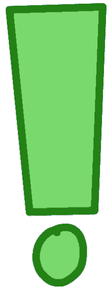
A Gittins Policy for Optimizing Tail Latency

Amit Harlev Cornell CAM

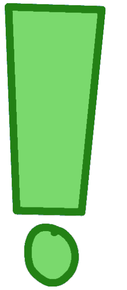
Joint work with

George Yu Cornell ORIE

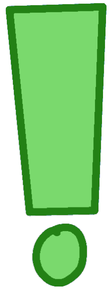
Ziv Scully Cornell ORIE



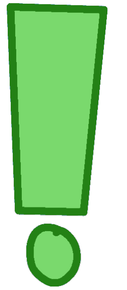
Main takeaway



We know how to minimize tail latency in queues!

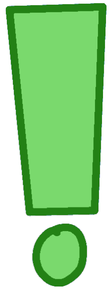


Main takeaway

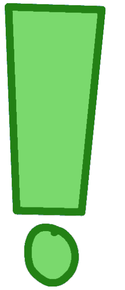


We know how to minimize tail latency in queues!

for most information models



Main takeaway

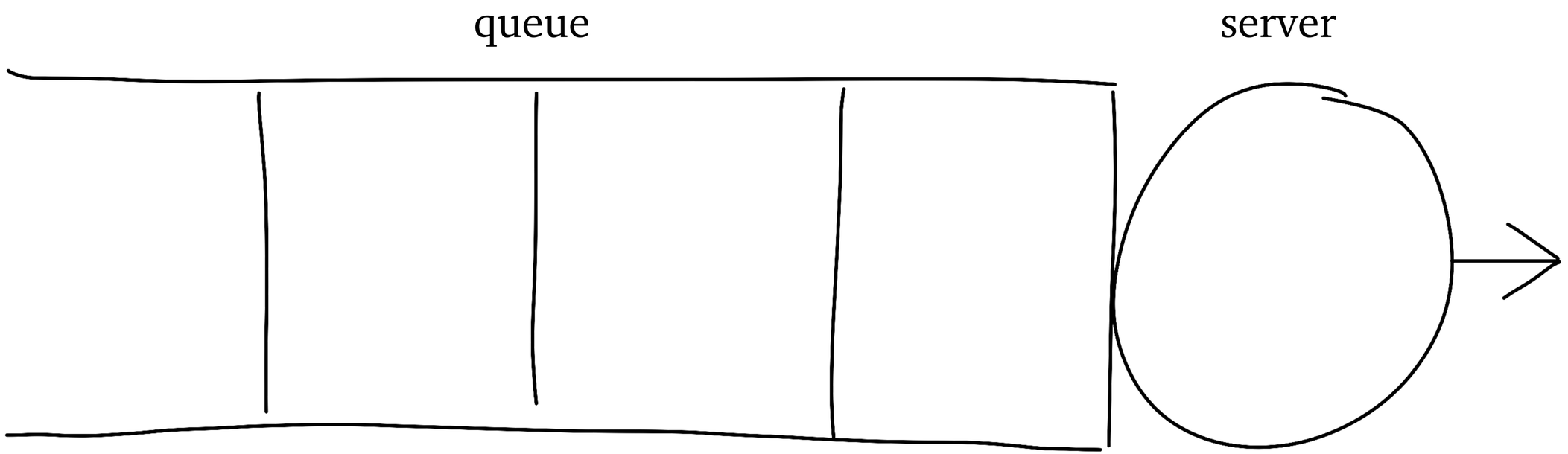


We know how to minimize tail latency in queues!

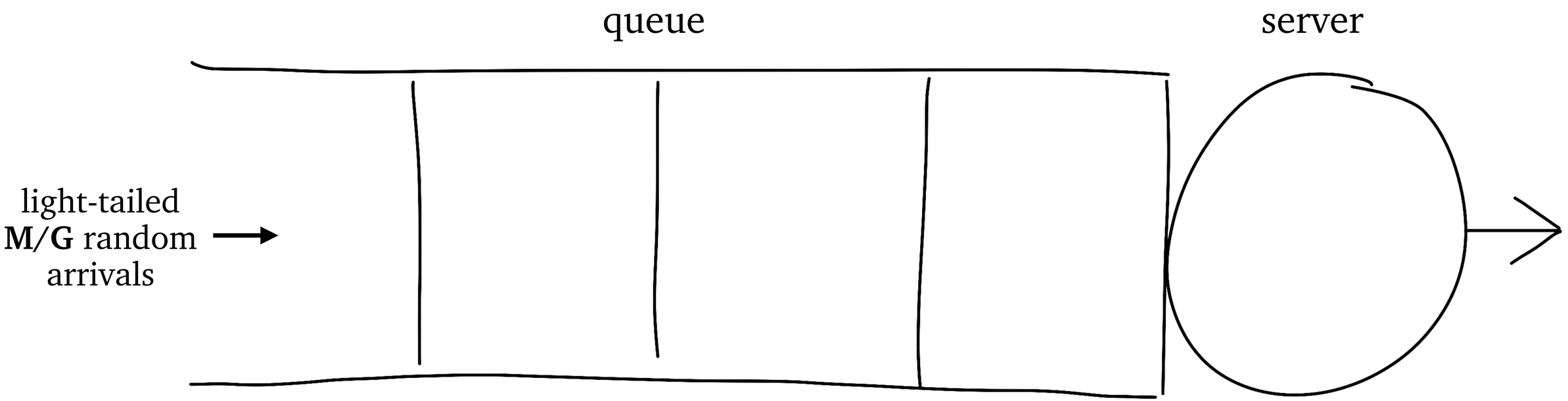
for most information models

If you have a problem where this is needed,
please come talk to us!

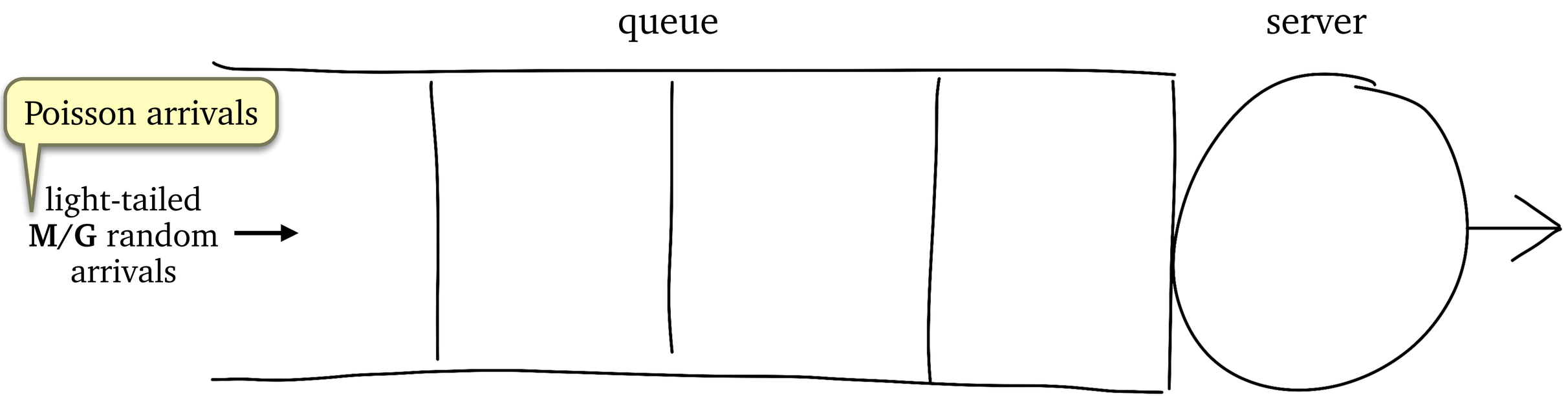
What is the problem?



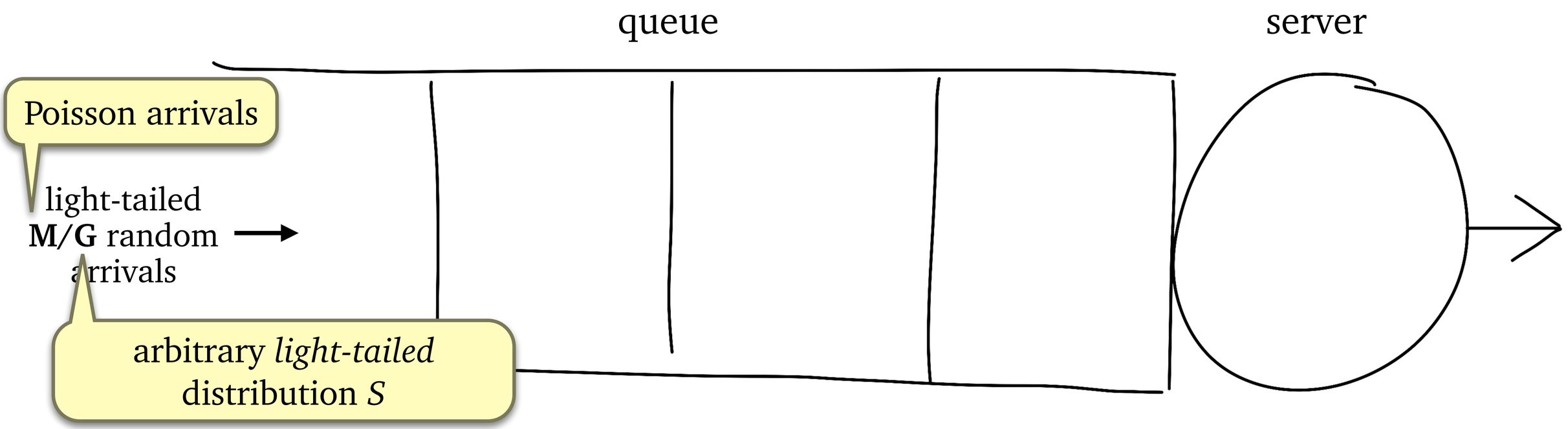
What is the problem?



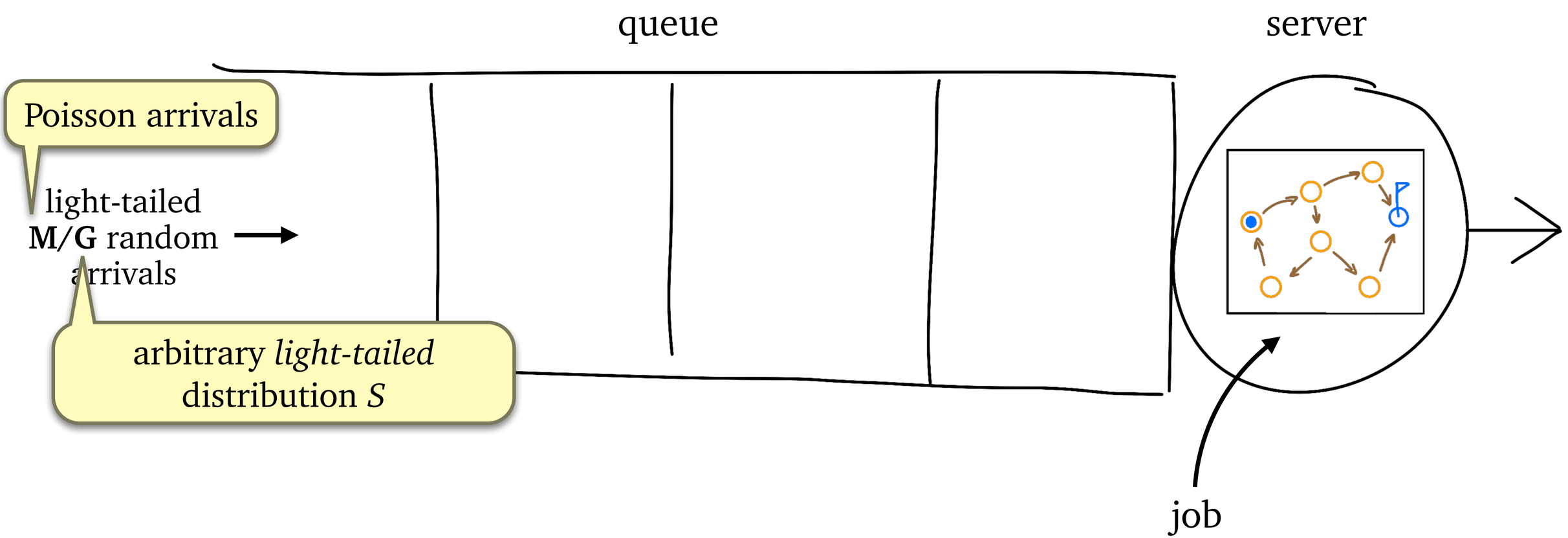
What is the problem?



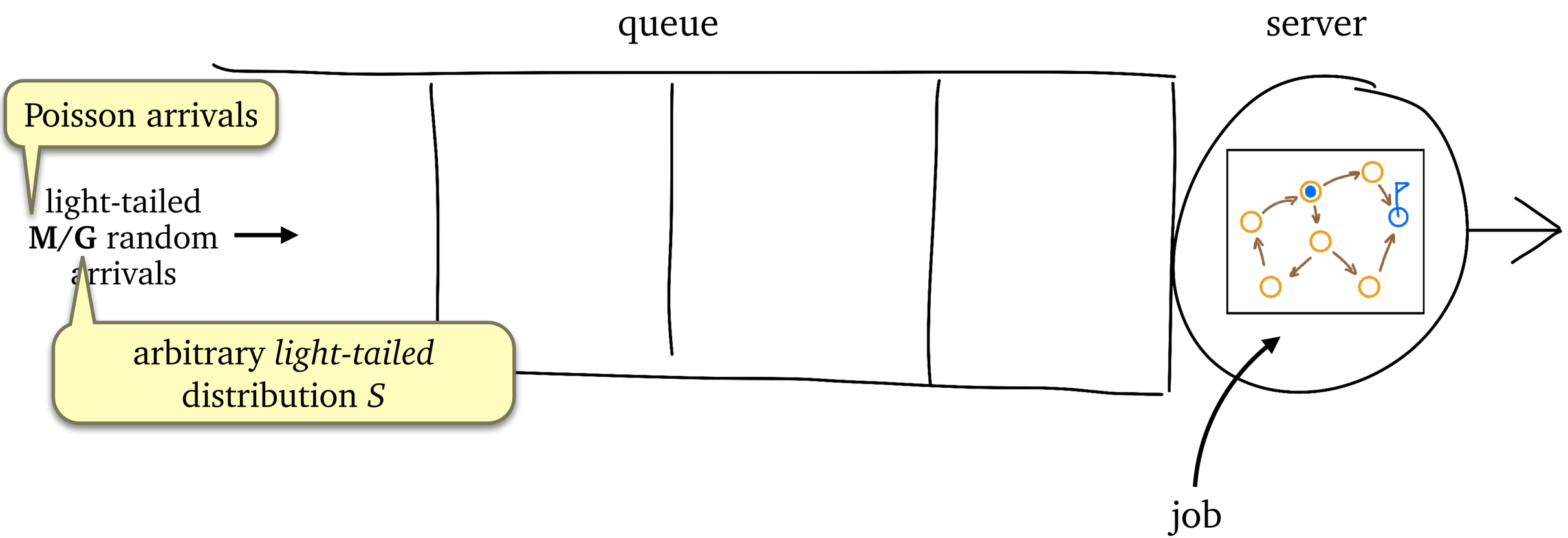
What is the problem?



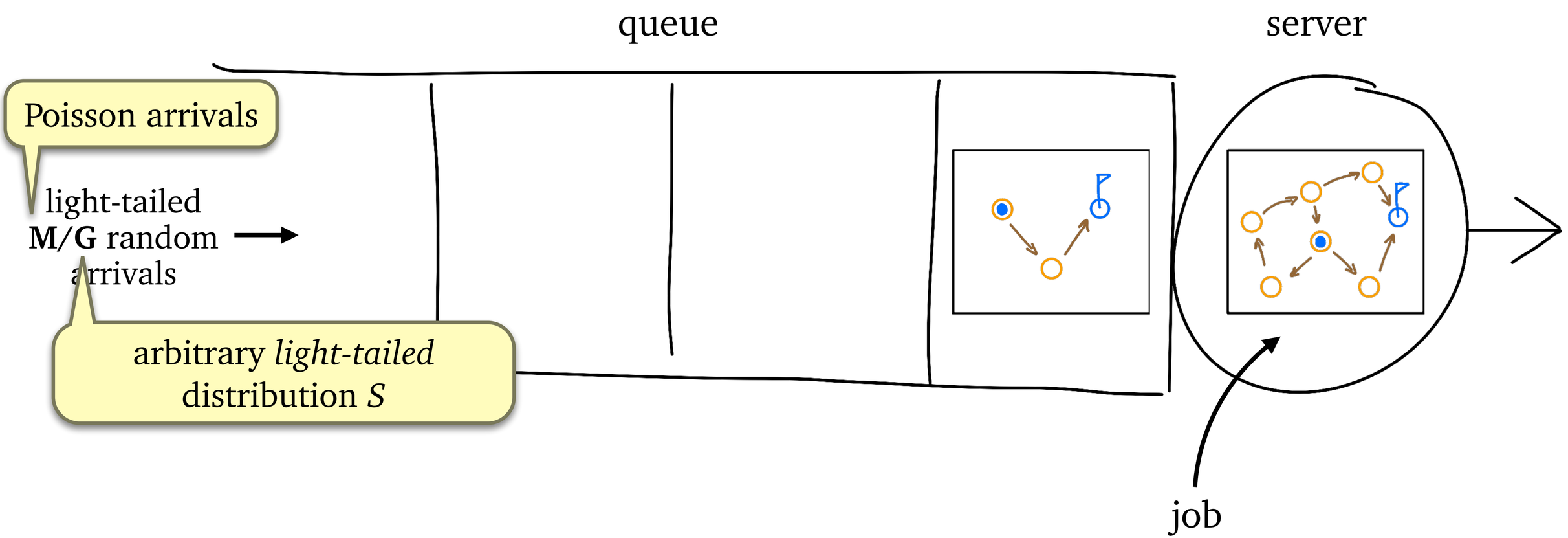
What is the problem?



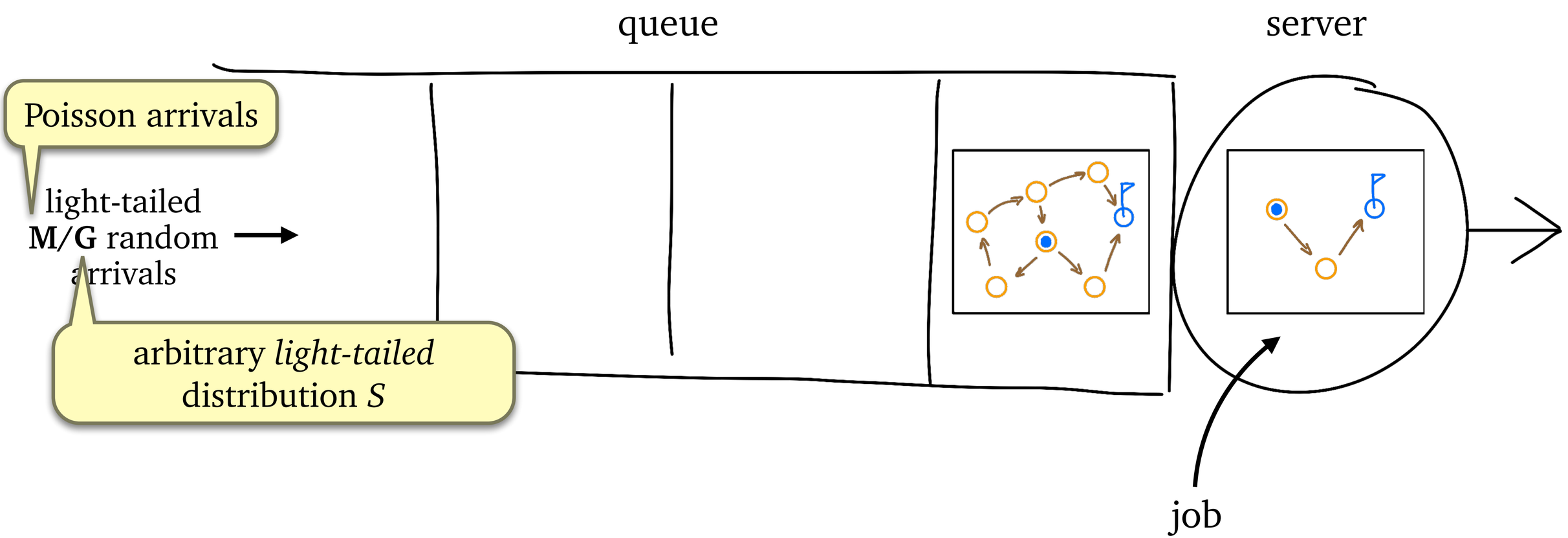
What is the problem?



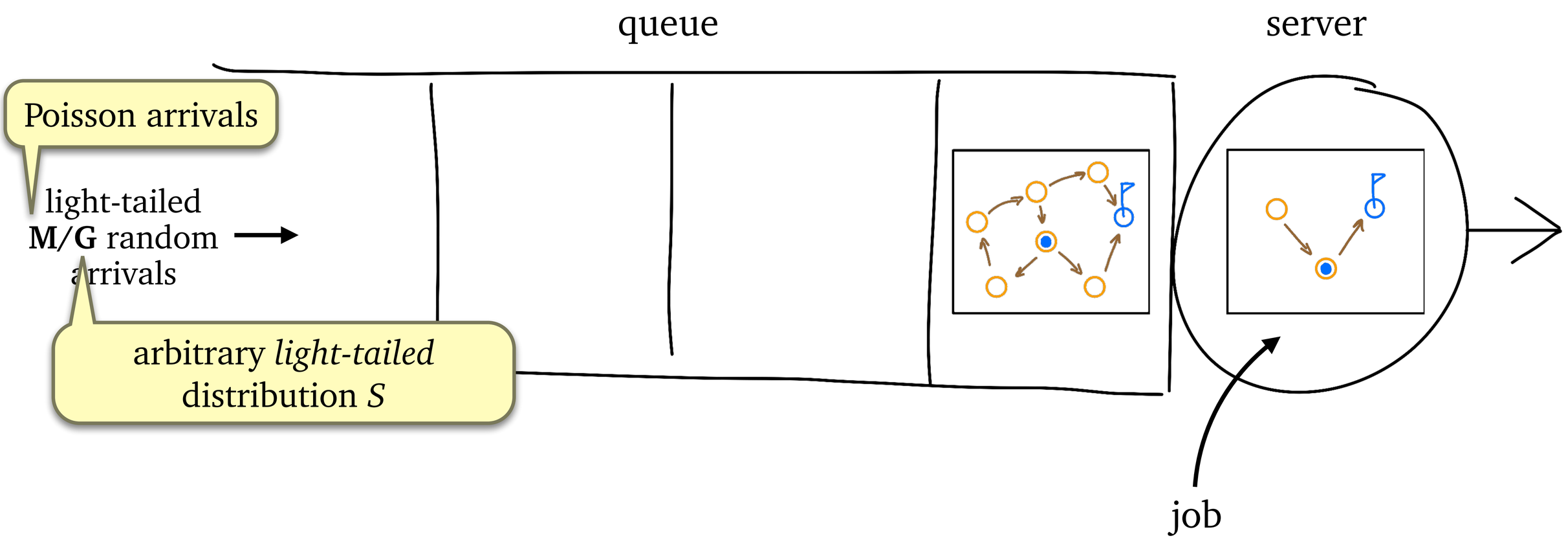
What is the problem?



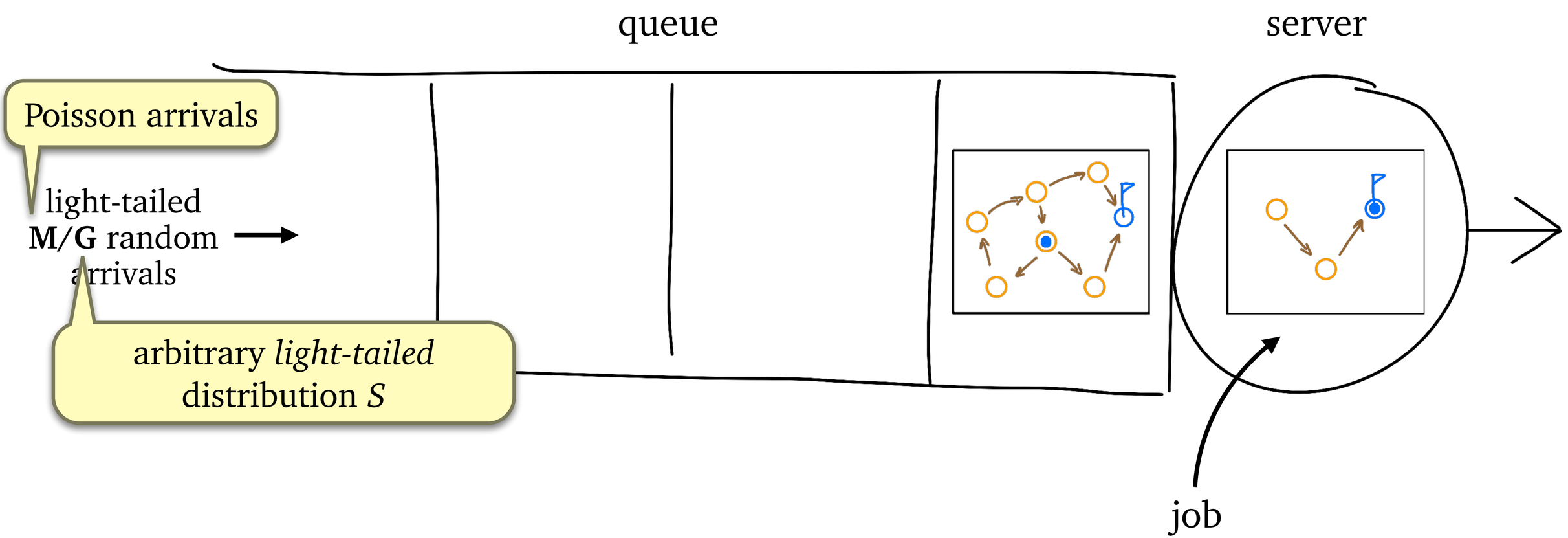
What is the problem?



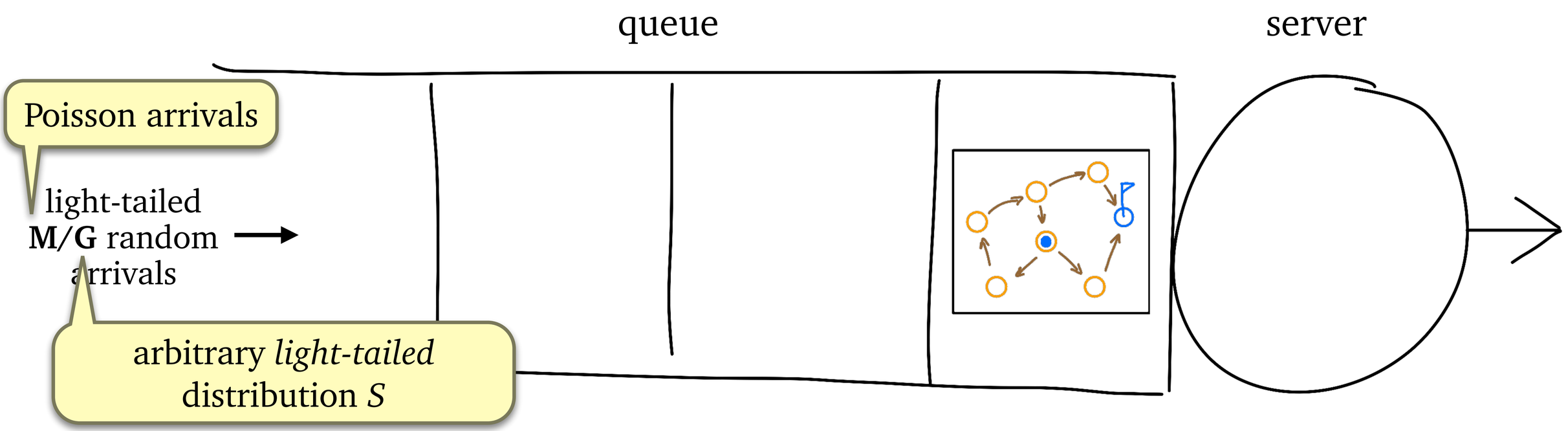
What is the problem?



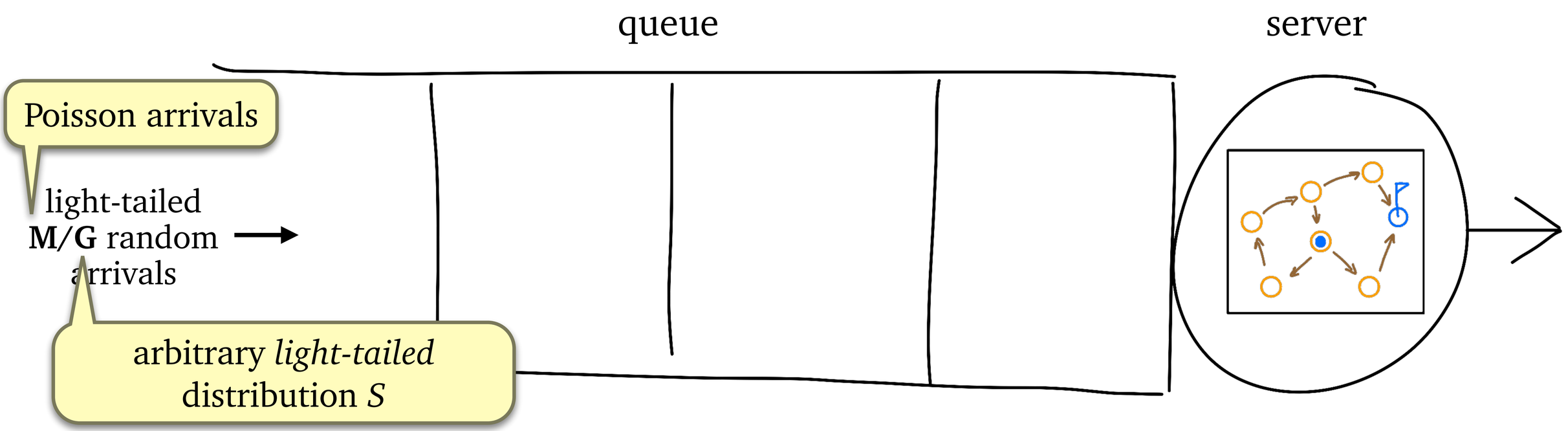
What is the problem?



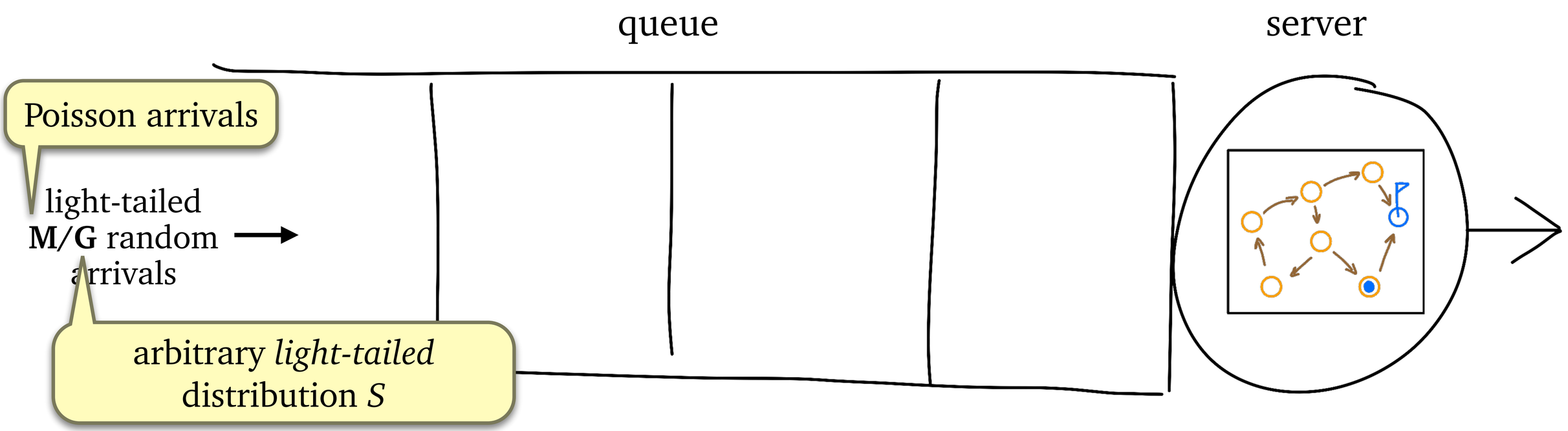
What is the problem?



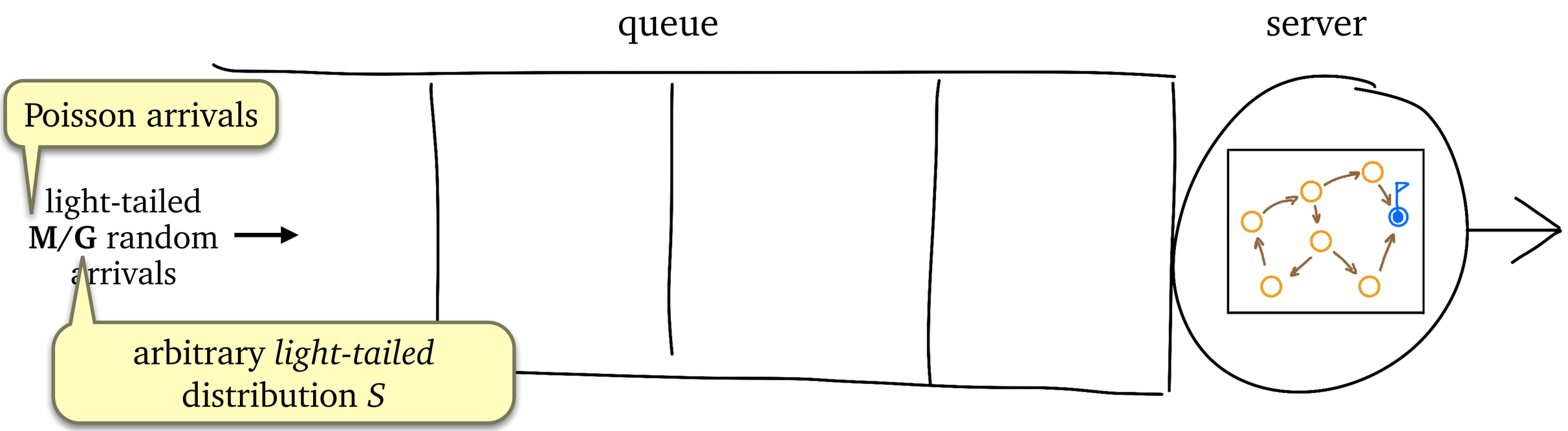
What is the problem?



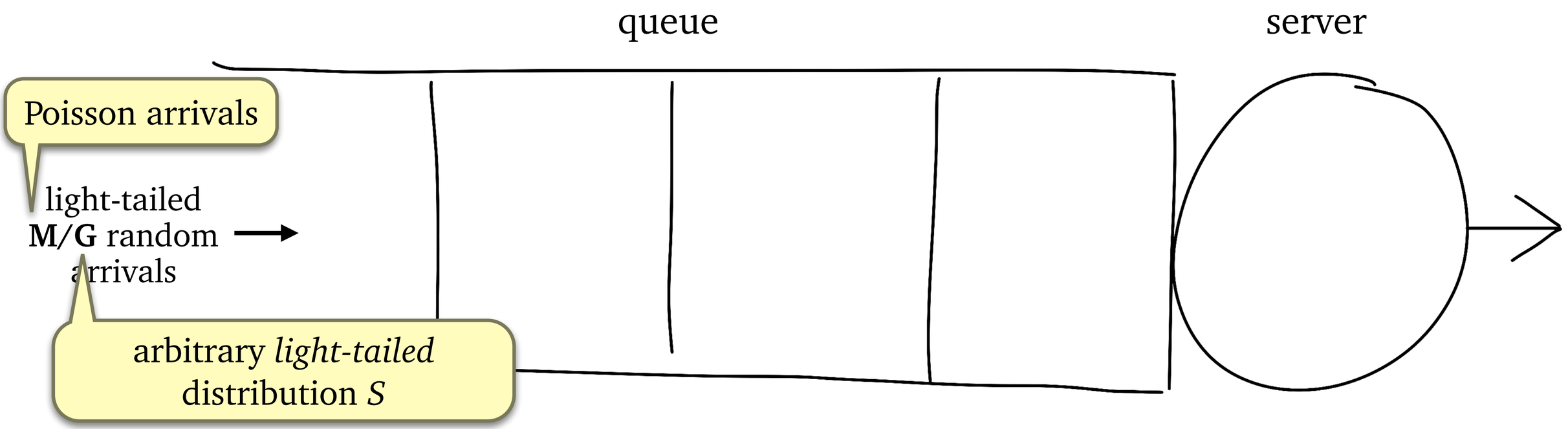
What is the problem?

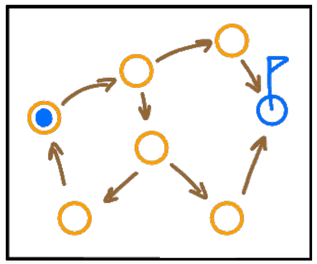


What is the problem?



What is the problem?





captures many information models in a single result

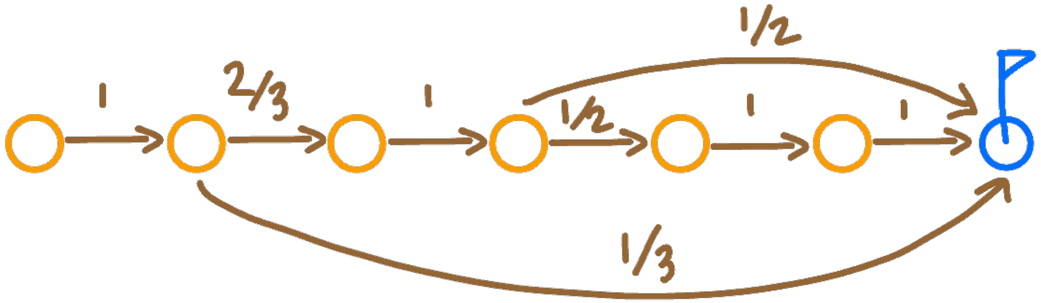
Examples of information models

Known Size
($S = 6$)

Examples of information models

Known Size
($S = 6$)

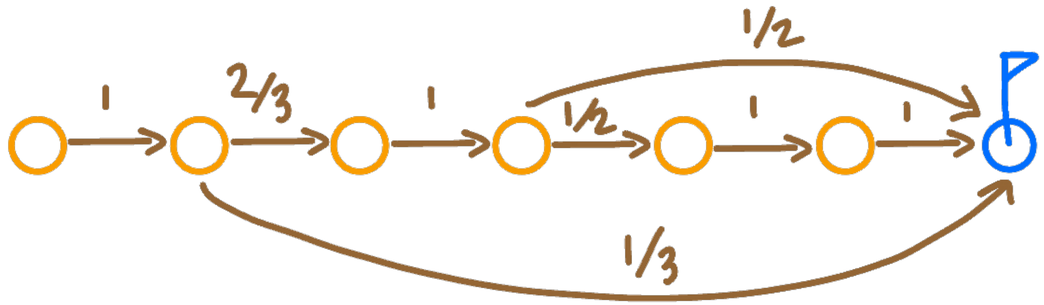
Unknown Size
($S \sim \text{Unif}\{2,4,6\}$)



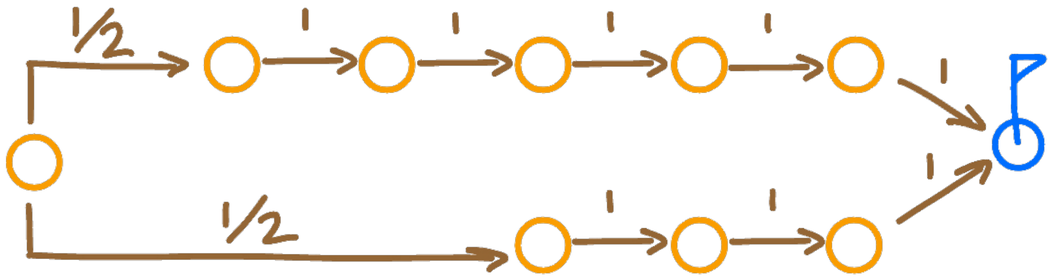
Examples of information models

Known Size
($S = 6$)

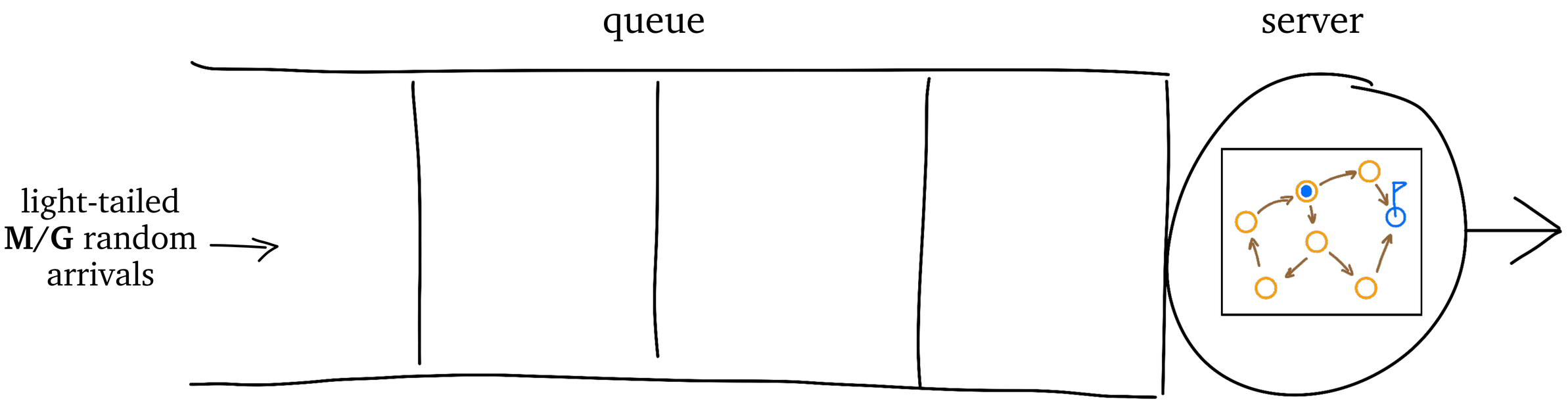
Unknown Size
($S \sim \text{Unif}\{2,4,6\}$)



Partial information
($S \sim \text{Unif}\{4,6\}$)

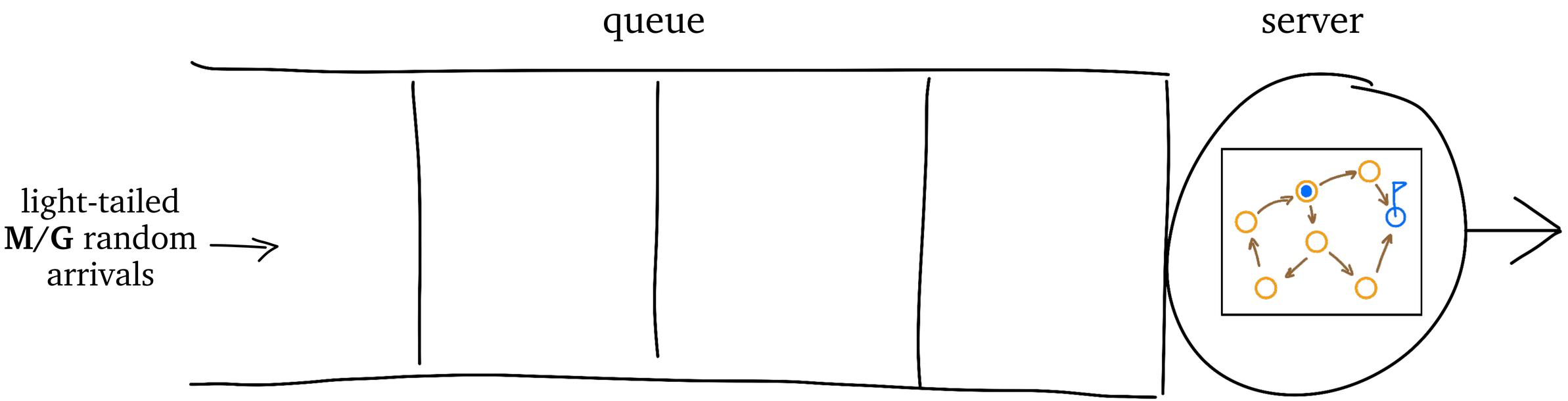


What is the problem?



Goal: scheduling policy π that minimizes *asymptotic behavior* of the *tail of response time,* $\mathbf{P}[T_\pi > x]$

What is the problem?

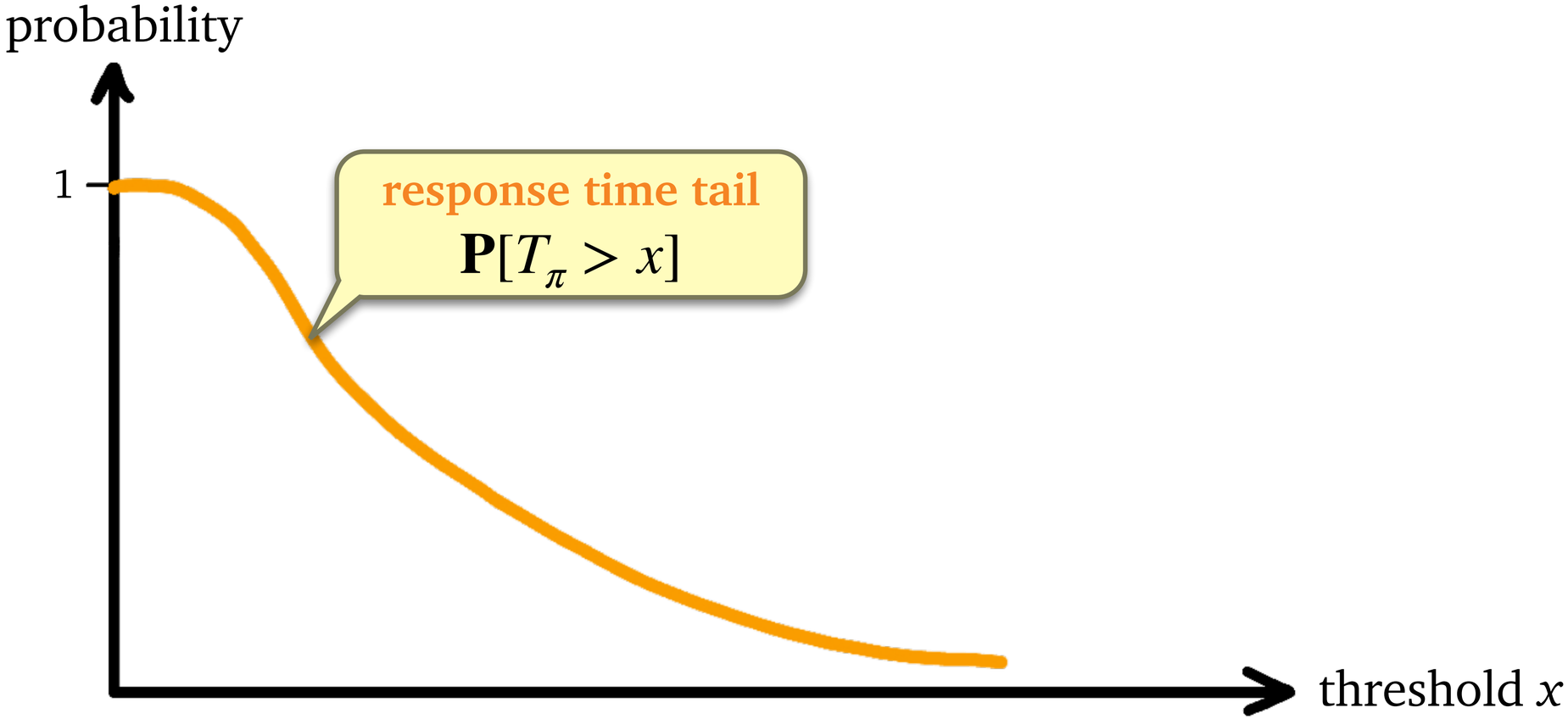


Goal: scheduling policy π that minimizes *asymptotic behavior* of the *tail of response time,* $\mathbf{P}[T_\pi > x]$

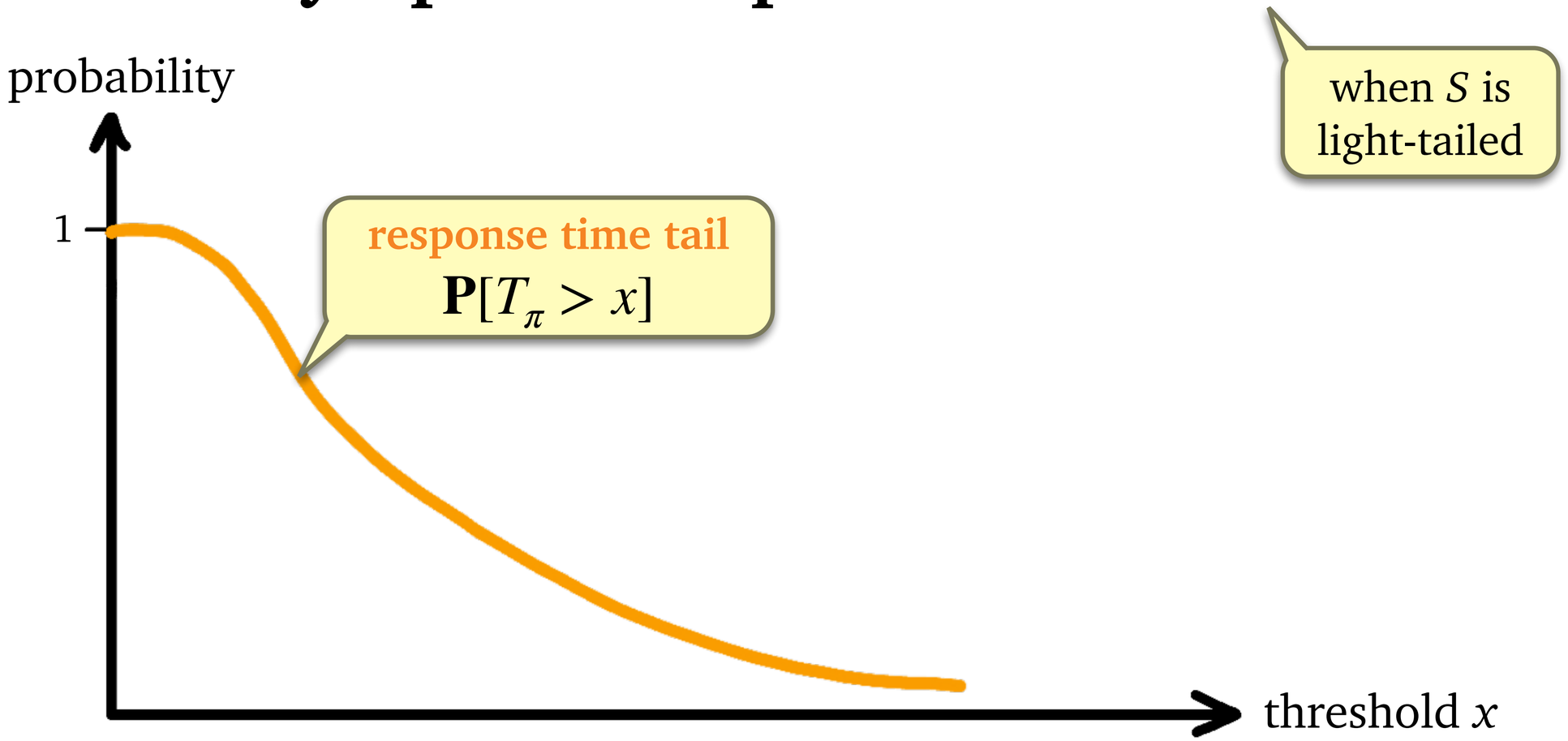
T_π = response time under policy π
("total time in system")

Asymptotic response time tail

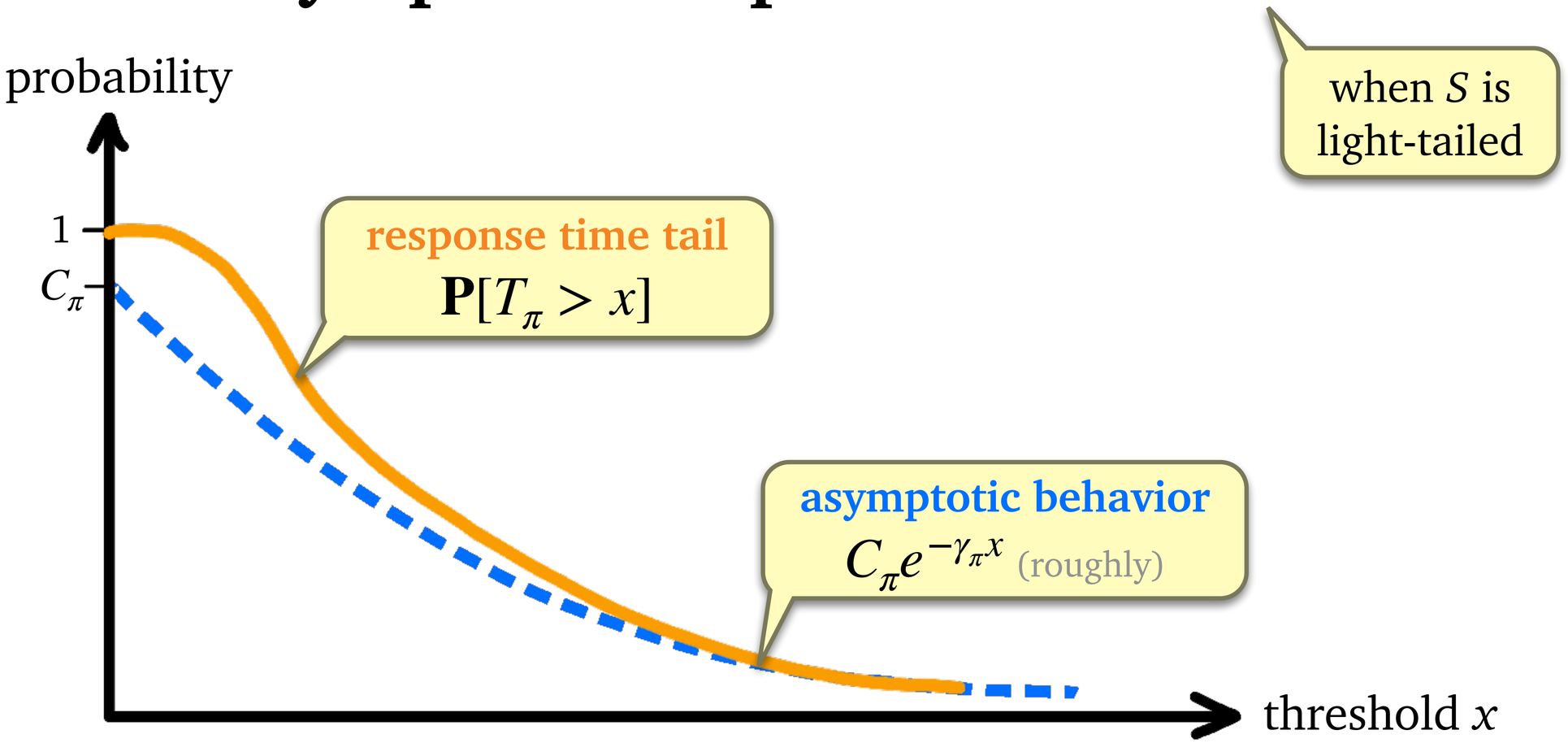
Asymptotic response time tail



Asymptotic response time tail



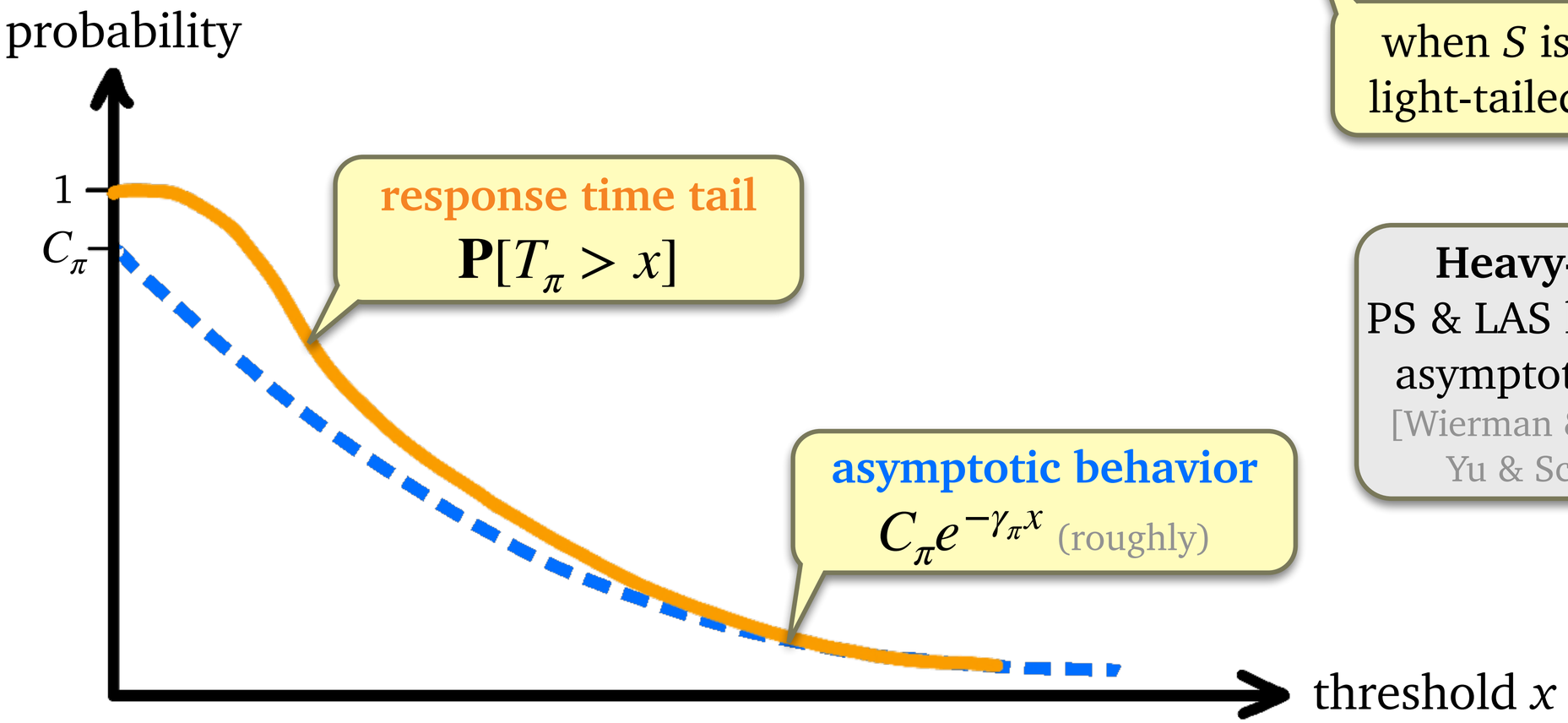
Asymptotic response time tail



$\gamma_\pi = \textit{decay rate of } \pi$

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

Asymptotic response time tail



when S is light-tailed

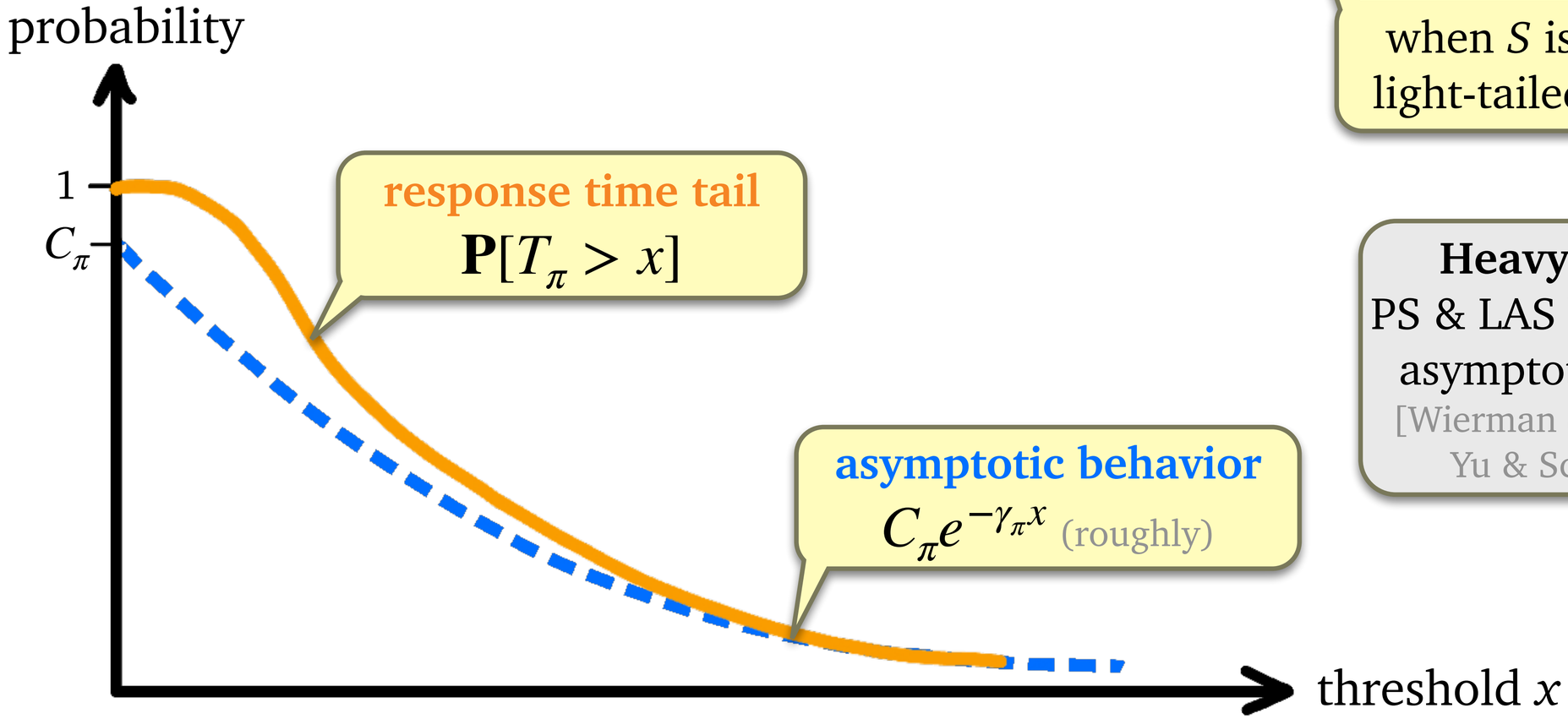
Heavy-tailed S:
PS & LAS have optimal asymptotic behavior
[Wierman & Zwart 2012, Yu & Scully 2024]

$\gamma_\pi = decay\ rate$ of π

$C_\pi = tail\ constant$ of π

Asymptotic response time tail

when S is light-tailed



Heavy-tailed S:
PS & LAS have optimal asymptotic behavior
[Wierman & Zwart 2012, Yu & Scully 2024]

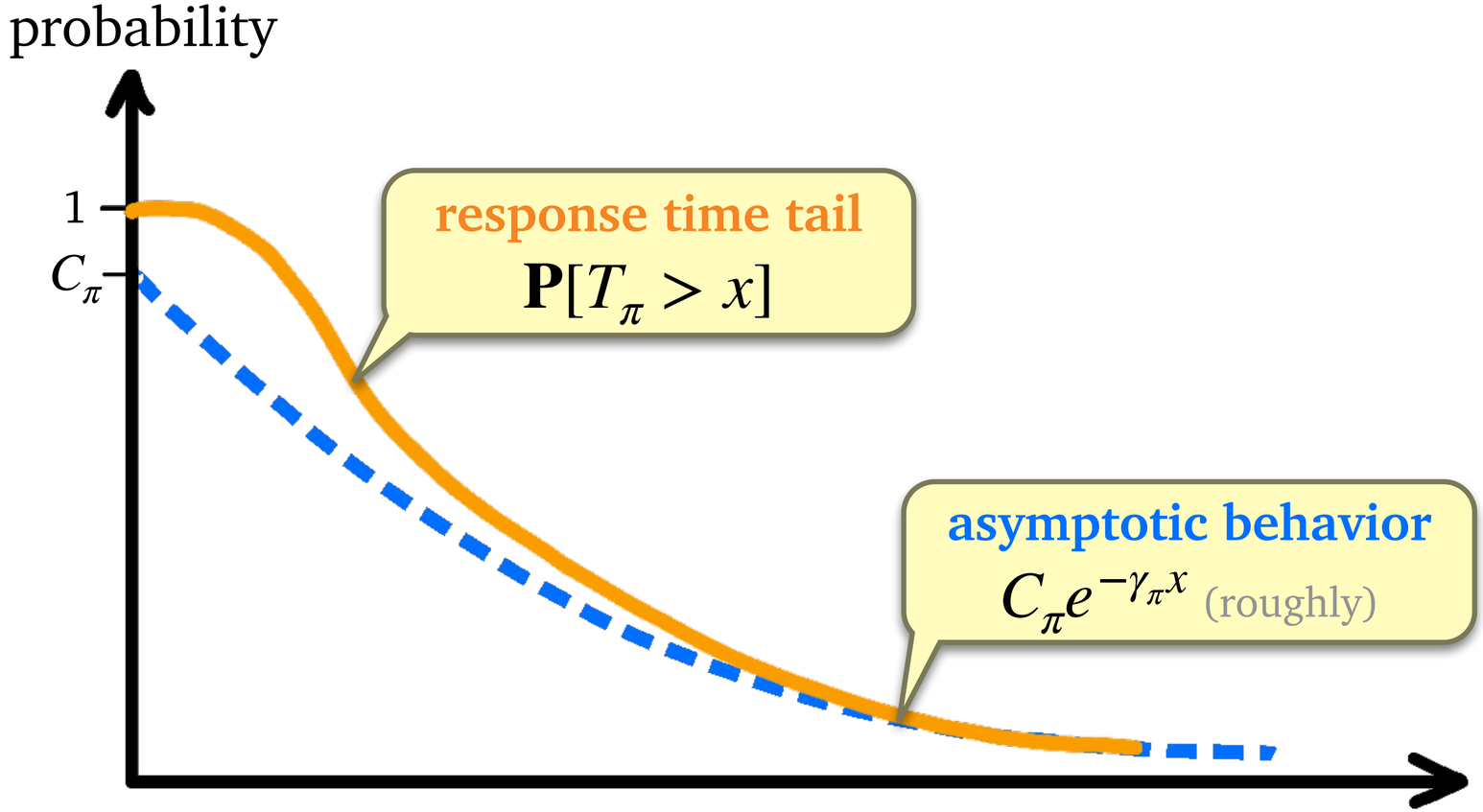
Weak optimality:
optimal γ_π

γ_π = *decay rate* of π

C_π = *tail constant* of π

Asymptotic response time tail

when S is light-tailed



Heavy-tailed S:
PS & LAS have optimal asymptotic behavior
[Wierman & Zwart 2012, Yu & Scully 2024]

Weak optimality:
optimal γ_π

$\gamma_\pi = decay\ rate$ of π

$C_\pi = tail\ constant$ of π

Strong optimality:
optimal γ_π and C_π

Why is tail scheduling hard?

Why is tail scheduling hard?

scheduling for response time: *short jobs before long jobs*

Why is tail scheduling hard?

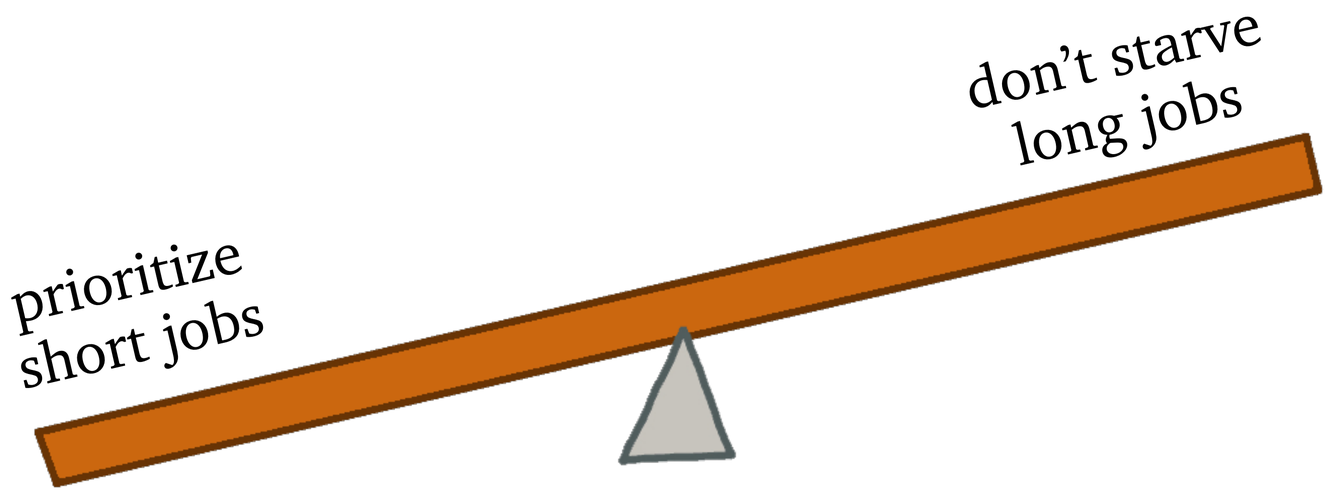
scheduling for response time: *short jobs before long jobs*

…leads to poor tail performance

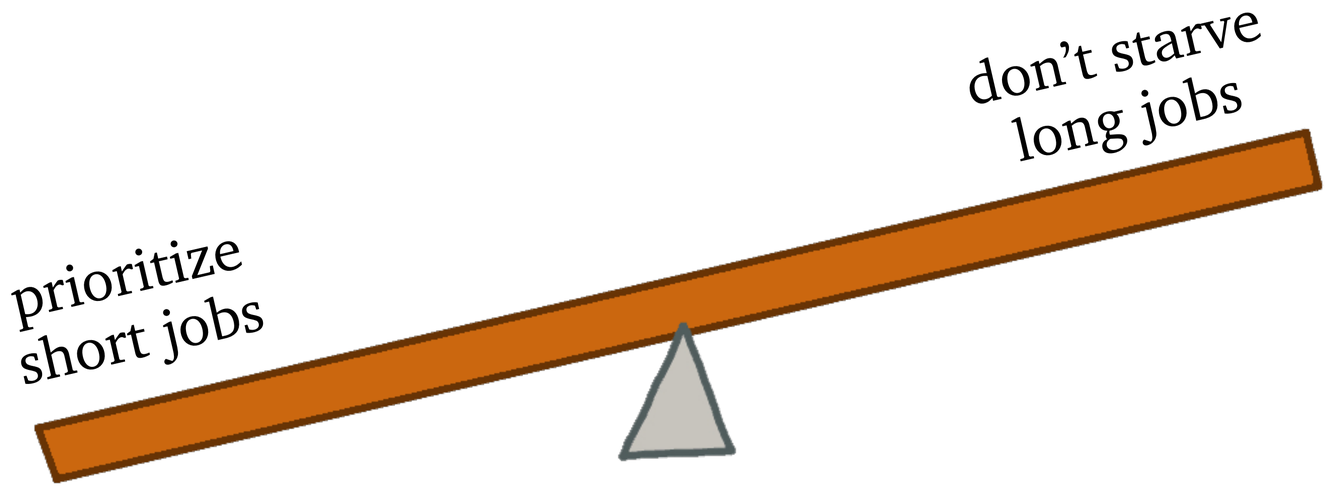
Why is tail scheduling hard?

scheduling for response time: *short jobs before long jobs*

…leads to poor tail performance

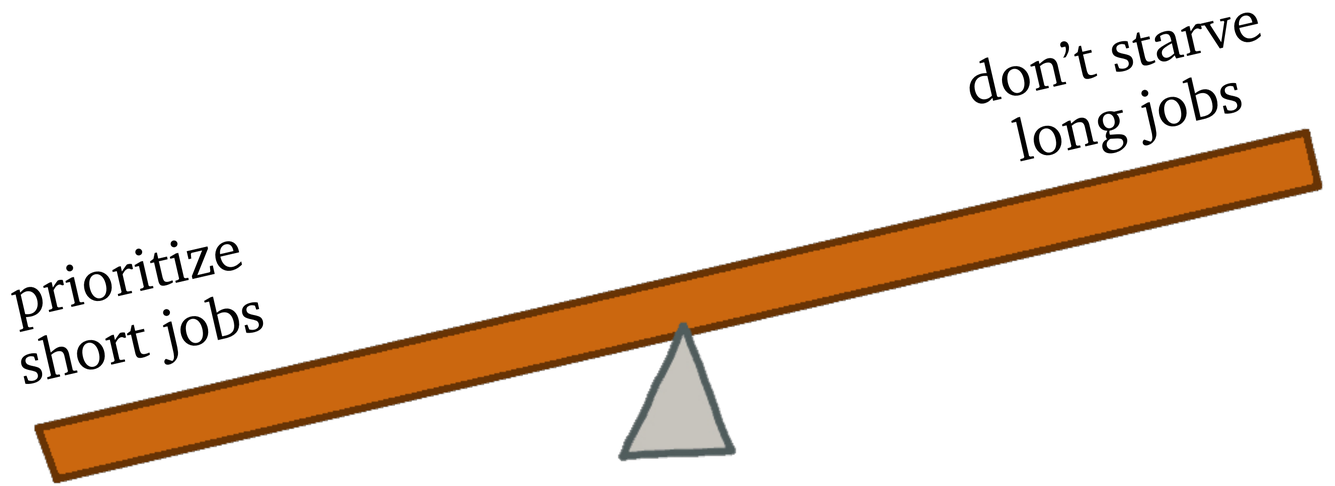


Managing the tradeoff



Does any policy in the literature balance this?

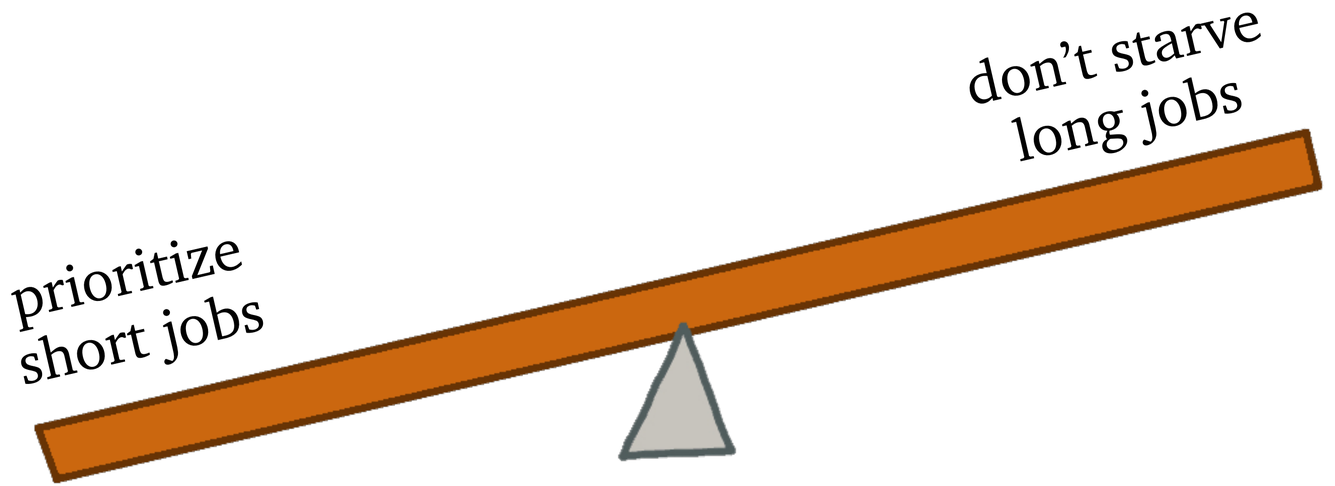
Managing the tradeoff



Does any policy in the literature balance this?

Most policies: priority based on job’s size or attained service

Managing the tradeoff



Does any policy in the literature balance this?

Most policies: priority based on job's size or attained service

not flexible enough!

Managing the tradeoff

[Scully & van Kreveld, 2024]:

FCFS is only SOAP policy that is even *weakly optimal.*

Does nce this?

Most policies: priority based on job’s size or attained service

not flexible enough!

Existing work

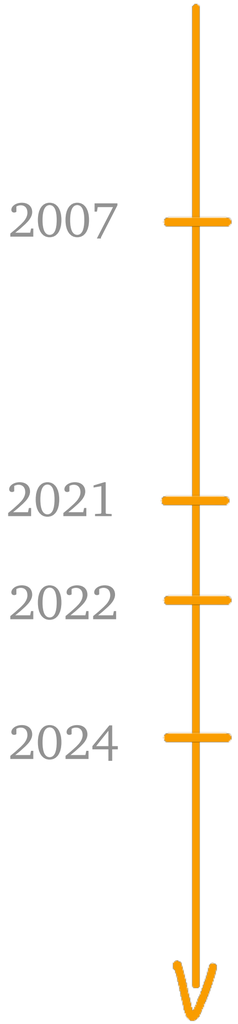
Existing work

	Known Sizes	Partial Information
2007	FCFS weakly optimal [Boxma & Zwart]	FCFS weakly optimal [Boxma & Zwart]

Existing work

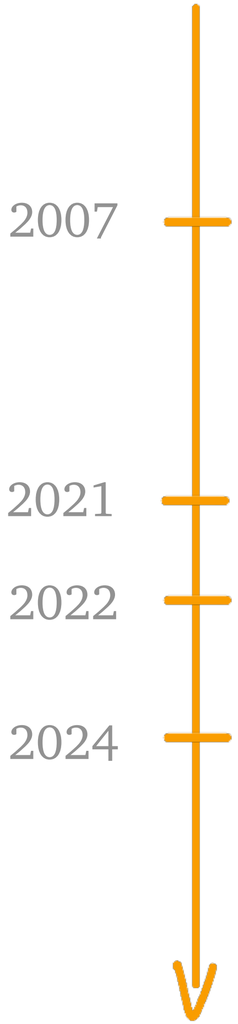
	Known Sizes	Partial Information
2007	FCFS weakly optimal [Boxma & Zwart]	FCFS weakly optimal [Boxma & Zwart]
2021	Nudge [Grosof et al]	

Existing work



	Known Sizes	Partial Information
2007	FCFS weakly optimal [Boxma & Zwart]	FCFS weakly optimal [Boxma & Zwart]
2021	Nudge [Grosof et al]	
2022	Nudge-K [Van Houdt]	
2024	Nudge-M [Charlet & Van Houdt]	

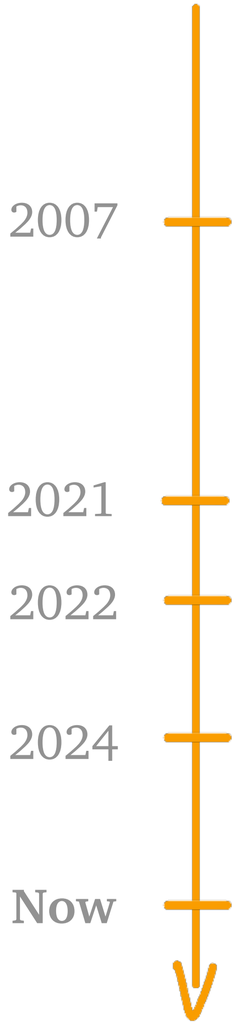
Existing work



	Known Sizes	Partial Information
2007	FCFS weakly optimal [Boxma & Zwart]	FCFS weakly optimal [Boxma & Zwart]
2021	Nudge [Grosof et al]	
2022	Nudge-K [Van Houdt]	
2024	Nudge-M [Charlet & Van Houdt]	
	γ-Boost [Yu & Scully, SIGMETRICS 2024]	

strongly optimal!

Existing work



	Known Sizes	Partial Information
2007	FCFS weakly optimal [Boxma & Zwart]	FCFS weakly optimal [Boxma & Zwart]
2021	Nudge [Grosof et al]	
2022	Nudge-K [Van Houdt]	
2024	Nudge-M [Charlet & Van Houdt]	
	γ-Boost [Yu & Scully, SIGMETRICS 2024]	
Now		γ-Gittins [**Harlev**, Yu & Scully]

strongly optimal!

strongly optimal!

Our contributions

New policy: γ-Gittins!

Theorem

γ-Gittins is strongly optimal,

$$C_{\gamma\text{-Gittins}} = \inf_{\pi} C_{\pi}$$

Our contributions

New policy: γ-Gittins!

Theorem

γ-Gittins is strongly optimal,

$$C_{\gamma\text{-Gittins}} = \inf_{\pi} C_{\pi}$$

$$\mathbf{P}[T_{\pi} > x] \sim C_{\pi} e^{-\gamma x} \quad (x \to \infty)$$

Our contributions

New policy: γ-Gittins!

Theorem

γ-Gittins is strongly optimal,

$$C_{\gamma\text{-Gittins}} = \inf_{\pi} C_{\pi}$$

$$\mathbf{P}[T_{\pi} > x] \sim C_{\pi} e^{-\gamma x} \quad (x \to \infty)$$

Practice: what are the design takeaways?

Our contributions

New policy: γ-Gittins!

Theorem

γ-Gittins is strongly optimal,

$$C_{\gamma\text{-Gittins}} = \inf_{\pi} C_{\pi}$$

$$\mathbf{P}[T_{\pi} > x] \sim C_{\pi} e^{-\gamma x} \quad (x \to \infty)$$

Practice: what are the design takeaways?

Theory: why was this hard to discover?

Our contributions

New policy: γ-Gittins!

Theorem

γ-Gittins is strongly optimal,

$$C_{\gamma\text{-Gittins}} = \inf_{\pi} C_{\pi}$$

$$\mathbf{P}[T_{\pi} > x] \sim C_{\pi} e^{-\gamma x} \quad (x \to \infty)$$

Practice: what are the design takeaways?

Theory: why was this hard to discover?

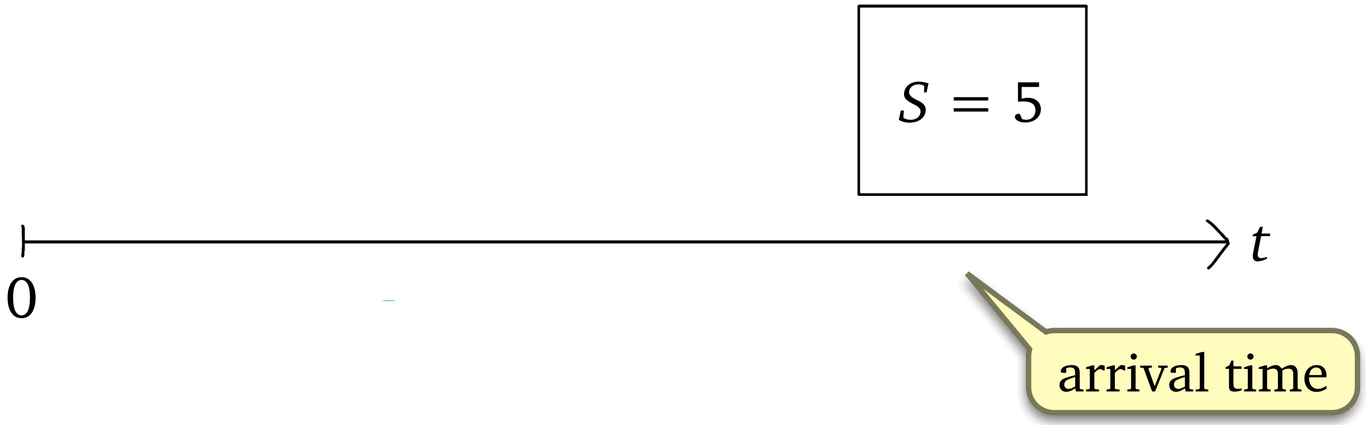
What is γ-Gittins?

What is γ-Gittins?

What is a boost policy?

What is γ-Gittins?

What is a boost policy?

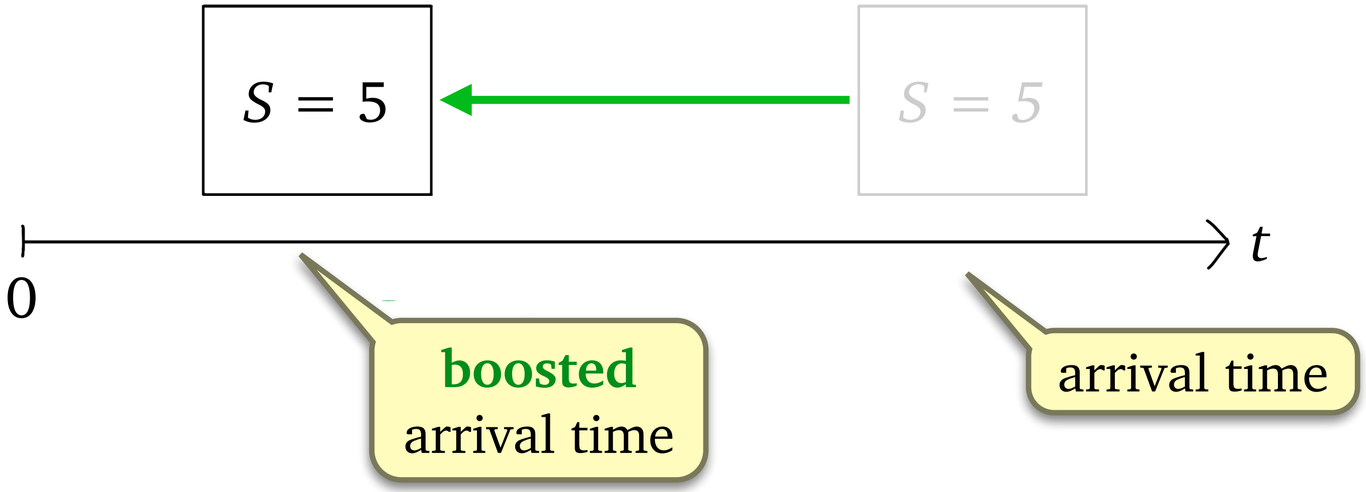


boosted arrival time

= arrival time - **boost**(job)

What is γ-Gittins?

What is a boost policy?



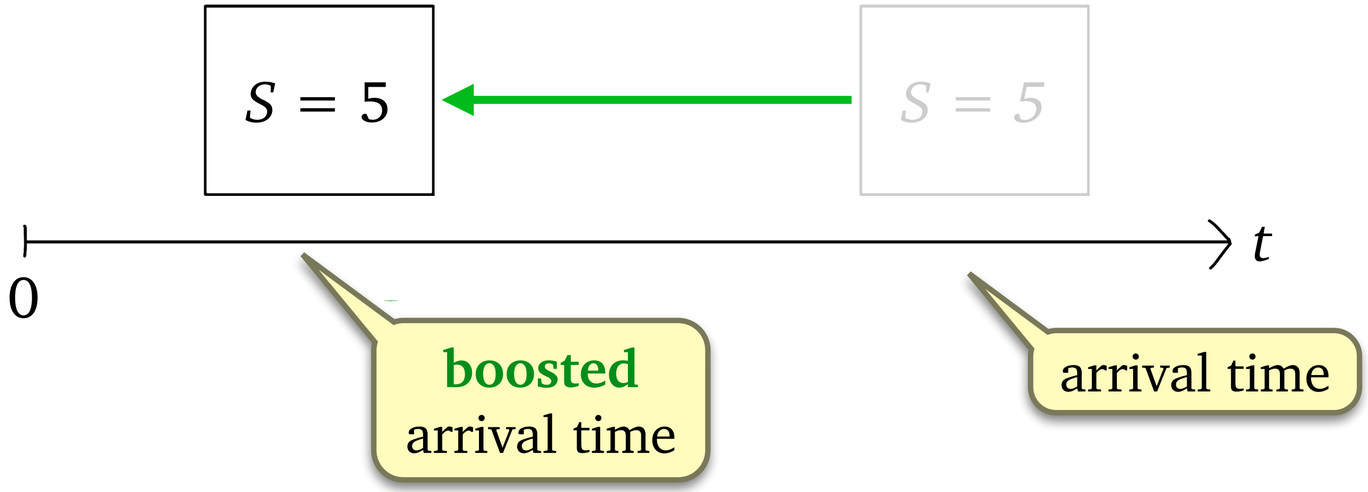
boosted arrival time

= arrival time - **boost**(job)

What is γ-Gittins?

What is a boost policy?

boost policy: *serve jobs in order of* ***boosted*** *arrival time*

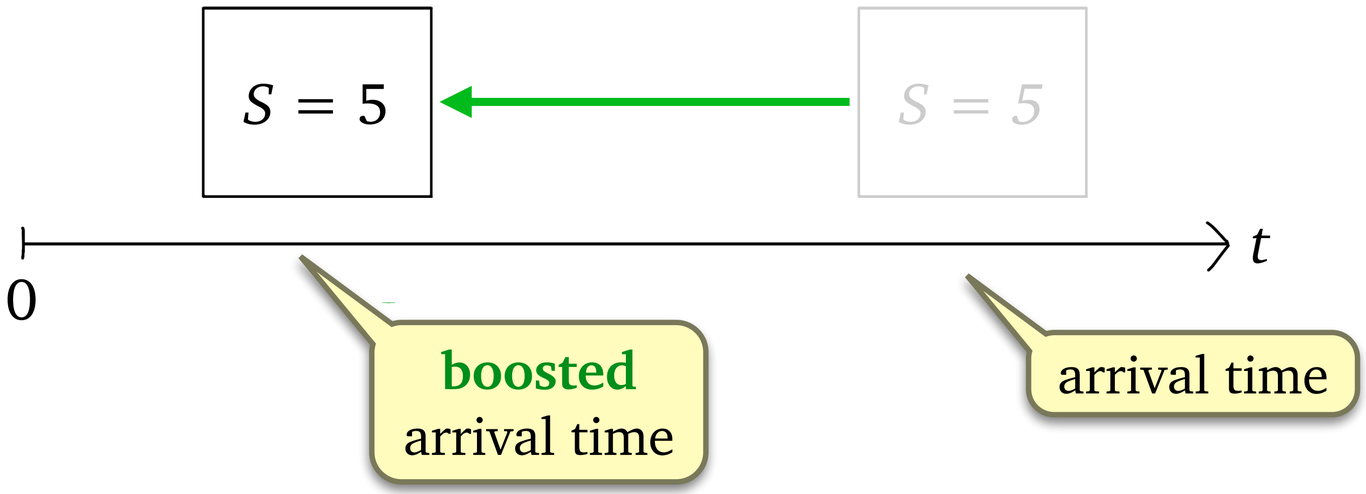


boosted arrival time
= arrival time - **boost**(job)

What is γ-Gittins?

What is a boost policy?

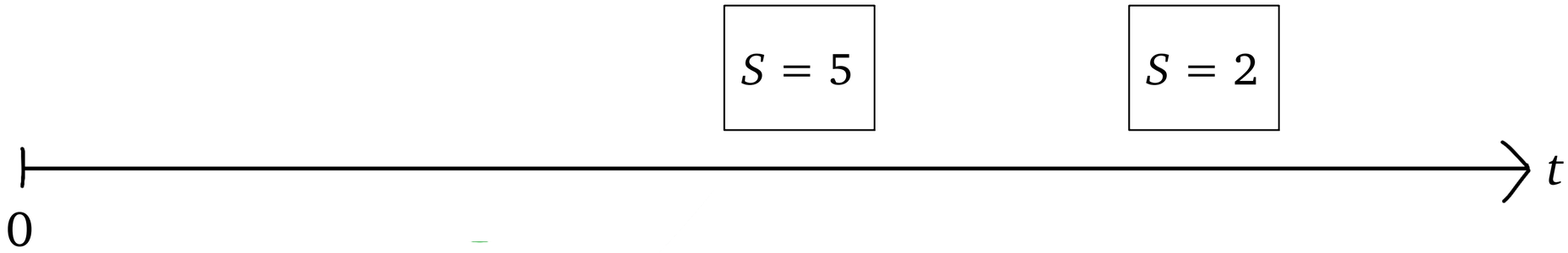
boost policy: *serve jobs in order of* ***boosted*** *arrival time*



boost(size) in
[Yu & Scully, 2024]

boosted arrival time
= arrival time - **boost**(job)

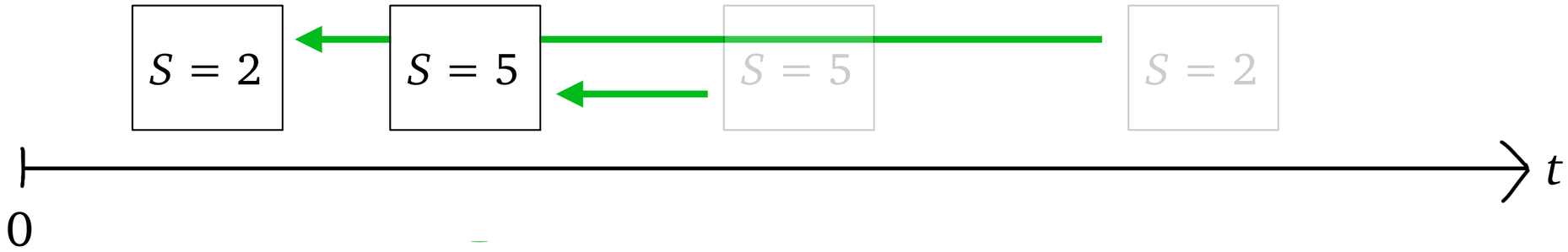
What is γ-Gittins?



boosted arrival time

= arrival time - **boost**(job)

What is γ-Gittins?



boosted arrival time

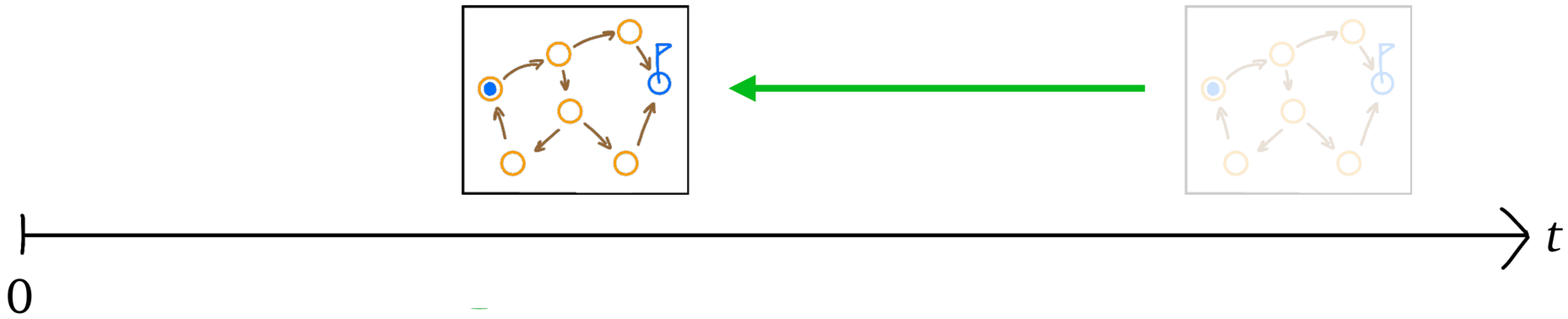
= arrival time - **boost**(job)

What is γ-Gittins?

boosted arrival time

= arrival time - **boost**(job)

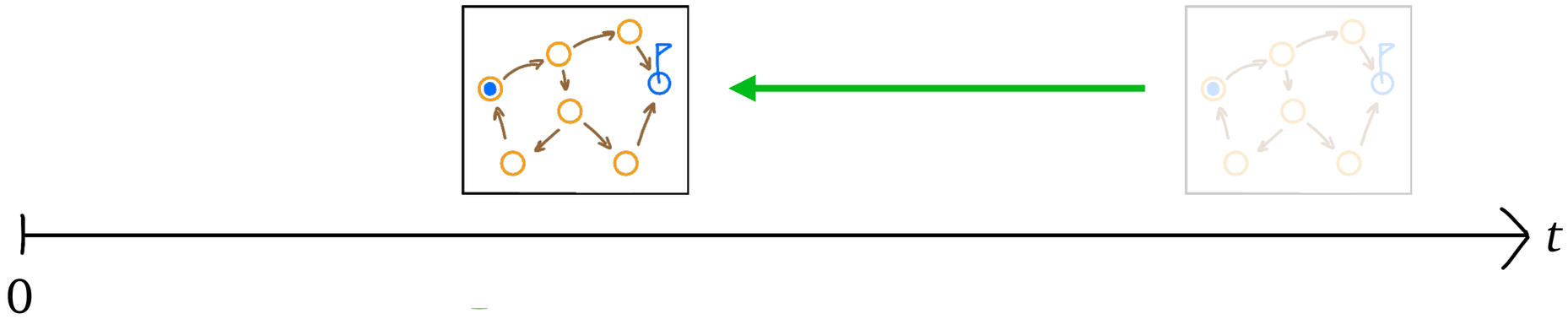
What is γ-Gittins?



boosted arrival time

= arrival time - **boost**(job)

What is γ-Gittins?

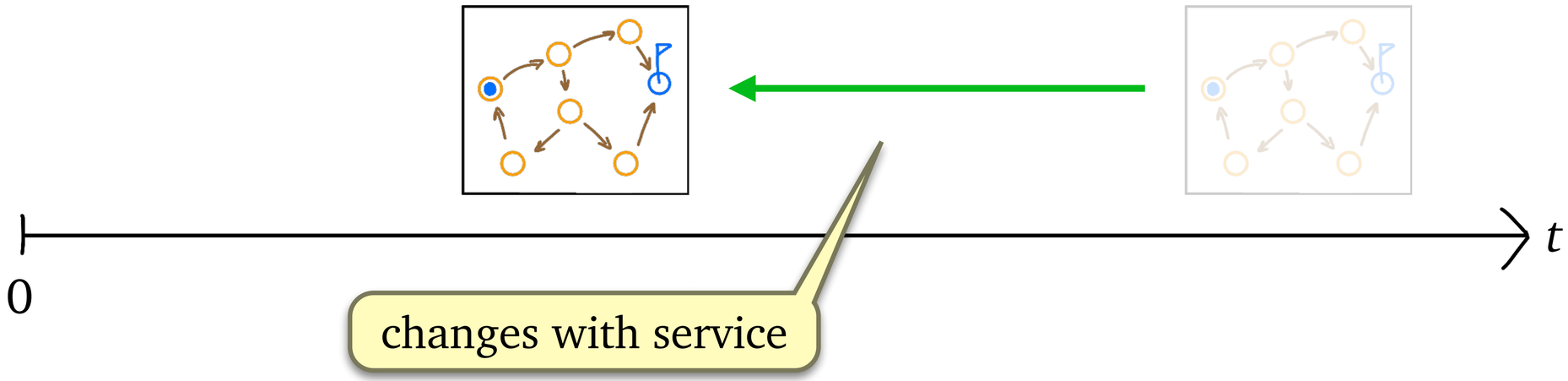


boosted arrival time

= arrival time - **boost**(job)

boost(state)

What is γ-Gittins?

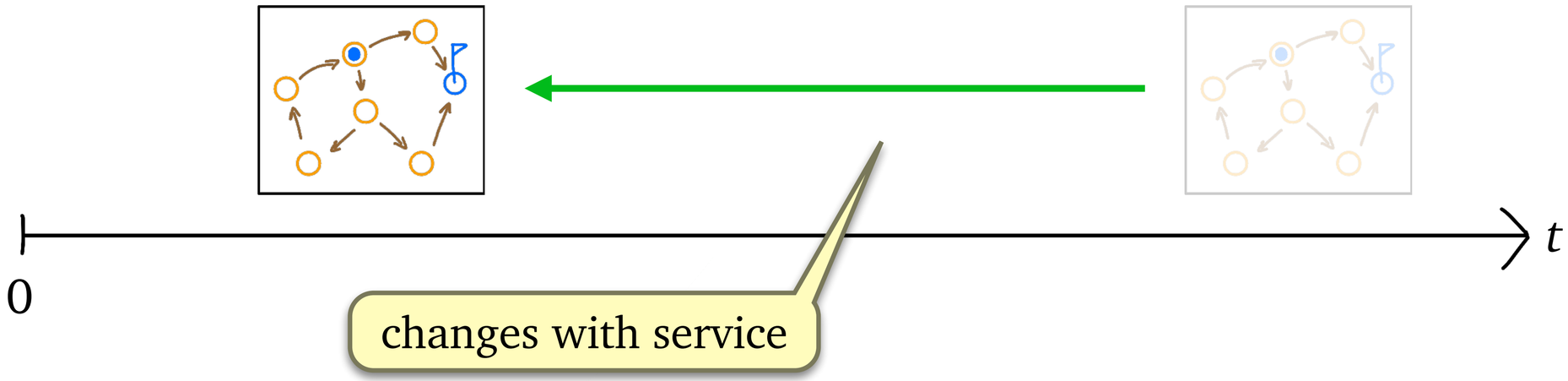


boosted arrival time

= arrival time - **boost**(job)

boost(state)

What is γ-Gittins?

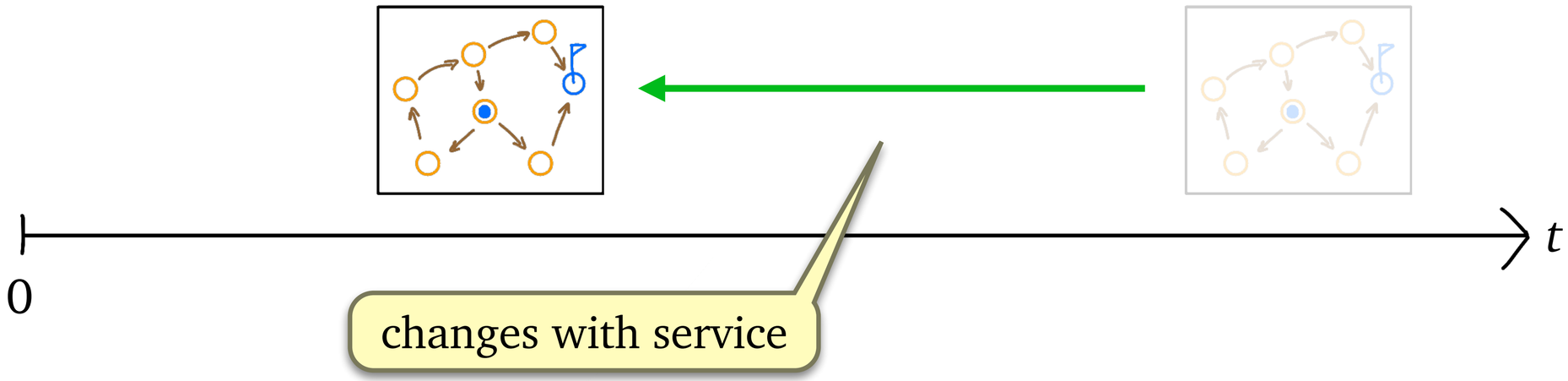


boosted arrival time

= arrival time - **boost**(job)

boost(state)

What is γ-Gittins?

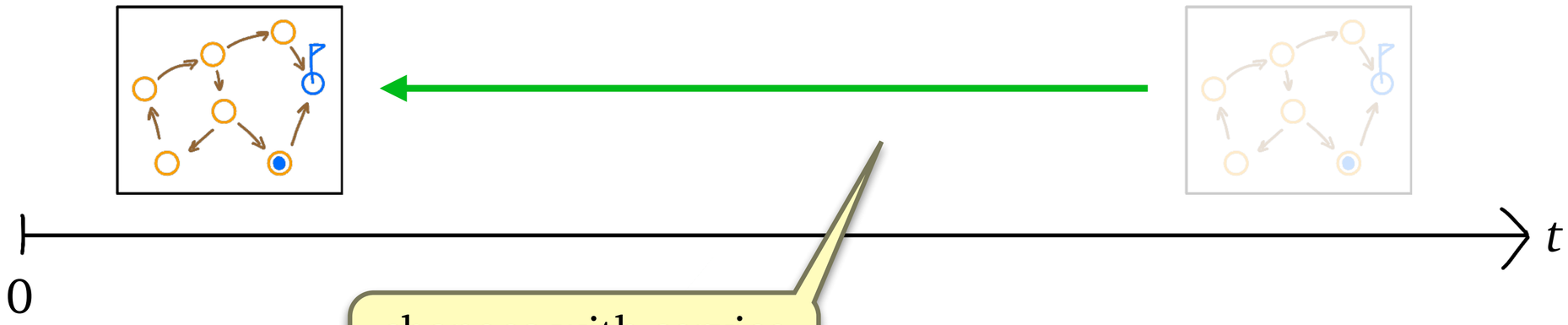


boosted arrival time

= arrival time - **boost**(job)

boost(state)

What is γ-Gittins?

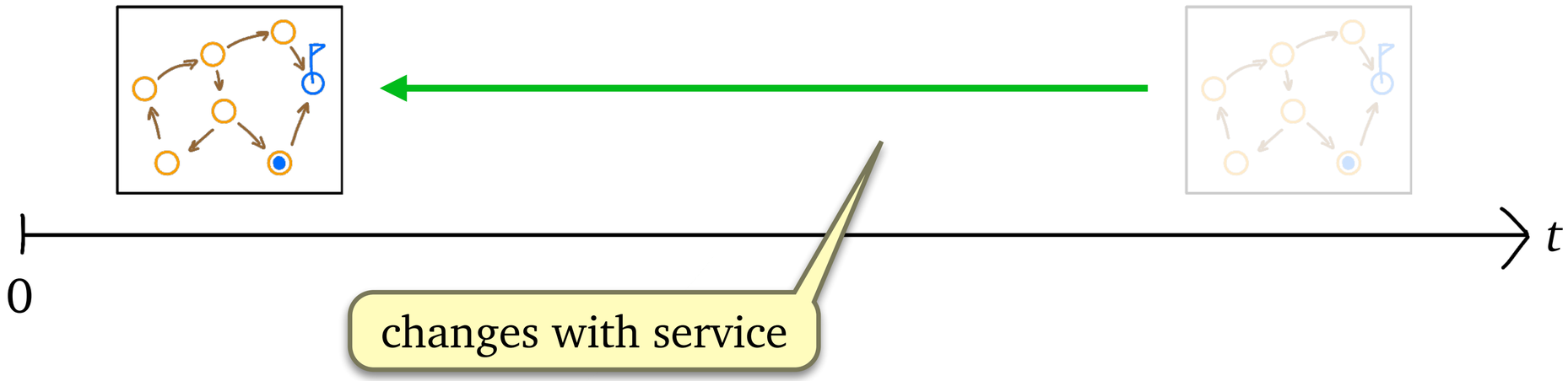


boosted arrival time

= arrival time - **boost**(job)

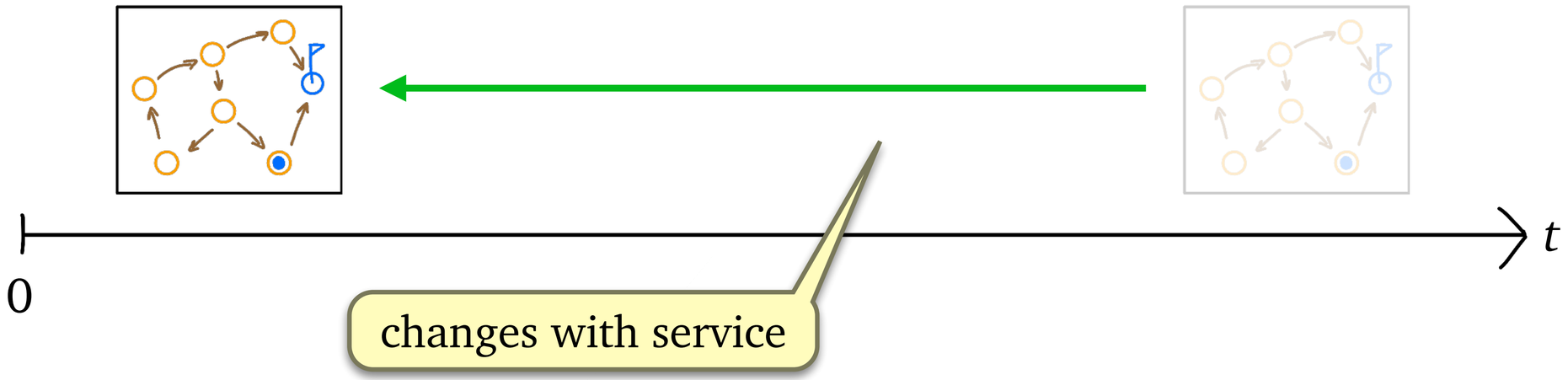
boost(state)

What is γ-Gittins?



γ-Gittins: policy with **boost**(state) $= \frac{1}{\gamma} \log(\text{Gittins index})$

What is γ-Gittins?



γ-Gittins: policy with **boost**(state) $= \frac{1}{\gamma} \log(\text{Gittins index})$

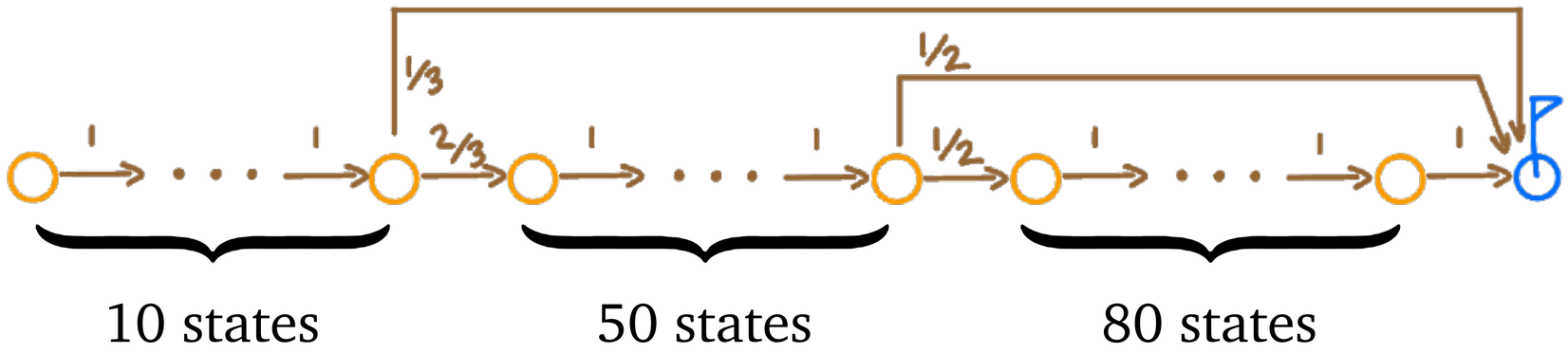
from multi-armed bandit literature
(but discounting → inflation)

What does priority under γ-Gittins look like?

What does priority under γ-Gittins look like?

(Classical) Unknown Size

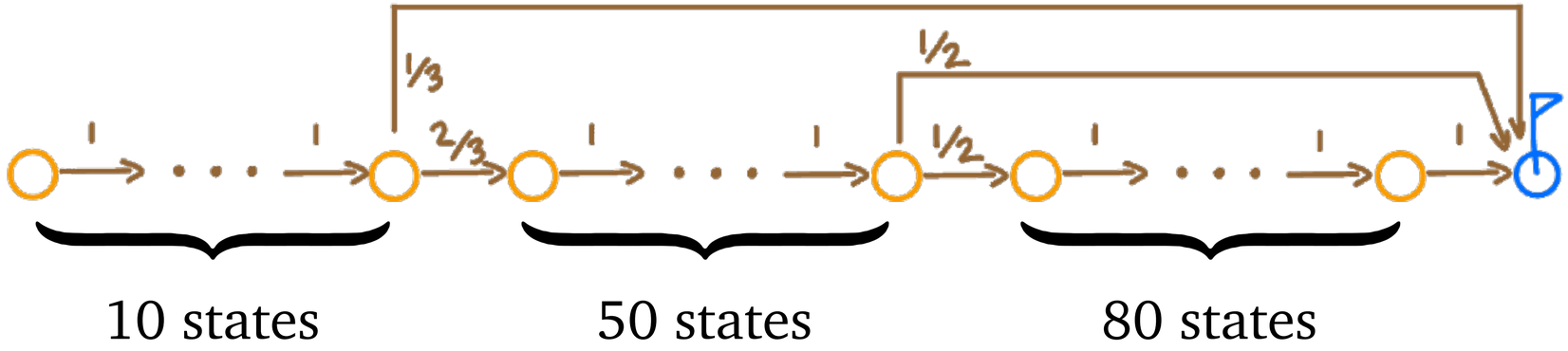
$S \sim$ **Unif**$\{10, 60, 140\}$

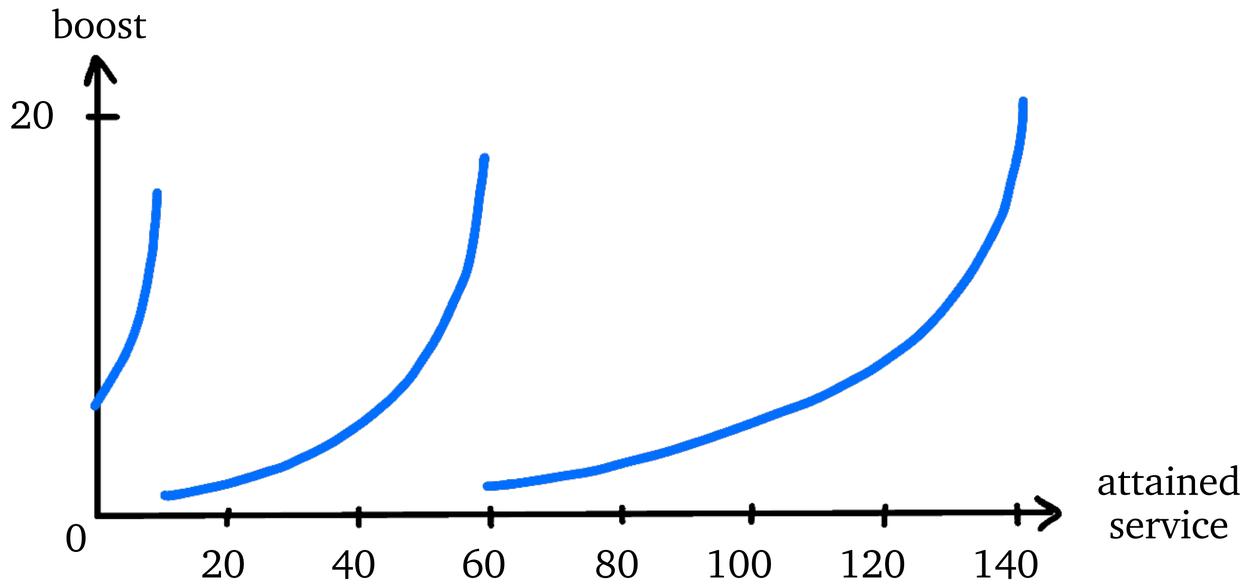


What does priority under γ-Gittins look like?

(Classical) Unknown Size

$S \sim$ **Unif**$\{10, 60, 140\}$

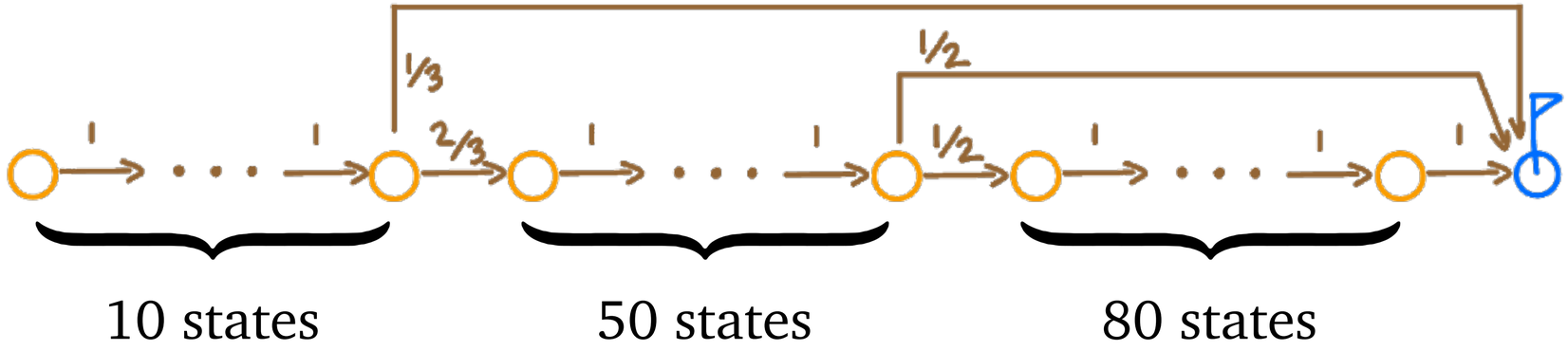


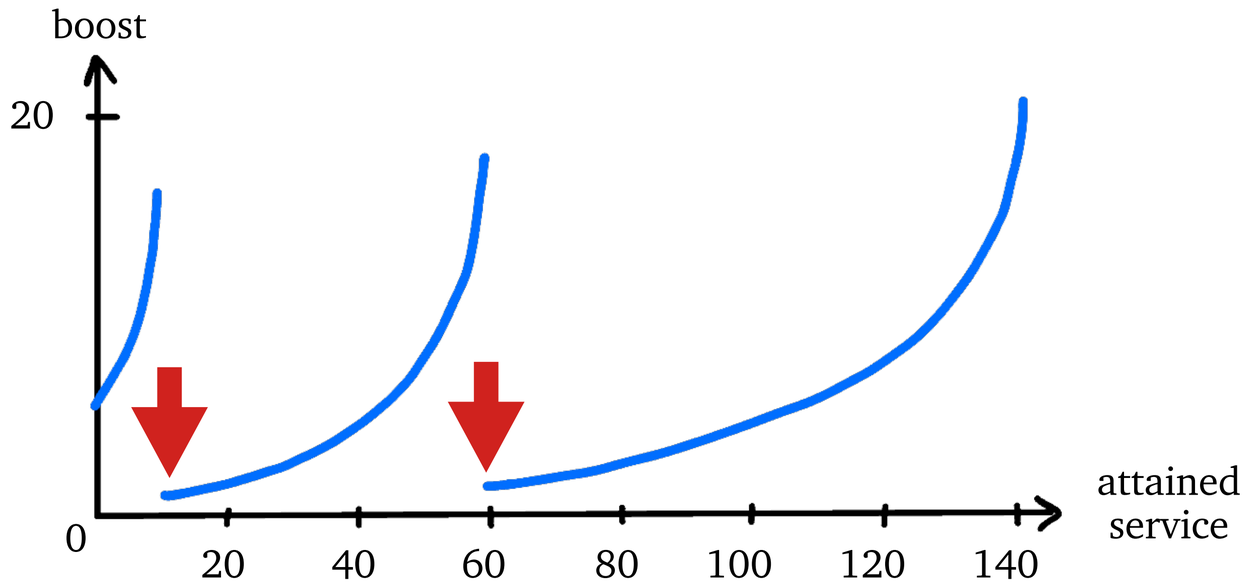


What does priority under γ-Gittins look like?

(Classical) Unknown Size

$S \sim$ **Unif**$\{10, 60, 140\}$

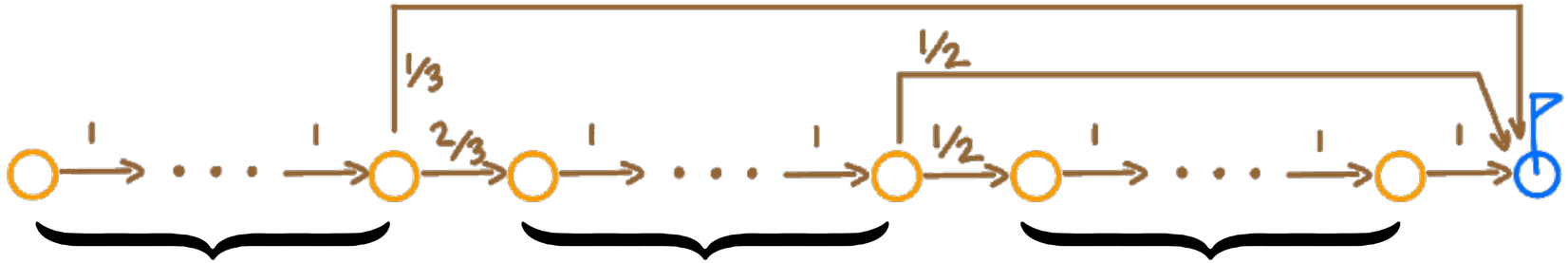


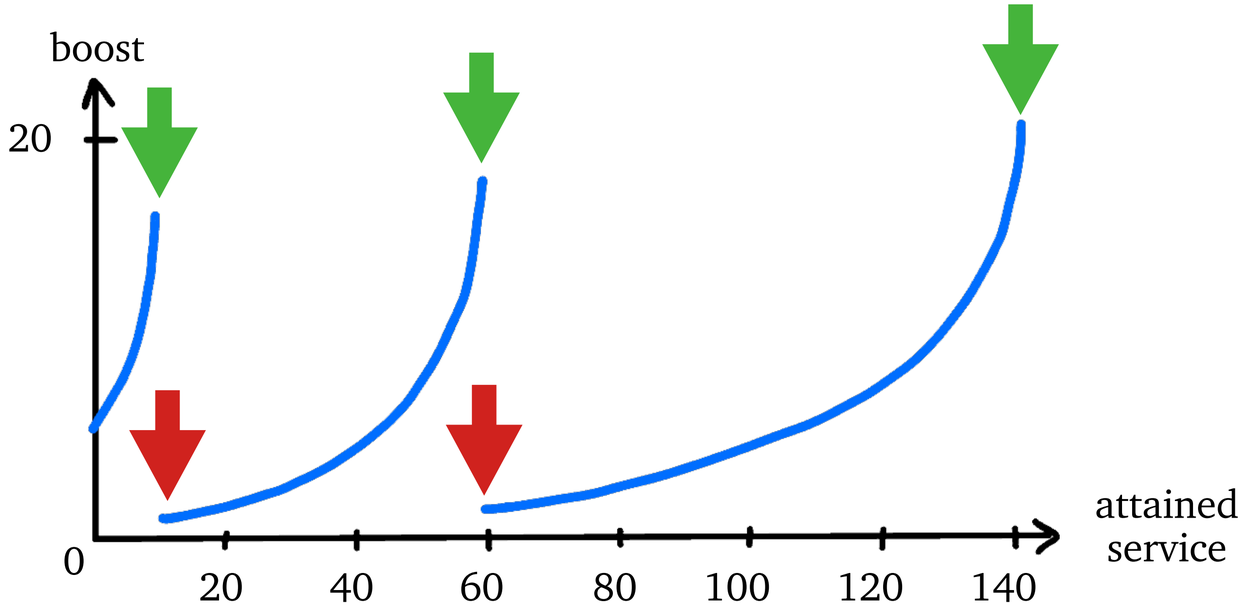


What does priority under γ-Gittins look like?

(Classical) Unknown Size

$S \sim$ **Unif**$\{10, 60, 140\}$





What does priority under γ-Gittins look like?

Partial Information

$S \sim \textbf{Unif}\{10, 60, 140\}$

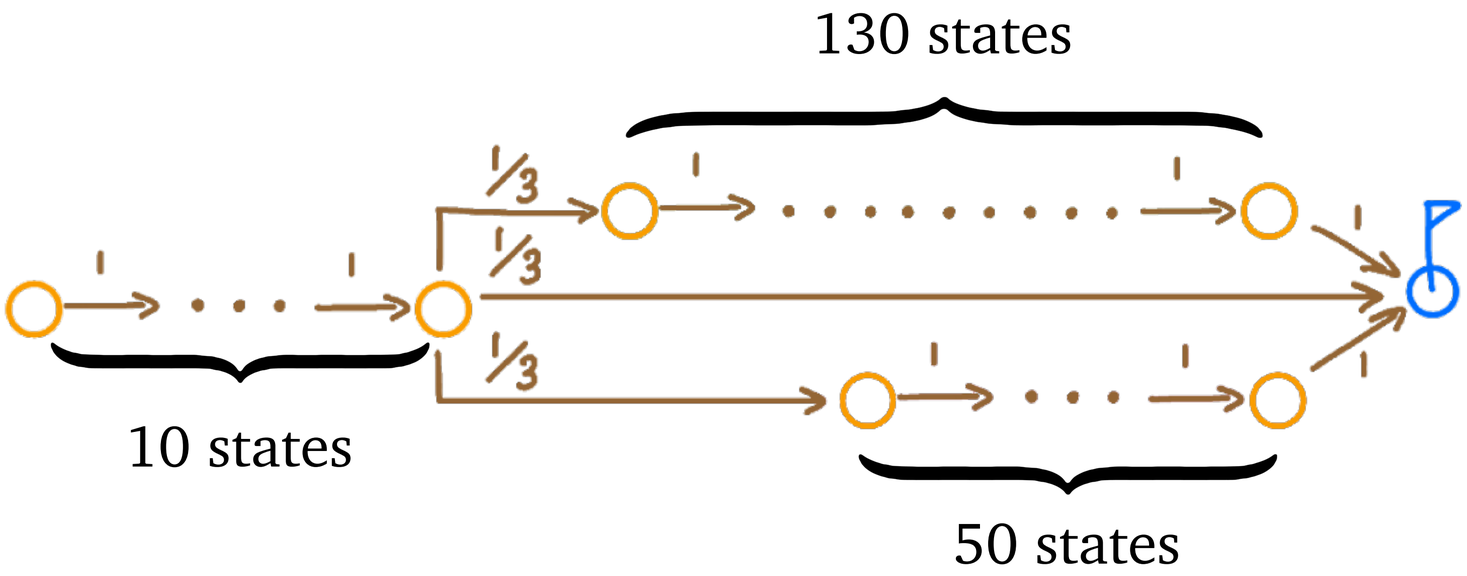
w/ size known after 10 units service

What does priority under γ-Gittins look like?

Partial Information

$S \sim$ **Unif**$\{10, 60, 140\}$

w/ size known after 10 units service

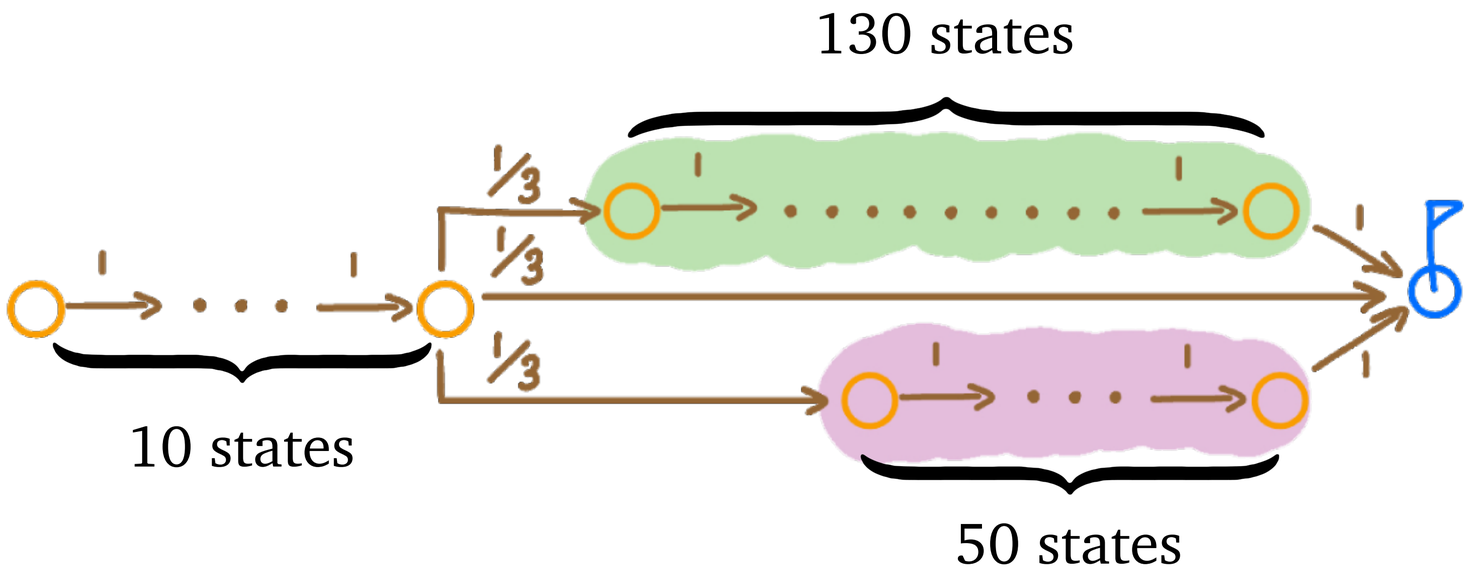


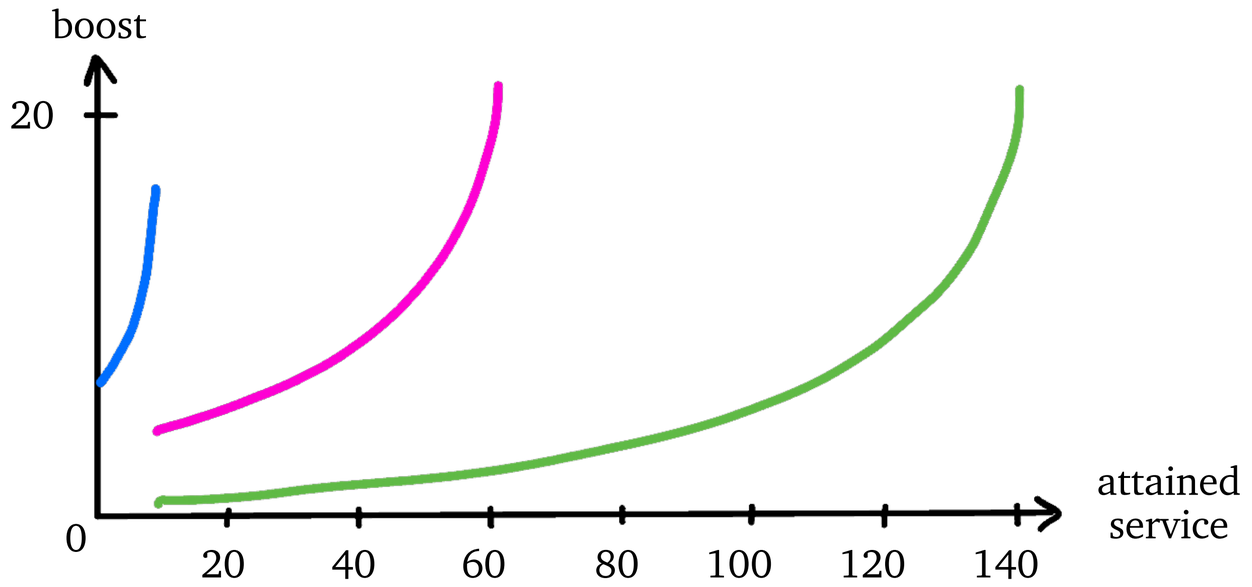
What does priority under γ-Gittins look like?

Partial Information

$S \sim$ **Unif**$\{10, 60, 140\}$

w/ size known after 10 units service



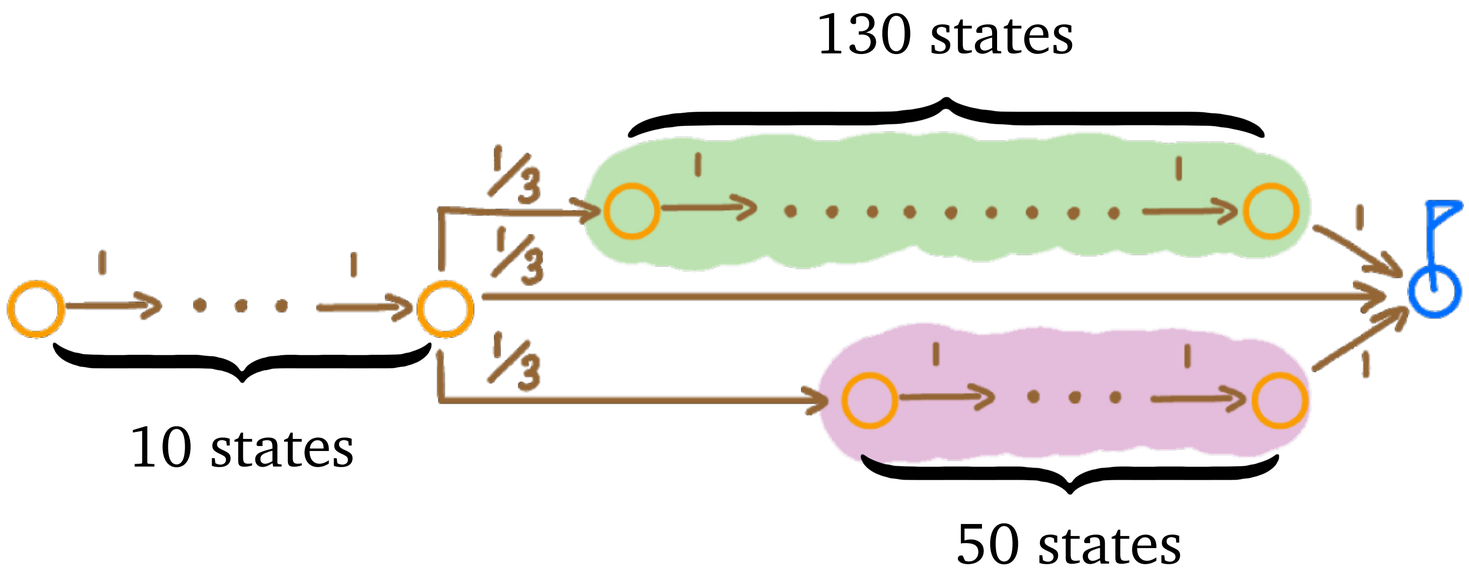


What does priority under γ-Gittins look like?

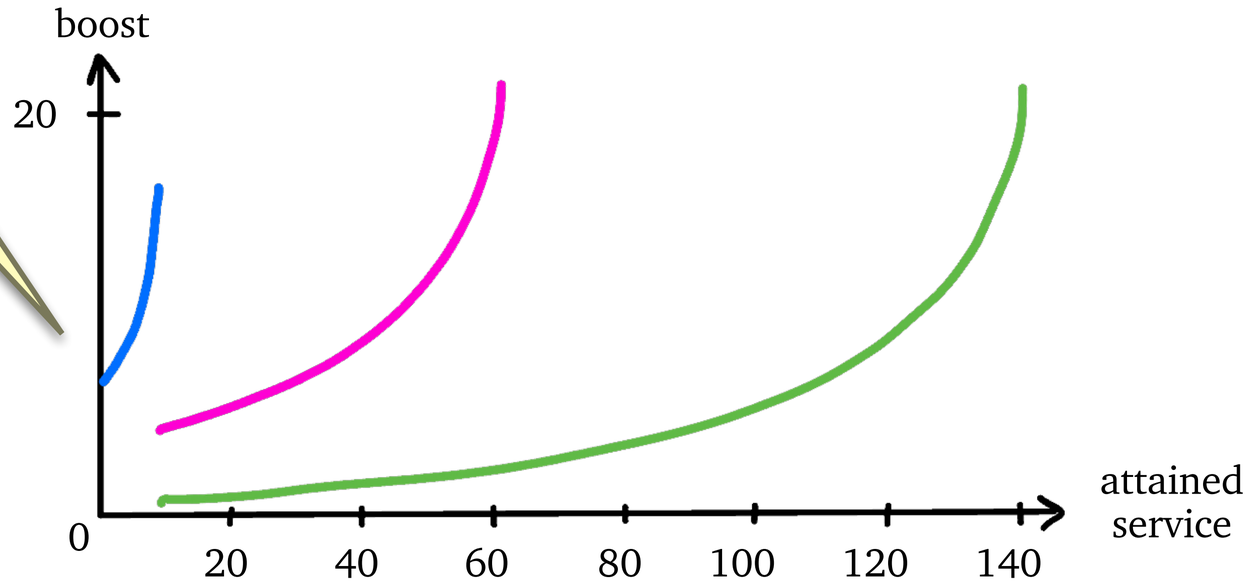
Partial Information

$S \sim$ **Unif**$\{10, 60, 140\}$

w/ size known after 10 units service

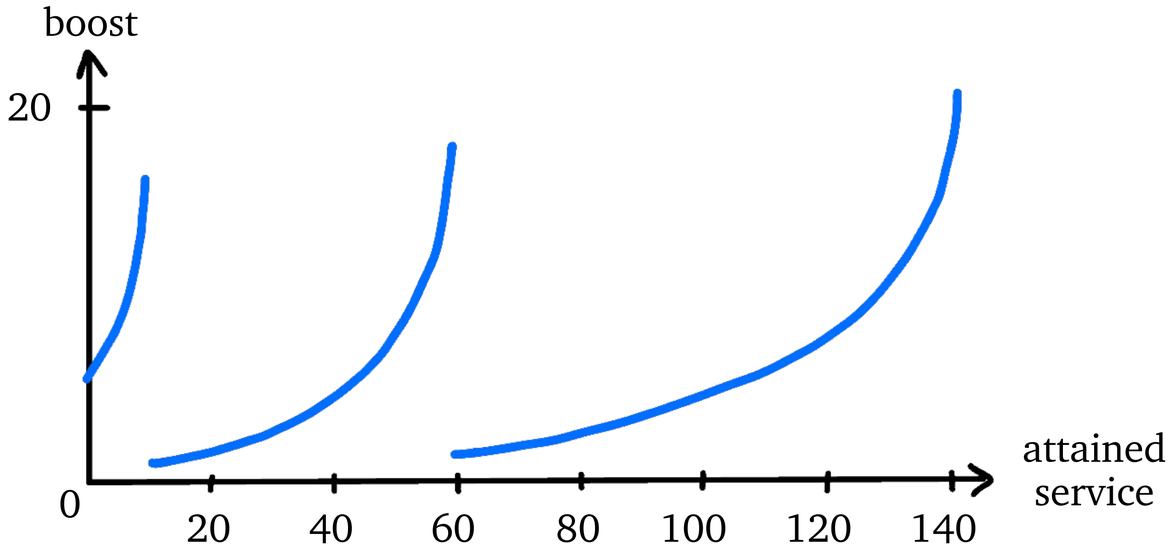


if this seems too complicated…

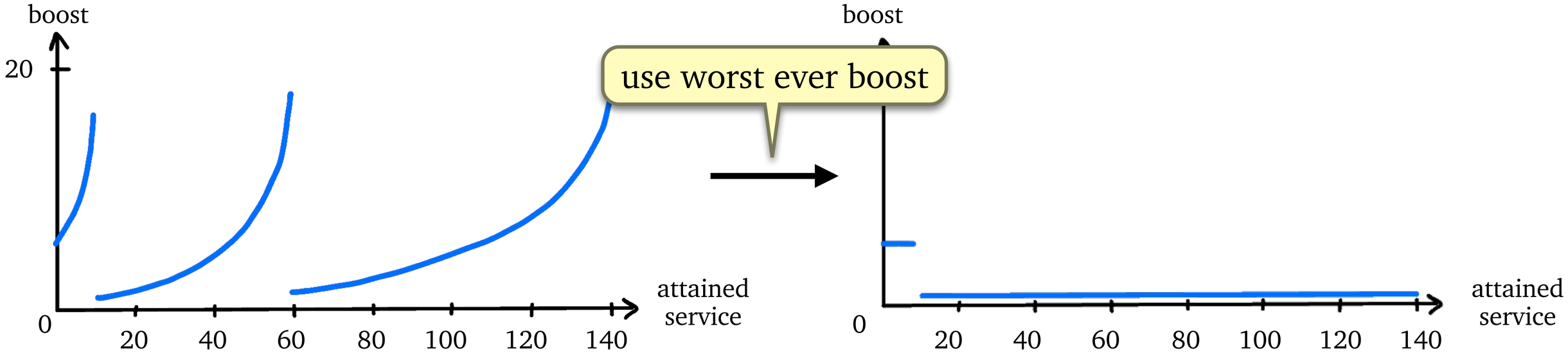


We can ignore boost increases

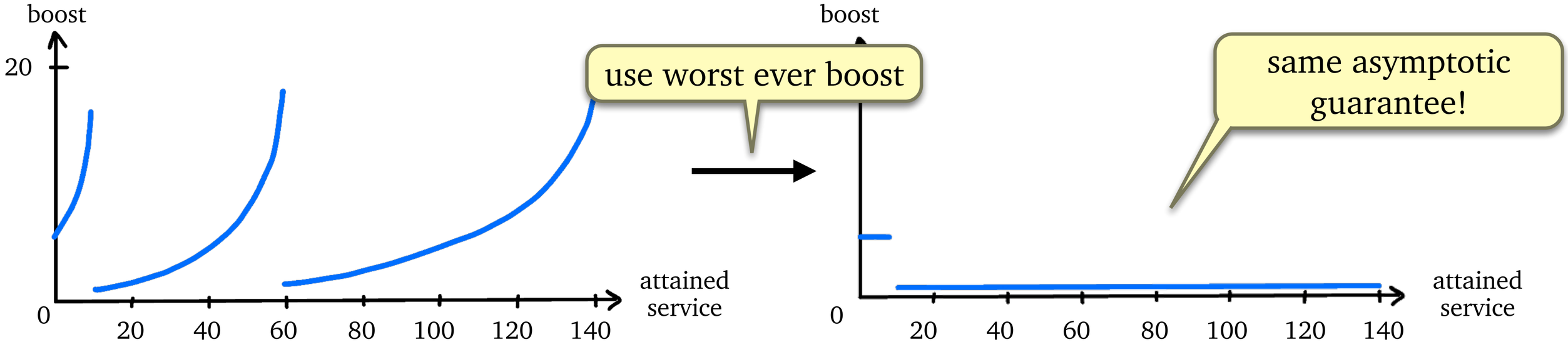
We can ignore boost increases



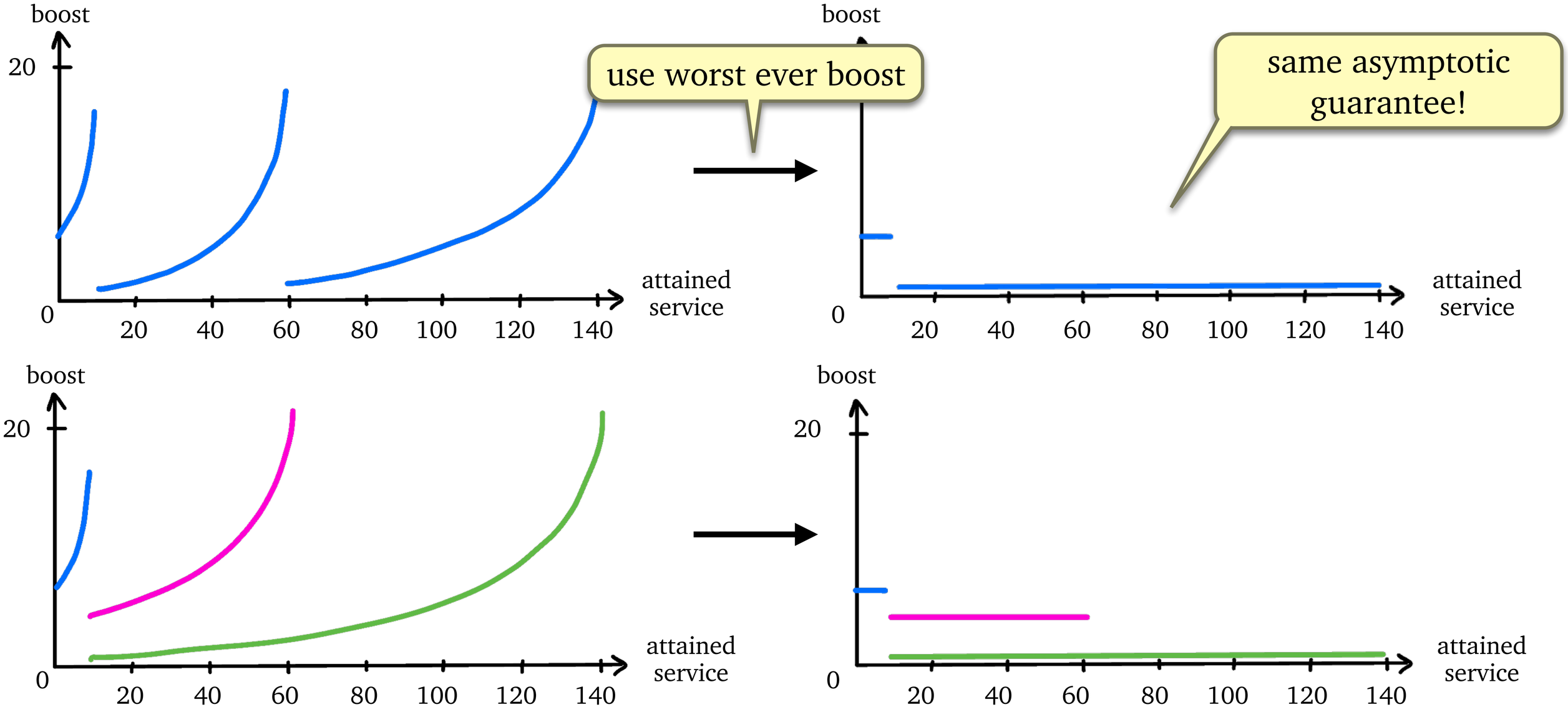
We can ignore boost increases



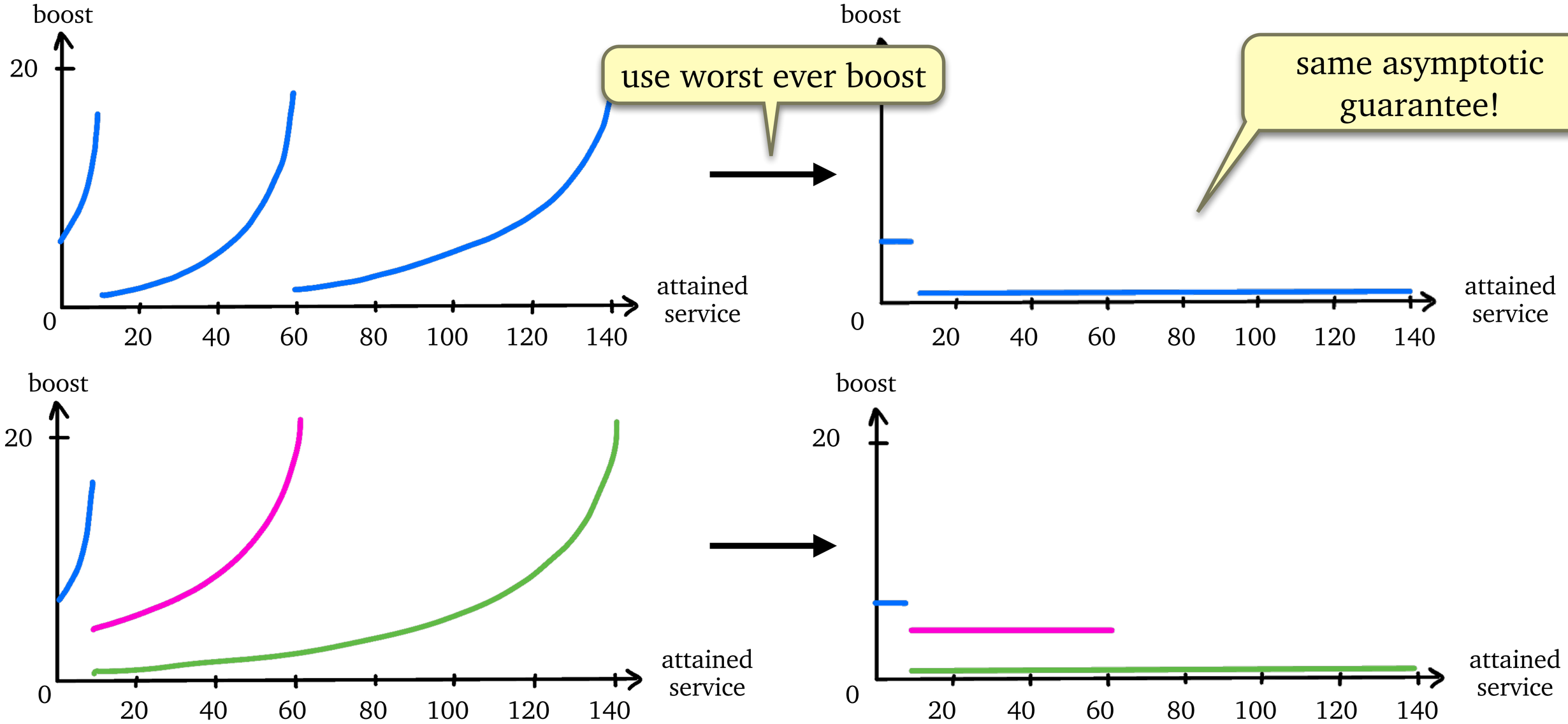
We can ignore boost increases



We can ignore boost increases



We can ignore boost increases



What are the takeaways for designing systems?

High level takeaways for designing systems

If you want to minimize the tail of response time:

High level takeaways for designing systems

If you want to minimize the tail of response time:

1. priority should depends on both arrival time and job state

High level takeaways for designing systems

If you want to minimize the tail of response time:

1. priority should depends on both arrival time and job state
2. priority $\approx$ big initial boost, then decrease based on state

High level takeaways for designing systems

If you want to minimize the tail of response time:

1. priority should depends on both arrival time and job state
2. priority ≈ big initial boost, then decrease based on state
3. priority can also increase based on state, but times where it decreases are more important

High level takeaways for designing systems

If you want to minimize the tail of response time:

1. priority should depends on both arrival time and job state
2. priority $\approx$ big initial boost, then decrease based on state
3. priority can also increase based on state, but times where it decreases are more important
4. can use any information model,
   but more information $\Longrightarrow$ better performance

Our contributions

New policy: γ-Gittins!

Theorem

γ-Gittins is strongly optimal,

$$C_{\gamma\text{-Gittins}} = \inf_{\pi} C_{\pi}$$

Practice: what are the design takeaways?

Our contributions

New policy: γ-Gittins!

Theorem

γ-Gittins is strongly optimal,

$$C_{\gamma\text{-Gittins}} = \inf_{\pi} C_{\pi}$$

Practice: what are the design takeaways?

Theory: why was this hard to discover?

Our contributions

New policy: γ-Gittins!

$$\text{boost} = \frac{1}{\gamma}\log(\text{Gittins index})$$

Theorem

γ-Gittins is strongly optimal,

$$C_{\gamma\text{-Gittins}} = \inf_{\pi} C_{\pi}$$

Practice: what are the design takeaways?

Theory: why was this hard to discover?

Our contributions

New policy: γ-Gittins!

boost $= \frac{1}{\gamma}\log(\text{Gittins index})$

Theorem

γ-Gittins is strongly optimal,

$$C_{\gamma\text{-Gittins}} = \inf_{\pi} C_{\pi}$$

Why are we only realizing that Gittins is good for tails now?

Practice: what are the design takeaways?

Theory: why was this hard to discover?

Optimizing tails as a restless bandit problem

Optimizing tails as a restless bandit problem

arm states can change when out of service

Optimizing tails as a restless bandit problem

arm states can change when out of service

Can view $\mathbf{P}[T_\pi > x]$ as:

$\mathbf{E}$[cost of job] when cost is $\mathbf{1}(T_\pi > x)$

Optimizing tails as a restless bandit problem

arm states can change when out of service

Can view $\mathbf{P}[T_\pi > x]$ as:

$\mathbf{E}$[cost of job] when cost is $\mathbf{1}(T_\pi > x)$

restless process

Optimizing tails as a restless bandit problem

arm states can change when out of service

Can view $\mathbf{P}[T_\pi > x]$ as:

$\mathbf{E}$[cost of job] when cost is $\mathbf{1}(T_\pi > x)$

restless process

restless bandits are hard

Optimizing tails as a restless bandit problem

arm states can change when out of service

Can view $\mathbf{P}[T_\pi > x]$ as:

$\mathbf{E}$[cost of job] when cost is $\mathbf{1}(T_\pi > x)$

restless process

restless bandits are hard

results typically only for “large-system limit”

Optimizing tails as a restless bandit problem

arm states can change when out of service

Can view $\mathbf{P}[T_\pi > x]$ as:

$\mathbf{E}$[cost of job] when cost is $\mathbf{1}(T_\pi > x)$

restless process

restless bandits are hard

results typically only for “large-system limit”

Gittins was designed specifically for *non-restless* bandits

Optimizing tails as a restless bandit problem

Can view $\mathbf{P}[T_\pi > x]$ as:

arm states can change when out of service

$\mathbf{E}$[cost of job] when cost is $\mathbf{1}(T_\pi > x)$

restless process

restless bandits are hard

results typically only for “large-system limit”

Gittins was designed specifically for *non-restless* bandits

for optimizing asymptotic behavior of $\mathbf{P}[T_\pi > x]$,
correct choice for “cost of job” is $e^{\gamma T_\pi}$

Optimizing tails as a restless bandit problem

arm states can change when out of service

Can view $\mathbf{P}[T_\pi > x]$ as:

$\mathbf{E}$[cost of job] when cost is $\mathbf{1}(T_\pi > x)$

restless process

restless bandits are hard

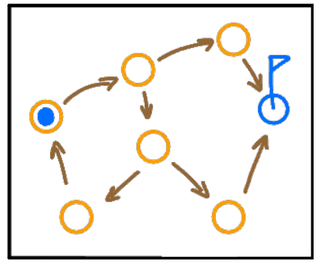
results typically only for "large-system limit"

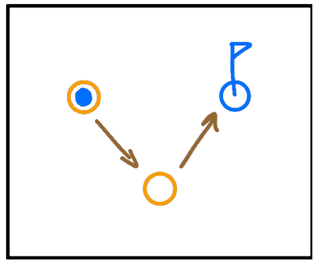
Gittins was designed specifically for *non-restless* bandits

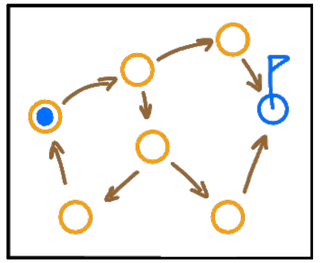
MAB with arrivals
(discounting → inflation)

or optimizing asymptotic behavior of $\mathbf{P}[T_\pi > x]$,
correct choice for "cost of job" is $e^{\gamma T_\pi}$

Optimizing tails as a multi-armed bandit with arrivals

Arm 1: 

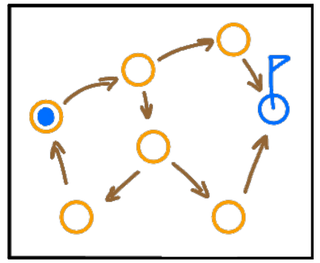
Arm 2: 

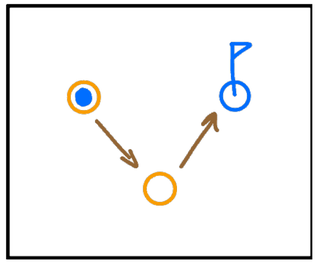
Arm 3: 

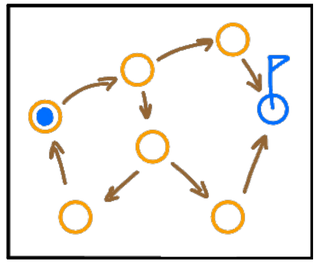
← light-tailed **M/G** random arrivals

Gittins is optimal for multi-armed bandits with arrivals

Optimizing tails as a multi-armed bandit with arrivals

Arm 1: 

Arm 2: 

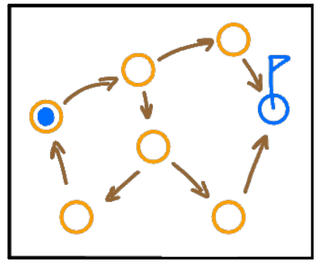
Arm 3: 

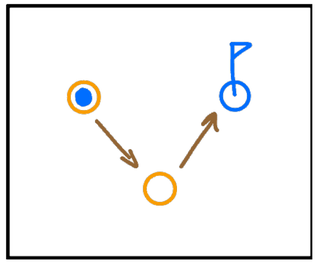
← light-tailed **M/G** random arrivals

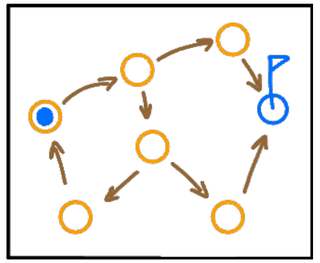
Gittins is optimal for multi-armed bandits with arrivals

…but only if arrivals are time-homogenous

Optimizing tails as a multi-armed bandit with arrivals

Arm 1: 

Arm 2: 

Arm 3: 

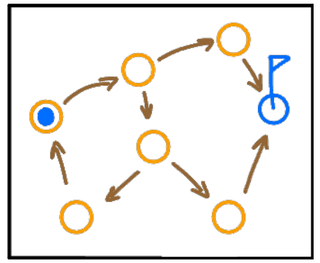
← light-tailed **M/G** random arrivals

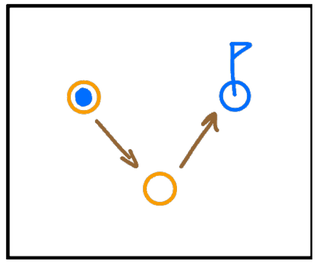
Gittins is optimal for multi-armed bandits with arrivals

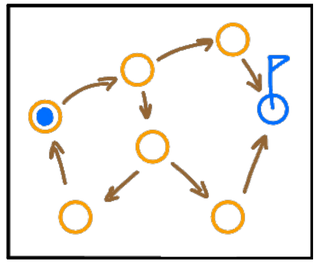
…but only if arrivals are time-homogenous

Goal: minimize $\mathbf{E}[e^{\gamma T_\pi}]$

Optimizing tails as a multi-armed bandit with arrivals

Arm 1: 

Arm 2: 

Arm 3: 

← light-tailed **M/G** random arrivals

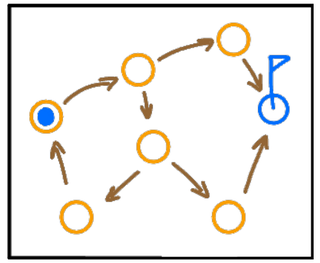
Gittins is optimal for multi-armed bandits with arrivals

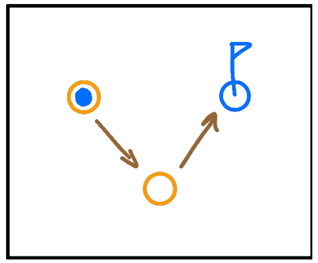
…but only if arrivals are time-homogenous

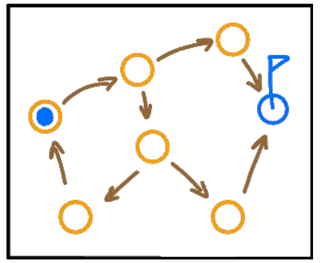
Goal: minimize $\mathbf{E}[e^{\gamma T_\pi}] = \mathbf{E}[e^{-\gamma A} e^{\gamma D_\pi}]$

A: arrival time
D_π: departure time

Optimizing tails as a multi-armed bandit with arrivals

Arm 1: 

Arm 2: 

Arm 3: 

← light-tailed **M/G** random arrivals

Gittins is optimal for multi-armed bandits with arrivals

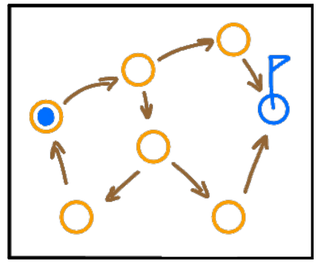
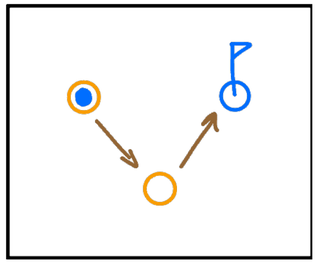
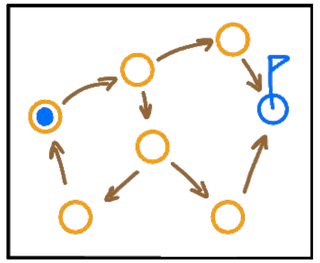
…but only if arrivals are time-homogenous

Goal: minimize $\mathbf{E}[e^{\gamma T_\pi}] = \mathbf{E}[e^{-\gamma A} e^{\gamma D_\pi}]$

A: arrival time
D_π: departure time

$e^{-\gamma A}$: job-specific cost
$e^{\gamma D_\pi}$: inflation factor

Optimizing tails as a multi-armed bandit with arrivals

Arm 1:  Arm 2:  Arm 3: 

← light-tailed **M/G** random arrivals

Gittins is optimal for multi-armed bandits with arrivals

…but only if arrivals are time-homogenous

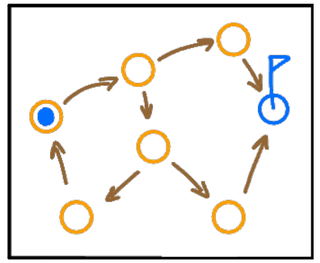
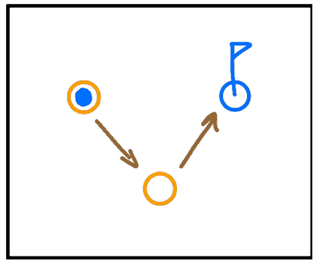
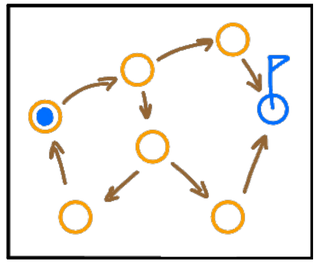
Goal: minimize $\mathbf{E}[e^{\gamma T_\pi}] = \mathbf{E}[e^{-\gamma A} e^{\gamma D_\pi}]$

A: arrival time
D_π: departure time

$e^{-\gamma A}$: job-specific cost
$e^{\gamma D_\pi}$: inflation factor

job-specific costs depends on arrival time → *time-inhomogeneous arrivals*

Optimizing tails as a multi-armed bandit with arrivals

Arm 1:  Arm 2:  Arm 3: 

← light-tailed **M/G** random arrivals

Gittins is optimal for multi-armed bandits with arrivals

…but only if arrivals are time-homogenous

Goal: minimize $\mathbf{E}[e^{\gamma T_\pi}] = \mathbf{E}[e^{-\gamma A} e^{\gamma D_\pi}]$

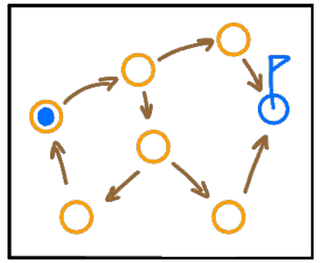
doesn't seem like Gittins should work…

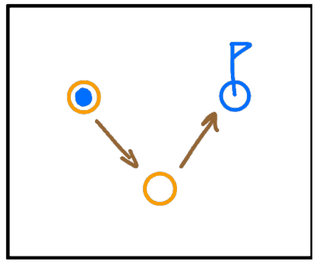
A: arrival time
D_π: departure time

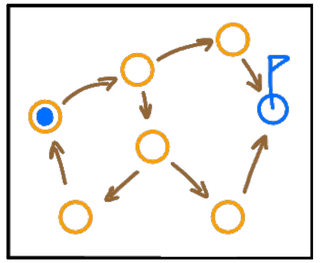
$e^{-\gamma A}$: job-specific cost
$e^{\gamma D_\pi}$: inflation factor

job-specific costs depends on arrival time → *time-inhomogeneous arrivals*

Optimizing tails as a multi-armed bandit with arrivals

Arm 1: 

Arm 2: 

Arm 3: 

← light-tailed **M/G** random arrivals

Gittins is opti with arrivals

[Yu & Scully, 2024]:
can pretend
there are no arrivals

re time-homogenous

Goal: minimize $\mathbf{E}[e^{\gamma T_\pi}] = \mathbf{E}[e^{-\gamma A} e^{\gamma D_\pi}]$

doesn't seem like Gittins should work…

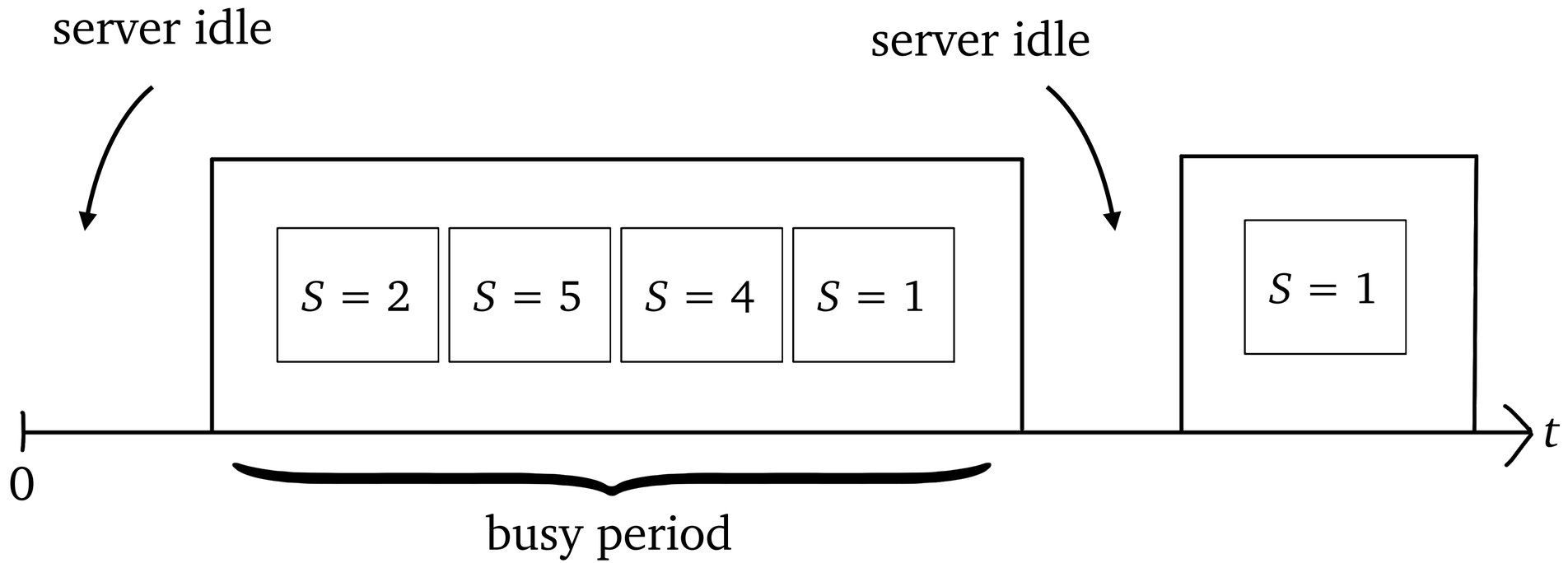
A: arrival time
D_π: departure time

$e^{-\gamma A}$: job-specific cost
$e^{\gamma D_\pi}$: inflation factor

job-specific costs depends on arrival time → *time-inhomogeneous arrivals*

Ignoring arrivals in the γ-Boost analysis

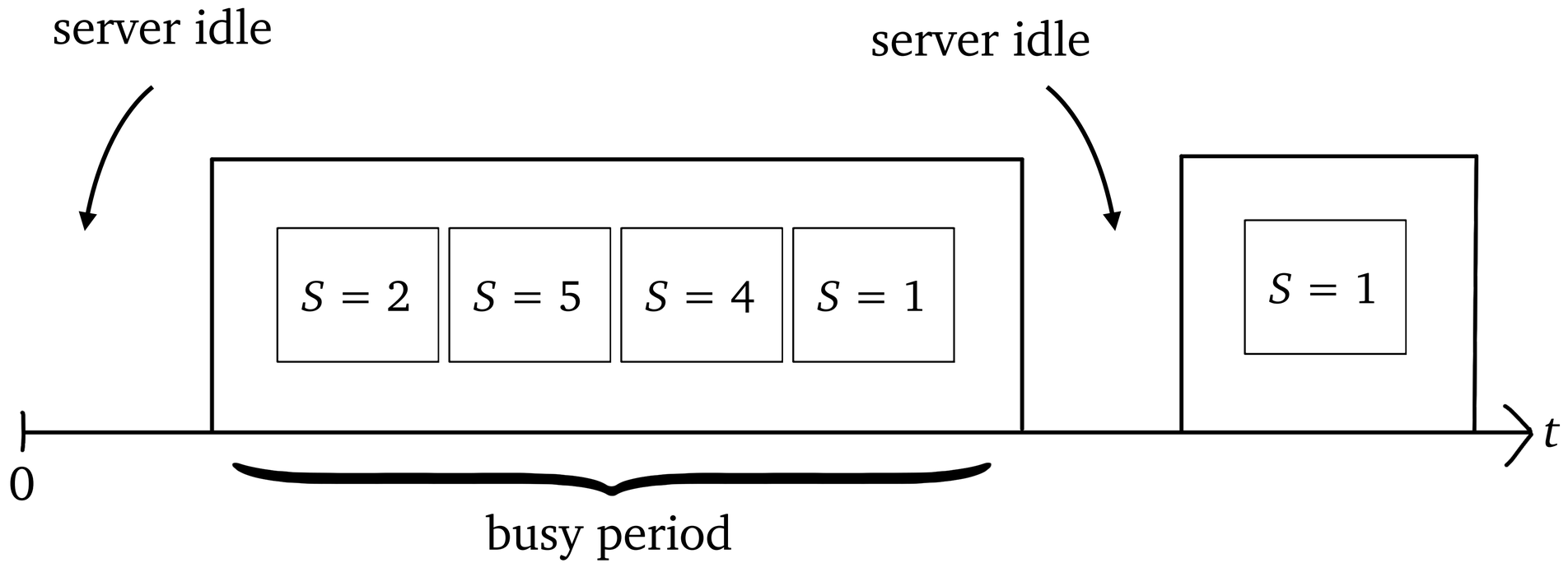
[Yu & Scully, 2024]



batch problem: $S = 2$ | $S = 5$ | $S = 4$ | $S = 1$

Ignoring arrivals in the γ-Boost analysis

[Yu & Scully, 2024]

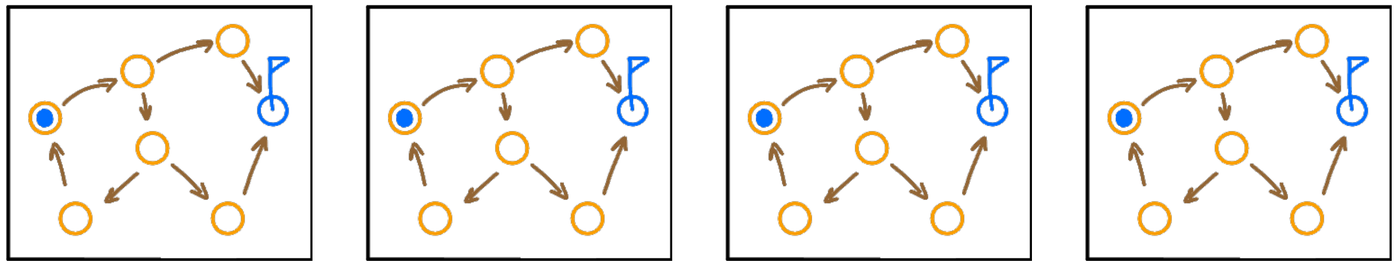


batch problem: $S = 2$ $S = 5$ $S = 4$ $S = 1$

γ-Boost solves the batch problem!

Ignoring arrivals in the γ-Gittins analysis?

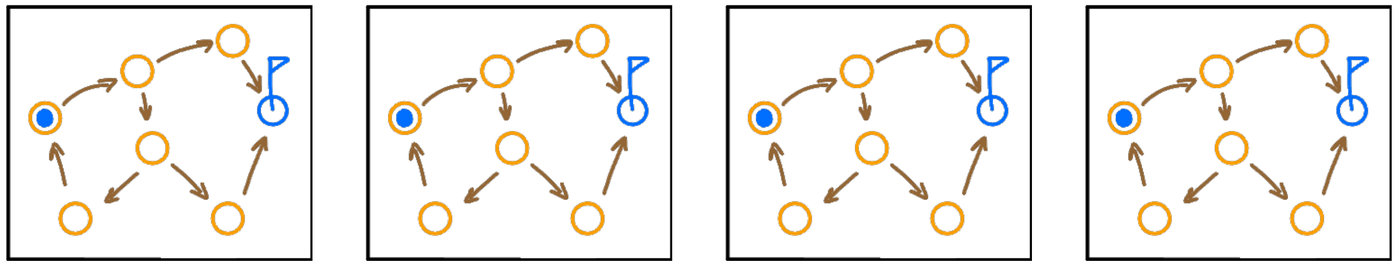
batch problem:



Goal: minimize $\mathbf{E}[e^{\gamma T_\pi}] = \mathbf{E}[e^{-\gamma A} e^{\gamma D_\pi}]$

Ignoring arrivals in the γ-Gittins analysis?

batch problem:

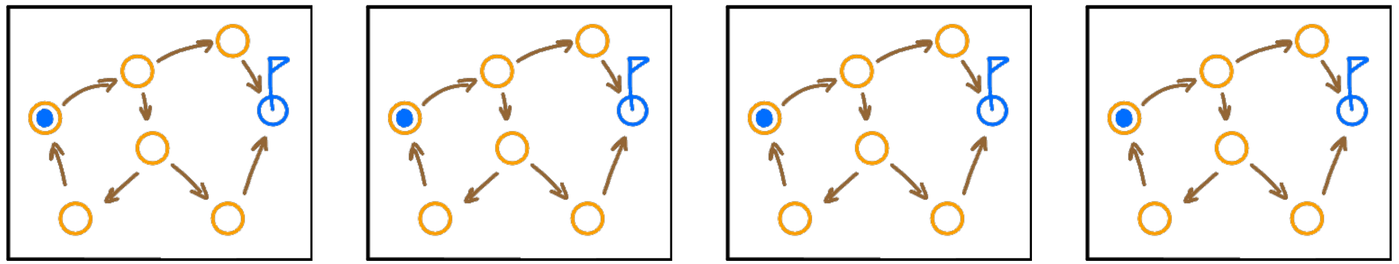


Goal: minimize $\mathbf{E}[e^{\gamma T_\pi}] = \mathbf{E}[e^{-\gamma A} e^{\gamma D_\pi}]$

multi-armed bandit problem

Ignoring arrivals in the γ-Gittins analysis?

batch problem:



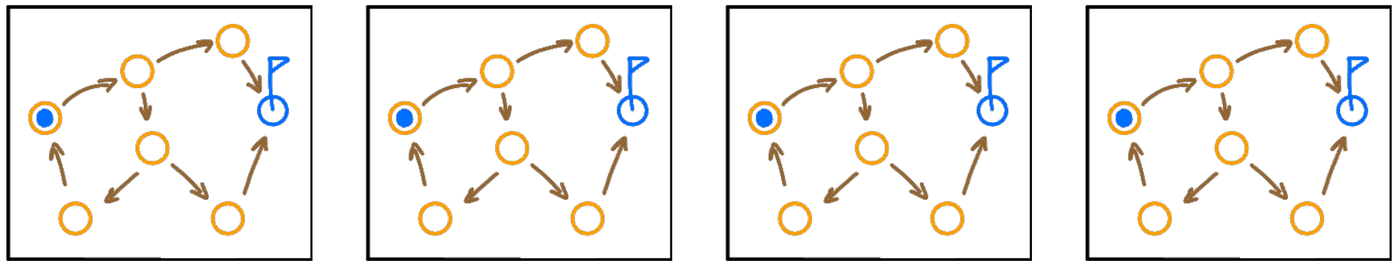
Goal: minimize $\mathbf{E}[e^{\gamma T_\pi}] = \mathbf{E}[e^{-\gamma A} e^{\gamma D_\pi}]$

multi-armed bandit problem

Solution: serve job with greatest Gittins(cost, state)

Ignoring arrivals in the γ-Gittins analysis?

batch problem:



Goal: minimize $\mathbf{E}[e^{\gamma T_\pi}] = \mathbf{E}[e^{-\gamma A} e^{\gamma D_\pi}]$

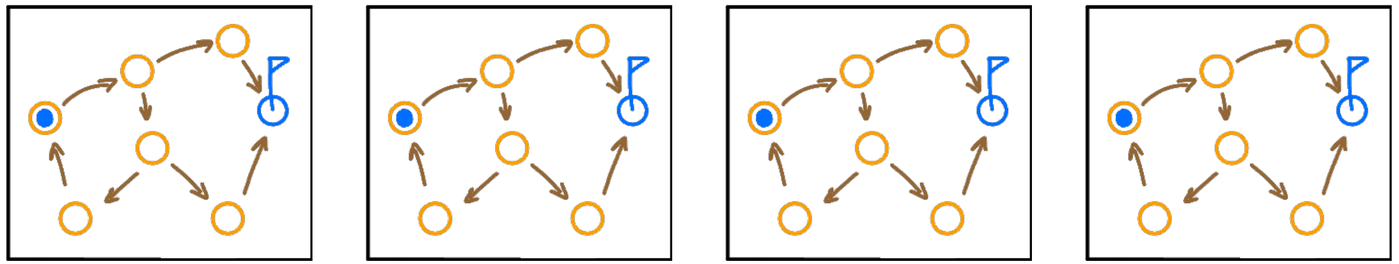
multi-armed bandit problem

Solution: serve job with greatest Gittins(cost, state)

Gittins(cost, state) = cost·Gittins(1, state)

Ignoring arrivals in the γ-Gittins analysis?

batch problem:



Goal: minimize $\mathbf{E}[e^{\gamma T_\pi}] = \mathbf{E}[e^{-\gamma A} e^{\gamma D_\pi}]$

multi-armed bandit problem

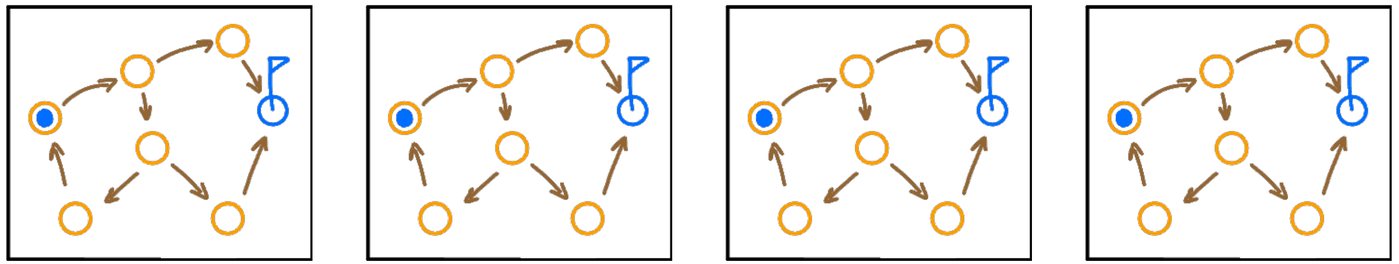
Solution: serve job with greatest Gittins(cost, state)

Gittins(cost, state) = cost·Gittins(1, state)

$$-\frac{1}{\gamma}\log(\text{Gittins}(e^{-\gamma A}, \text{state})) = A - \frac{1}{\gamma}\log(\text{Gittins}(1, \text{state}))$$

Ignoring arrivals in the γ-Gittins analysis?

batch problem:



Goal: minimize $\mathbf{E}[e^{\gamma T_\pi}] = \mathbf{E}[e^{-\gamma A} e^{\gamma D_\pi}]$

multi-armed bandit problem

Solution: serve job with greatest Gittins(cost, state)

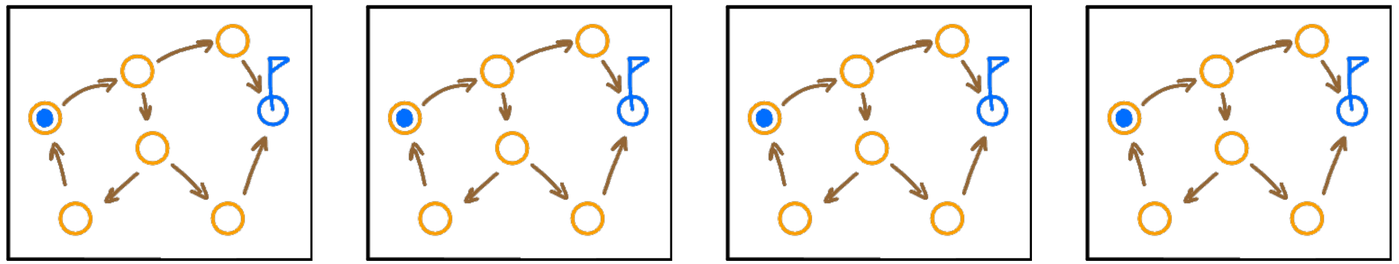
Gittins(cost, state) = cost·Gittins(1, state)

boost $= \frac{1}{\gamma}\log(\text{Gittins})$

$$-\frac{1}{\gamma}\log(\text{Gittins}(e^{-\gamma A}, \text{state})) = A - \frac{1}{\gamma}\log(\text{Gittins}(1, \text{state}))$$

Ignoring arrivals in the γ-Gittins analysis?

batch problem:



Goal: minimize $\mathbf{E}[e^{\gamma T_\pi}] = \mathbf{E}[e^{-\gamma A} e^{\gamma D_\pi}]$

multi-armed bandit problem

this is γ-Gittins!

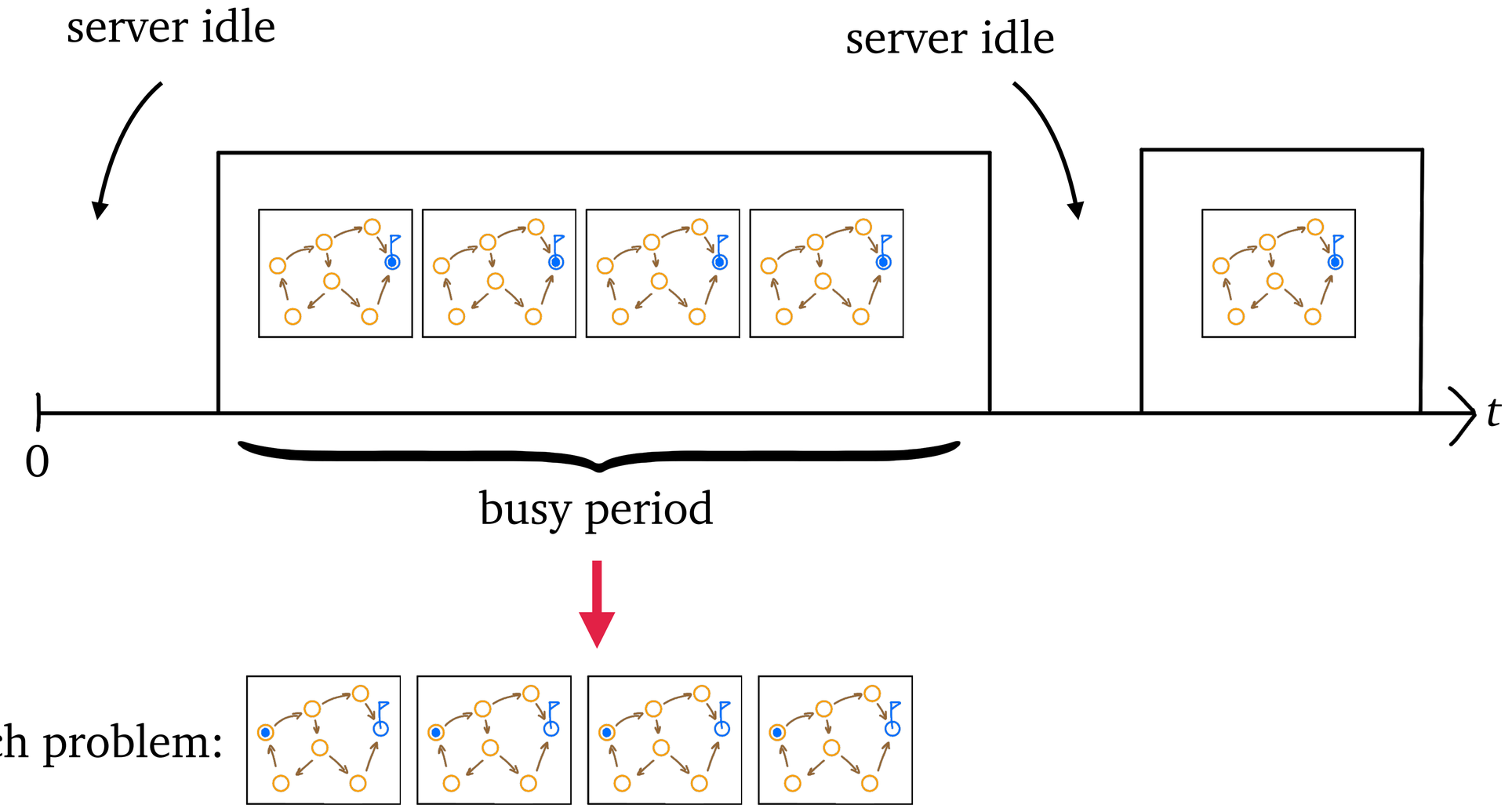
Solution: serve job with greatest Gittins(cost, state)

Gittins(cost, state) = cost·Gittins(1, state)

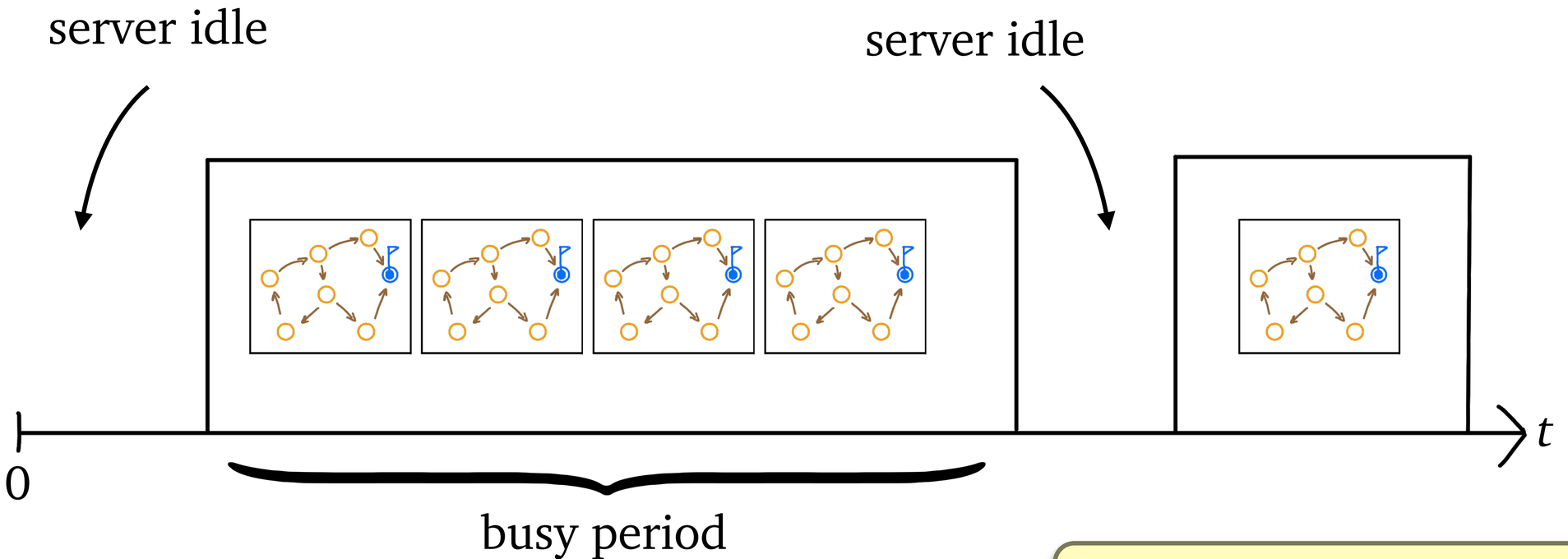
boost $= \frac{1}{\gamma}\log(\text{Gittins})$

$$-\frac{1}{\gamma}\log(\text{Gittins}(e^{-\gamma A}, \text{state})) = A - \frac{1}{\gamma}\log(\text{Gittins}(1, \text{state}))$$

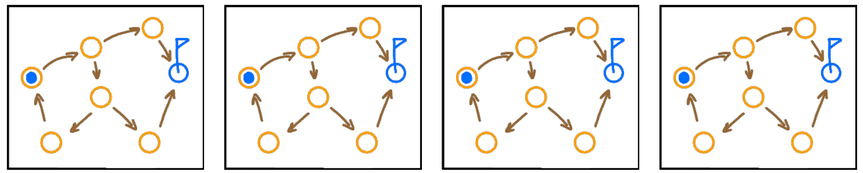
Ignoring arrivals in the γ-Gittins analysis?



Ignoring arrivals in the γ-Gittins analysis?

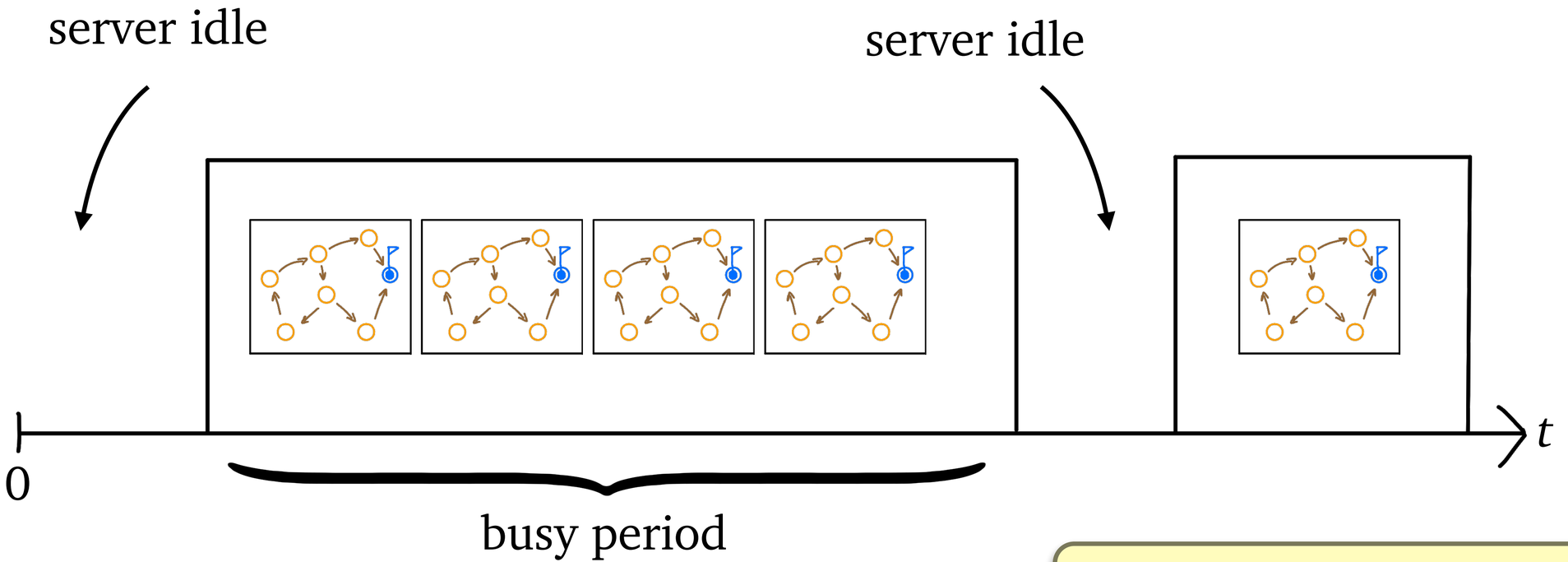


batch problem:

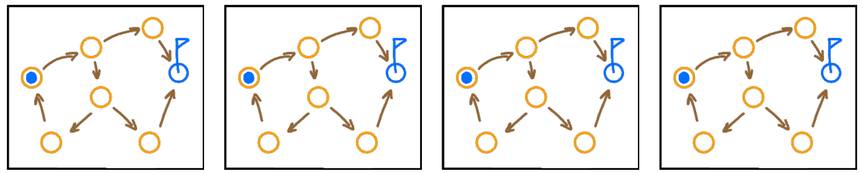


job sizes may be correlated when sampled as busy period

Ignoring arrivals in the γ-Gittins analysis?



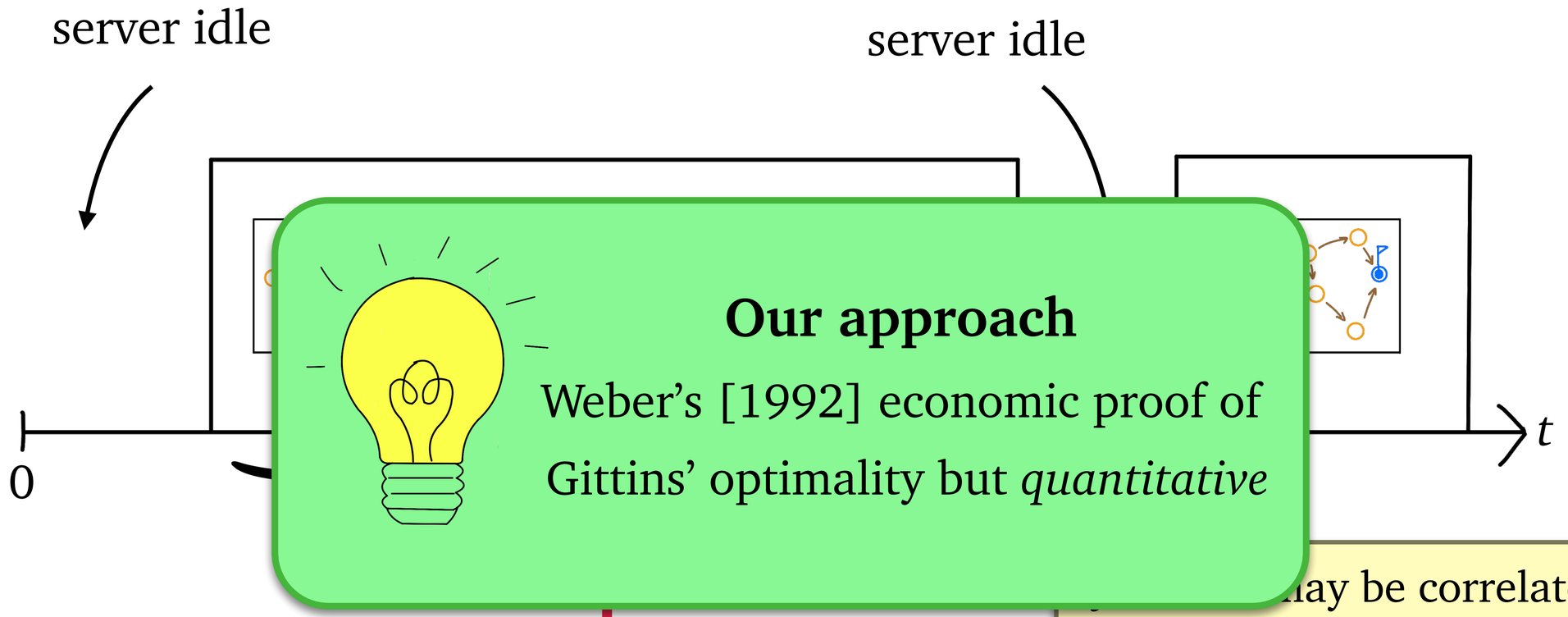
batch problem:



job sizes may be correlated when sampled as busy period

Gittins is no longer optimal…

Ignoring arrivals in the γ-Gittins analysis?



In summary

why is it surprising that Gittins is the right choice?

Theory: why was this hard to discover?

In summary

why is it surprising that Gittins is the right choice?

Theory: why was this hard to discover?

- initially looks like restless bandit: hard, needs different tools

In summary

why is it surprising that Gittins is the right choice?

Theory: why was this hard to discover?

- initially looks like restless bandit: hard, needs different tools
- MAB with arrivals: have time-inhomogeneous arrivals

In summary

why is it surprising that Gittins is the right choice?

Theory: why was this hard to discover?

- initially looks like restless bandit: hard, needs different tools
- MAB with arrivals: have time-inhomogeneous arrivals
- ignoring arrivals: busy period → batch problem leads to correlation

In summary

why is it surprising that Gittins is the right choice?

Theory: why was this hard to discover?

- initially looks like restless bandit: hard, needs different tools
- MAB with arrivals: have time-inhomogeneous arrivals
- ignoring arrivals: busy period → batch problem leads to correlation

quantitative economic argument!

In summary

γ-Gittins

$$\textbf{boost}(\text{state}) = \frac{1}{\gamma}\log(\text{Gittins index})$$

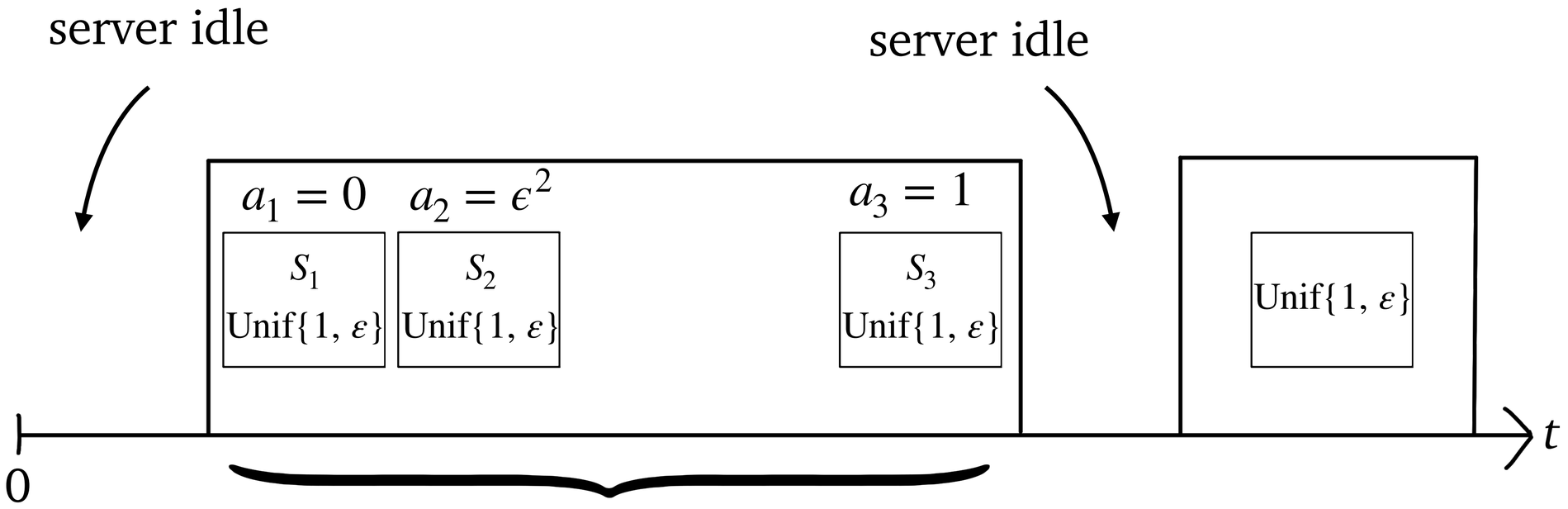
strongly optimal!

If you want to minimize the tail of response time:

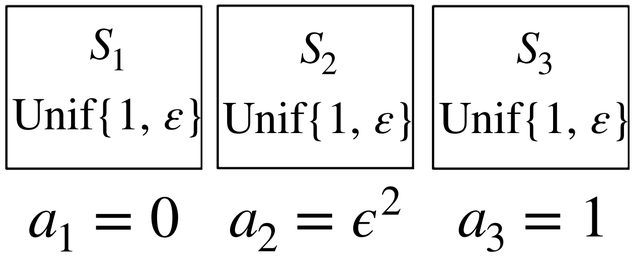
1. priority should depends on both arrival time and job state
2. priority $\approx$ big initial boost, then decrease based on state
3. priority can also increase based on state, but times where it decreases are more important
4. can use any information model,
   but more information $\Longrightarrow$ better performance

Please reach out!
ah843@cornell.edu

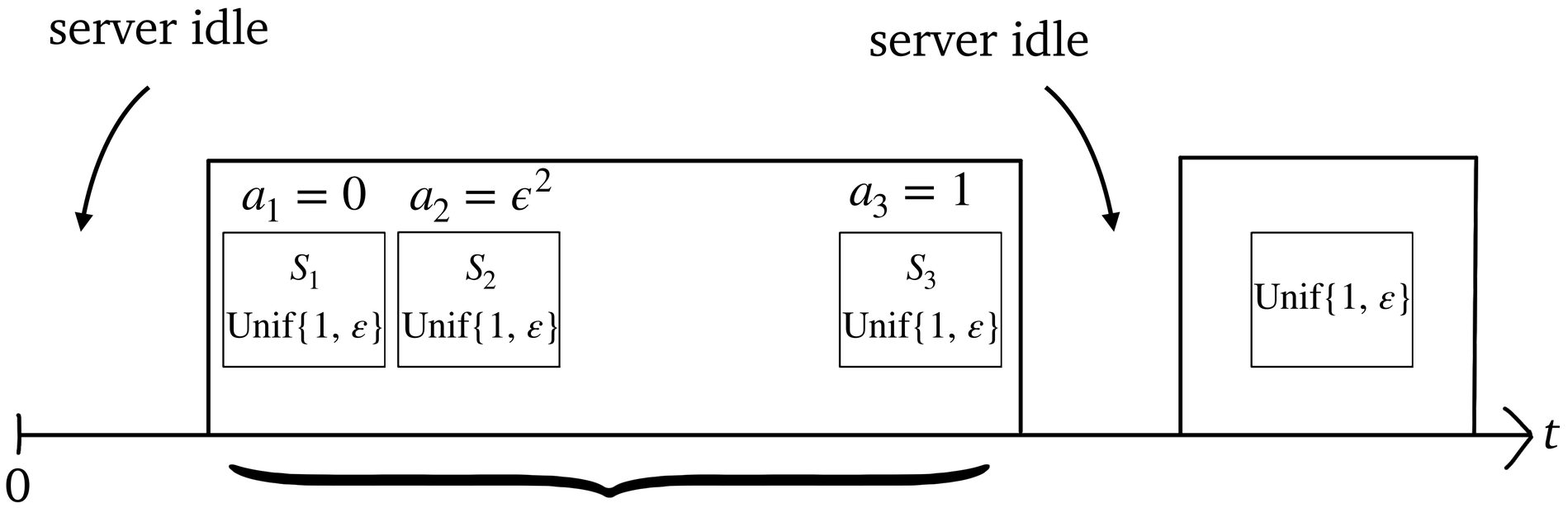
bonus: why might there be job size correlations?



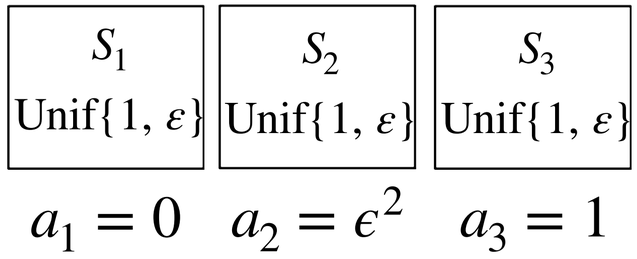
batch problem:



bonus: why might there be job size correlations?



batch problem:



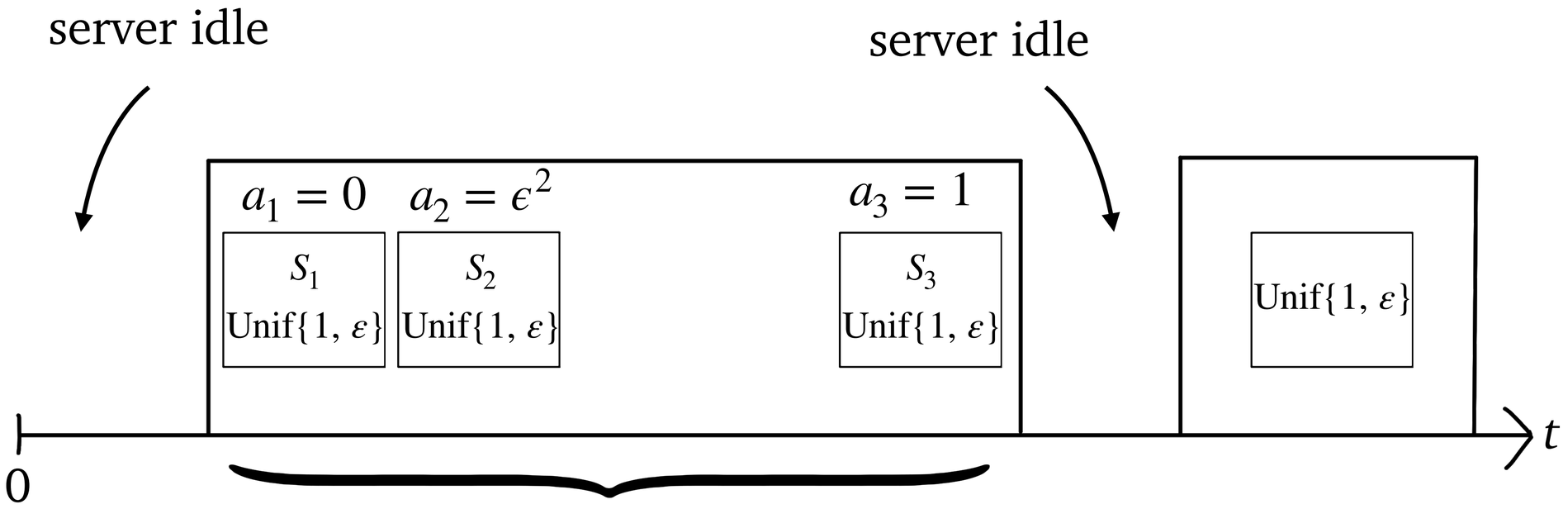
Example

Let $\varepsilon \ll 1$ and $S = \text{Unif}\{1, \varepsilon\}$

3 jobs: $a_1 = 0$, $a_2 = \varepsilon^2$, $a_3 = 1$

If $S_1 = \varepsilon$ then $S_2 = 1$

bonus: why might there be job size correlations?



batch problem:

S_1 Unif$\{1, \varepsilon\}$	S_2 Unif$\{1, \varepsilon\}$	S_3 Unif$\{1, \varepsilon\}$
$a_1 = 0$	$a_2 = \epsilon^2$	$a_3 = 1$

Example

Let $\varepsilon \ll 1$ and $S = \text{Unif}\{1, \varepsilon\}$

3 jobs: $a_1 = 0$, $a_2 = \varepsilon^2$, $a_3 = 1$

If $S_1 = \varepsilon$ then $S_2 = 1$