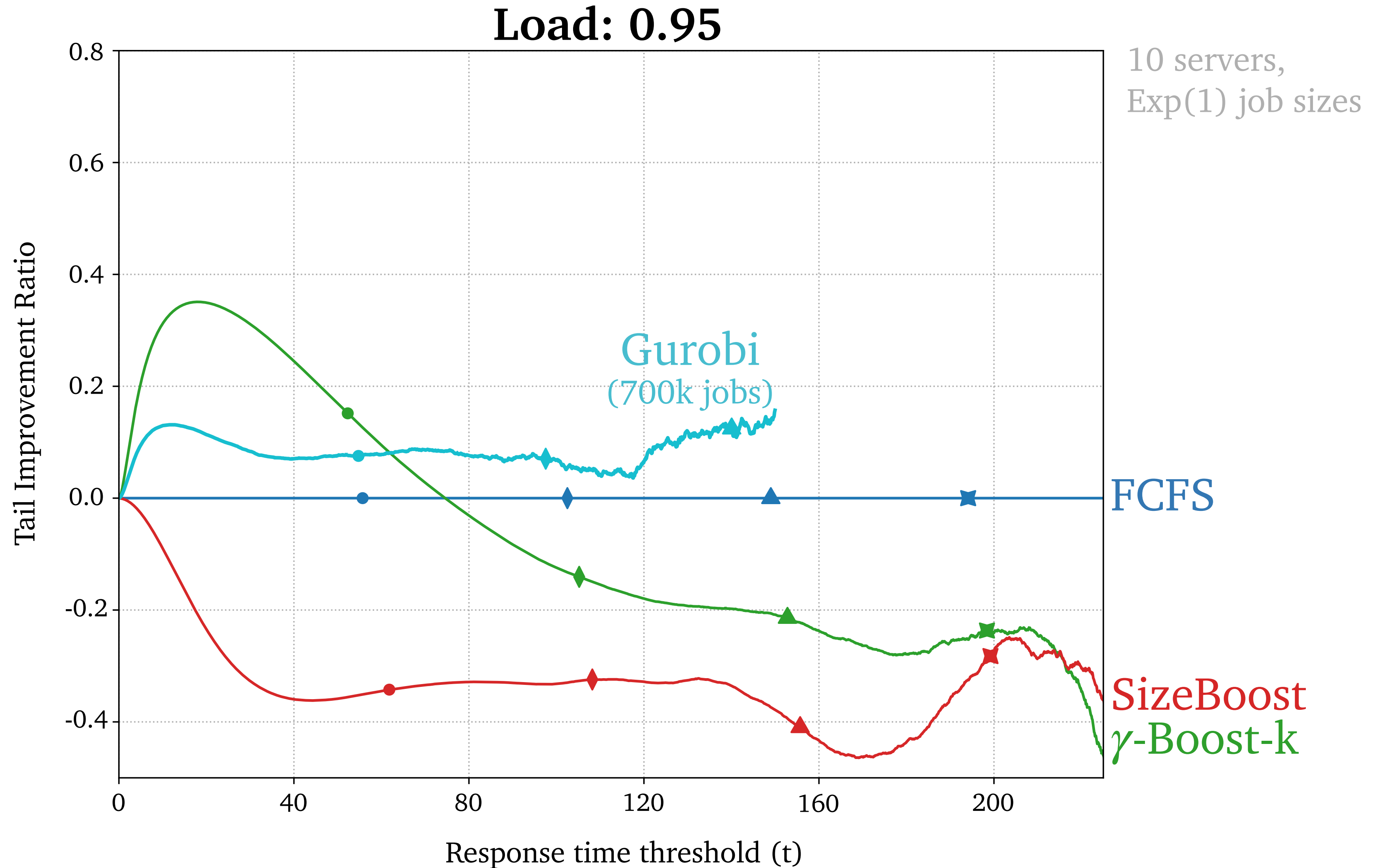
Load: 0.999



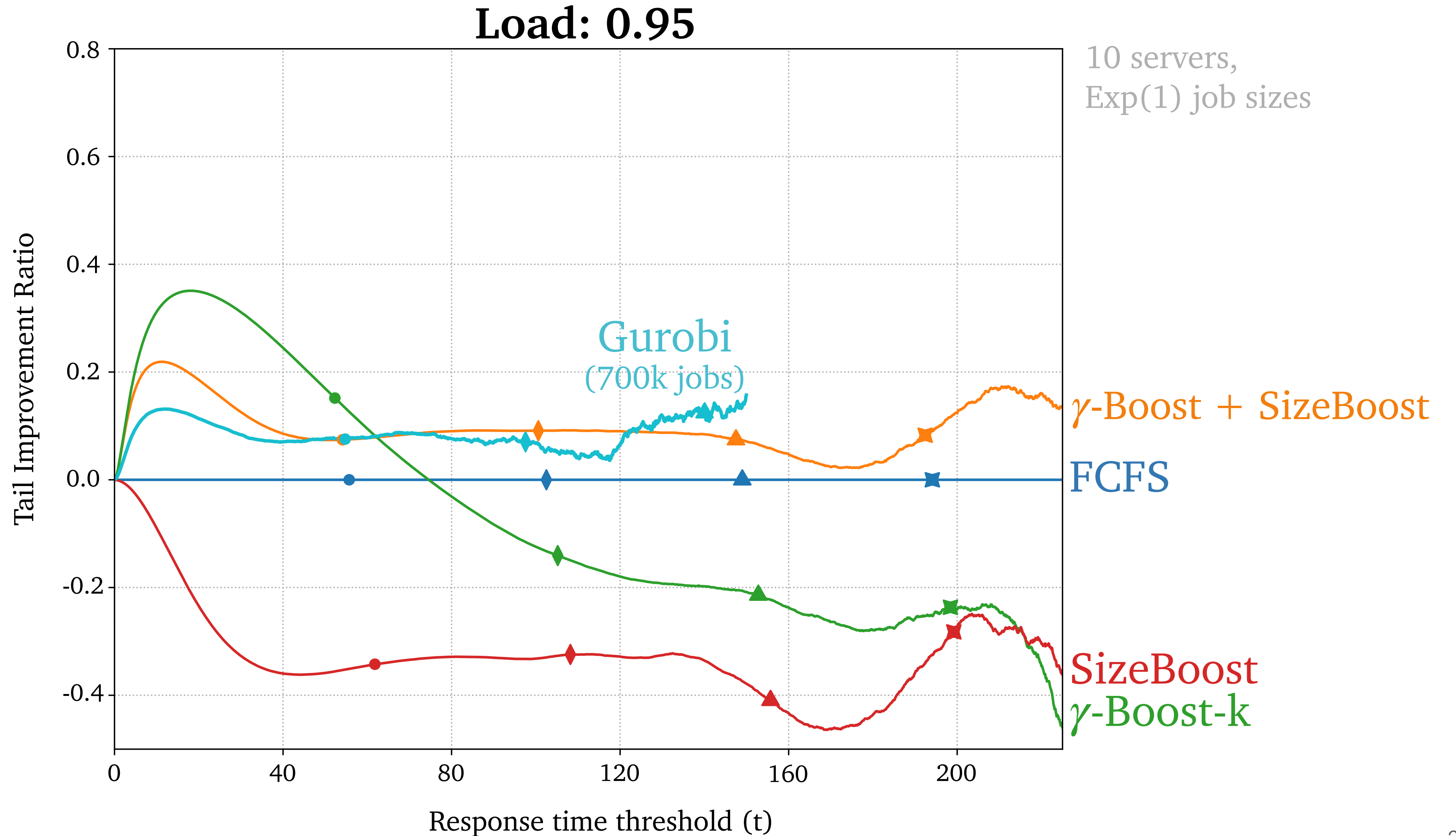
10 servers,
Exp(1) job sizes

- 90th percentile
- ◆ 99th percentile
- ▲ 99.9th percentile
- 99.99th percentile
- ★ 99.999th percentile

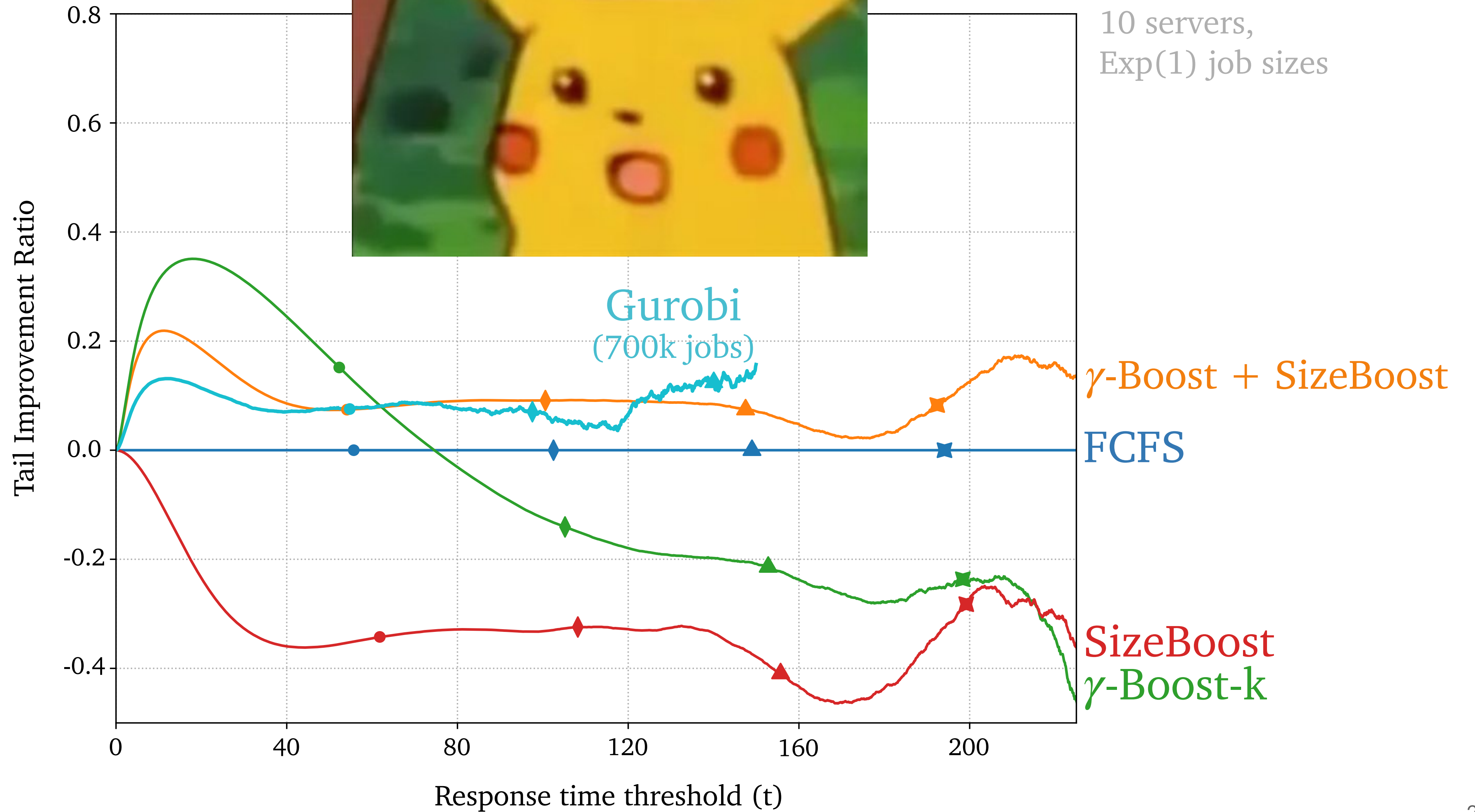
How does this policy perform at $\rho = 0.95$?



How does this policy perform at $\rho = 0.95$?



How does this m at $\rho = 0.95$?



Summary

Setting	single-server ($k = 1$)	multi-server (light traffic, $\rho \ll 1$)	multi-server (heavy traffic, $\rho \rightarrow 1$)
Objective			
$\min \mathbf{E}[T]$	SRPT (shortest remaining processing time)	SRPT	SRPT
$\min \mathbf{P}[T > t]$ "for all large t "	γ -Boost	γ -Boost + SizeBoost	γ -Boost

Hard! (pointing to multi-server light traffic)

Hope ideas carry over (arc from multi-server light traffic to multi-server heavy traffic)

[Groszof et al. 2018] (pointing to SRPT in heavy traffic)

In practice (pointing to $\min \mathbf{P}[T > t]$)

[Yu & Scully 2024] (pointing to γ -Boost in single-server)

Empirically (pointing to SRPT in single-server)

Empirically (pointing to γ -Boost + SizeBoost in multi-server light traffic)

Optimality proof! (pointing to γ -Boost in multi-server heavy traffic)

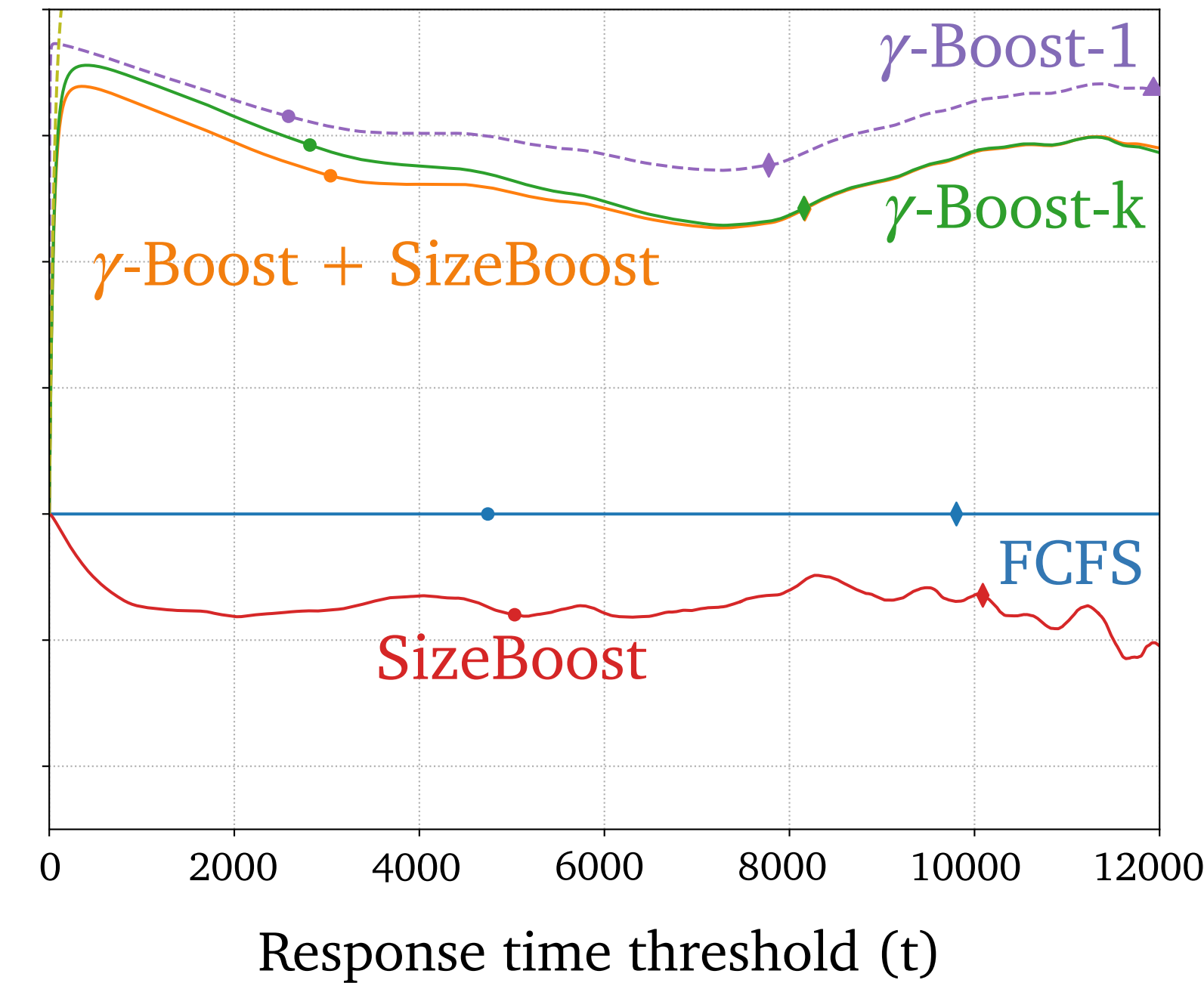
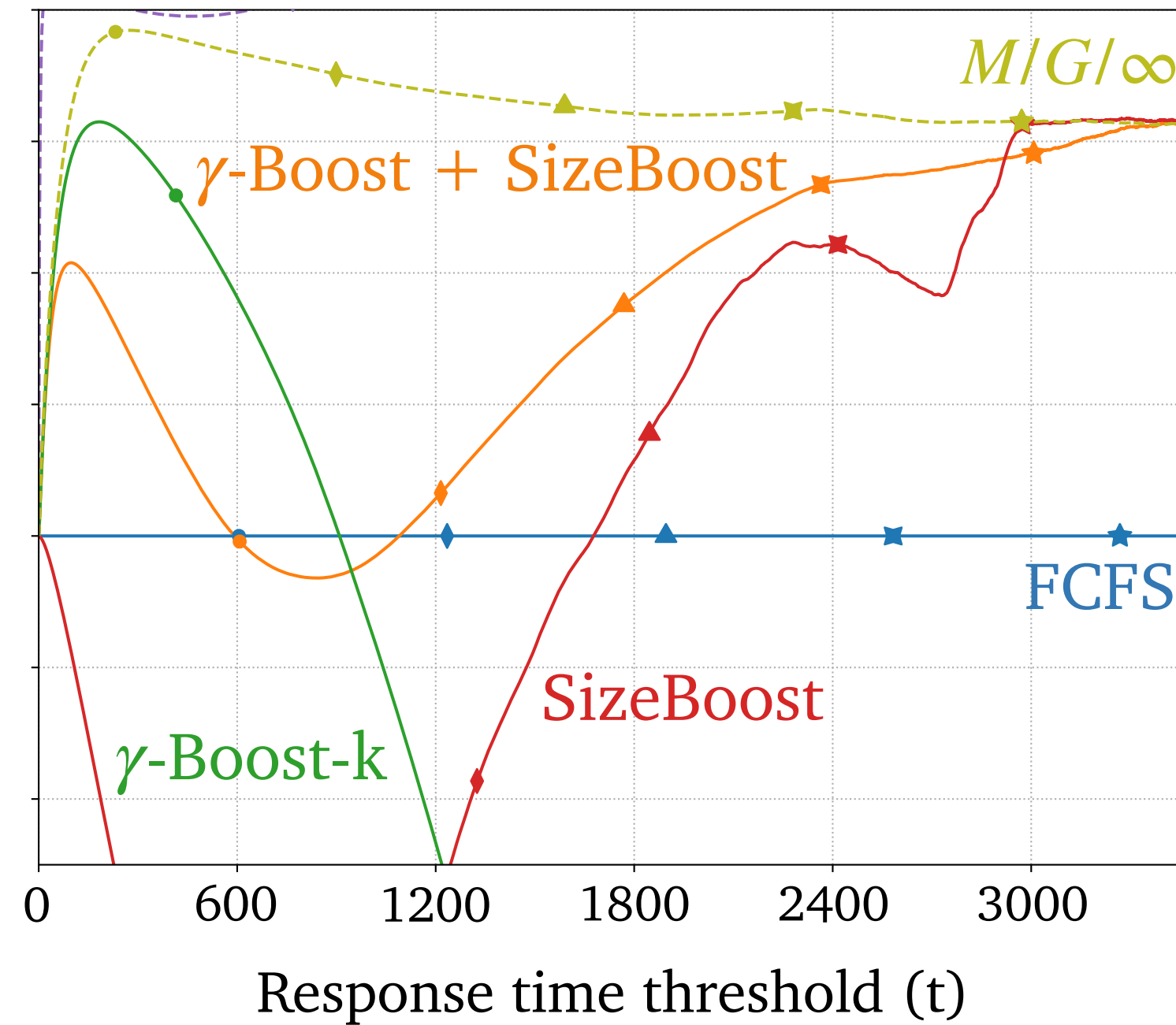
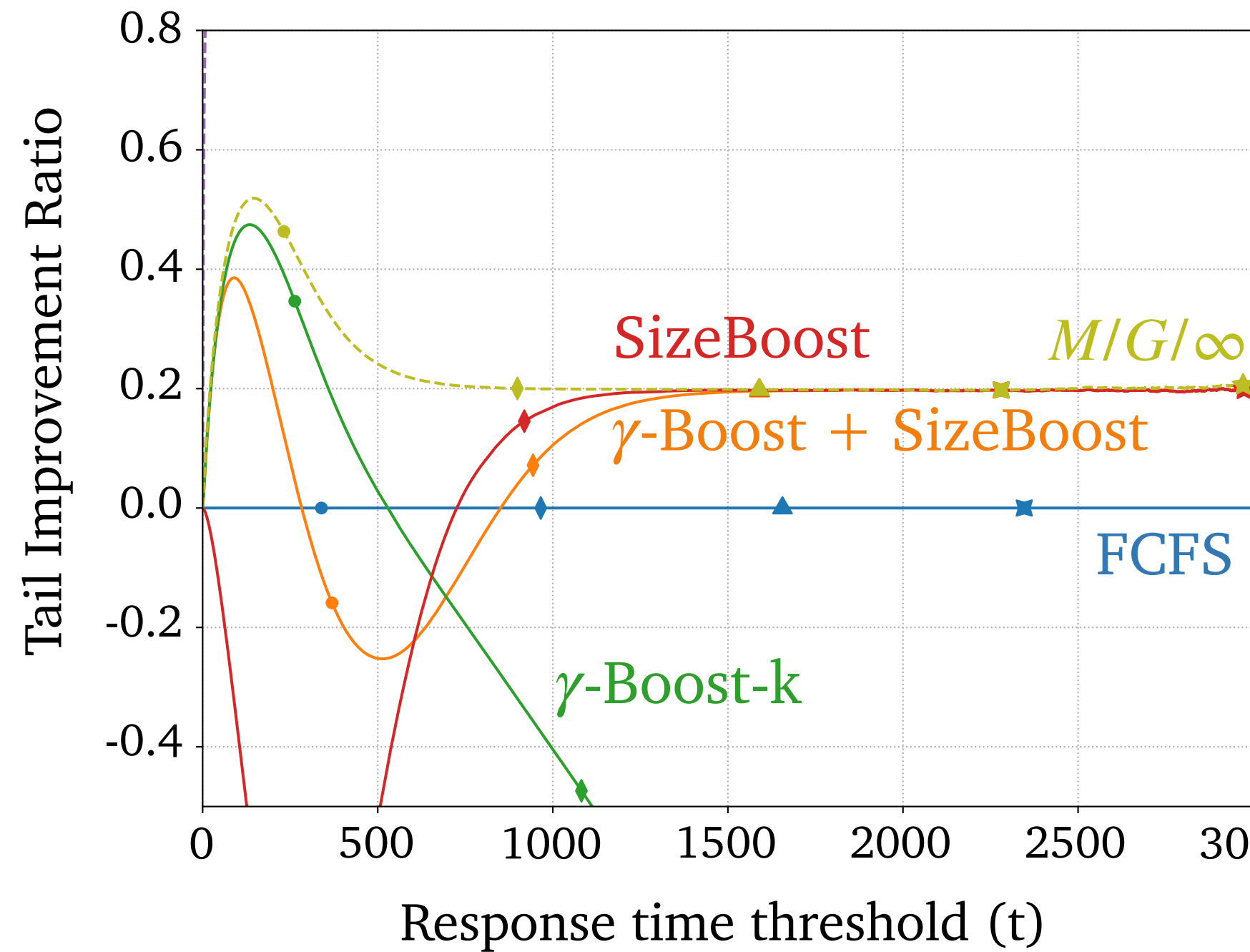
Our Contribution (green text, encompassing γ -Boost + SizeBoost and γ -Boost in heavy traffic)

100 servers

Load: 0.975

Load: 0.99

Load: 0.999



100 servers,
Hyperexponential job sizes