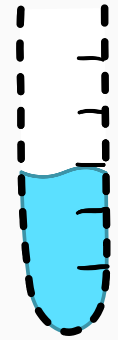
What TCS Can Do for Queueing

and

What Queueing Can Do for TCS

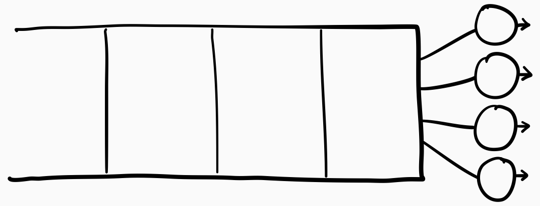
in Scheduling Theory

Ziv Scully
Cornell University



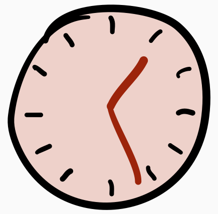
Part I

Handling job size uncertainty



Part II

Analyzing multiserver scheduling

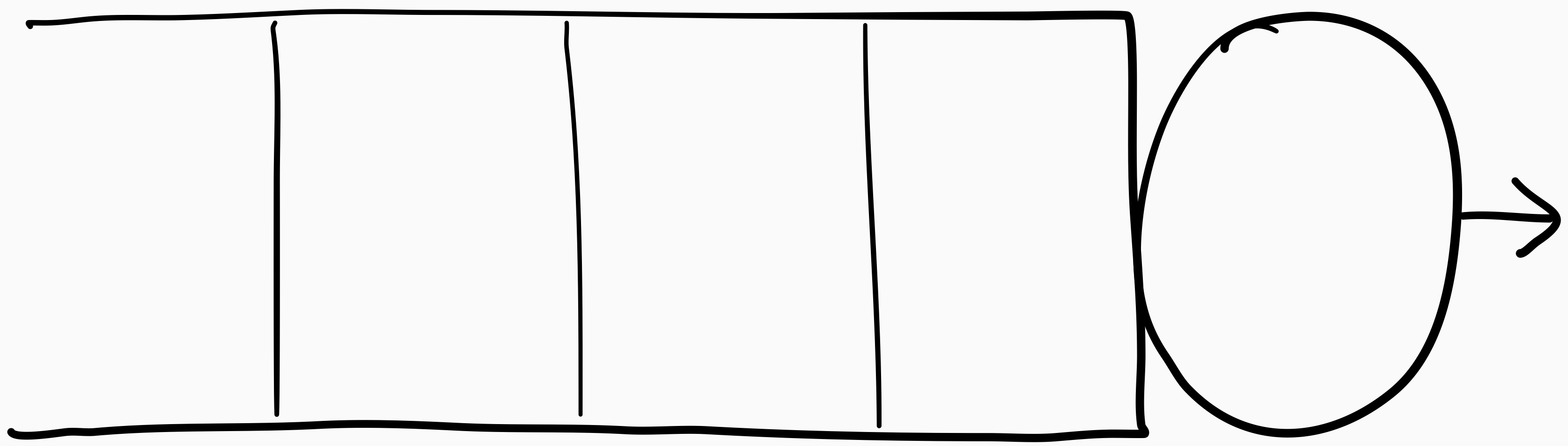


Part III

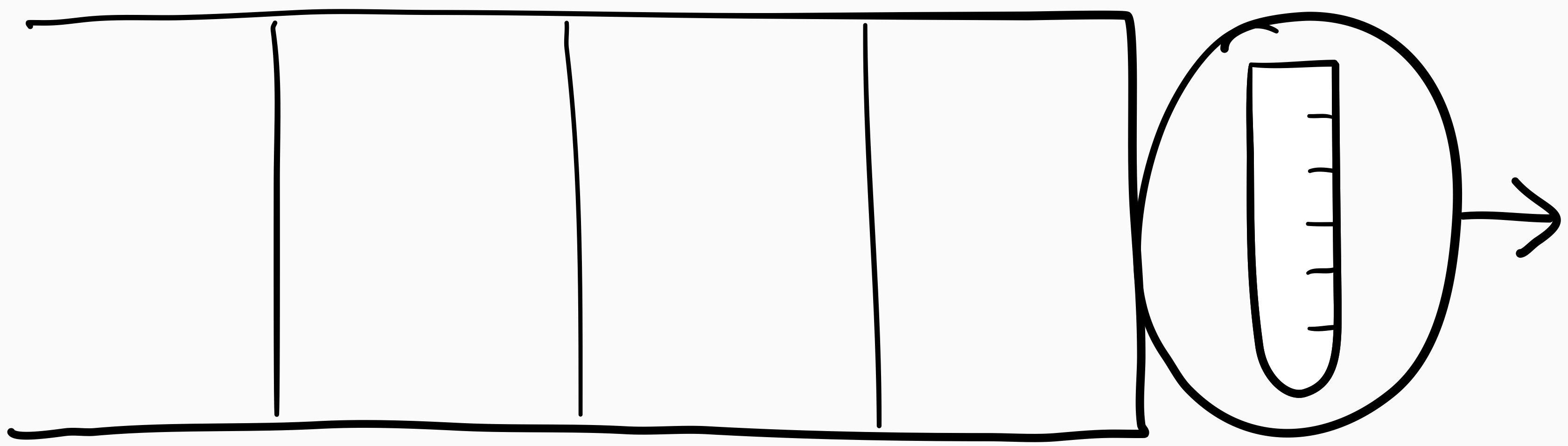
Optimizing tail metrics

How should we schedule jobs to minimize delay?

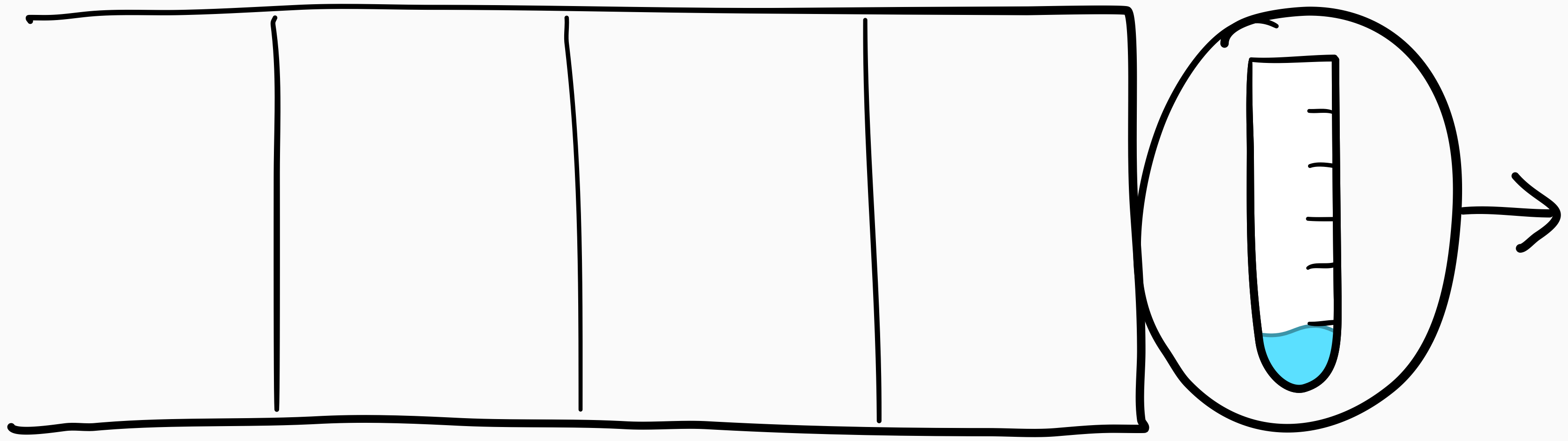
How should we schedule jobs to minimize delay?



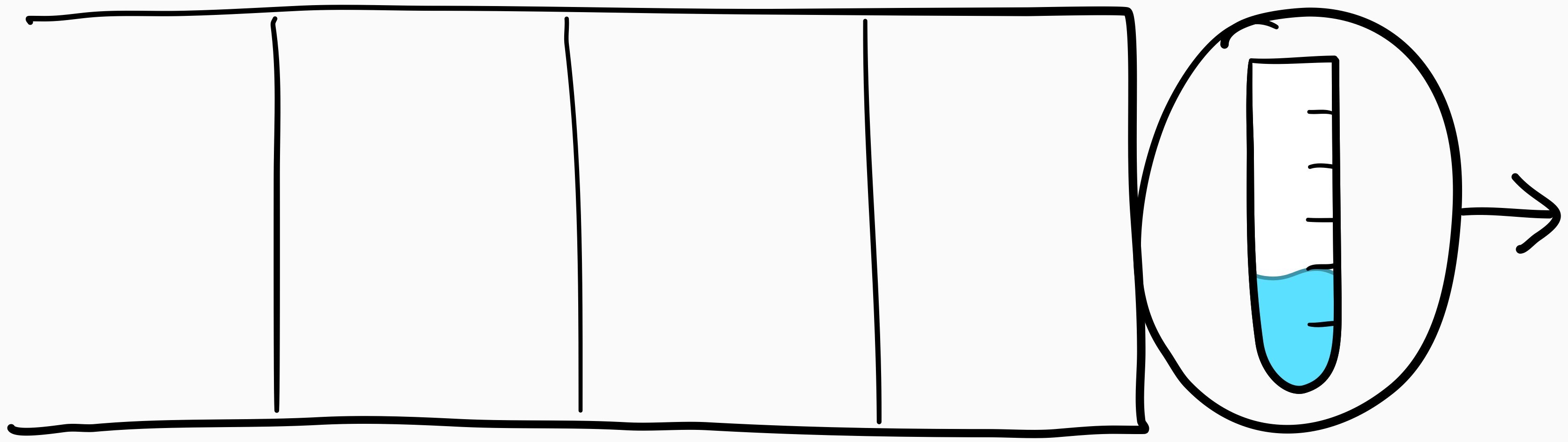
How should we schedule jobs to minimize delay?



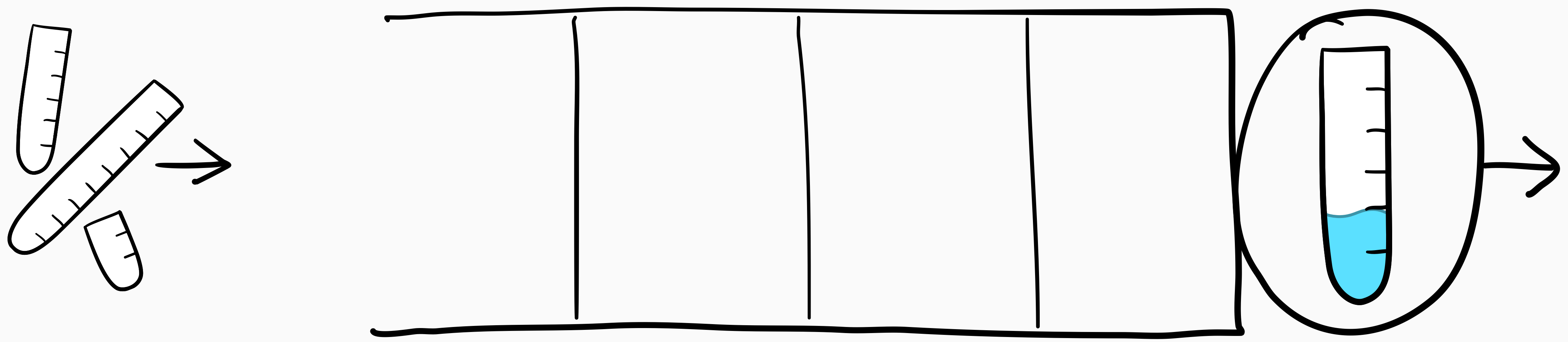
How should we schedule jobs to minimize delay?



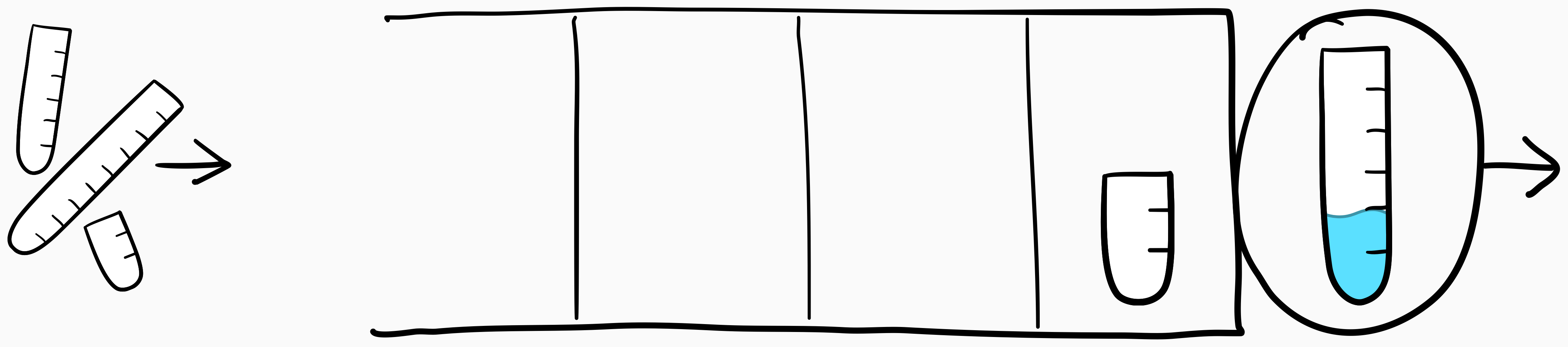
How should we schedule jobs to minimize delay?



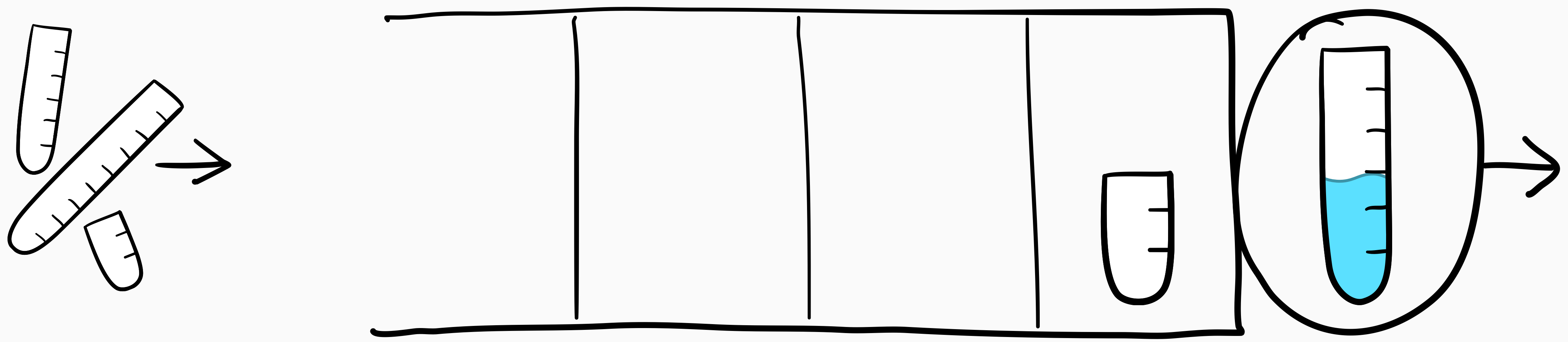
How should we schedule jobs to minimize delay?



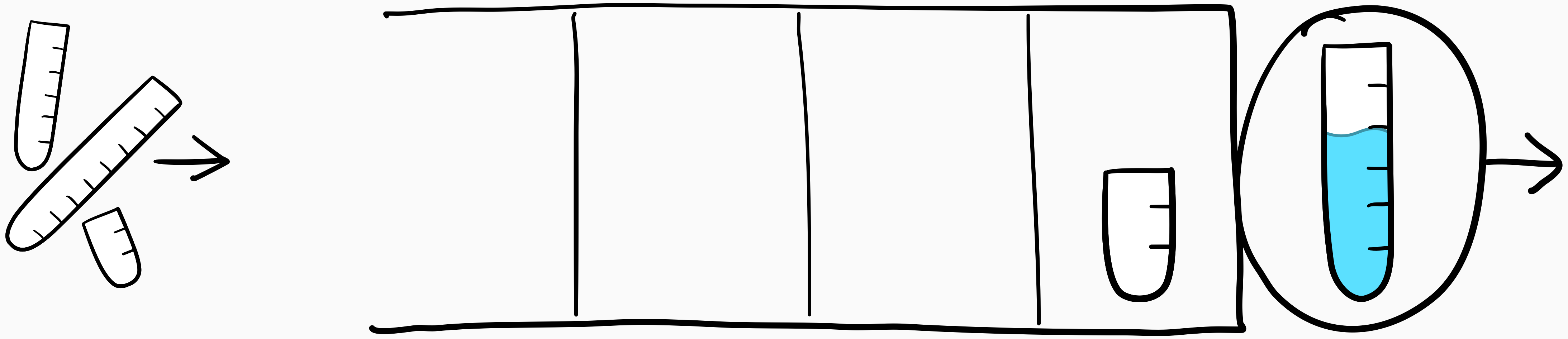
How should we schedule jobs to minimize delay?



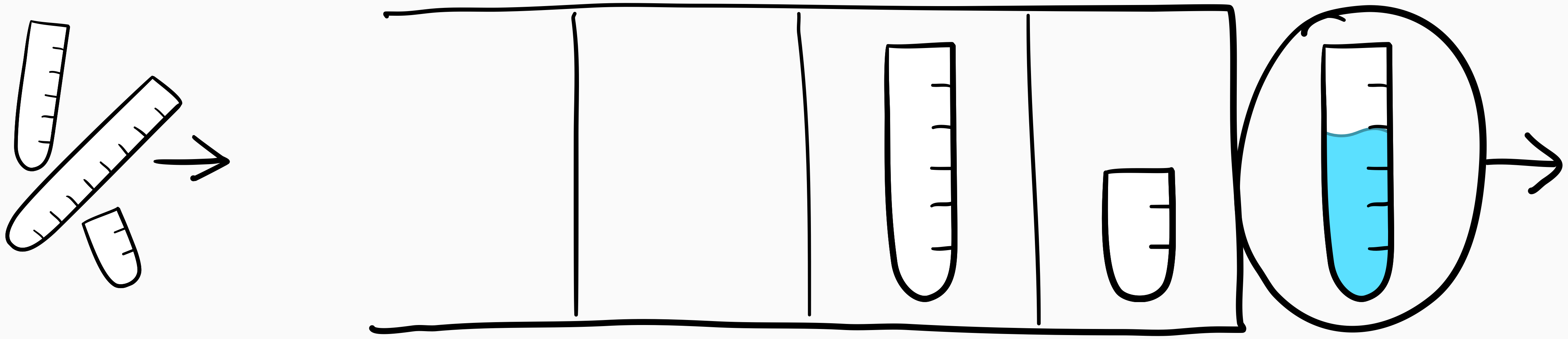
How should we schedule jobs to minimize delay?



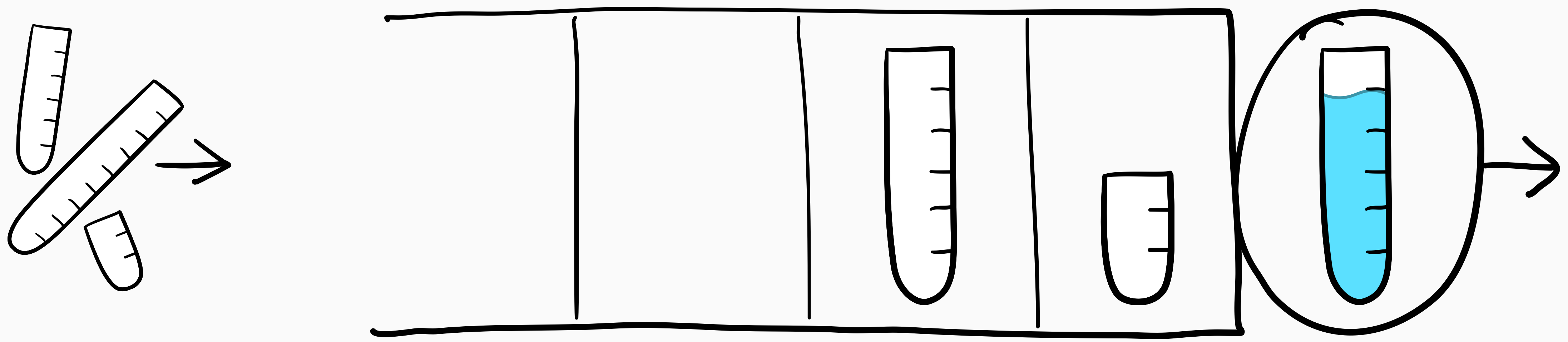
How should we schedule jobs to minimize delay?



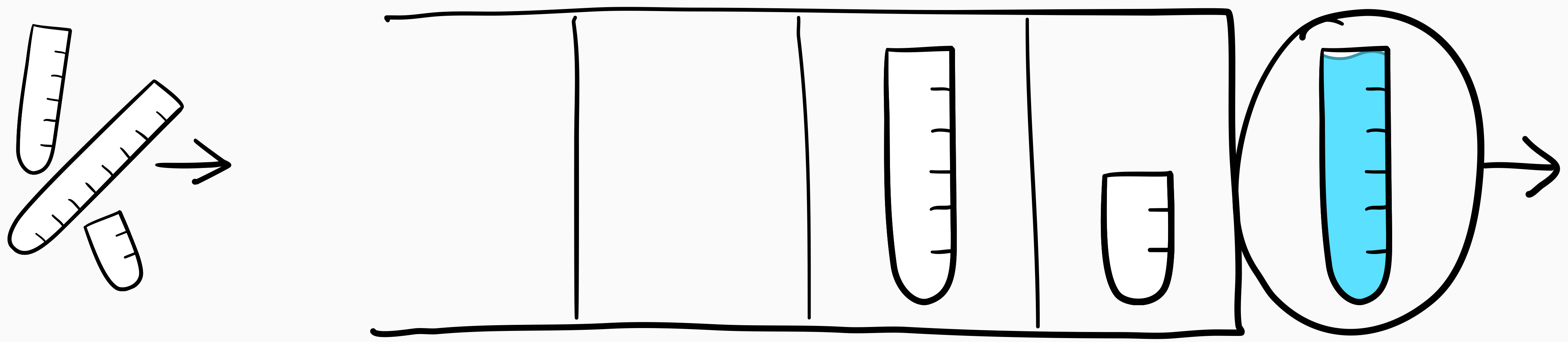
How should we schedule jobs to minimize delay?



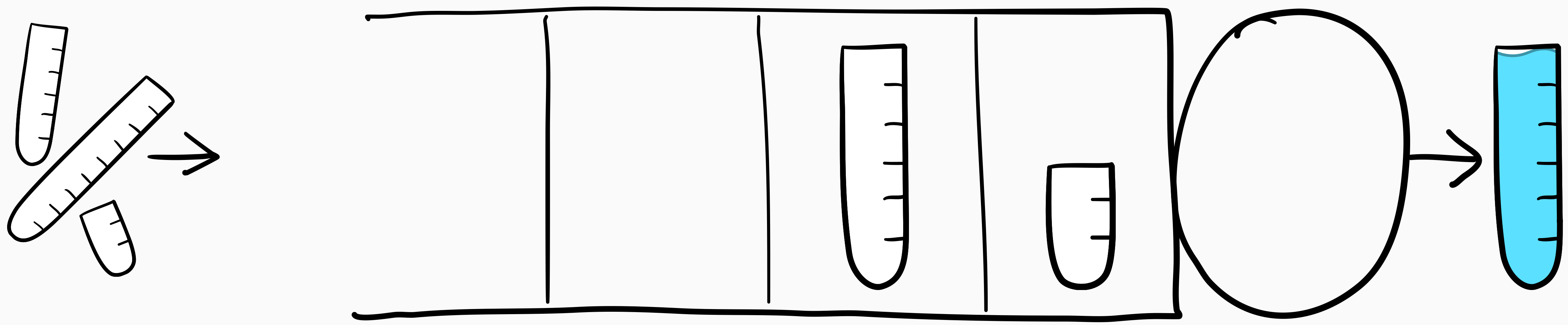
How should we schedule jobs to minimize delay?



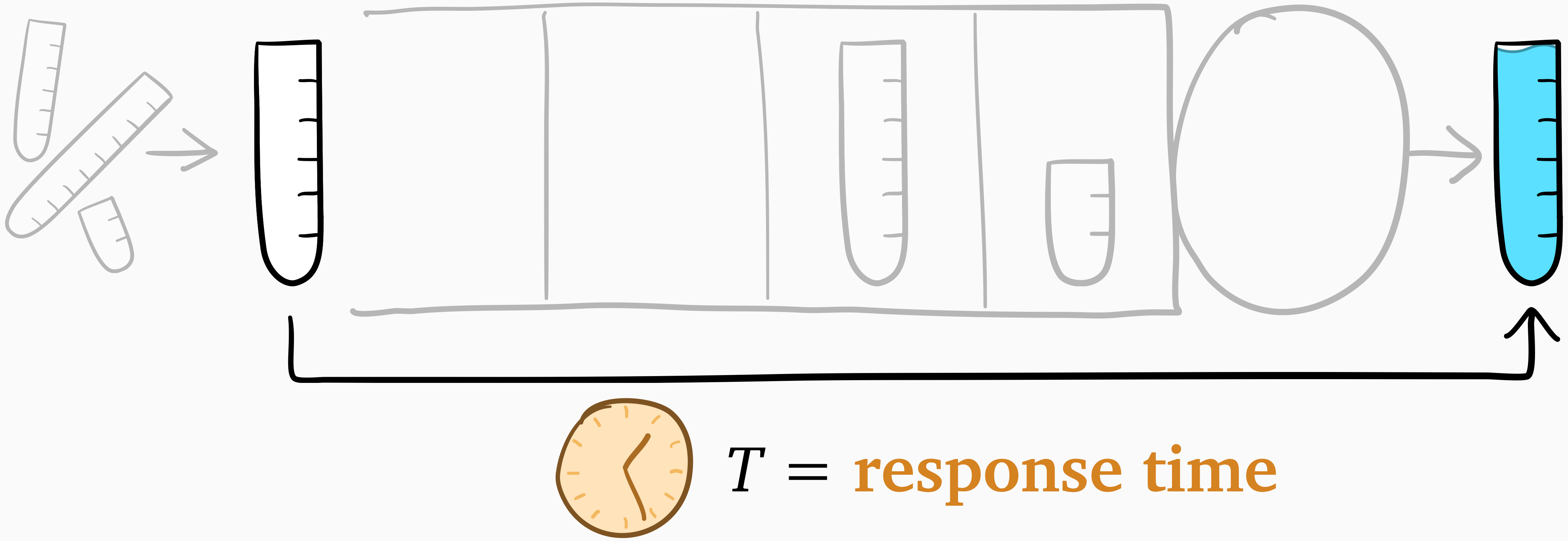
How should we schedule jobs to minimize delay?



How should we schedule jobs to minimize delay?

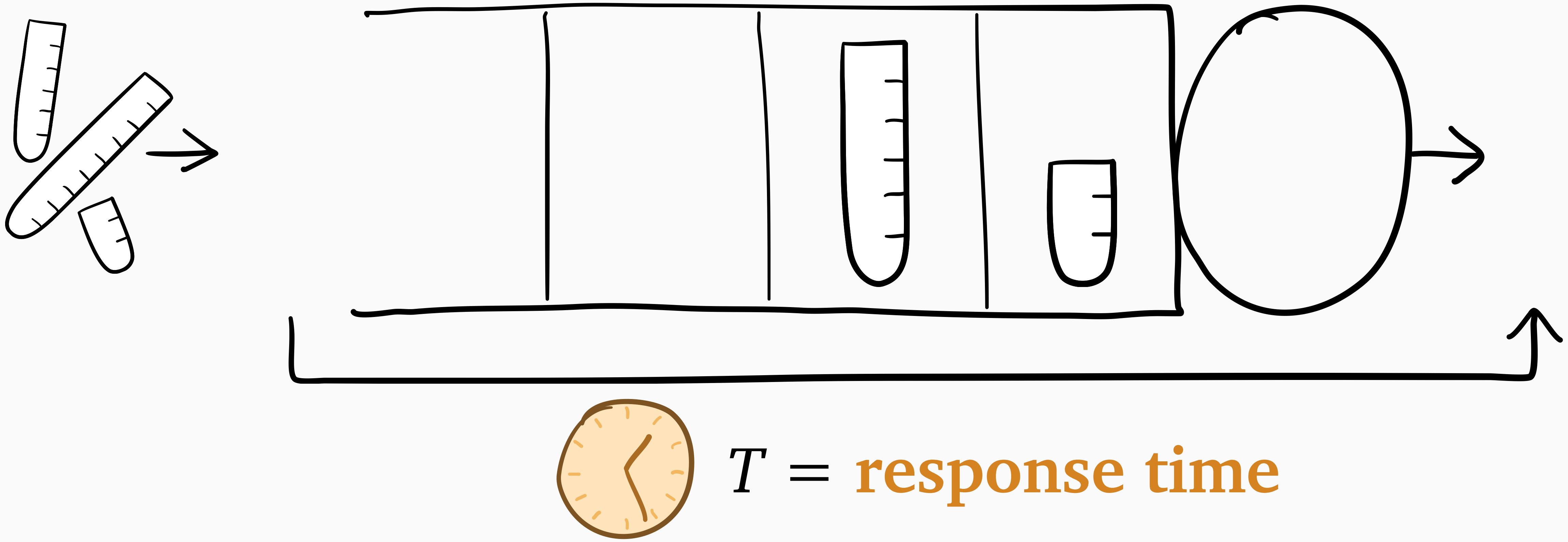


How should we schedule jobs to minimize delay?



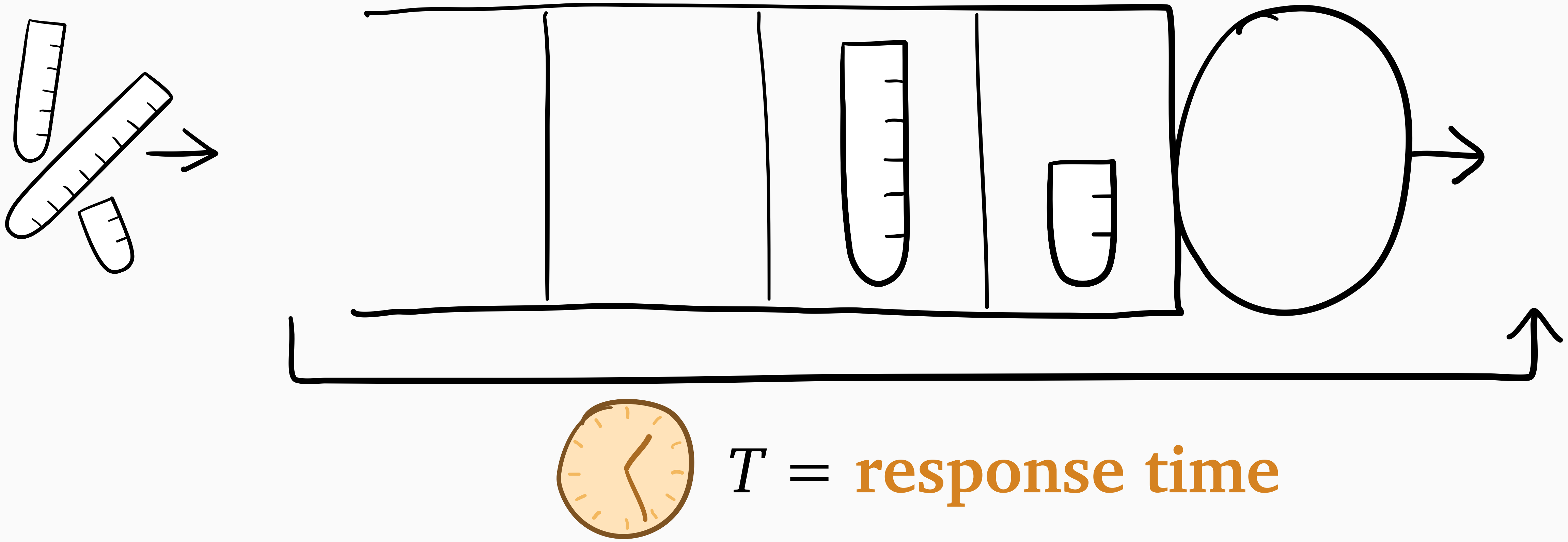
T = **response time**

How should we schedule jobs to minimize delay?



T = **response time**

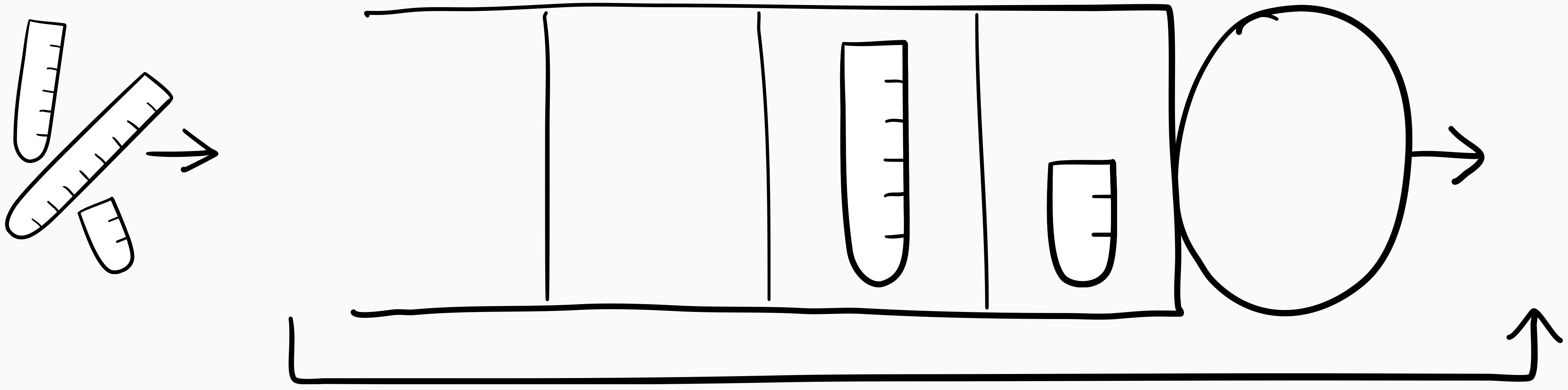
How should we schedule jobs to minimize delay?

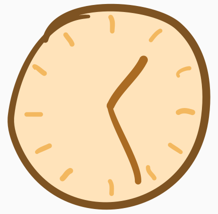


T = **response time**

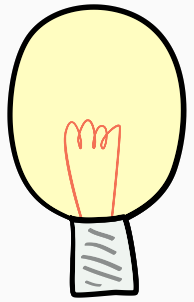
Minimize $\mathbf{E}[T]$?

How should we schedule jobs to minimize delay?

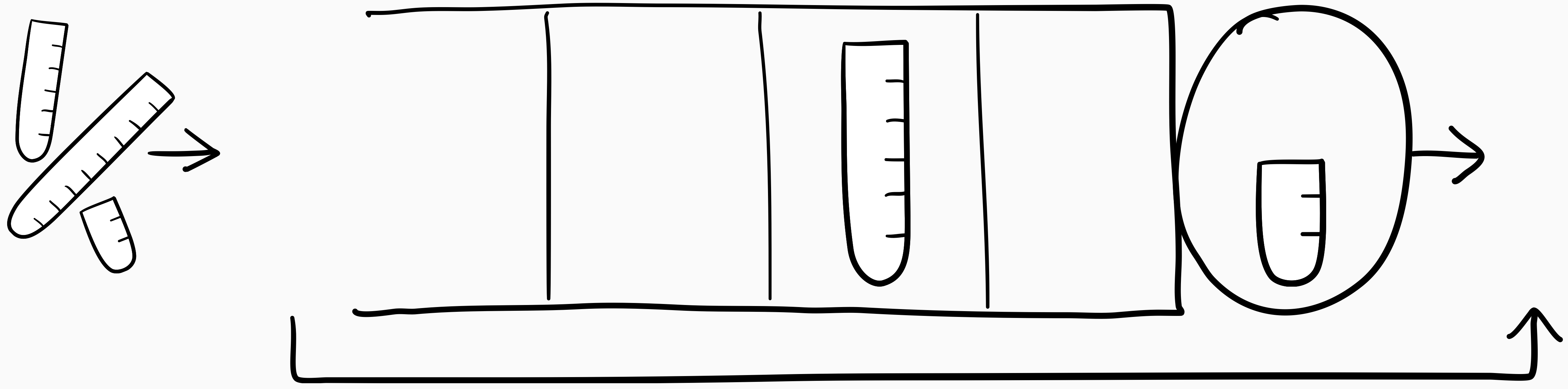


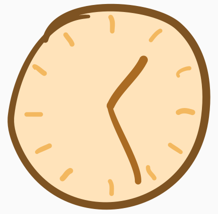
 T = **response time**

 Minimize $\mathbf{E}[T]$?

 Serve short jobs before long jobs

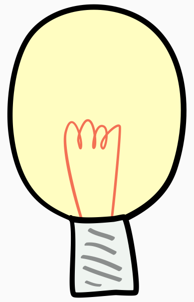
How should we schedule jobs to minimize delay?





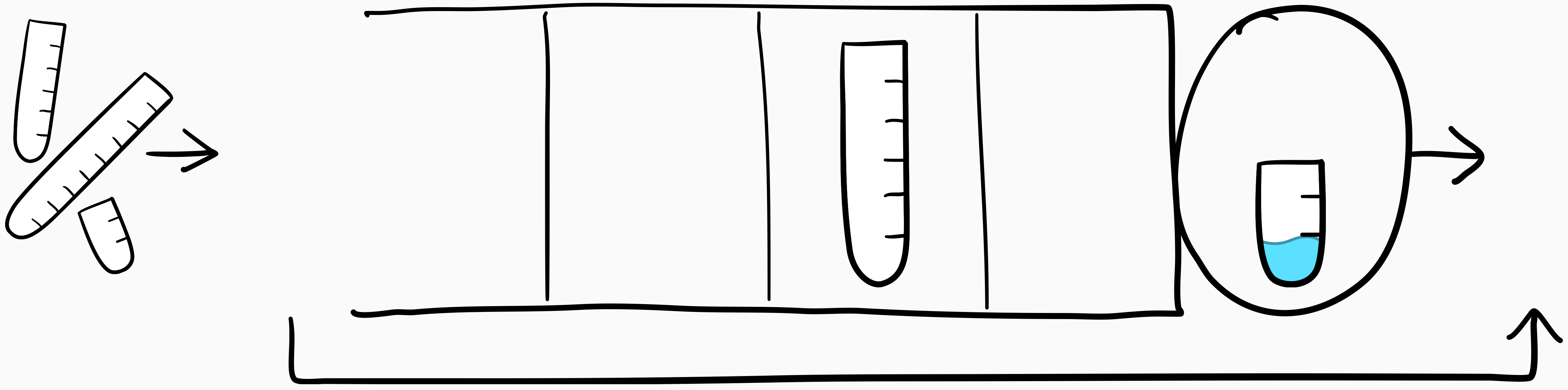
T = **response time**

Minimize $\mathbf{E}[T]$?



Serve short jobs
before long jobs

How should we schedule jobs to minimize delay?

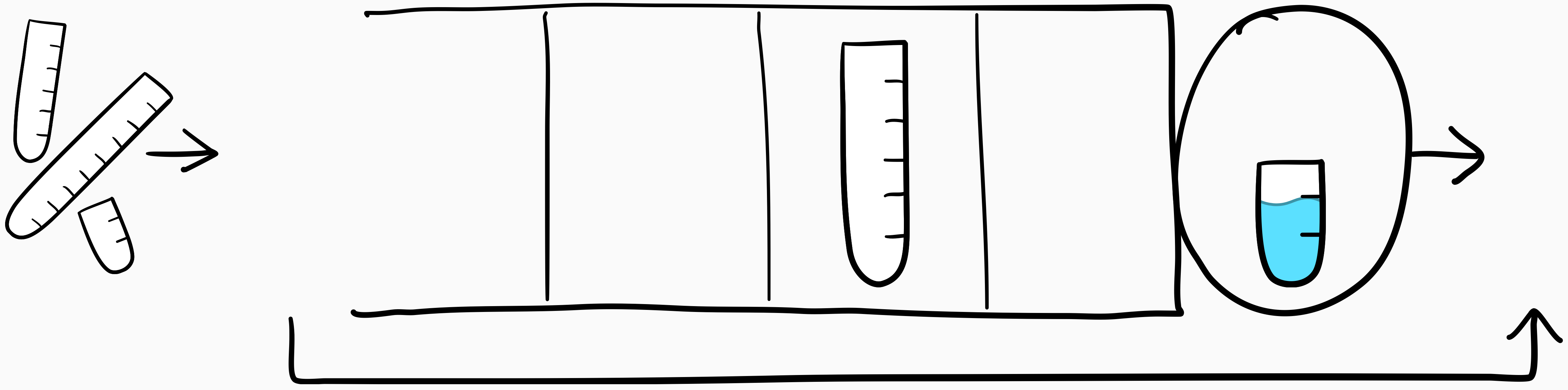


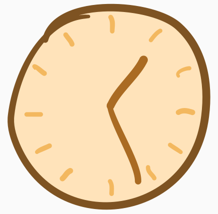
T = **response time**

Minimize $\mathbf{E}[T]$?

Serve short jobs before long jobs

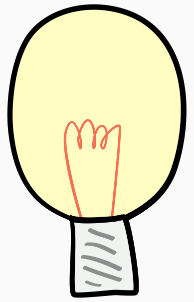
How should we schedule jobs to minimize delay?





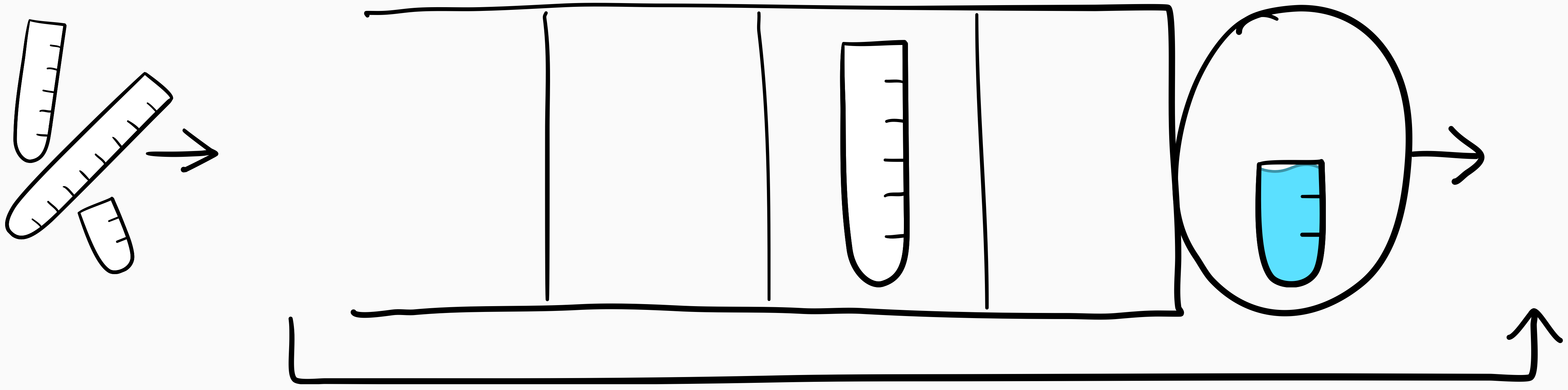
T = **response time**

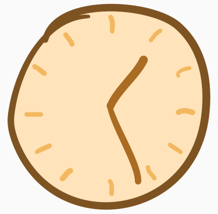
Minimize $\mathbf{E}[T]$?



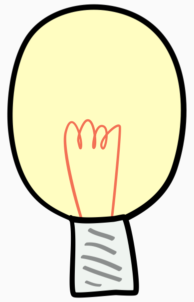
Serve short jobs
before long jobs

How should we schedule jobs to minimize delay?

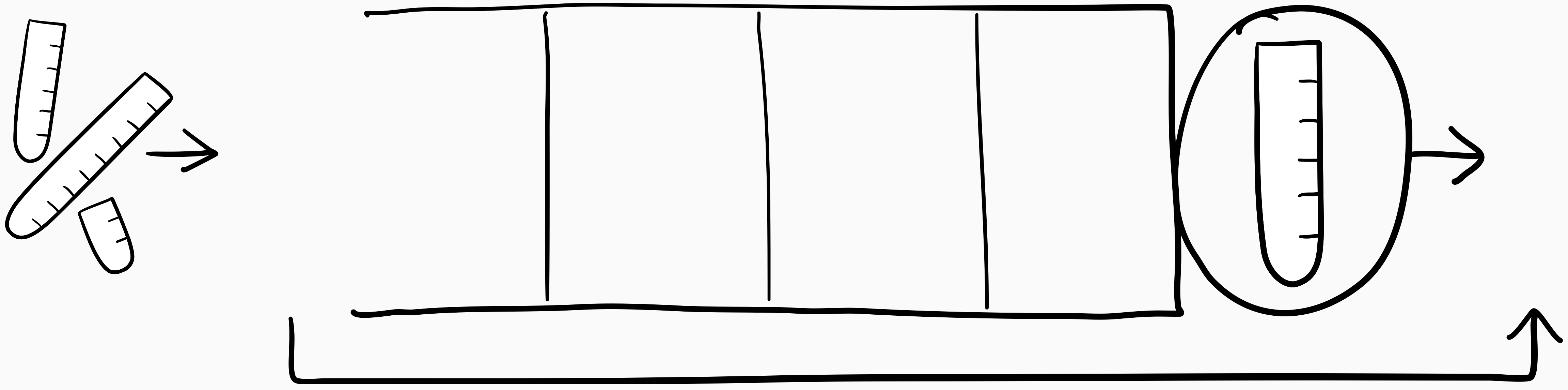


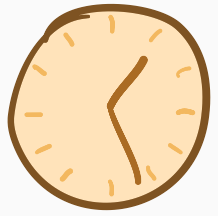
 T = **response time**

 Minimize $\mathbf{E}[T]$?

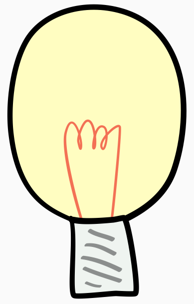
 Serve short jobs before long jobs

How should we schedule jobs to minimize delay?

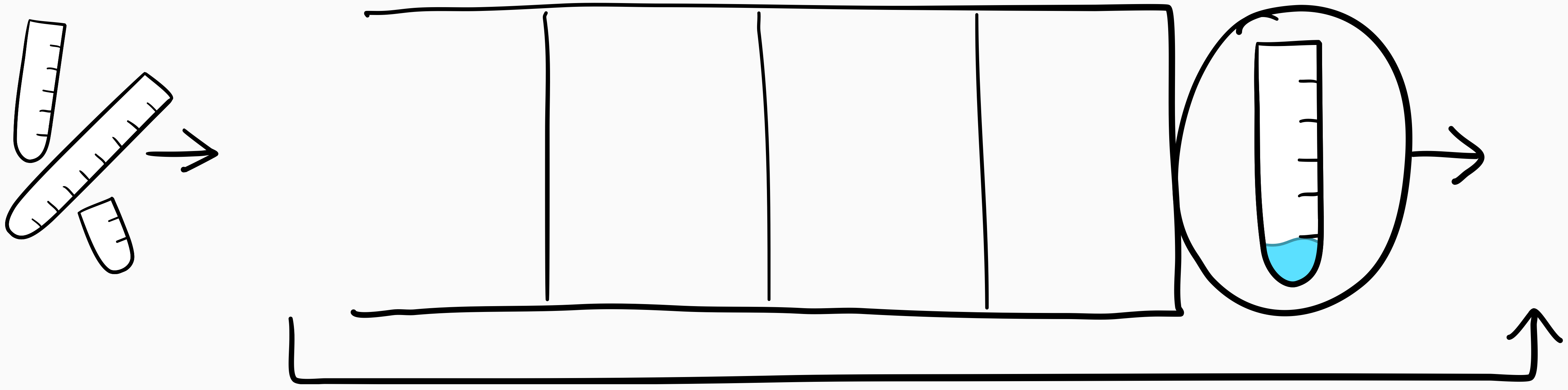


 T = **response time**

 Minimize $\mathbf{E}[T]$?

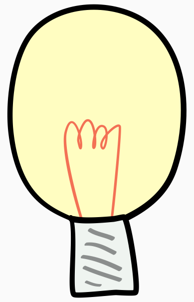
 Serve short jobs
before long jobs

How should we schedule jobs to minimize delay?

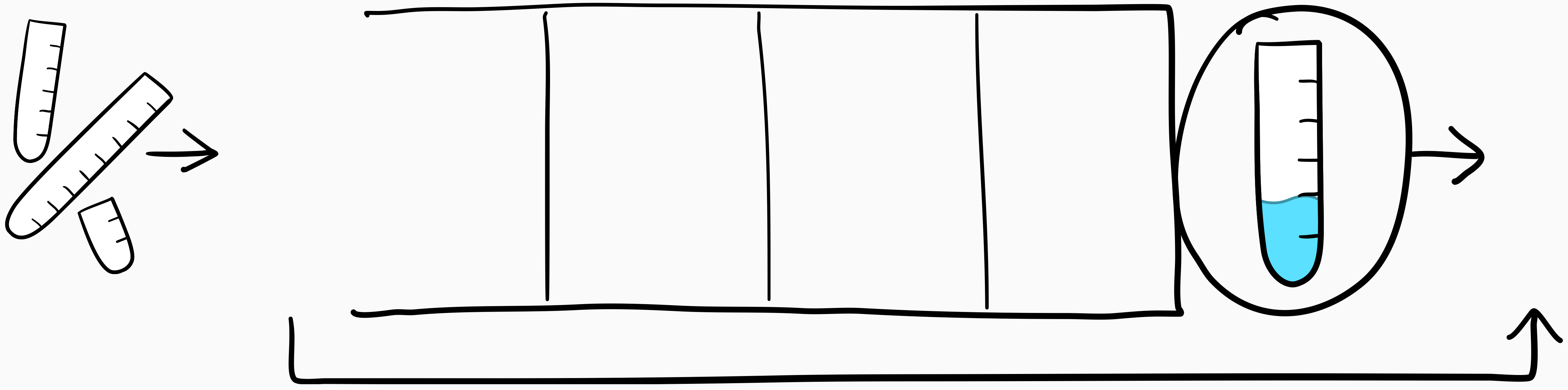


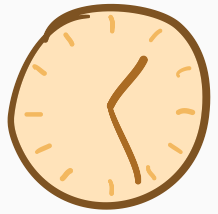
 T = **response time**

 Minimize $\mathbf{E}[T]$?

 Serve short jobs before long jobs

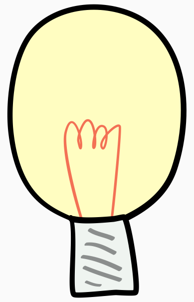
How should we schedule jobs to minimize delay?





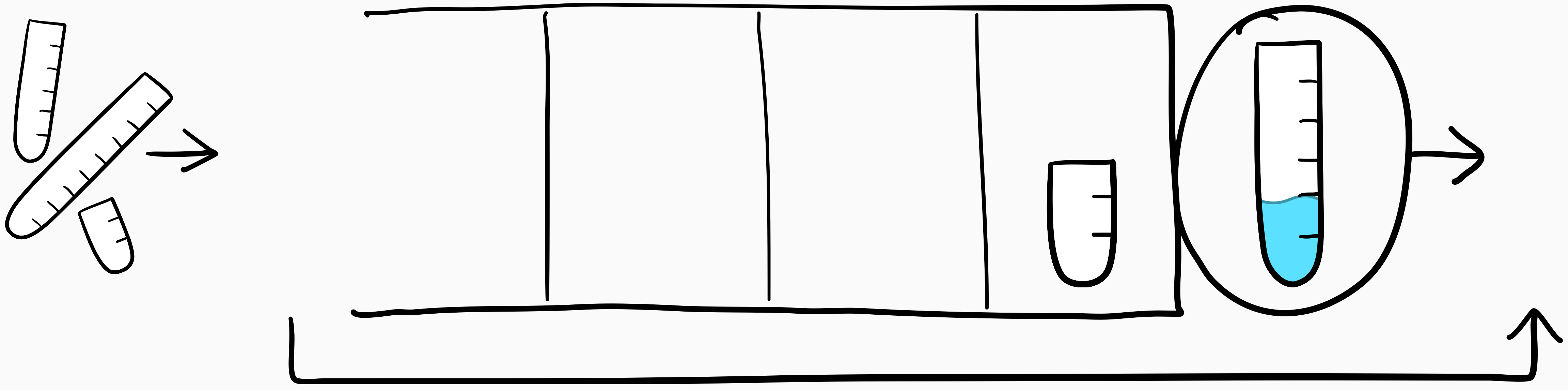
T = **response time**

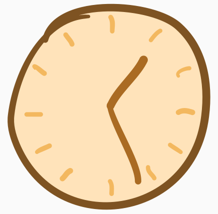
Minimize $\mathbf{E}[T]$?



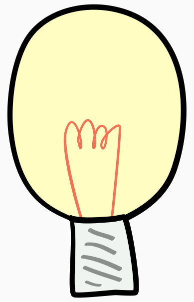
Serve short jobs
before long jobs

How should we schedule jobs to minimize delay?

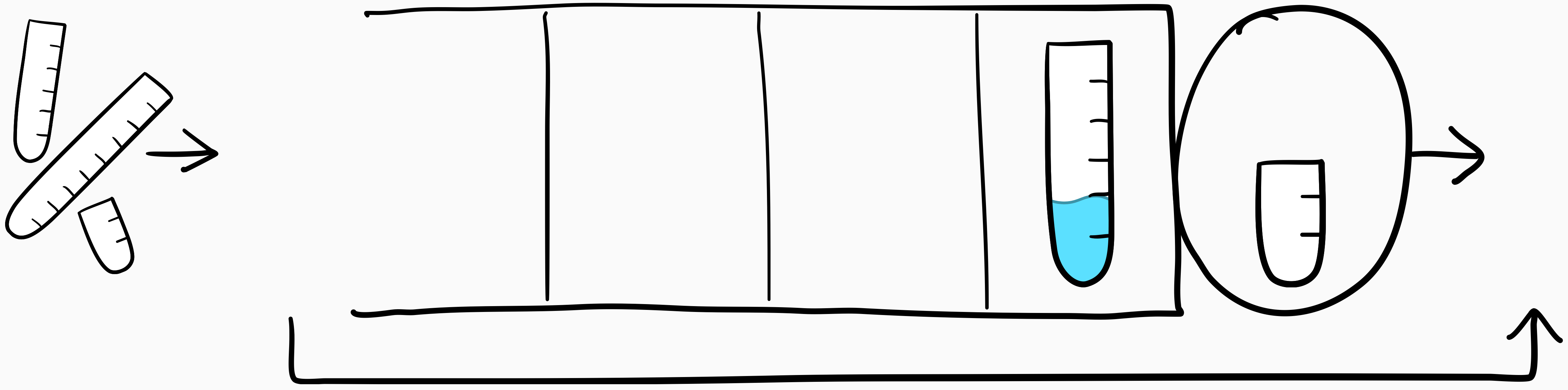


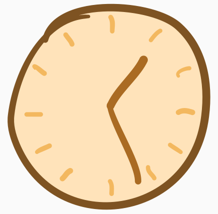
 T = **response time**

 Minimize $\mathbf{E}[T]$?

 Serve short jobs
before long jobs

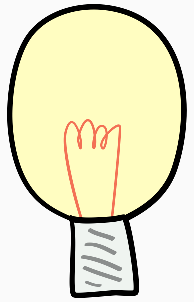
How should we schedule jobs to minimize delay?





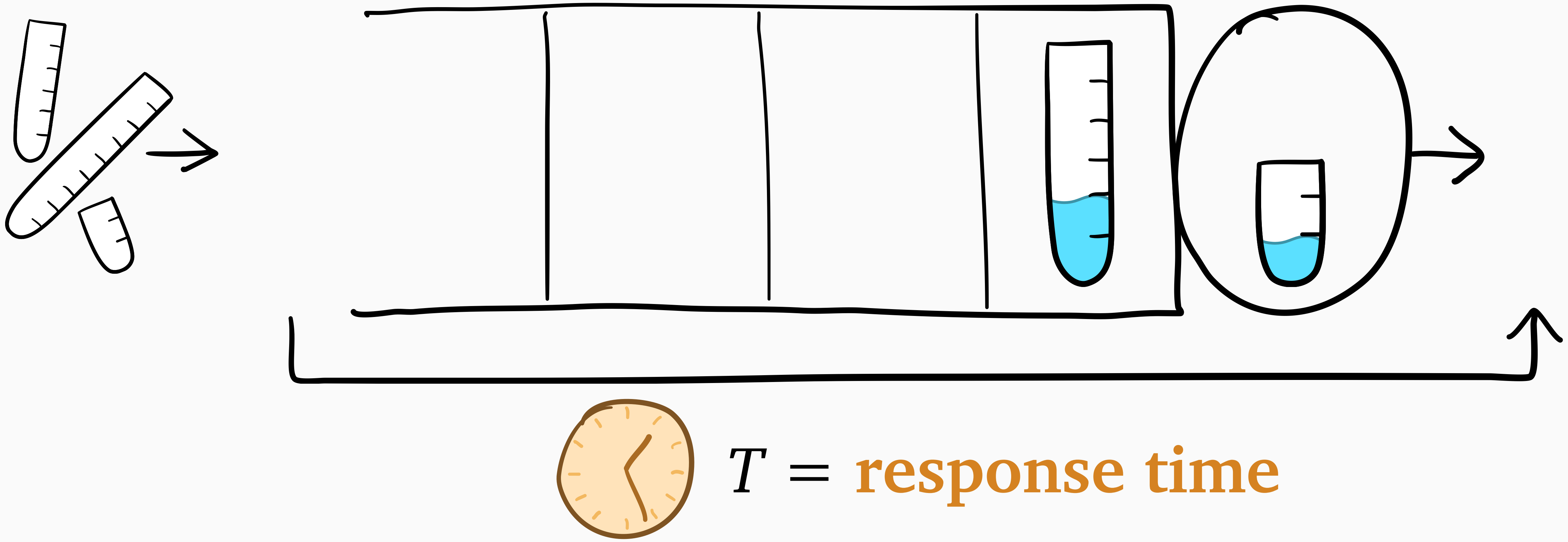
T = **response time**

Minimize $\mathbf{E}[T]$?



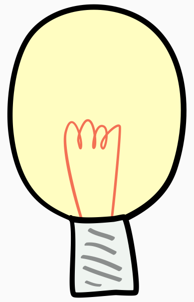
Serve short jobs
before long jobs

How should we schedule jobs to minimize delay?



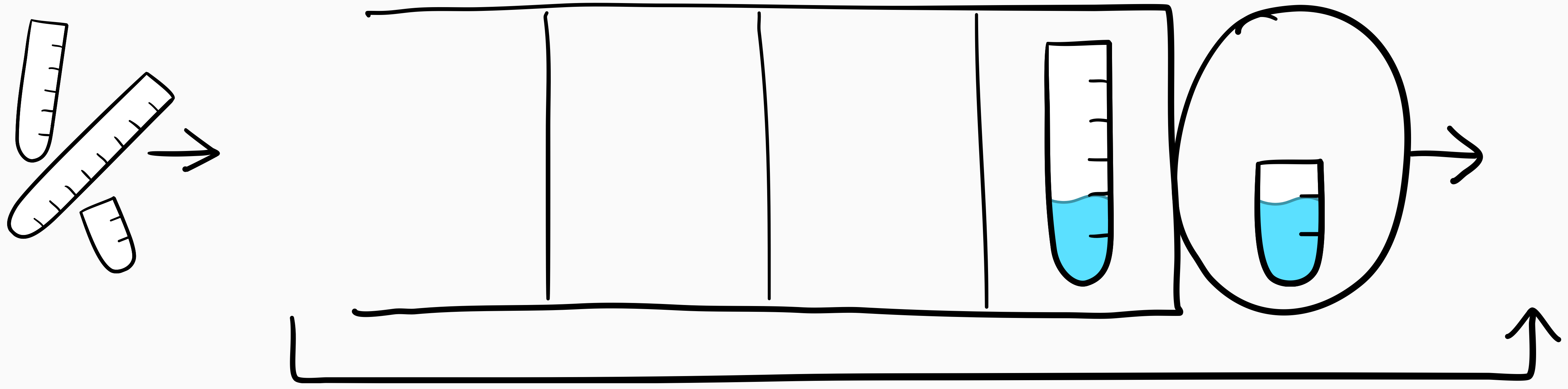
T = **response time**

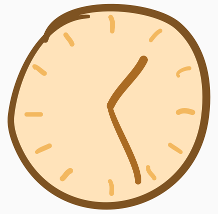
Minimize $\mathbf{E}[T]$?



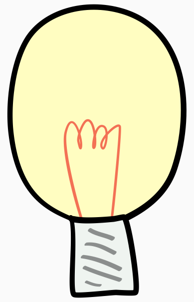
Serve short jobs
before long jobs

How should we schedule jobs to minimize delay?

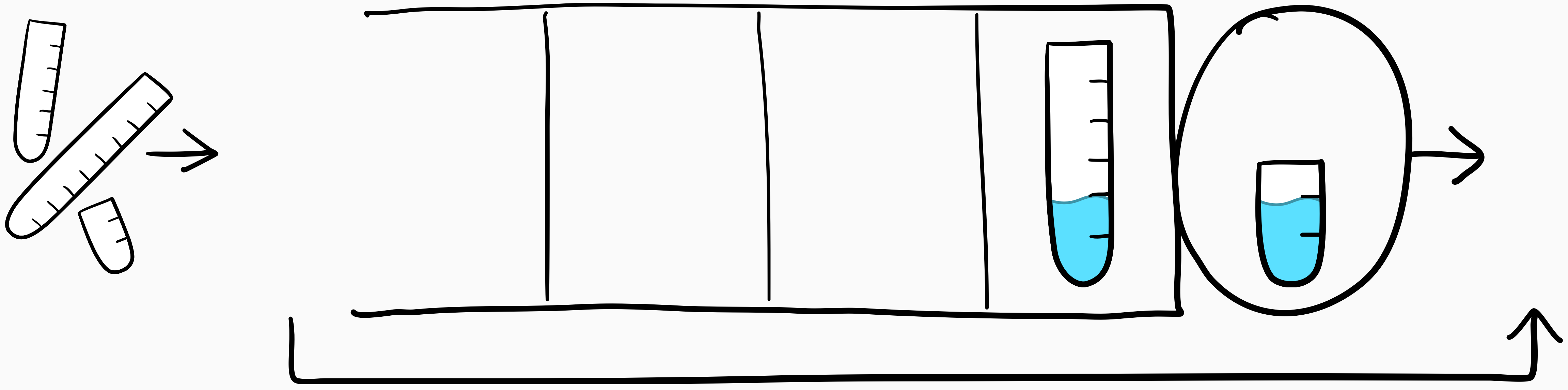


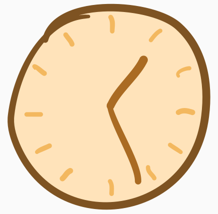
 T = **response time**

 Minimize $\mathbf{E}[T]$?

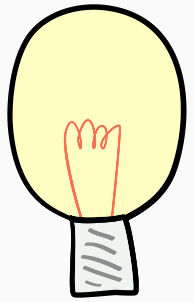
 Serve short jobs
before long jobs

How should we schedule jobs to minimize delay?




T = **response time**

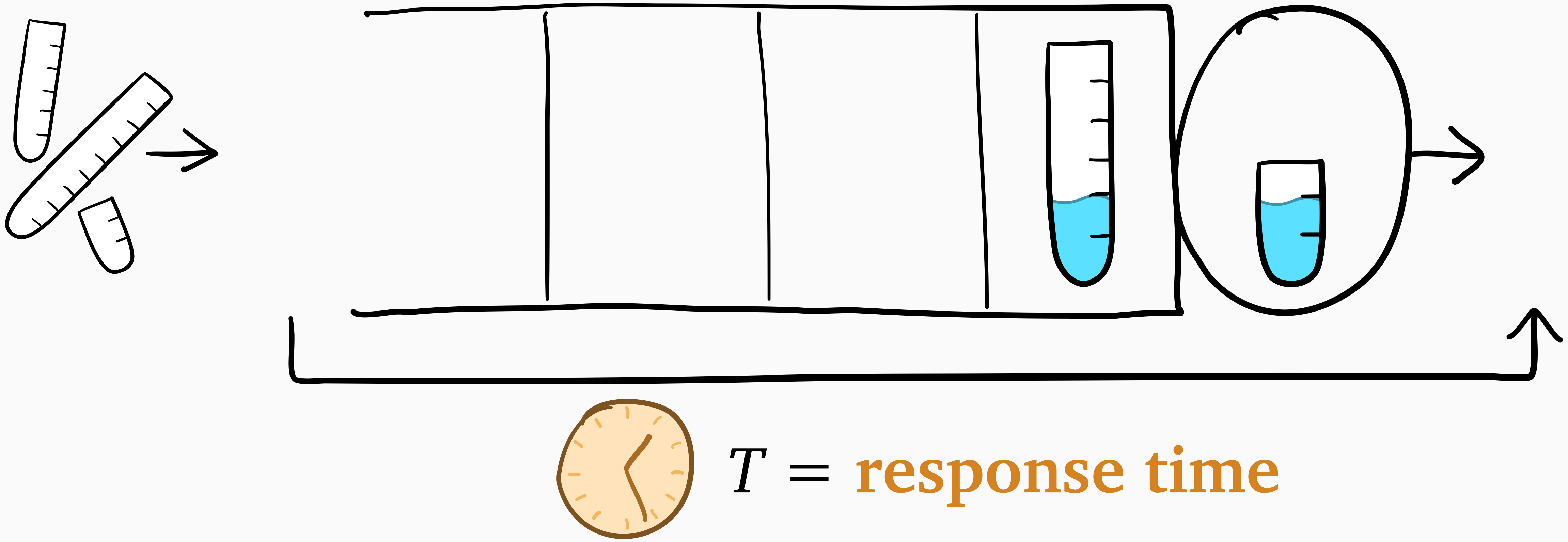

Minimize $\mathbf{E}[T]$?


Serve short jobs
before long jobs

shortest remaining
processing time

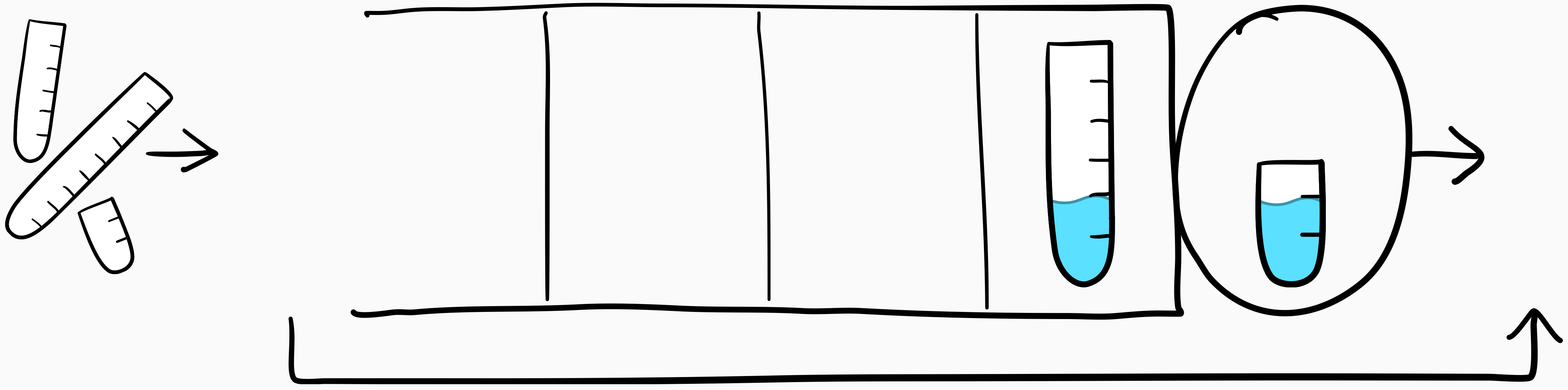

SRPT: minimizes $\mathbf{E}[T]$

TCS vs. Queueing



T = **response time**

TCS vs. Queueing

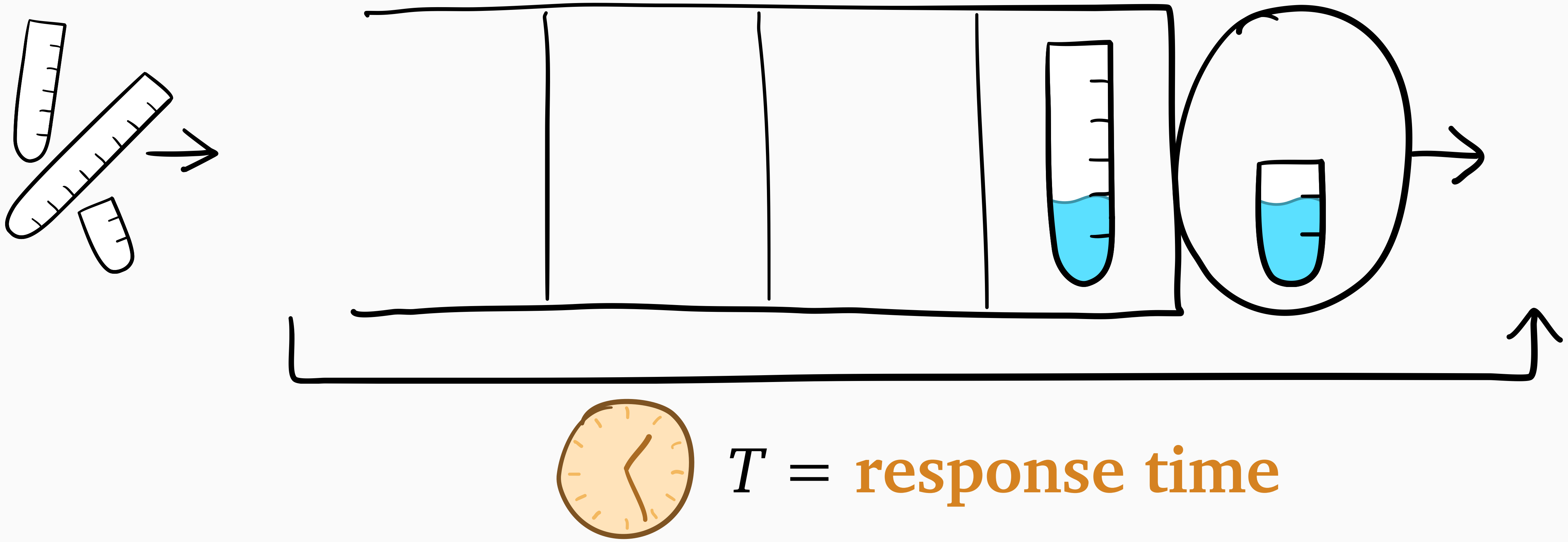


T = **response time**

TCS

n arbitrary arrivals
T is tuple of n times

TCS vs. Queueing



T = response time

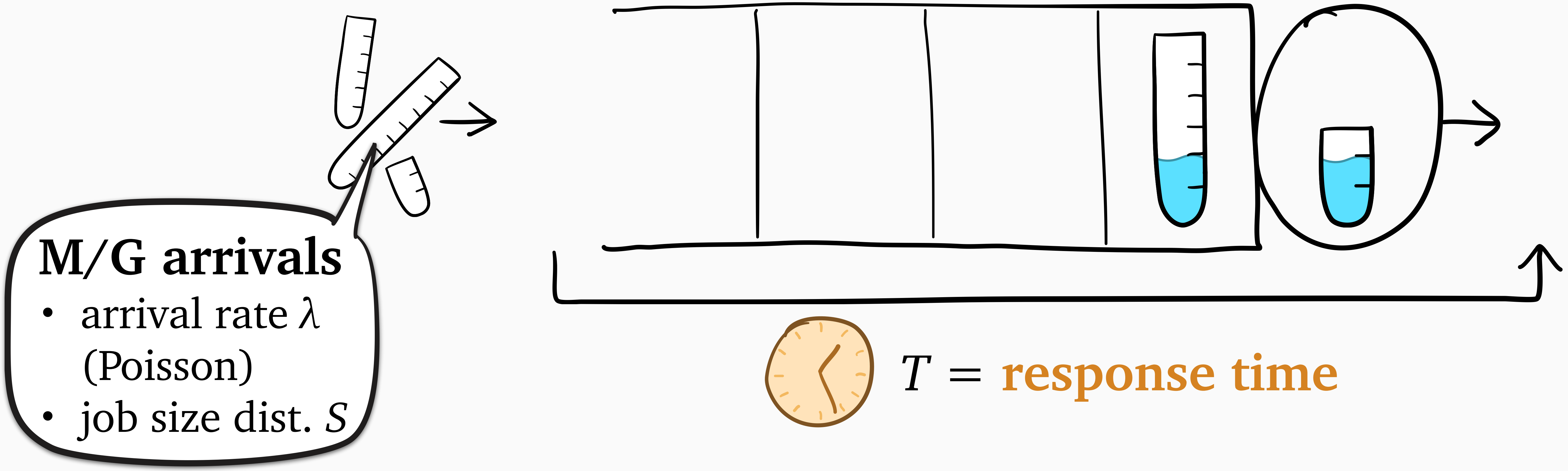
TCS

n arbitrary arrivals
T is tuple of n times

Queueing

infinite stochastic sequence of arrivals
T is a limiting distribution

TCS vs. Queueing



TCS

n arbitrary arrivals

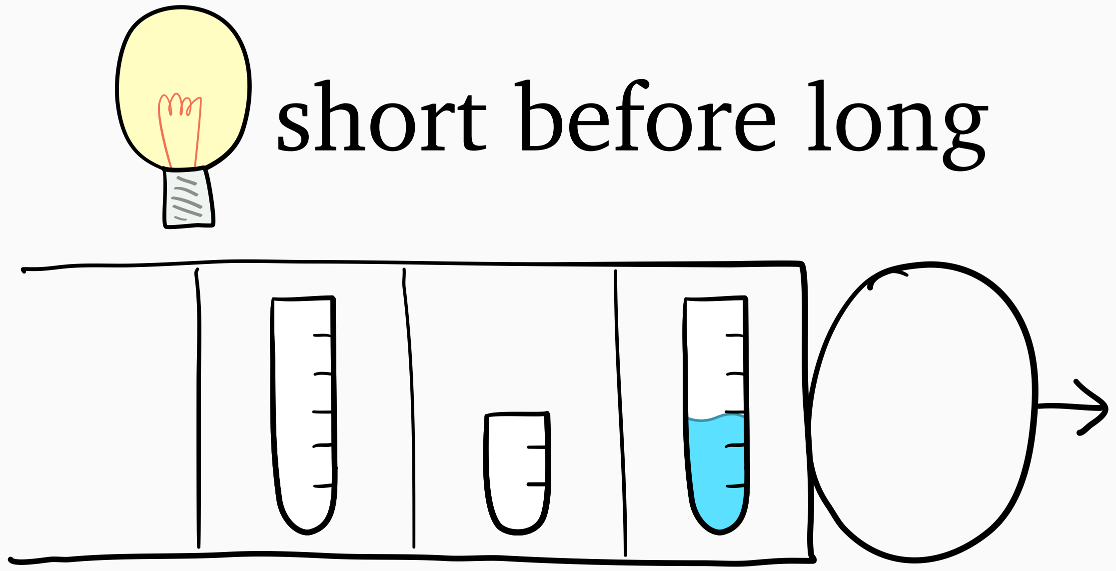
T is tuple of n times

Queueing

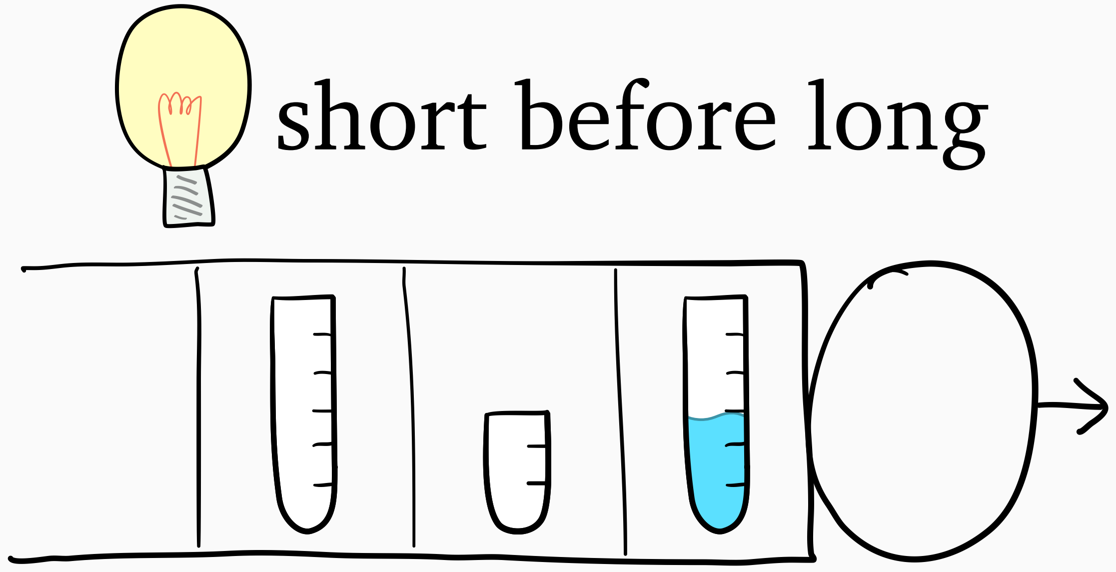
infinite stochastic sequence of arrivals

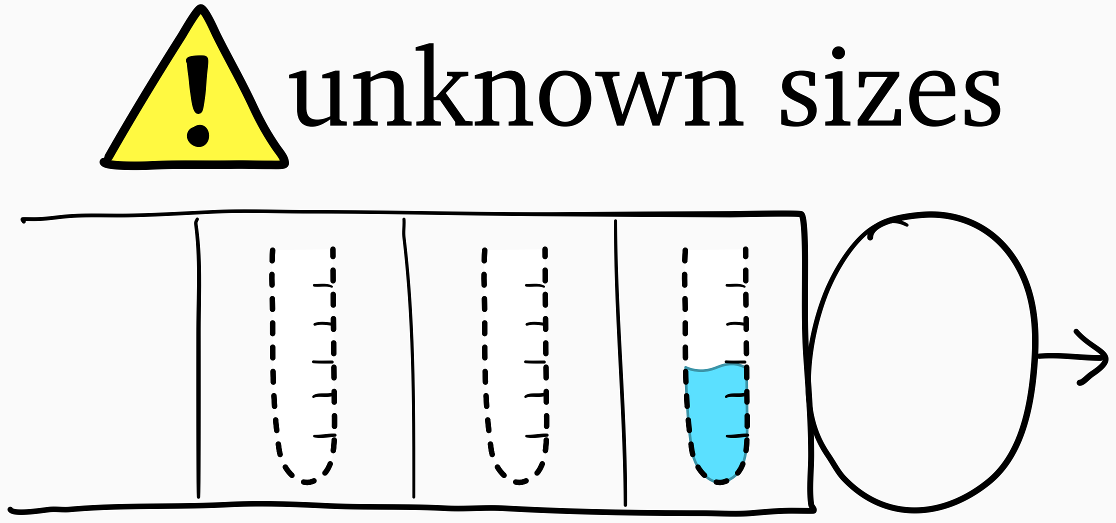
T is a limiting distribution

Job size uncertainty



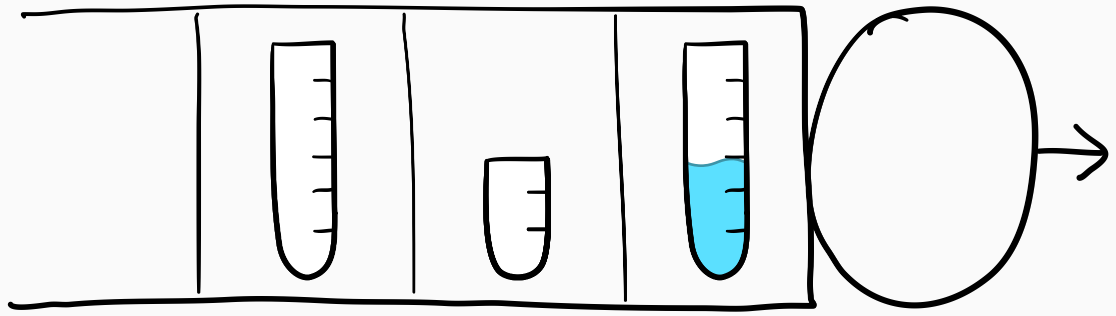
Job size uncertainty



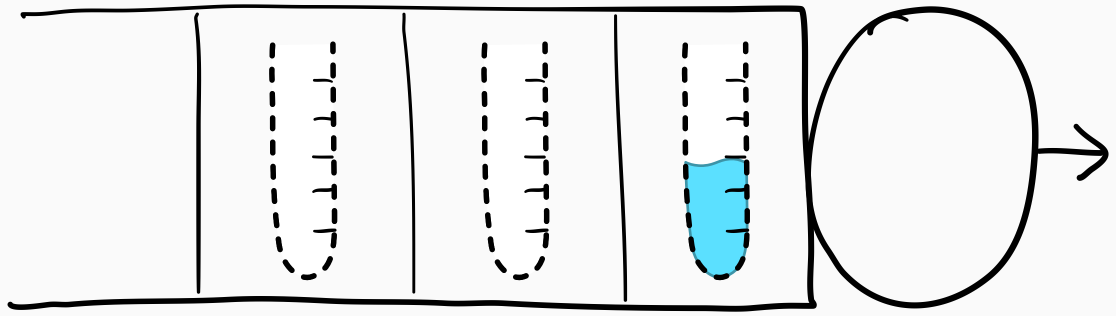


Job size uncertainty

short before long

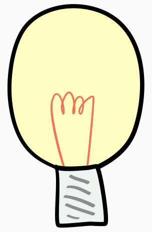


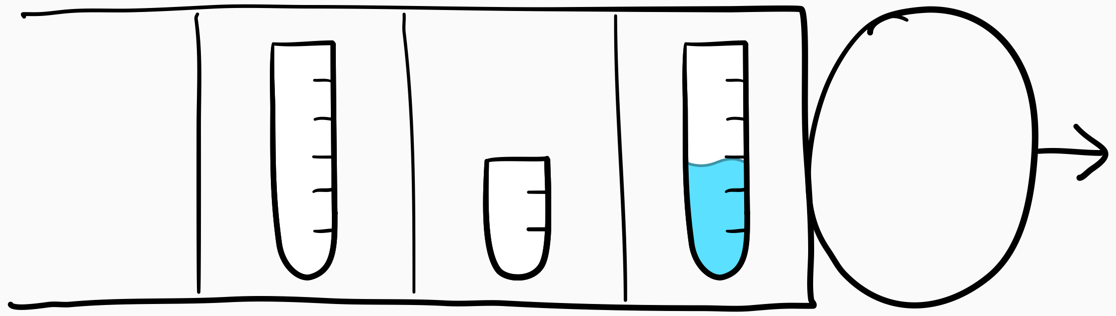
unknown sizes



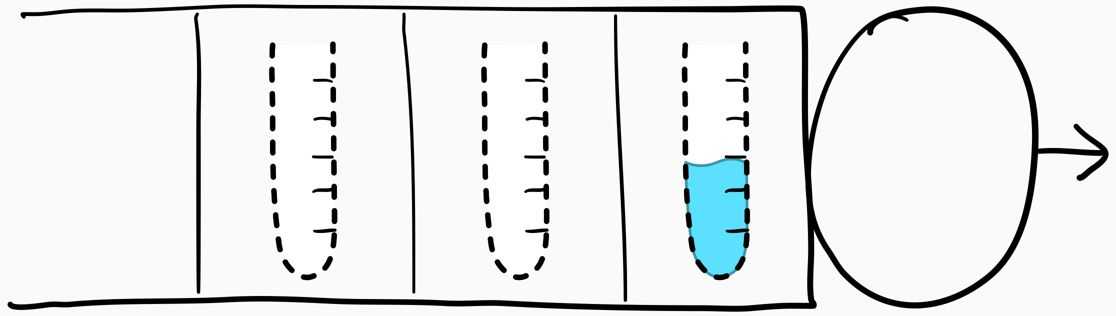
What info can we use?

Job size uncertainty

 short before long



 unknown sizes

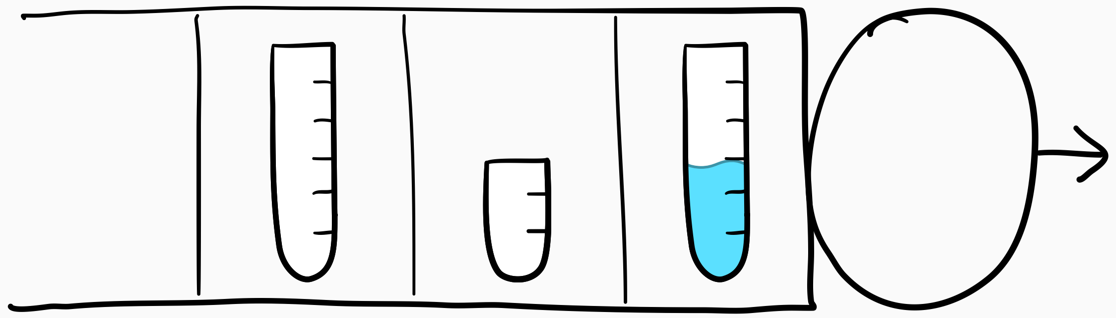


What info can we use?

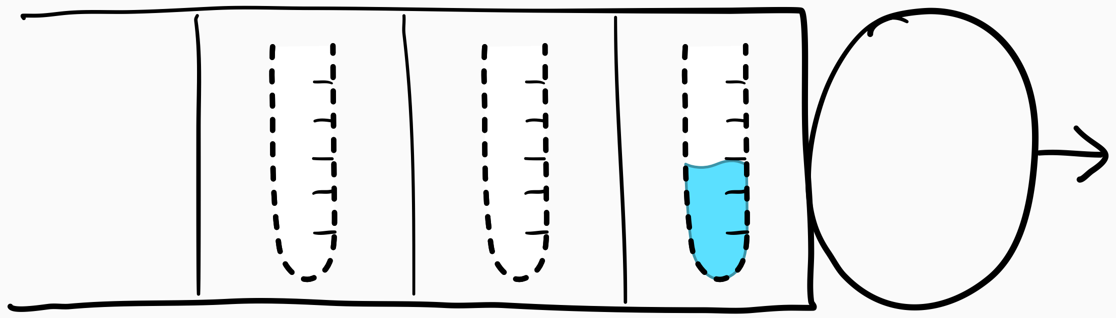
- job **ages**

Job size uncertainty

short before long



unknown sizes



What info can we use?

- job **ages**
- size distribution S

Scheduling with unknown job sizes

What info can we use?

- job **ages**
- size distribution S

Scheduling with unknown job sizes

What info can we use?

- job **ages**
- size distribution S

Gittins policy construction:

size distribution $S \mapsto$ policy **Gittins**(S)

Scheduling with unknown job sizes

What info can we use?

- job **ages**
- size distribution S

Gittins policy construction:

size distribution $S \mapsto$ policy **Gittins**(S)

age $a \mapsto$ priority **rank**(a)

Outline of Part 1

What is **Gittins**?

main focus

Why are **Gittins** (and **SRPT**) optimal?

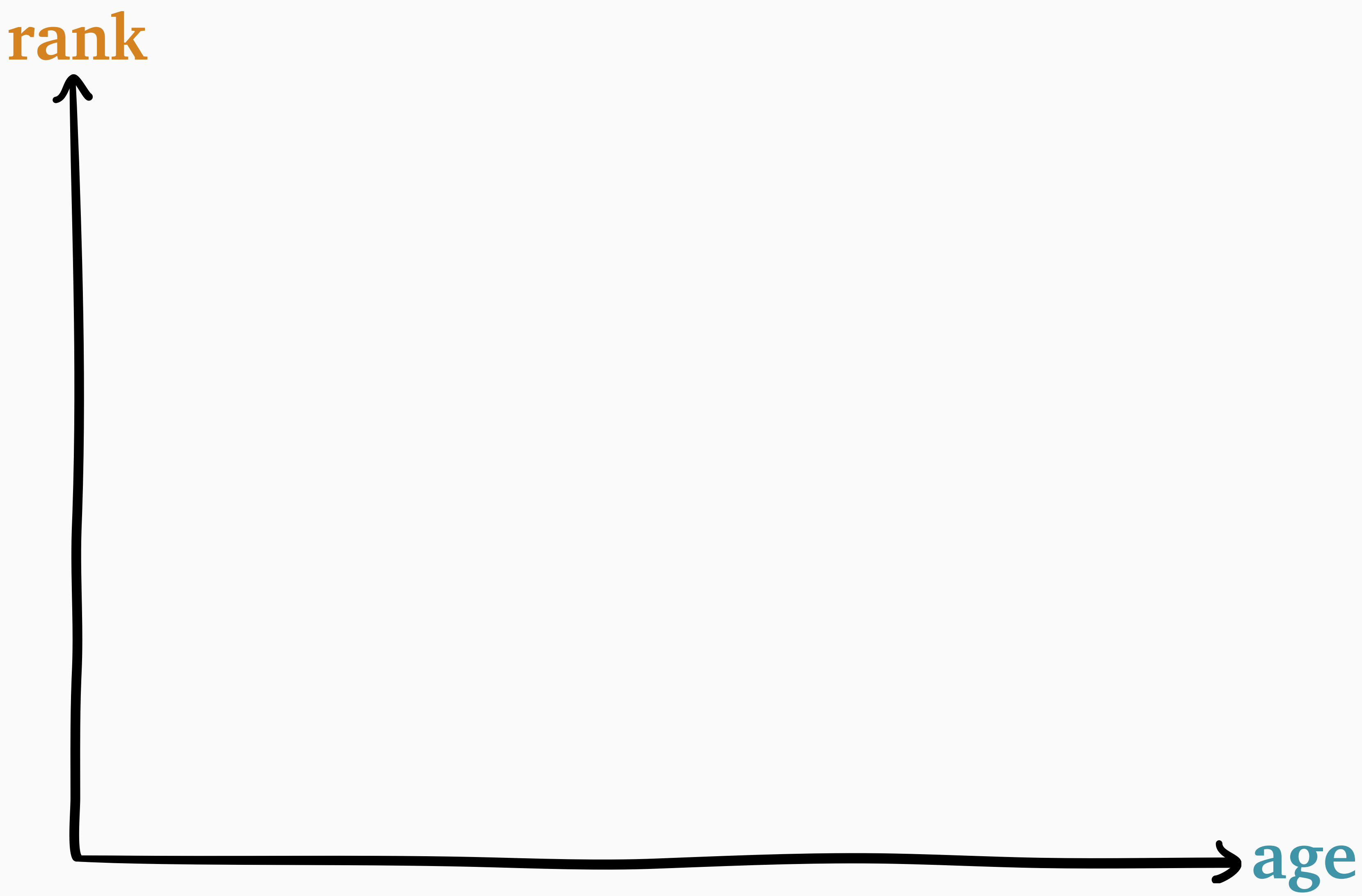
Predictions: what if we don't know exact distributions (or sizes)?

Scheduling with **rank** functions

Scheduling with rank functions

SERPT

shortest *expected* remaining processing time

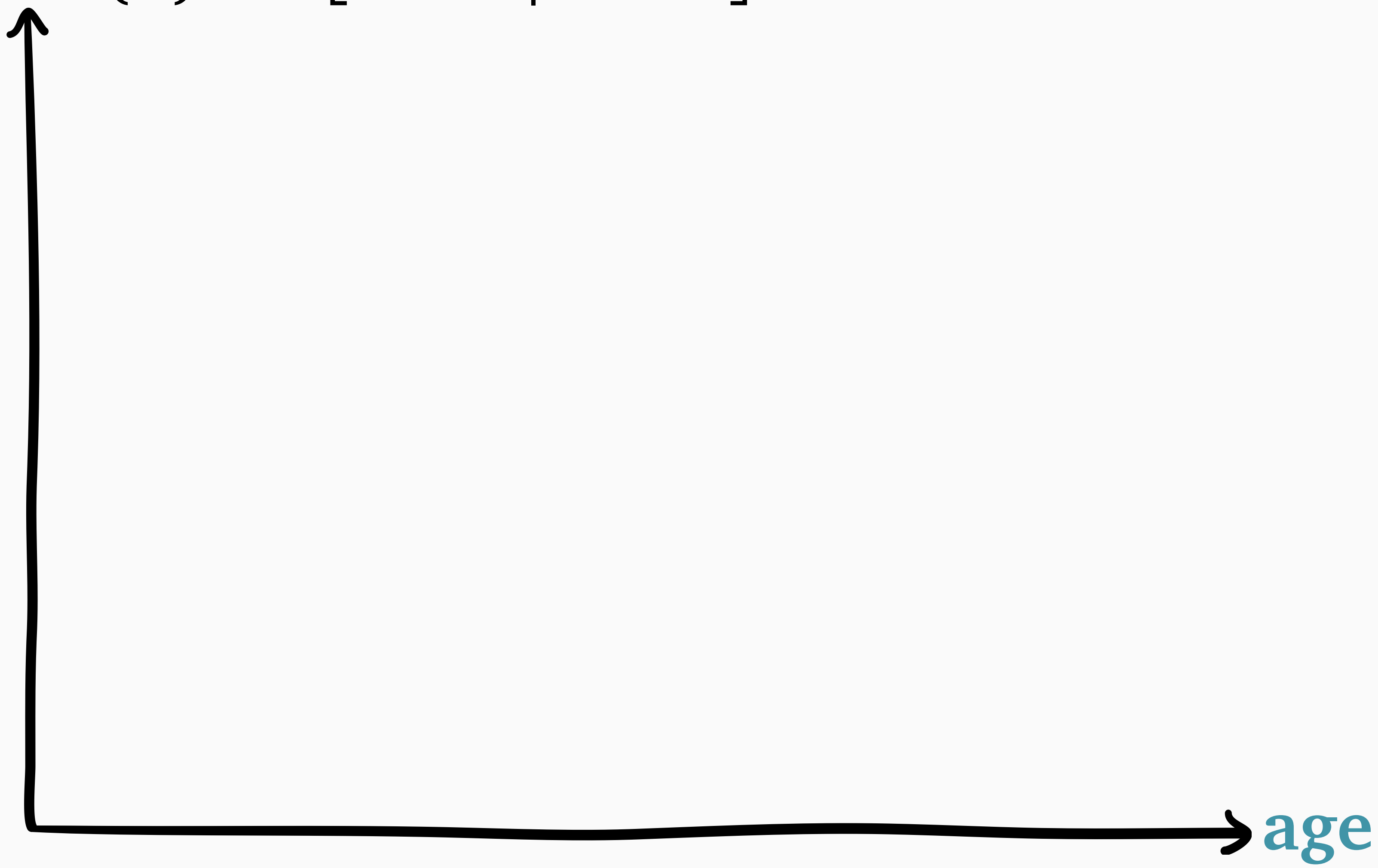


Scheduling with **rank** functions

SERPT

shortest *expected* remaining processing time

$$\text{rank}(a) = \mathbf{E}[S - a \mid S > a]$$



Scheduling with **rank** functions

SERPT

shortest *expected* remaining processing time

$\text{rank}(a) = \mathbf{E}[S - a \mid S > a]$

Job size distribution:

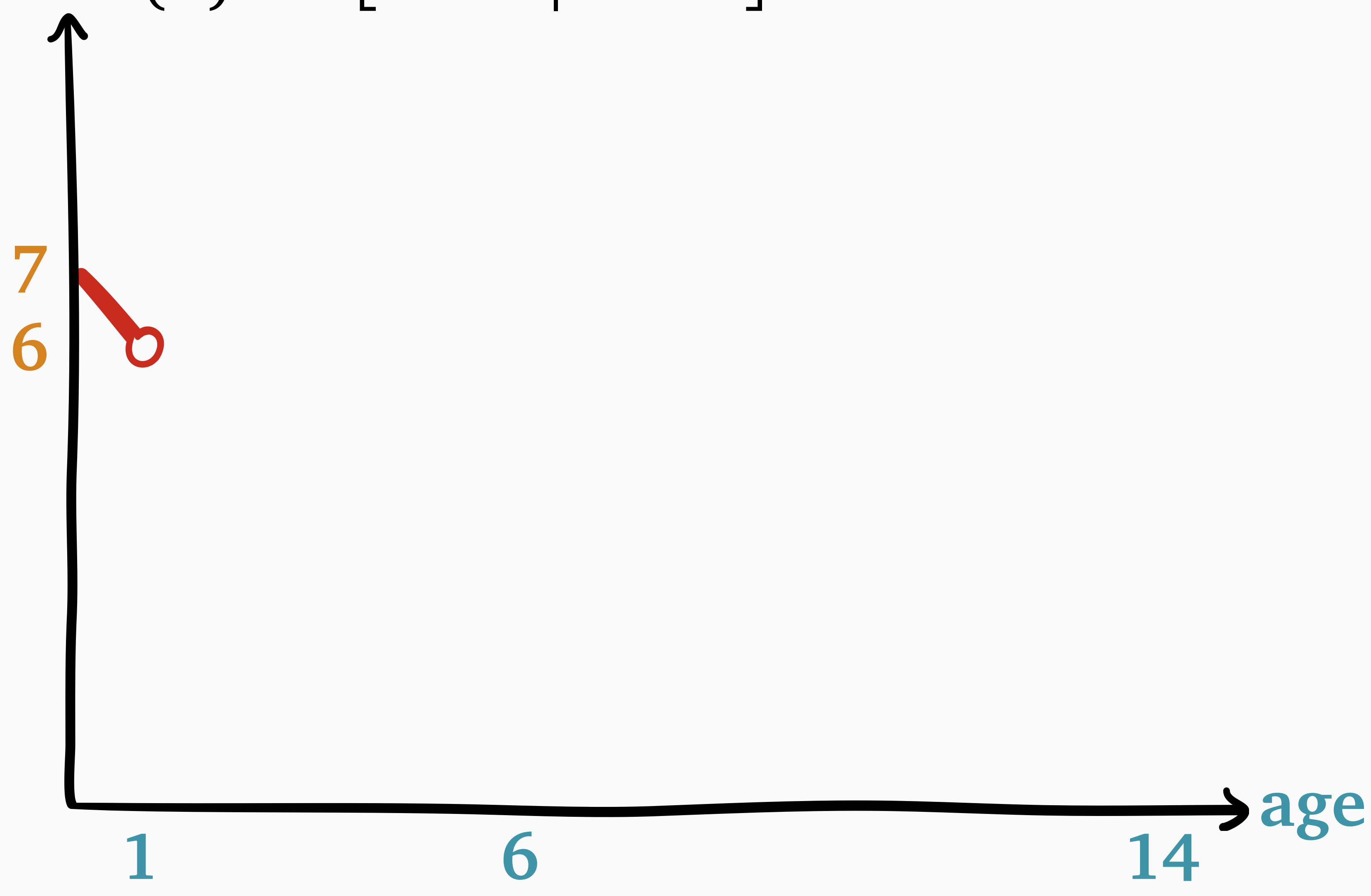
$$S = \begin{cases} 1 & \text{w.p. } \frac{1}{3} \\ 6 & \text{w.p. } \frac{1}{3} \\ 14 & \text{w.p. } \frac{1}{3} \end{cases}$$

Scheduling with rank functions

SERPT

shortest *expected* remaining processing time

$\mathrm{rank}(a) = \mathbf{E}[S - a \mid S > a]$



Job size distribution:

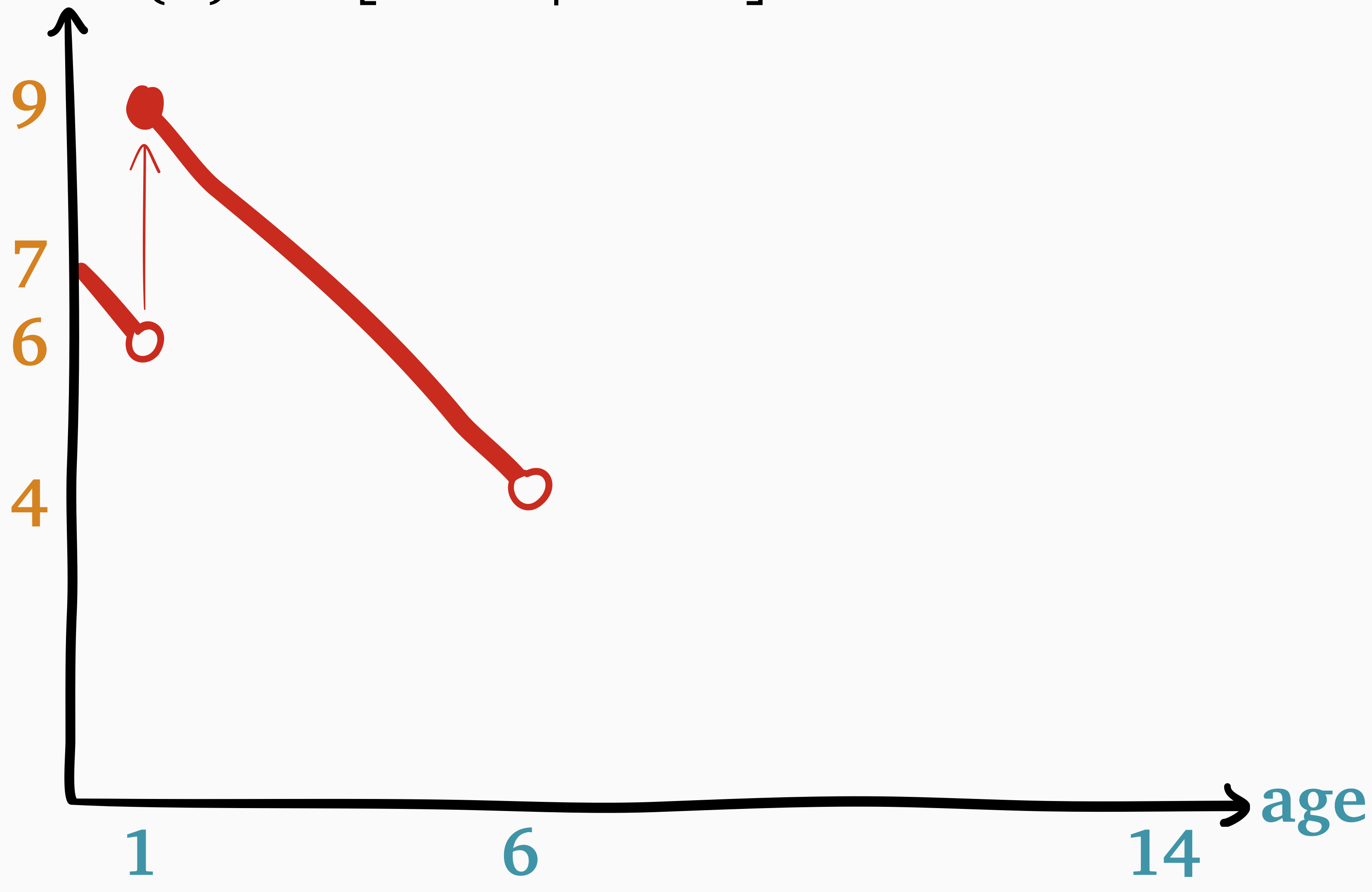
$$S = \begin{cases} 1 & \text{w.p. } \frac{1}{3} \\ 6 & \text{w.p. } \frac{1}{3} \\ 14 & \text{w.p. } \frac{1}{3} \end{cases}$$

Scheduling with rank functions

SERPT

shortest *expected* remaining processing time

$\text{rank}(a) = \mathbf{E}[S - a \mid S > a]$



Job size distribution:

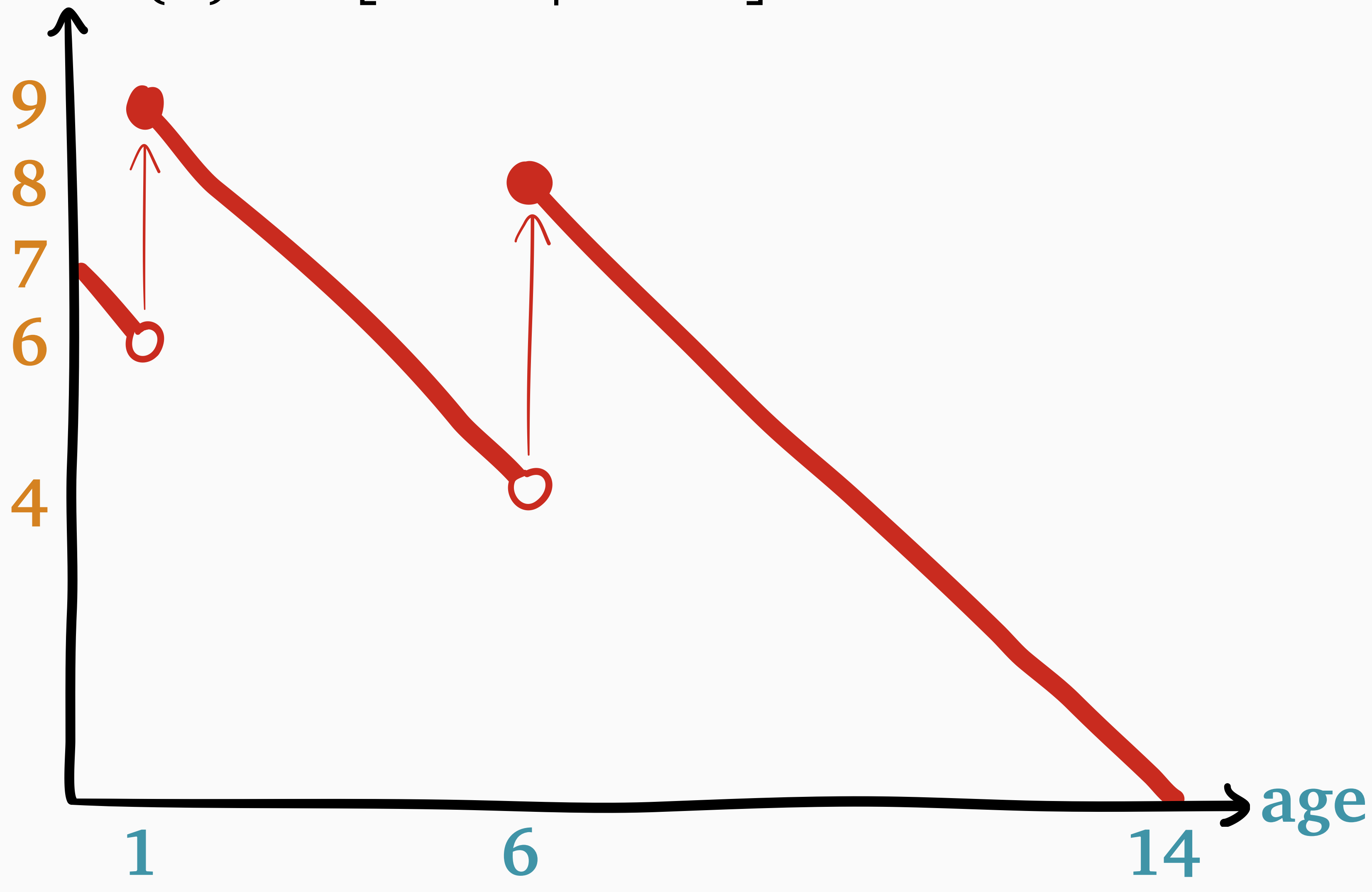
$$S = \begin{cases} 1 & \text{w.p. } \frac{1}{3} \\ 6 & \text{w.p. } \frac{1}{3} \\ 14 & \text{w.p. } \frac{1}{3} \end{cases}$$

Scheduling with rank functions

SERPT

shortest *expected* remaining processing time

$\text{rank}(a) = \mathbf{E}[S - a \mid S > a]$



Job size distribution:

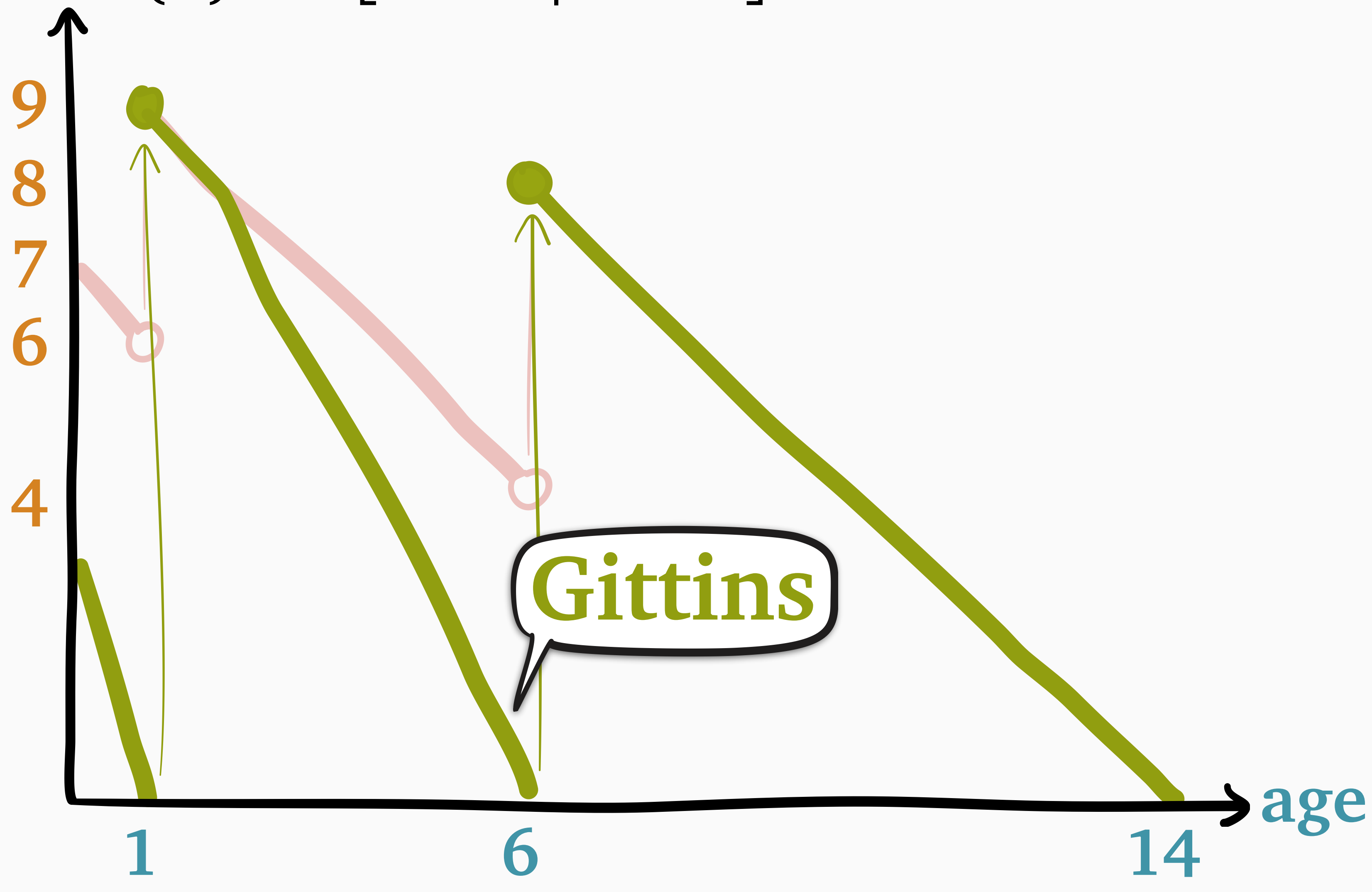
$$S = \begin{cases} 1 & \text{w.p. } \frac{1}{3} \\ 6 & \text{w.p. } \frac{1}{3} \\ 14 & \text{w.p. } \frac{1}{3} \end{cases}$$

Scheduling with rank functions

SERPT

shortest *expected* remaining processing time

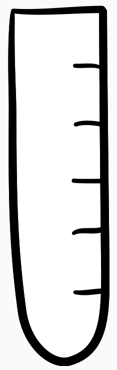
$\mathrm{rank}(a) = \mathbf{E}[S - a \mid S > a]$



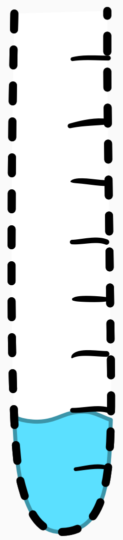
Job size distribution:

$$S = \begin{cases} 1 & \text{w.p. } \frac{1}{3} \\ 6 & \text{w.p. } \frac{1}{3} \\ 14 & \text{w.p. } \frac{1}{3} \end{cases}$$

Defining the Gittins rank

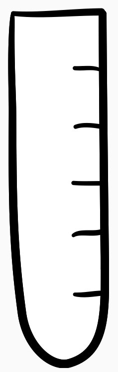


known
size r

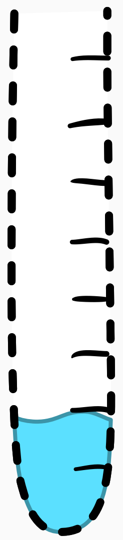


unknown size
$(S \mid S > \text{age})$

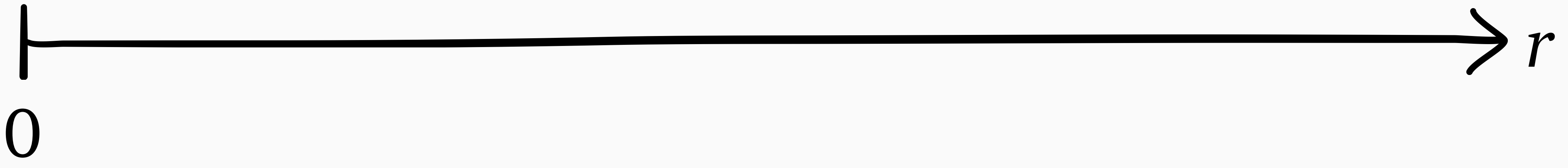
Defining the Gittins rank



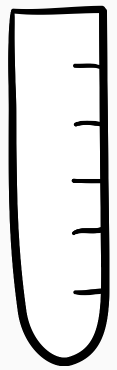
known
size r



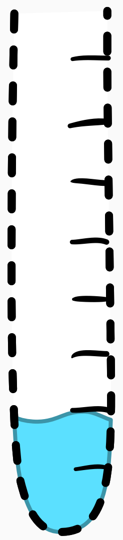
unknown size
$(S \mid S > \text{age})$



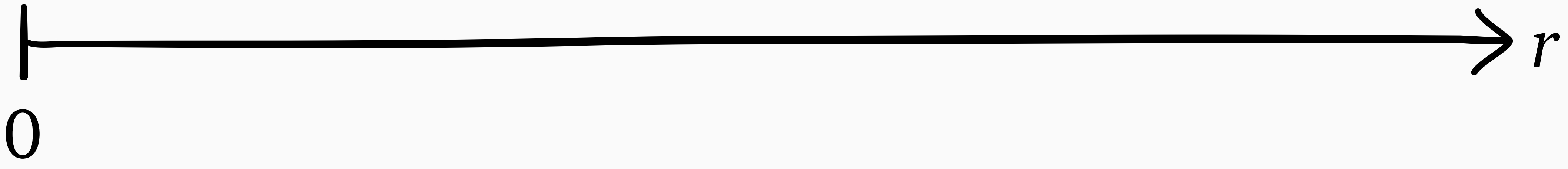
Defining the Gittins rank



known
size r

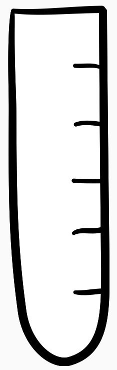


unknown size
$(S \mid S > \mathbf{age})$

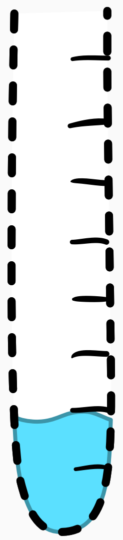


Key question: for which r should we serve **unknown**?

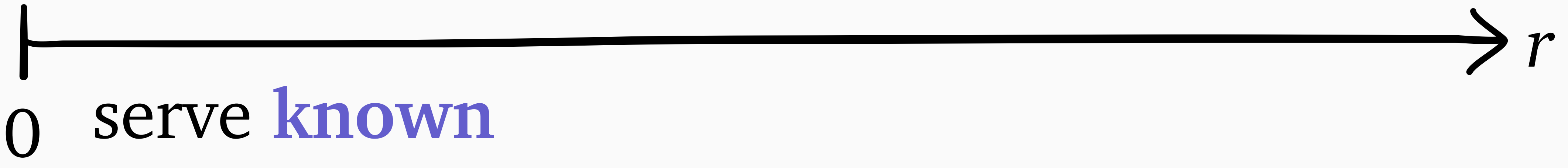
Defining the Gittins rank



known
size r

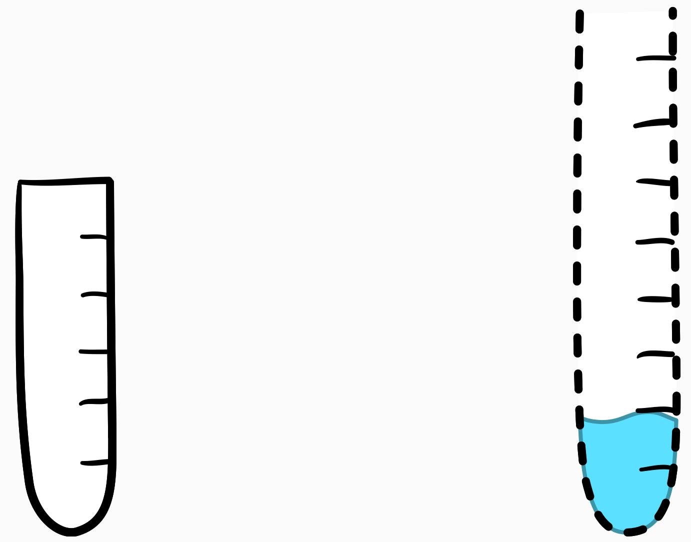


unknown size
$(S \mid S > \textbf{age})$



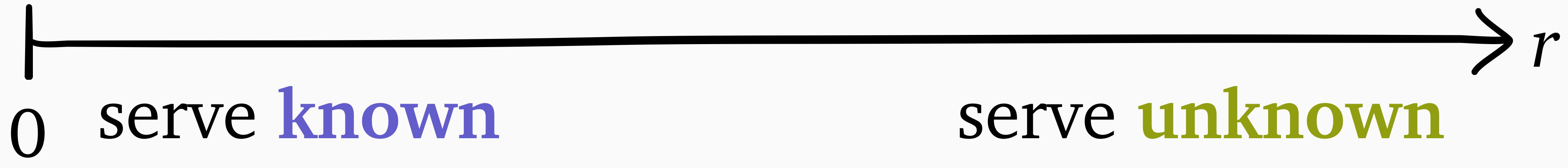
Key question: for which r should we serve **unknown**?

Defining the Gittins rank



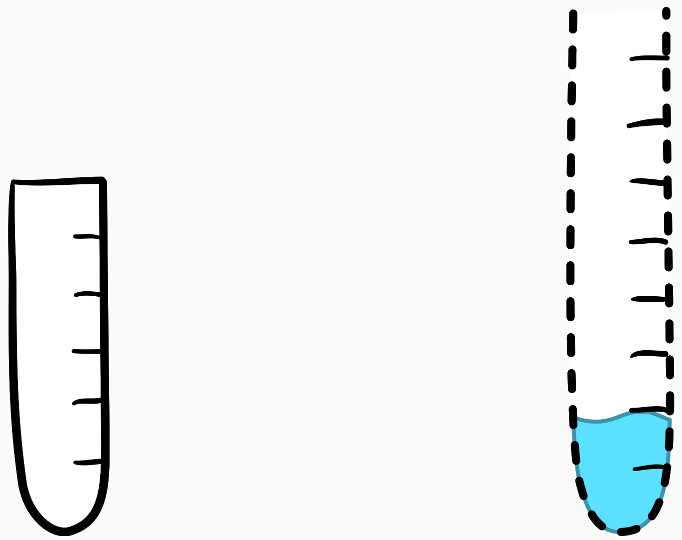
known size r

unknown size $(S \mid S > \text{age})$



Key question: for which r should we serve unknown?

Defining the Gittins rank

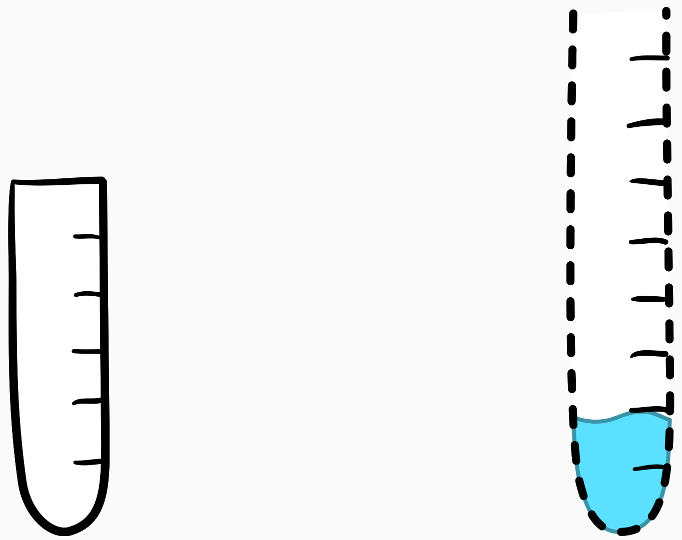


known size r

unknown size $(S \mid S > \textbf{age})$

Key question: for which r should we serve **unknown**?

Defining the Gittins rank



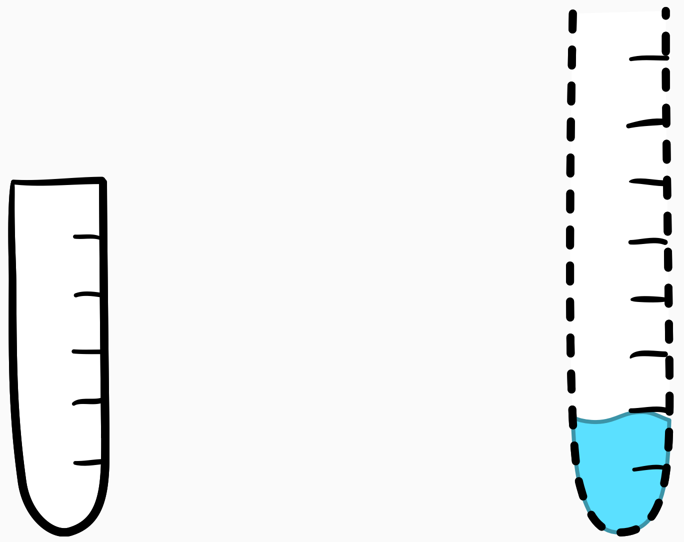
known
size r

unknown size
$(S \mid S > \text{age})$

1 (50%)
19 (50%)

Key question: for which r should we serve **unknown**?

Defining the Gittins rank

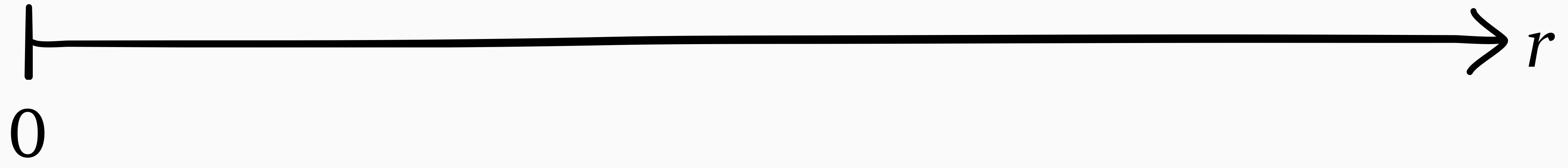


known size r

unknown size $(S \mid S > \textbf{age})$

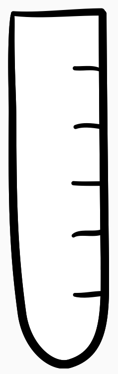
1 (50%)

19 (50%)

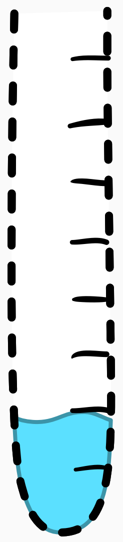


Key question: for which r should we serve **unknown**?

Defining the Gittins rank

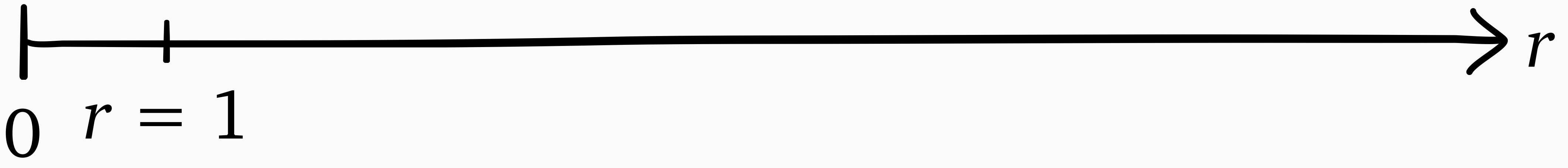


known
size r



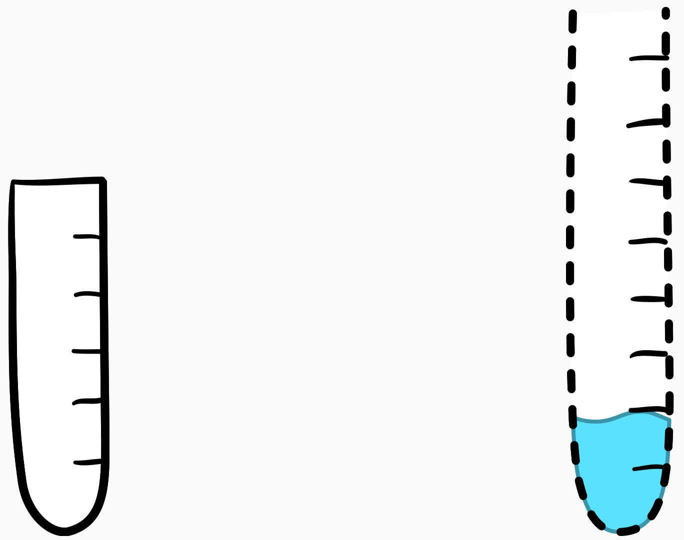
unknown size
$(S \mid S > \textbf{age})$

1 (50%)
19 (50%)



Key question: for which r should we serve **unknown**?

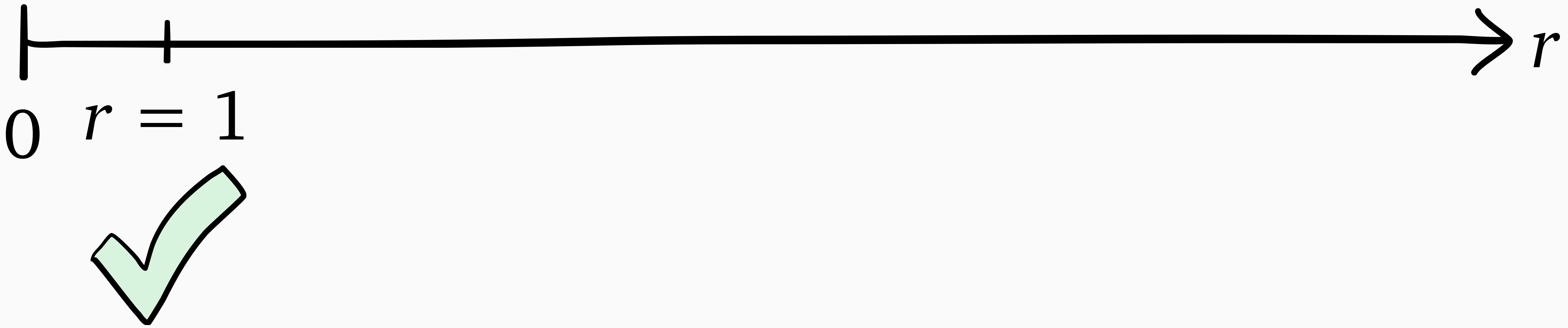
Defining the Gittins rank



known size r

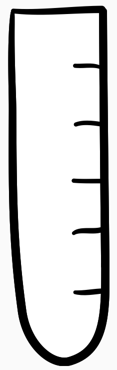
unknown size $(S \mid S > \textbf{age})$

1 (50%)
19 (50%)

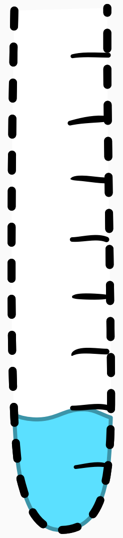


Key question: for which r should we serve **unknown**?

Defining the Gittins rank

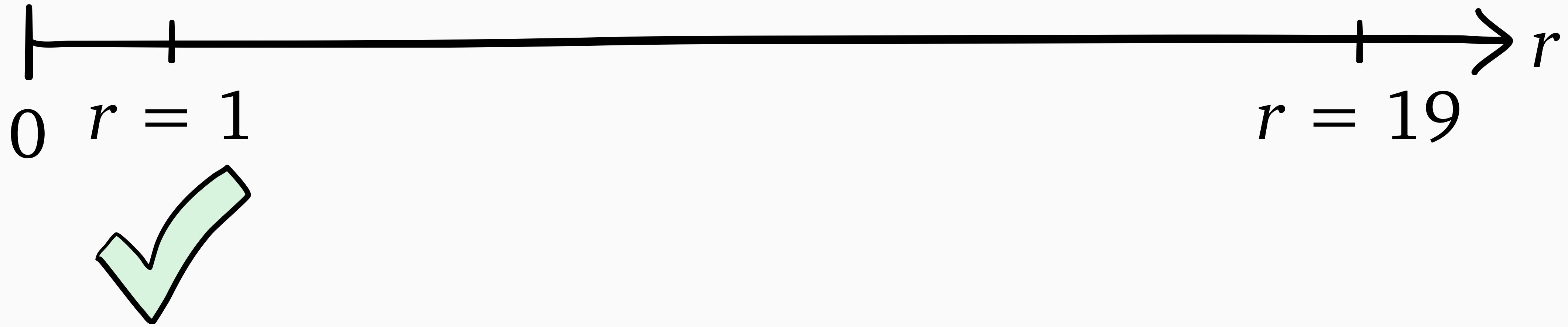


known size r



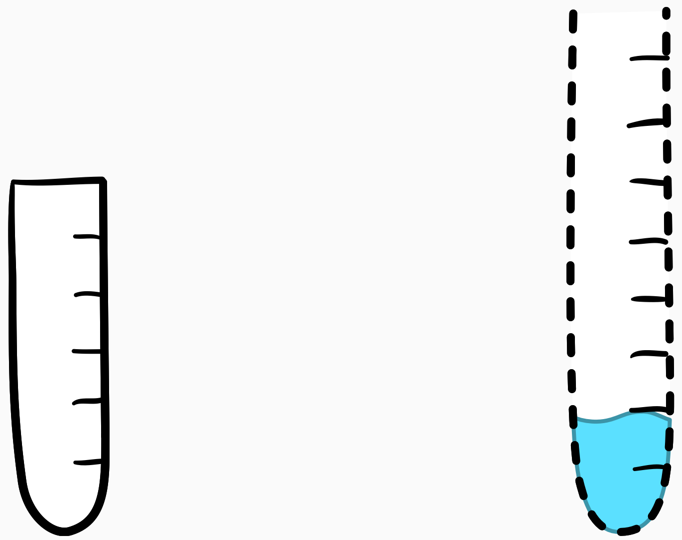
unknown size $(S \mid S > \textbf{age})$

1	(50%)
19	(50%)



Key question: for which r should we serve **unknown**?

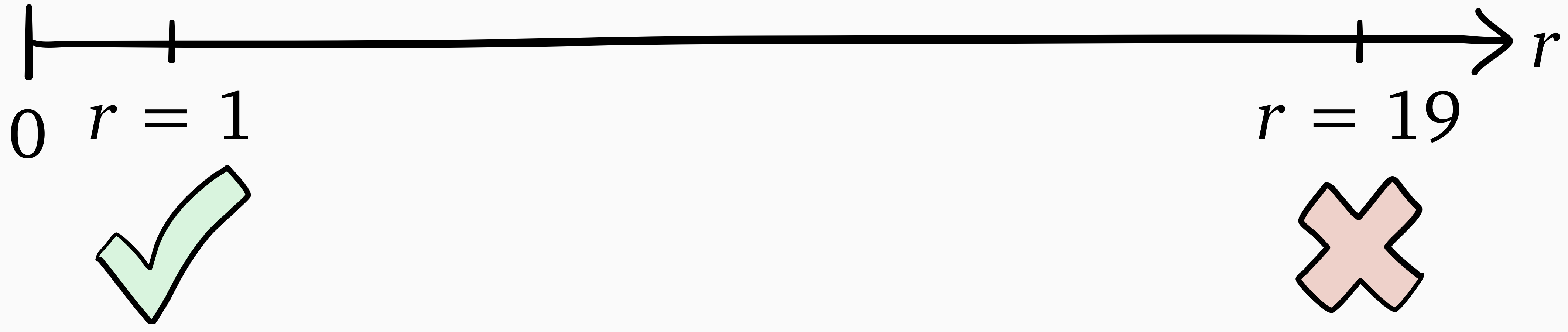
Defining the Gittins rank



known size r

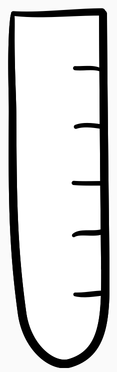
unknown size $(S \mid S > \text{age})$

1 (50%)
19 (50%)

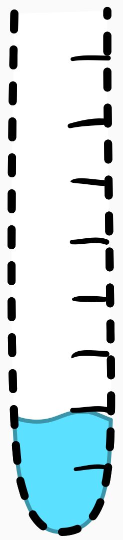


Key question: for which r should we serve **unknown**?

Defining the Gittins rank

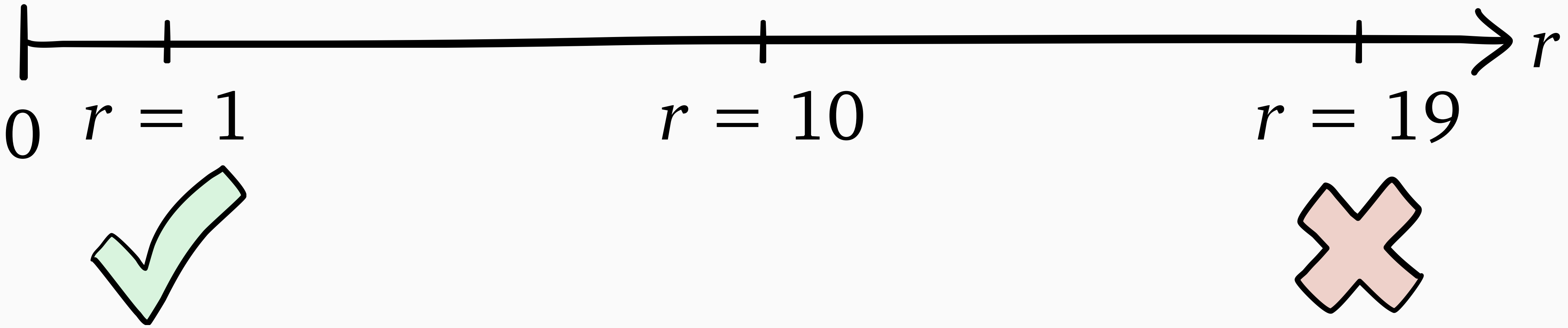


known
size r



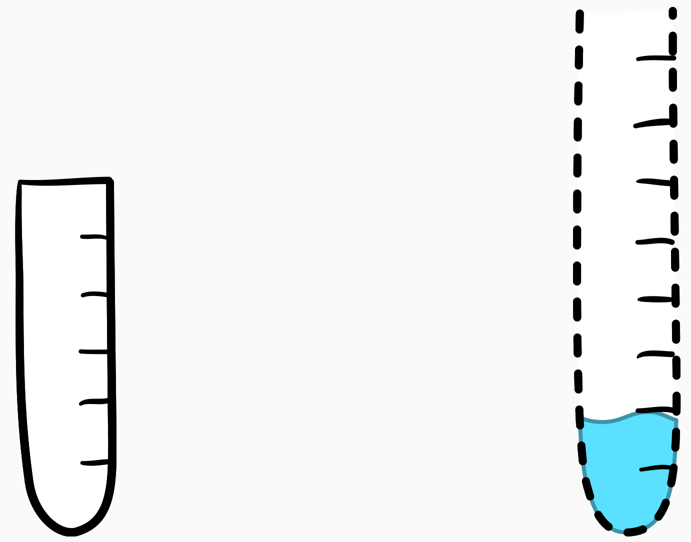
unknown size
$(S \mid S > \textbf{age})$

1 (50%)
19 (50%)



Key question: for which r should we serve **unknown**?

Defining the Gittins rank

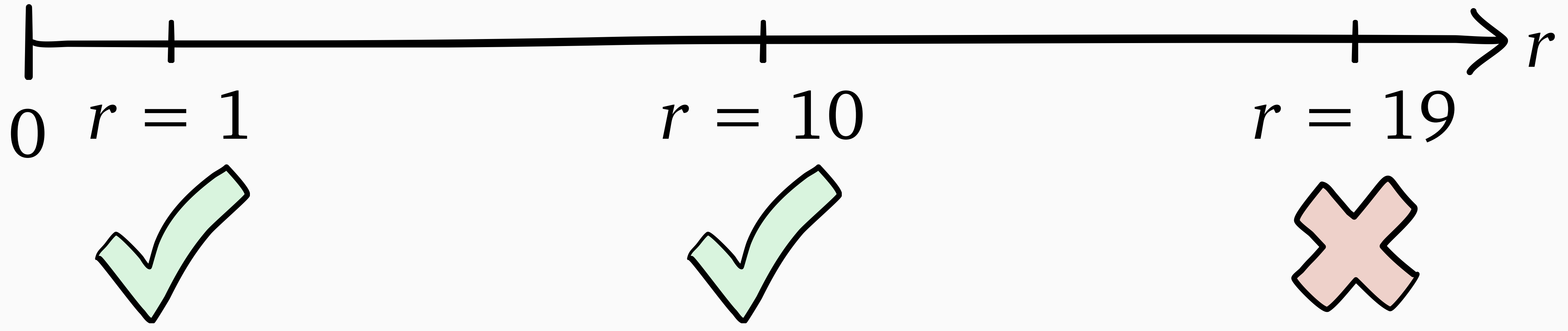


known size r

unknown size $(S \mid S > \textbf{age})$

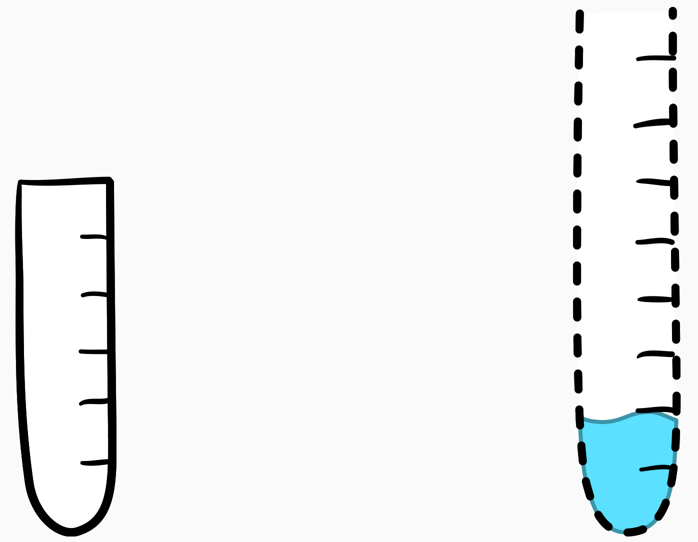
1 (50%)

19 (50%)



Key question: for which r should we serve **unknown**?

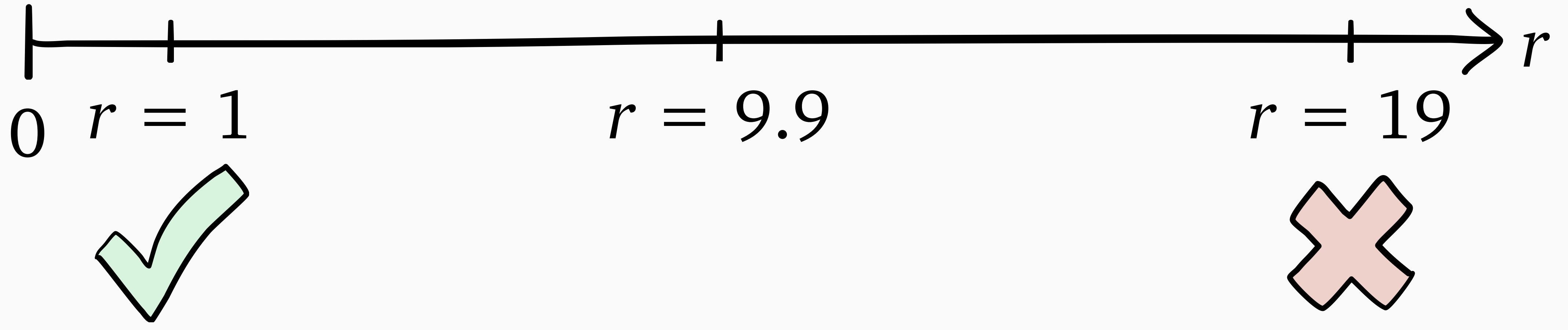
Defining the **Gittins rank**



known
size r

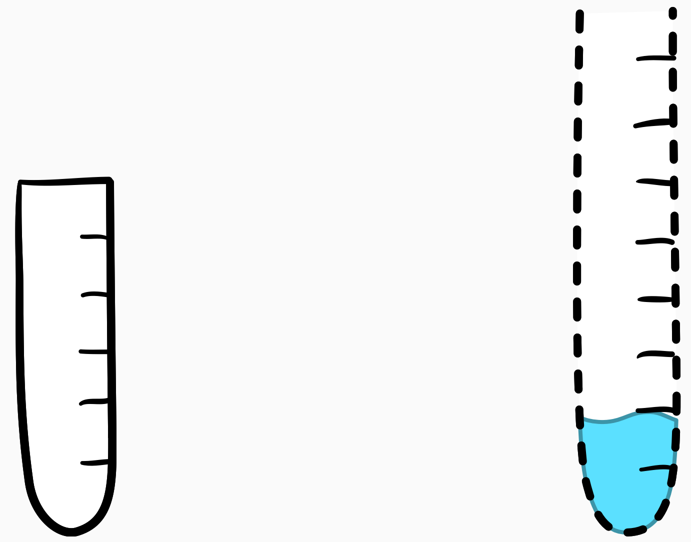
unknown size
$(S \mid S > \textbf{age})$

1 (50%)
19 (50%)



Key question: for which r should we serve **unknown**?

Defining the Gittins rank

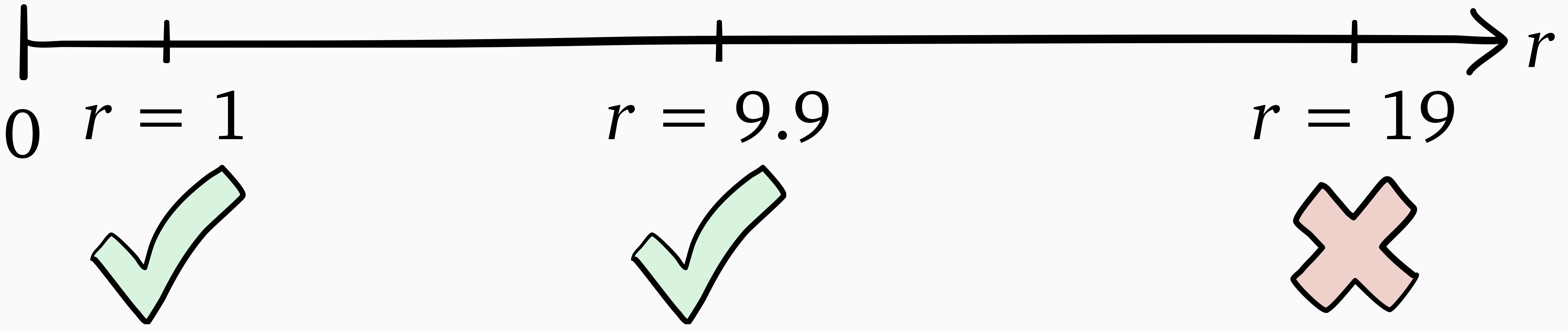


known size r

unknown size $(S \mid S > \text{age})$

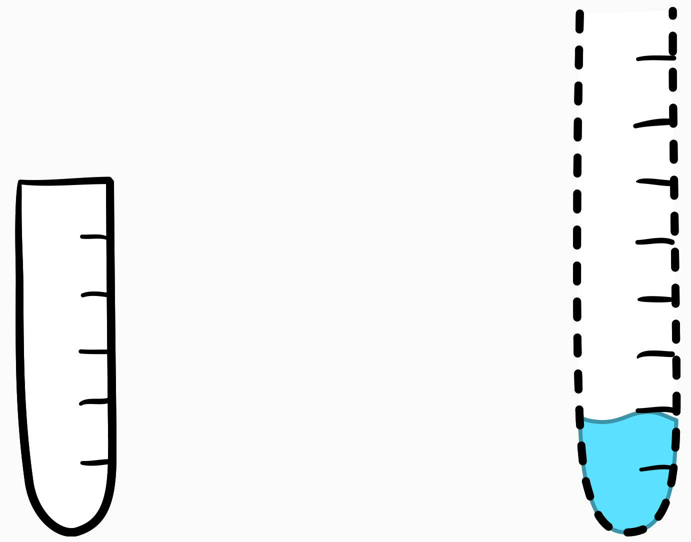
1 (50%)

19 (50%)



Key question: for which r should we serve **unknown**?

Defining the Gittins rank

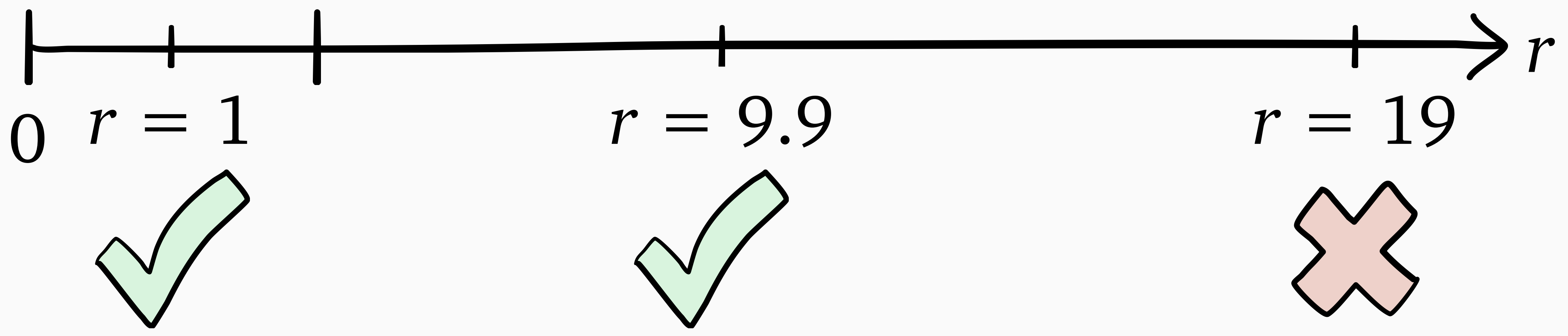


known size r

unknown size $(S \mid S > \text{age})$

1 (50%)
19 (50%)

rank = 2



Key question: for which r should we serve **unknown**?

How to prove **SRPT** is optimal?

How to prove SRPT is optimal?

Little's law: $\mathbf{E}[N] = \lambda \mathbf{E}[T]$

How to prove SRPT is optimal?

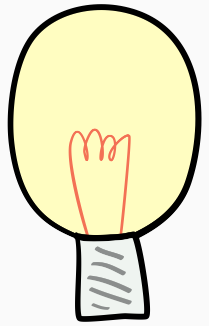
Little's law: $\mathbf{E}[N] = \lambda \mathbf{E}[T]$

jobs present

How to prove **SRPT** is optimal?

Little's law: $\mathbf{E}[N] = \lambda\mathbf{E}[T]$

jobs present

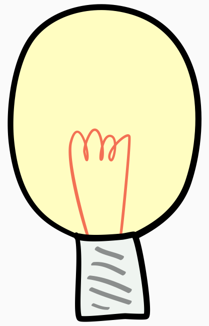


Reward each time
we complete a job

How to prove SRPT is optimal?

Little's law: $\mathbf{E}[N] = \lambda \mathbf{E}[T]$

jobs present



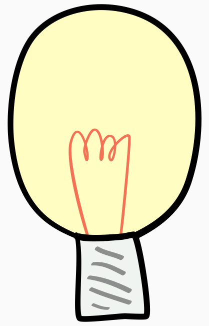
Reward each time we complete a job

Rewards aren't immediate

How to prove SRPT is optimal?

Little's law: $\mathbf{E}[N] = \lambda \mathbf{E}[T]$

jobs present



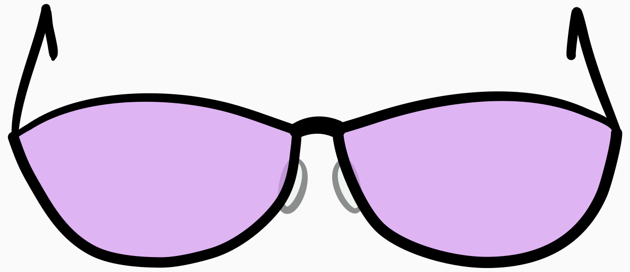
Reward each time we complete a job

Rewards aren't immediate

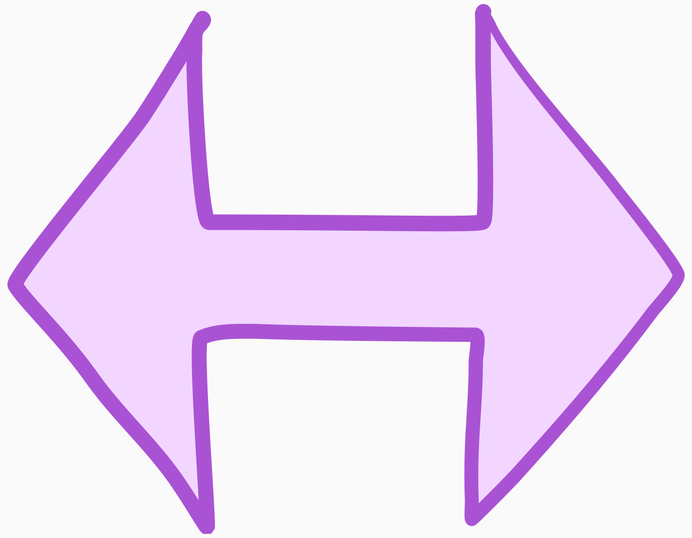
Write N in terms of "smoother" quantity?

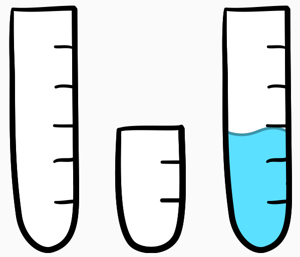
WINE

Work **I**ntegral **N**umber **E**quality



r-work $W(r)$

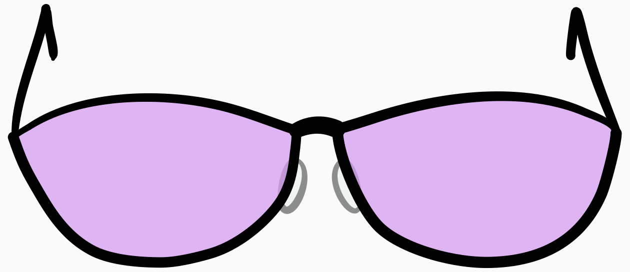




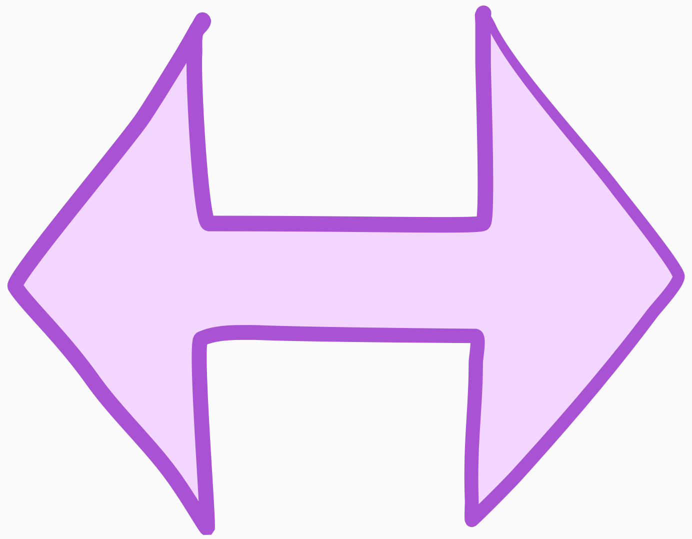
number of jobs N

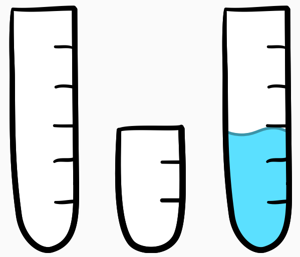
WINE

Work Integral Number Equality



r-work $W(r)$



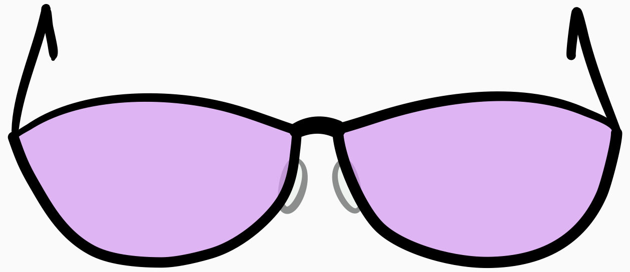


number of jobs N

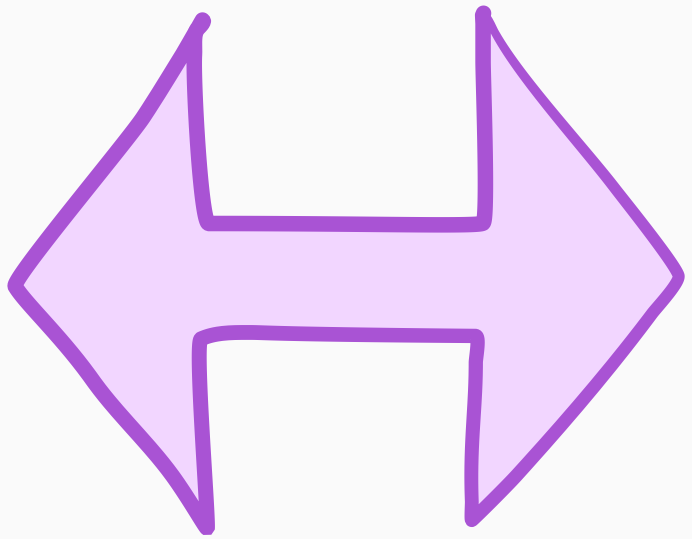
$\mathbf{E}[T]$ bounds for SRPT and Gittins in:

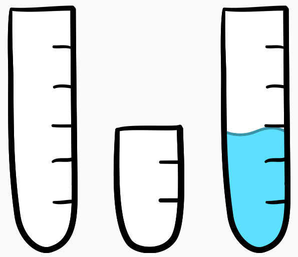
WINE

Work **I**ntegral **N**umber **E**quality



r-work $W(r)$

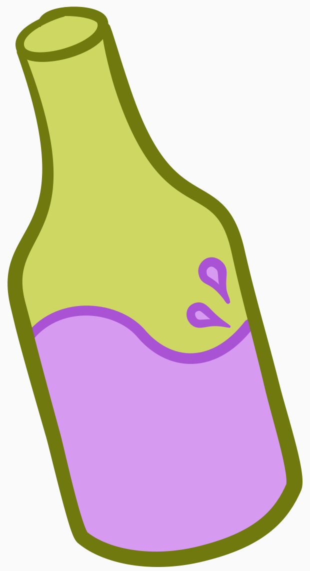




number of jobs N

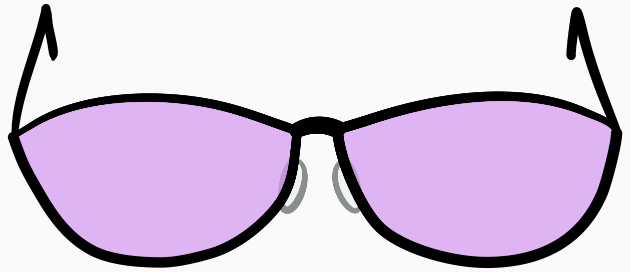
$\mathbf{E}[T]$ bounds for **SRPT** and **Gittins** in:

- **M/G/k** and **G/G/k**

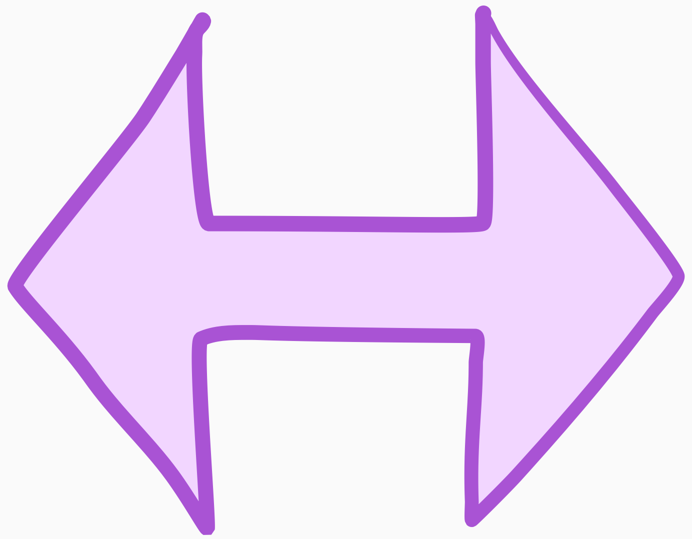


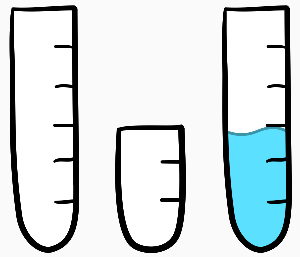
WINE

Work Integral Number Equality



r-work $W(r)$





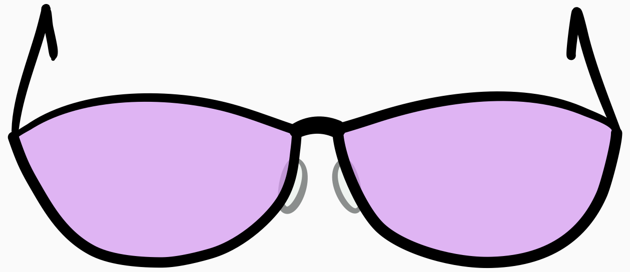
number of jobs N

$\mathbf{E}[T]$ bounds for **SRPT** and **Gittins** in:

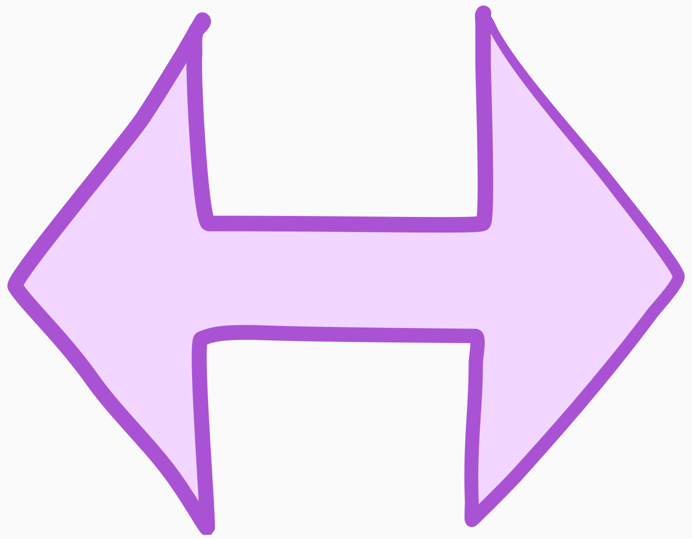
- **M/G/k** and **G/G/k**
- systems with **multiserver jobs**

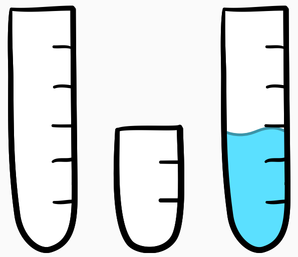
WINE

Work **I**ntegral **N**umber **E**quality



r-work $W(r)$





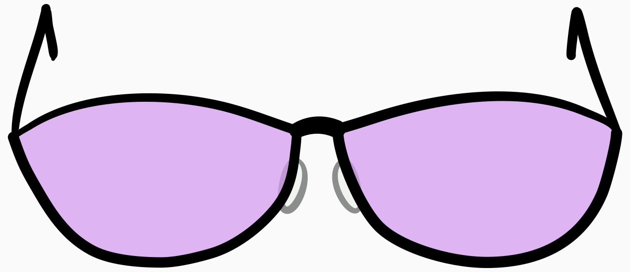
number of jobs N

$\mathbf{E}[T]$ bounds for **SRPT** and **Gittins** in:

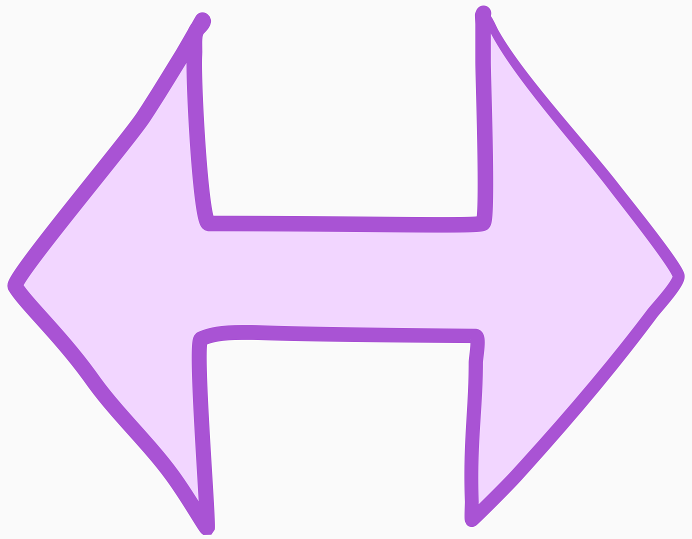
- **M/G/k** and **G/G/k**
- systems with **multiserver jobs**
- systems with **noisy size estimates**

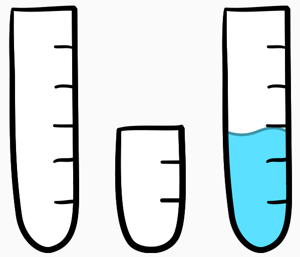
WINE

Work **I**ntegral **N**umber **E**quality



r-work $W(r)$



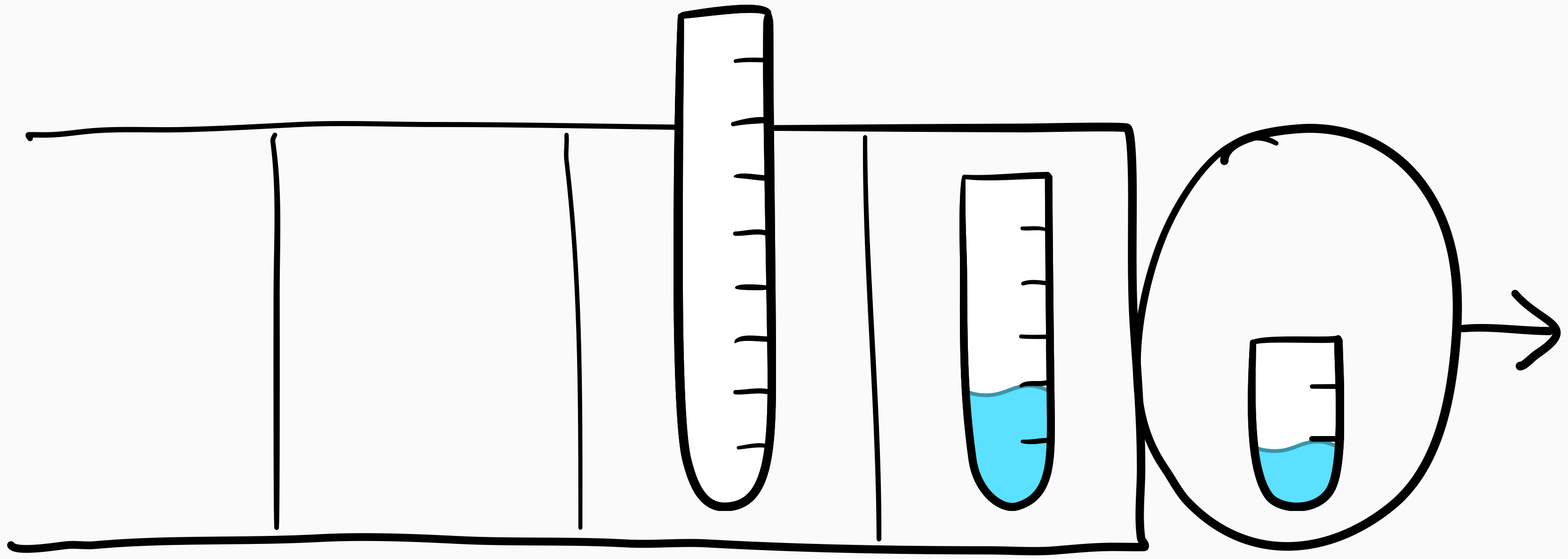


number of jobs N

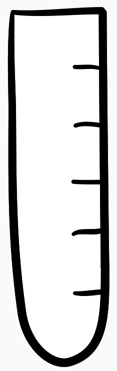
$\mathbf{E}[T]$ bounds for **SRPT** and **Gittins** in:

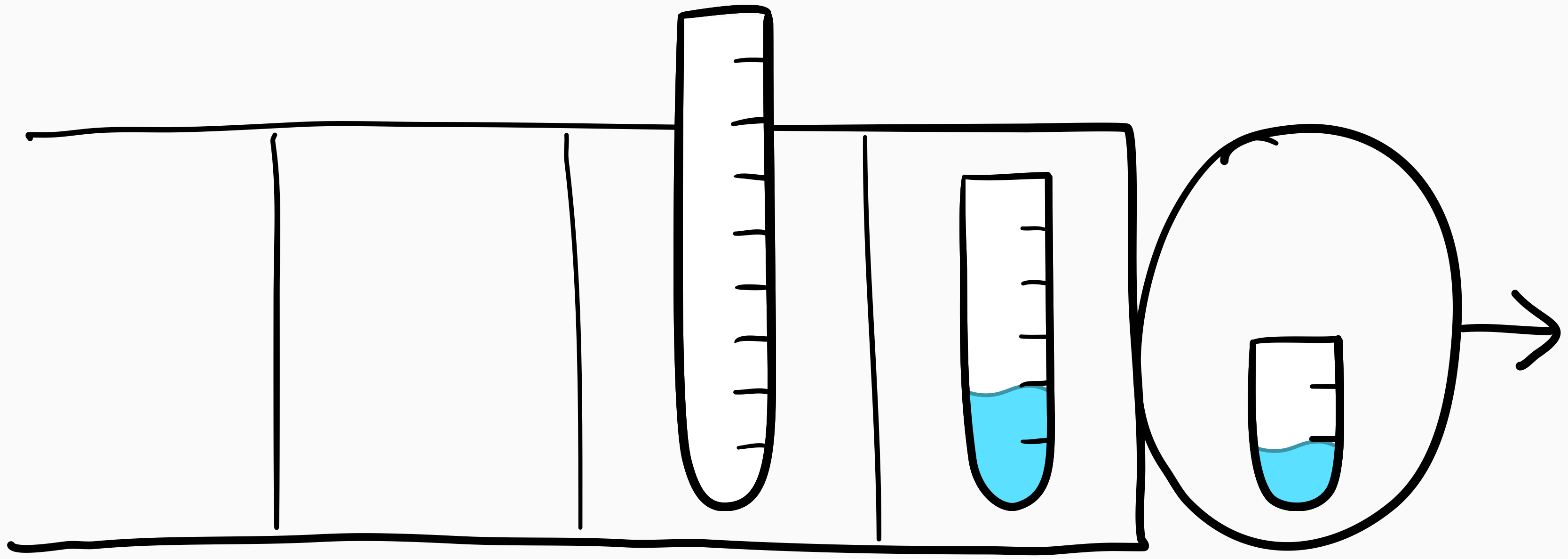
- **M/G/k** and **G/G/k**
- systems with **multiserver jobs**
- systems with **noisy size estimates**
- systems with **unknown size distribution**

Key quantity: r-work

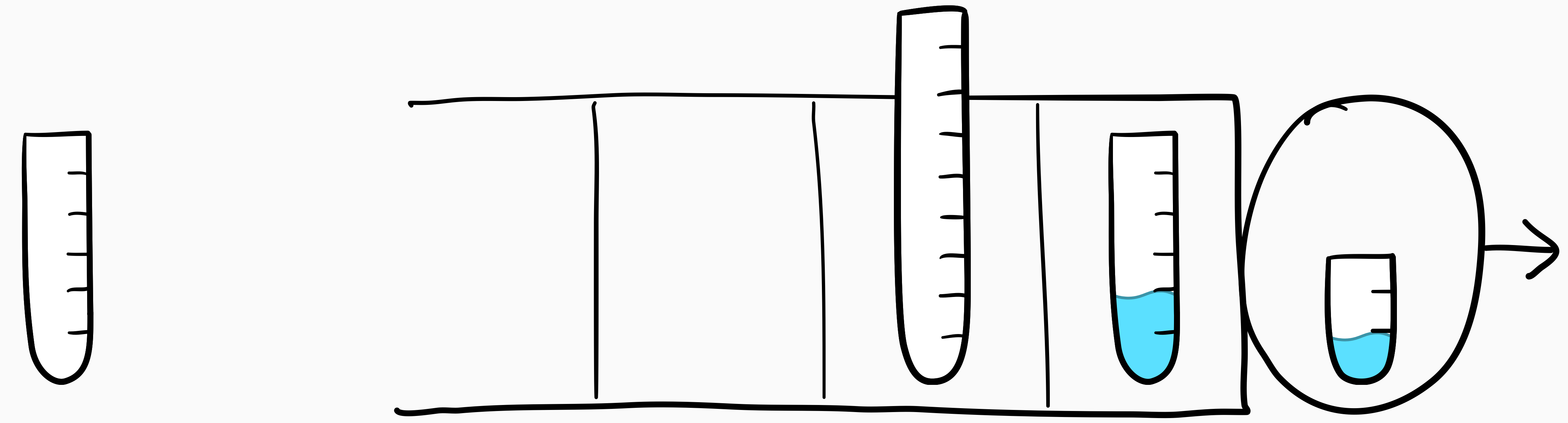


Key quantity: r-work


“me”

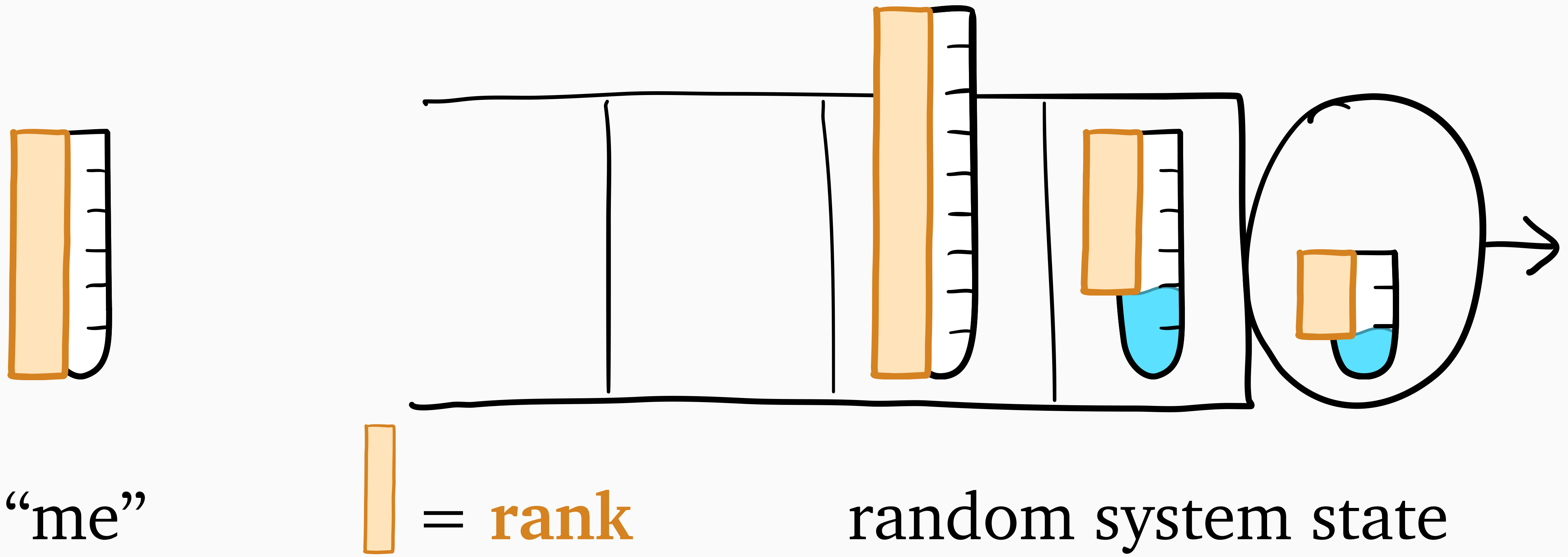


Key quantity: *r*-work

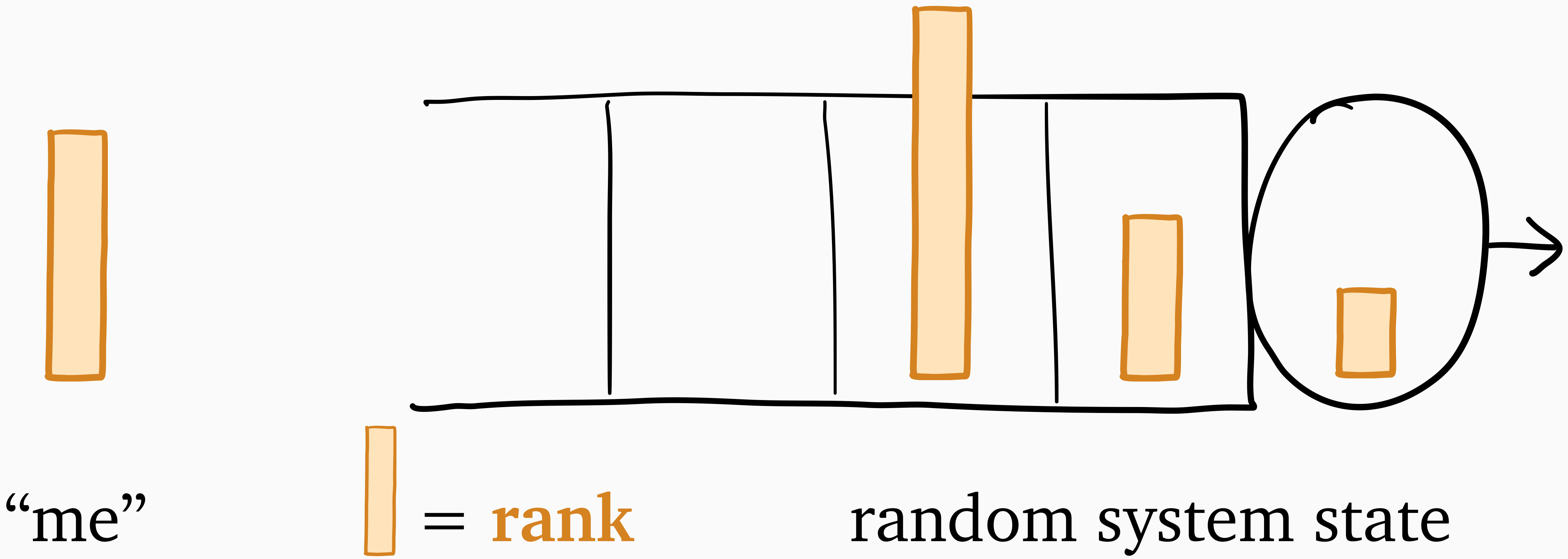


“me” random system state

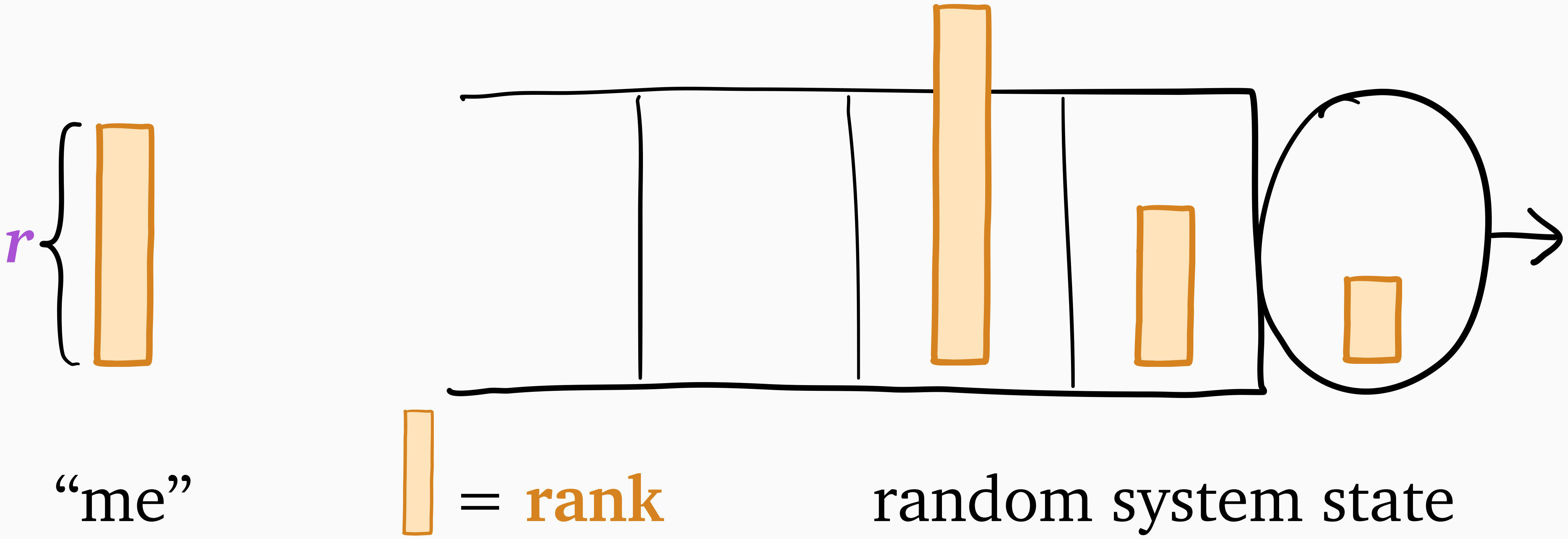
Key quantity: *r*-work



Key quantity: r-work



Key quantity: r-work

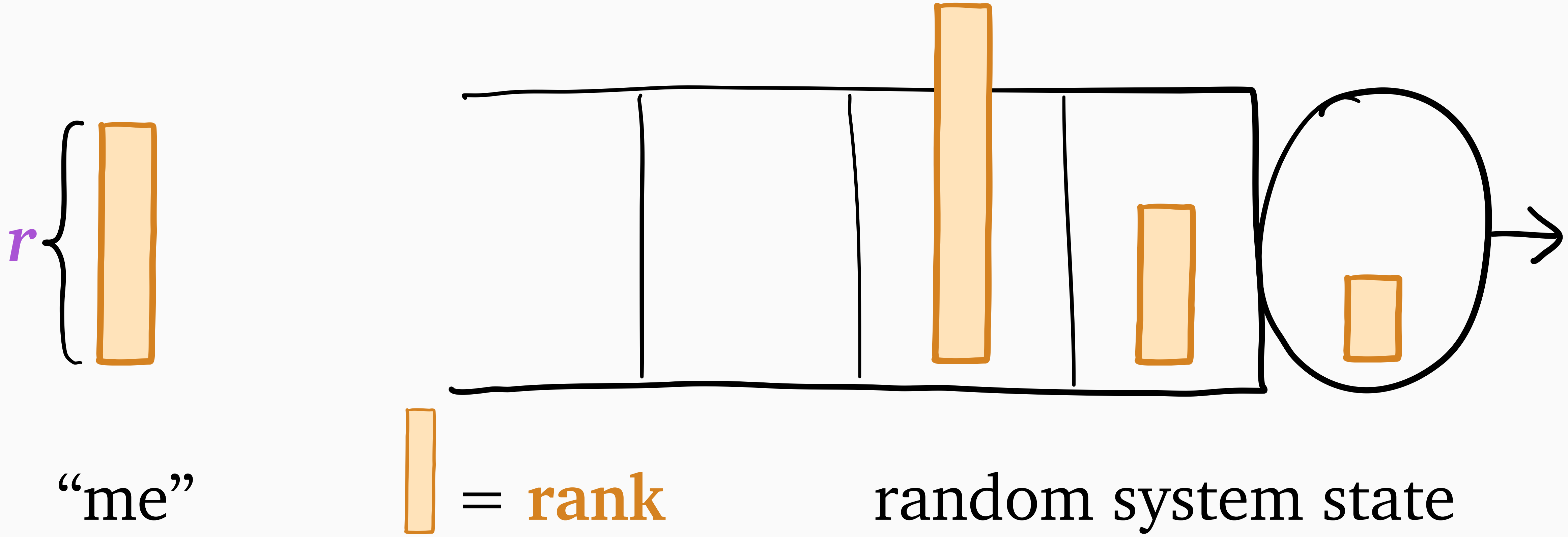


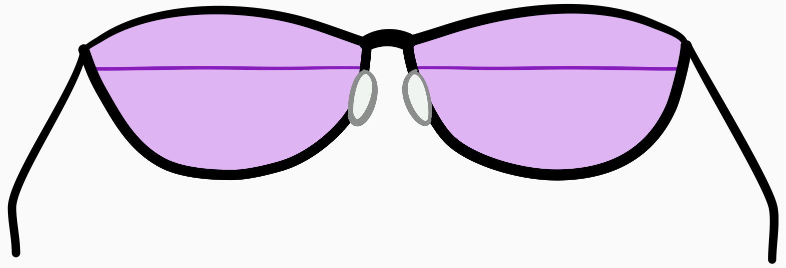
Key quantity:

$W(r)$ = work relevant to job of **rank** r

r-work

Key quantity: r-work



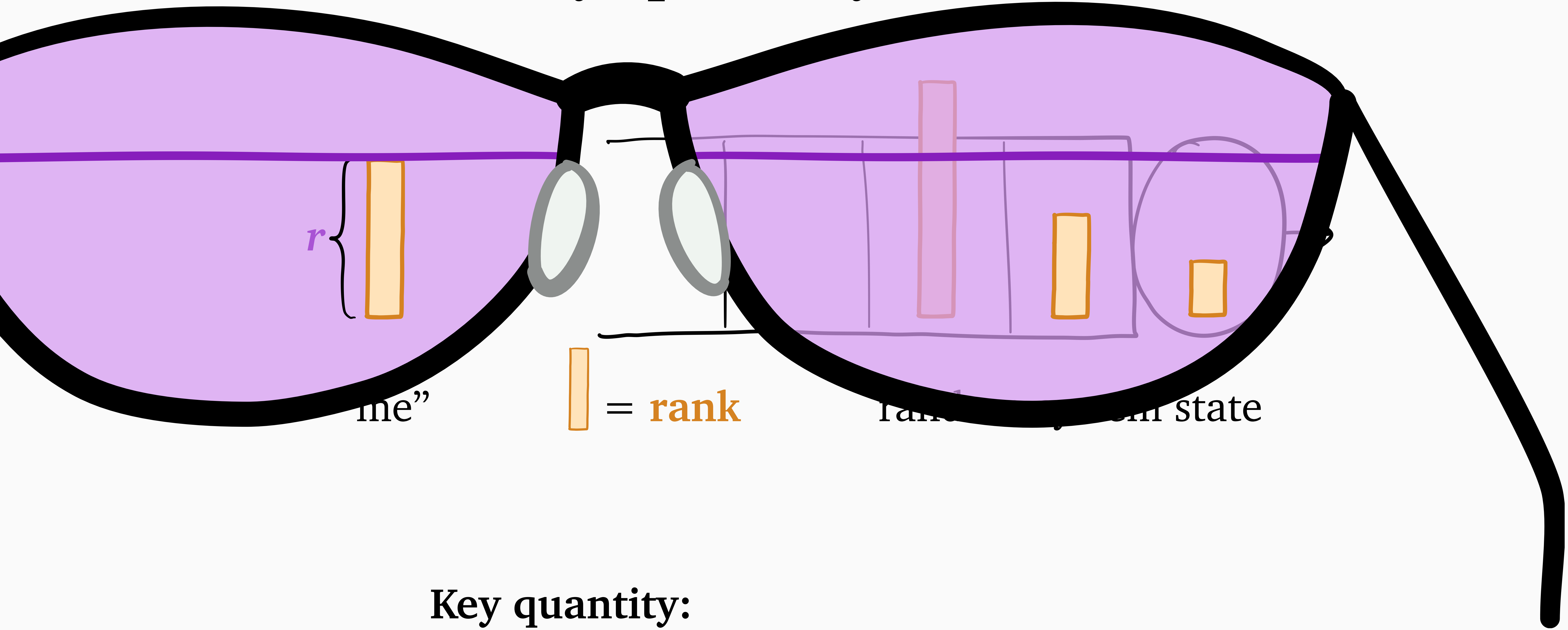


Key quantity:

$W(r)$ = work relevant to job of **rank** r

r-work

Key quantity: r-work

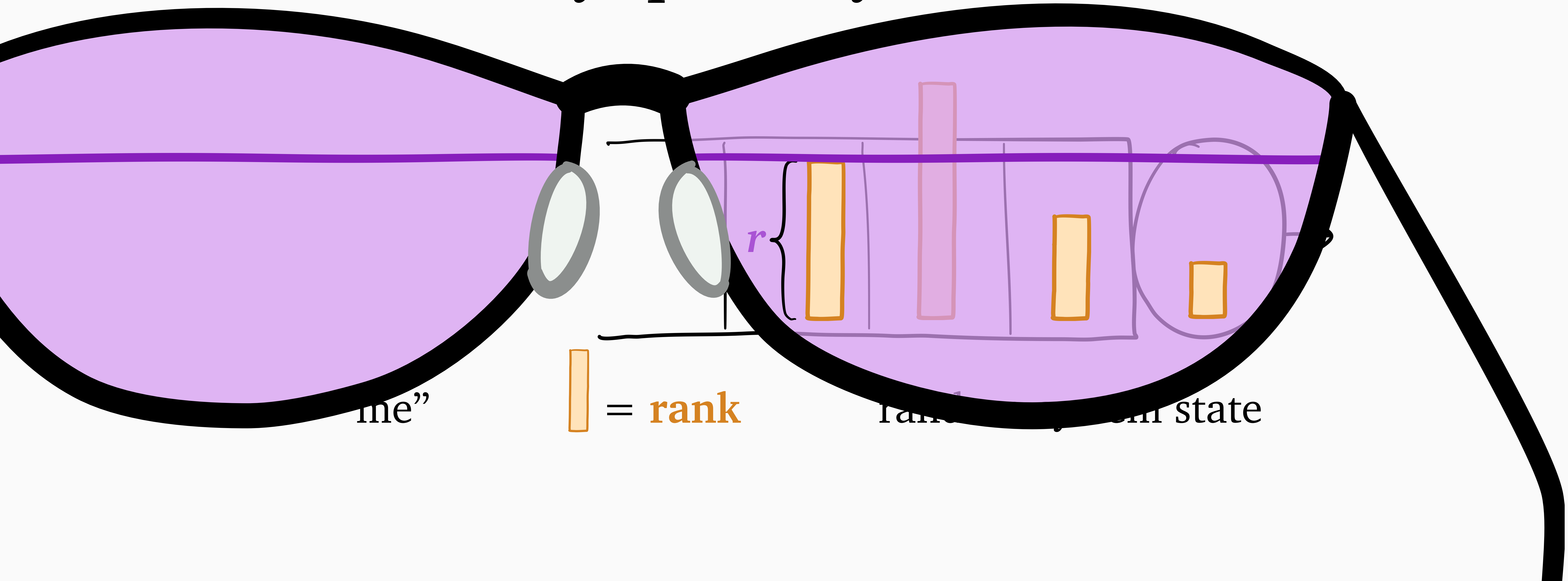


Key quantity:

$W(r)$ = work relevant to job of **rank** r

r-work

Key quantity: r-work

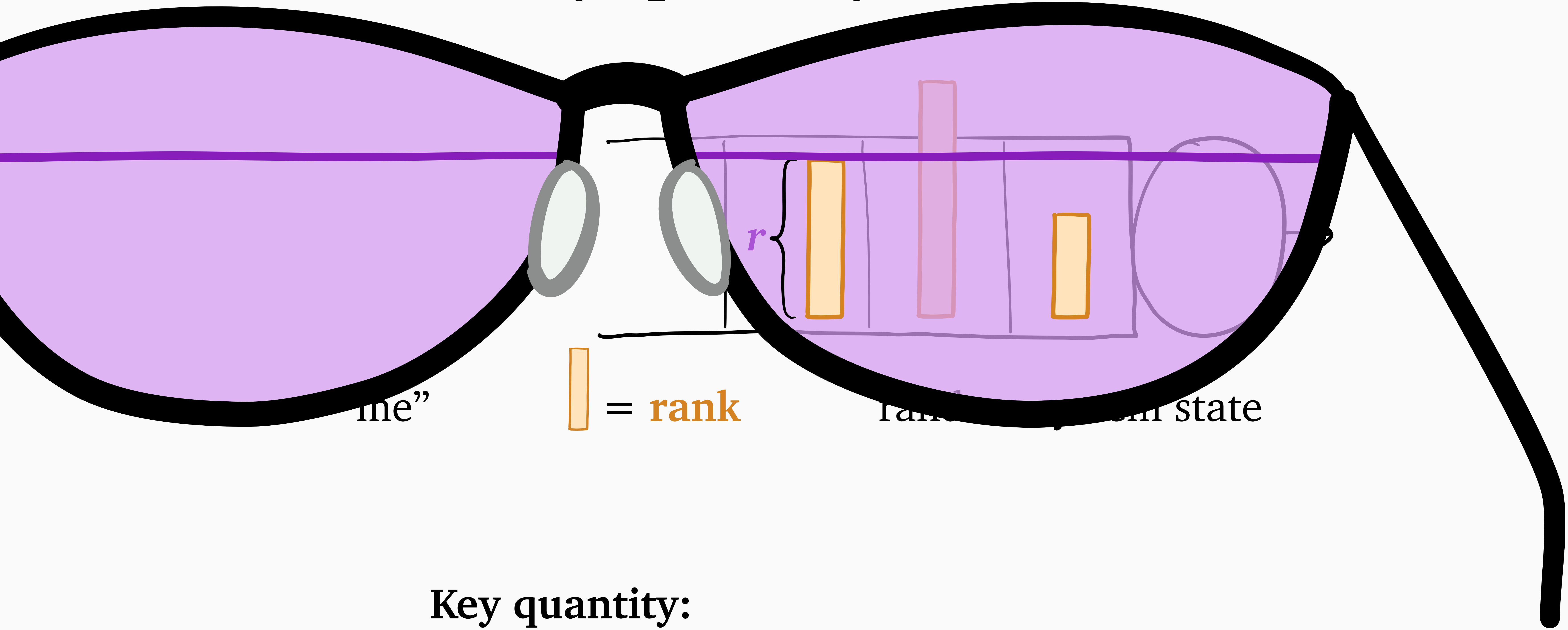


Key quantity:

$W(r)$ = work relevant to job of **rank** r

r-work

Key quantity: r-work



Key quantity:

$W(r)$ = work relevant to job of **rank** r

r-work

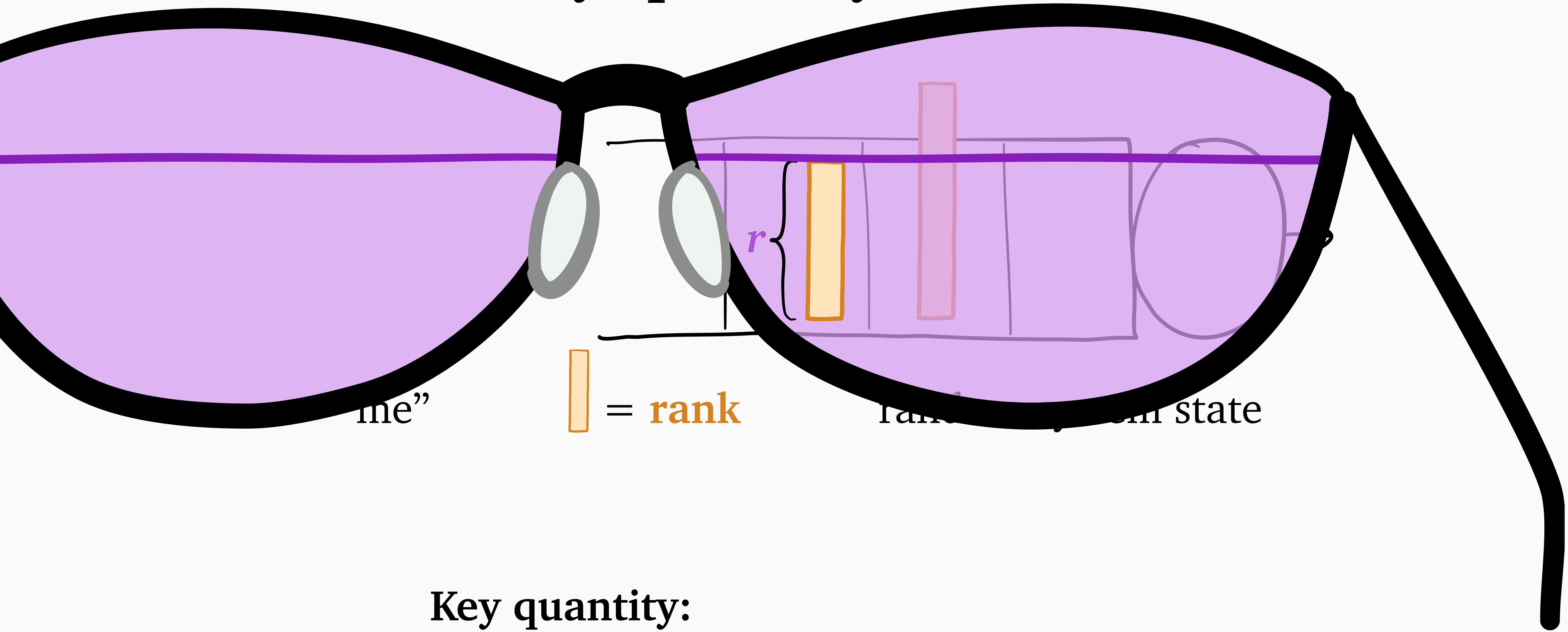
Key quantity: r-work

Key quantity:

$W(r)$ = work relevant to job of **rank** r

r-work

Key quantity: r-work

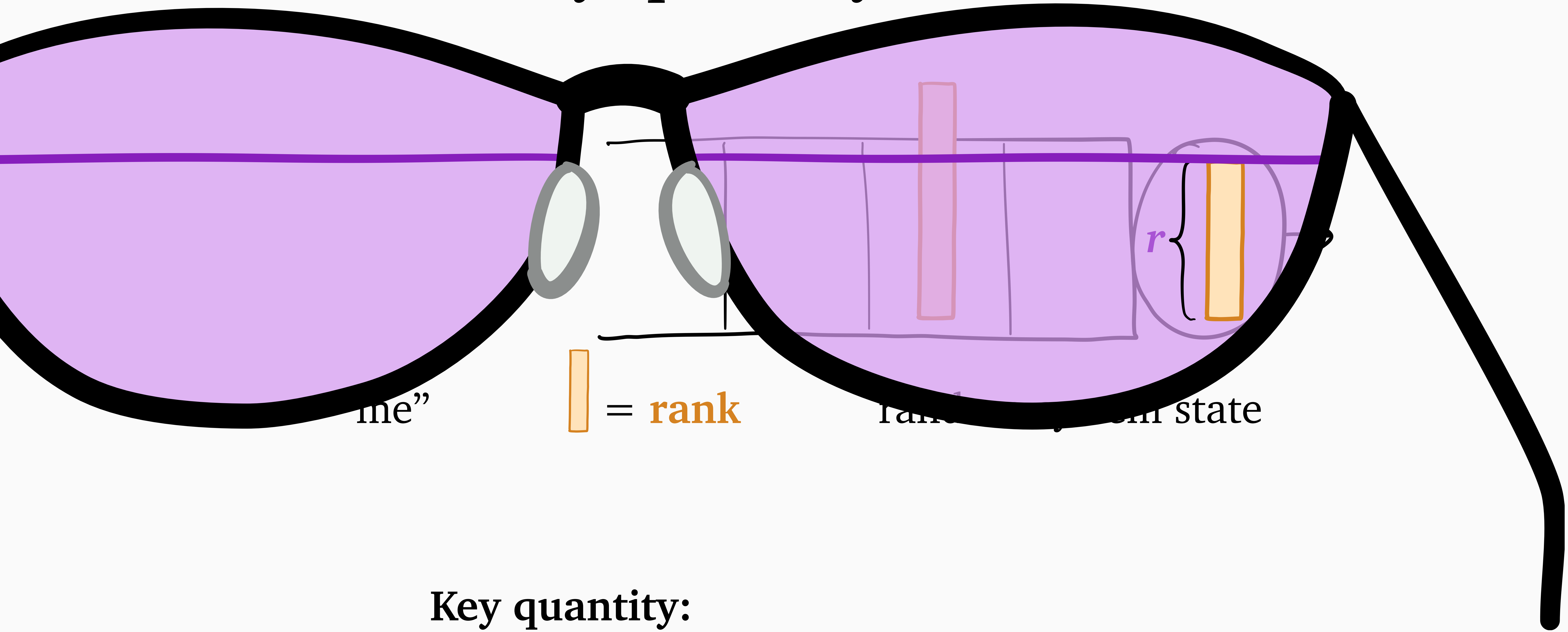


Key quantity:

$W(r)$ = work relevant to job of **rank** r

r-work

Key quantity: r-work



Key quantity:

$W(r)$ = work relevant to job of **rank** r

r-work

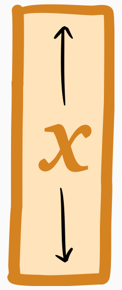
Defining one job's r-work

$W(r)$ = work relevant to **rank** r

Defining one job's r-work

$W(r)$ = work relevant to **rank** r

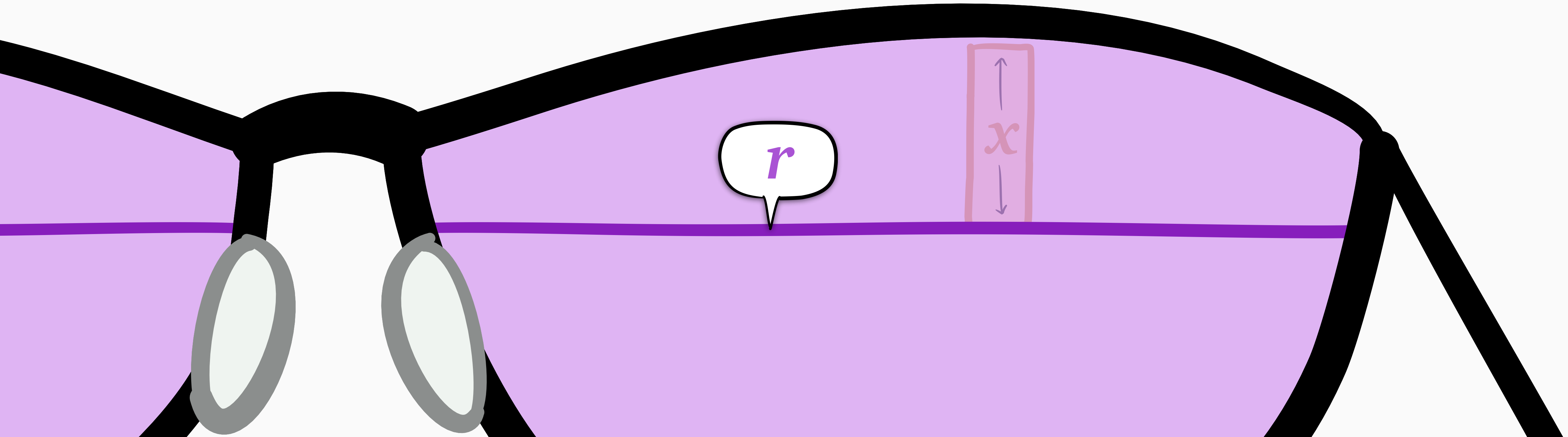
$w_x(r)$ = r-work of *single job* of rem. size x $= \left\{ \vphantom{\begin{matrix} a \\ b \\ c \end{matrix}} \right.$



Defining one job's r-work

$W(r)$ = work relevant to **rank** r

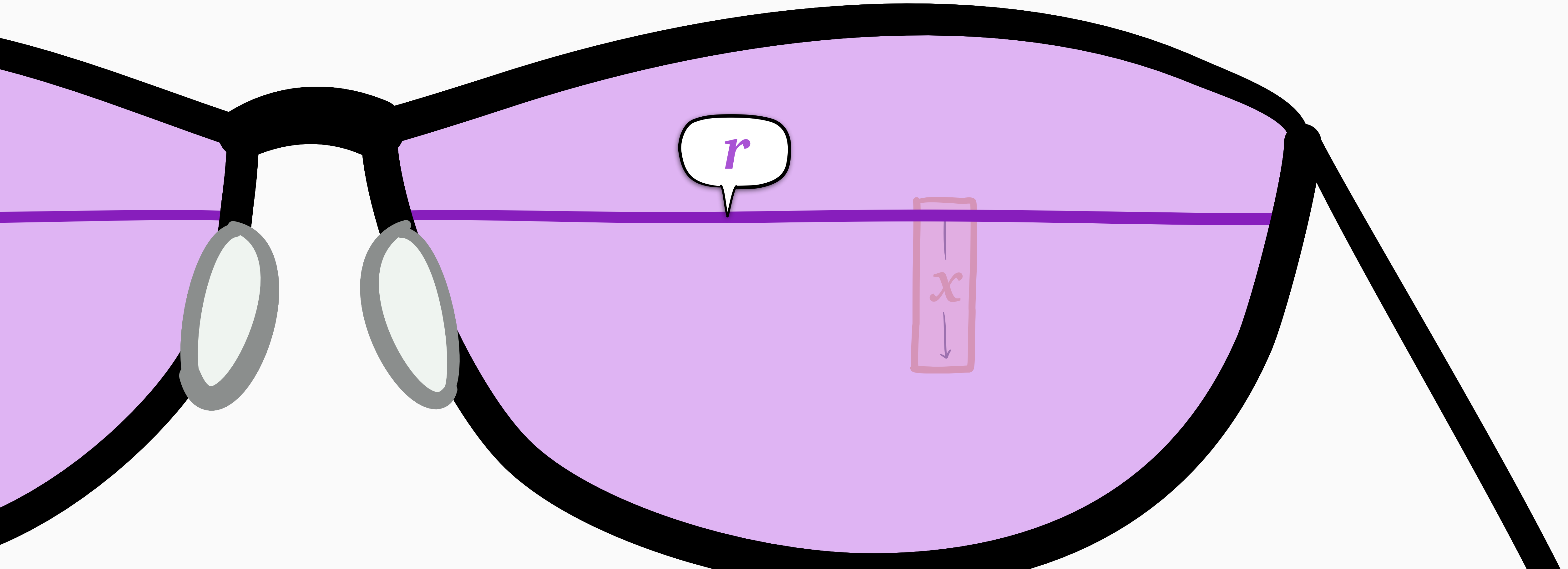
$w_x(r)$ = r-work of *single job* of rem. size x = $\Bigg\{$



Defining one job's r-work

$W(r)$ = work relevant to **rank** r

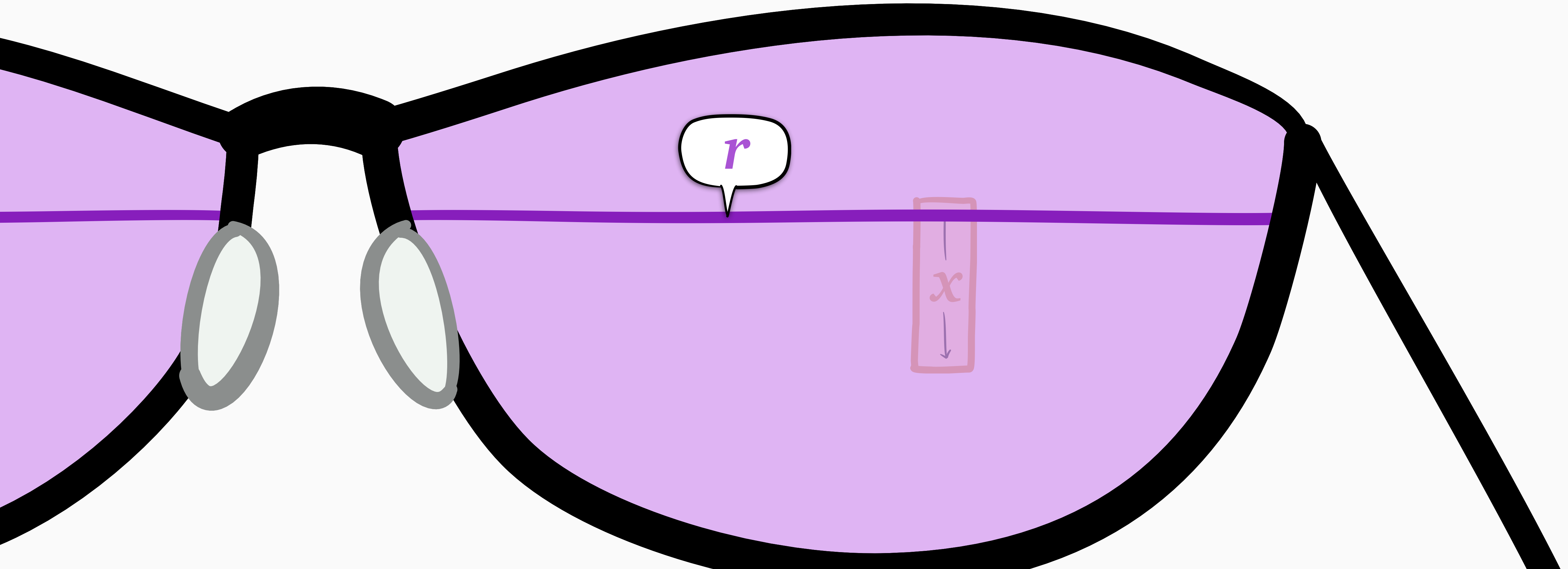
$w_x(r)$ = r-work of *single job* of rem. size x = $\Bigg\{$



Defining one job's r-work

$W(r)$ = work relevant to **rank** r

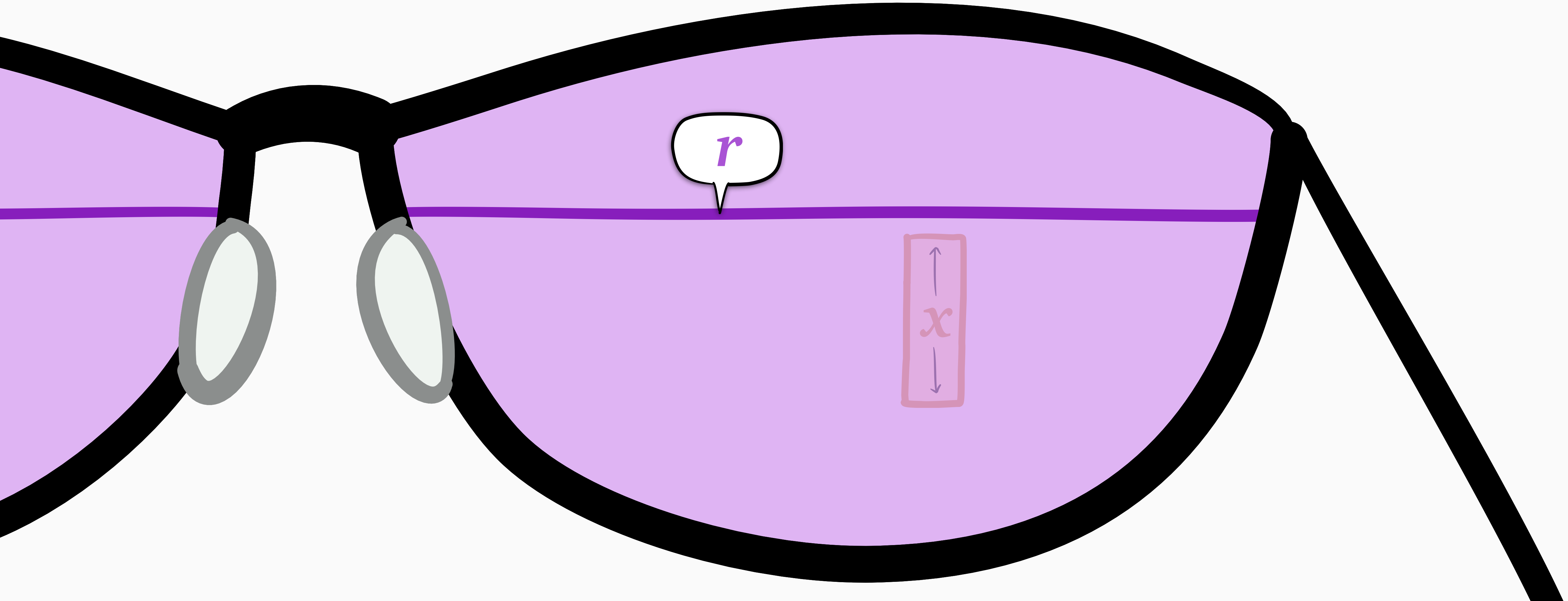
$w_x(r)$ = r-work of *single job* of rem. size x = $\begin{cases} 0 & \text{if } r < x \\ & \end{cases}$



Defining one job's r-work

$W(r)$ = work relevant to **rank** r

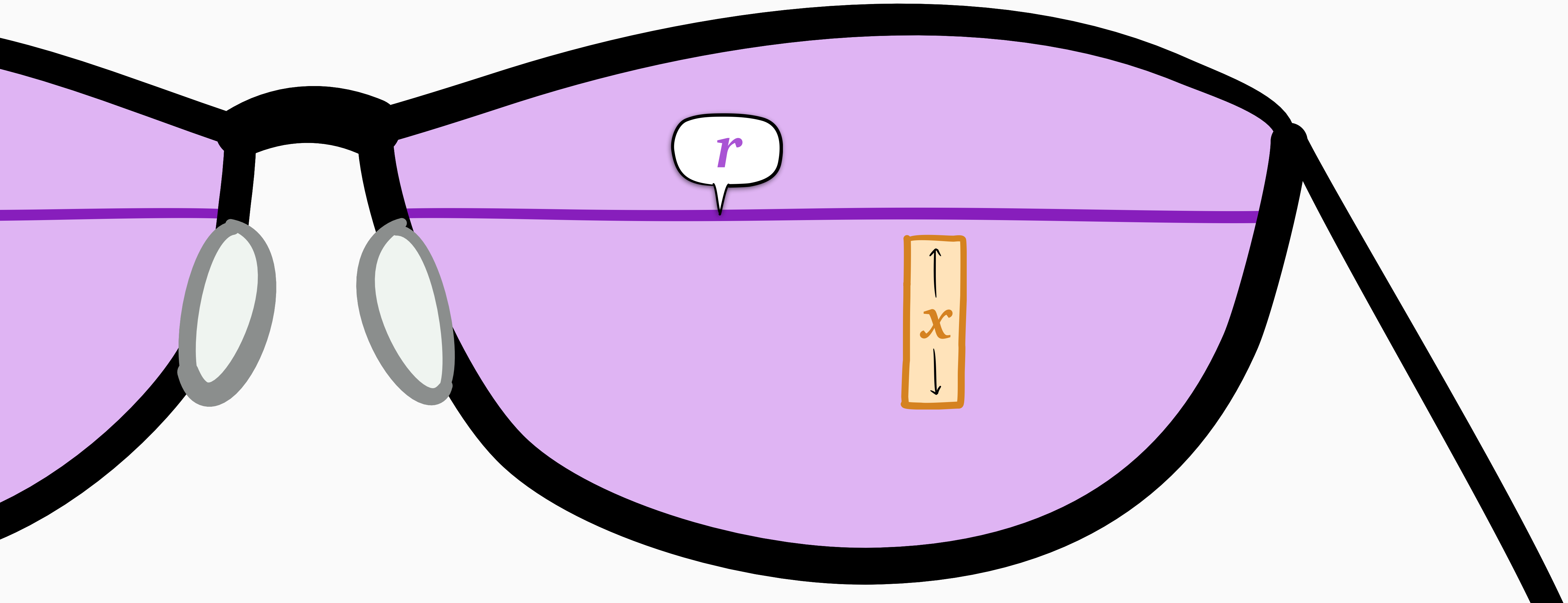
$w_x(r)$ = r-work of *single job* of rem. size x = $\begin{cases} 0 & \text{if } r < x \\ & \end{cases}$



Defining one job's r-work

$W(r)$ = work relevant to **rank** r

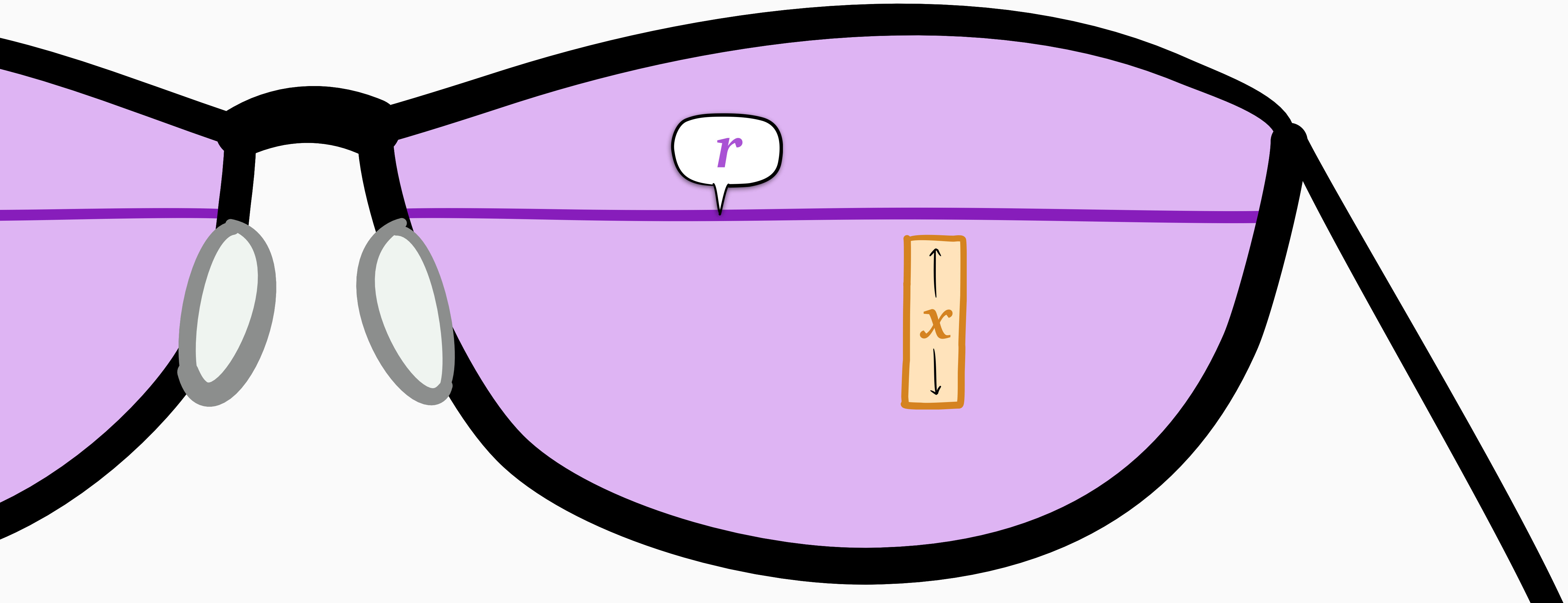
$w_x(r)$ = r-work of *single job* of rem. size x = $\begin{cases} 0 & \text{if } r < x \\ & \end{cases}$



Defining one job's r-work

$W(r)$ = work relevant to **rank** r

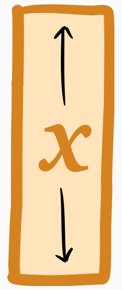
$$w_x(r) = r\text{-work of } \textit{single job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$



Defining one job's r-work

$W(r)$ = work relevant to **rank r**

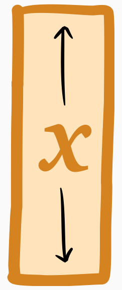
$$w_x(r) = r\text{-work of } \textit{single job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$



Defining one job's r-work

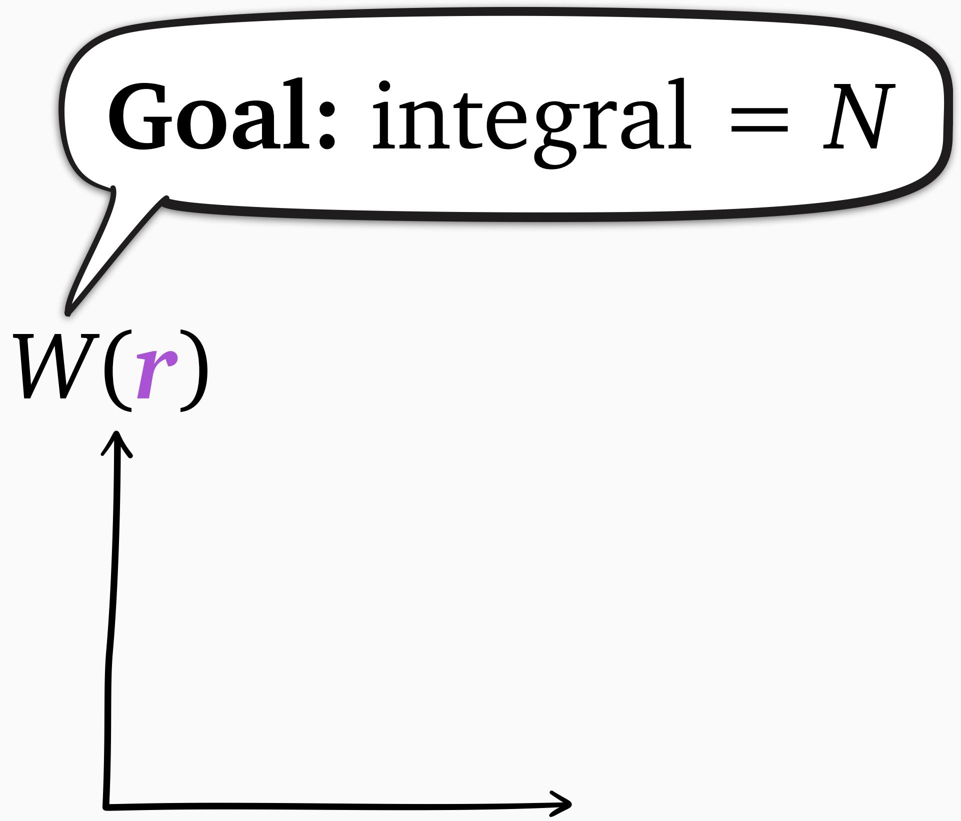
$$\begin{aligned} W(r) &= \text{work relevant to } \textbf{rank } r \\ &= \text{total } r\text{-work of all jobs} \end{aligned}$$

$$w_x(r) = r\text{-work of } \textit{single job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$

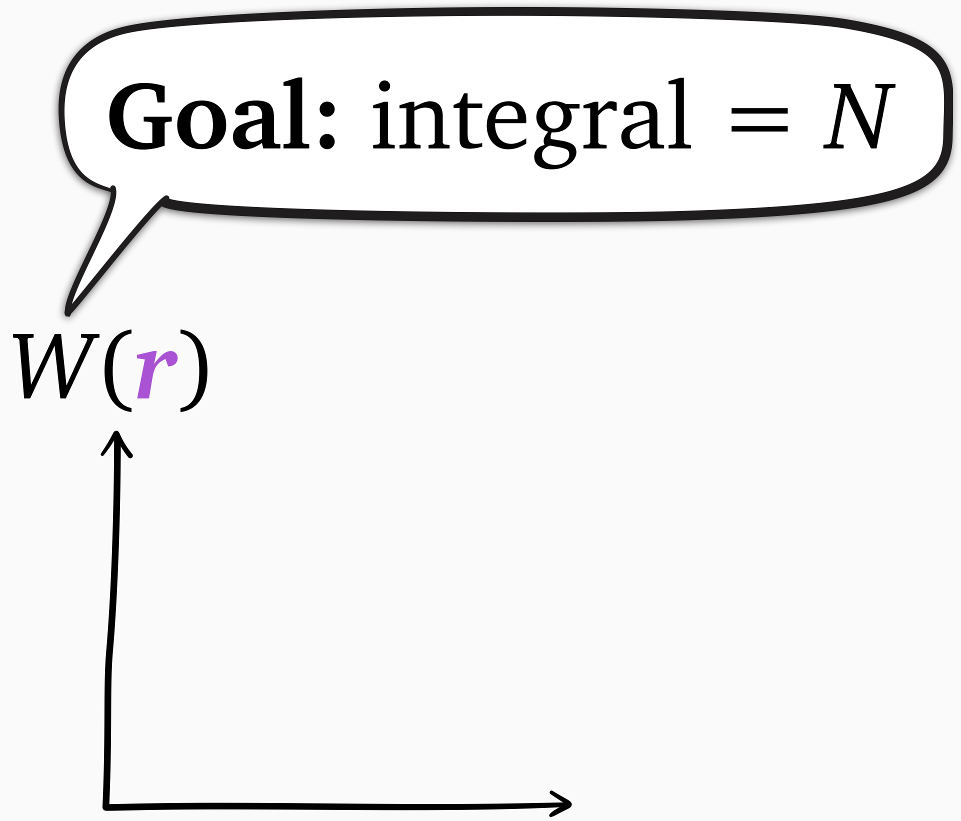


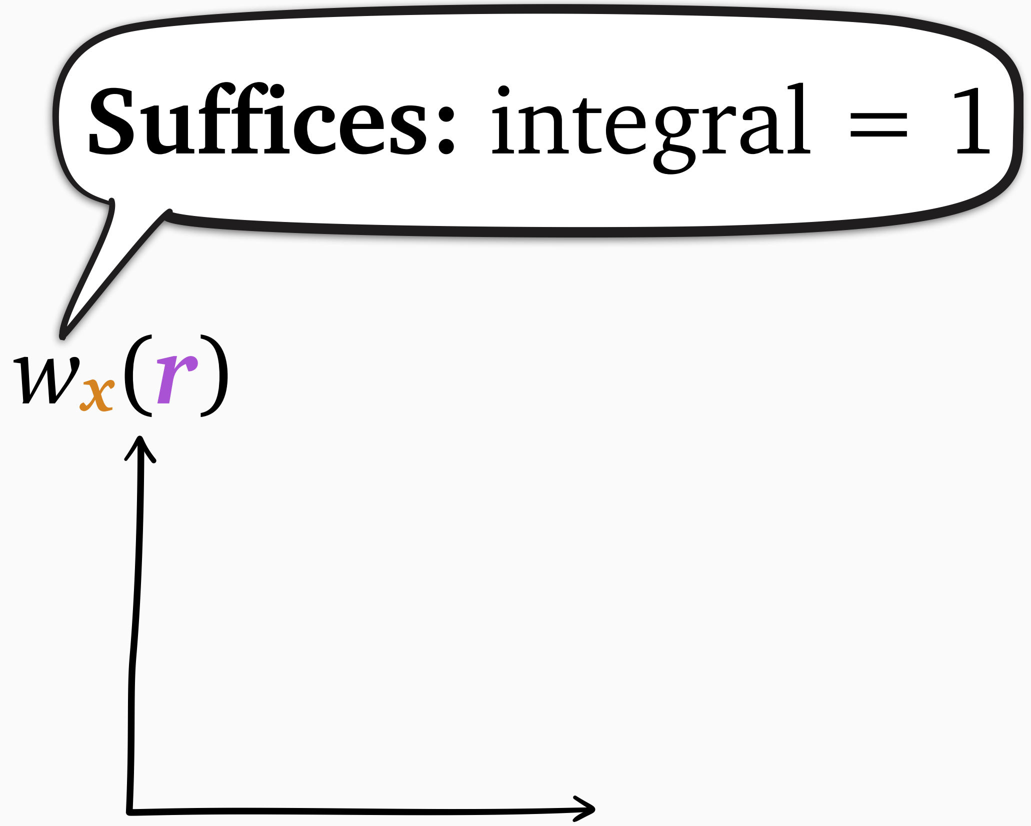
From r-work to number of jobs

From r-work to number of jobs



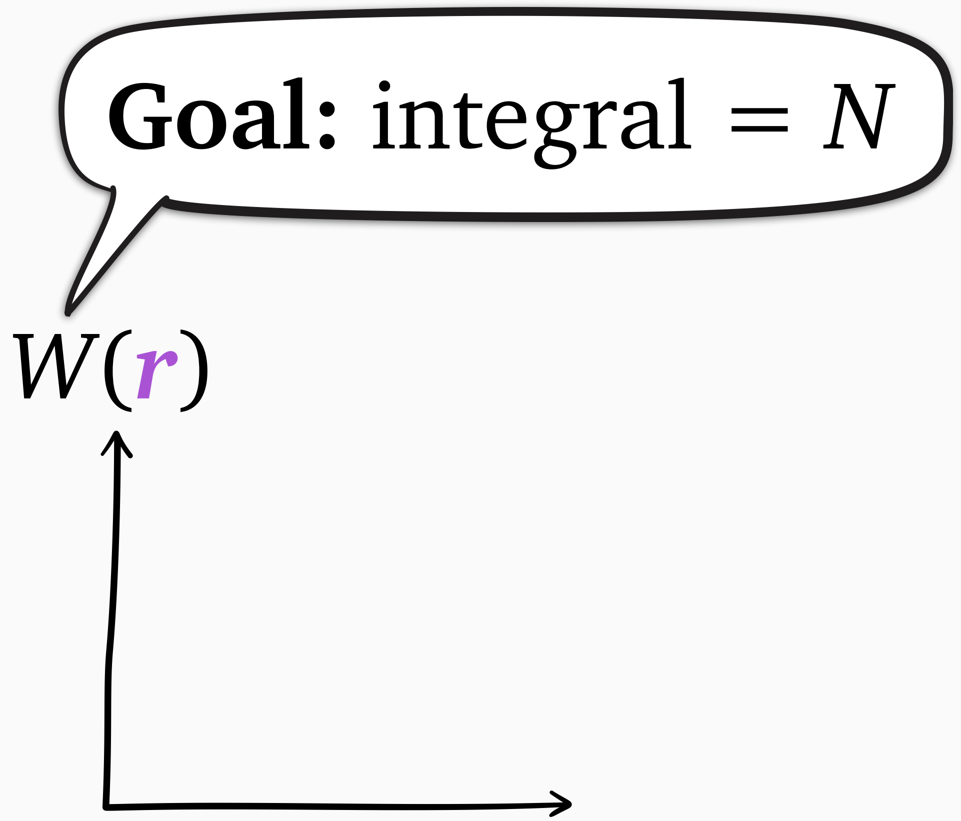
From r-work to number of jobs

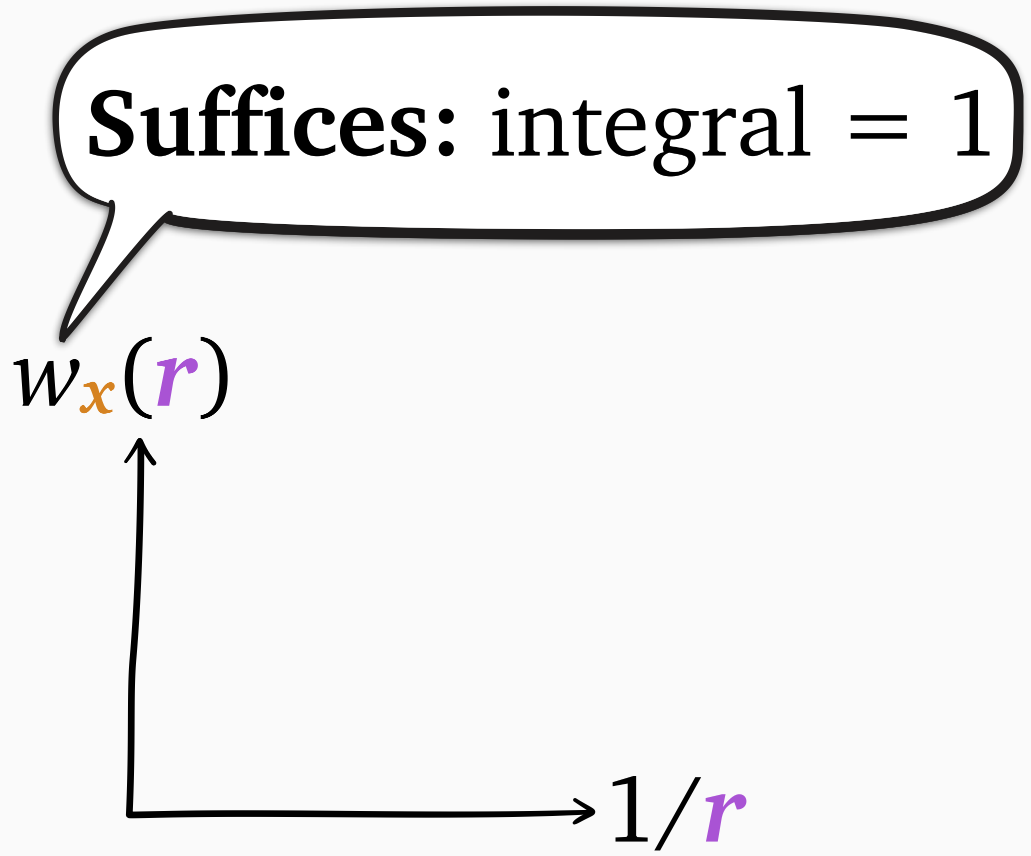




$$w_x(r) = r\text{-work of } \textit{job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$

From r-work to number of jobs

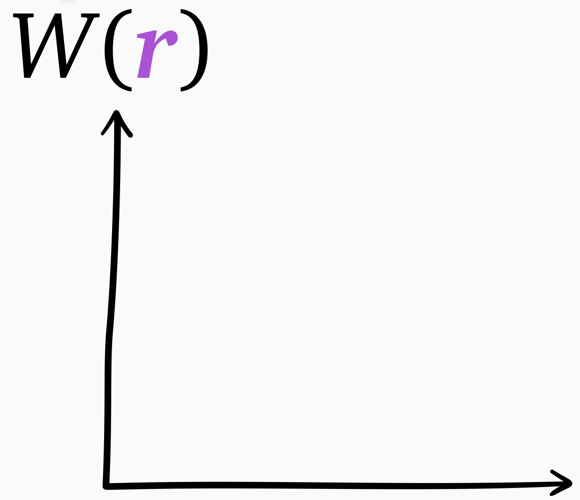




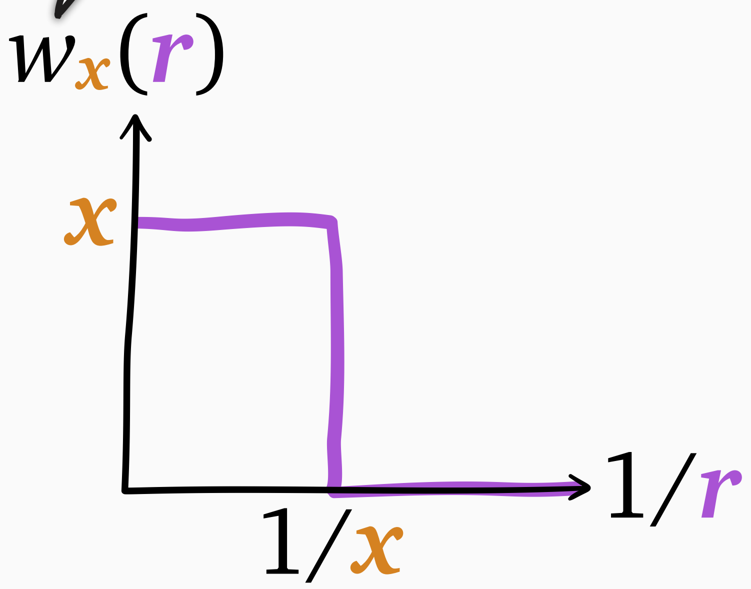
$$w_x(r) = r\text{-work of } \textit{job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$

From r-work to number of jobs

Goal: integral $= N$

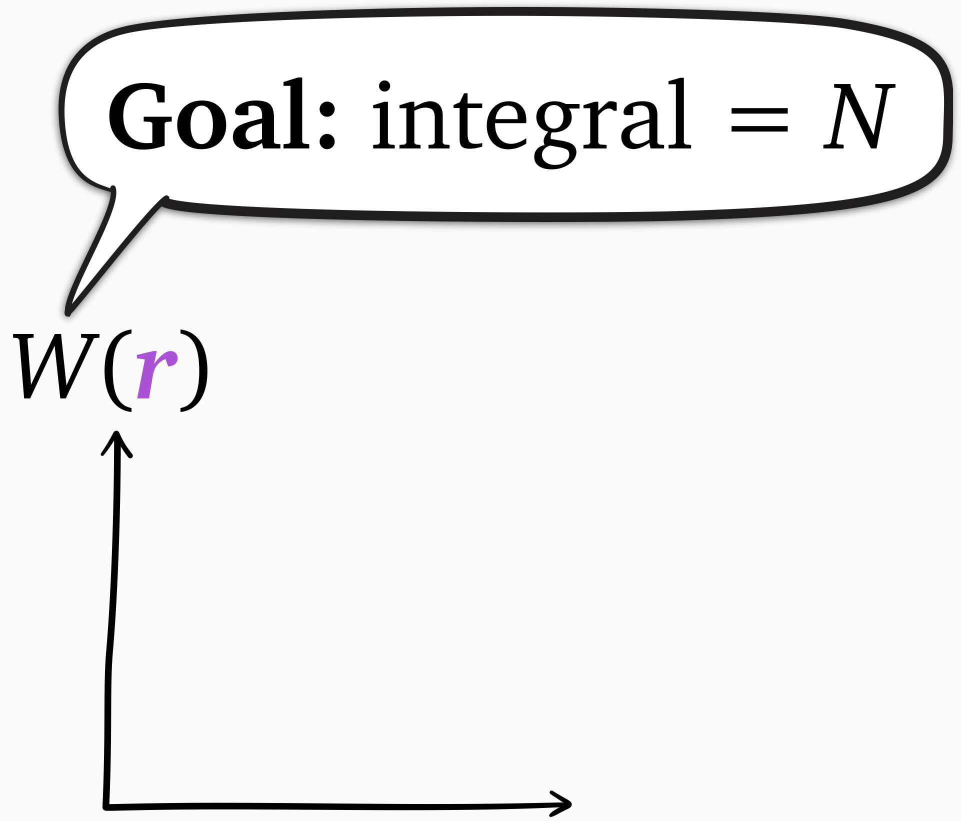


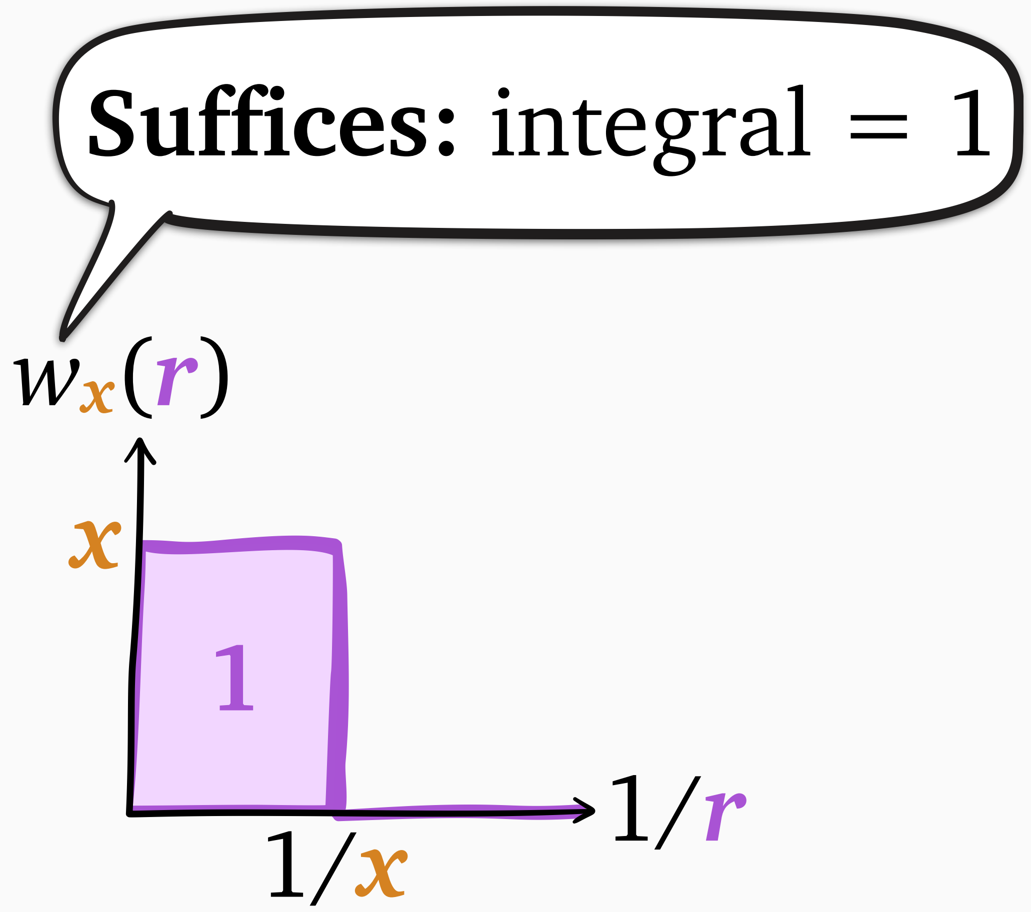
Suffices: integral $= 1$



$$w_x(r) = r\text{-work of } \textit{job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$

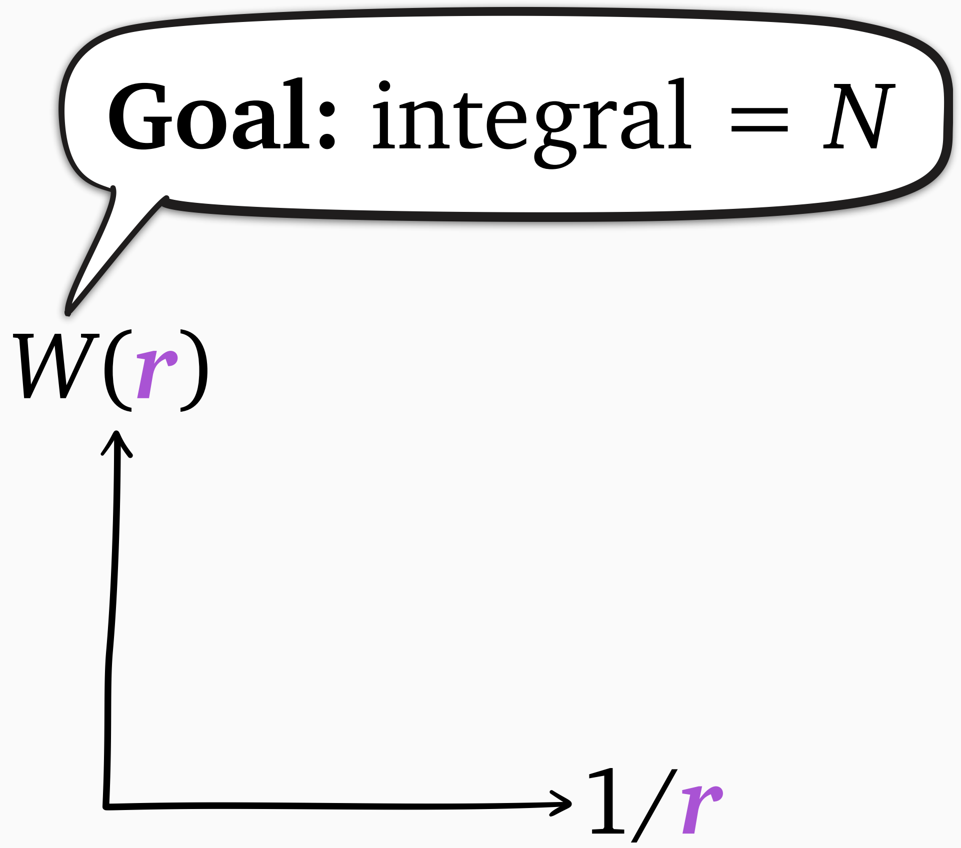
From r-work to number of jobs

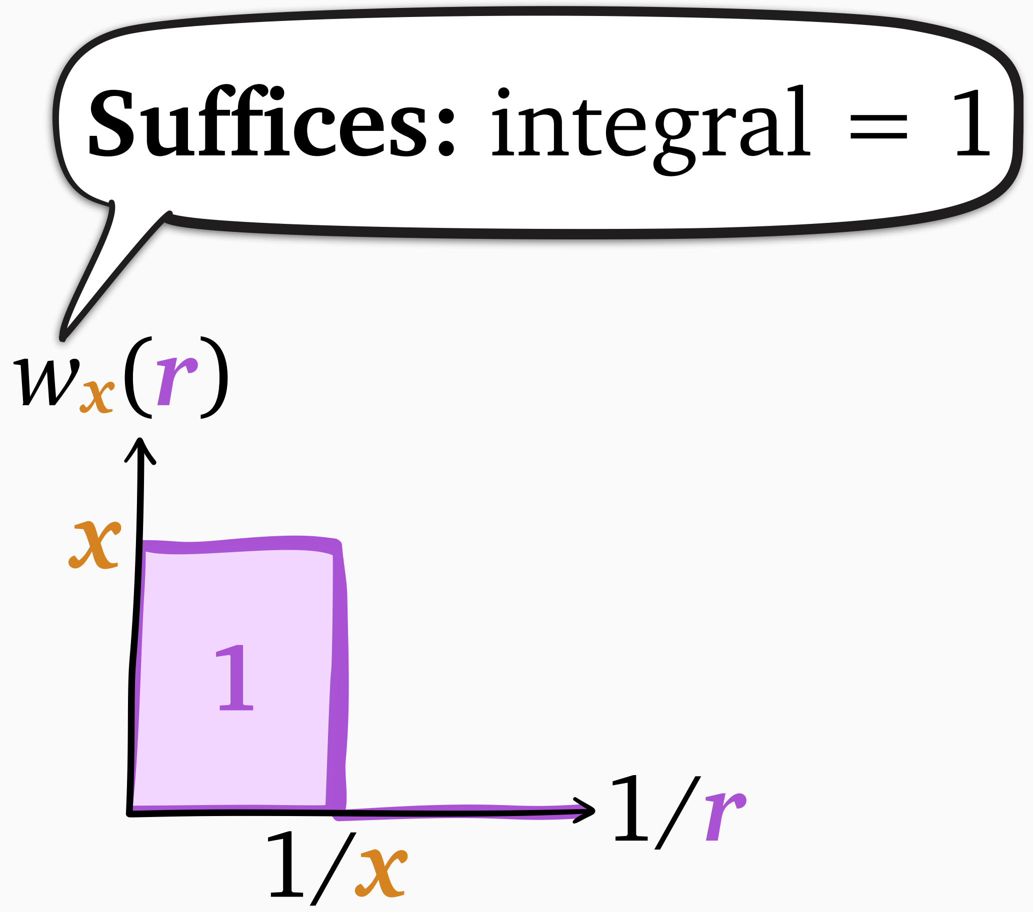




$$w_x(r) = r\text{-work of } \textit{job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$

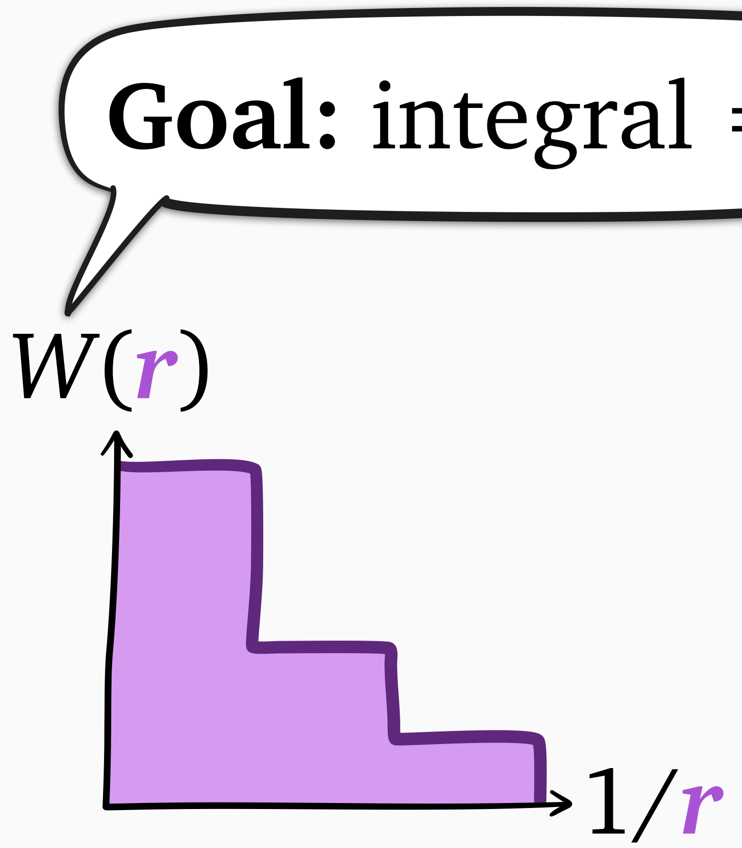
From r-work to number of jobs

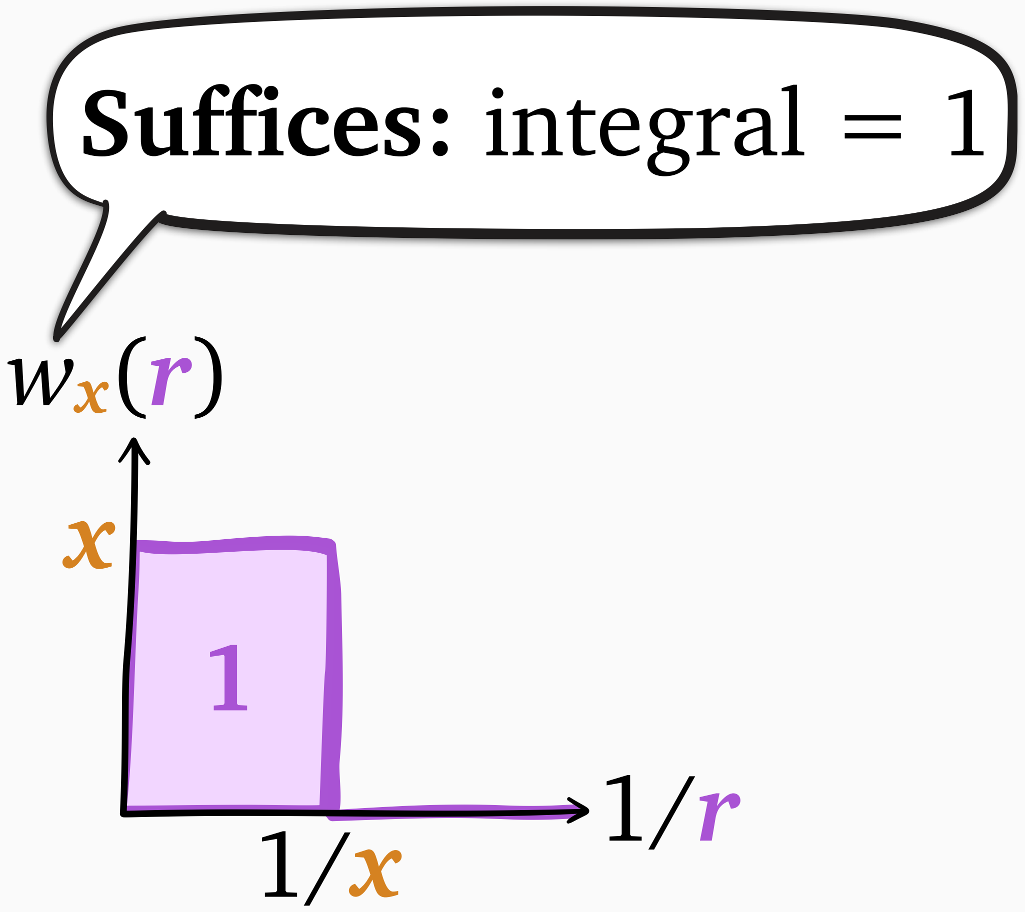




$$w_x(r) = r\text{-work of } \textit{job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$

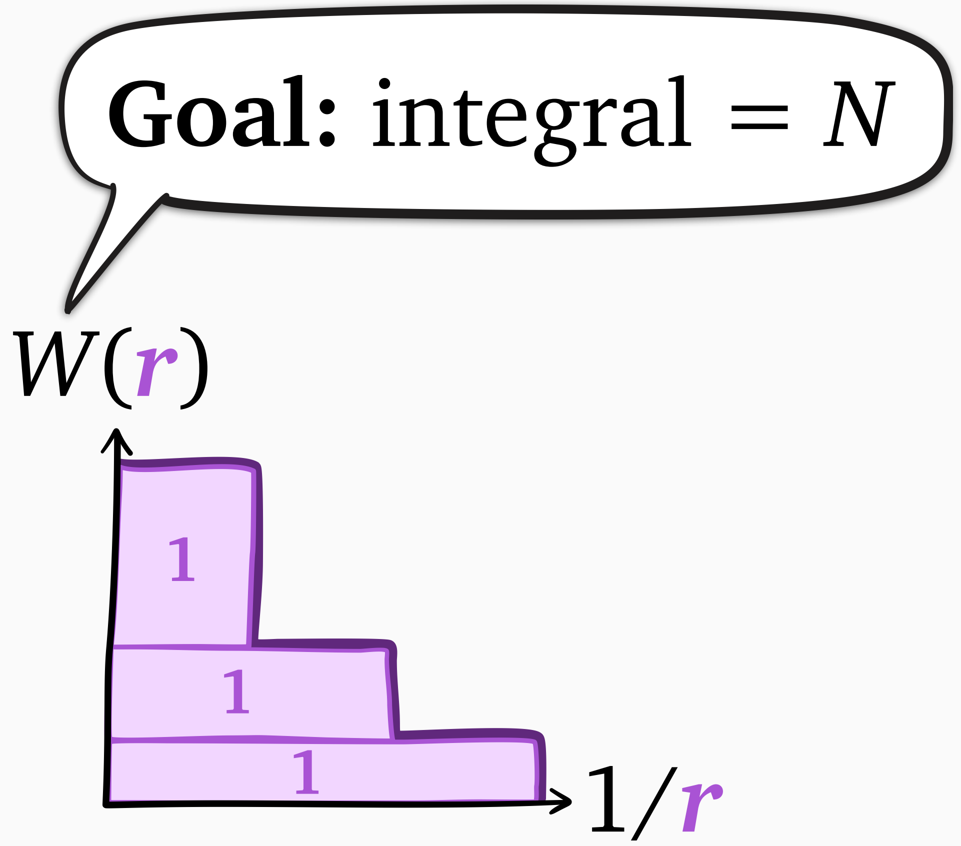
From r-work to number of jobs

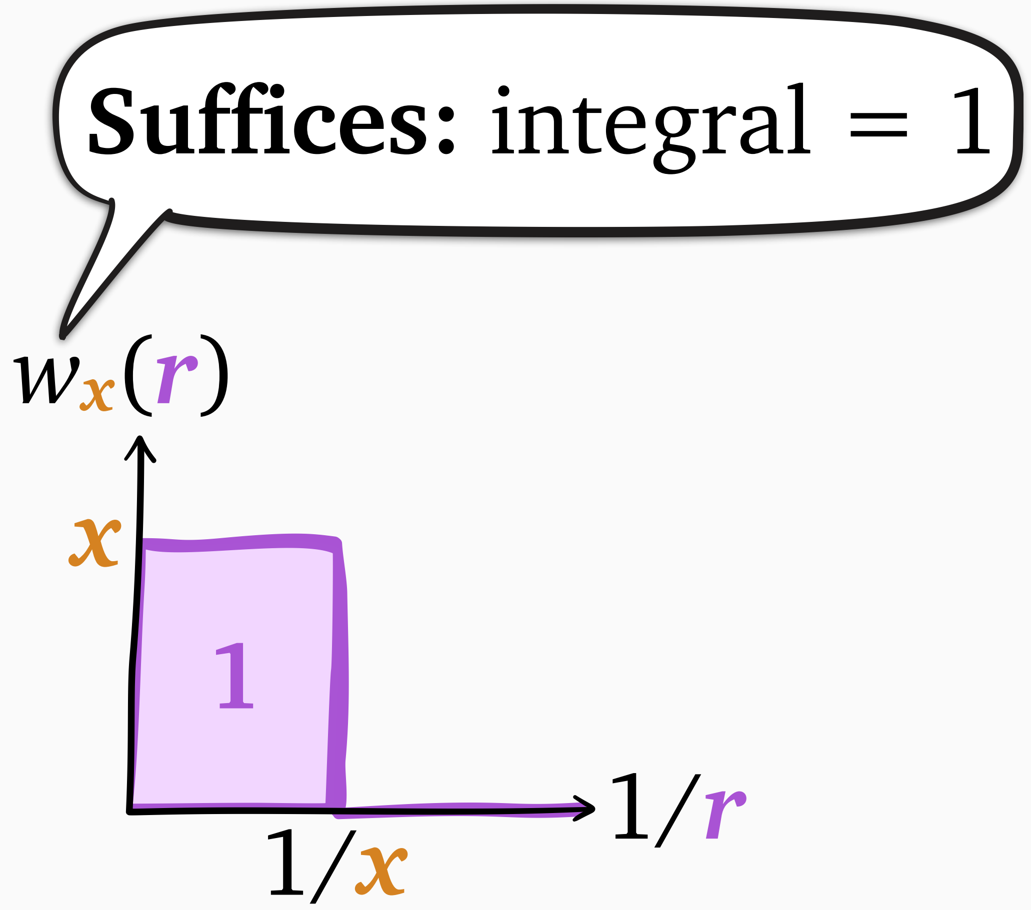




$$w_x(r) = r\text{-work of } \textit{job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$

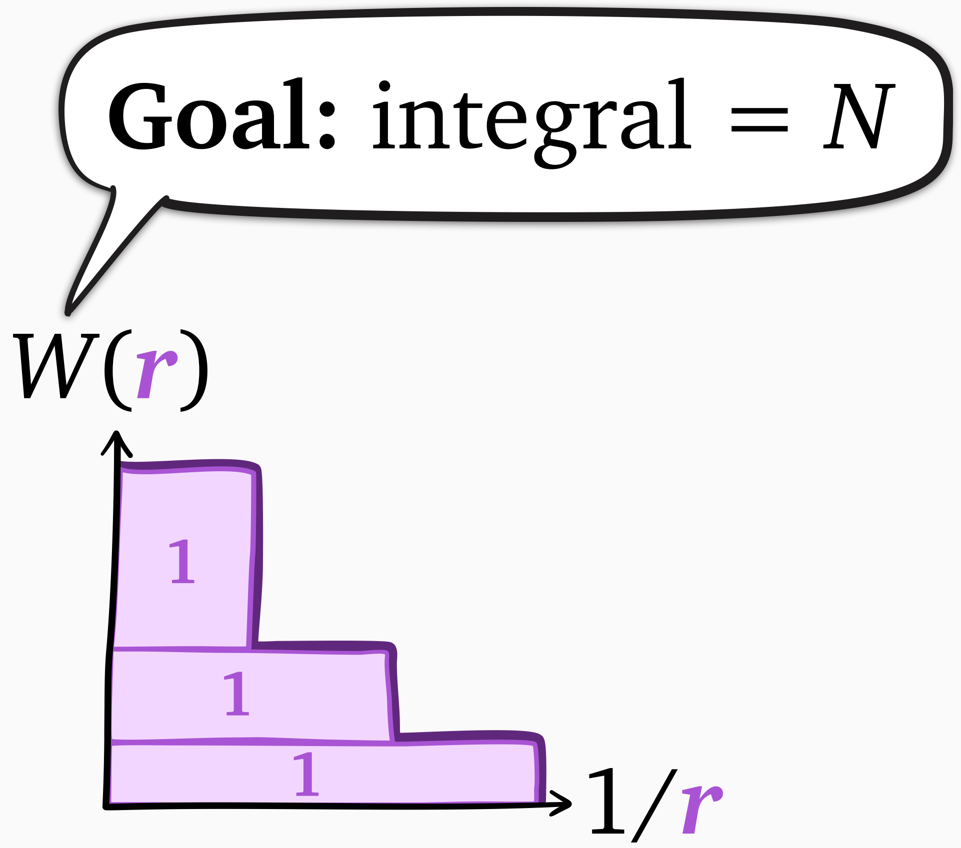
From r-work to number of jobs

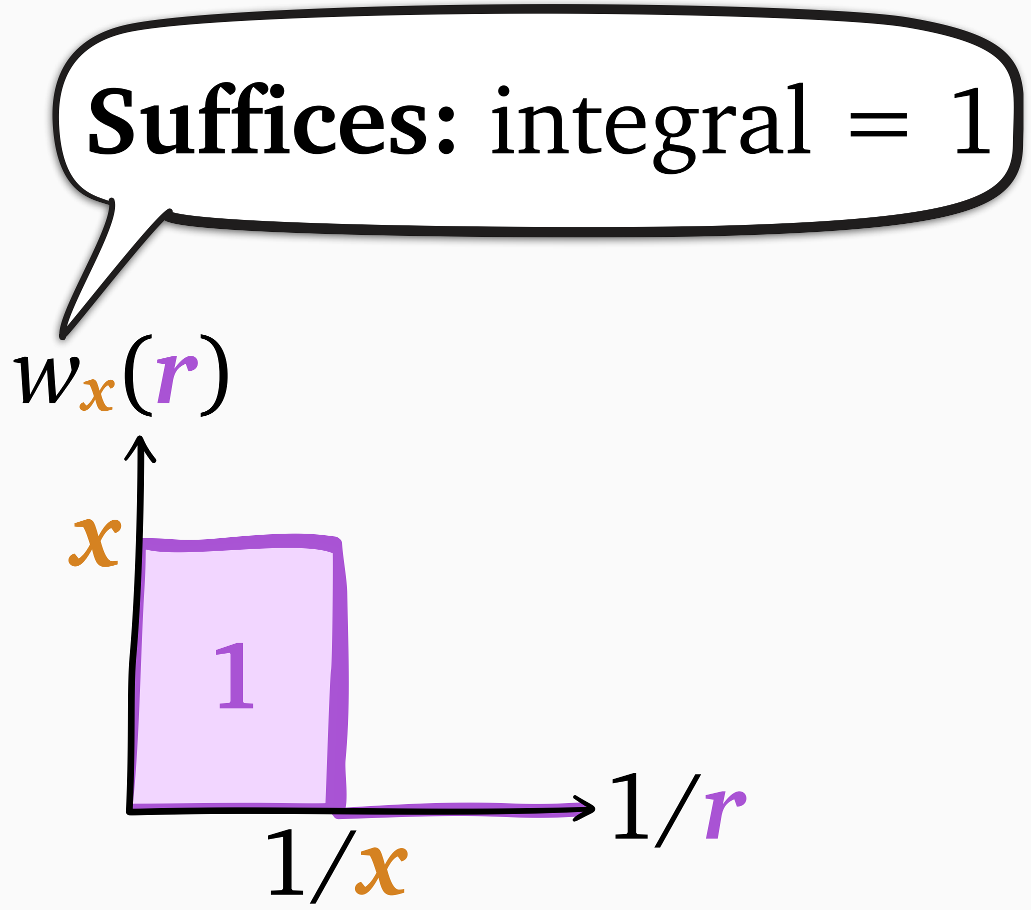




$$w_x(r) = r\text{-work of } \textit{job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$

From r-work to number of jobs



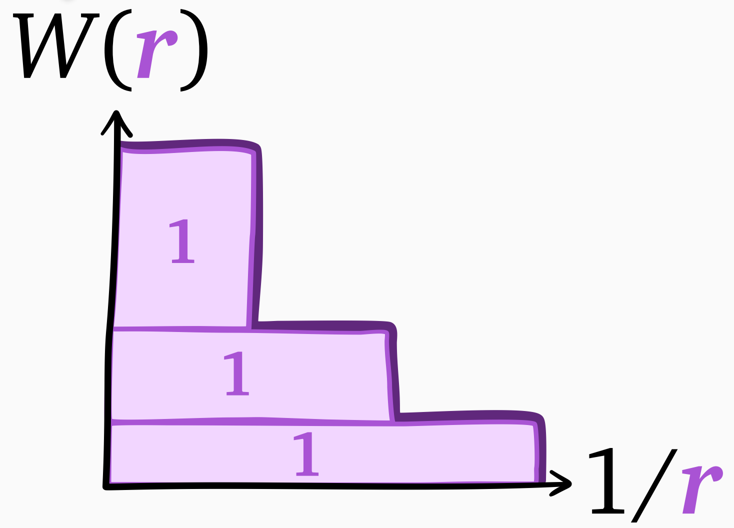


Theorem:

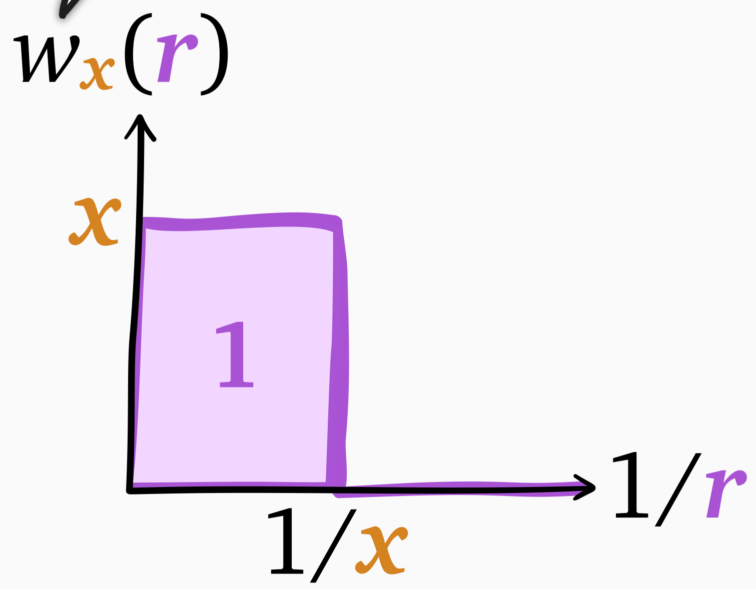
$$N = \int_0^\infty \frac{W(r)}{r^2}\, \mathrm{d}r$$

From r-work to number of jobs

Goal: integral $= N$



Suffices: integral $= 1$



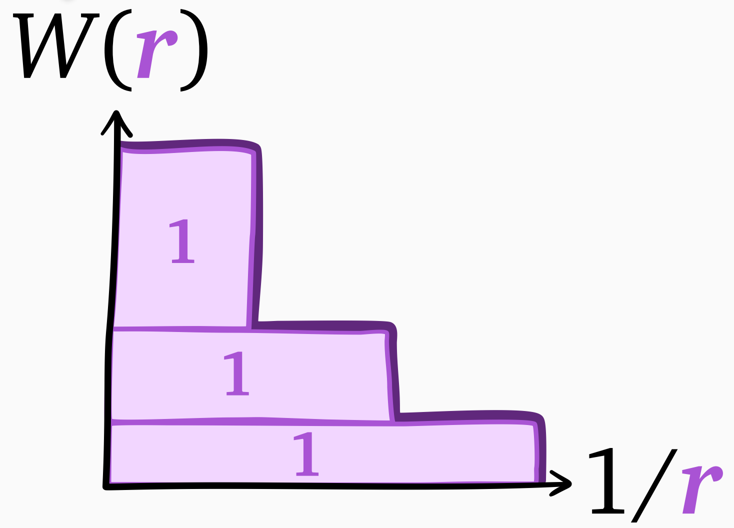
WINE

Theorem:

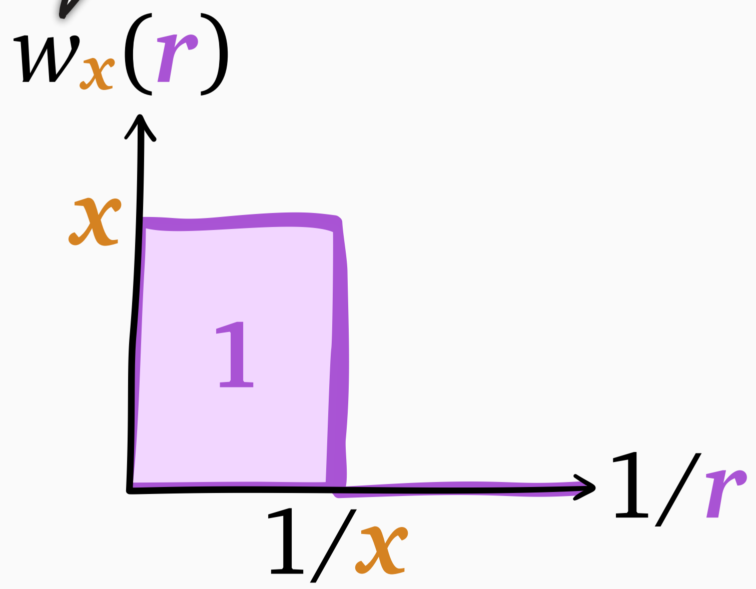
$$N = \int_0^\infty \frac{W(r)}{r^2}\,\mathrm{d}r$$

From r-work to number of jobs

Goal: integral $= N$



Suffices: integral $= 1$



Theorem:

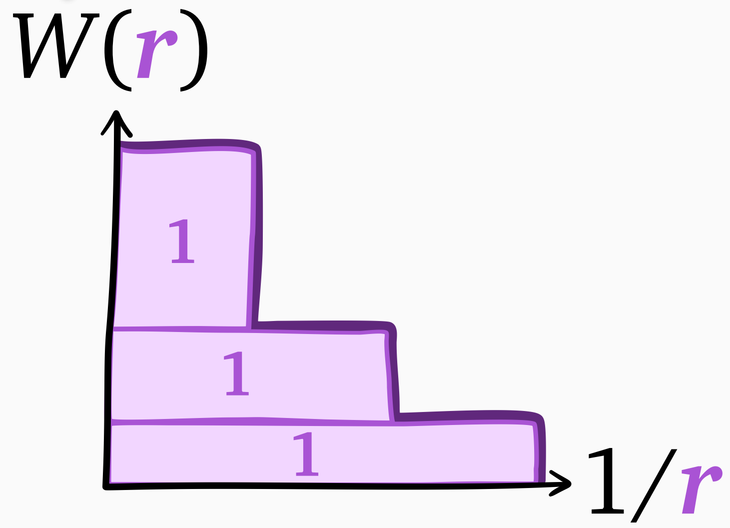
WINE

$$N = \int_0^\infty \frac{W(r)}{r^2}\,\mathrm{d}r$$

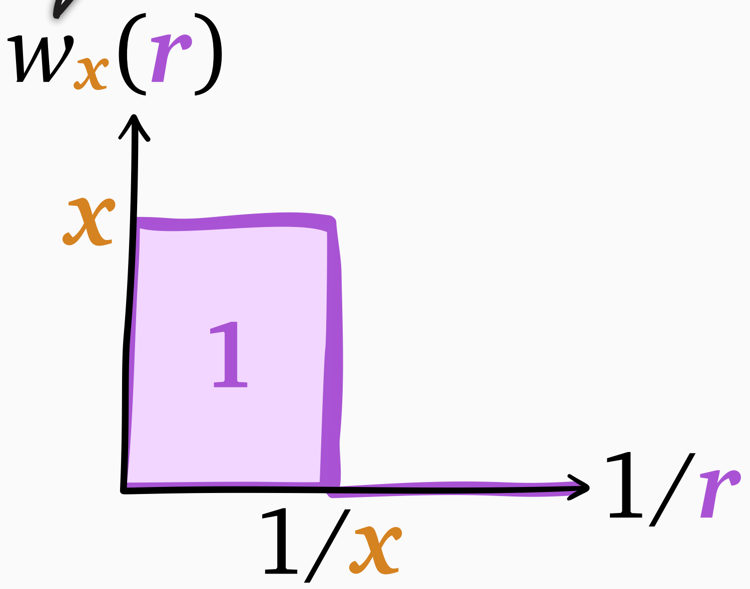
uses **rank** = rem. size

From r-work to number of jobs

Goal: integral $= N$



Suffices: integral $= 1$



WINE

Theorem: under *any* policy,

$$N = \int_0^\infty \frac{W(r)}{r^2}\,\mathrm{d}r$$

uses **rank** = rem. size

How does **WINE** help?

WINE

Theorem:

$$N = \int_0^\infty \frac{W(r)}{r^2}\, \mathrm{d}r$$

How does WINE help?

WINE

Theorem:

$$N = \int_0^\infty \frac{W(r)}{r^2}\,\mathrm{d}r$$

How to minimize $W(r)$?

How does WINE help?

WINE

Theorem:

$$N = \int_0^\infty \frac{W(r)}{r^2}\,\mathrm{d}r$$

How to minimize $W(r)$?

Prioritize jobs with ***rank*** $\leq r$

How does WINE help?

WINE

Theorem:

$$N = \int_0^\infty \frac{W(r)}{r^2}\,\mathrm{d}r$$

How to minimize $W(r)$?

Prioritize jobs with ***rank*** $\leq r$

To do for all r:
always serve job of minimum ***rank***

How does **WINE** help?

Theorem:

WINE

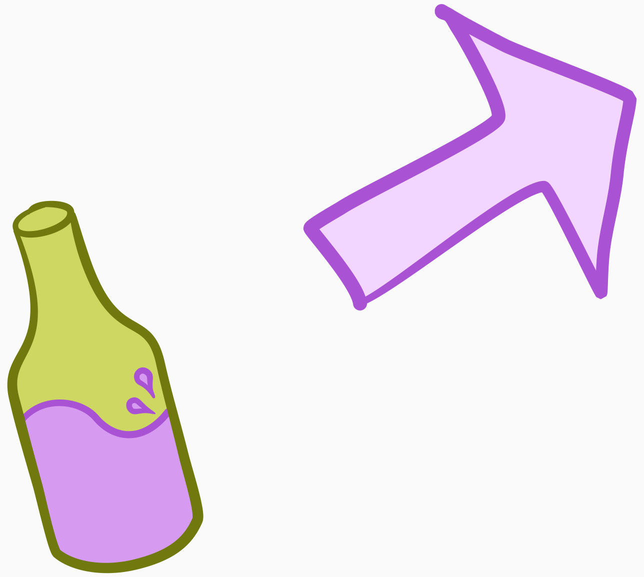
$$N = \int_0^\infty \frac{W(r)}{r^2}\,\mathrm{d}r$$

How to minimize $W(r)$?

Prioritize jobs with ***rank*** $\leq r$

under Poisson arrivals

To do for all r:
always serve job of minimum ***rank***



SRPT-flavored WINE:

$$N = \int_0^\infty \frac{W(r)}{r^2}\,\mathrm{d}r$$

SRPT-flavored WINE:

$$N = \int_0^\infty \frac{W(r)}{r^2}\,\mathrm{d}r$$

Gittins-flavored WINE:

$$\mathbf{E}[N] = \int_0^\infty \frac{\mathbf{E}[W(r)]}{r^2}\,\mathrm{d}r$$

SRPT-flavored WINE:

$$N = \int_0^\infty \frac{W(r)}{r^2}\,\mathrm{d}r$$

Gittins-flavored WINE:

$$\mathbf{E}[N] = \int_0^\infty \frac{\mathbf{E}[W(r)]}{r^2}\,\mathrm{d}r$$

Lemma: using **Gittins**-flavored r-work,

$$1 = \int_0^\infty \frac{\mathbf{E}[\text{one job's } r\text{-work} \mid \text{job's state}]}{r^2}\,\mathrm{d}r$$

SRPT-flavored WINE:

$$N = \int_0^\infty \frac{W(r)}{r^2}\,\mathrm{d}r$$

Gittins-flavored WINE:

$$\mathbf{E}[N] = \int_0^\infty \frac{\mathbf{E}[W(r)]}{r^2}\,\mathrm{d}r$$

Related to *achievable region method*
[Bertsimas & Niño-Mora]

Lemma: using Gittins-flavored r-work,

$$1 = \int_0^\infty \frac{\mathbf{E}[\text{one job's } r\text{-work} \mid \text{job's state}]}{r^2}\,\mathrm{d}r$$

Robustness of Gittins

$$\mathbf{E}[N] = \int_0^\infty \frac{\mathbf{E}[W(r)]}{r^2}\,\mathrm{d}r$$

Robustness of Gittins

$$\mathbf{E}[N] = \int_0^\infty \frac{\mathbf{E}[W(r)]}{r^2}\,\mathrm{d}r$$

Corollary: if rank function is within c factor of Gittins's, then $\mathbf{E}[N]$ is within c^2 of optimal

Robustness of Gittins

$$\mathbf{E}[N] = \int_0^\infty \frac{\mathbf{E}[W(r)]}{r^2}\,\mathrm{d}r$$

Corollary: if rank function is within c factor of Gittins's, then $\mathbf{E}[N]$ is within c^2 of optimal

Proof: change of variables in integral

Robustness of Gittins

$$\mathbf{E}[N] = \int_0^\infty \frac{\mathbf{E}[W(r)]}{r^2}\,\mathrm{d}r$$

robustness to noisy job size predictions
[Scully et al., 2022]

Corollary: if rank function is within c factor of Gittins's, then $\mathbf{E}[N]$ is within c^2 of optimal

Proof: change of variables in integral

Robustness of Gittins

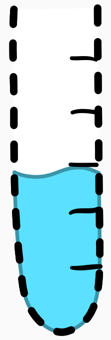
$$\mathbf{E}[N] = \int_0^\infty \frac{\mathbf{E}[W(r)]}{r^2}\,\mathrm{d}r$$

Corollary: if **rank** function is within c factor of **Gittins**'s, then $\mathbf{E}[N]$ is within c^2 of optimal

Proof: change of variables in integral

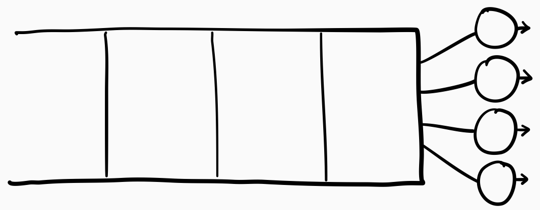
robustness to noisy job size predictions
[Scully et al., 2022]

can substitute n samples for true distribution S
[Ramakrishna et al., 2025]



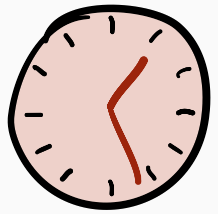
Part I

Handling job size uncertainty



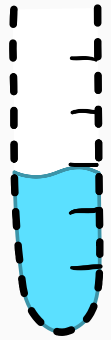
Part II

Analyzing multiserver scheduling



Part III

Optimizing tail metrics

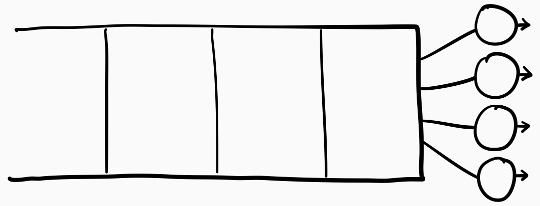


Part I

Handling job size uncertainty

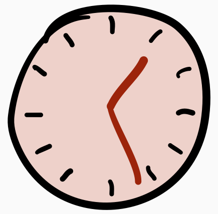
Queueing for **TCS**

*Use **WINE** to analyze **Gittins** with arbitrary release dates?*



Part II

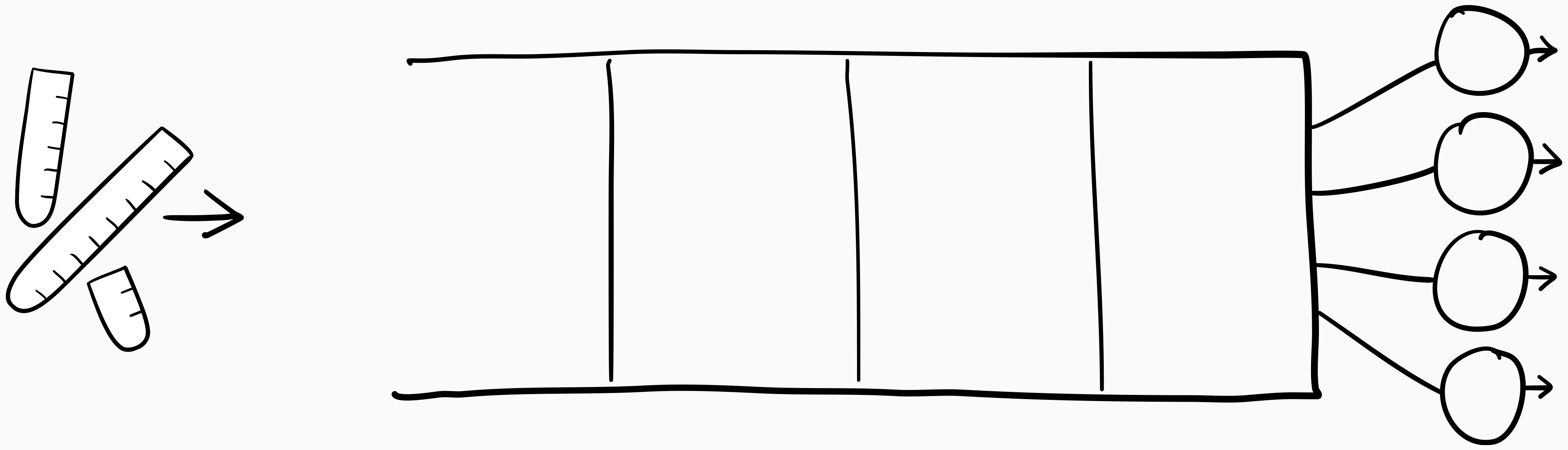
Analyzing multiserver scheduling



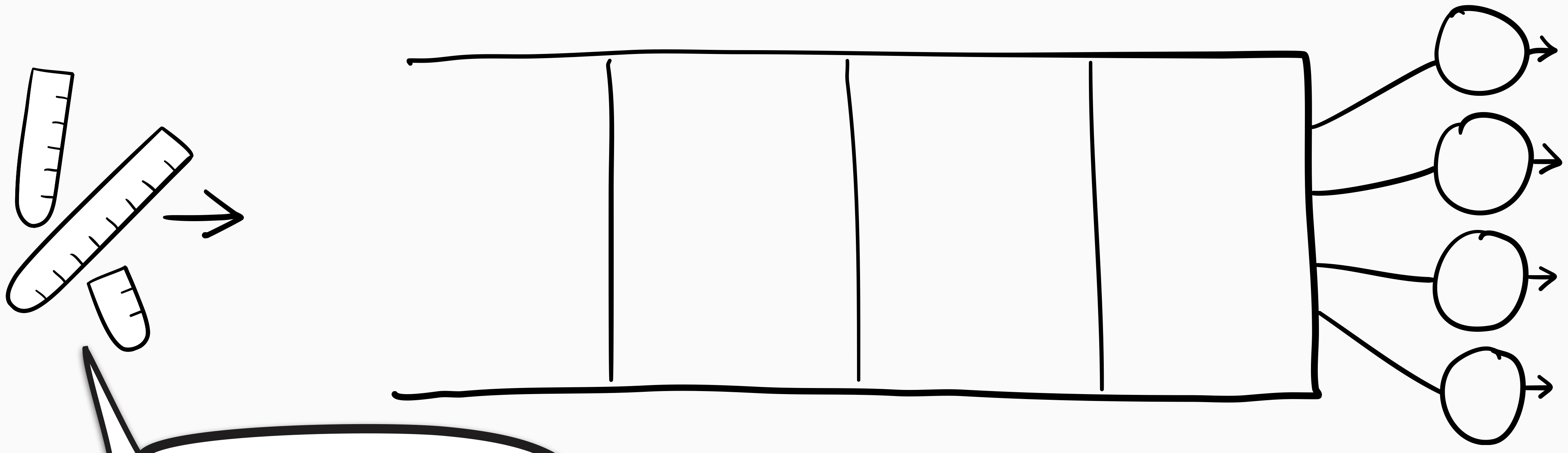
Part III

Optimizing tail metrics

Scheduling in the **M/G/*k***



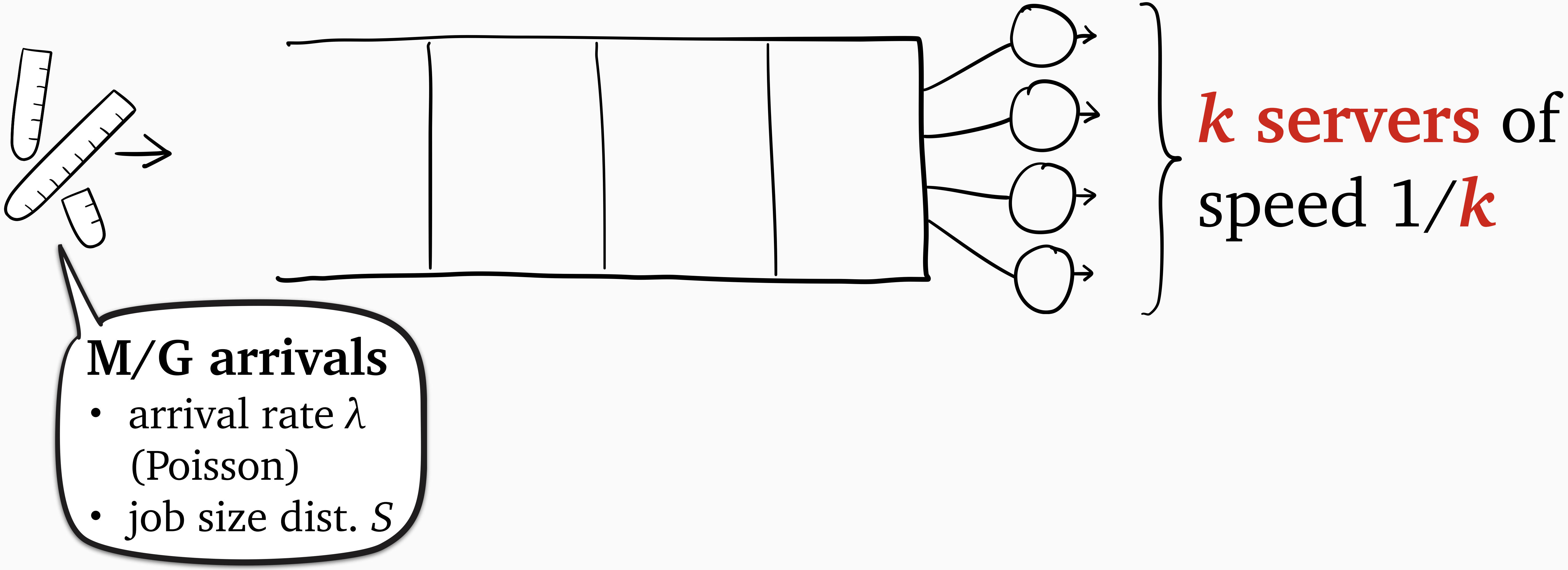
Scheduling in the M/G/k



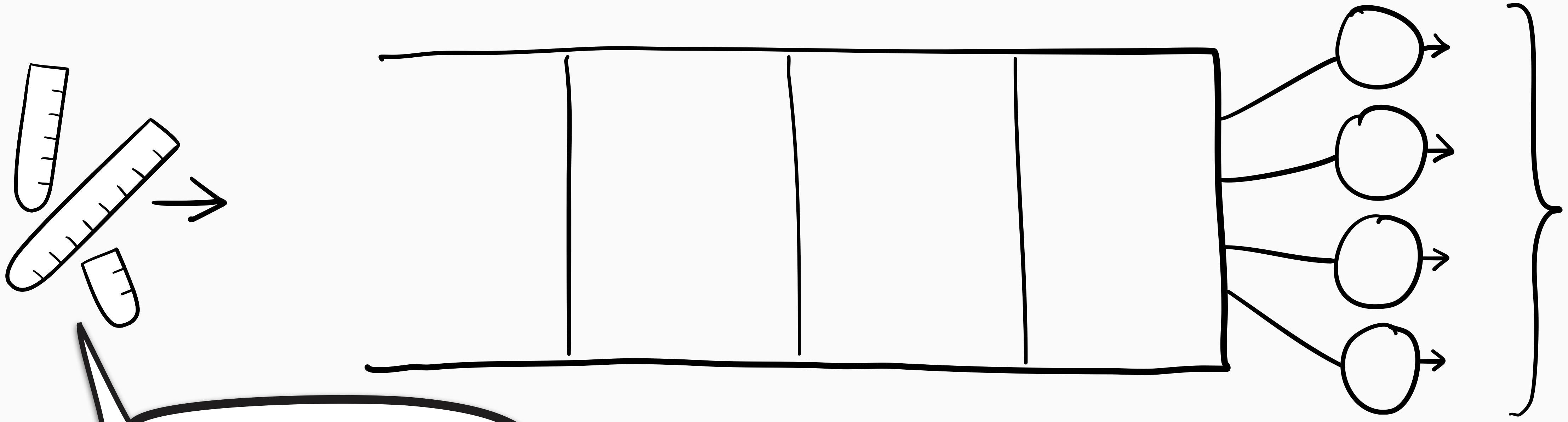
M/G arrivals

- arrival rate λ (Poisson)
- job size dist. S

Scheduling in the M/G/k



Scheduling in the M/G/k



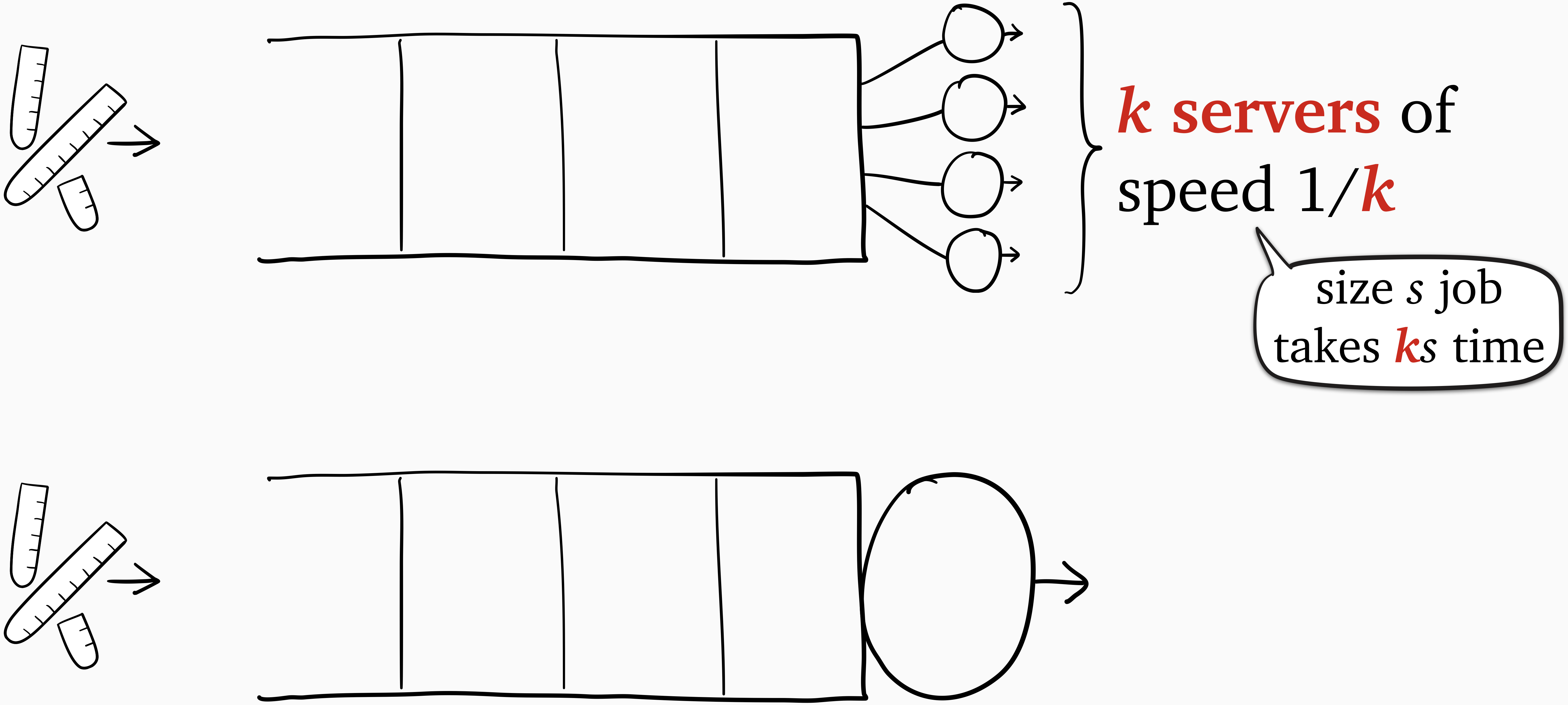
k servers of speed $1/k$

size s job
takes ks time

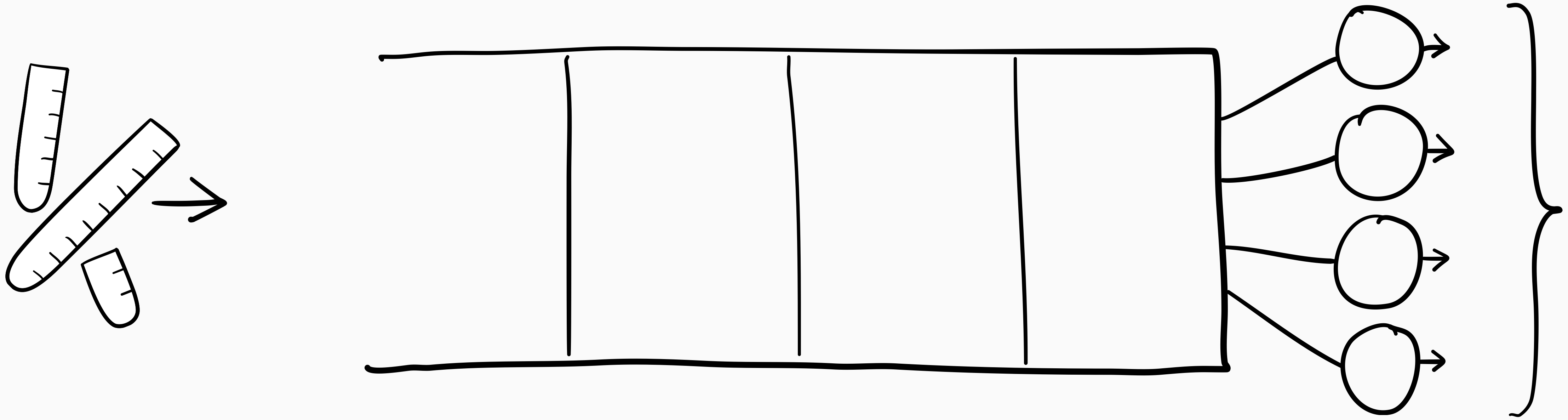
M/G arrivals

- arrival rate λ (Poisson)
- job size dist. S

Scheduling in the **M/G/*k***

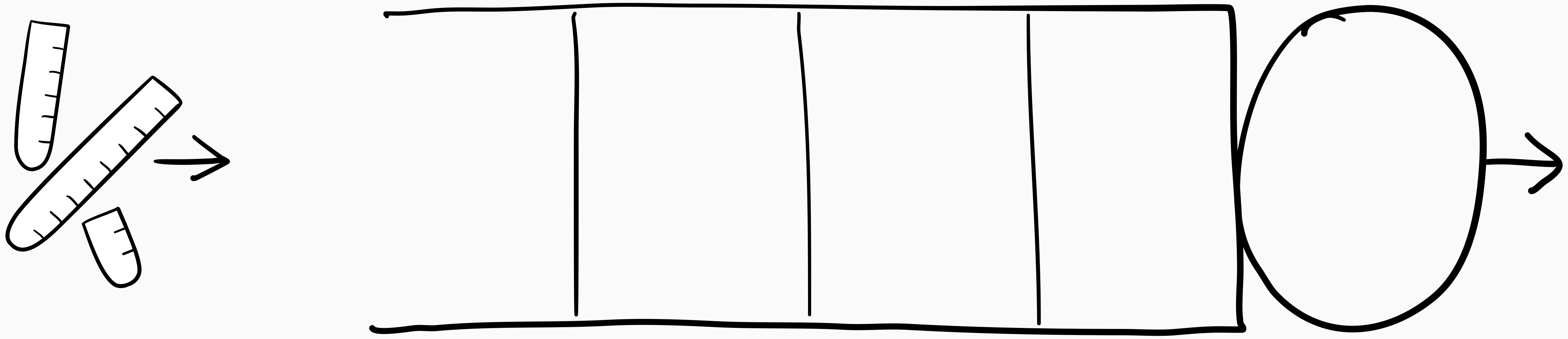


Scheduling in the M/G/k



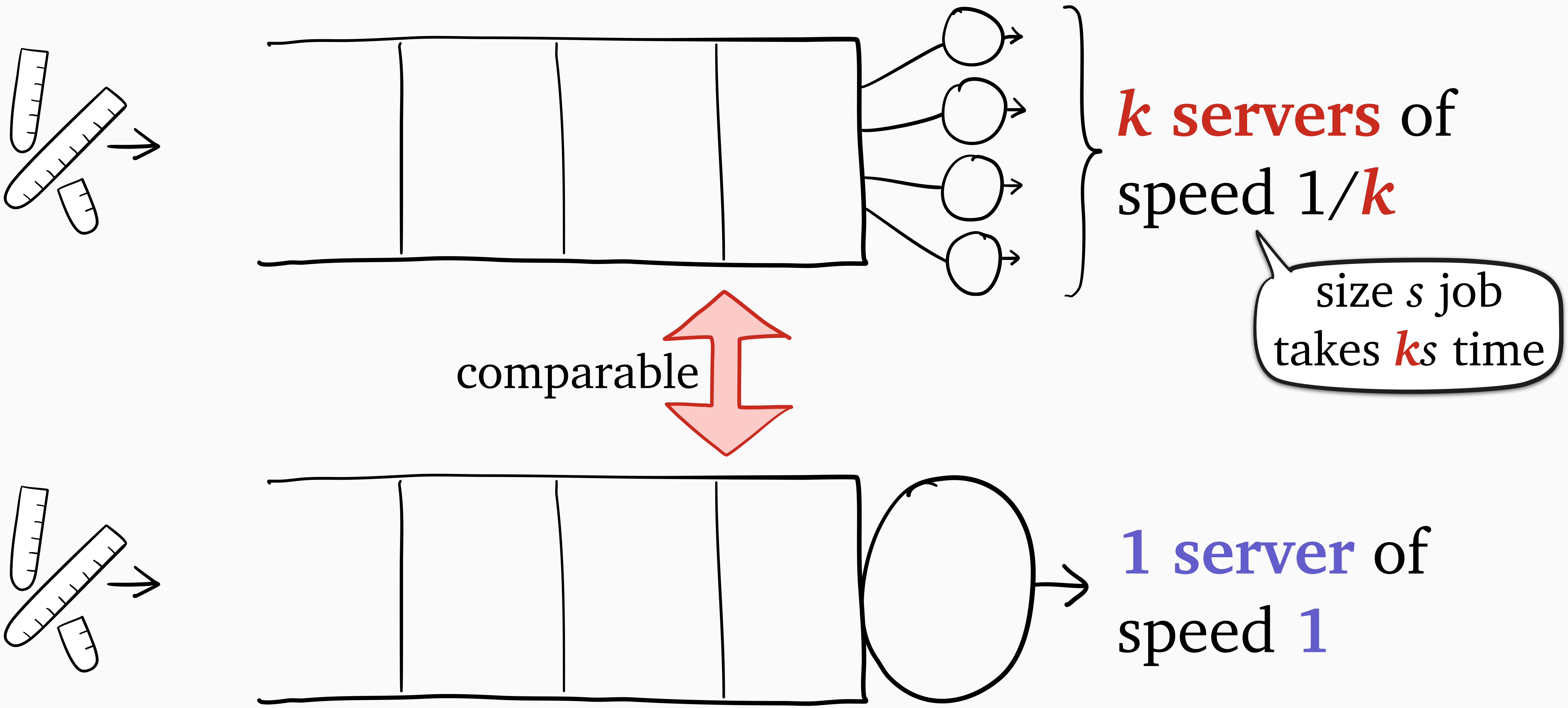
k servers of
speed $1/k$

size s job
takes ks time

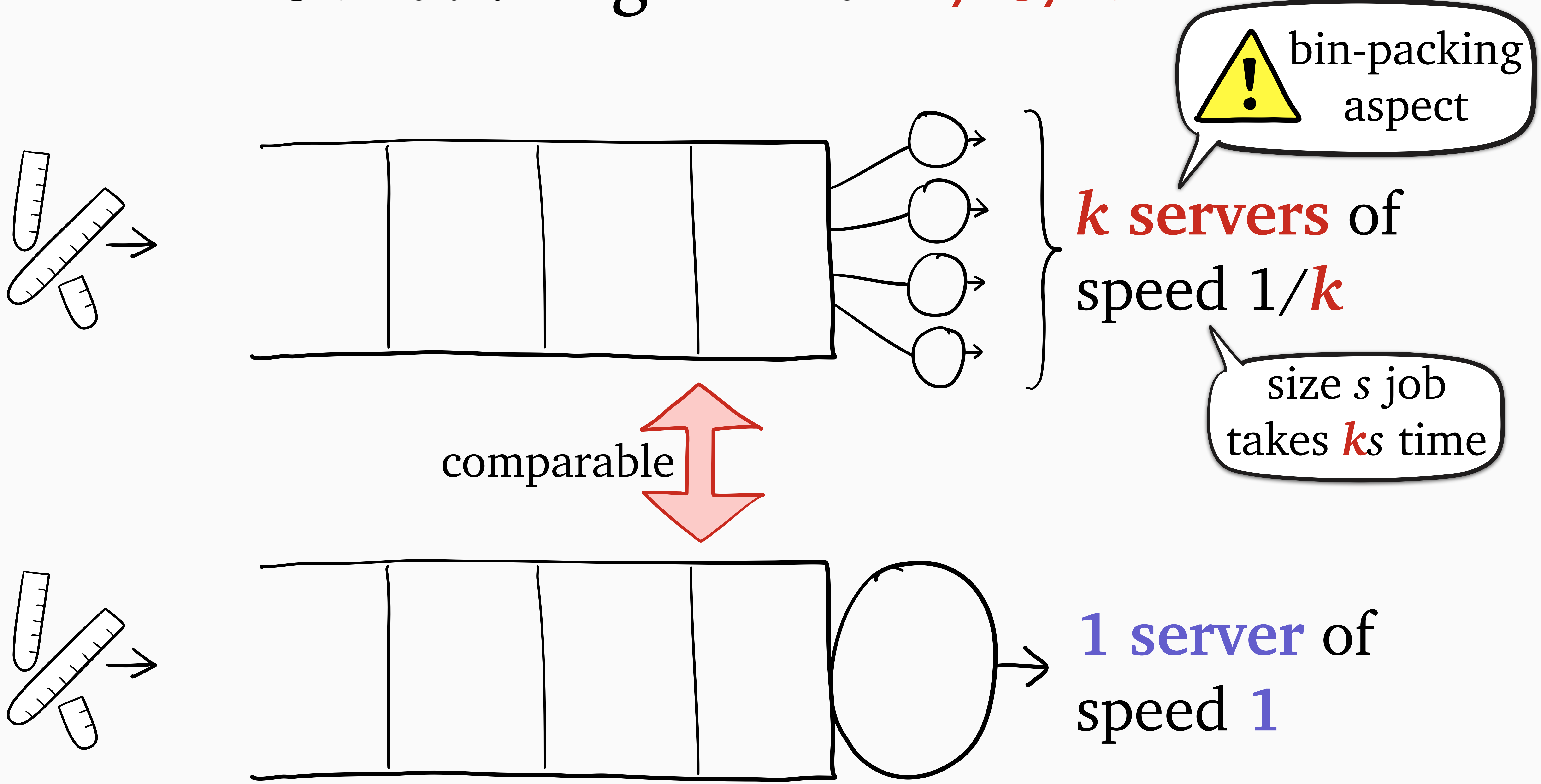


1 server of
speed **1**

Scheduling in the M/G/k



Scheduling in the M/G/k



Q: Are SRPT and Gittins still good in the $\mathbf{M/G/}\boldsymbol{k}$?

Q: Are SRPT and Gittins still good in the **M/G/*k***?

A: Yes, especially in the *heavy traffic limit*

$\rho = \lambda \mathbf{E}[S] \to 1$

Q: Are SRPT and Gittins still good in the M/G/k?

A: Yes, especially in the *heavy traffic limit*

$\rho = \lambda \mathbf{E}[S] \to 1$

Theorem: for SRPT and Gittins,

$$\mathbf{E}[T_k] \le \mathbf{E}[T_1] + (k-1) \cdot O\left(\log \frac{1}{1-\rho}\right)$$

Q: Are SRPT and Gittins still good in the M/G/k?

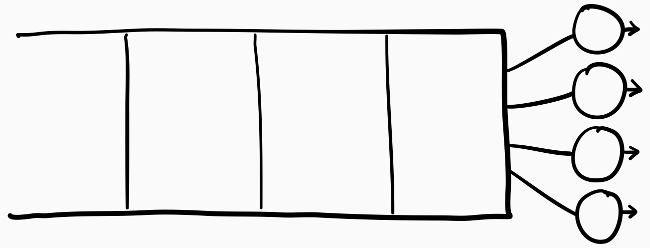
A: Yes, especially in the *heavy traffic limit*

$\rho = \lambda \mathbf{E}[S] \to 1$

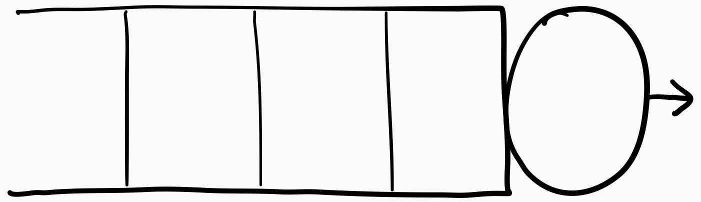
Theorem: for SRPT and Gittins,

$$\mathbf{E}[T_k] \le \mathbf{E}[T_1] + \underbrace{(k-1) \cdot O\left(\log \frac{1}{1-\rho}\right)}_{\mathbf{E}[S^2(\log S)^+] < \infty \;\Rightarrow\; o(\mathbf{E}[T_1])}$$

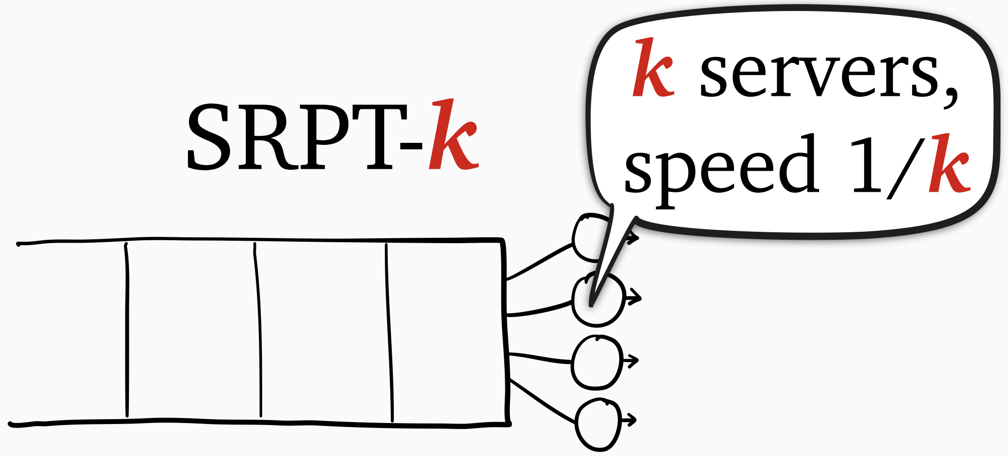
SRPT-k



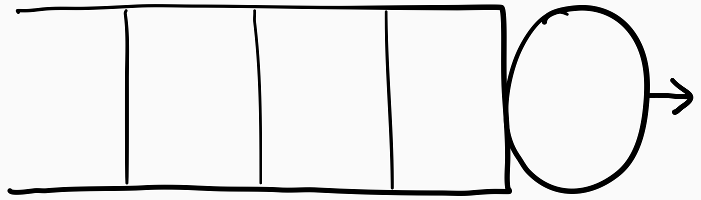
SRPT-1

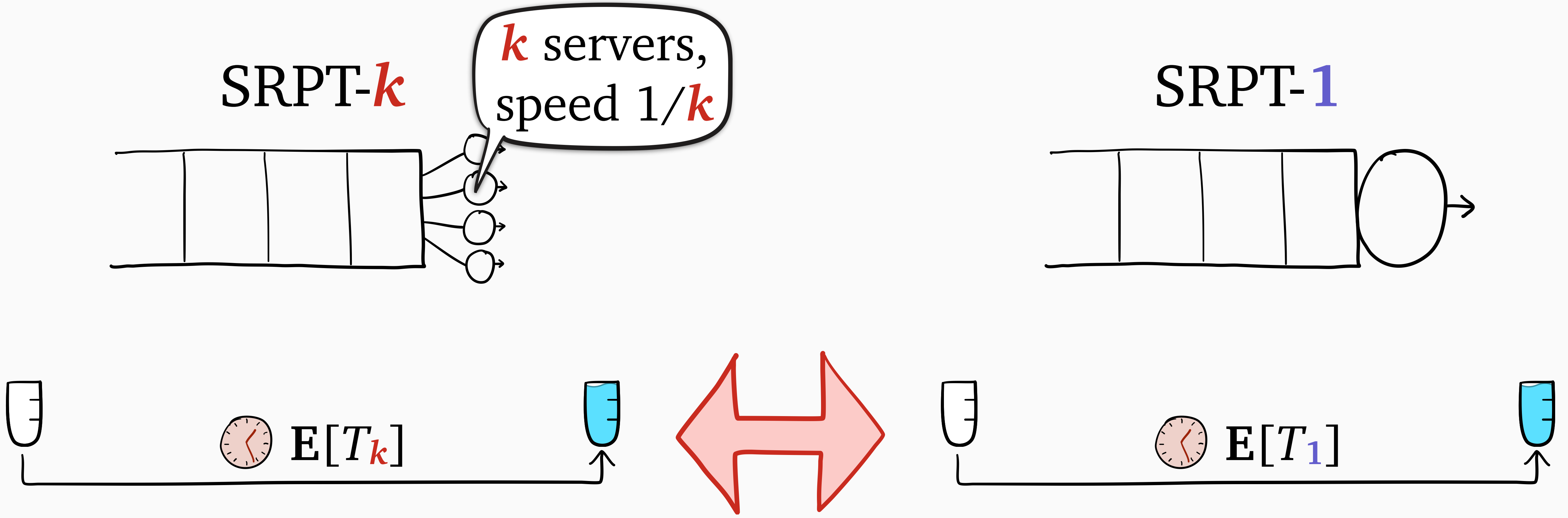


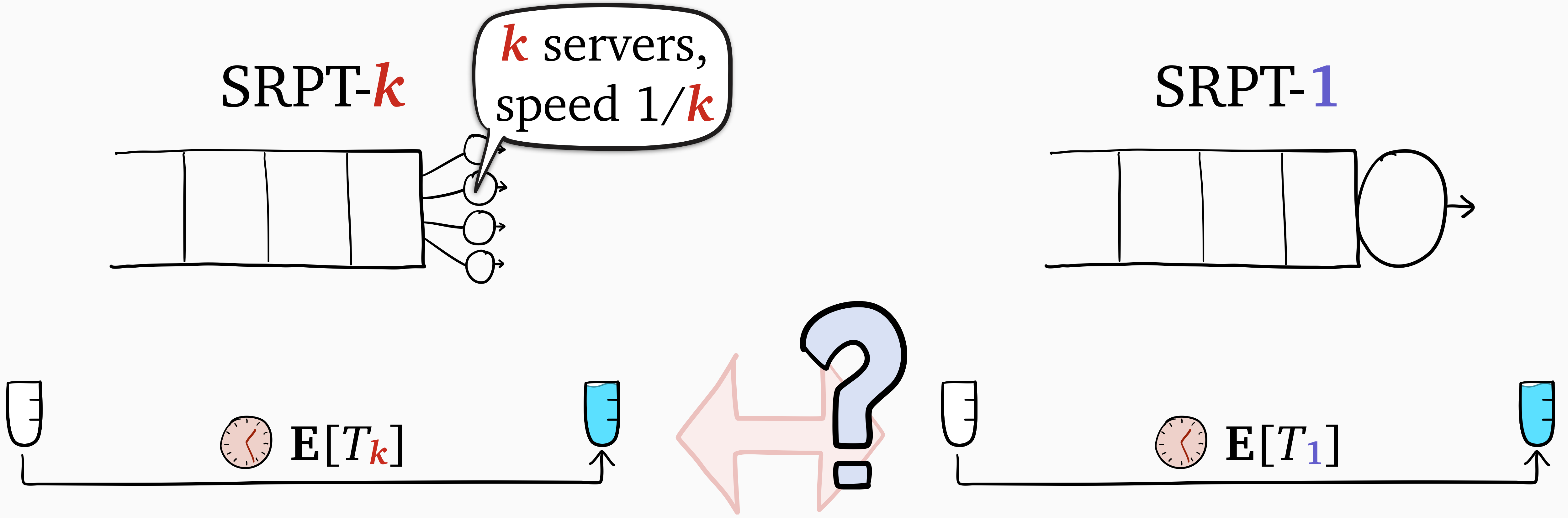
SRPT-k

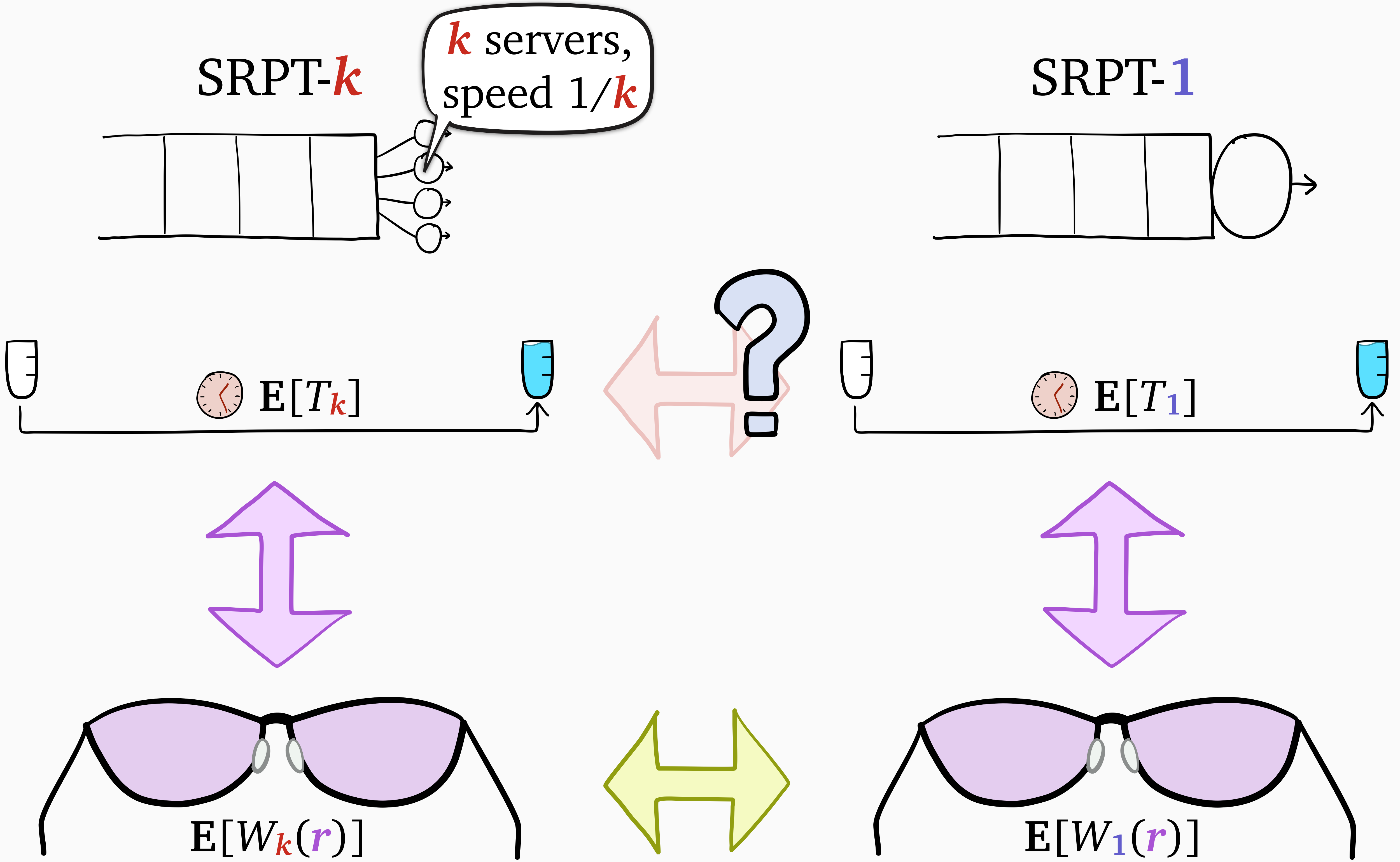


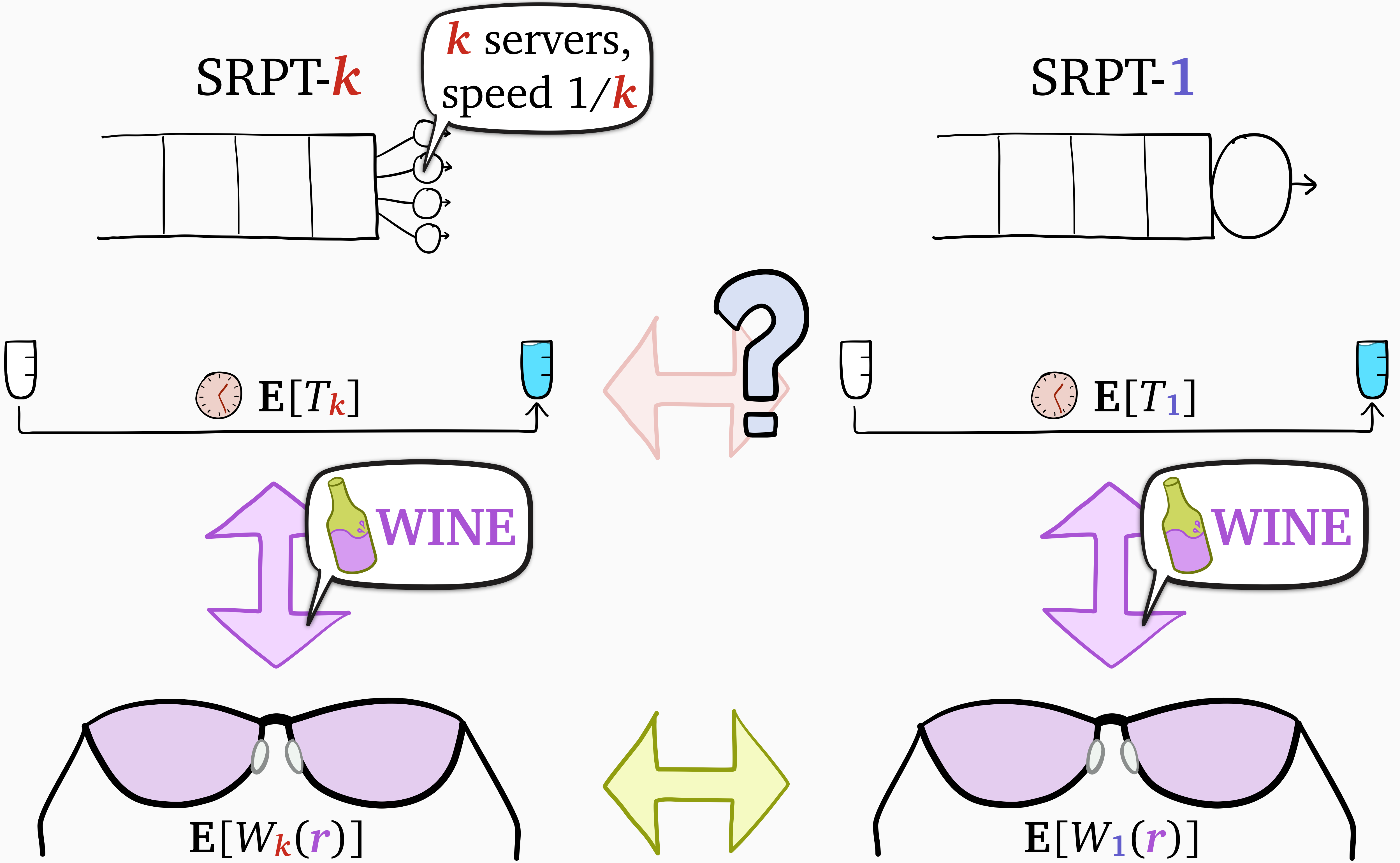
SRPT-1

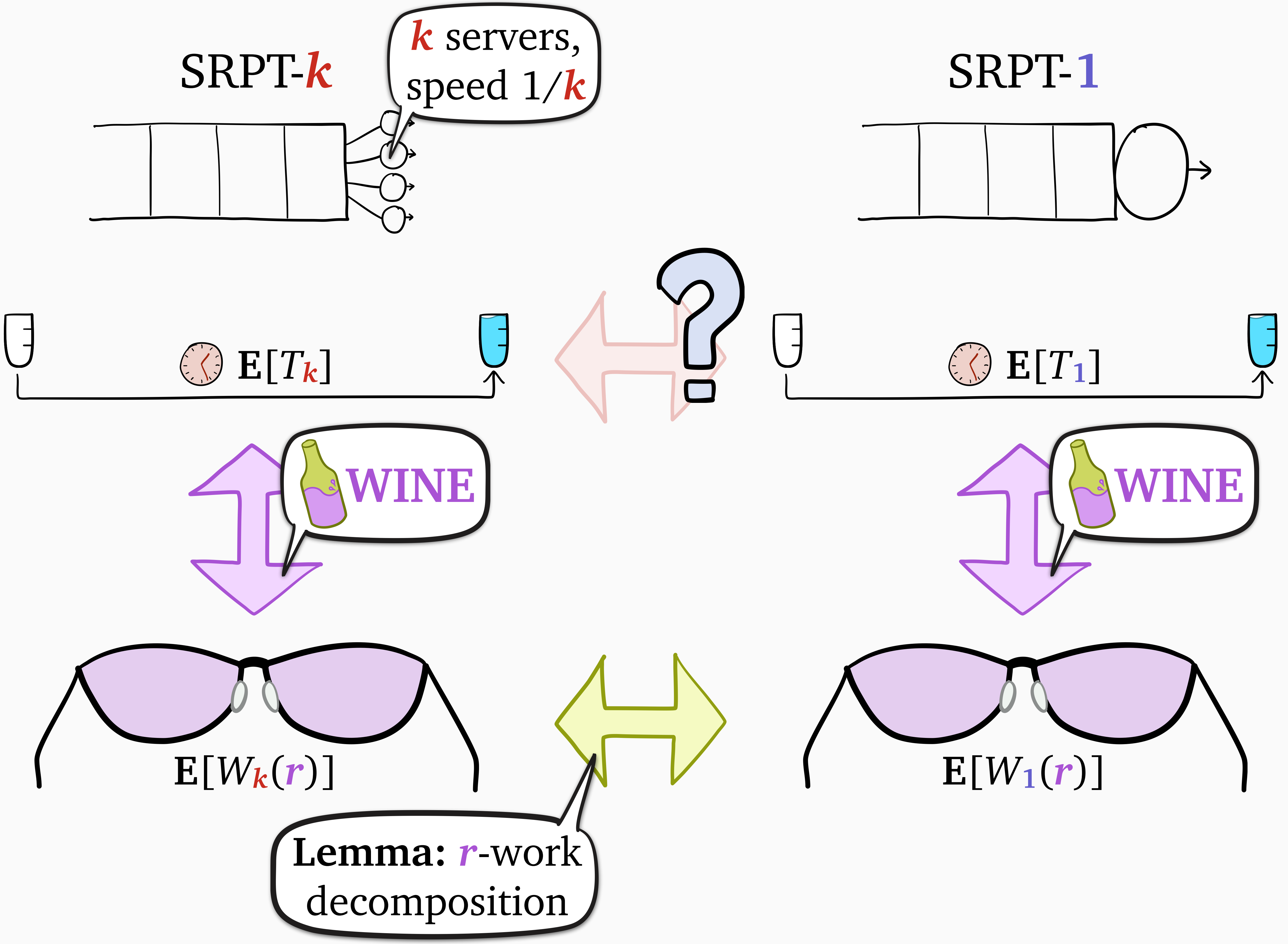


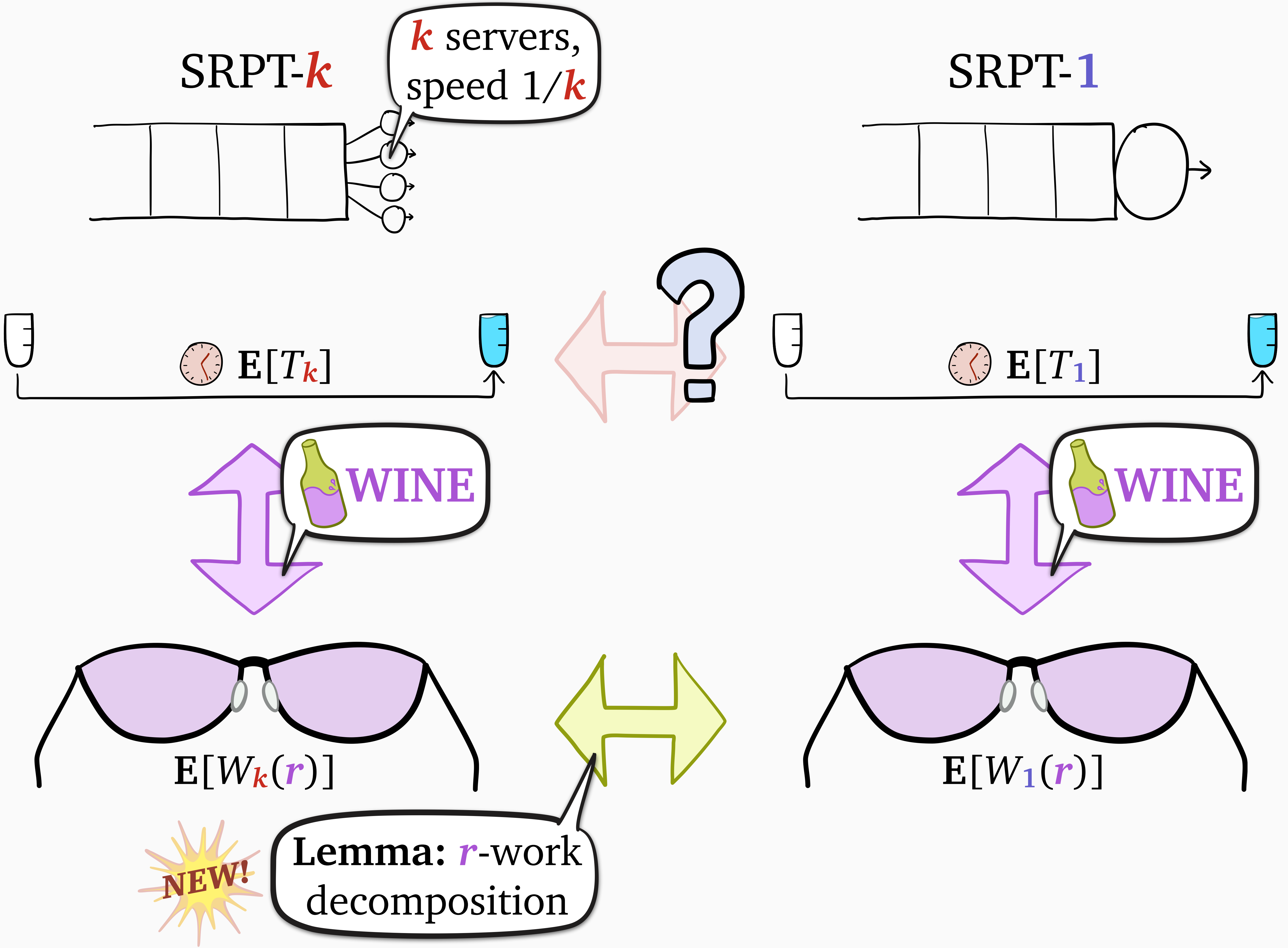

SRPT-k
k servers, speed $1/k$
SRPT-1
$\mathbf{E}[T_k]$
$\mathbf{E}[T_1]$


SRPT-k
k servers,
speed 1/k
SRPT-1
E[T_k]
E[T_1]
?

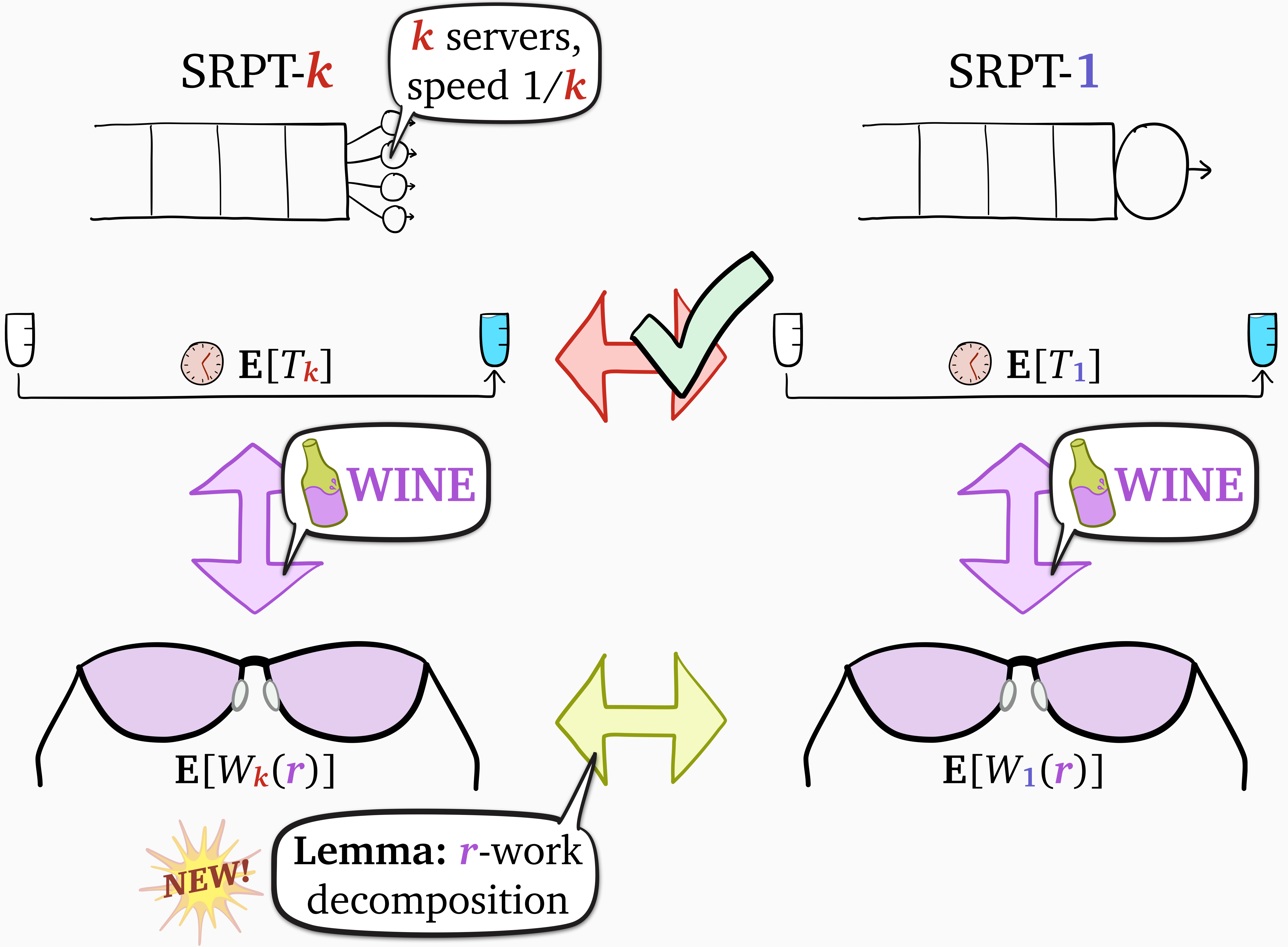

SRPT-k
k servers, speed $1/k$
SRPT-1
$\mathbf{E}[T_k]$
$\mathbf{E}[T_1]$
$\mathbf{E}[W_k(r)]$
$\mathbf{E}[W_1(r)]$

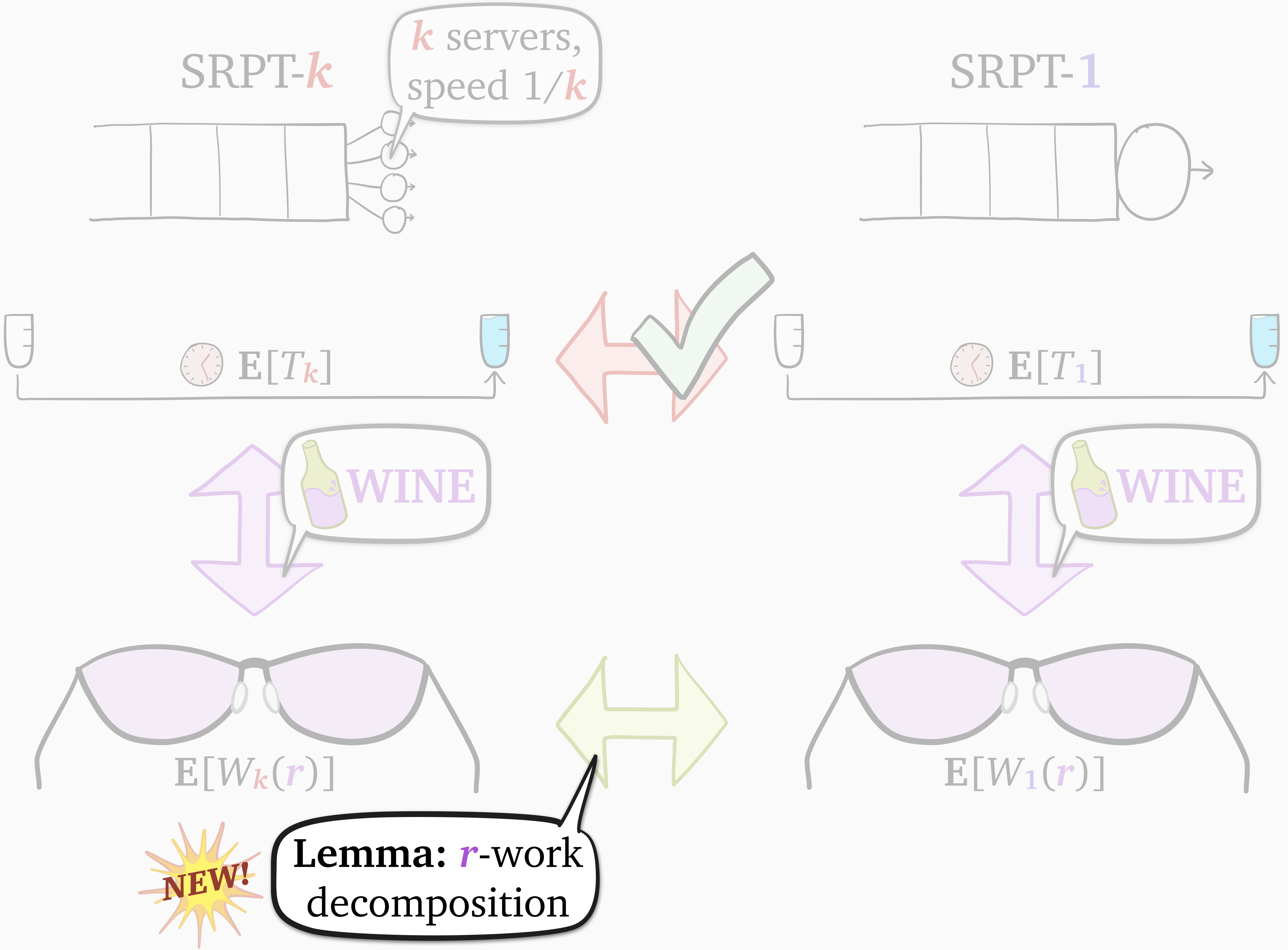

SRPT-k
k servers, speed $1/k$
SRPT-1
$\mathbf{E}[T_k]$
$\mathbf{E}[T_1]$
WINE
WINE
$\mathbf{E}[W_k(r)]$
$\mathbf{E}[W_1(r)]$


SRPT-k
k servers, speed $1/k$
SRPT-1
$\mathbf{E}[T_k]$
$\mathbf{E}[T_1]$
WINE
WINE
$\mathbf{E}[W_k(r)]$
$\mathbf{E}[W_1(r)]$
Lemma: r-work decomposition

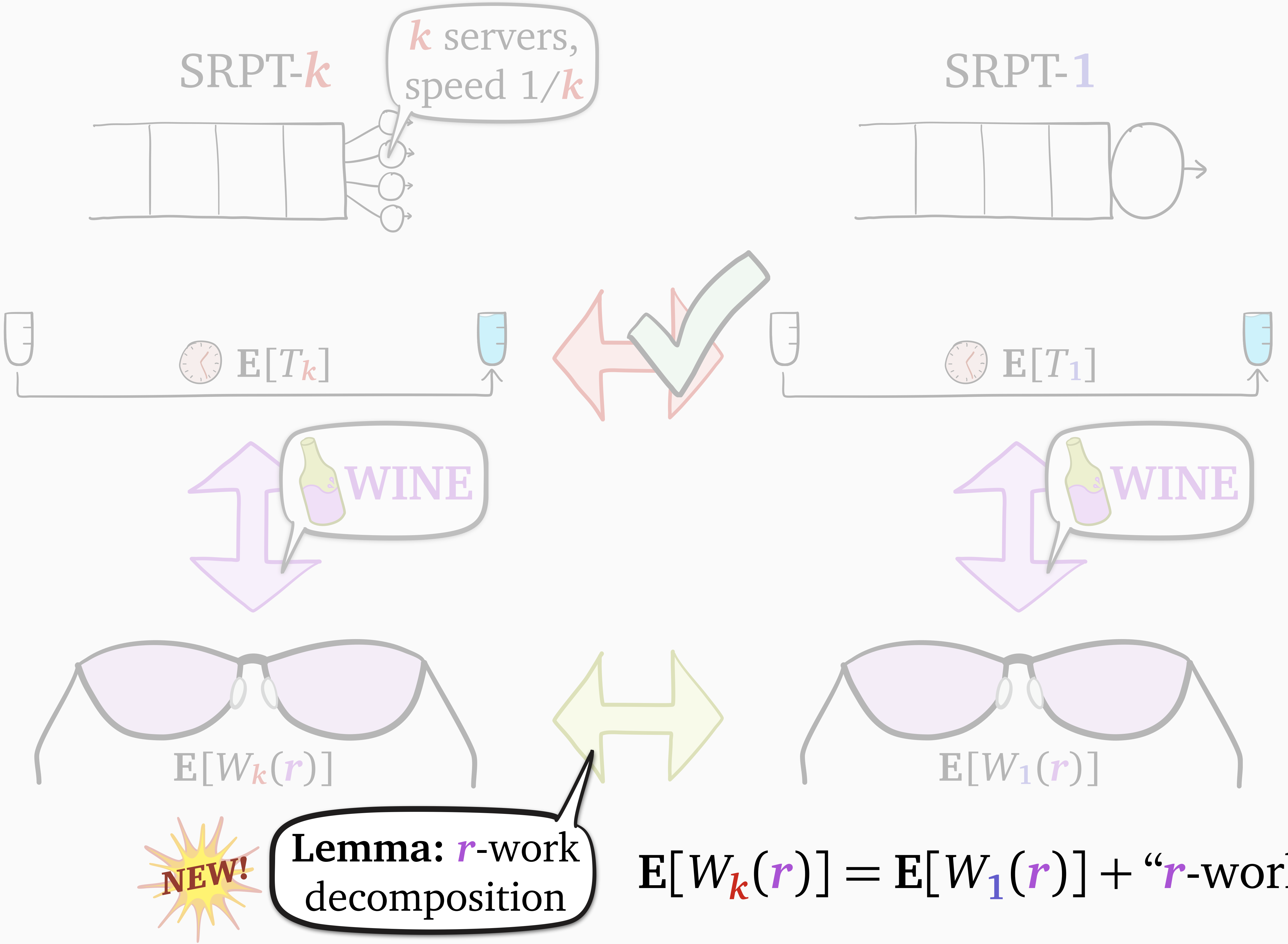


SRPT-k
k servers, speed $1/k$
SRPT-1
$\mathbf{E}[T_k]$
$\mathbf{E}[T_1]$
WINE
WINE
$\mathbf{E}[W_k(r)]$
$\mathbf{E}[W_1(r)]$
NEW!
Lemma: r-work decomposition

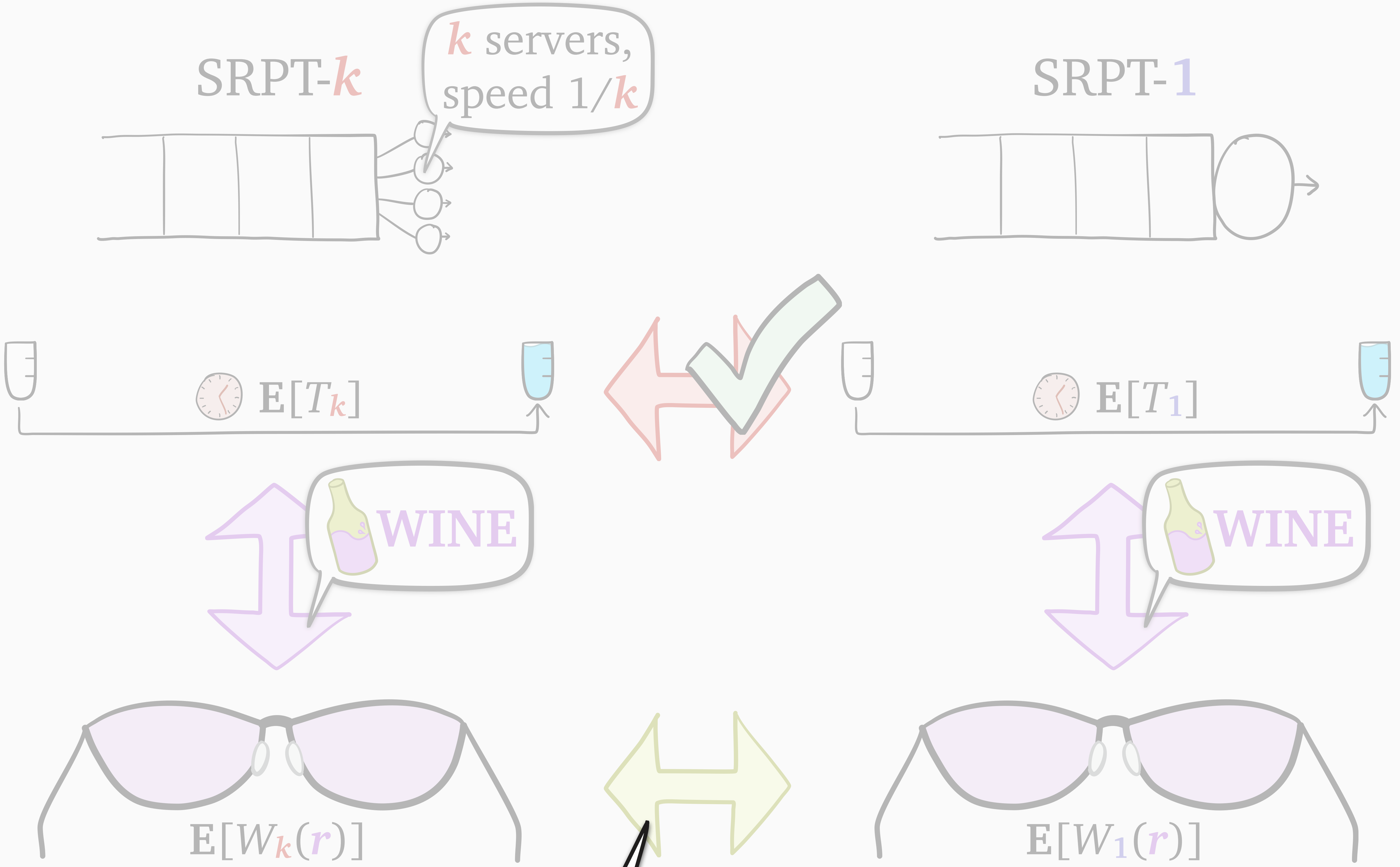

SRPT-k
k servers, speed $1/k$
SRPT-1
$\mathbf{E}[T_k]$
$\mathbf{E}[T_1]$
WINE
WINE
$\mathbf{E}[W_k(r)]$
$\mathbf{E}[W_1(r)]$
NEW!
Lemma: r-work decomposition



SRPT-k
k servers, speed $1/k$
SRPT-1
$\mathbf{E}[T_k]$
$\mathbf{E}[T_1]$
WINE
WINE
$\mathbf{E}[W_k(r)]$
$\mathbf{E}[W_1(r)]$
NEW!
Lemma: r-work decomposition



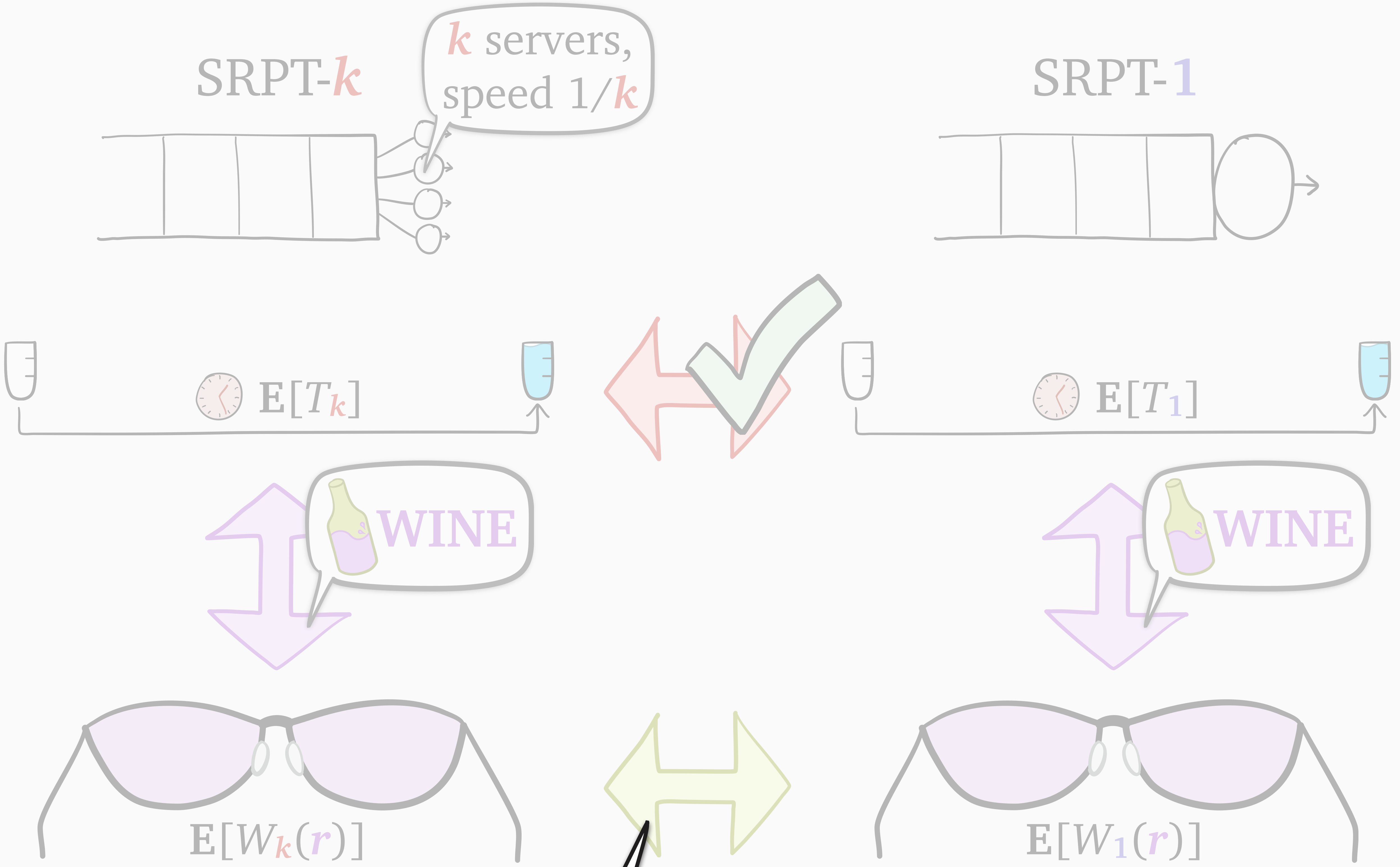
$$\mathbf{E}[W_k(r)] = \mathbf{E}[W_1(r)] + \text{“}r\text{-work of } k-1 \text{ jobs”}$$



NEW!

Lemma: r-work decomposition

$$\mathbf{E}[W_k(r)] = \mathbf{E}[W_1(r)] + \text{“}r\text{-work of } k-1 \text{ jobs”}$$
$$\leq \mathbf{E}[W_1(r)] + (k-1)r$$

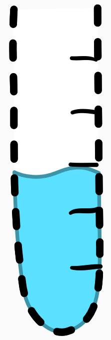


NEW!

Lemma: r-work decomposition

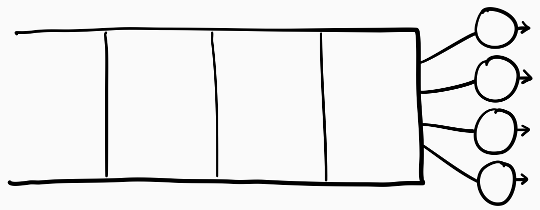
$$\mathbf{E}[W_k(r)] = \mathbf{E}[W_1(r)] + \text{“}r\text{-work of } k-1 \text{ jobs”}$$
$$\leq \mathbf{E}[W_1(r)] + (k-1)r$$

can improve



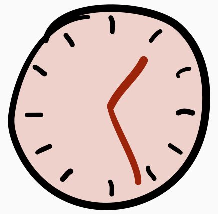
Part I

Handling job size uncertainty



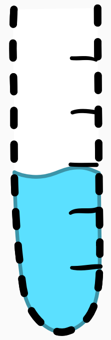
Part II

Analyzing multiserver scheduling



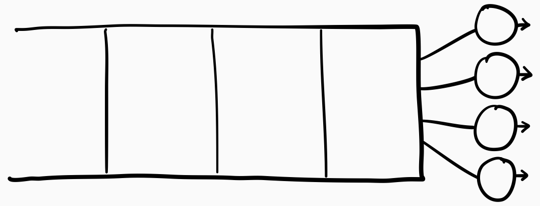
Part III

Optimizing tail metrics



Part I

Handling job size uncertainty

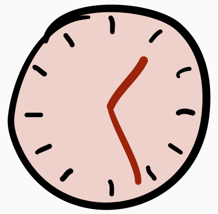


Part II

Analyzing multiserver scheduling

Queueing for TCS

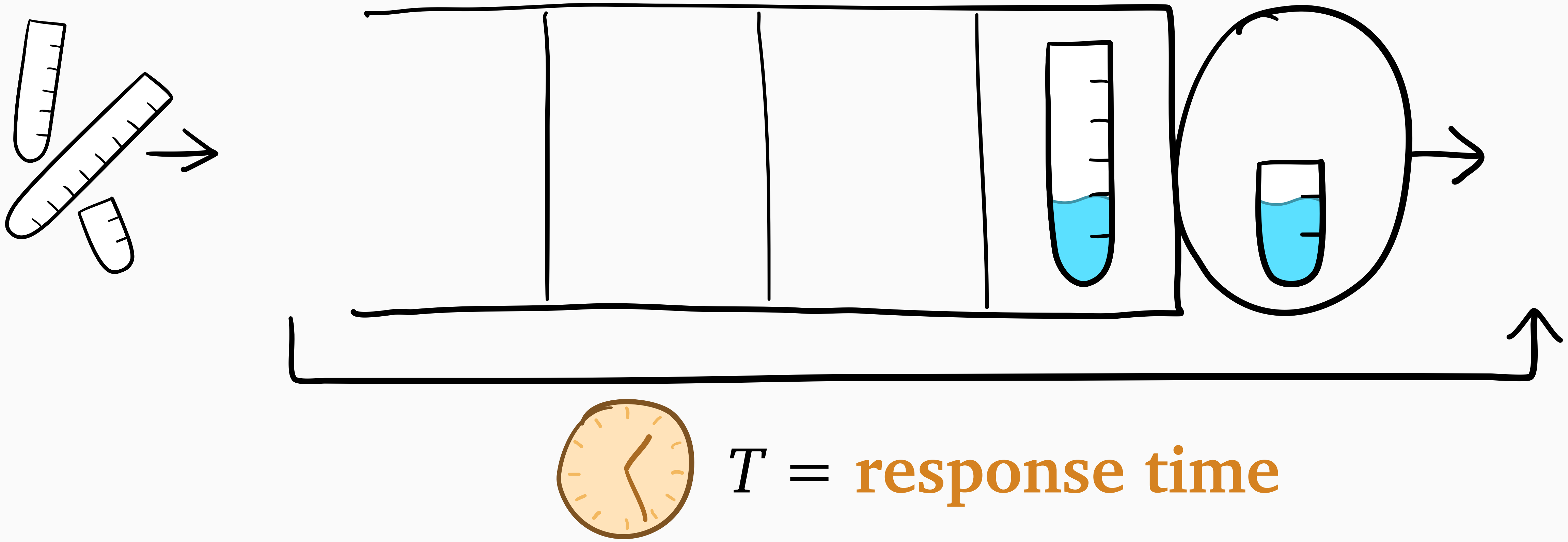
*Use **WINE** to analyze SRPT-**k** with arbitrary release dates?*



Part III

Optimizing tail metrics

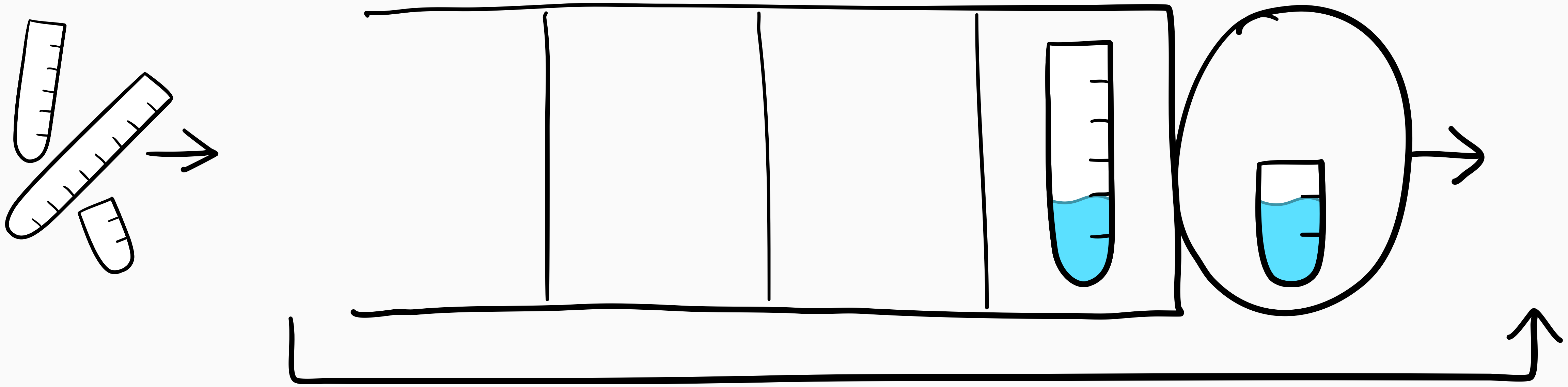
Tail metrics



T = **response time**

Tail metrics

Minimize $\begin{cases} \mathbf{P}[T > t]? \\ \mathbf{E}[(T - t)^+]? \\ \text{quantiles of } T? \end{cases}$

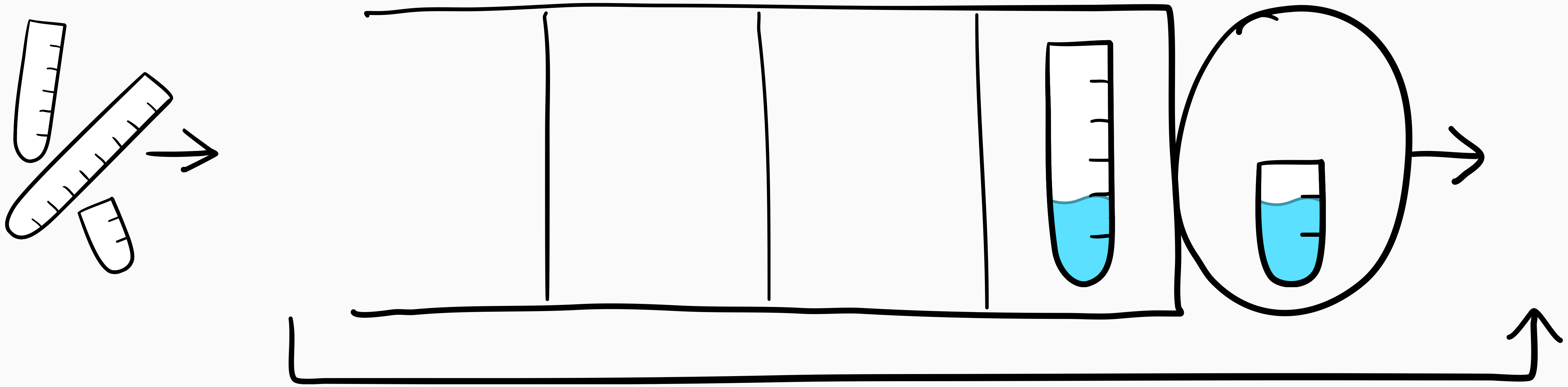


T = **response time**

Tail metrics

$$\text{Minimize} \begin{cases} \mathbf{P}[T > t]? \\ \mathbf{E}[(T - t)^+]? \\ \text{quantiles of } T? \end{cases}$$

Practice: important



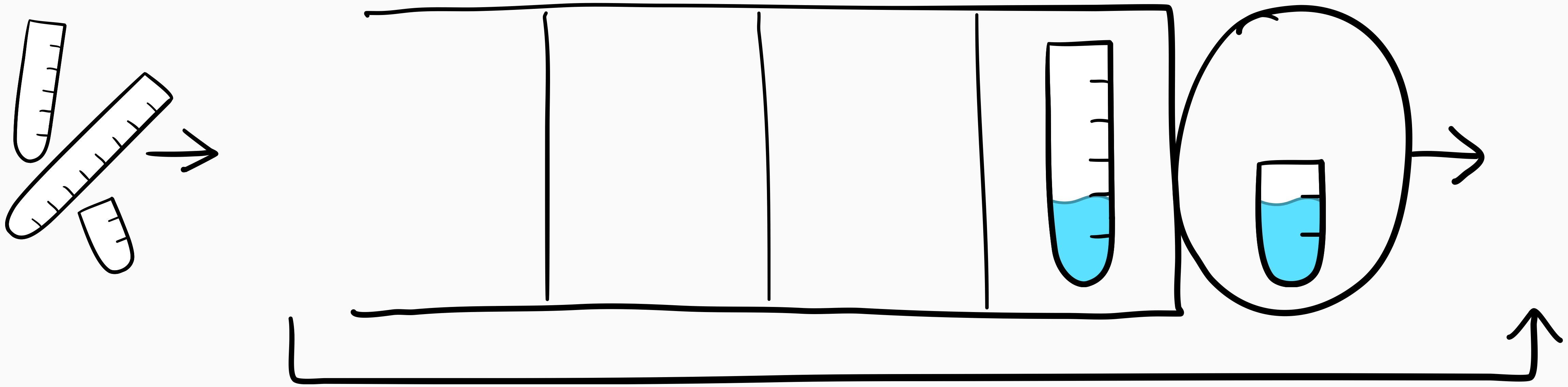
T = **response time**

Tail metrics

$$\text{Minimize} \begin{cases} \mathbf{P}[T > t]? \\ \mathbf{E}[(T - t)^+]? \\ \text{quantiles of } T? \end{cases}$$

Practice: important

Theory: very hard



T = **response time**

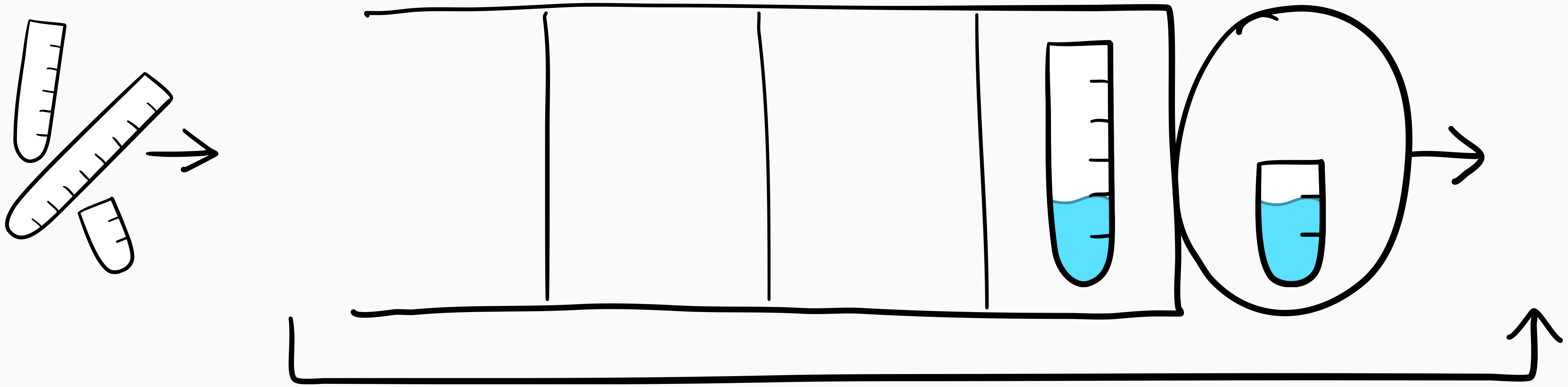
Tail metrics

Minimize $\begin{cases} \mathbf{P}[T > t]? \\ \mathbf{E}[(T - t)^+]? \\ \text{quantiles of } T? \end{cases}$

Practice: important

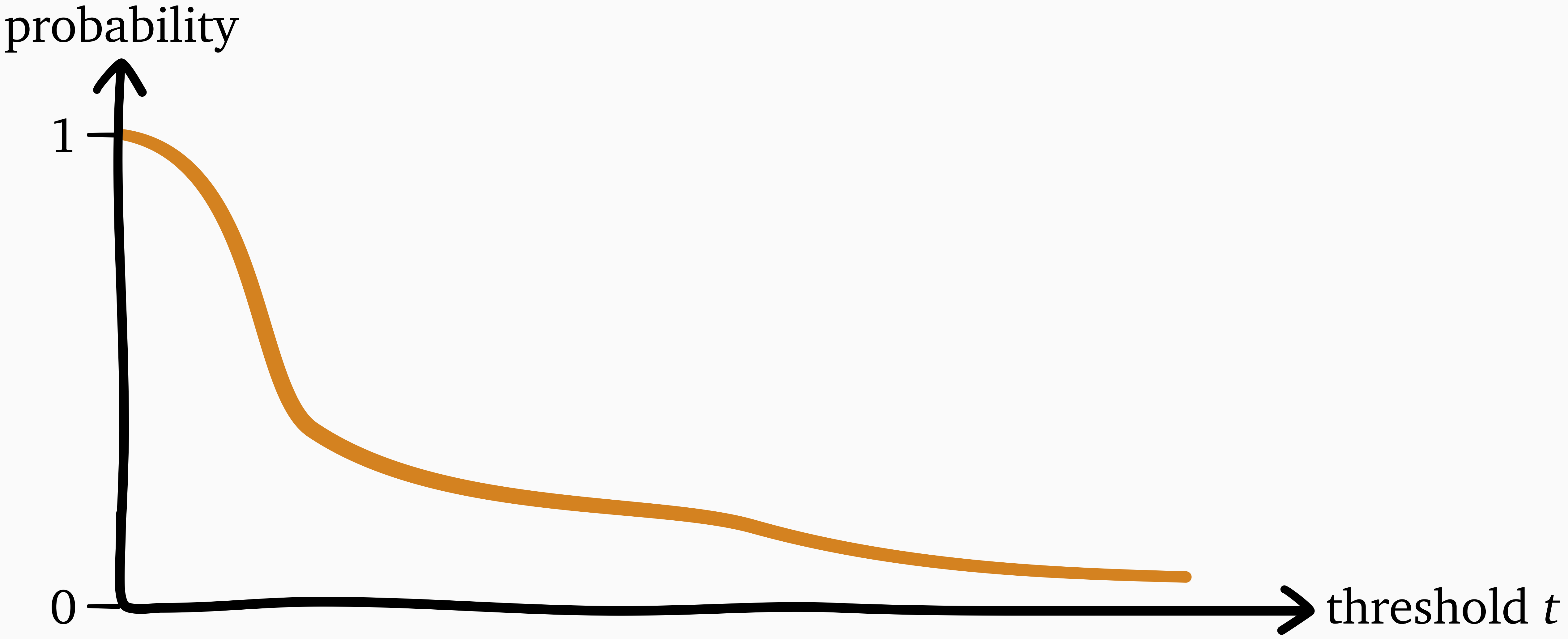
Theory: very hard

Tractable:
study $t \to \infty$
asymptotics

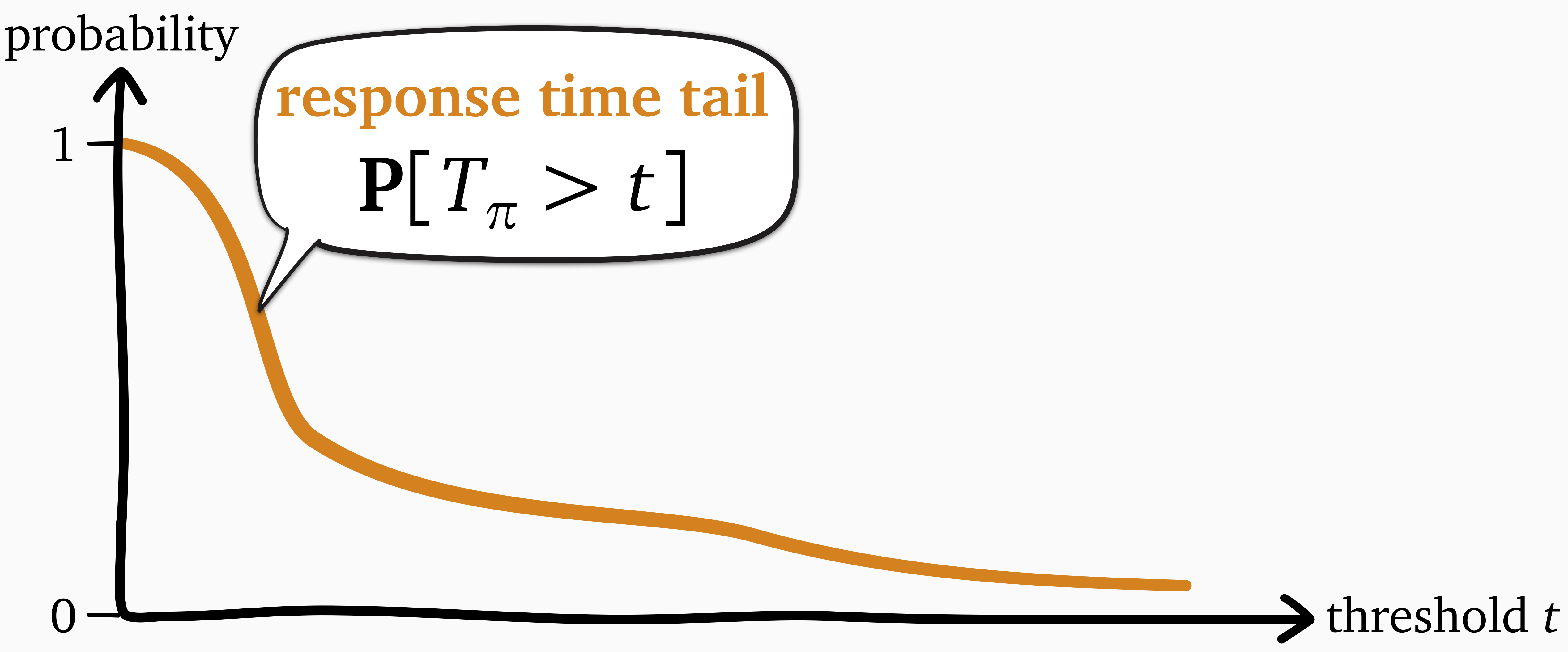


T = **response time**

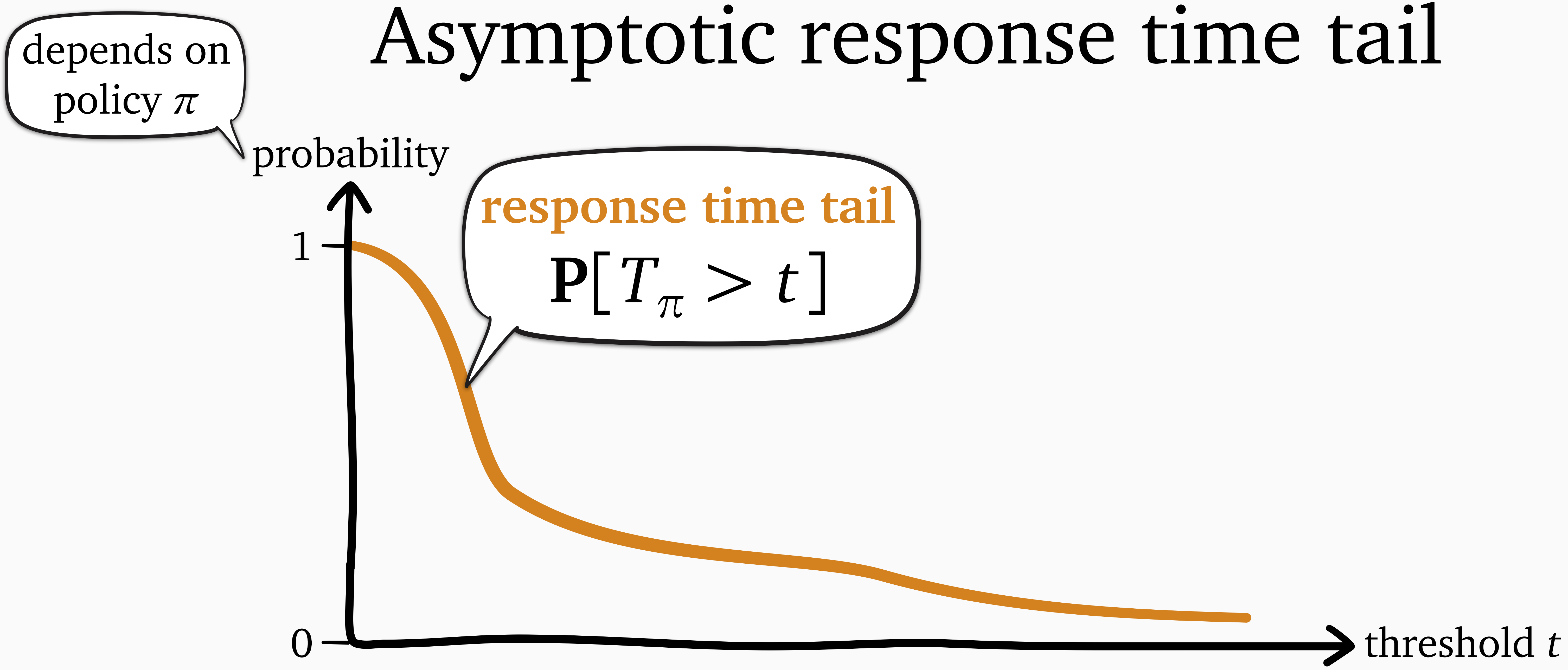
Asymptotic response time tail



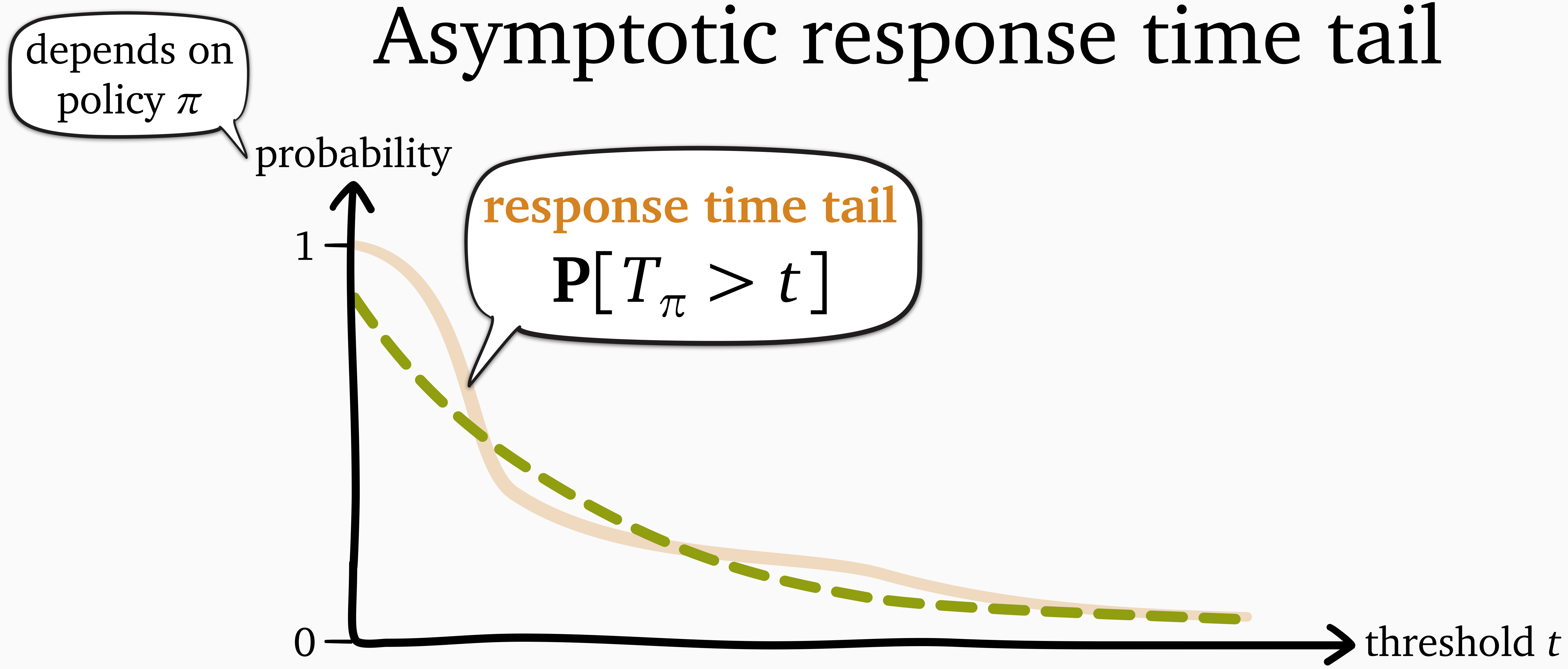
Asymptotic response time tail



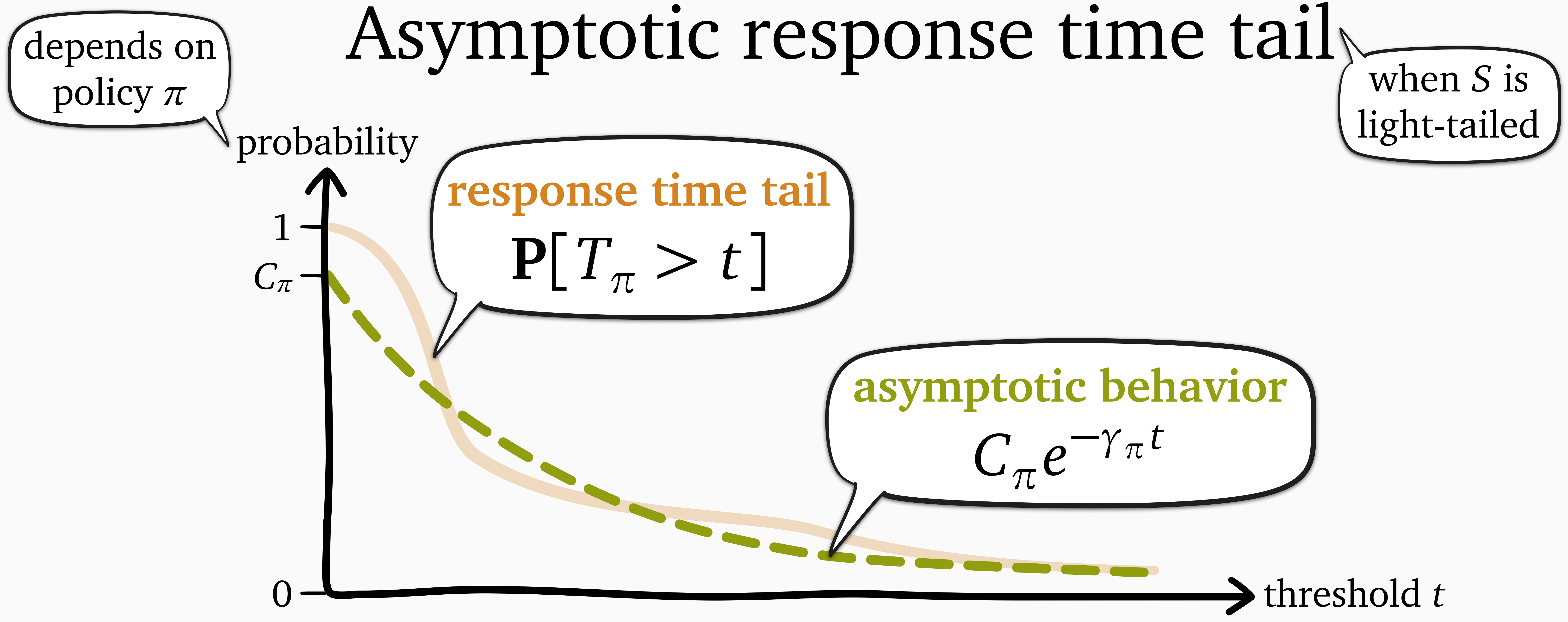
Asymptotic response time tail



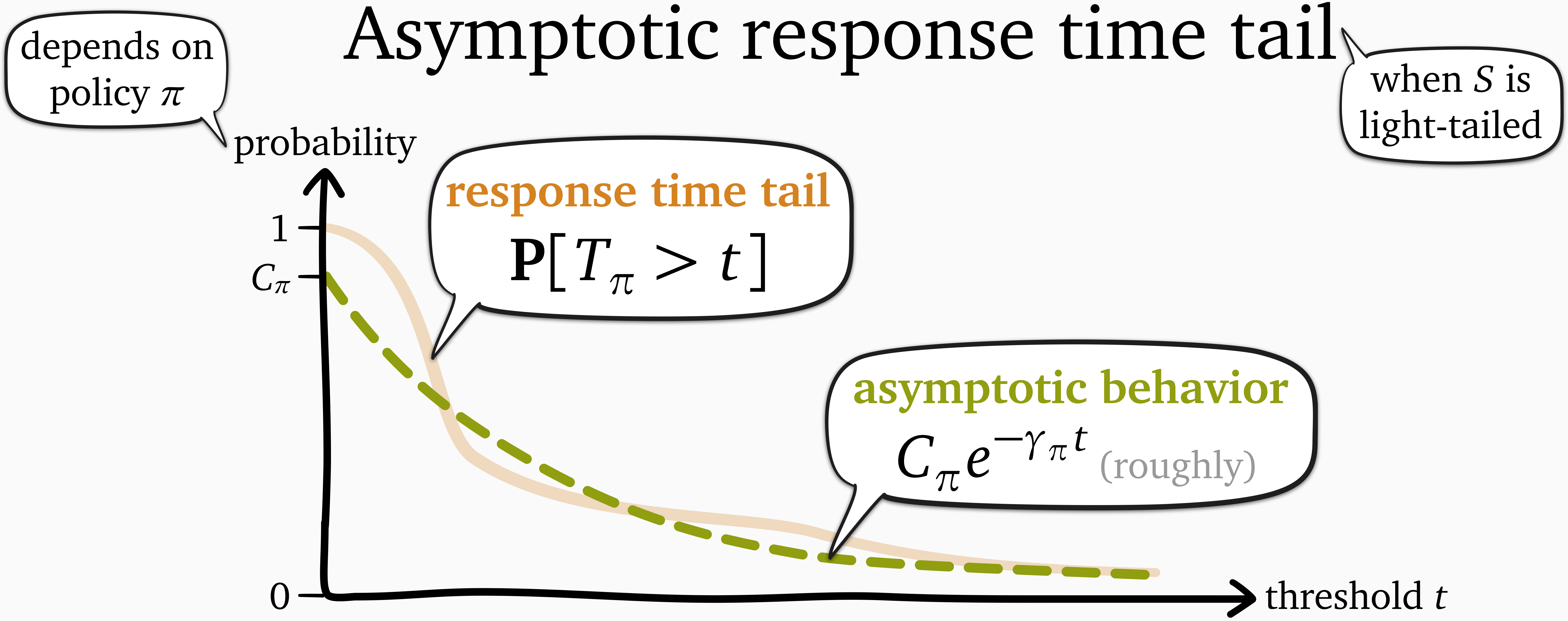
Asymptotic response time tail



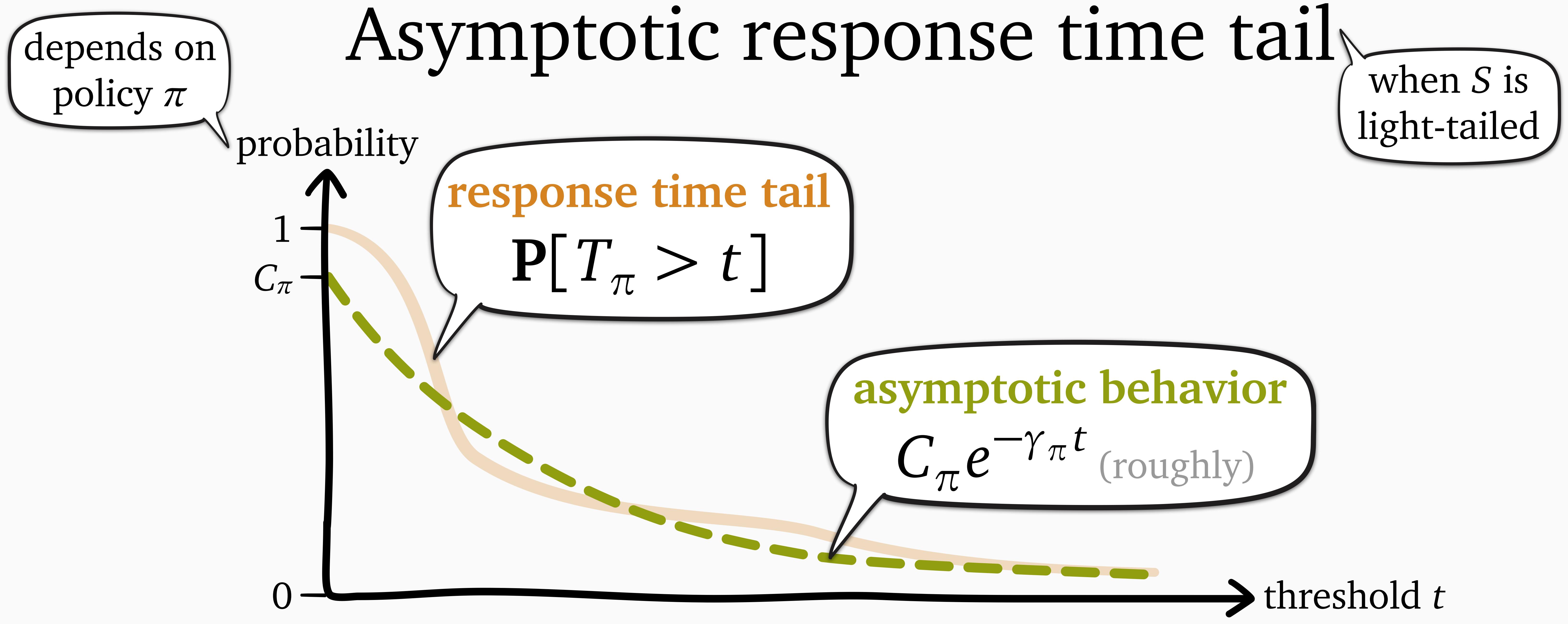
Asymptotic response time tail



Asymptotic response time tail



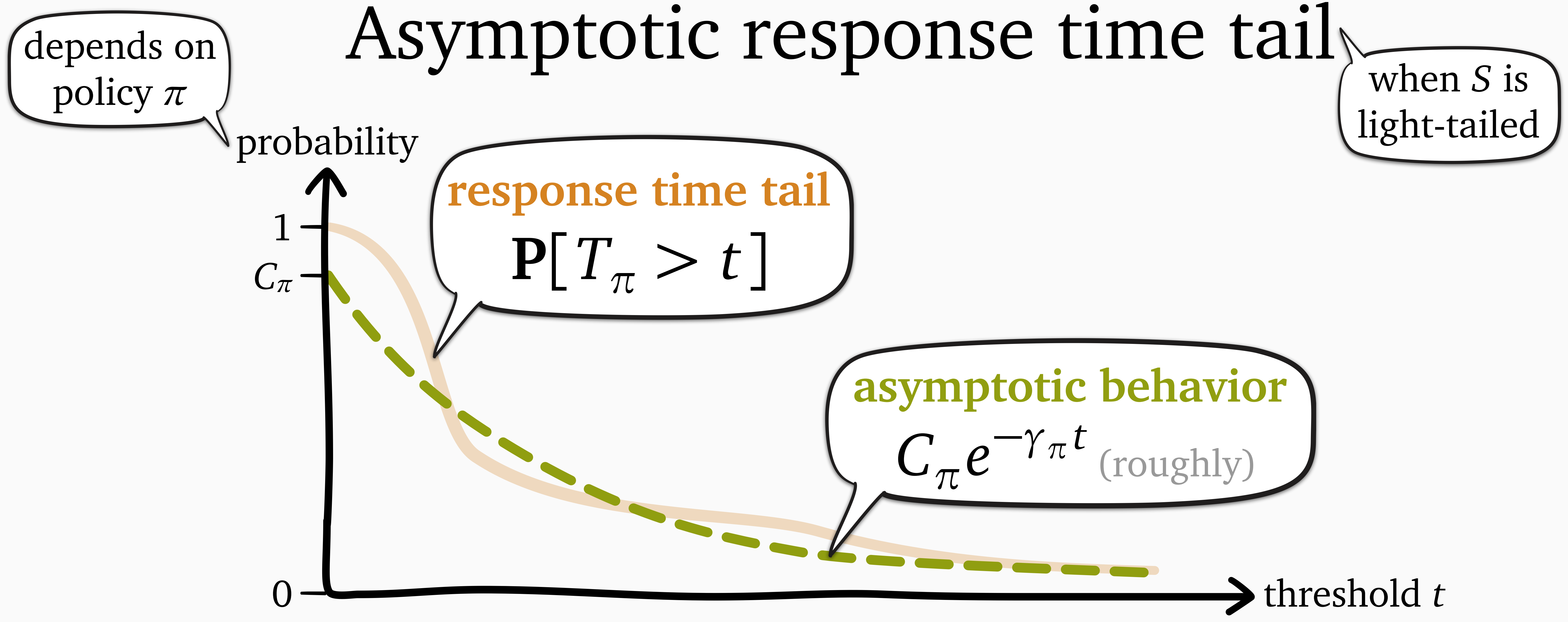
Asymptotic response time tail



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

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

Asymptotic response time tail



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

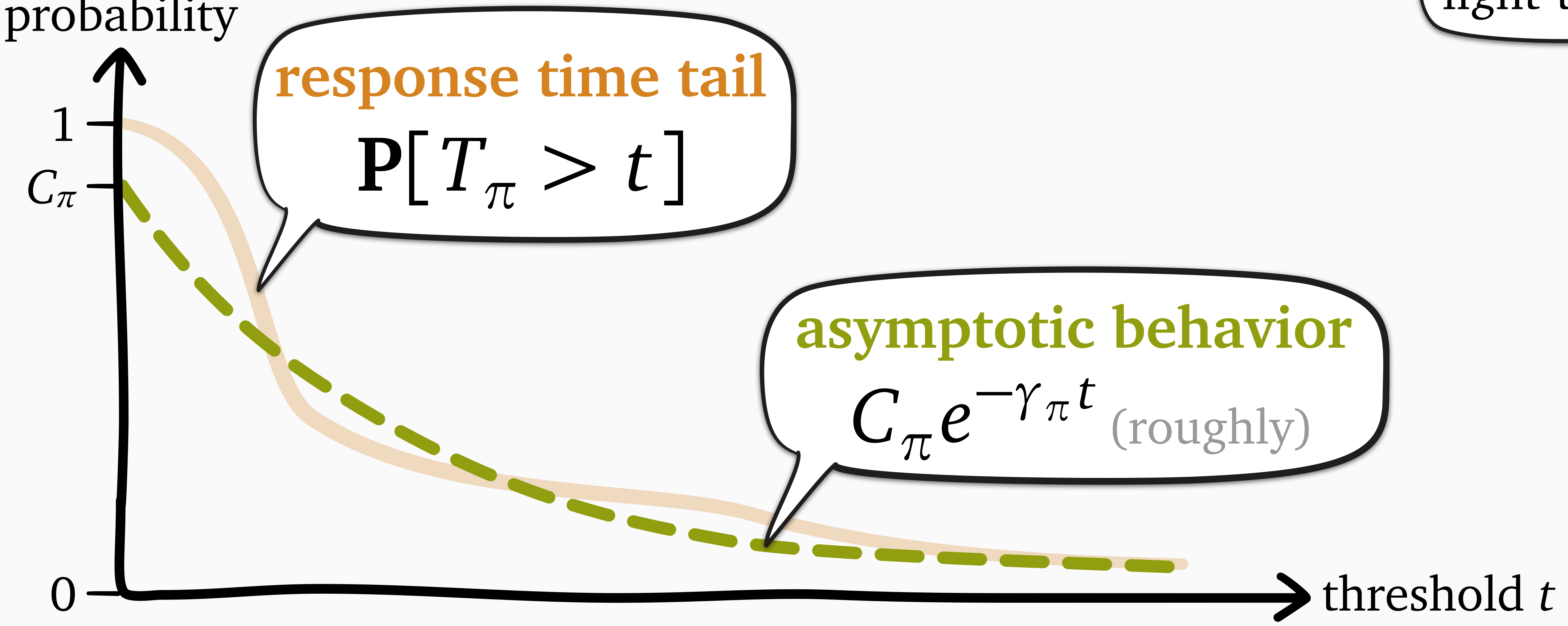
$C_\pi = \textit{tail constant}$ of π

Weak optimality:
optimal γ_π

Asymptotic response time tail

depends on policy π

when S is light-tailed



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

$C_\pi = \textit{tail constant}$ of π

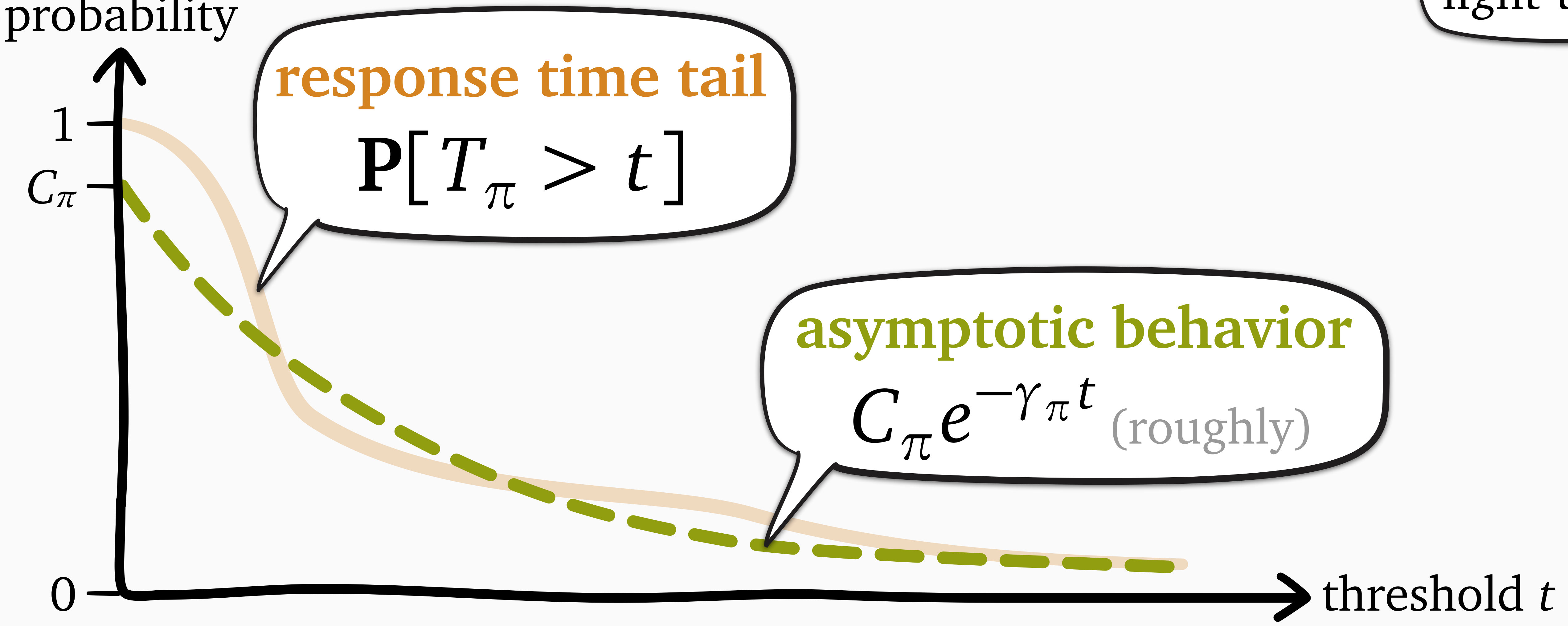
Weak optimality:
optimal γ_π

Strong optimality:
optimal γ_π and C_π

Asymptotic response time tail

depends on policy π

when S is light-tailed



γ_π = *decay rate* of π

C_π = *tail constant* of π

Weak optimality:
optimal γ_π

Strong optimality:
optimal γ_π and C_π

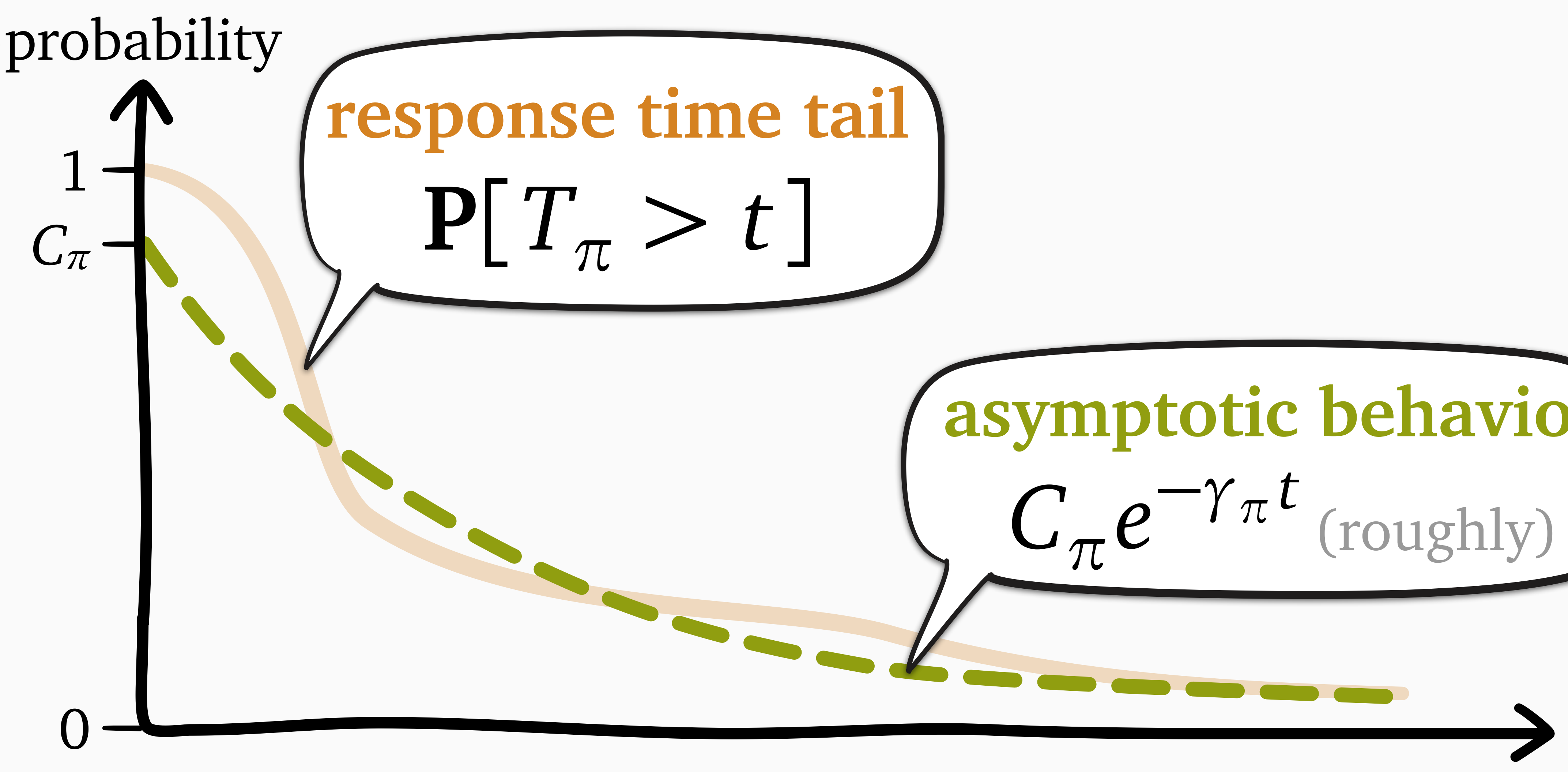
(roughly)

Asymptotic response time tail

depends on policy π

when S is light-tailed

Heavy-tailed S: SRPT and others are strongly optimal



response time tail
$\mathbf{P}[T_\pi > t]$

asymptotic behavior
$C_\pi e^{-\gamma_\pi t}$ (roughly)

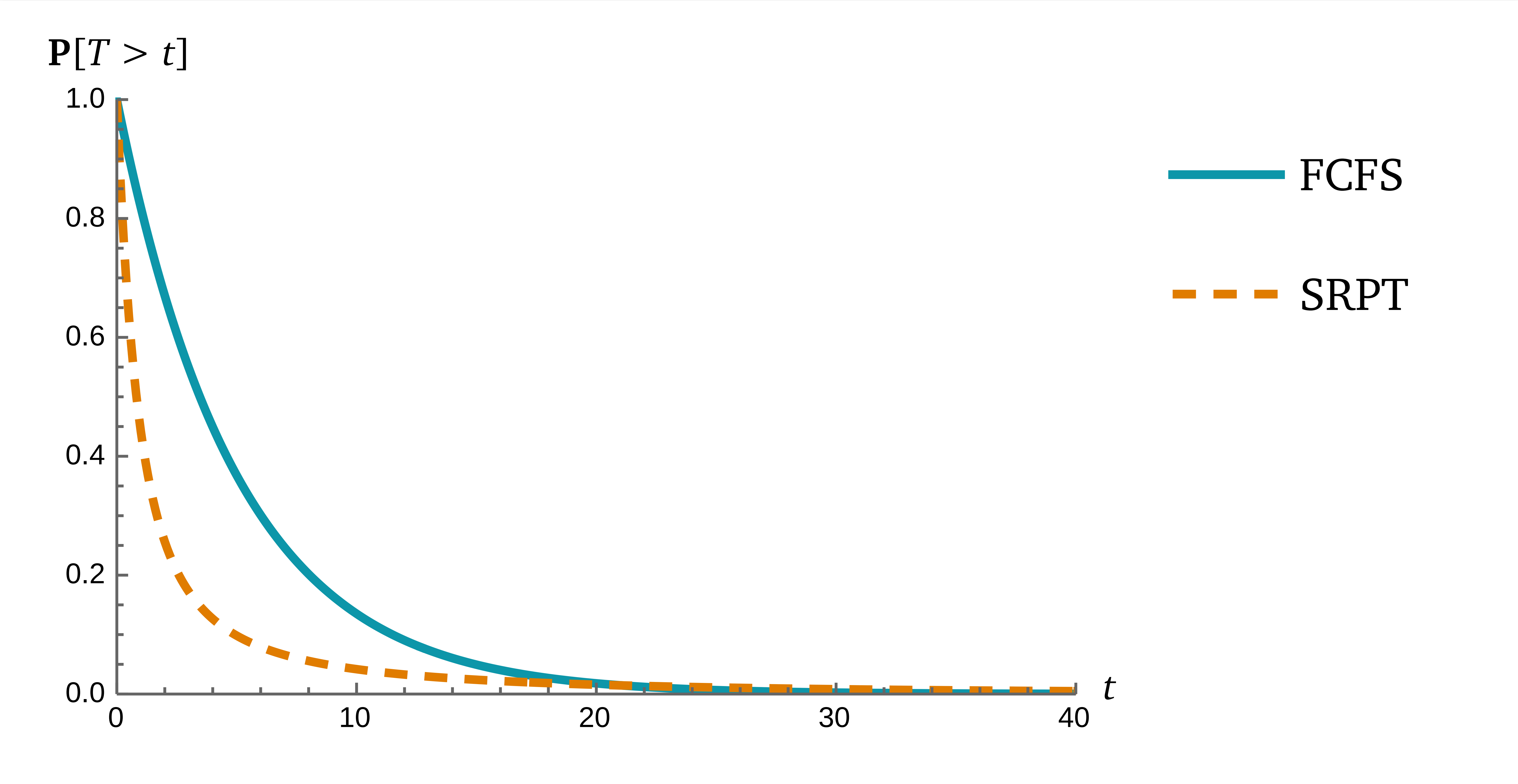
$\gamma_\pi = \textit{decay rate}$ of π
$C_\pi = \textit{tail constant}$ of π

Weak optimality:
optimal γ_π

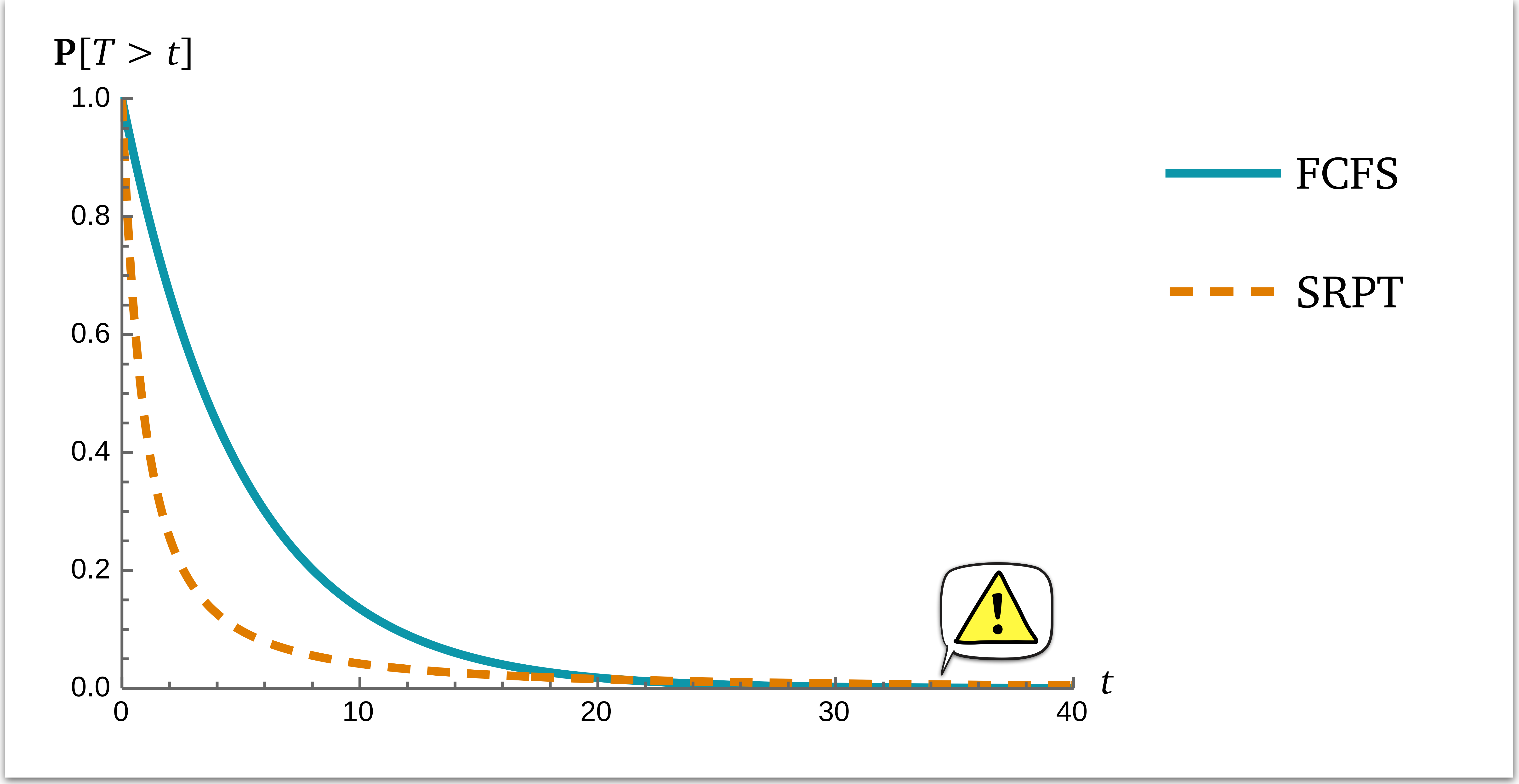
Strong optimality:
optimal γ_π and C_π

(roughly)

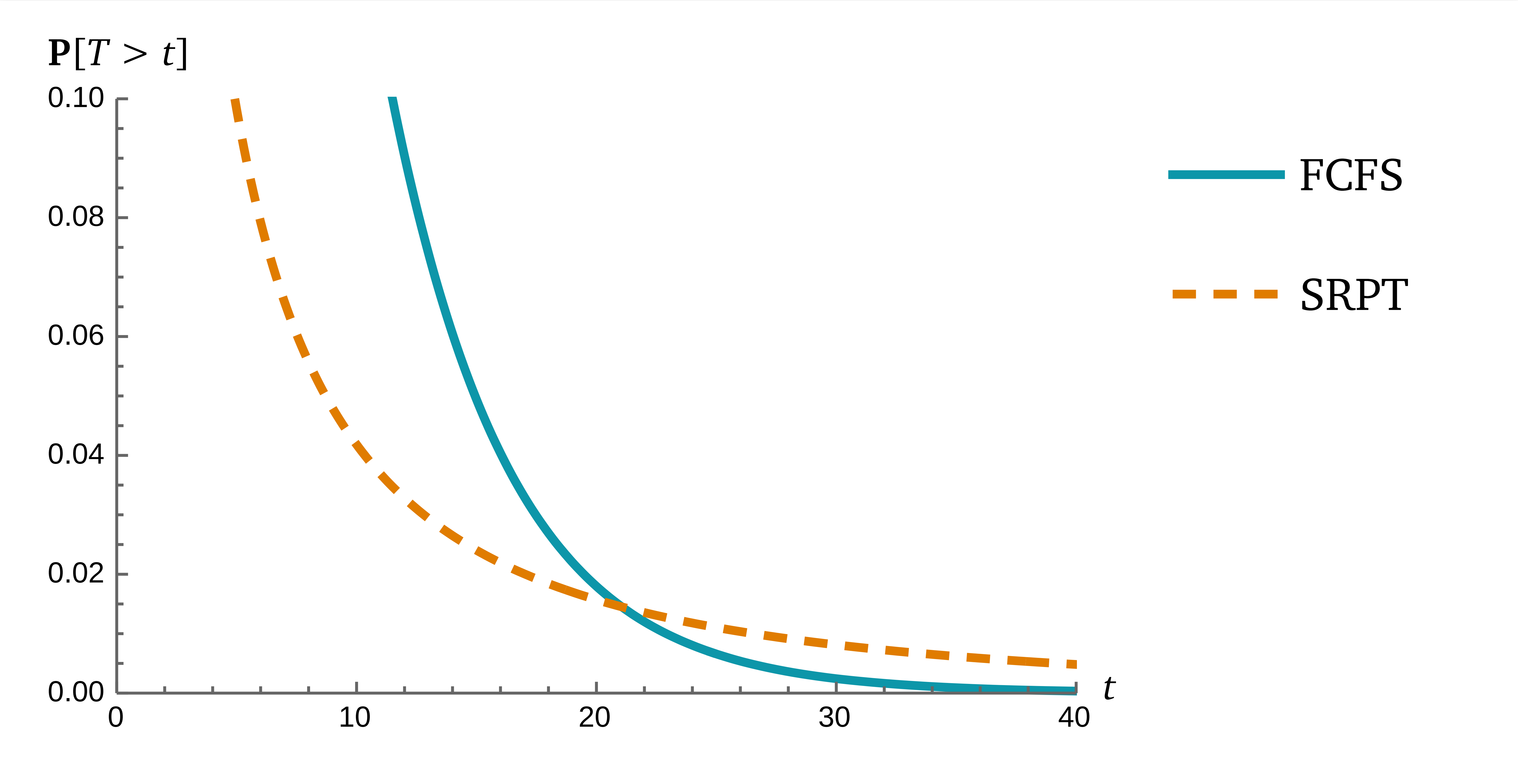
Optimizing the decay rate γ



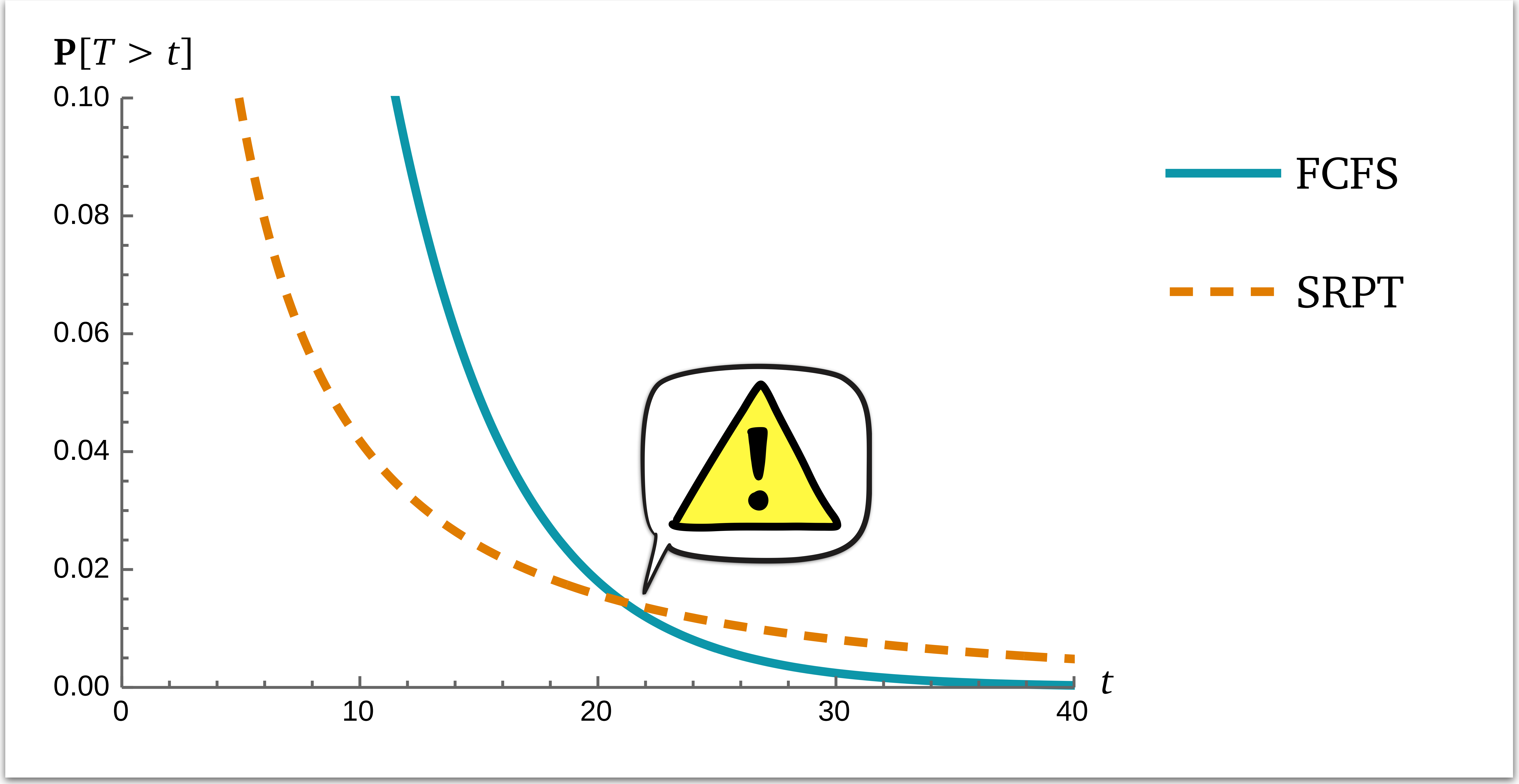
Optimizing the decay rate γ



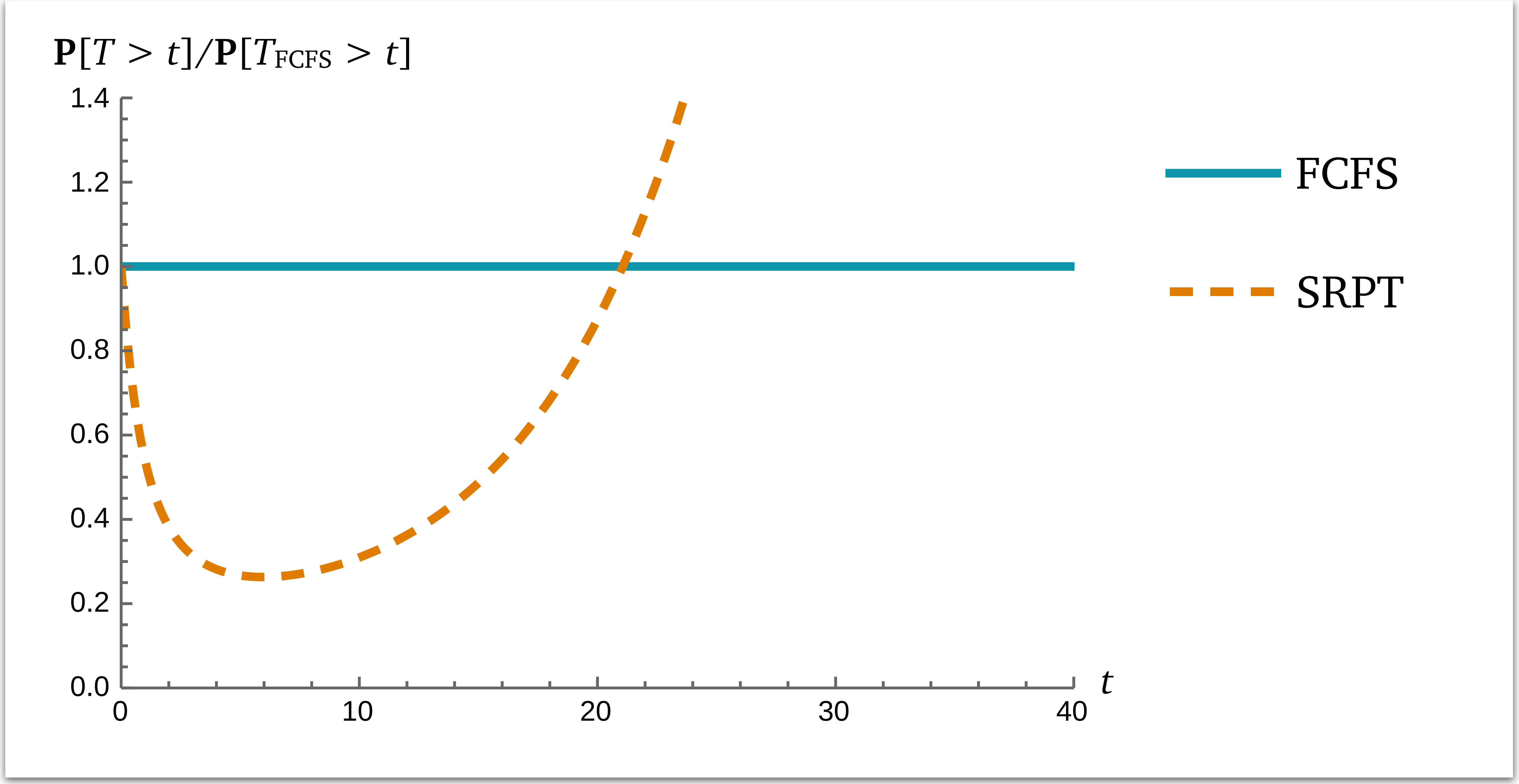
Optimizing the decay rate γ



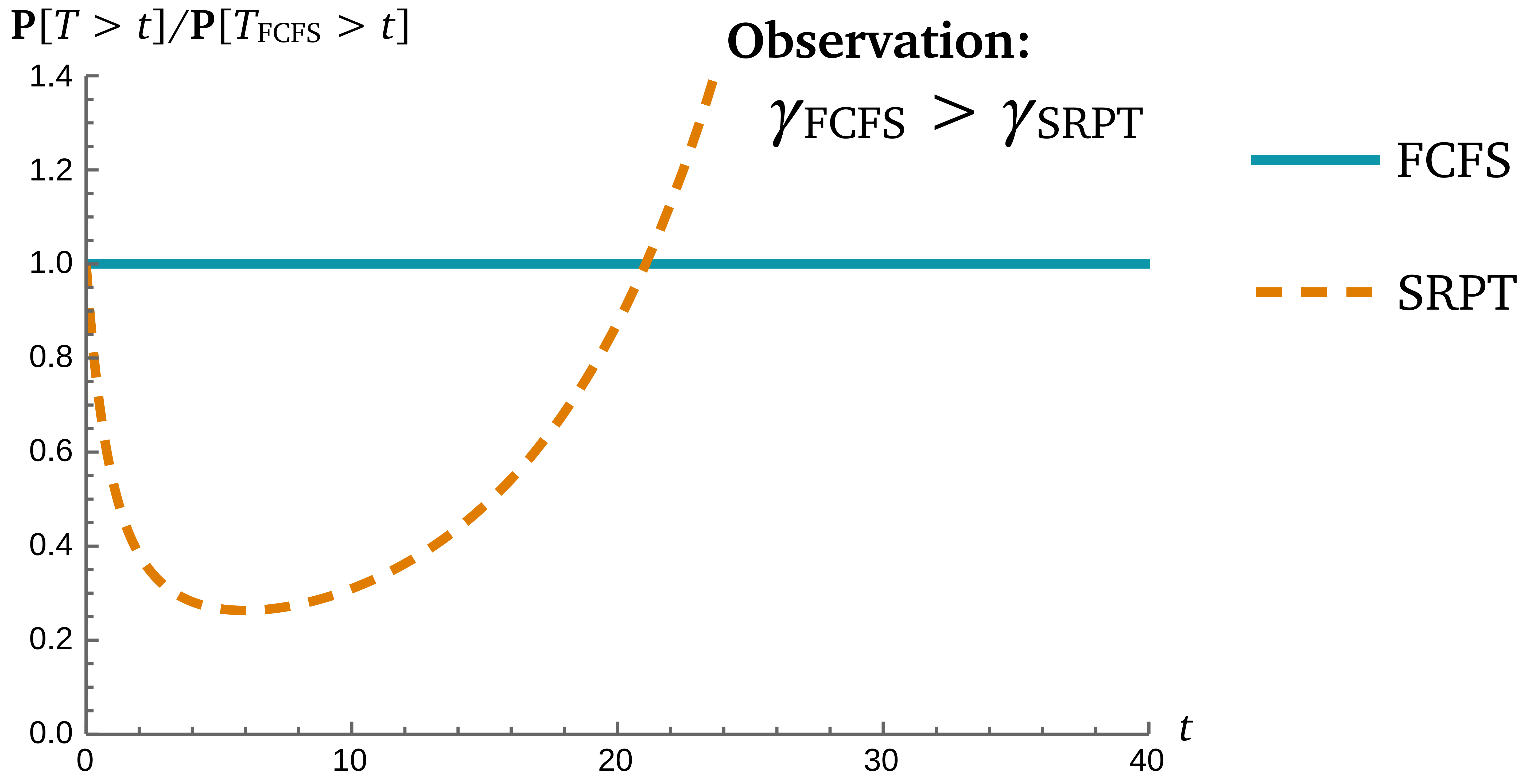
Optimizing the decay rate γ



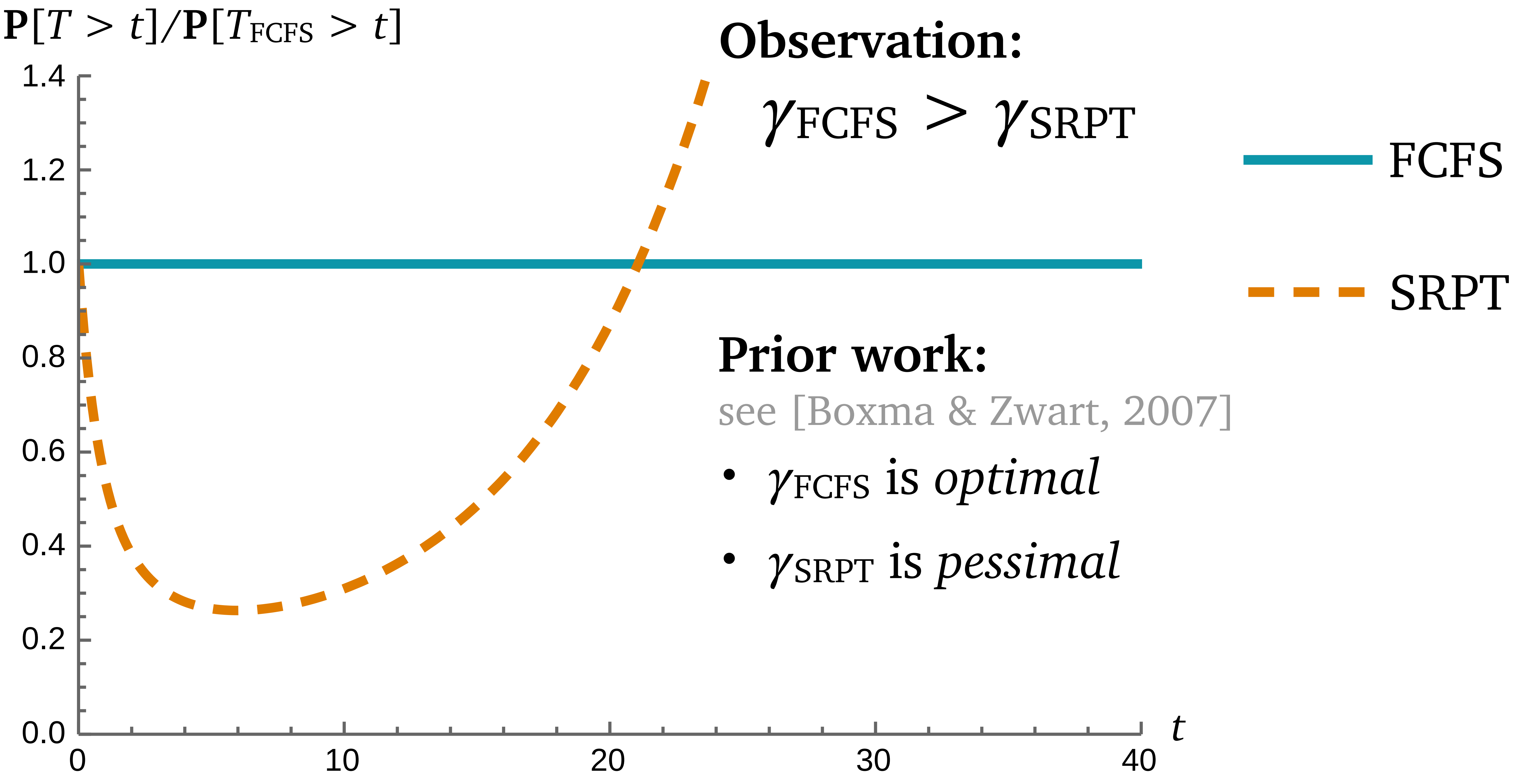
Optimizing the decay rate γ



Optimizing the decay rate γ



Optimizing the decay rate γ



Observation:

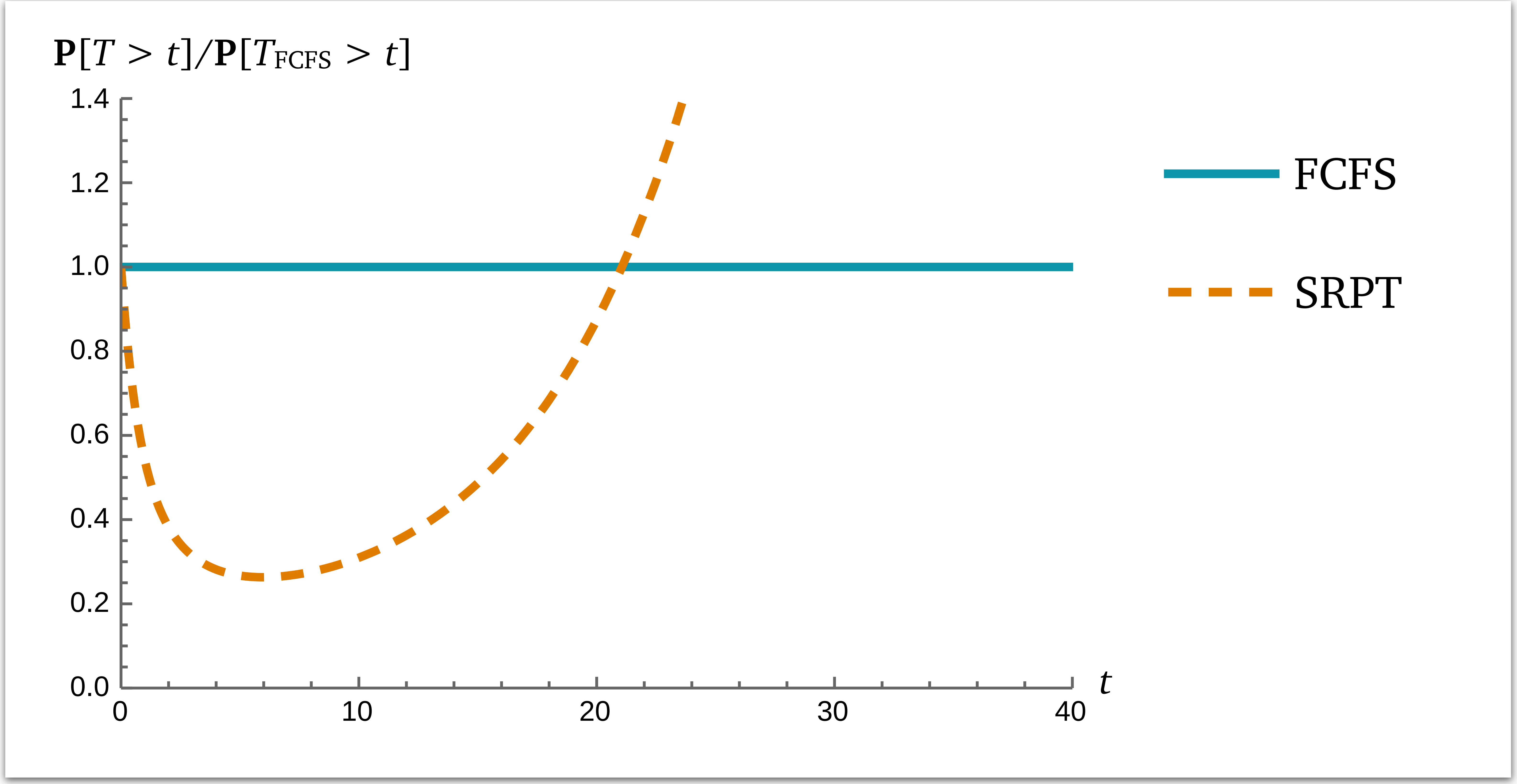
$\gamma_{\text{FCFS}} > \gamma_{\text{SRPT}}$

Prior work:

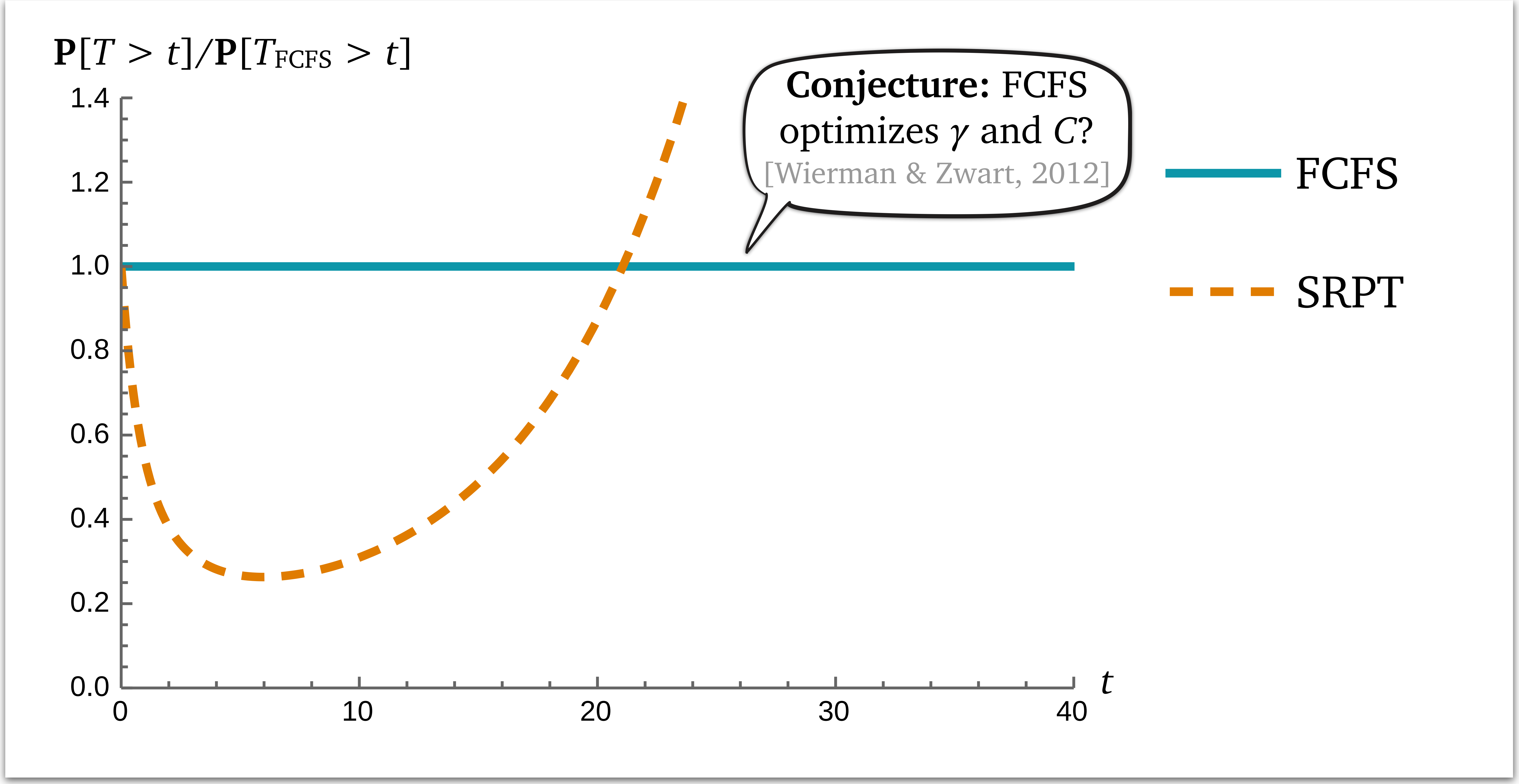
see [Boxma & Zwart, 2007]

- γ_{FCFS} is *optimal*
- γ_{SRPT} is *pessimal*

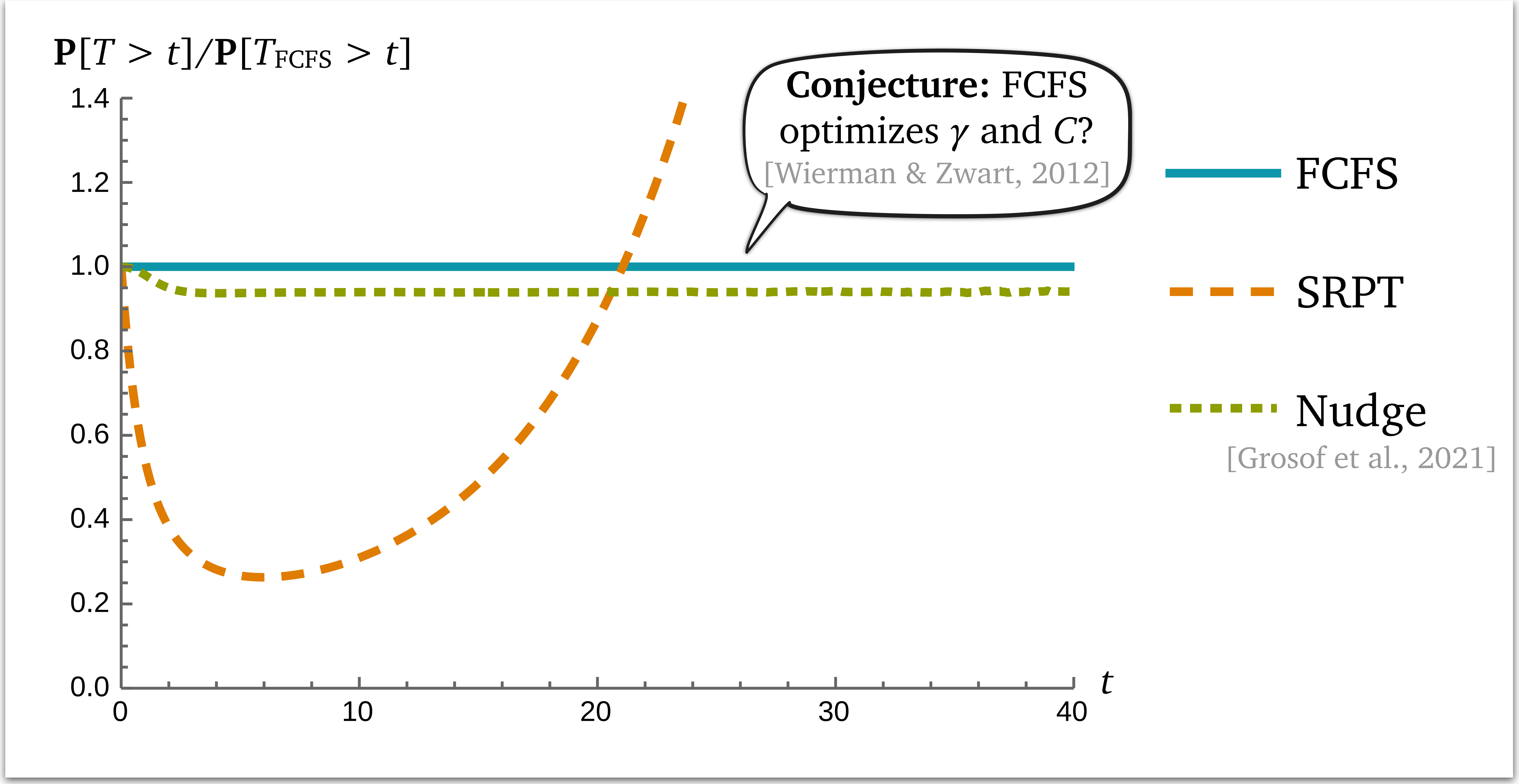
Optimizing the tail constant C



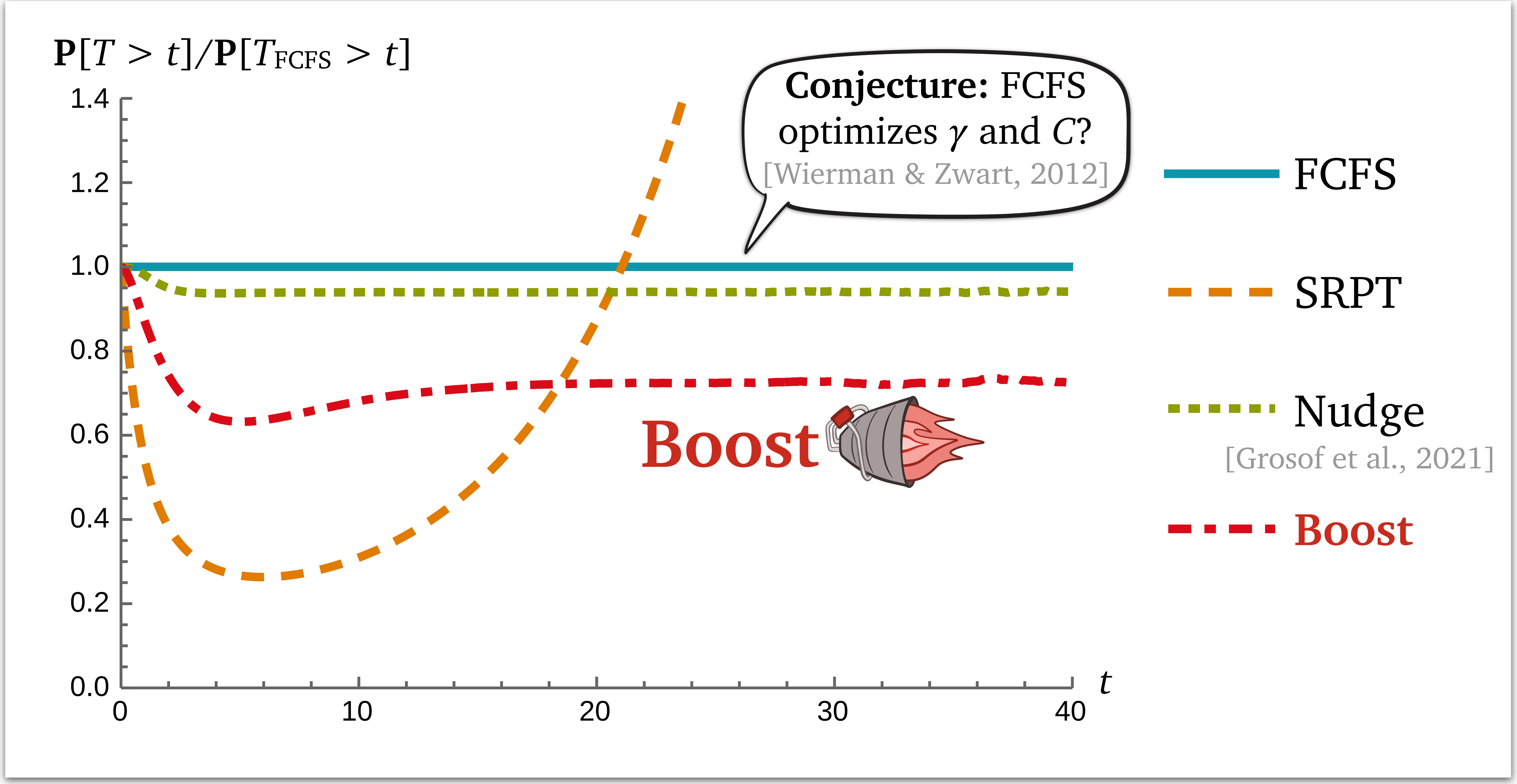
Optimizing the tail constant C



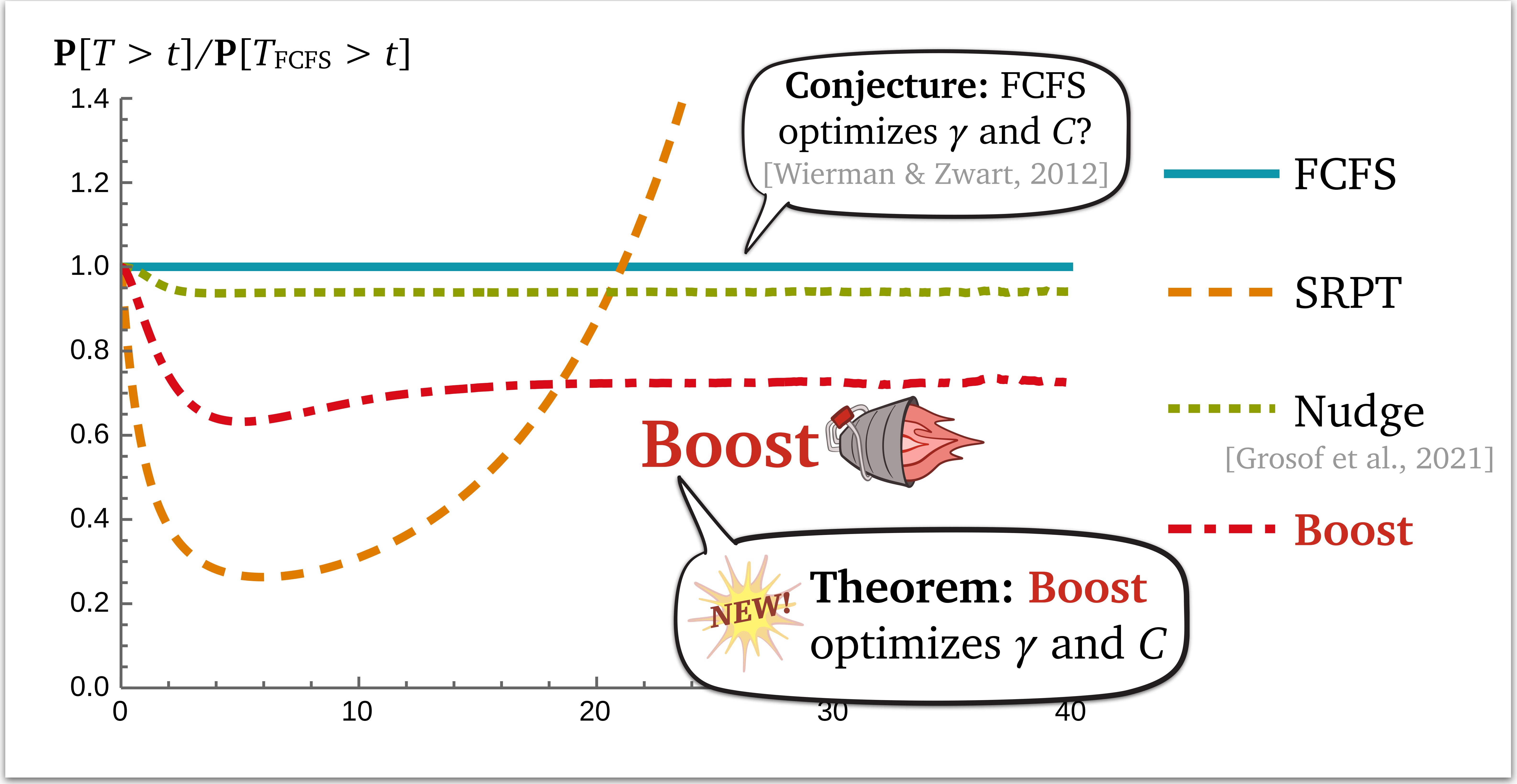
Optimizing the tail constant C



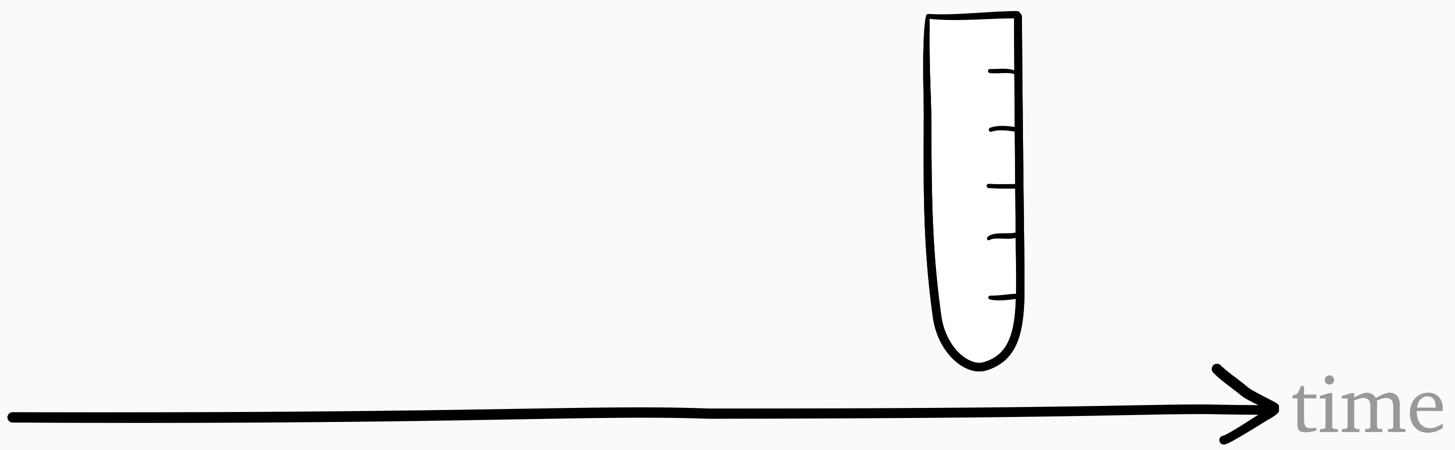
Optimizing the tail constant C



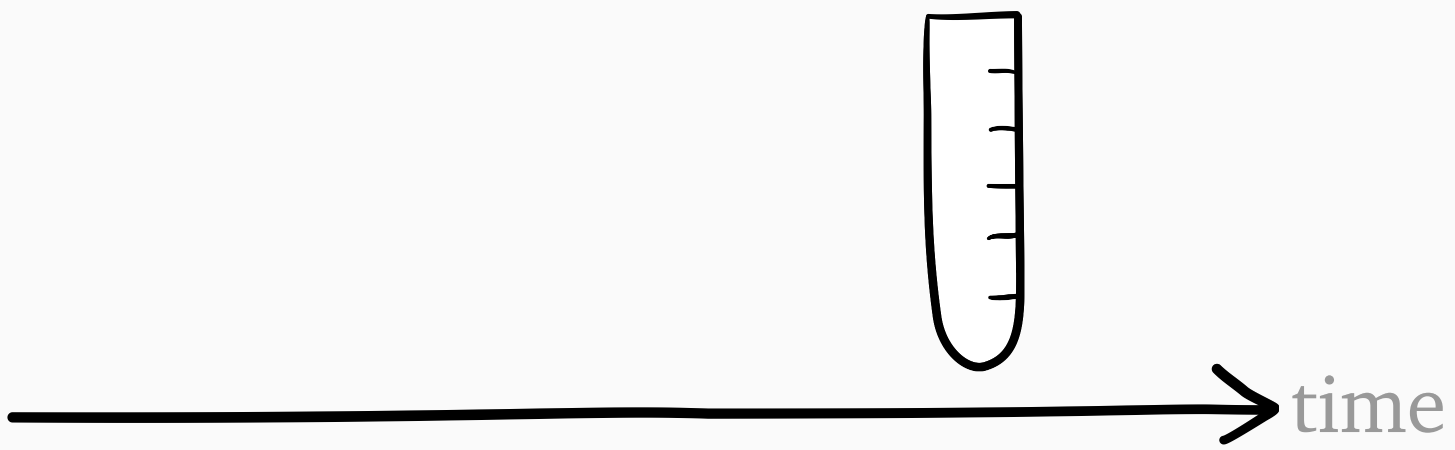
Optimizing the tail constant C



How **Boost** works

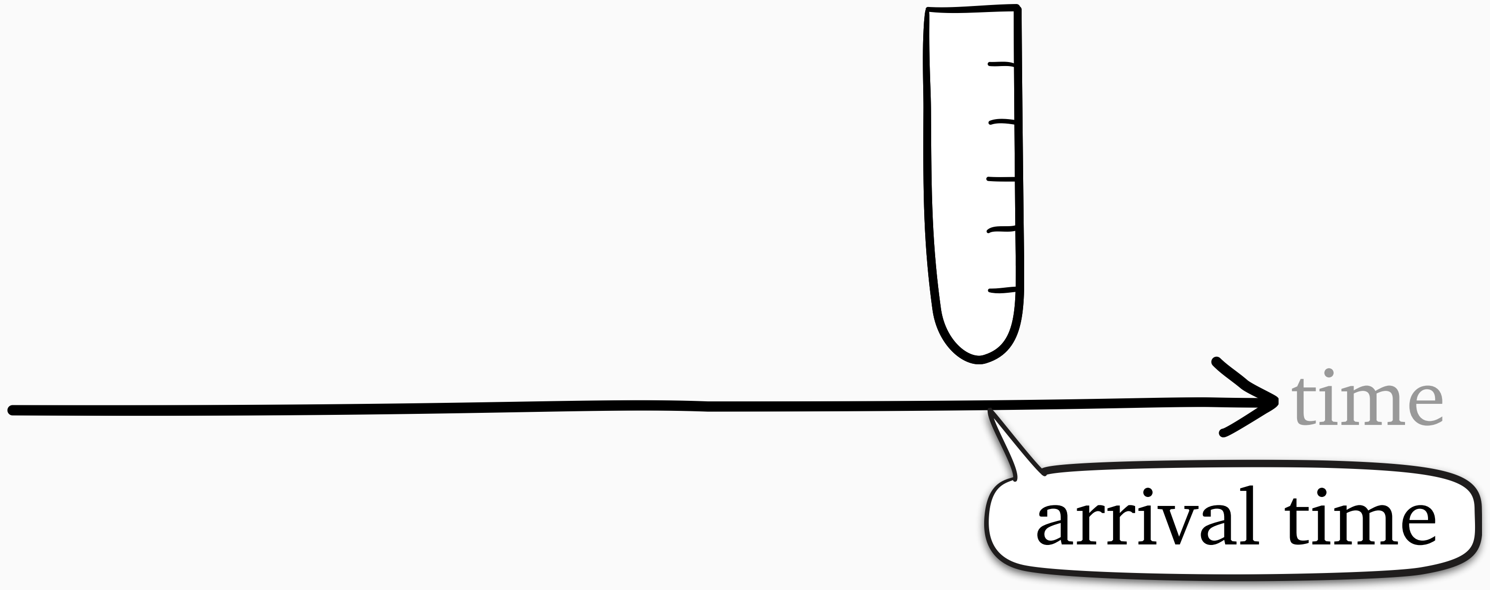


How **Boost** works



boosted arrival time
$= \text{arrival time} - \boldsymbol{b}(\text{size})$

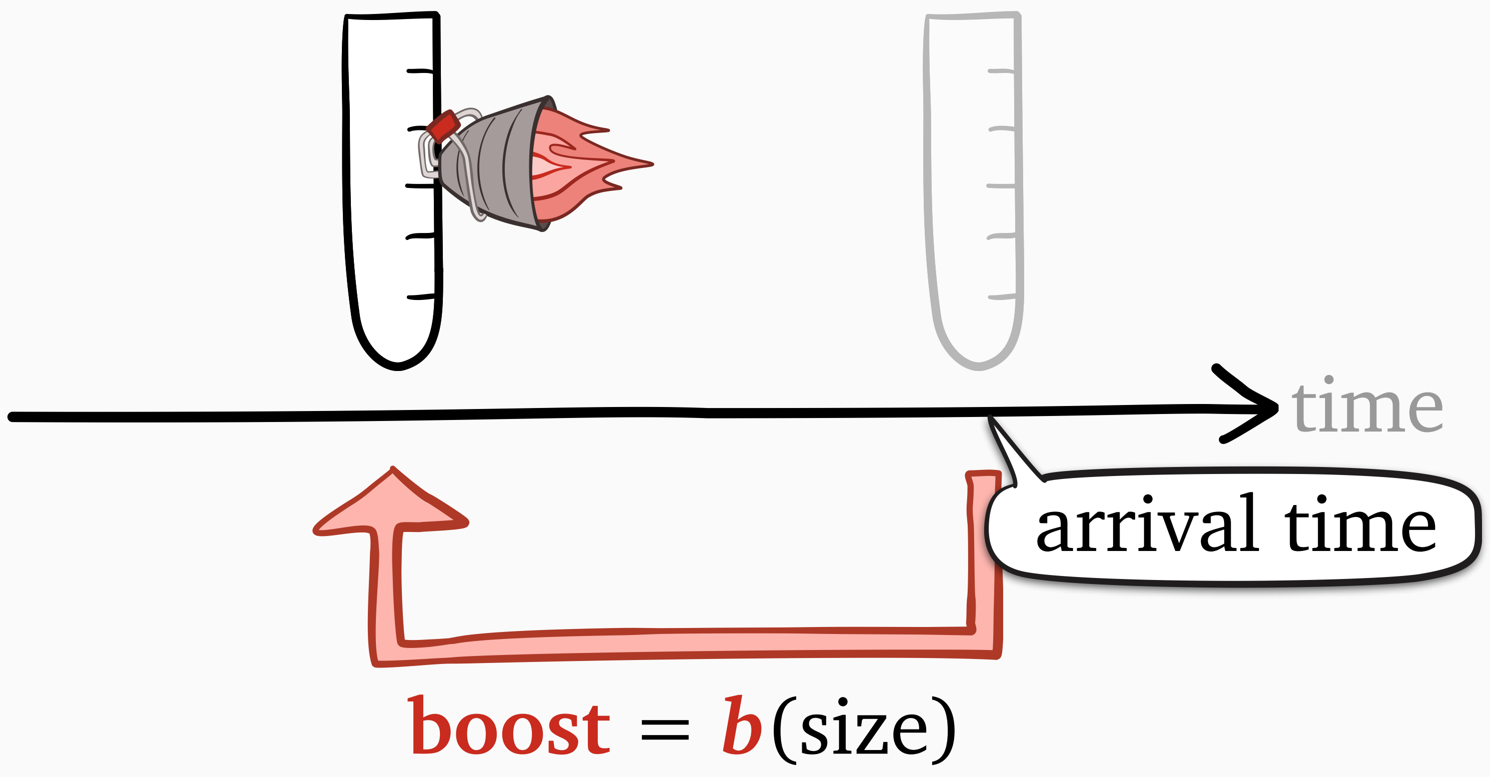
How **Boost** works



boosted arrival time

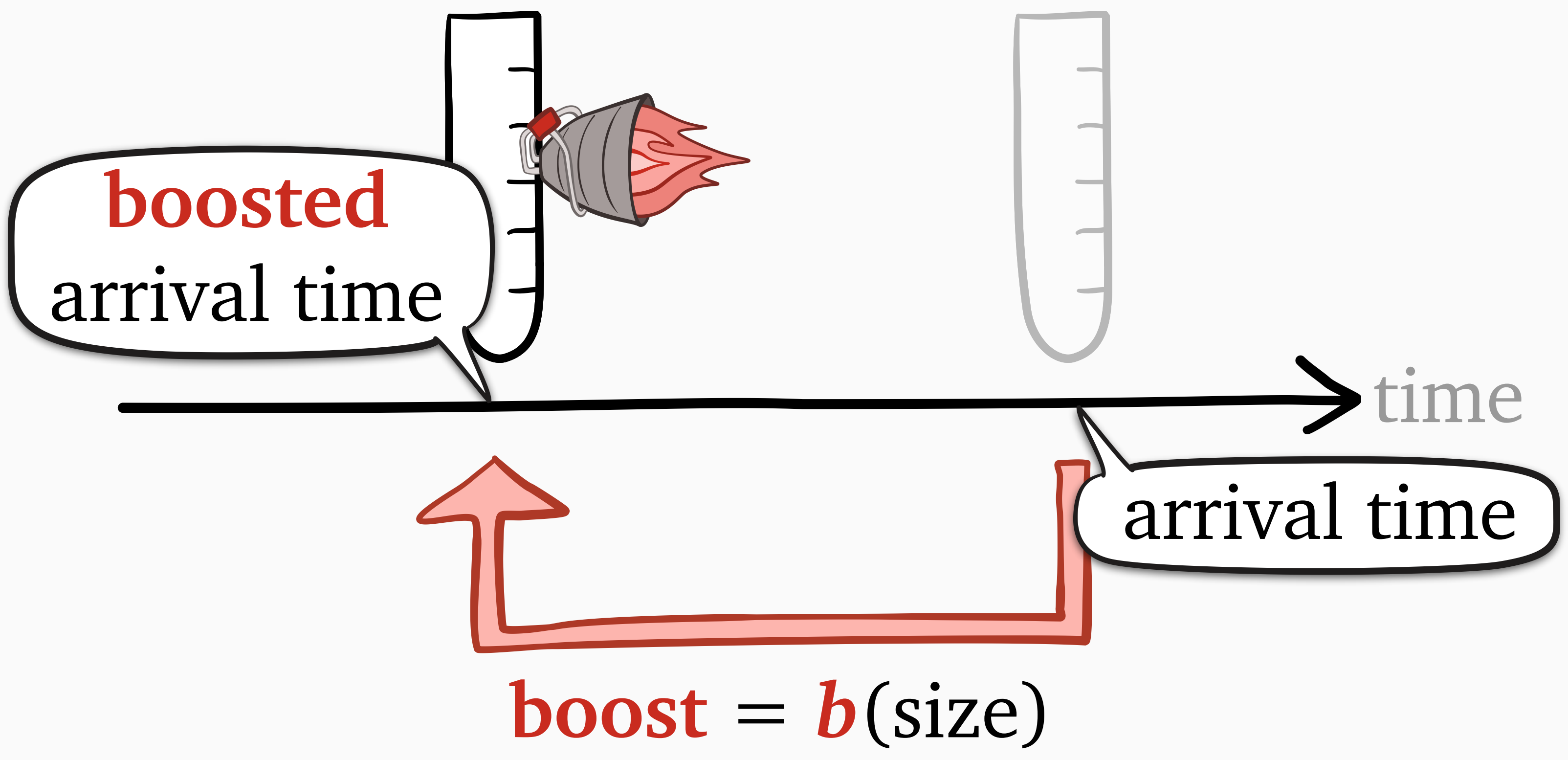
$= \text{arrival time} - \boldsymbol{b}(\text{size})$

How **Boost** works



boosted arrival time
= arrival time − ***b***(size)

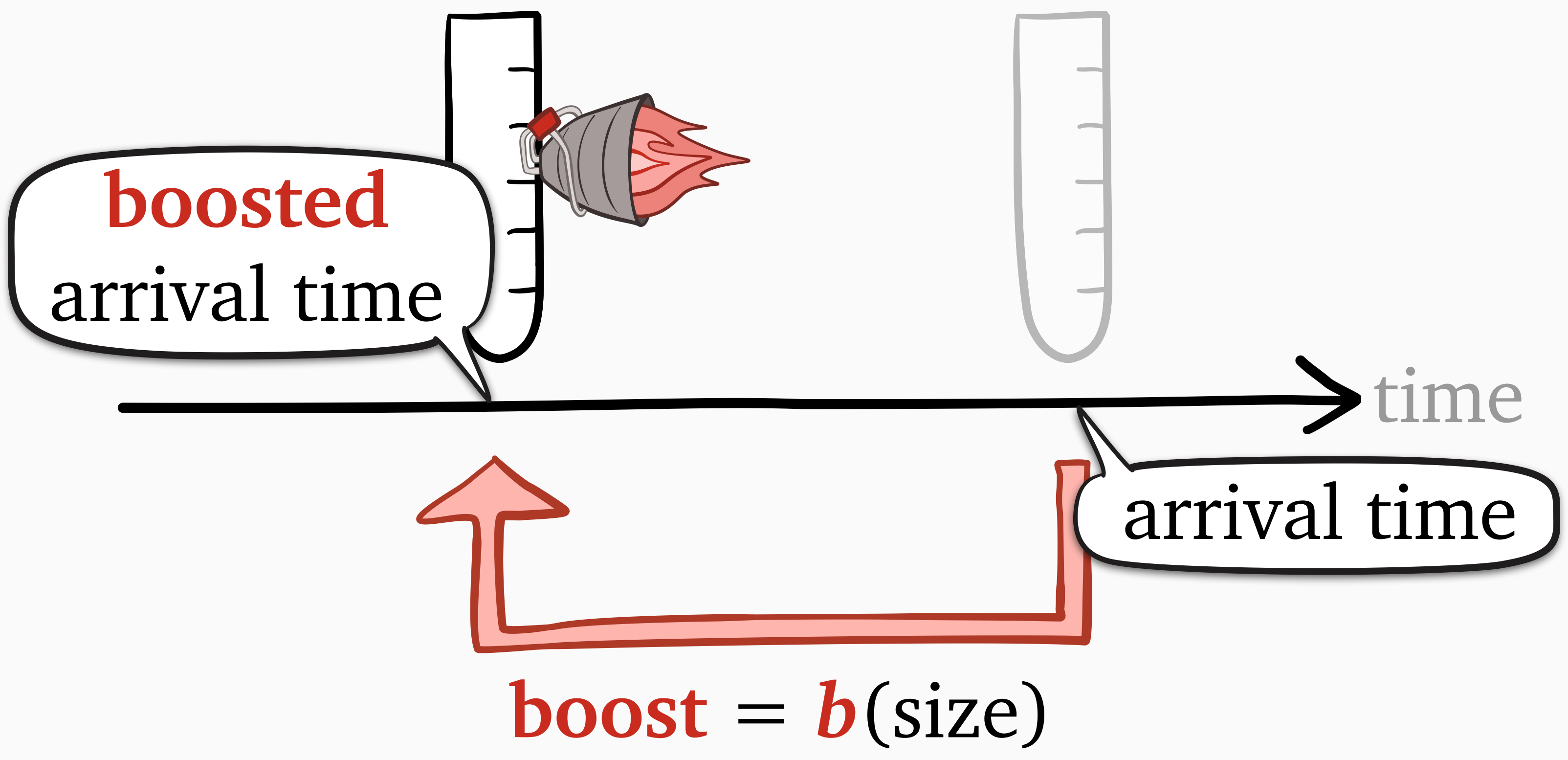
How **Boost** works



boosted arrival time

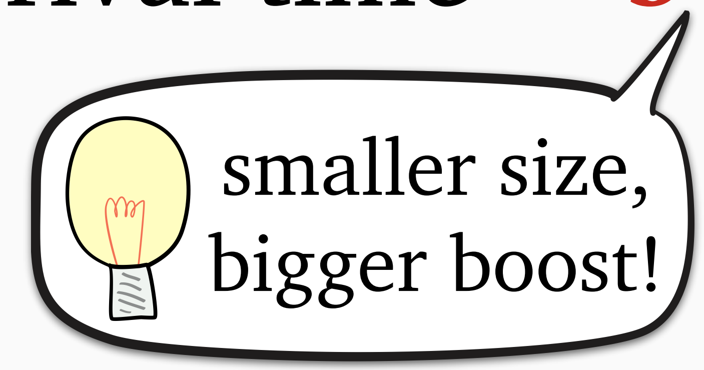
$= \text{arrival time} - b(\text{size})$

How **Boost** works

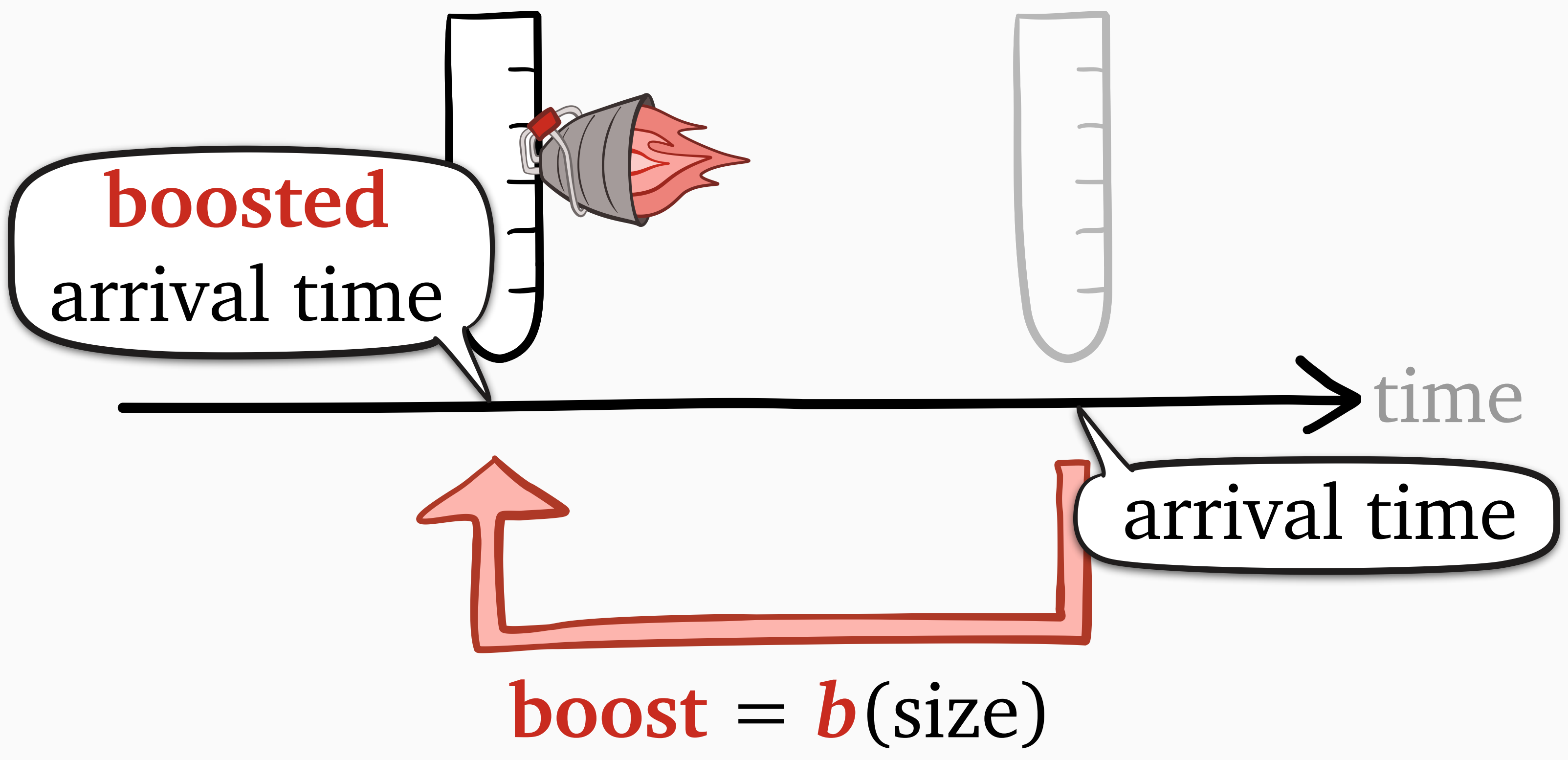


boosted arrival time

$$= \text{arrival time} - b(\text{size})$$



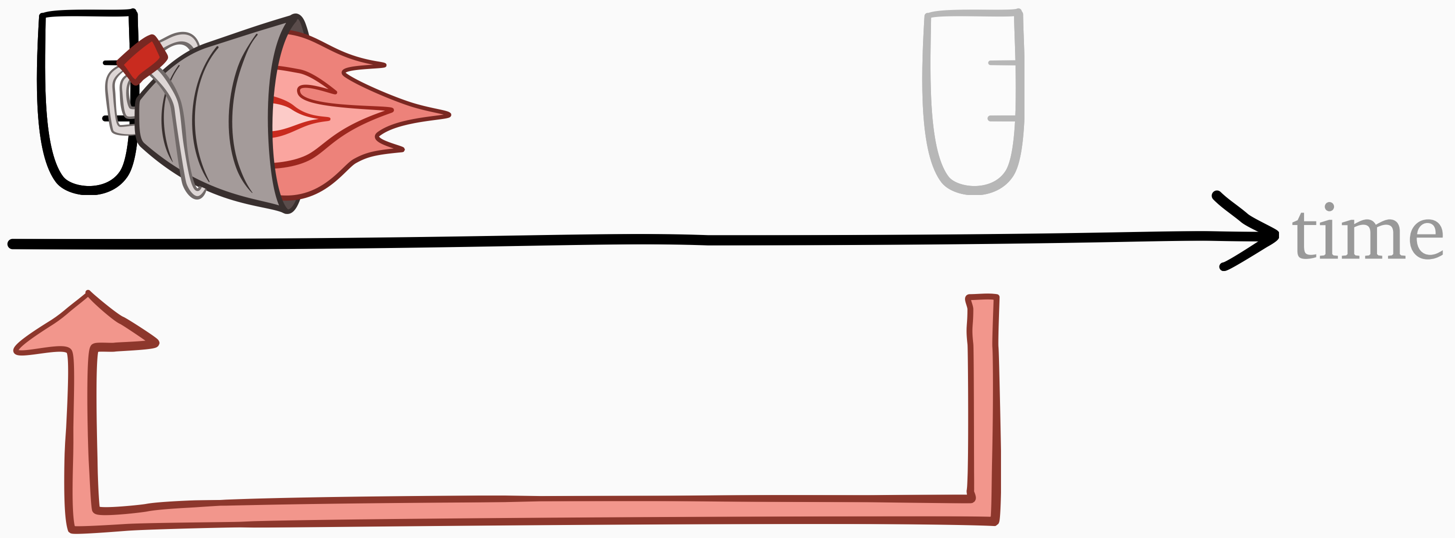
How **Boost** works



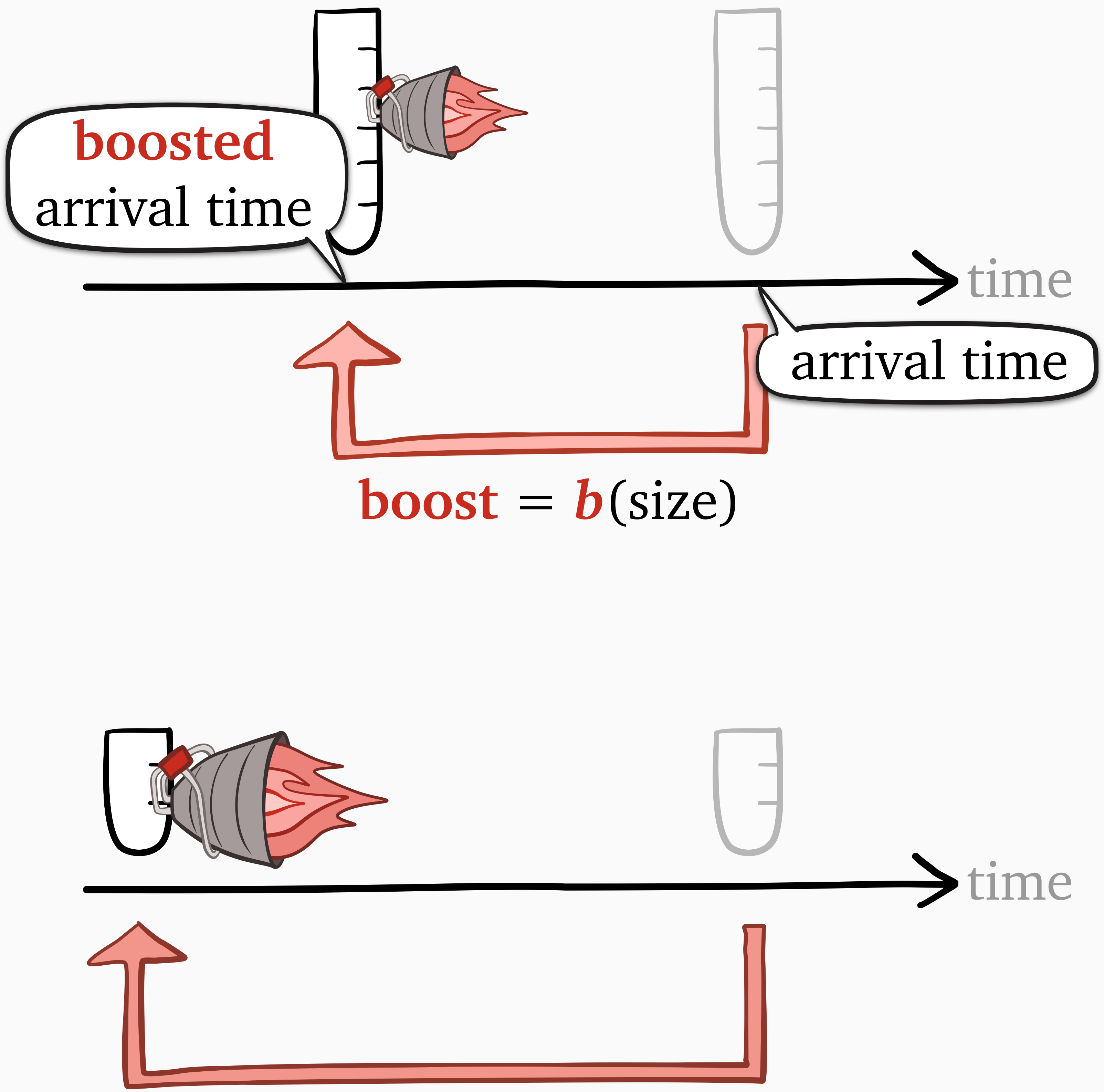
boosted arrival time

$= \text{arrival time} - b(\text{size})$

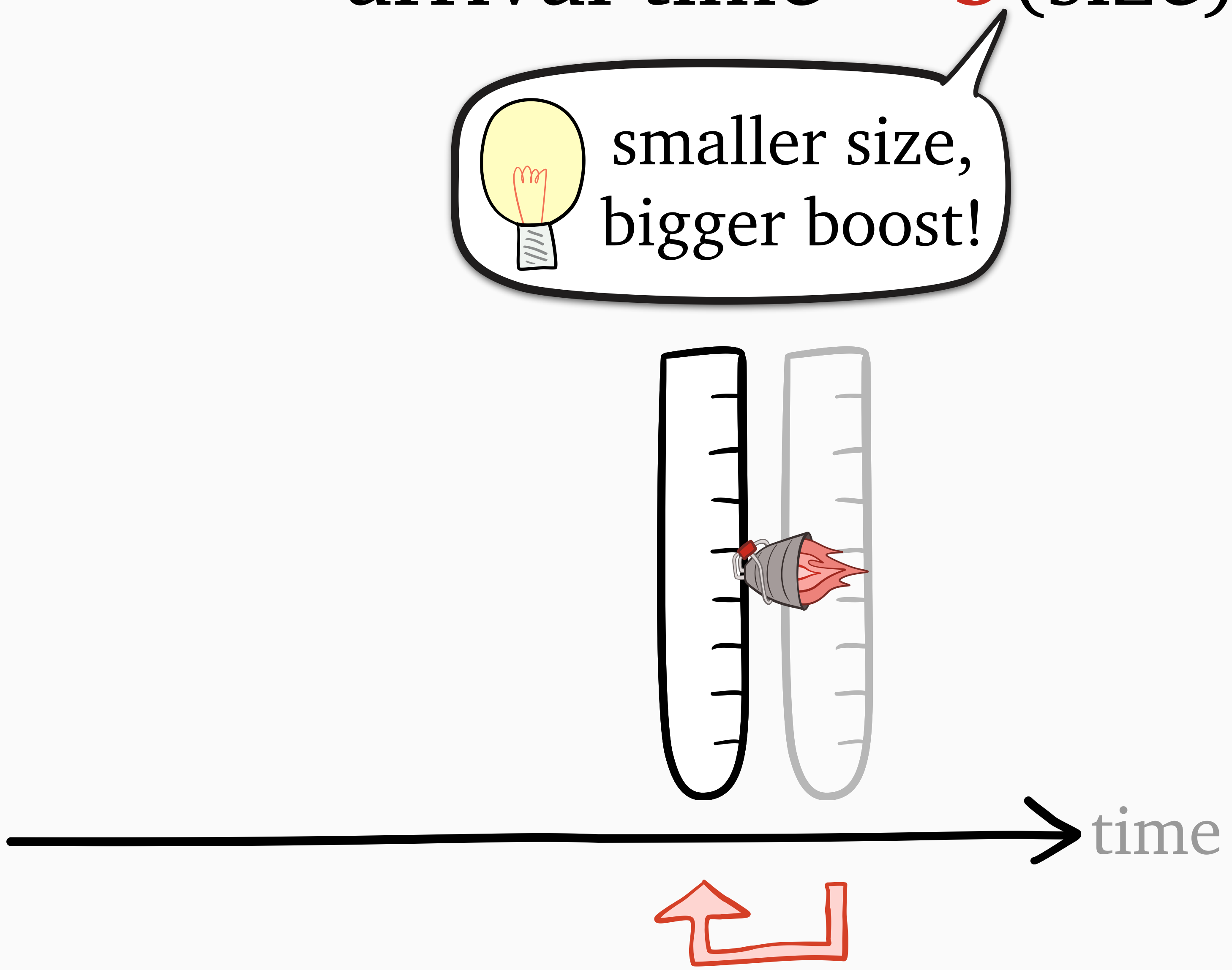
smaller size, bigger boost!



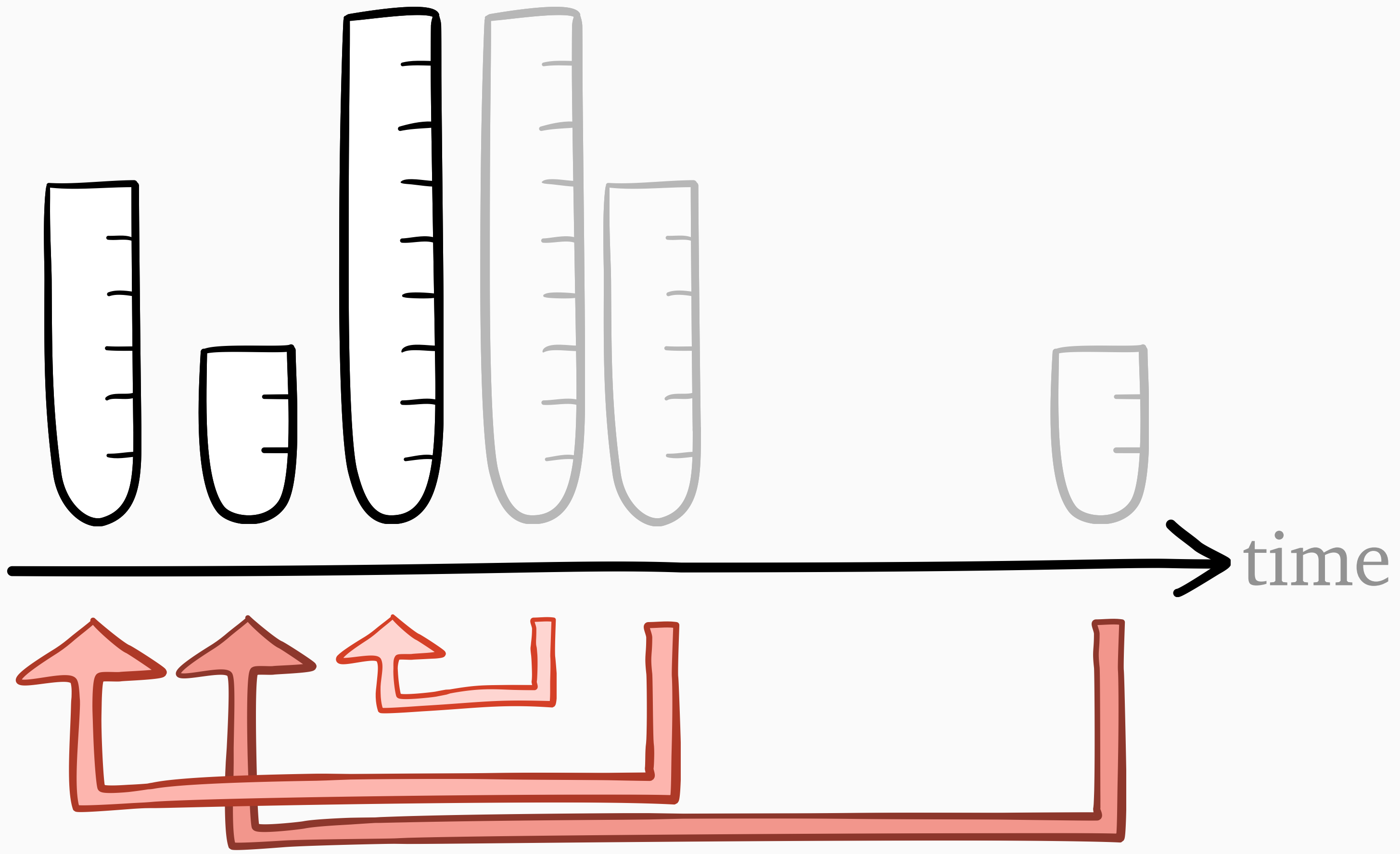
How **Boost** works



boosted arrival time
$= \text{arrival time} - b(\text{size})$



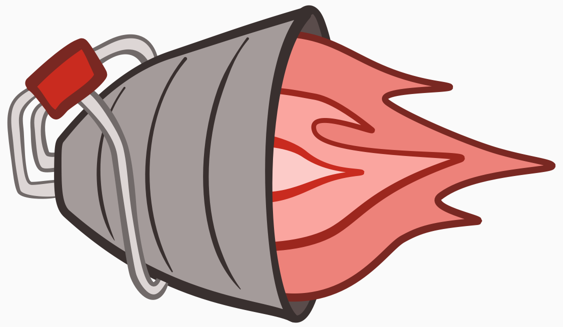
How **Boost** works



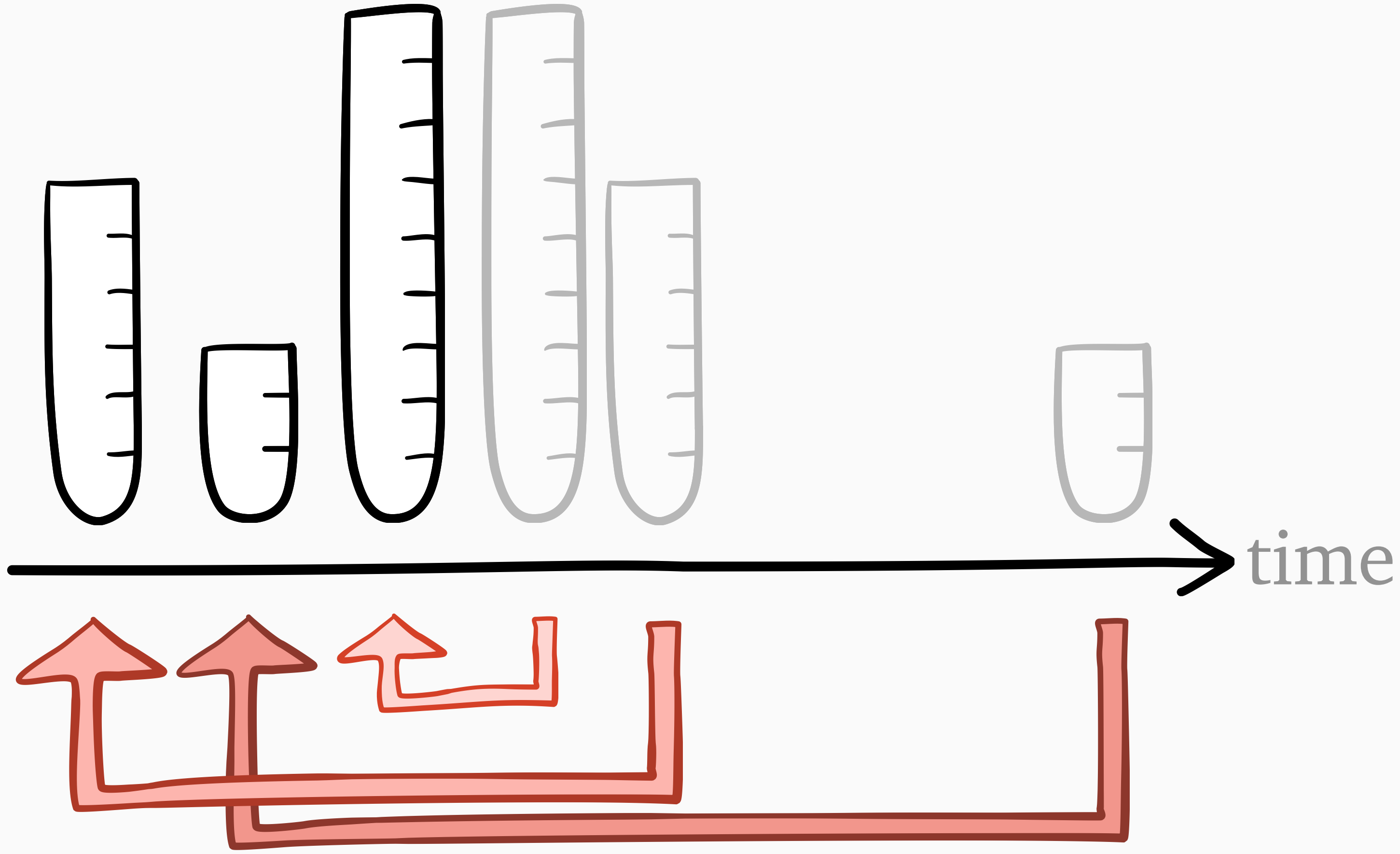
boosted arrival time

$= \text{arrival time} - \boldsymbol{b}(\text{size})$

How **Boost** works



Scheduling rule: always serve job of *minimum **boosted** arrival time*

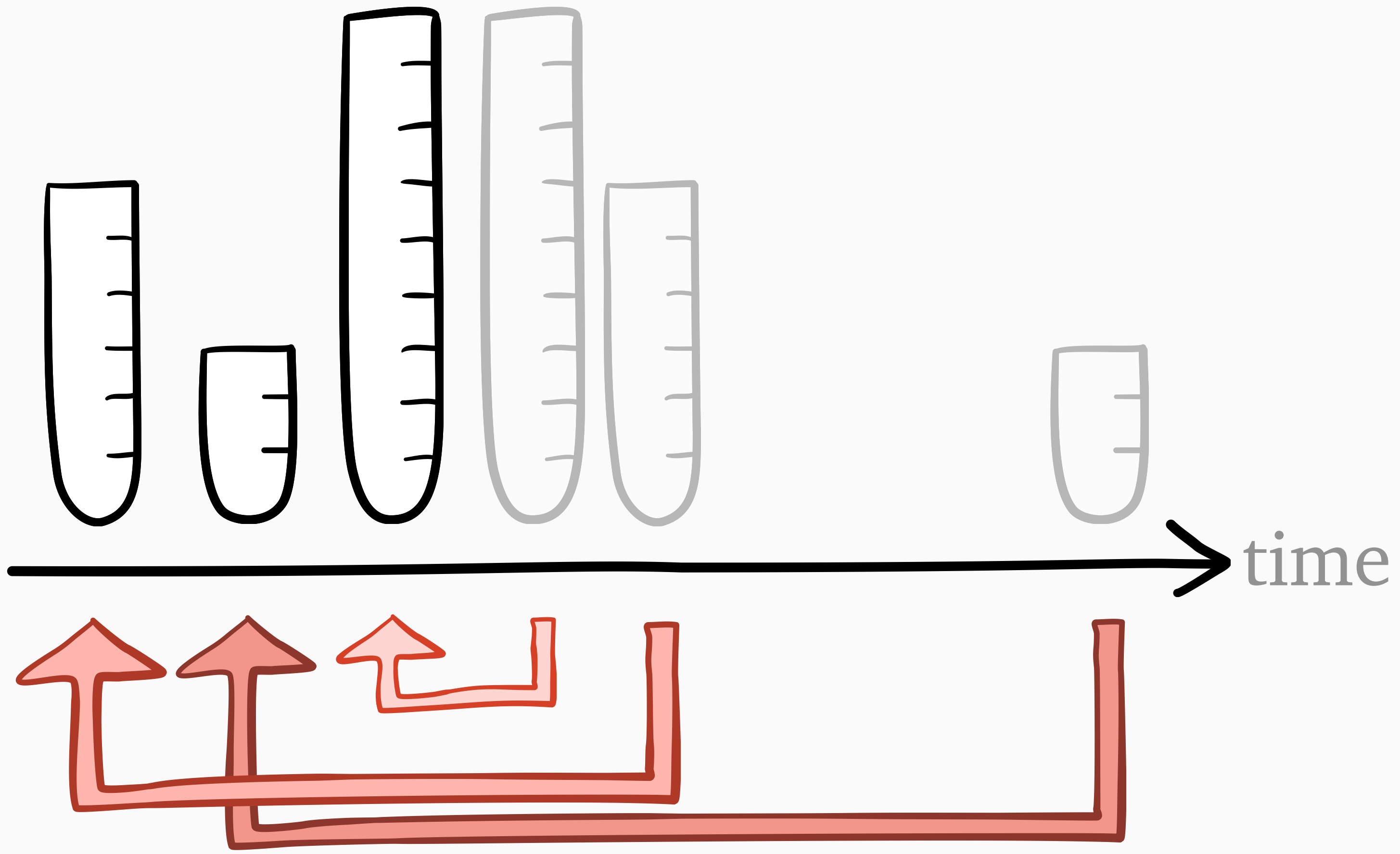


boosted arrival time

$= \text{arrival time} - \boldsymbol{b}(\text{size})$

How **Boost** works

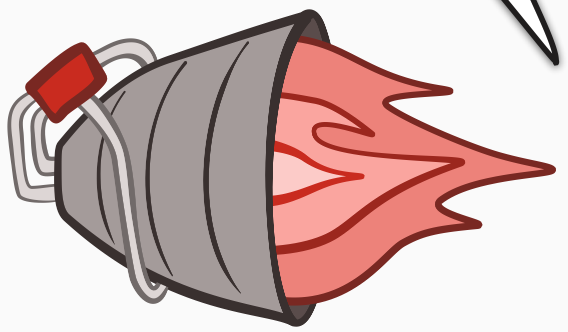
Scheduling rule: always serve job of
*minimum **boosted** arrival time*



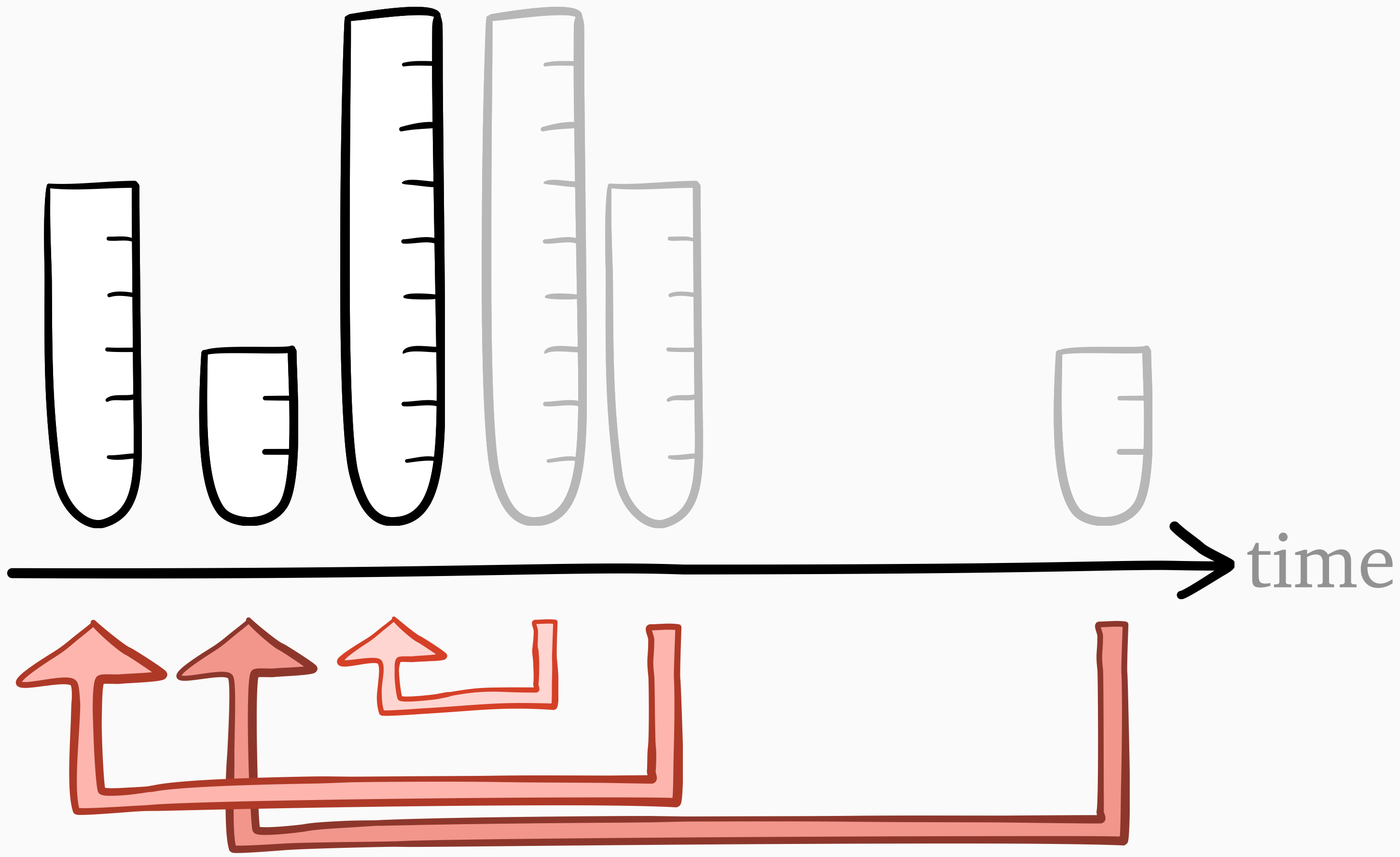
boosted arrival time
= arrival time − ***b***(size)

How **Boost** works

can be preemptive
or nonpreemptive



Scheduling rule: always serve job of
*minimum **boosted** arrival time*



boosted arrival time
$= \text{arrival time} - \boldsymbol{b}(\text{size})$

What's the right
boost function?

Queueing problem

$$\text{minimize} \quad C = \lim_{t \to \infty} e^{\gamma t} \mathbf{P}[T > t]$$

Queueing problem

$$\text{minimize} \quad C = \lim_{t \to \infty} e^{\gamma t} \mathbf{P}[T > t]$$

Batch problem

$$t_i = d_i - a_i$$

$$a_i = \text{arrival time of job } i$$

$$d_i = \text{departure time of job } i$$

Queueing problem

$$\text{minimize}\quad C = \underbrace{\lim_{t\to\infty} e^{\gamma t}\mathbf{P}[T > t]}_{\text{“}\infty\cdot\mathbf{P}[T>\infty]\text{”}}$$

Batch problem

$$t_i = d_i - a_i$$
$$a_i = \text{arrival time of job } i$$
$$d_i = \text{departure time of job } i$$

Queueing problem

$$\text{minimize} \;\; C = \underbrace{\lim_{t\to\infty} e^{\gamma t}\mathbf{P}[T > t]}_{\text{“}\infty \cdot \mathbf{P}[T > \infty]\text{”}}$$

Batch problem

$$\text{minimize} \;\; \mathbf{P}[T > \infty]$$

$$t_i = d_i - a_i$$
a_i = arrival time of job i
d_i = departure time of job i

Queueing problem

$$\text{minimize} \quad C = \underbrace{\lim_{t\to\infty} e^{\gamma t}\mathbf{P}[T > t]}_{\text{“}\infty \cdot \mathbf{P}[T > \infty]\text{”}}$$

Batch problem

$$\text{minimize} \quad \mathbf{P}[T > \infty] = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(t_i > \infty)$$

$t_i = d_i - a_i$

$a_i =$ arrival time of job i

$d_i =$ departure time of job i

Queueing problem

$$\text{minimize} \quad C = \underbrace{\lim_{t\to\infty} e^{\gamma t}\mathbf{P}[T > t]}_{\text{“}\infty \cdot \mathbf{P}[T > \infty]\text{”}}$$

Batch problem

$$\text{minimize} \quad \mathbf{P}[T > \infty] = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(t_i > \infty) = 0$$

$t_i = d_i - a_i$

a_i = arrival time of job i

d_i = departure time of job i

Queueing problem

$$\text{minimize}\quad C = \underbrace{\lim_{t\to\infty} e^{\gamma t}\mathbf{P}[T > t]}_{\text{“}\infty \cdot \mathbf{P}[T > \infty]\text{”}}$$

Batch problem

$$\text{minimize}\quad \mathbf{P}[T > \infty] = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(t_i > \infty) = 0$$

$t_i = d_i - a_i$

a_i = arrival time of job i

d_i = departure time of job i

Queueing problem

$$\text{minimize} \quad C = \underbrace{\lim_{t\to\infty} e^{\gamma t}\mathbf{P}[T > t]}_{\text{“}\infty \cdot \mathbf{P}[T > \infty]\text{”}} = \lim_{\theta\to\gamma} \frac{\gamma-\theta}{\gamma}\mathbf{E}[e^{\theta T}]$$

(by final value theorem
for Laplace transforms)

Batch problem

$$\text{minimize} \quad \mathbf{P}[T > \infty] = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(t_i > \infty) = 0$$

$t_i = d_i - a_i$

$a_i =$ arrival time of job i

$d_i =$ departure time of job i

Queueing problem

$$\text{minimize} \;\; C = \underbrace{\lim_{t\to\infty} e^{\gamma t}\mathbf{P}[T > t]}_{\text{“}\infty \cdot \mathbf{P}[T > \infty]\text{”}} = \underbrace{\lim_{\theta\to\gamma} \frac{\gamma - \theta}{\gamma}\mathbf{E}[e^{\theta T}]}_{\text{“}0 \cdot \mathbf{E}[e^{\gamma T}]\text{”}}$$

(by final value theorem for Laplace transforms)

Batch problem

$$\text{minimize} \;\; \mathbf{P}[T > \infty] = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(t_i > \infty) = 0$$

$t_i = d_i - a_i$

$a_i =$ arrival time of job i

$d_i =$ departure time of job i

Queueing problem

$$\text{minimize}\quad C = \underbrace{\lim_{t\to\infty} e^{\gamma t}\mathbf{P}[T > t]}_{\text{“}\infty \cdot \mathbf{P}[T > \infty]\text{”}} = \underbrace{\lim_{\theta\to\gamma} \frac{\gamma - \theta}{\gamma}\mathbf{E}[e^{\theta T}]}_{\text{“}0 \cdot \mathbf{E}[e^{\gamma T}]\text{”}}$$

(by final value theorem for Laplace transforms)

Batch problem

$$\text{minimize}\quad \mathbf{P}[T > \infty] = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(t_i > \infty) = 0$$

$$t_i = d_i - a_i$$

a_i = arrival time of job i

d_i = departure time of job i

$$\text{minimize}\quad \mathbf{E}[e^{\gamma T}]$$

Queueing problem

$$\text{minimize} \quad C = \underbrace{\lim_{t\to\infty} e^{\gamma t}\mathbf{P}[T > t]}_{\text{“}\infty\cdot\mathbf{P}[T>\infty]\text{”}} = \underbrace{\lim_{\theta\to\gamma} \frac{\gamma-\theta}{\gamma}\mathbf{E}[e^{\theta T}]}_{\text{“}0\cdot\mathbf{E}[e^{\gamma T}]\text{”}}$$

(by final value theorem for Laplace transforms)

Batch problem

$$\text{minimize} \quad \mathbf{P}[T > \infty] = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(t_i > \infty) = 0$$

$$\text{minimize} \quad \mathbf{E}[e^{\gamma T}] = \frac{1}{n}\sum_{i=1}^{n} e^{\gamma t_i}$$

$t_i = d_i - a_i$

a_i = arrival time of job i

d_i = departure time of job i

Queueing problem

$$\text{minimize} \quad C = \underbrace{\lim_{t\to\infty} e^{\gamma t}\mathbf{P}[T > t]}_{\text{“}\infty\cdot\mathbf{P}[T>\infty]\text{”}} = \underbrace{\lim_{\theta\to\gamma} \frac{\gamma-\theta}{\gamma}\mathbf{E}[e^{\theta T}]}_{\text{“}0\cdot\mathbf{E}[e^{\gamma T}]\text{”}}$$

(by final value theorem for Laplace transforms)

Batch problem

$$\text{minimize} \quad \mathbf{P}[T > \infty] = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(t_i > \infty) = 0$$

$$\text{minimize} \quad \mathbf{E}[e^{\gamma T}] = \frac{1}{n}\sum_{i=1}^{n} e^{\gamma t_i}$$

$t_i = d_i - a_i$

$a_i =$ arrival time of job i

$d_i =$ departure time of job i

Queueing problem

$$\text{minimize} \quad C = \underbrace{\lim_{t\to\infty} e^{\gamma t}\mathbf{P}[T > t]}_{\text{“}\infty \cdot \mathbf{P}[T > \infty]\text{”}} = \underbrace{\lim_{\theta\to\gamma} \frac{\gamma - \theta}{\gamma}\mathbf{E}[e^{\theta T}]}_{\text{“}0 \cdot \mathbf{E}[e^{\gamma T}]\text{”}}$$

(by final value theorem for Laplace transforms)

Batch problem

$$\text{minimize} \quad \mathbf{P}[T > \infty] = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(t_i > \infty) = 0$$

$$\text{minimize} \quad \mathbf{E}[e^{\gamma T}] = \frac{1}{n}\sum_{i=1}^{n} e^{\gamma t_i} = \frac{1}{n}\sum_{i=1}^{n} e^{-\gamma a_i} e^{\gamma d_i}$$

$t_i = d_i - a_i$

$a_i =$ arrival time of job i

$d_i =$ departure time of job i

Queueing problem

$$\text{minimize} \quad C = \lim_{t\to\infty} e^{\gamma t}\mathbf{P}[T > t] = \lim_{\theta\to\gamma} \frac{\gamma-\theta}{\gamma}\mathbf{E}[e^{\theta T}]$$

$$\underbrace{\lim_{t\to\infty} e^{\gamma t}\mathbf{P}[T > t]}_{\text{“}\infty\cdot\mathbf{P}[T>\infty]\text{”}} \qquad \underbrace{\lim_{\theta\to\gamma} \frac{\gamma-\theta}{\gamma}\mathbf{E}[e^{\theta T}]}_{\text{“}0\cdot\mathbf{E}[e^{\gamma T}]\text{”}}$$

(by final value theorem for Laplace transforms)

Batch problem

$$\text{minimize} \quad \mathbf{P}[T > \infty] = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(t_i > \infty)$$

almost classic problem

$$\text{minimize} \quad \mathbf{E}[e^{\gamma T}] = \frac{1}{n}\sum_{i=1}^{n} e^{\gamma t_i} = \frac{1}{n}\sum_{i=1}^{n} e^{-\gamma a_i} e^{\gamma d_i}$$

$t_i = d_i - a_i$

a_i = arrival time of job i

d_i = departure time of job i

Solving the batch problem

$$\text{minimize} \;\; \mathbf{E}[e^{\gamma T}] = \frac{1}{n}\sum_{i=1}^{n} e^{\gamma t_i} = \frac{1}{n}\sum_{i=1}^{n} e^{-\gamma a_i} e^{\gamma d_i}$$

$t_i = d_i - a_i$

a_i = arrival time of job i

d_i = departure time of job i

Solving the batch problem

minimize $\mathbf{E}[e^{\gamma T}] = \frac{1}{n}\sum_{i=1}^{n} e^{\gamma t_i} = \frac{1}{n}\sum_{i=1}^{n} e^{-\gamma a_i} e^{\gamma d_i}$

$t_i = d_i - a_i$
$a_i =$ arrival time of job i
$d_i =$ departure time of job i

Classic metric: mean weighted discounted departure time $\frac{1}{n}\sum_{i=1}^{n} w_i e^{-\theta d_i}$

Solving the batch problem

$$\text{minimize} \quad \mathbf{E}[e^{\gamma T}] = \frac{1}{n}\sum_{i=1}^{n} e^{\gamma t_i} = \frac{1}{n}\sum_{i=1}^{n} e^{-\gamma a_i} e^{\gamma d_i}$$

$t_i = d_i - a_i$

$a_i =$ arrival time of job i

$d_i =$ departure time of job i

⚠ **Classic metric:** mean weighted discounted departure time $\frac{1}{n}\sum_{i=1}^{n} w_i e^{-\theta d_i}$

Solving the batch problem

$$\text{minimize} \quad \mathbf{E}[e^{\gamma T}] = \frac{1}{n}\sum_{i=1}^{n} e^{\gamma t_i} = \frac{1}{n}\sum_{i=1}^{n} e^{-\gamma a_i} e^{\gamma d_i}$$

$\gamma > 0$

$t_i = d_i - a_i$

a_i = arrival time of job i

d_i = departure time of job i

Classic metric: mean weighted discounted departure time $\frac{1}{n}\sum_{i=1}^{n} w_i e^{-\theta d_i}$

Solving the batch problem

$$\text{minimize} \quad \mathbf{E}[e^{\gamma T}] = \frac{1}{n}\sum_{i=1}^{n} e^{\gamma t_i} = \frac{1}{n}\sum_{i=1}^{n} e^{-\gamma a_i} e^{\gamma d_i}$$

$\gamma > 0$

$t_i = d_i - a_i$

$a_i =$ arrival time of job i

$d_i =$ departure time of job i

negative discount rate

Classic metric: mean weighted discounted departure time $\frac{1}{n}\sum_{i=1}^{n} w_i e^{-\theta d_i}$

Solving the batch problem

$t_i = d_i - a_i$
$a_i =$ arrival time of job i
$d_i =$ departure time of job i

minimize $\mathbf{E}[e^{\gamma T}] = \frac{1}{n}\sum_{i=1}^{n} e^{\gamma t_i} = \frac{1}{n}\sum_{i=1}^{n} e^{-\gamma a_i} e^{\gamma d_i}$

$\gamma > 0$

negative discount rate

can't start i before a_i

Classic metric: mean weighted discounted departure time $\frac{1}{n}\sum_{i=1}^{n} w_i e^{-\theta d_i}$

Solving the batch problem

$t_i = d_i - a_i$
$a_i =$ arrival time of job i
$d_i =$ departure time of job i

minimize $\mathbf{E}[e^{\gamma T}] = \frac{1}{n}\sum_{i=1}^{n} e^{\gamma t_i} = \frac{1}{n}\sum_{i=1}^{n} e^{-\gamma a_i} e^{\gamma d_i}$

$\gamma > 0$

negative discount rate

can't start i before a_i

Classic metric: mean weighted discounted departure time $\frac{1}{n}\sum_{i=1}^{n} w_i e^{-\theta d_i}$

Relaxation solved by (sign-flipped) WDSPT, which is **Boost** with

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

Solving the batch problem

$$\text{minimize}\;\; \mathbf{E}[e^{\gamma T}] = \frac{1}{n}\sum_{i=1}^{n} e^{\gamma t_i} = \frac{1}{n}\sum_{i=1}^{n} e^{-\gamma a_i} e^{\gamma d_i}$$

$\gamma > 0$

$t_i = d_i - a_i$
$a_i =$ arrival time of job i
$d_i =$ departure time of job i

negative discount rate

can't start i before a_i

Classic metric: mean weighted discounted departure time $\frac{1}{n}\sum_{i=1}^{n} w_i e^{-\theta d_i}$

Relaxation solved by (sign-flipped) WDSPT, which is **Boost** with

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

γ-Boost

Solving the batch problem

$$\text{minimize}\ \ \mathbf{E}[e^{\gamma T}] = \frac{1}{n}\sum_{i=1}^{n} e^{\gamma t_i} = \frac{1}{n}\sum_{i=1}^{n} e^{-\gamma a_i} e^{\gamma d_i}$$

$\gamma > 0$

$t_i = d_i - a_i$

$a_i =$ arrival time of job i

$d_i =$ departure time of job i

negative discount rate

can't start i before a_i

Cl…

dis…

$\dots w_i e^{-\theta d_i}$

Takeaway:

$\mathbf{E}[e^{\gamma T}]$ for critical γ is a *"smoothed" tail metric*

Relaxation solved by (sign-flipped) WDSPT, which is **Boost** with

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

γ-Boost

Solving the batch problem

$$\text{minimize}\quad \mathbf{E}[e^{\gamma T}] = \frac{1}{n}\sum_{i=1}^{n} e^{\gamma t_i} = \frac{1}{n}\sum_{i=1}^{n} e^{-\gamma a_i} e^{\gamma d_i}$$

$\gamma > 0$

$t_i = d_i - a_i$
a_i = arrival time of job i
d_i = departure time of job i

negative discount rate

can't start i before a_i

Cl… dis… $w_i e^{-\theta d_i}$

Takeaway:
$\mathbf{E}[e^{\gamma T}]$ for critical γ is a "*smoothed*" *tail metric*

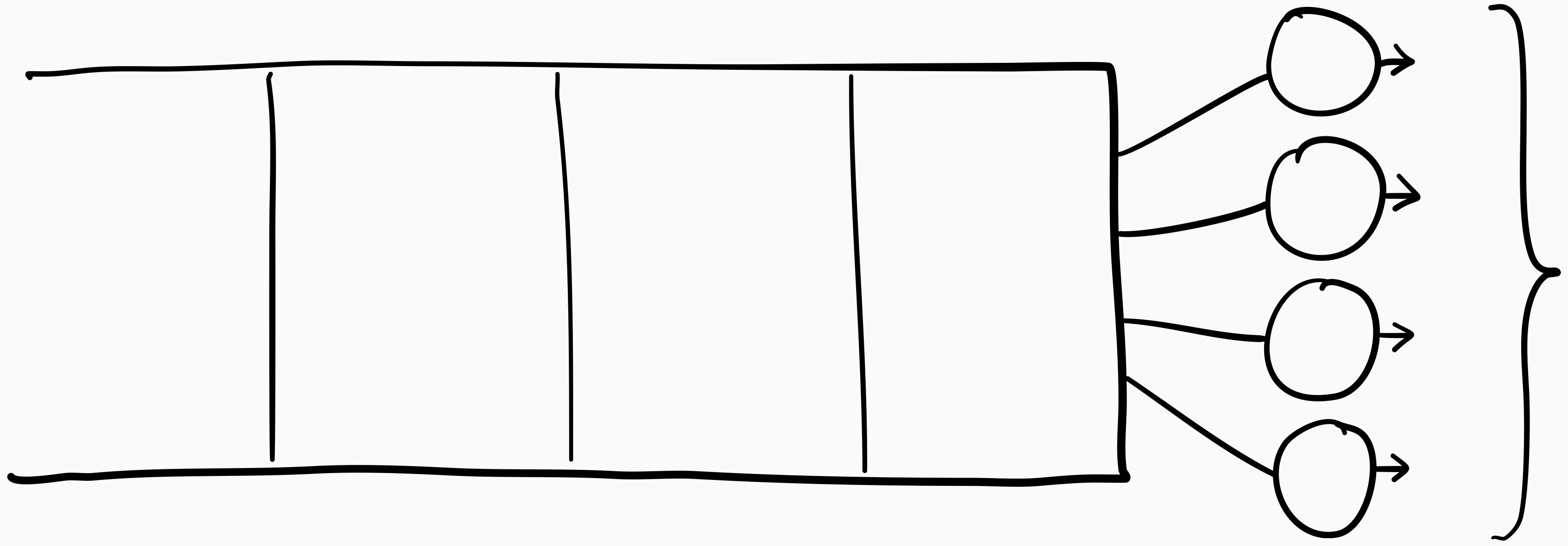
Relaxation solved by (sign-flipped) WDSPT, which is **Boost** with

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

***γ*-Boost**

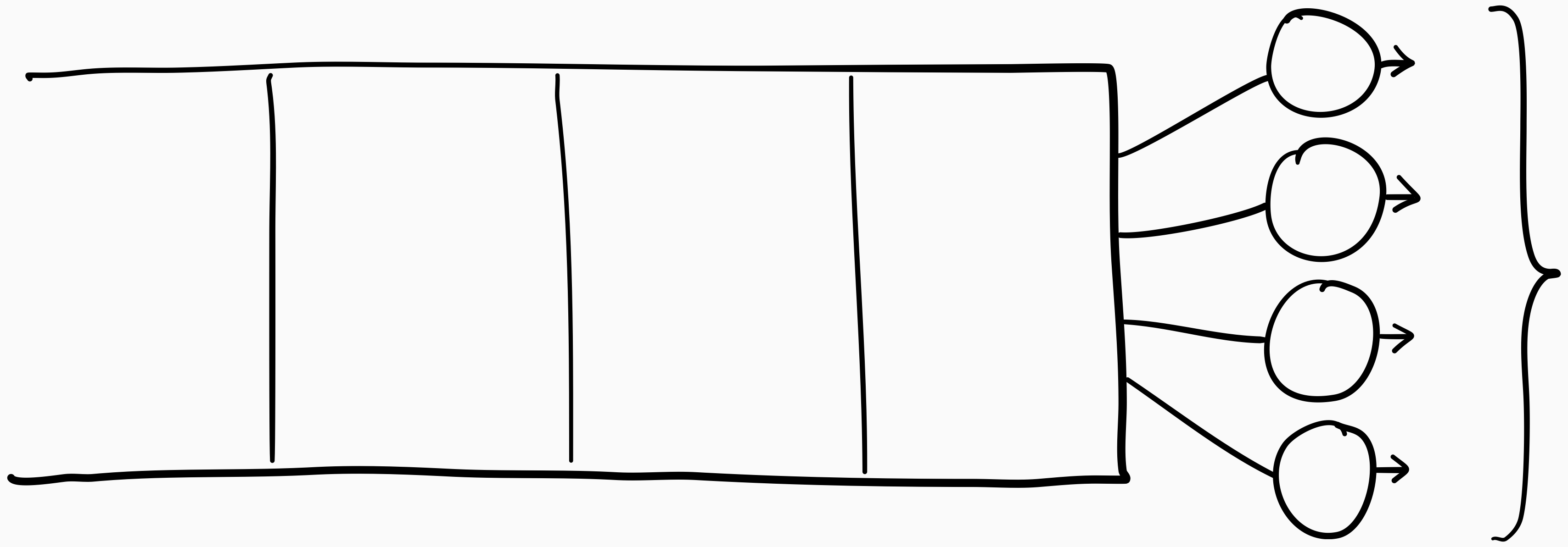
Unknown sizes:
swap WDSPT for *Gittins*

Multiserver tail optimization



k servers of speed $1/k$

Multiserver tail optimization

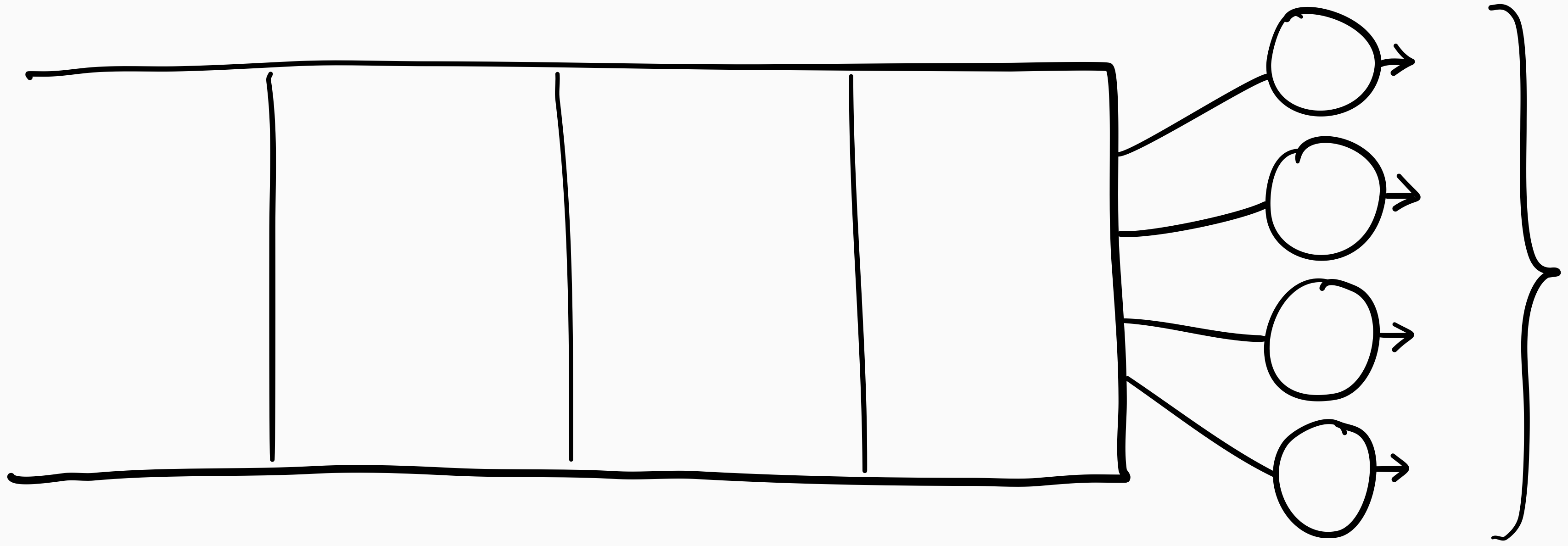


k servers of speed $1/k$

Theorem: **γ-Boost** is strongly tail-optimal in the heavy-traffic limit [Yu et al., 2025]

$\rho = \lambda \mathbf{E}[S] \to 1$

Multiserver tail optimization

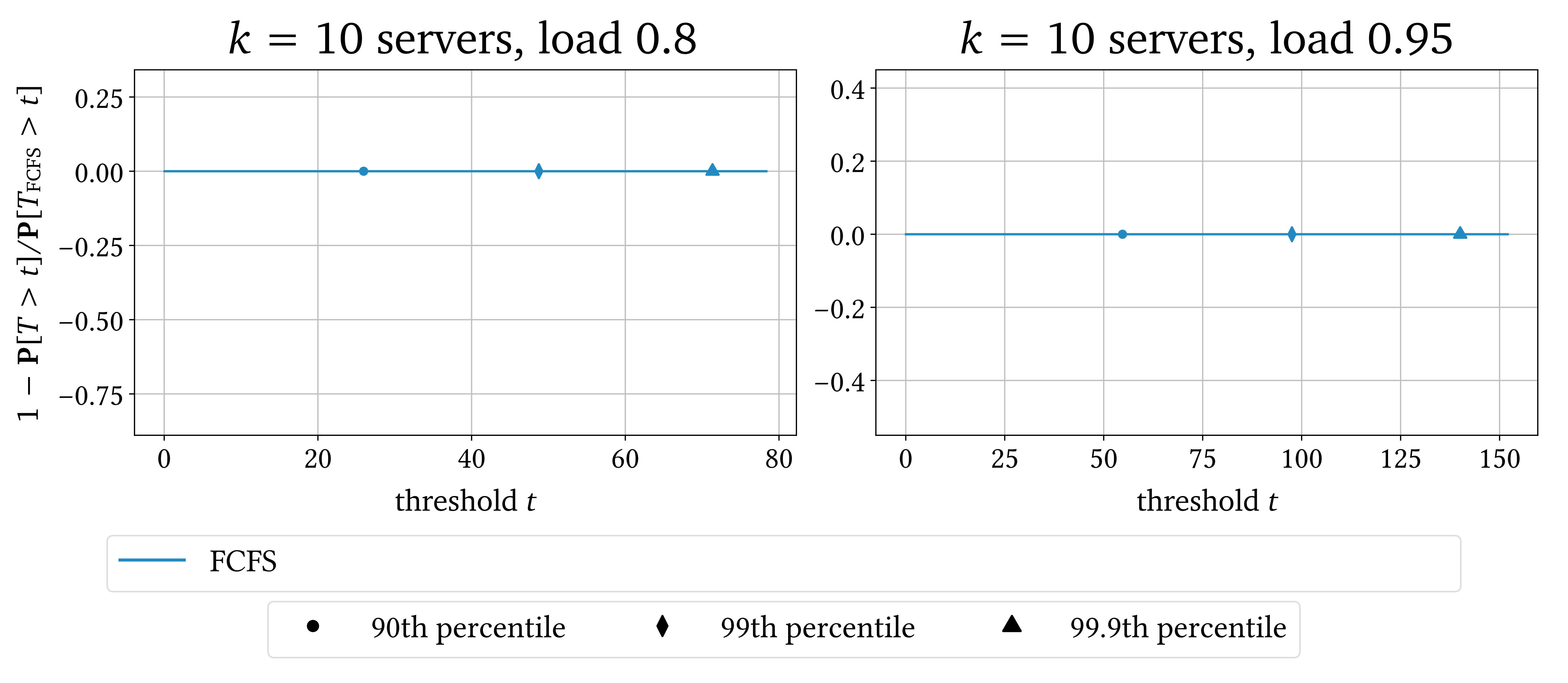


k servers of speed $1/k$

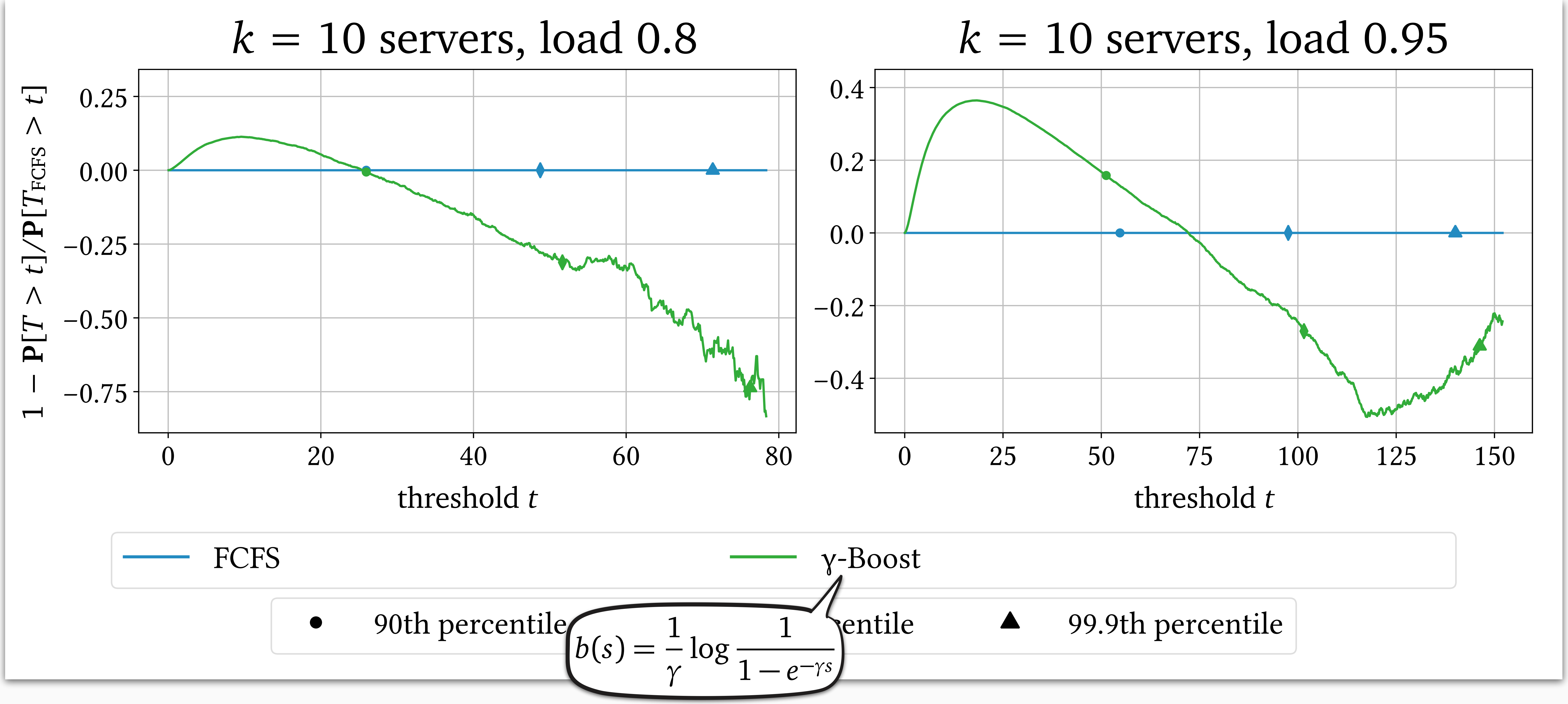
Theorem: **γ-Boost** is strongly tail-optimal in the heavy-traffic limit [Yu et al., 2025]

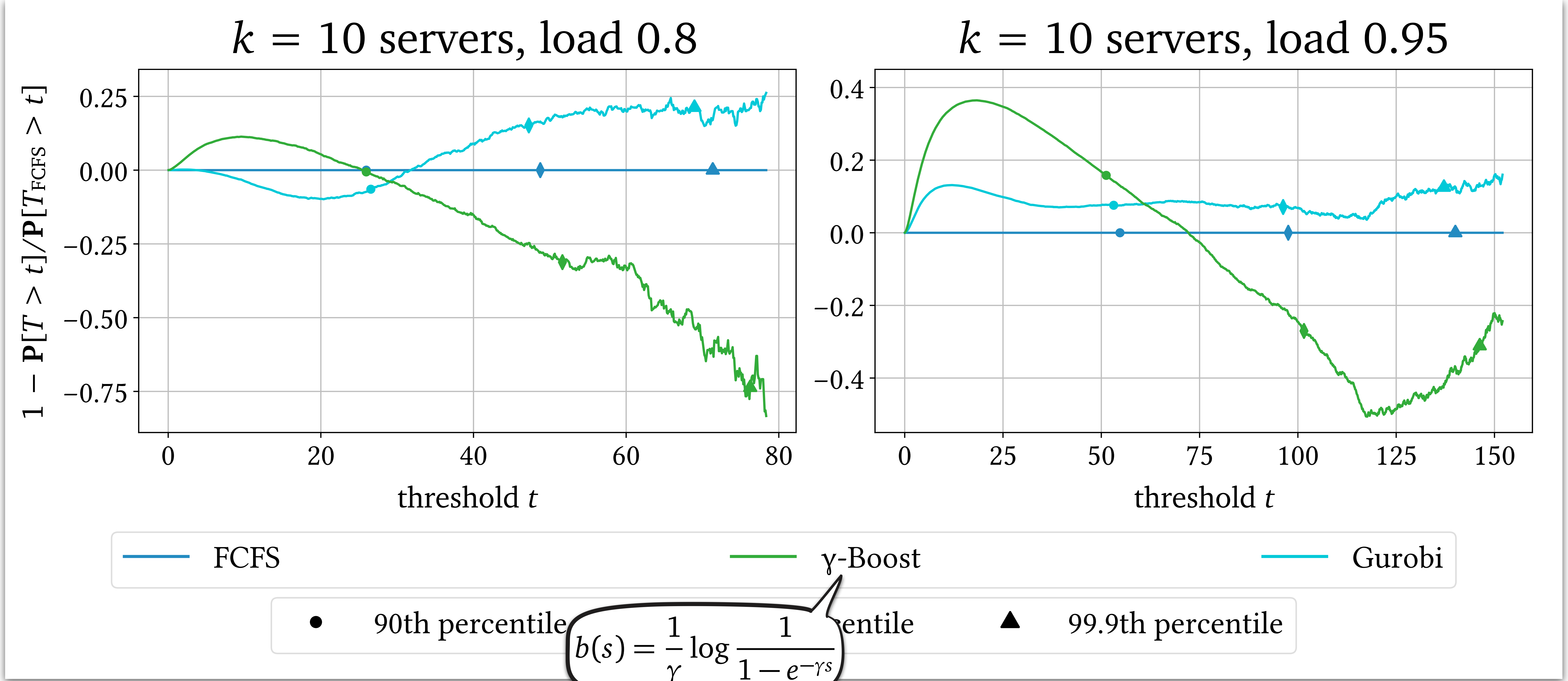
$\rho = \lambda \mathbf{E}[S] \to 1$

load < 1?

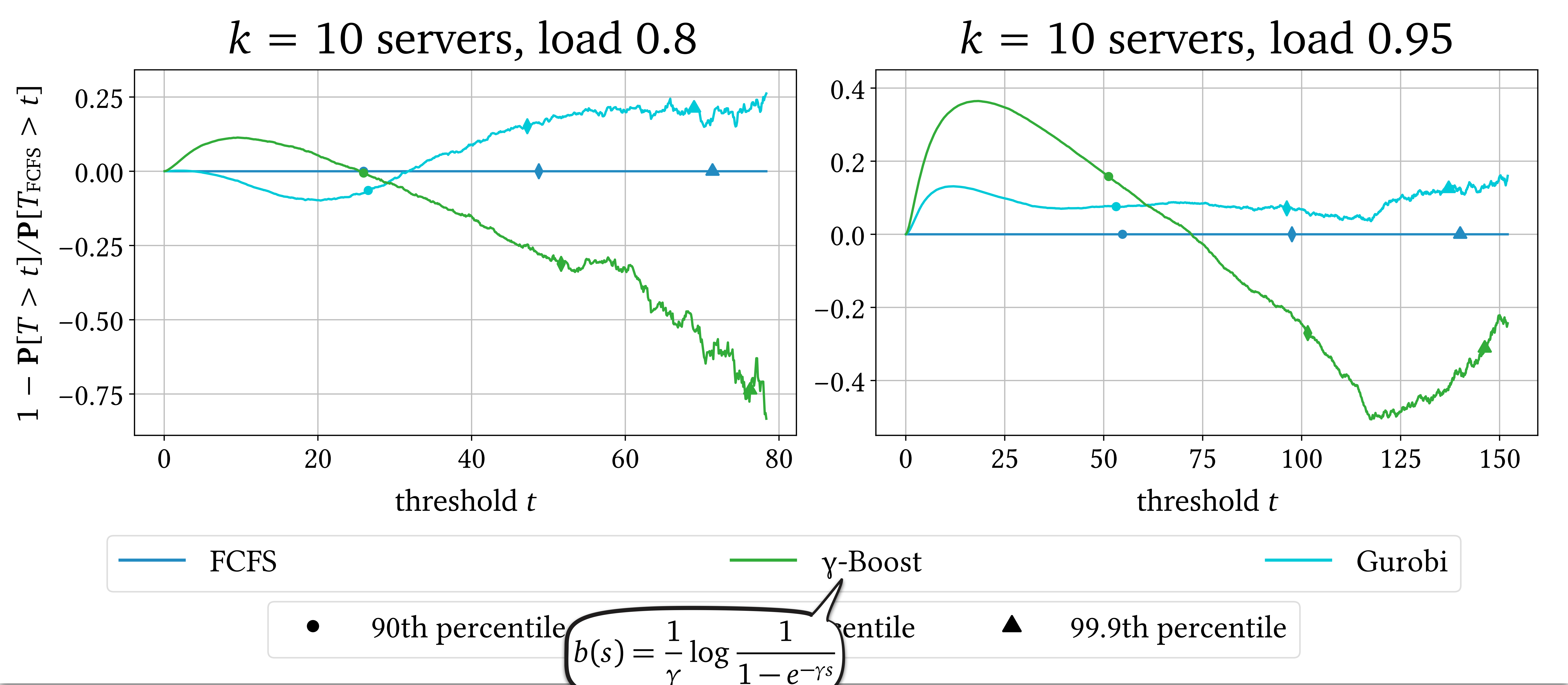


k = 10 servers, load 0.8
k = 10 servers, load 0.95
$1 - \mathbf{P}[T > t]/\mathbf{P}[T_{\mathrm{FCFS}} > t]$
0.25
0.00
−0.25
−0.50
−0.75
0
20
40
60
80
0.4
0.2
0.0
−0.2
−0.4
0
25
50
75
100
125
150
threshold t
threshold t
FCFS
90th percentile
99th percentile
99.9th percentile

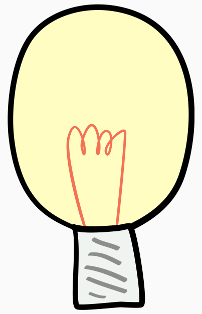

k = 10 servers, load 0.8
k = 10 servers, load 0.95
$1 - \mathbf{P}[T > t]/\mathbf{P}[T_{\text{FCFS}} > t]$
0.25
0.00
−0.25
−0.50
−0.75
0 20 40 60 80
threshold t
0.4
0.2
0.0
−0.2
−0.4
0 25 50 75 100 125 150
threshold t
FCFS
γ-Boost
90th percentile
99th percentile
99.9th percentile
$b(s) = \frac{1}{\gamma} \log \frac{1}{1 - e^{-\gamma s}}$



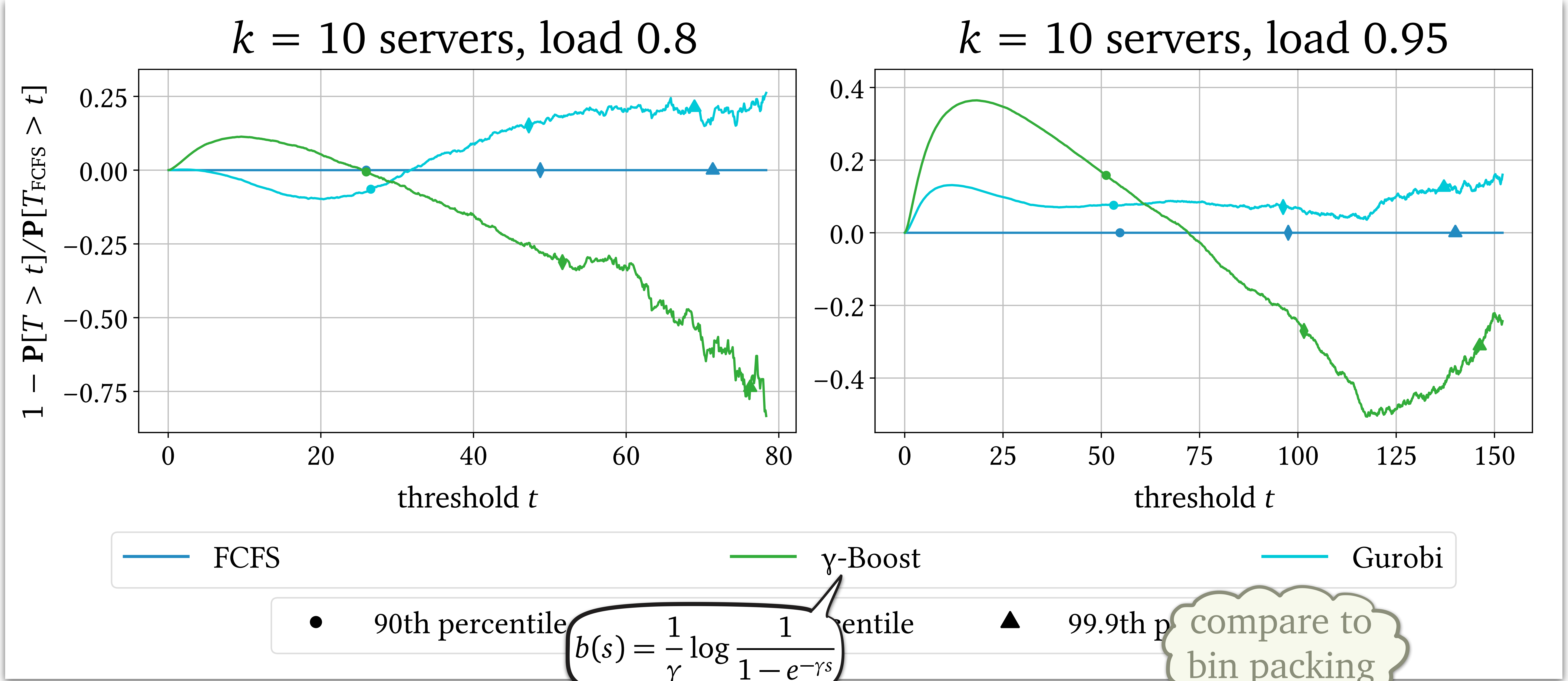
$$b(s) = \frac{1}{\gamma} \log \frac{1}{1 - e^{-\gamma s}}$$



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



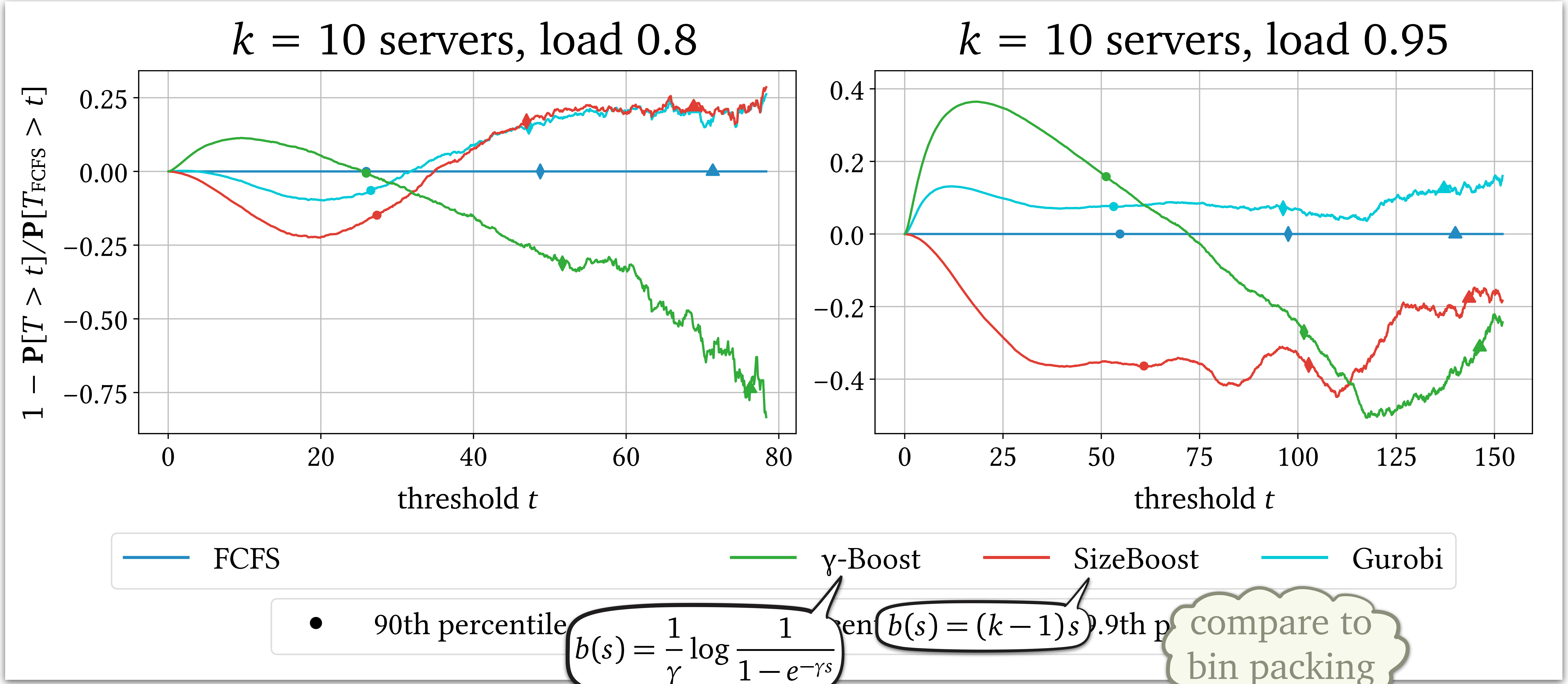
Boost large jobs?



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

compare to bin packing

Boost large jobs?

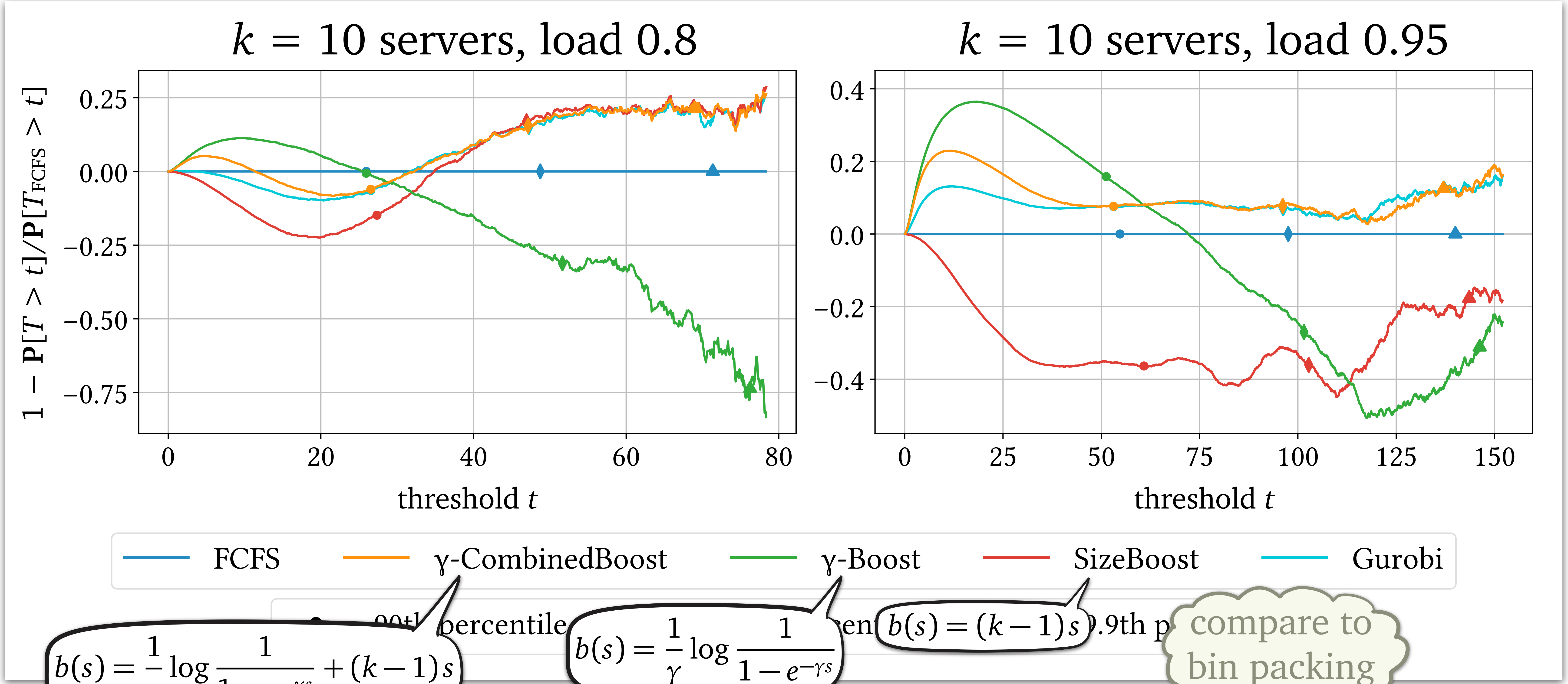


$$b(s) = \frac{1}{\gamma} \log \frac{1}{1 - e^{-\gamma s}}$$

$$b(s) = (k-1)s$$

compare to bin packing

Boost large jobs?

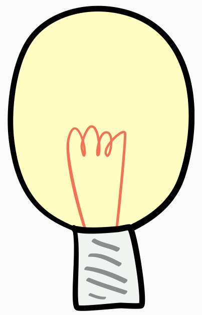


$$b(s) = \frac{1}{\gamma} \log \frac{1}{1 - e^{-\gamma s}} + (k-1)s$$

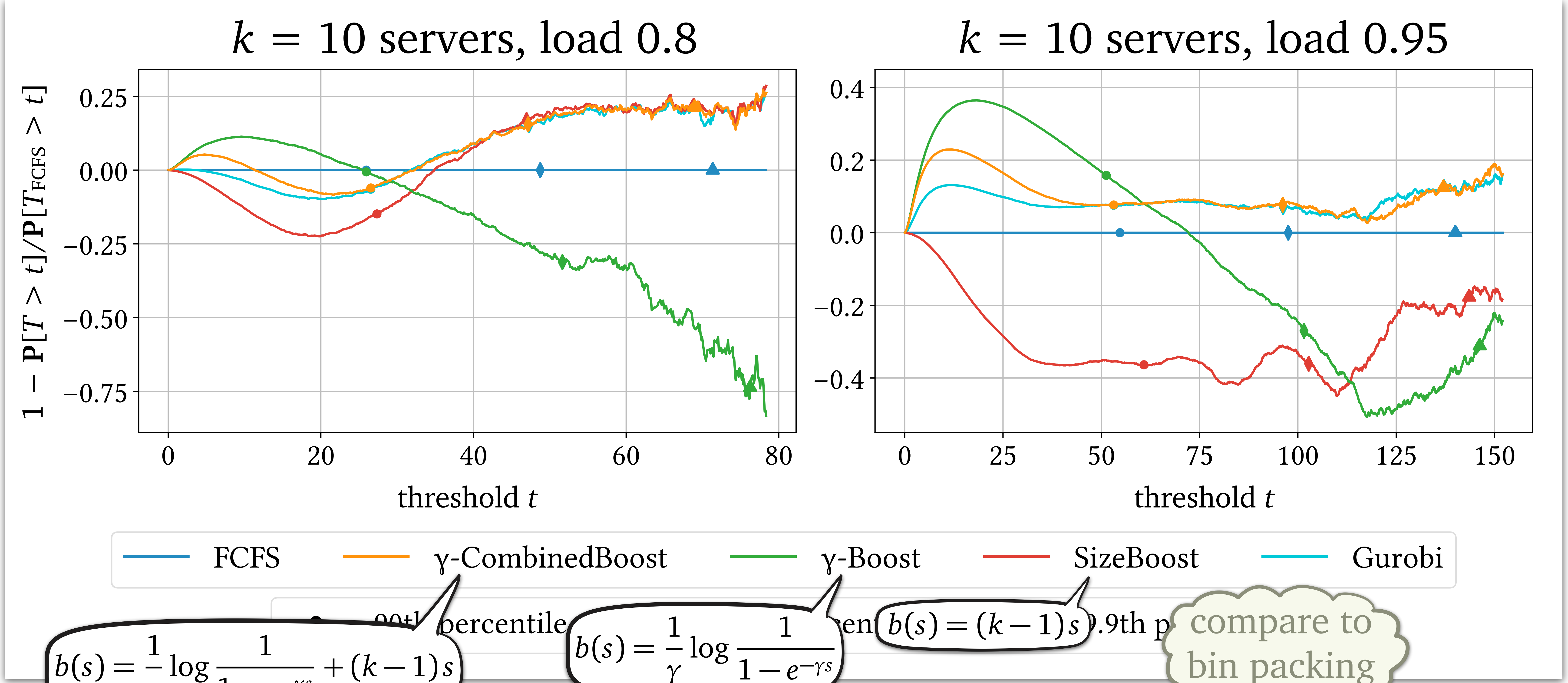
$$b(s) = \frac{1}{\gamma} \log \frac{1}{1 - e^{-\gamma s}}$$

$$b(s) = (k-1)s$$

compare to bin packing



Boost large jobs?

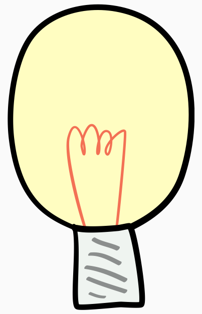


$$b(s) = \frac{1}{\gamma} \log \frac{1}{1 - e^{-\gamma s}} + (k-1)s$$

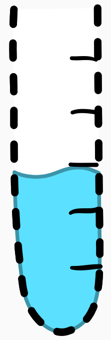
$$b(s) = \frac{1}{\gamma} \log \frac{1}{1 - e^{-\gamma s}}$$

$$b(s) = (k-1)s$$

compare to bin packing

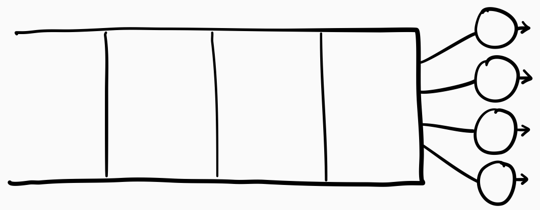


Boost large *and* small jobs!?



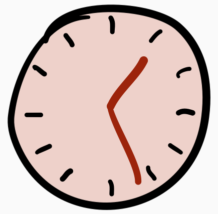
Part I

Handling job size uncertainty



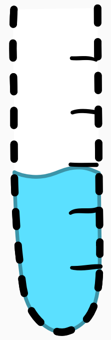
Part II

Analyzing multiserver scheduling



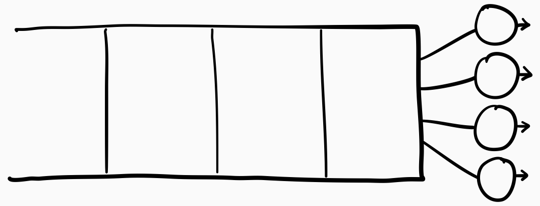
Part III

Optimizing tail metrics



Part I

Handling job size uncertainty

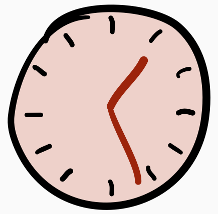


Part II

Analyzing multiserver scheduling

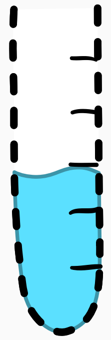
TCS for Queueing

Approximation algorithms for smoothed tail metric $\mathbf{E}[e^{\gamma T}]$*?*



Part III

Optimizing tail metrics



Part I

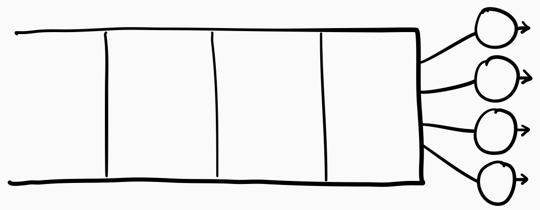
Handling job size uncertainty

Queueing for TCS

*Use **WINE** to analyze **Gittins** with arbitrary release dates?*

Queueing for TCS

*Use **WINE** to analyze SRPT-**k** with arbitrary release dates?*

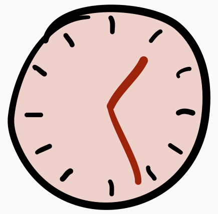


Part II

Analyzing multiserver scheduling

TCS for Queueing

Approximation algorithms for smoothed tail metric $\mathbf{E}[e^{\gamma T}]$?



Part III

Optimizing tail metrics