Recent Progress in

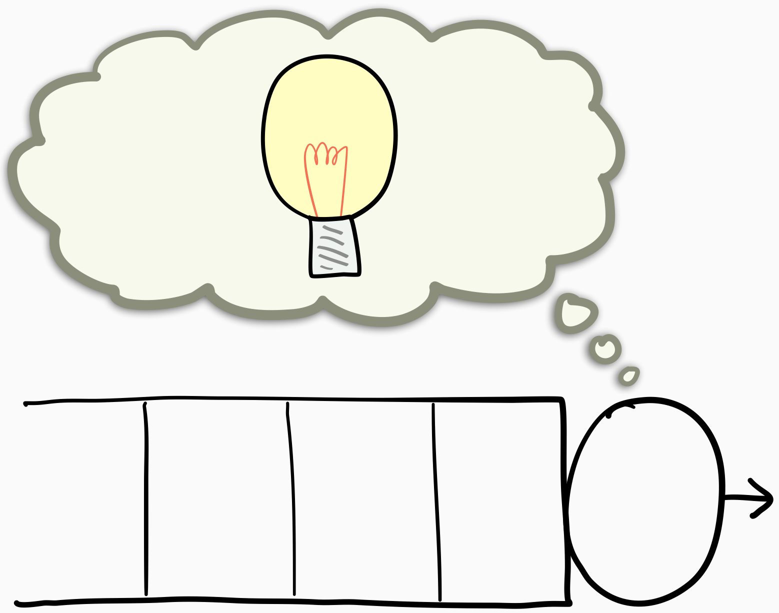
Queueing and Scheduling Theory

(for a TCS Audience)

Ziv Scully

Harvard & MIT → Cornell

`zivscully@cornell.edu`
`https://ziv.codes`

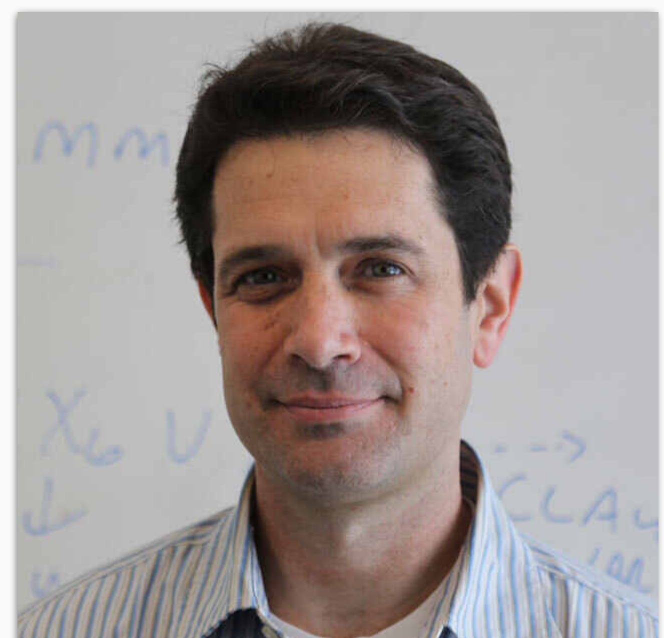


Collaborators

Isaac Grosof
CMU

Mor Harchol-Balter
CMU

Alan Scheller-Wolf
CMU



Michael Mitzenmacher
Harvard

Contention

↓

Queueing

↓

Delay

healthcare

Contention

Queueing

Delay

supply chains

healthcare

Contention

Queueing

Delay

your local supermarket

supply chains

healthcare

Contention

↓

Queueing

↓

Delay

your local supermarket

supply chains

Contention

call centers

healthcare

Queueing

Delay

your local supermarket

supply chains

Contention

call centers

healthcare

transportation

Queueing

Delay

your local supermarket

supply chains

Contention

call centers

healthcare

transportation

Queueing

databases

Delay

your local supermarket

supply chains

Contention

call centers

healthcare

transportation

Queueing

databases

Delay

networks

your local supermarket

supply chains **Contention** call centers

healthcare transportation

Queueing

databases

Delay networks

operating systems

your local supermarket

supply chains

Contention

call centers

healthcare

transportation

Queueing

databases

computer architecture

Delay

networks

operating systems

your local supermarket

supply chains

Contention

call centers

healthcare

transportation

Queueing

supercomputing

databases

computer architecture

Delay

networks

operating systems

your local supermarket

supply chains

call centers

healthcare

transportation

supercomputing

databases

computer architecture

networks

operating systems

Contention

Queueing

Delay

How to reduce delays?

your local supermarket

supply chains **Contention** call centers

healthcare transportation

Queueing

supercomputing databases

computer architecture **Delay** networks

operating systems

How to reduce delays?

Scheduling

Good news:

scheduling can reduce delay

Good news:

scheduling can reduce delay

Bad news:

limited understanding of scheduling

Good news:

scheduling can reduce delay

Bad news:

limited understanding of scheduling

evaluation

Good news:

scheduling can reduce delay

Bad news:

limited understanding of scheduling

design

evaluation

Good news:

scheduling can reduce delay

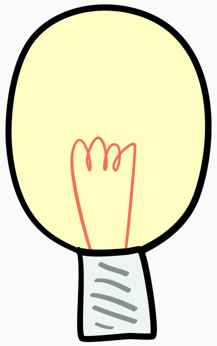
Bad news:

limited understanding of scheduling

design evaluation

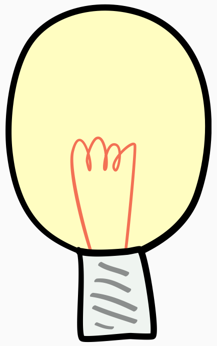
We need:

rigorous theory of scheduling



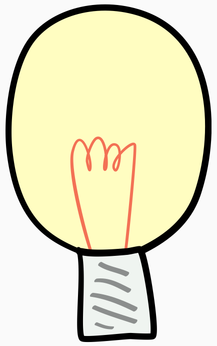
We need:

rigorous theory of scheduling



We need:

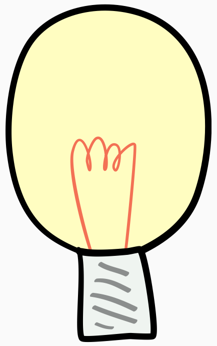
rigorous theory of scheduling



We need:

rigorous theory of scheduling

CS Theory

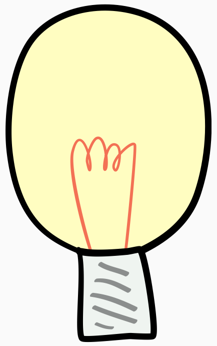


We need:

rigorous theory of scheduling

CS Theory

- Worst-case modeling
- Complex algorithms



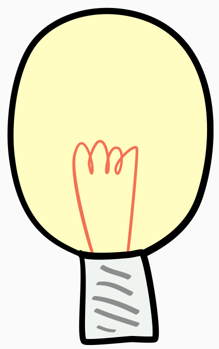
We need:

rigorous theory of scheduling

CS Theory

Queueing Theory

- Worst-case modeling
- Complex algorithms



We need:

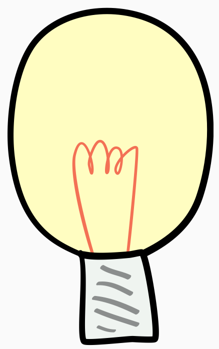
rigorous theory of scheduling

CS Theory

- Worst-case modeling
- Complex algorithms

Queueing Theory

- Stochastic modeling
- Simple algorithms



We need:

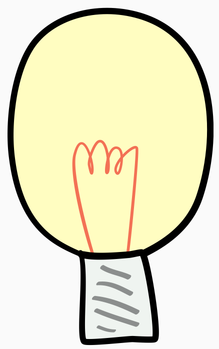
rigorous theory of scheduling

CS Theory

- Worst-case modeling
- Complex algorithms

Queueing Theory

- Stochastic modeling
- Simple algorithms



We need:

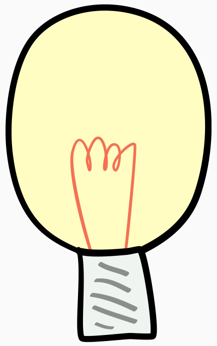
rigorous theory of scheduling

CS Theory

- Worst-case modeling
- Complex algorithms

Queueing Theory

- Stochastic modeling
- Simple algorithms



We need:

rigorous theory of scheduling

CS Theory

- Worst-case modeling
- Complex algorithms

Queueing Theory

- Stochastic modeling
- Simple algorithms

We need:

rigorous theory of scheduling

CS Theory

- Worst-case modeling
- Complex algorithms

Queueing Theory

- Stochastic modeling
- Simple algorithms

We need:

rigorous theory of scheduling

CS Theory

- Worst-case modeling
- Complex algorithms

Queueing Theory

- Stochastic modeling
- Simple algorithms

Best to learn from both

We need:

rigorous theory of scheduling

CS Theory

- Worst-case modeling
- Complex algorithms

Queueing Theory

- Stochastic modeling
- Simple algorithms

Best to learn from both

Today's talk

Today's talk

scheduling with

multiple servers

Today's talk

scheduling with
multiple servers

scheduling with
noisy predictions

Today's talk

scheduling with

multiple servers

TCS **Queueing**

scheduling with

noisy predictions

TCS **Queueing**

Today's talk

scheduling with

multiple servers

TCS **Queueing**

scheduling with

noisy predictions

TCS **Queueing**

Powered by **new tools**
in **queueing theory**

Today's talk

scheduling with
multiple servers

TCS **Queueing**

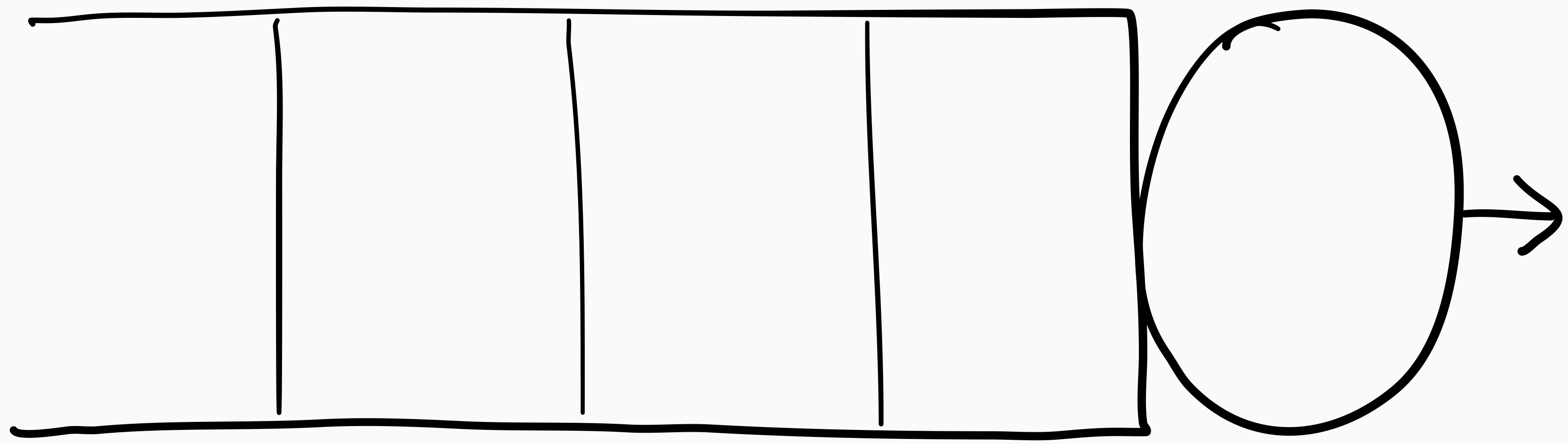
scheduling with
noisy predictions

TCS **Queueing**

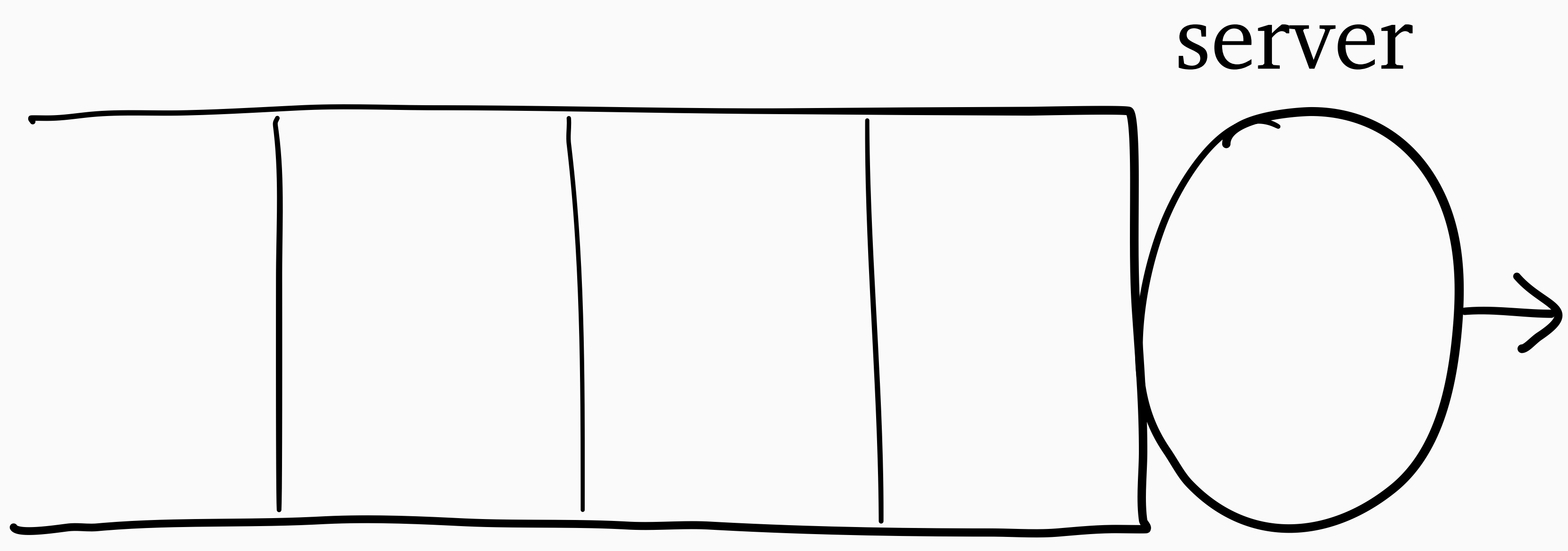
Powered by **new tools**
in **queueing theory**

possible
crossover

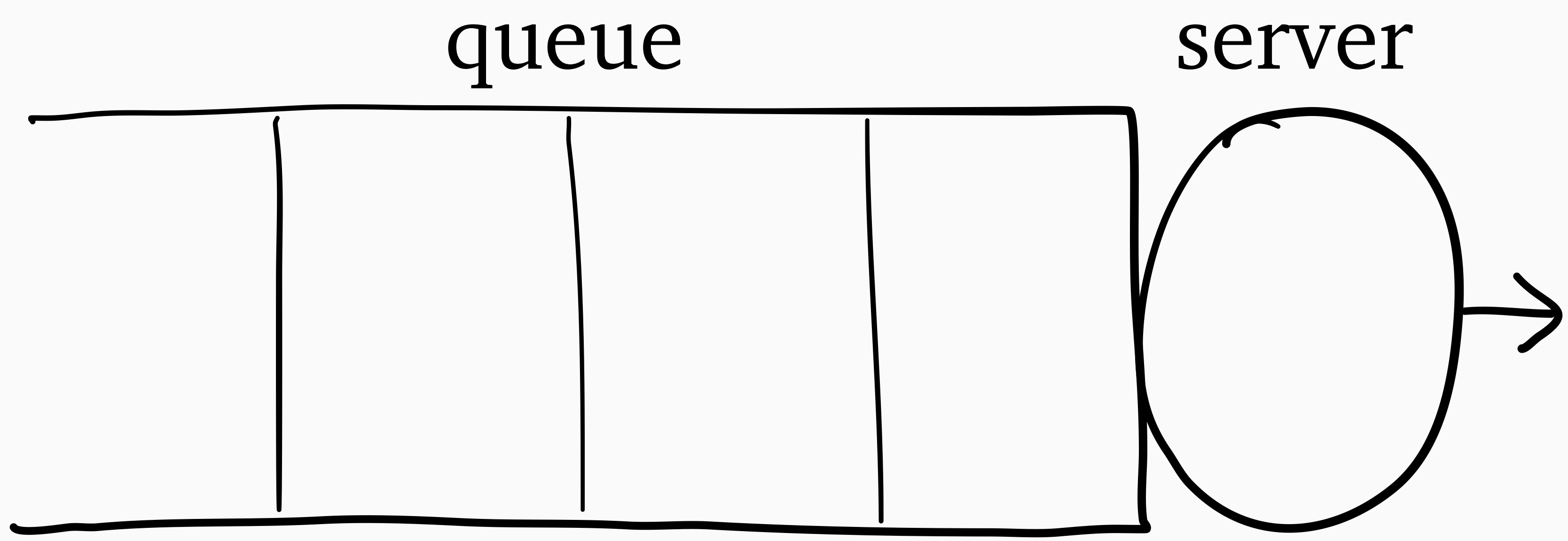
What is scheduling?



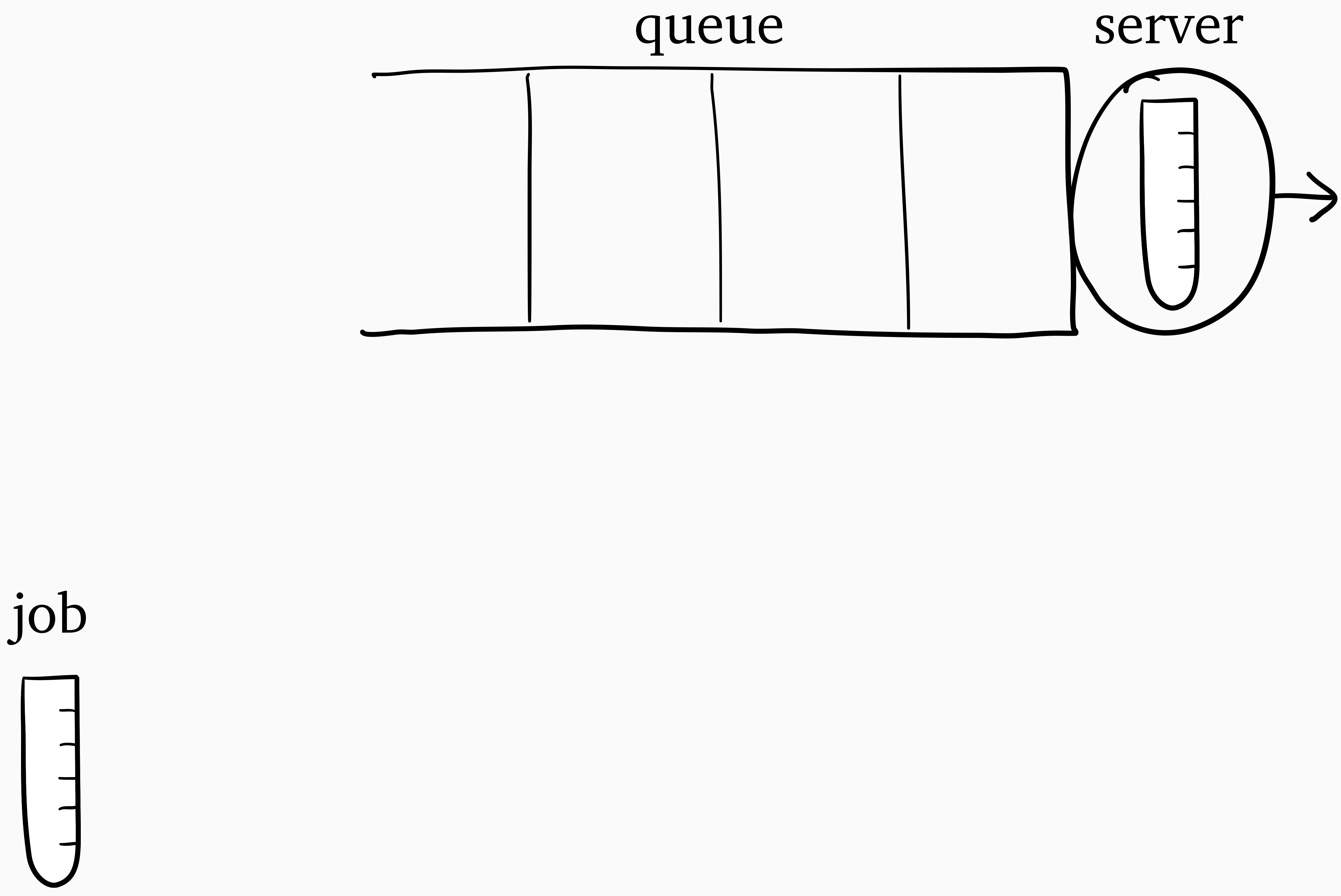
What is scheduling?



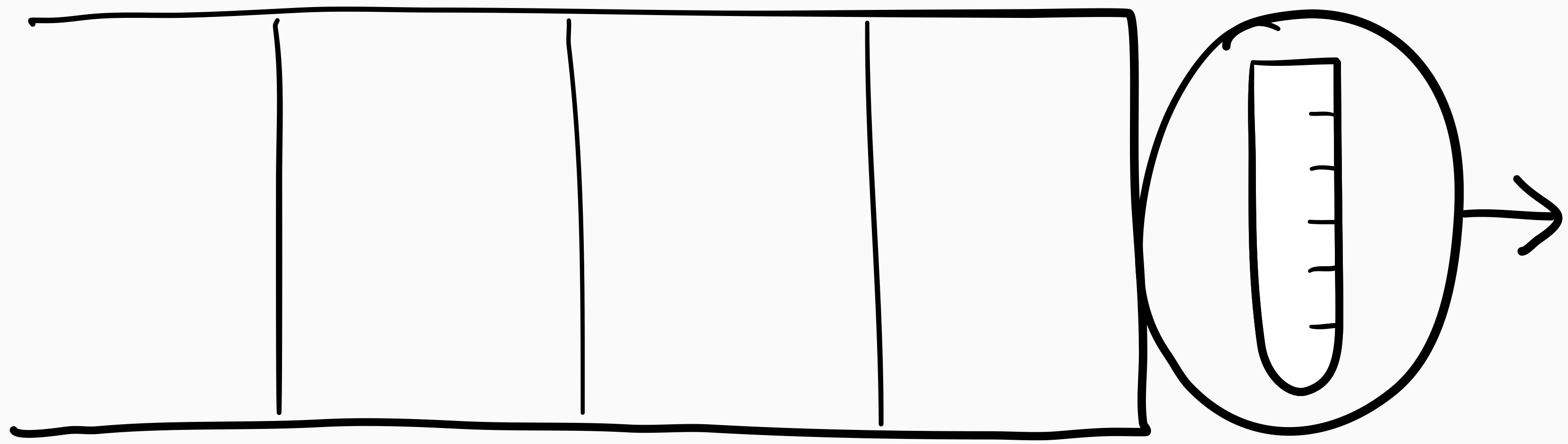
What is scheduling?

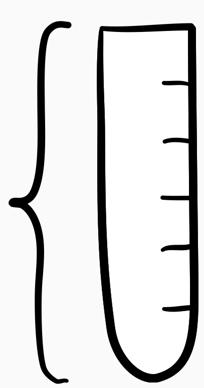


What is scheduling?

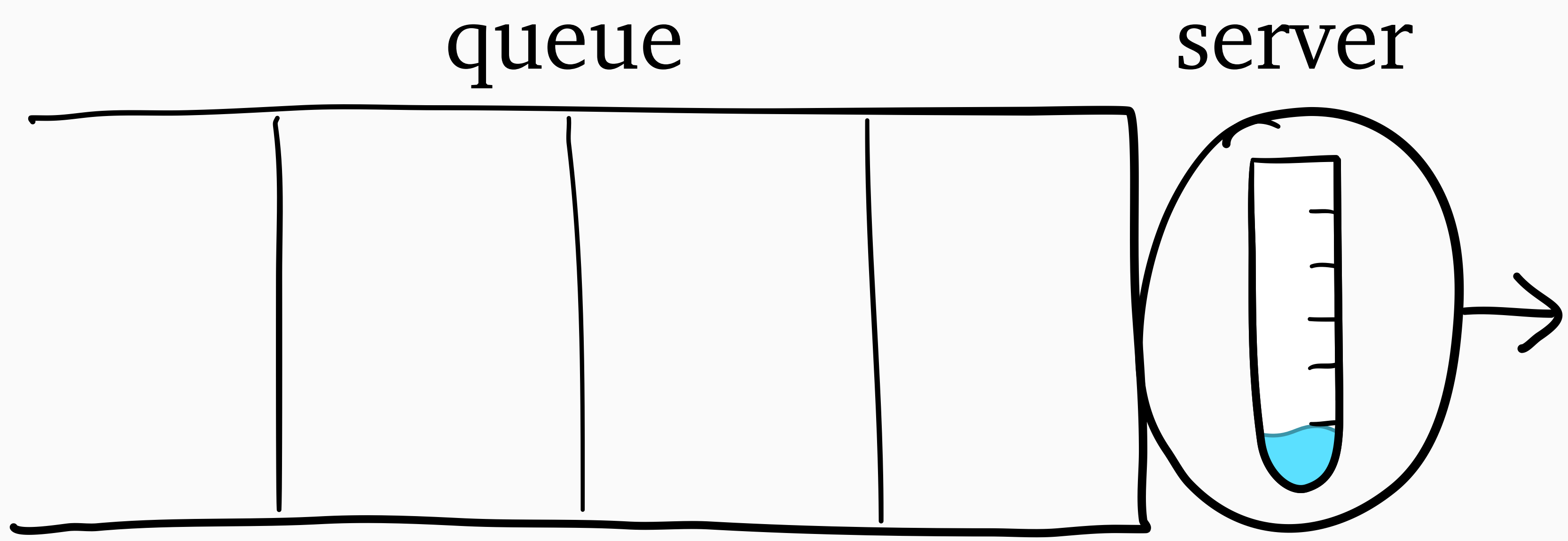


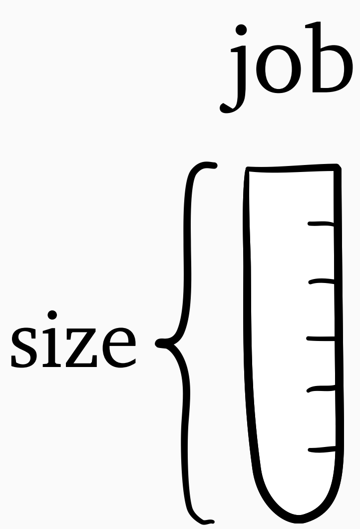
What is scheduling?



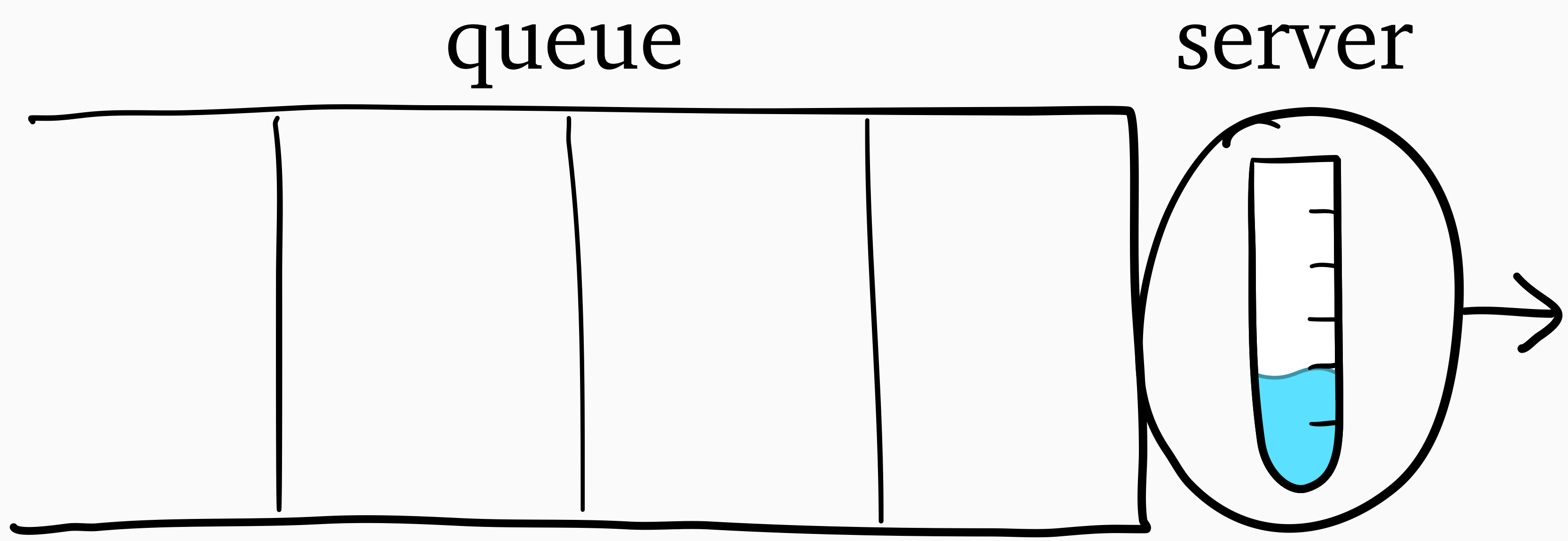


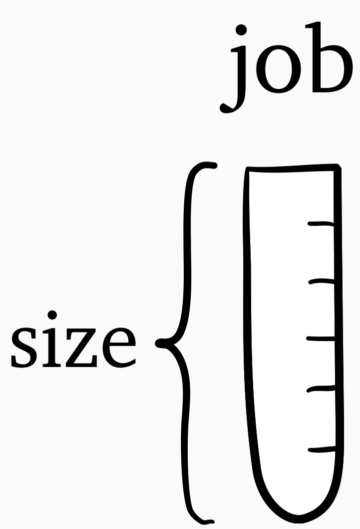
What is scheduling?



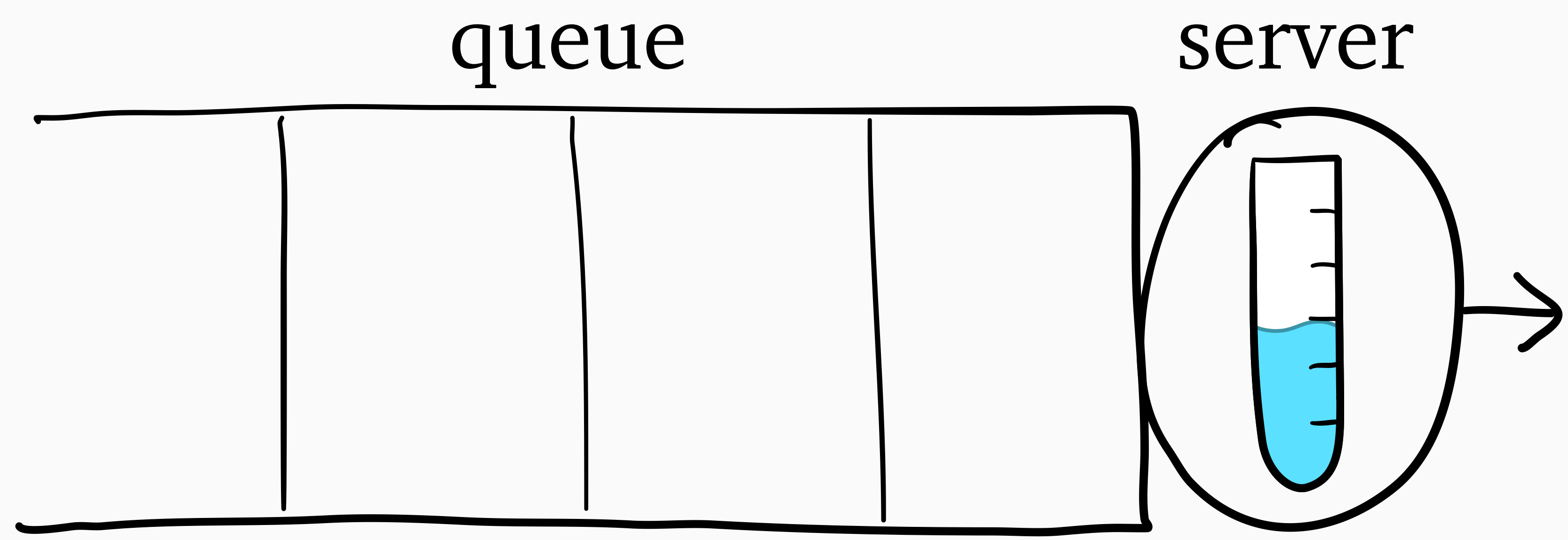


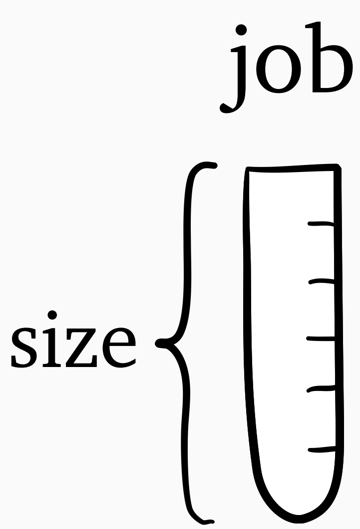
What is scheduling?



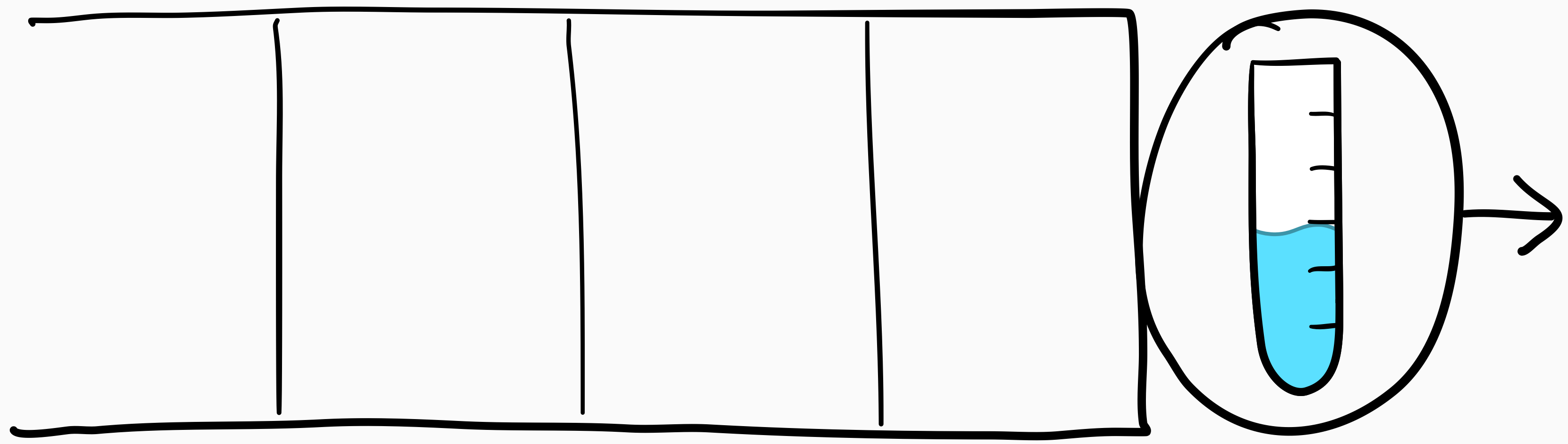


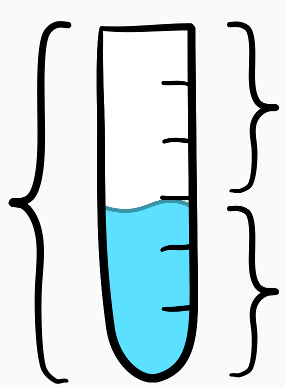
What is scheduling?



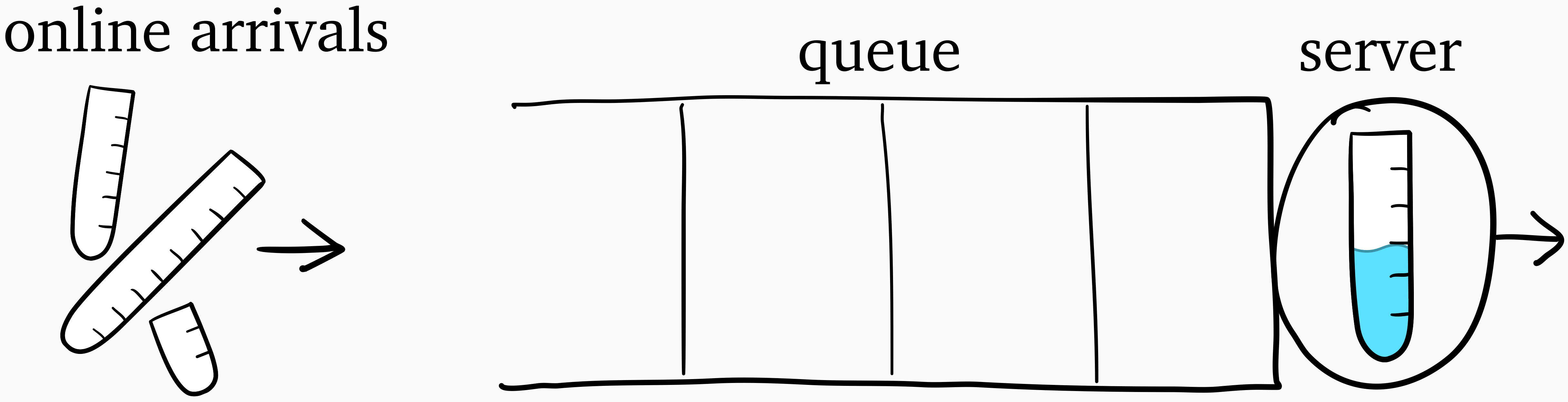


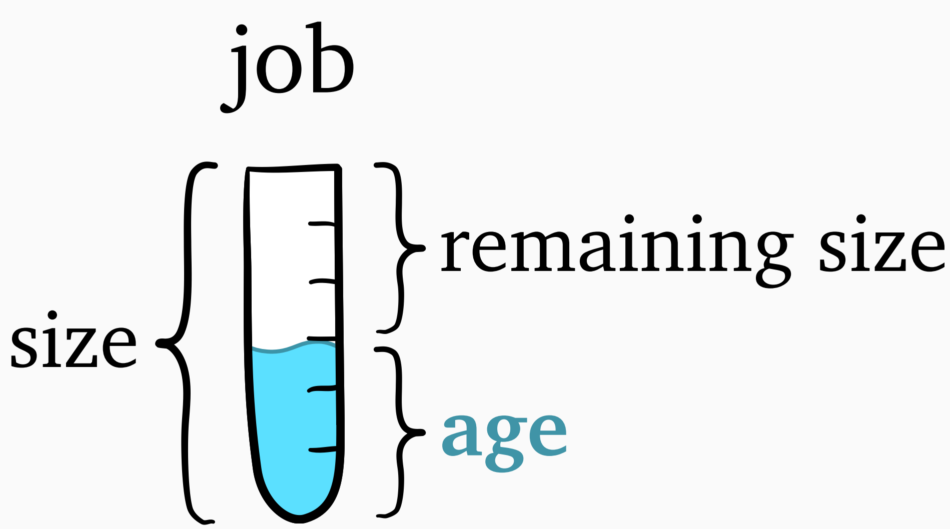
What is scheduling?



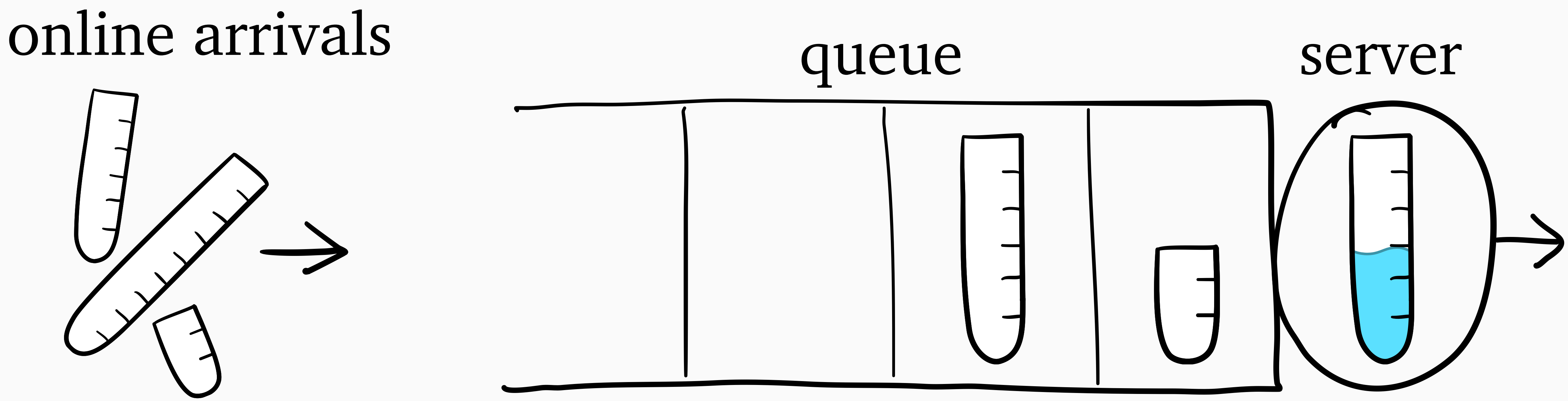


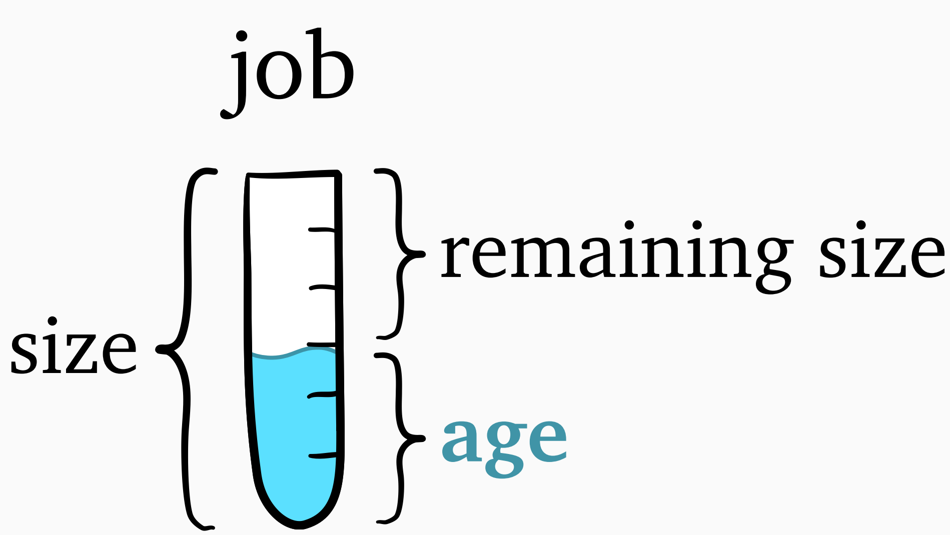
What is scheduling?



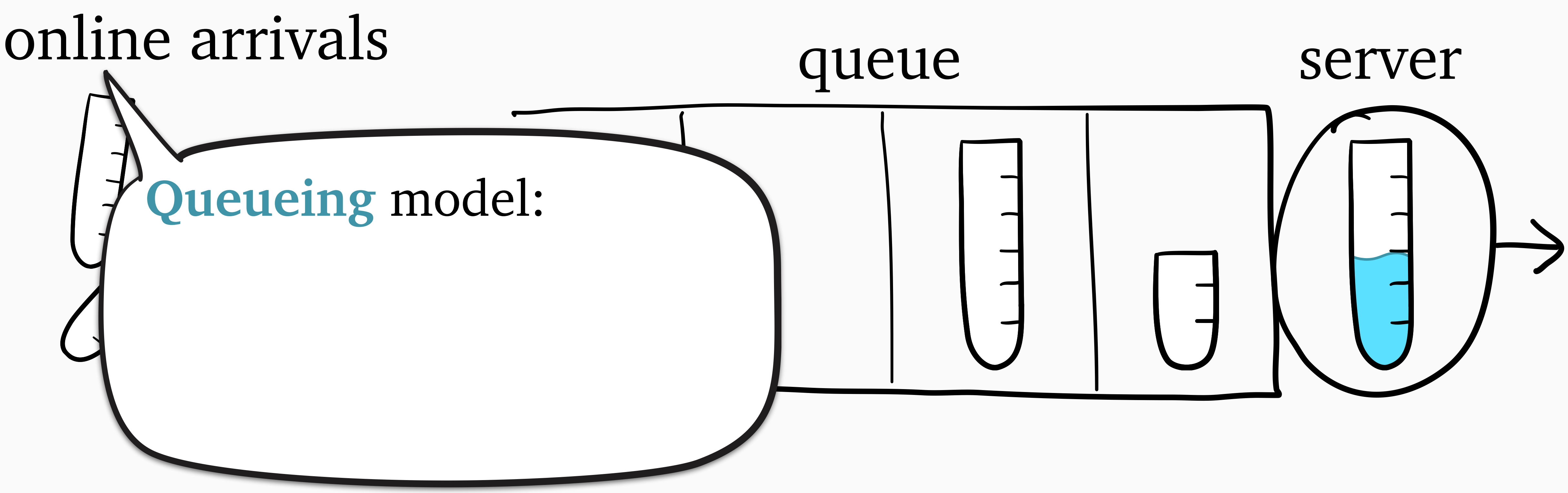


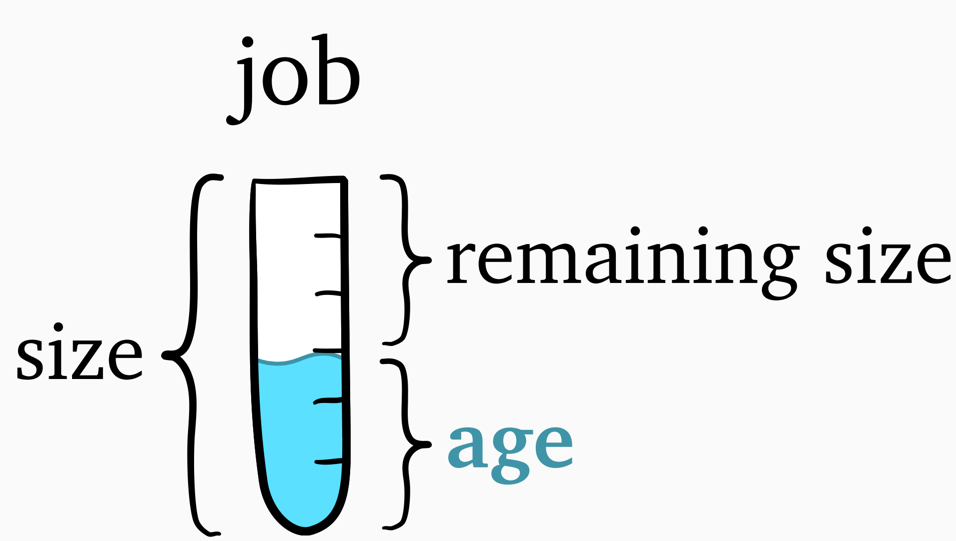
What is scheduling?



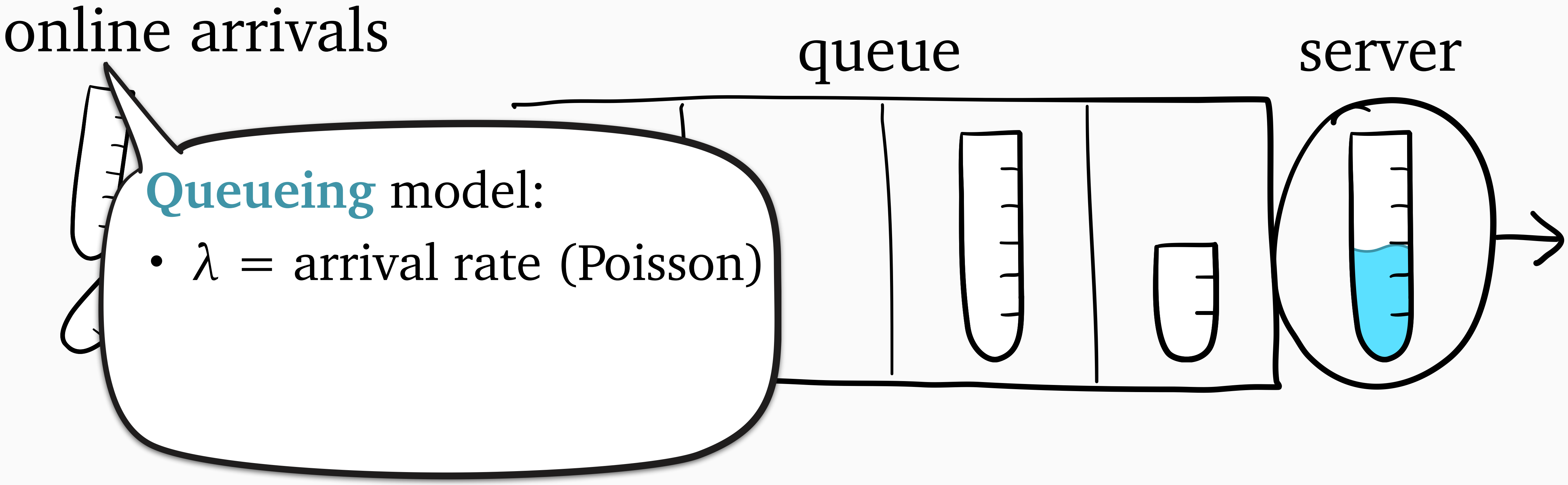


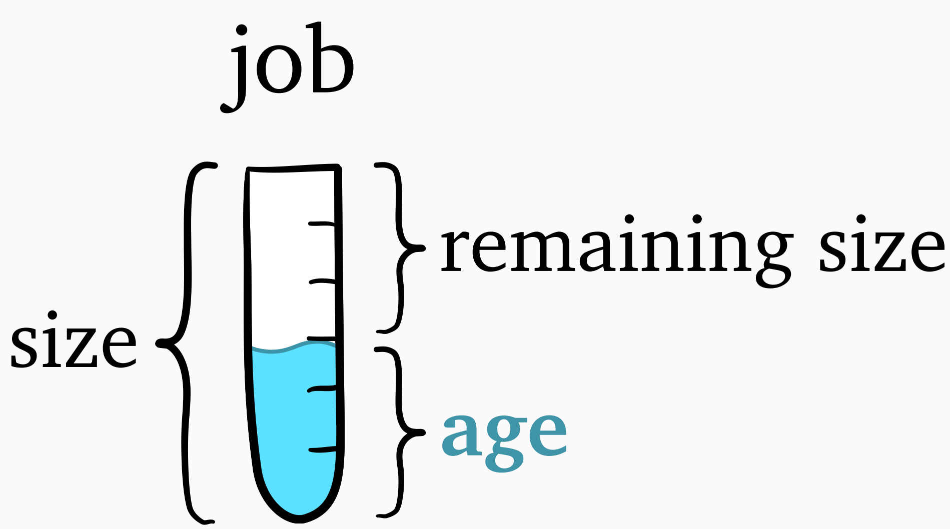
What is scheduling?



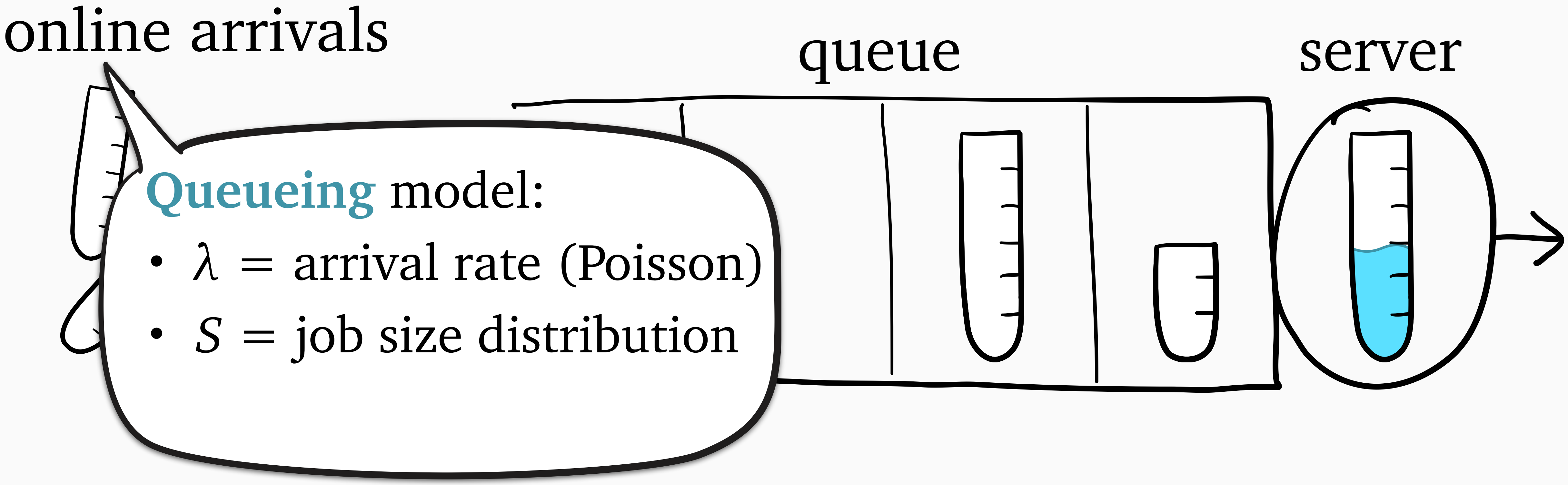


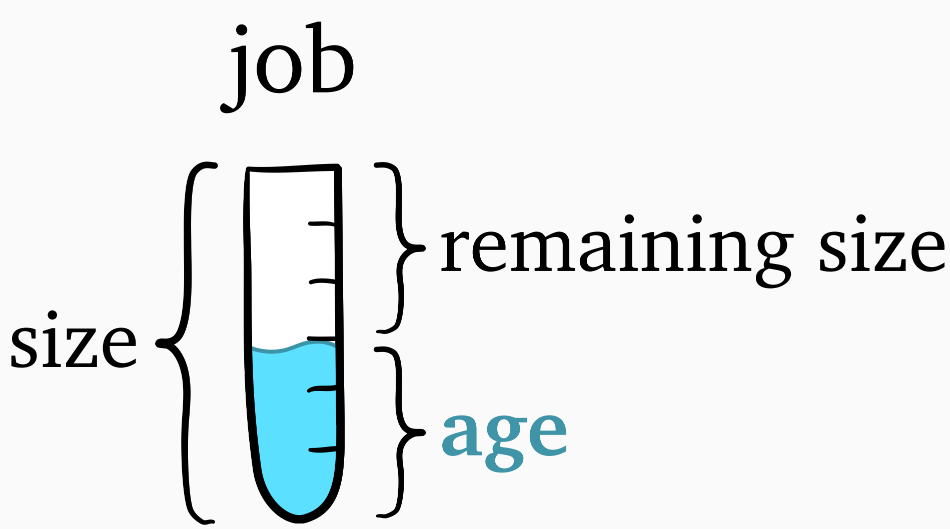
What is scheduling?



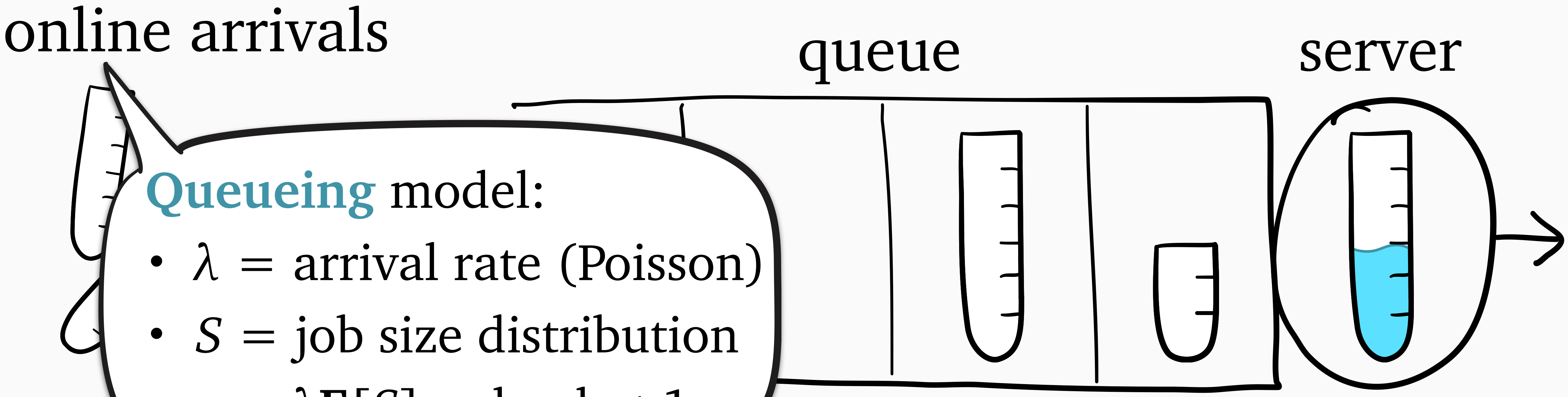


What is scheduling?



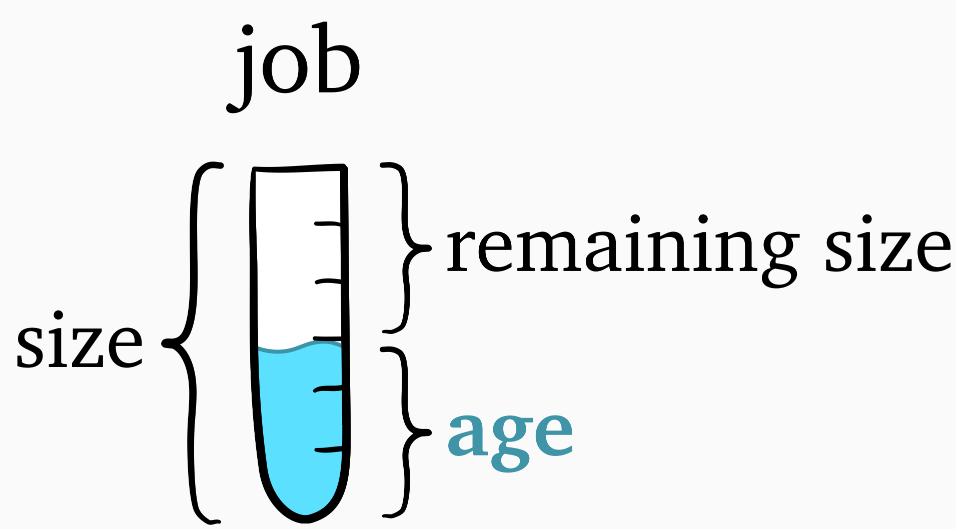


What is scheduling?

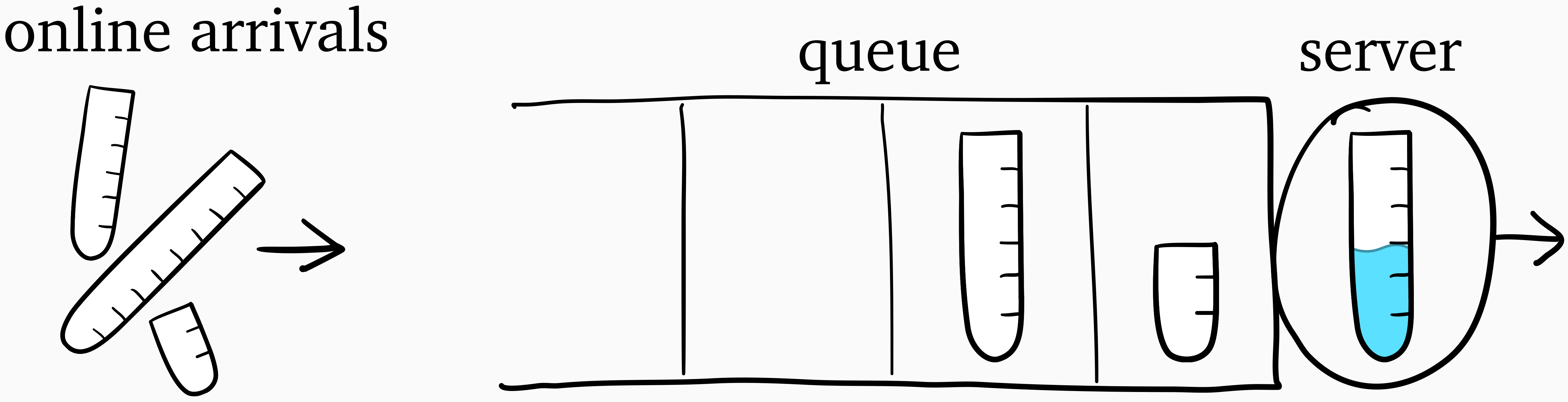


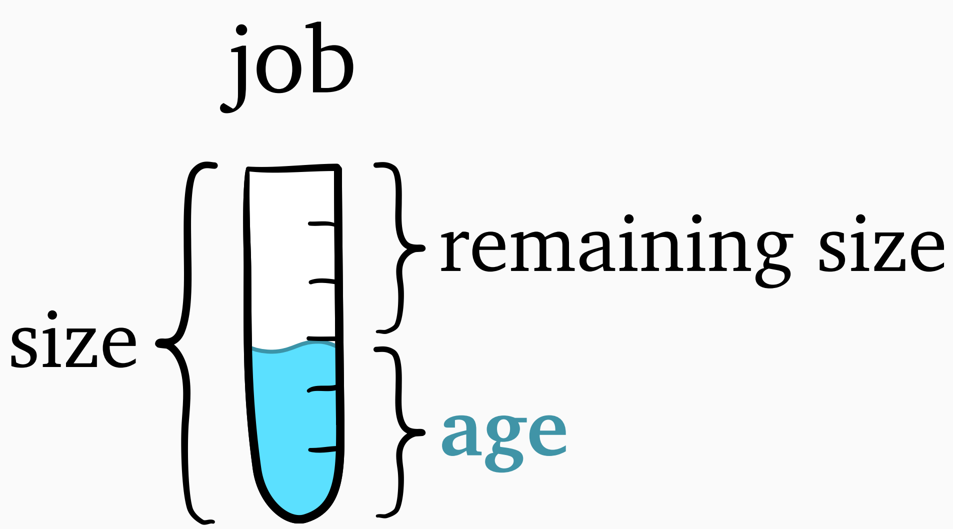
Queueing model:

- λ = arrival rate (Poisson)
- S = job size distribution
- $\rho = \lambda \mathbf{E}[S]$ = load < 1

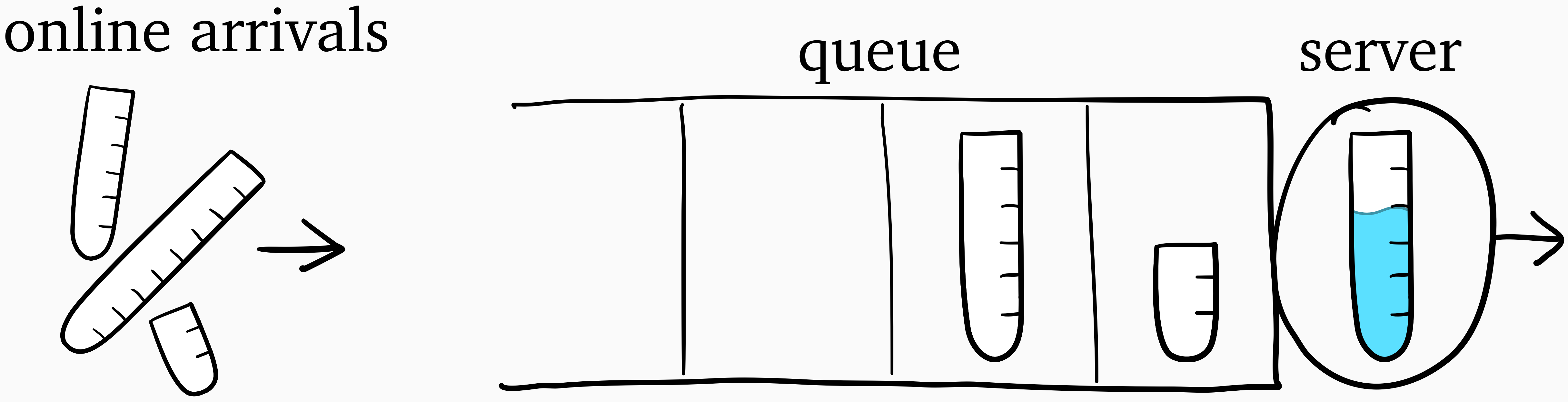


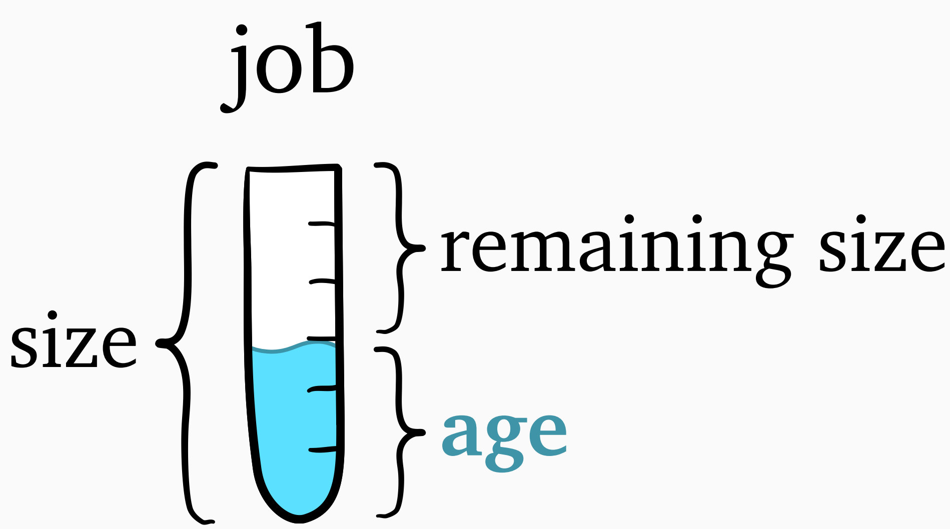
What is scheduling?



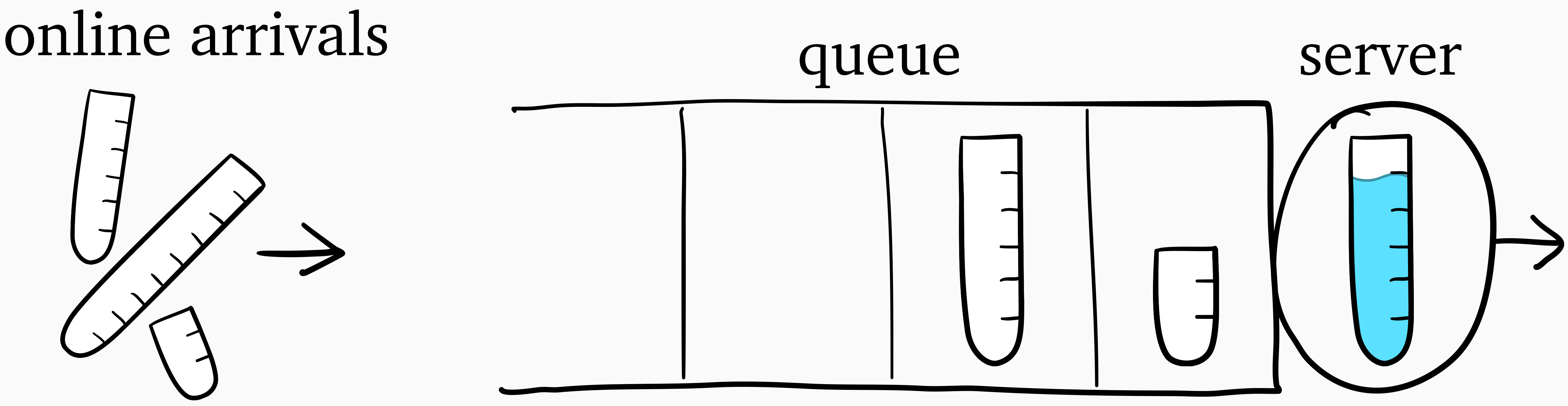


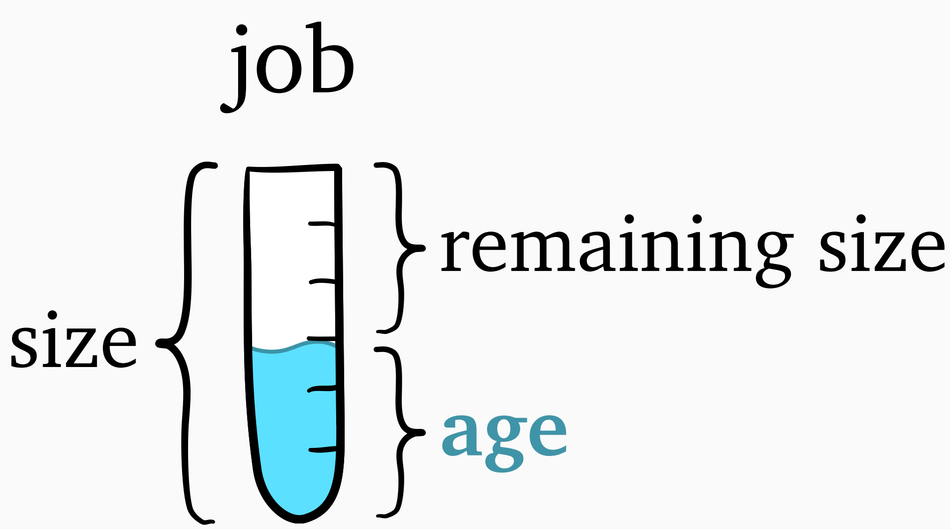
What is scheduling?



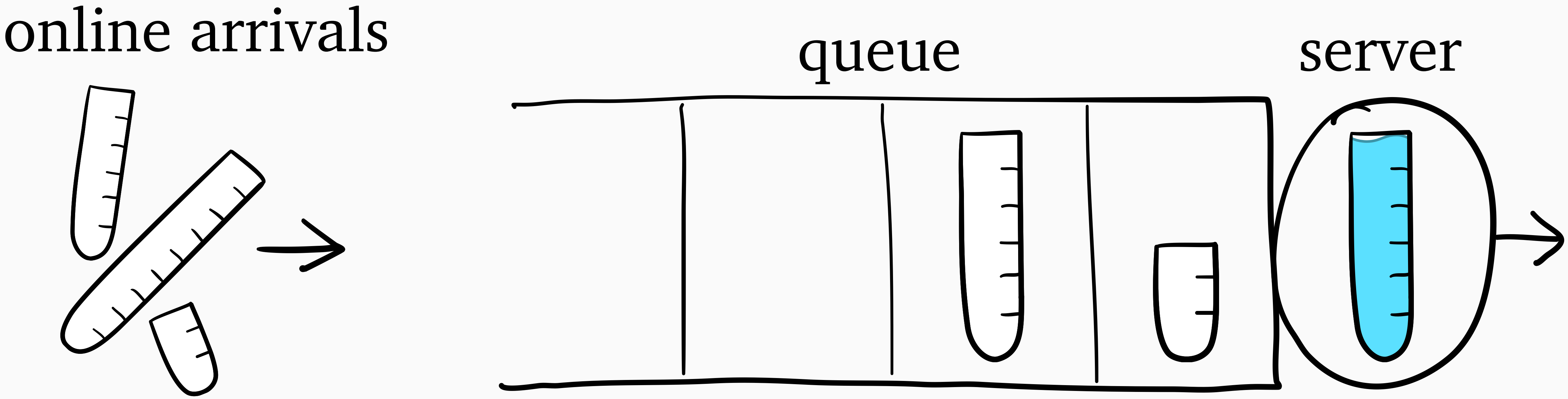


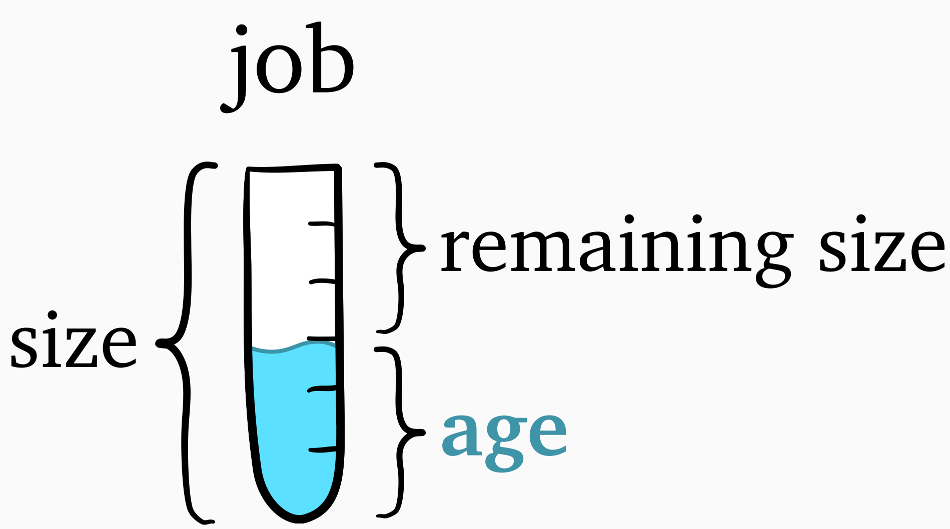
What is scheduling?



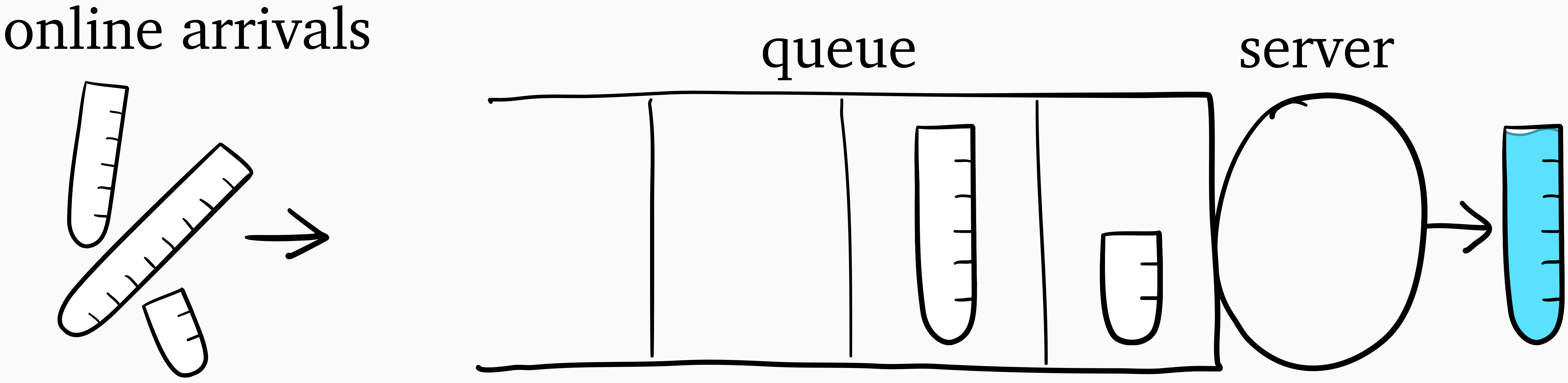


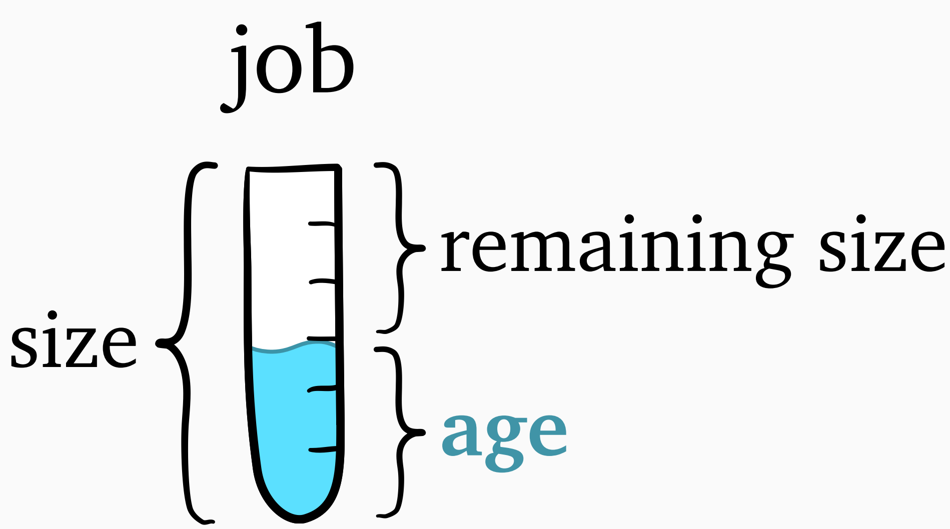
What is scheduling?



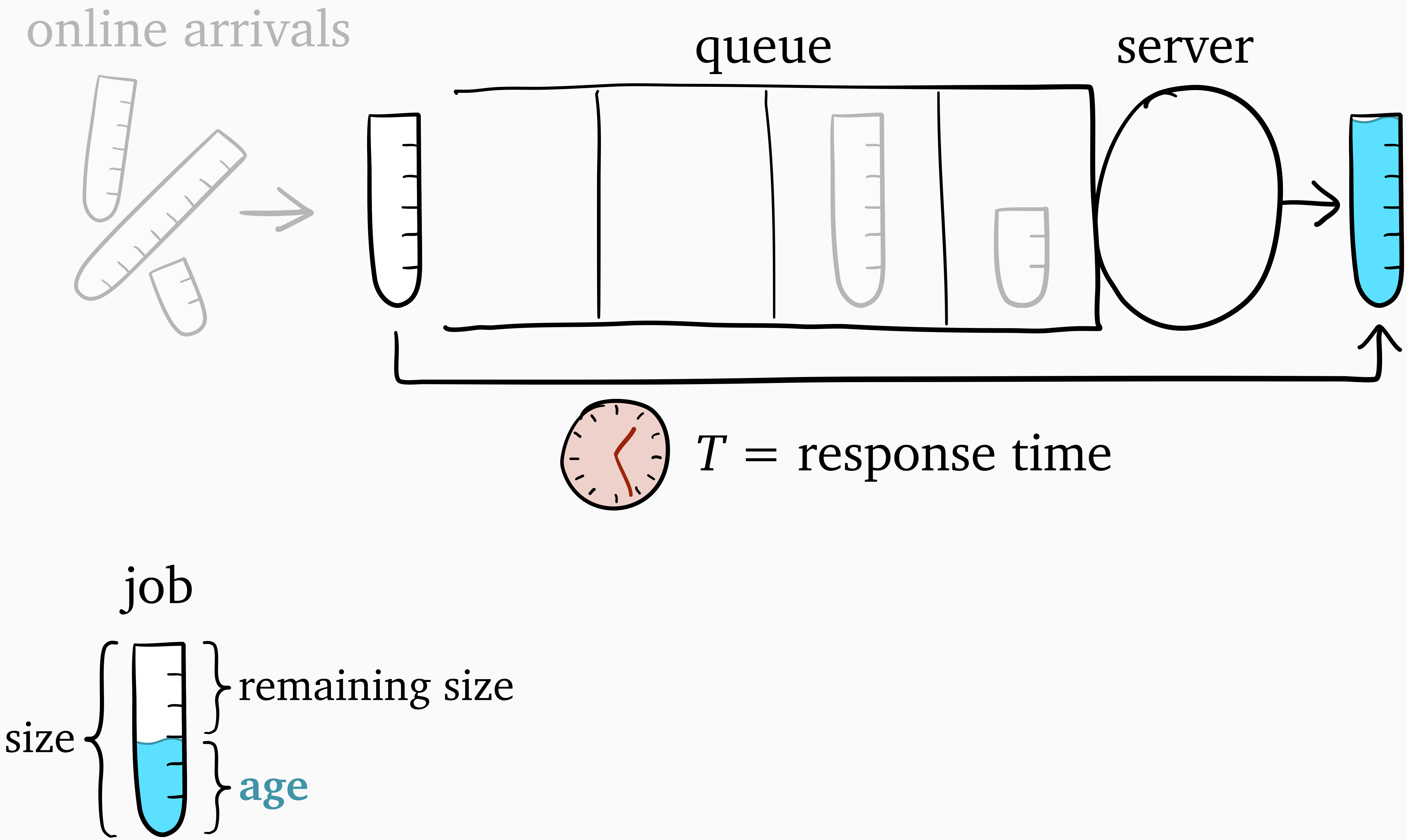


What is scheduling?

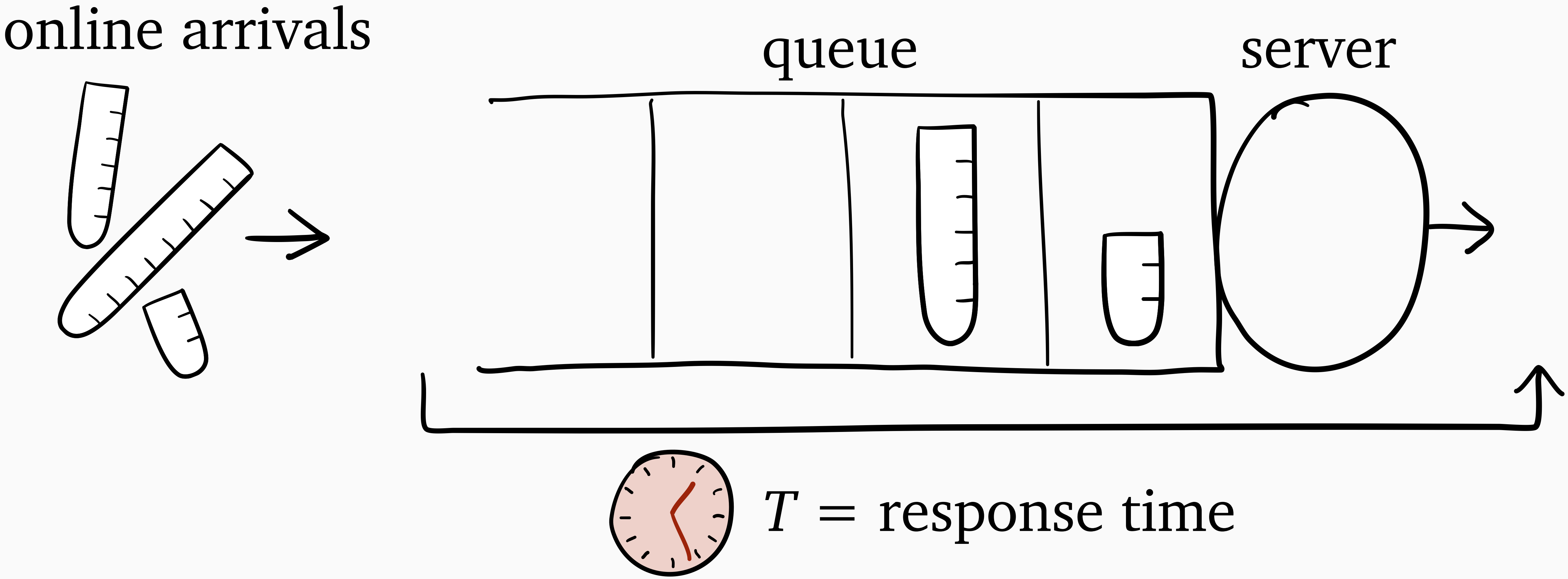


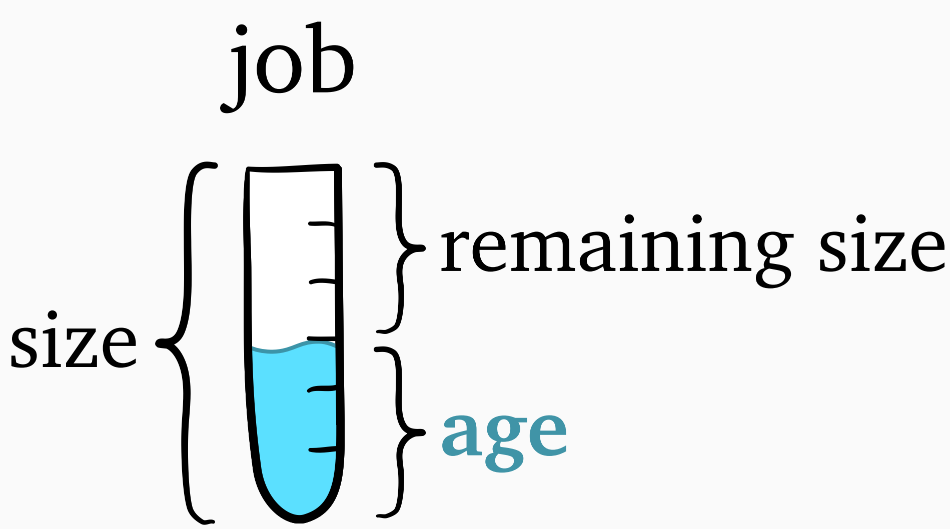


What is scheduling?

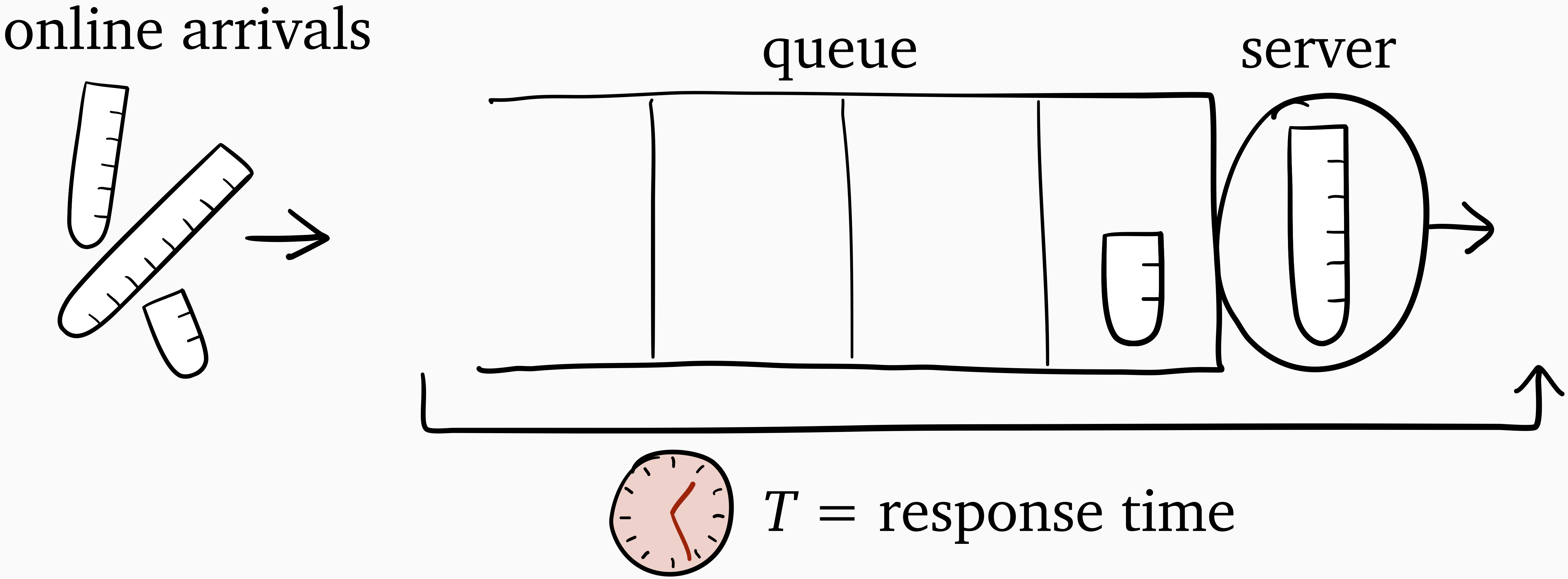


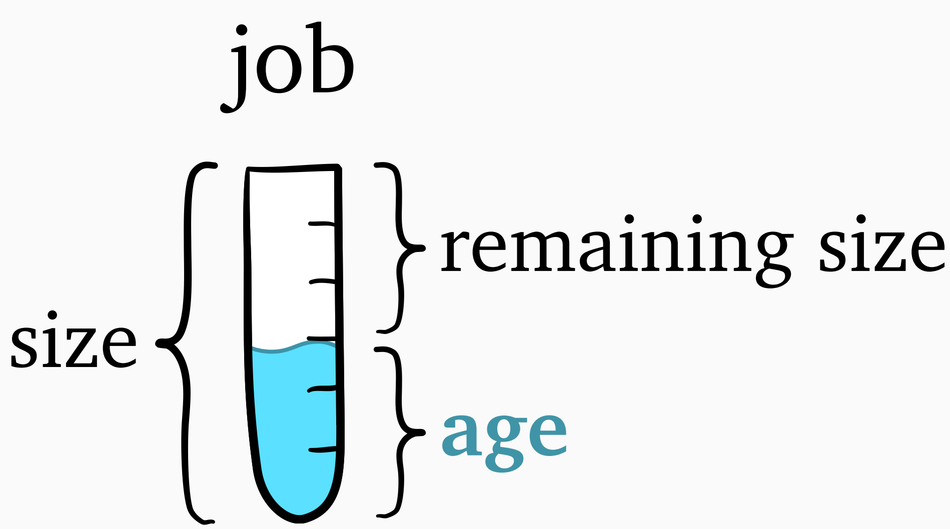
What is scheduling?



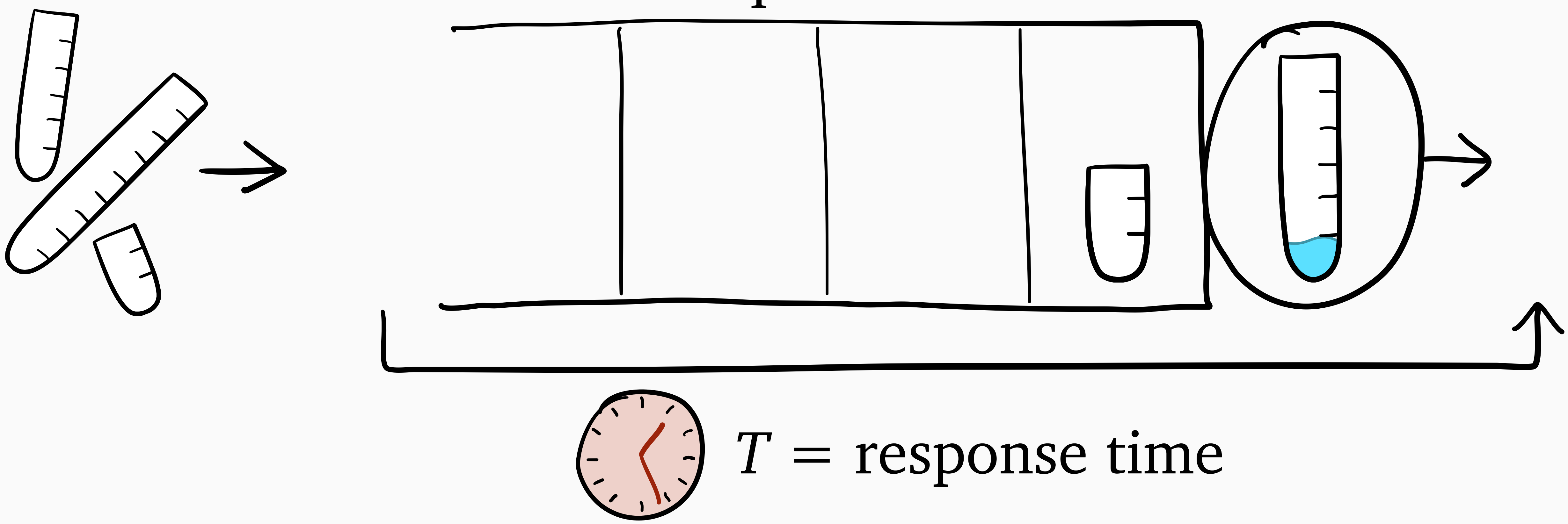


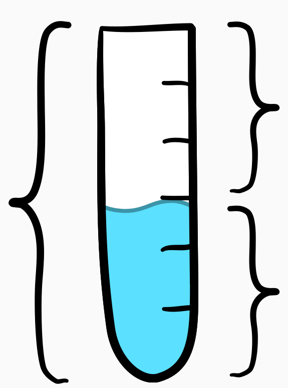
What is scheduling?



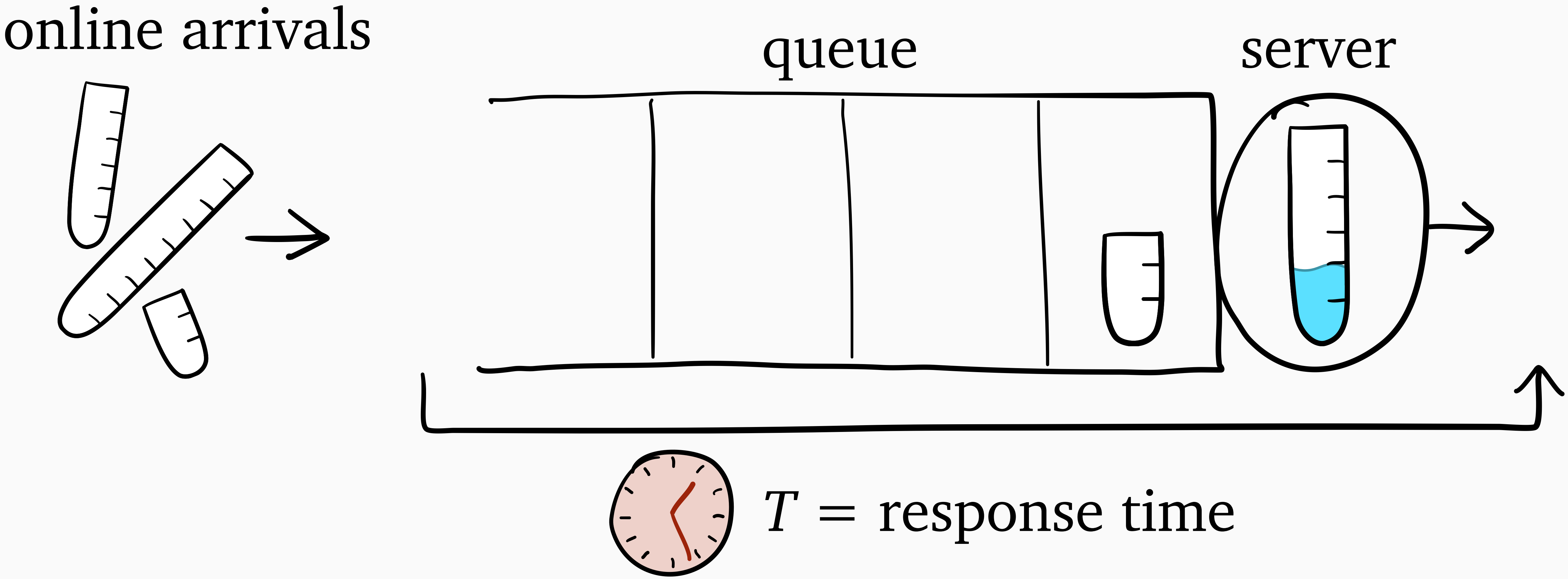


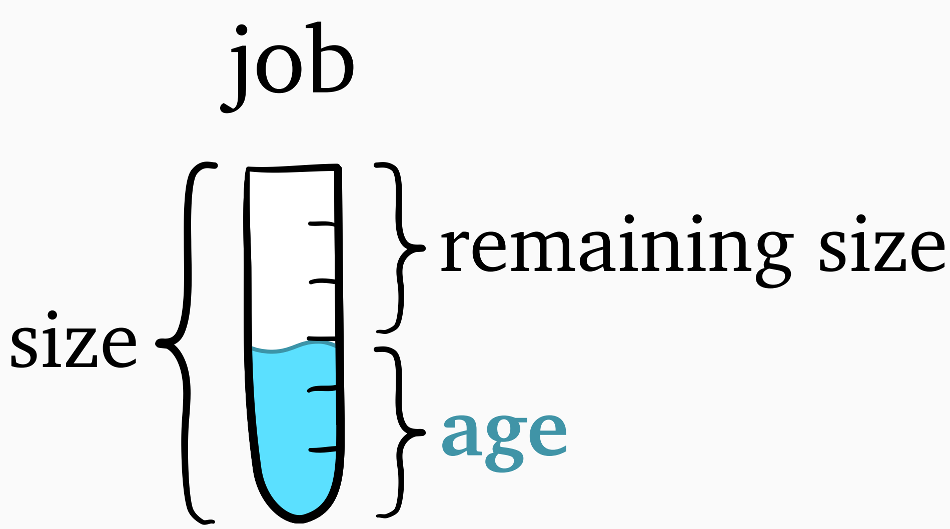
What is scheduling?



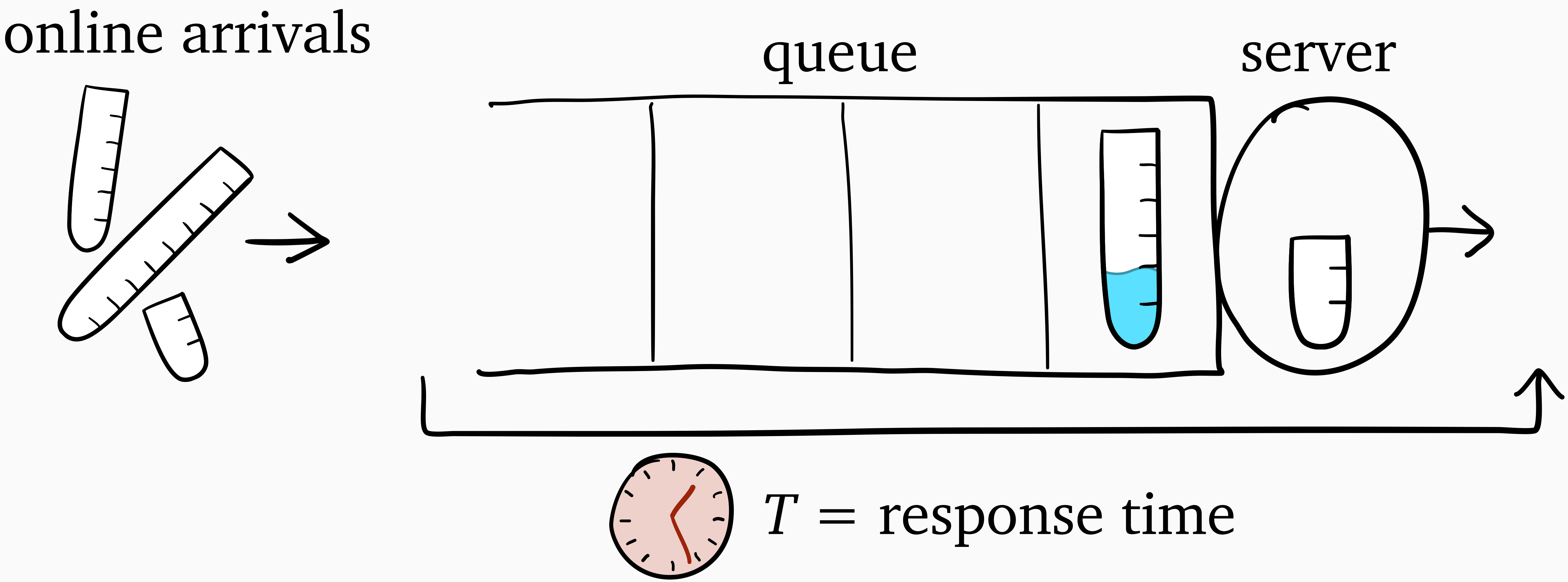


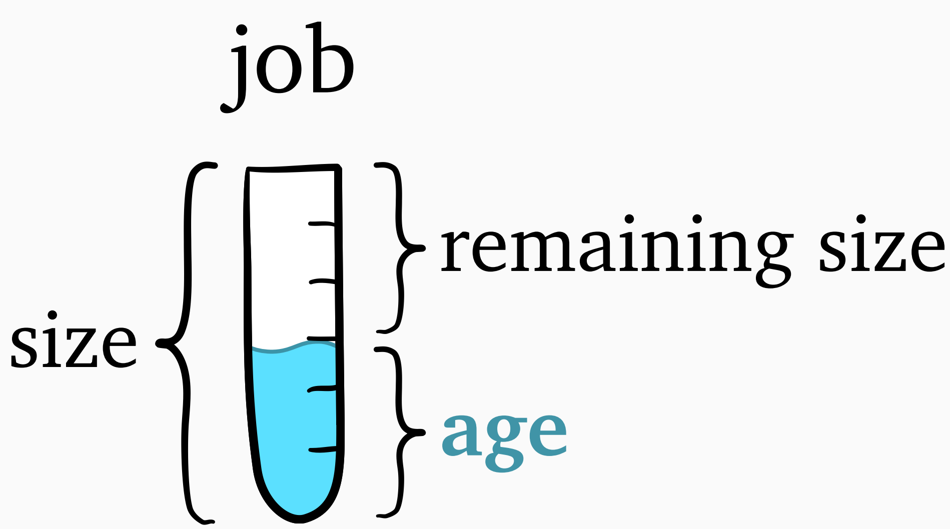
What is scheduling?



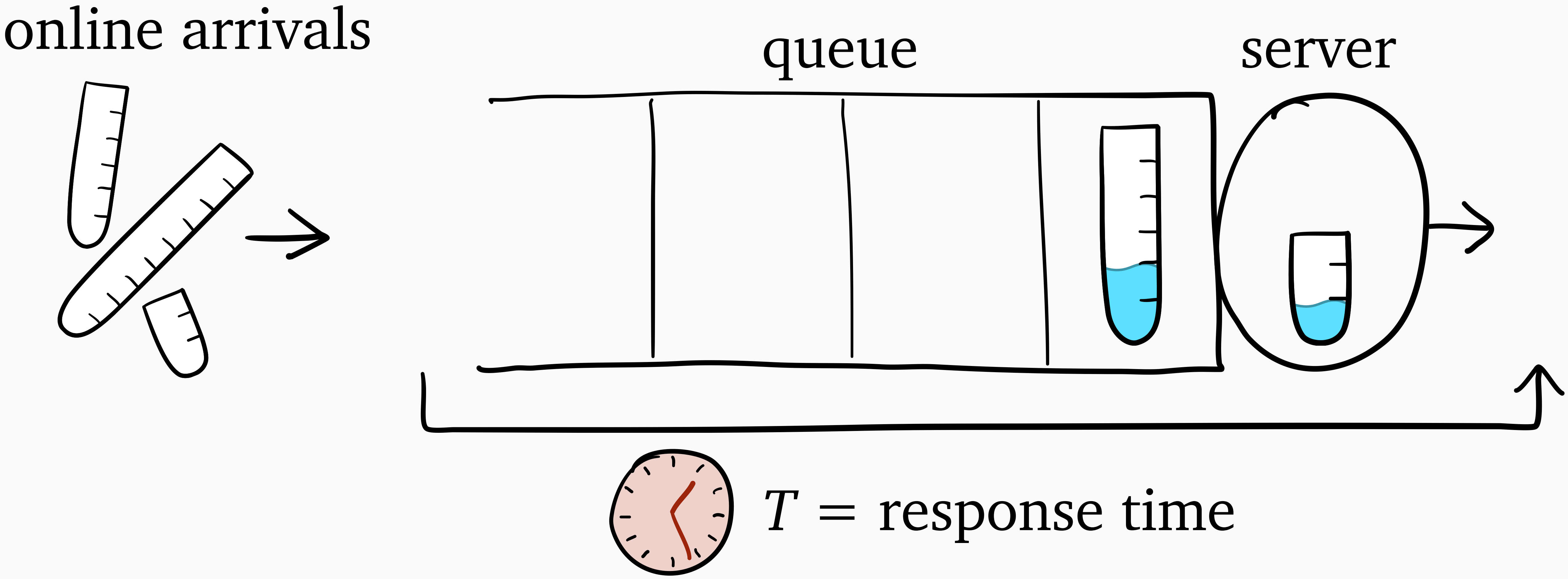


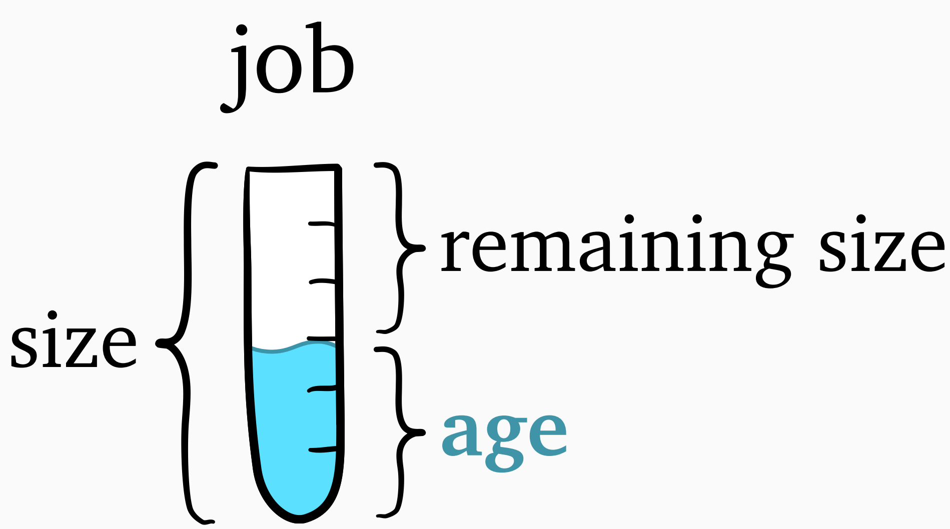
What is scheduling?



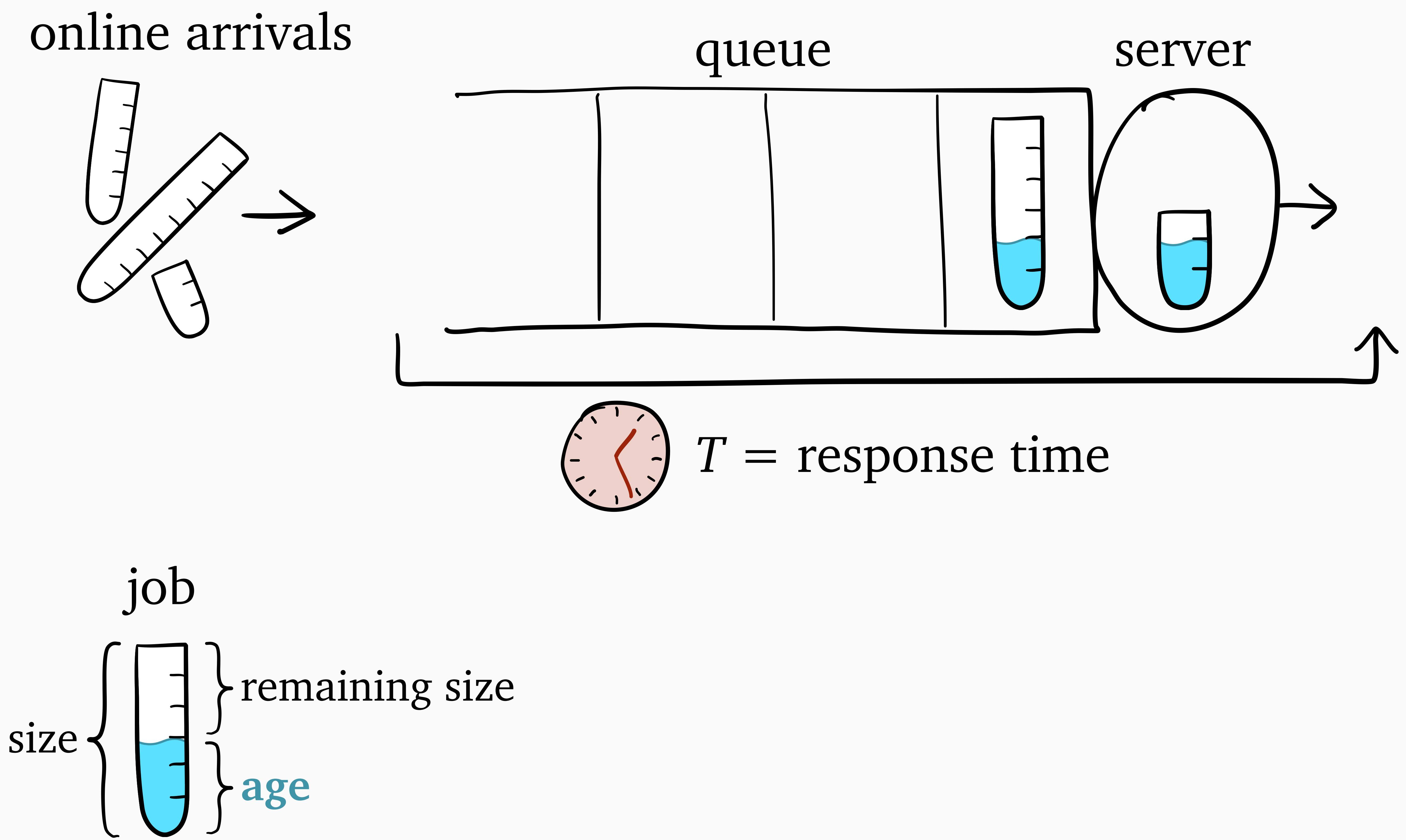


What is scheduling?

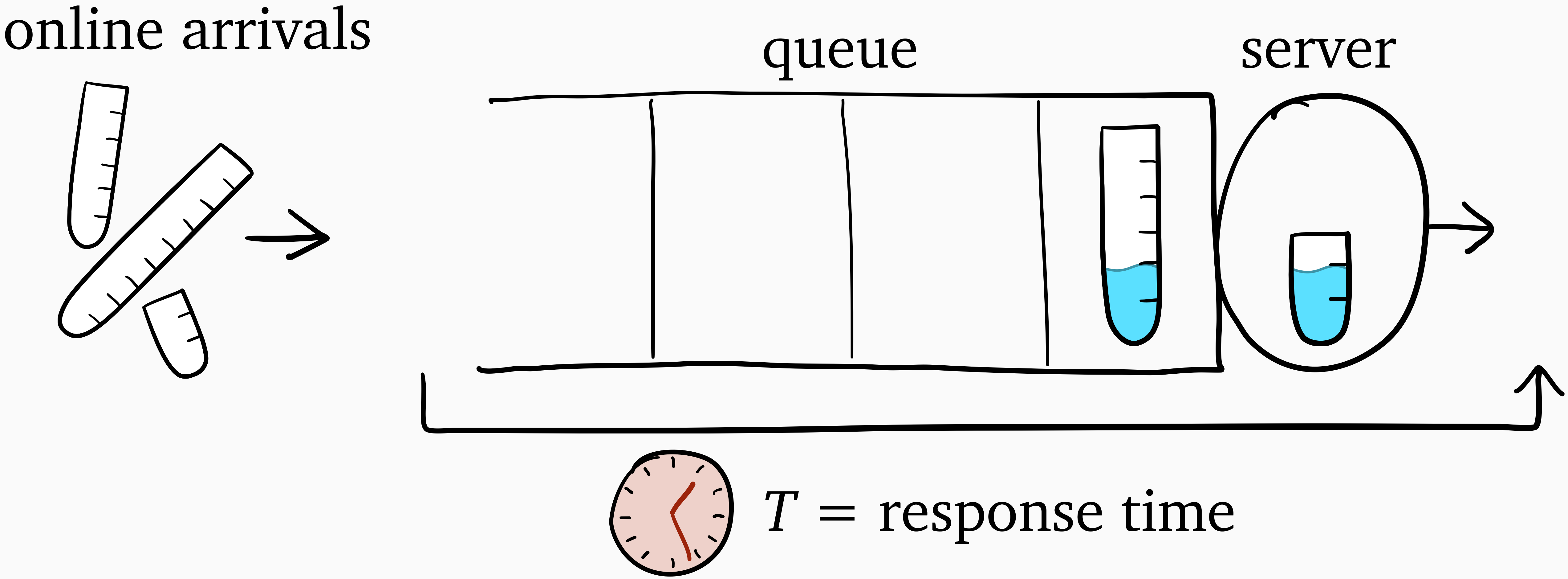


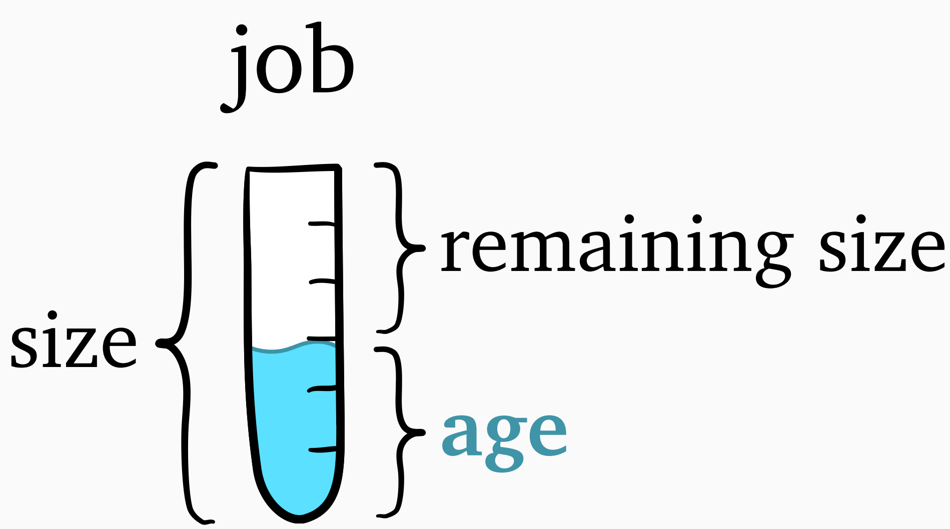


What is scheduling?



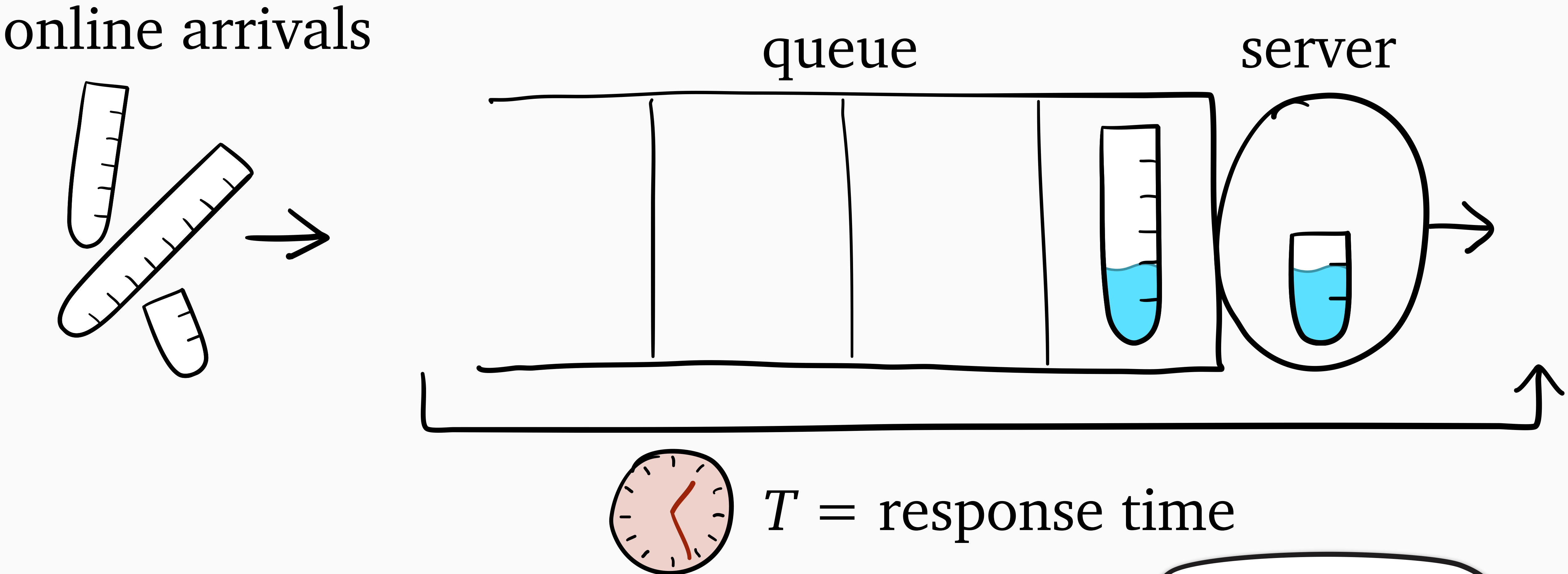
What is scheduling?

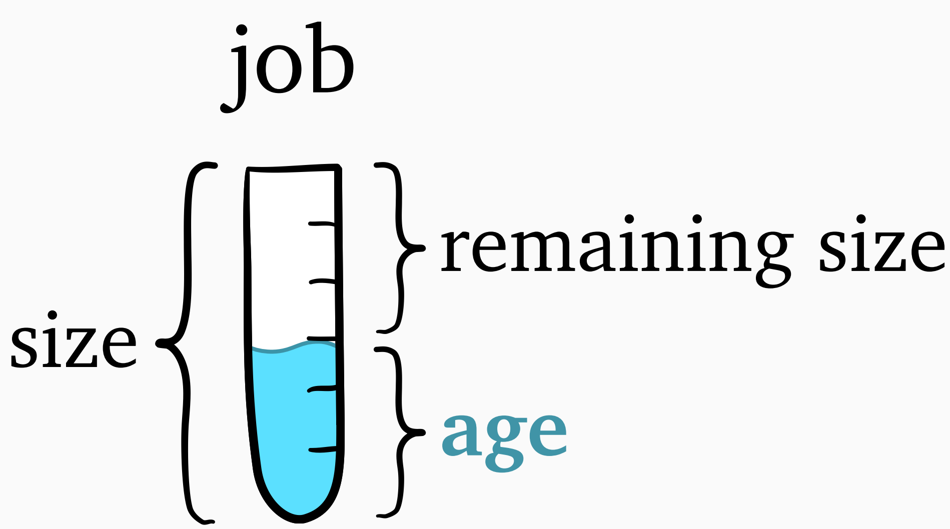


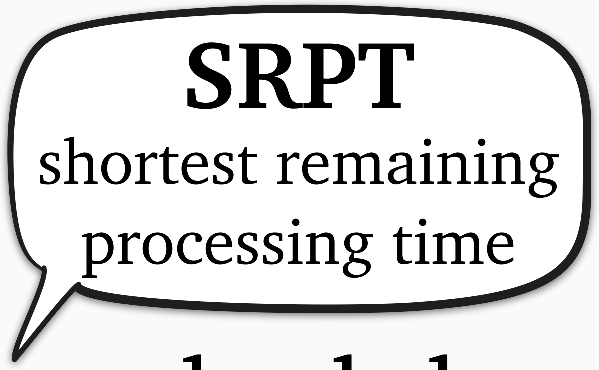


Question: schedule to minimize $\mathbf{E}[T]$?

What is scheduling?



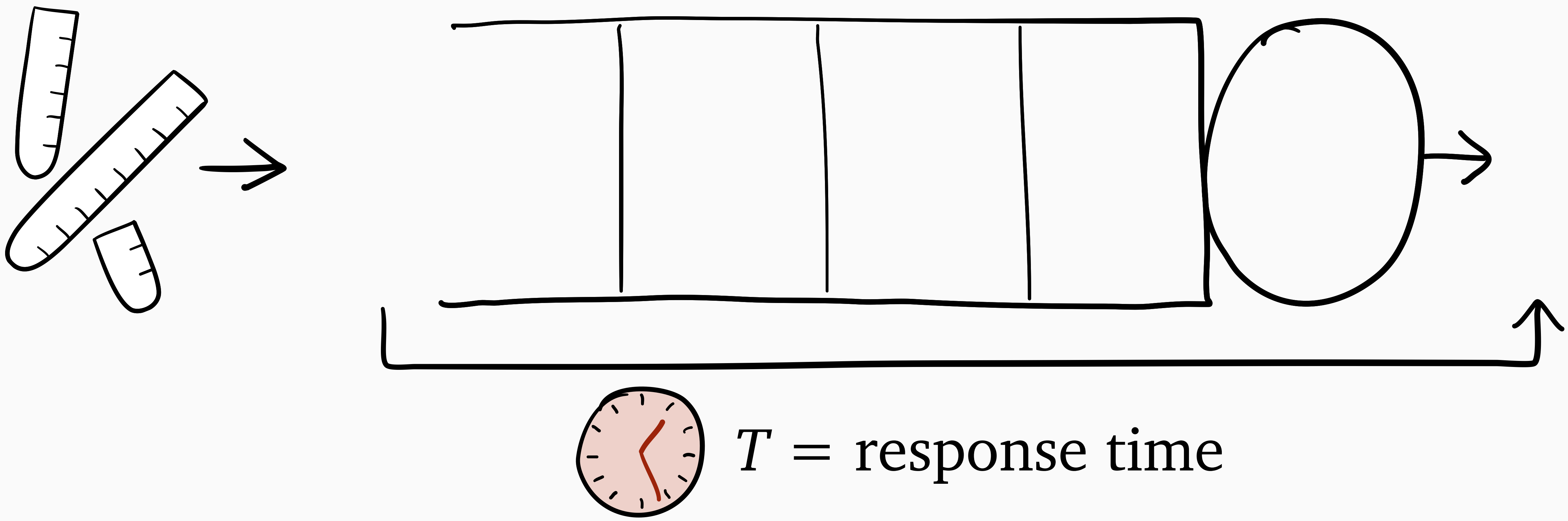




Question: schedule to minimize $\mathbf{E}[T]$?

Multiserver scheduling

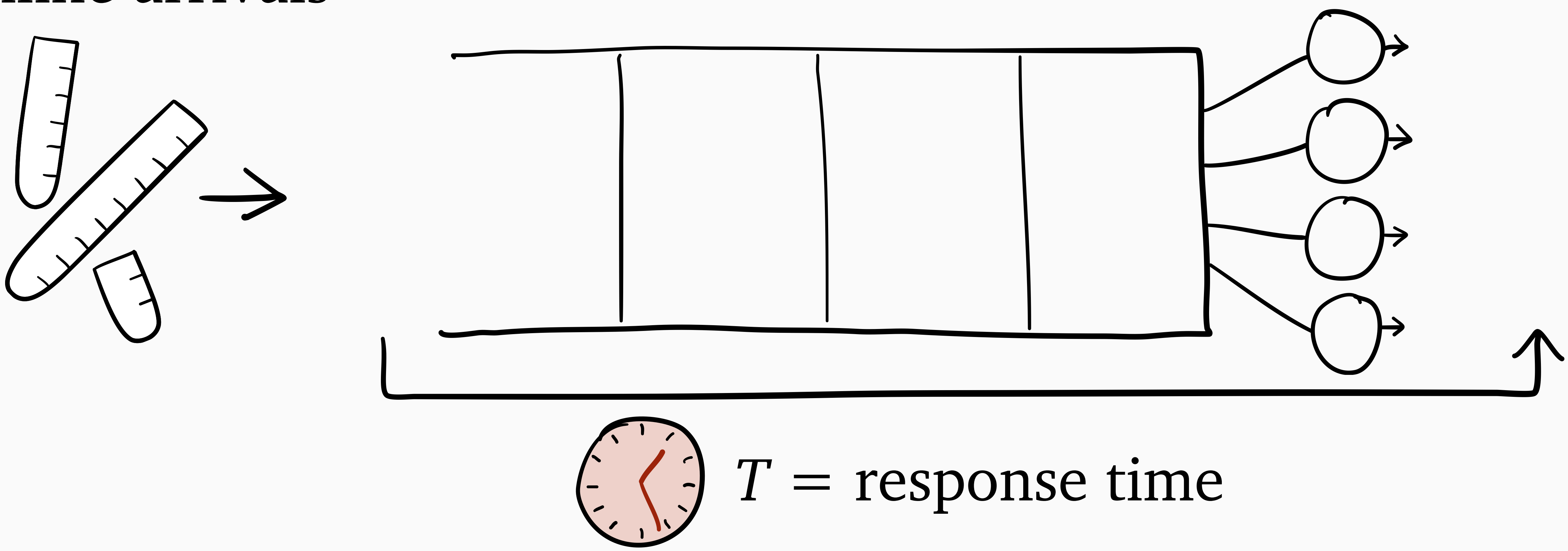
online arrivals



T = response time

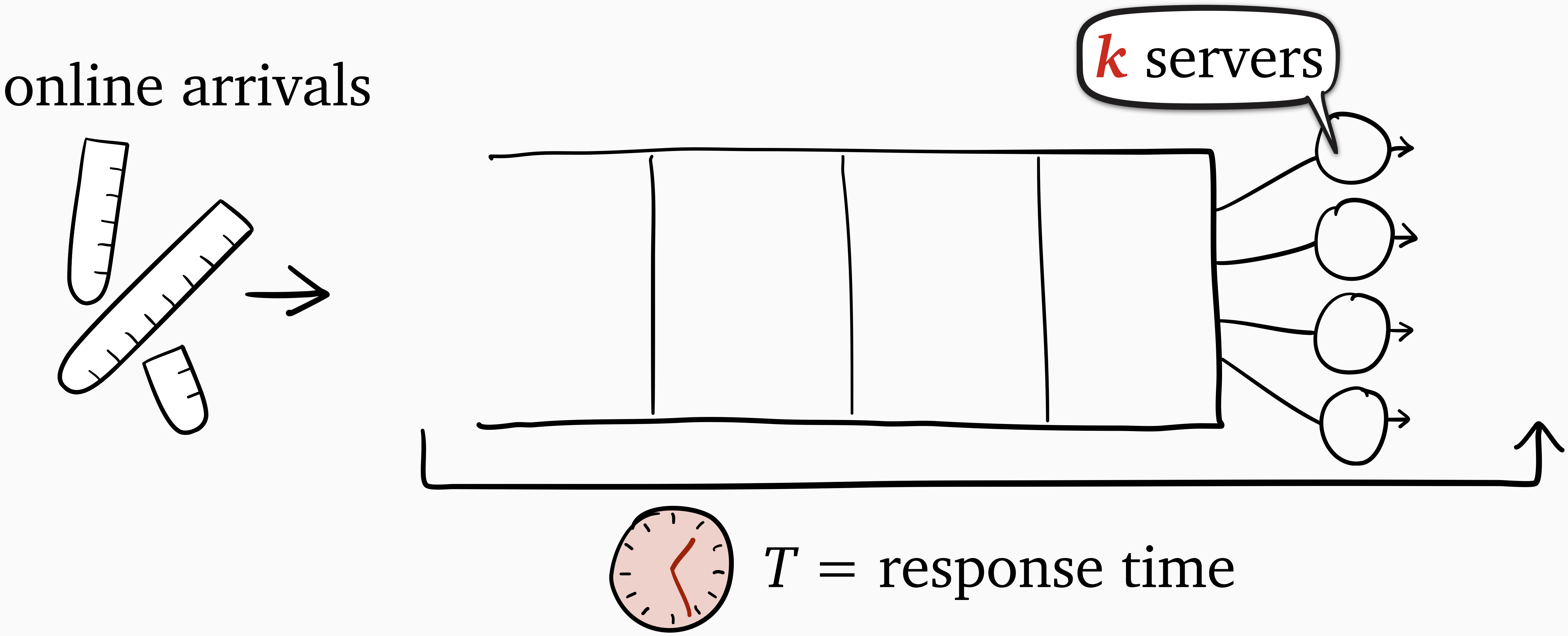
Multiserver scheduling

online arrivals

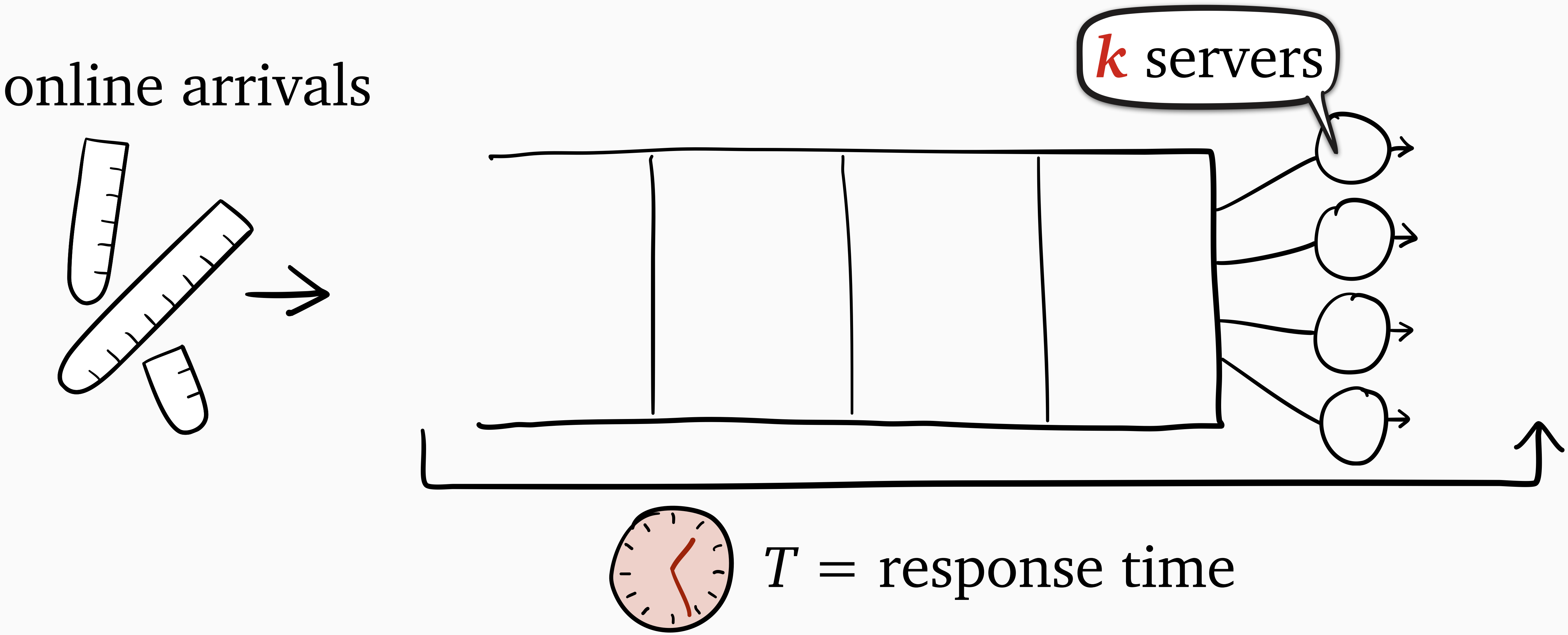


T = response time

Multiserver scheduling

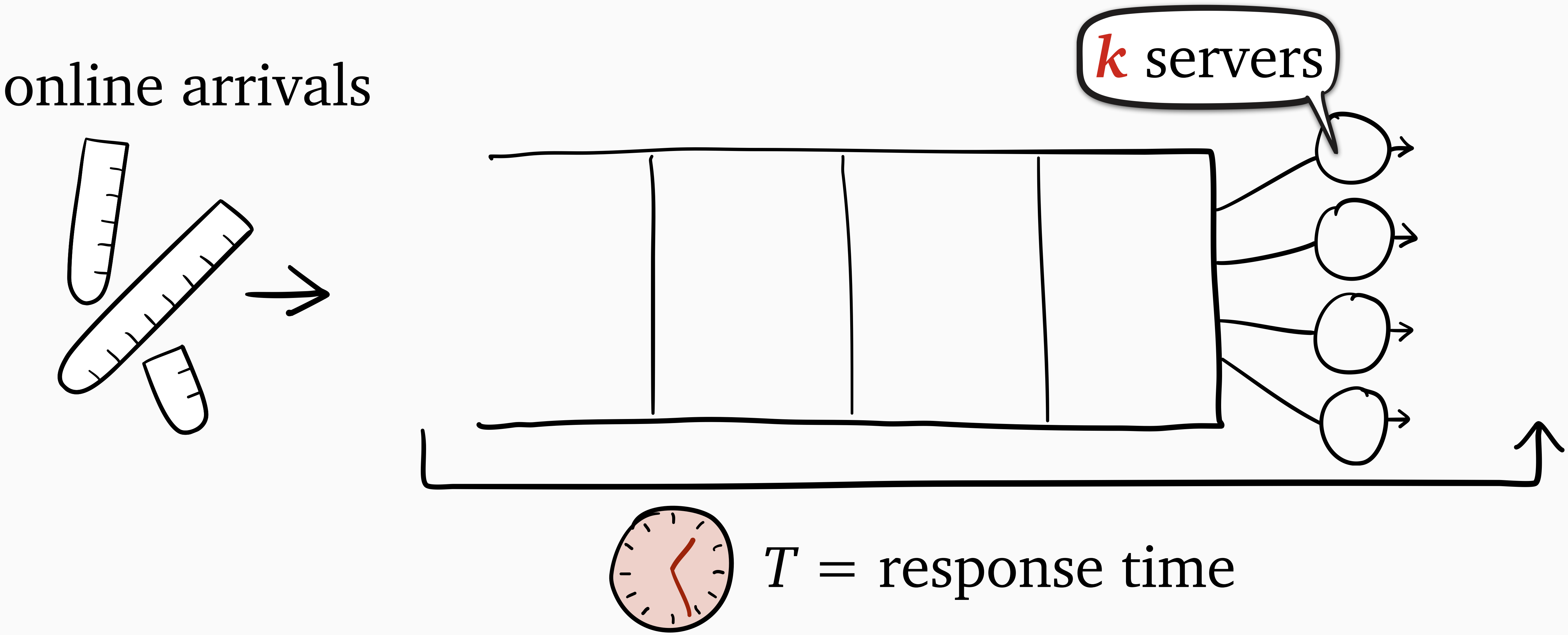


Multiserver scheduling



SRPT-1 (single-server): serves job of least remaining size

Multiserver scheduling



SRPT-1 (**single-server**): serves job of least remaining size

SRPT-*k* (**multiserver**): serves *k* jobs of least remaining size

How good is SRPT-k?

How good is SRPT-***k***?

TCS [Leonardi & Raz, 2007]: not great, but best possible

How good is SRPT-*k*?

TCS [Leonardi & Raz, 2007]: not great, but best possible

Theorem: competitive ratio of SRPT-*k* vs. offline OPT-*k* is

$$\frac{\mathbf{E}[T_{\text{SRPT-}k}]}{\mathbf{E}[T_{\text{OPT-}k}]} \leq O\left(\min\left\{\log \frac{\#\text{ jobs}}{k}, \log \frac{\text{max size}}{\text{min size}}\right\}\right)$$

with matching lower bound

How good is SRPT-*k*?

TCS [Leonardi & Raz, 2007]: not great, but best possible

Theorem: competitive ratio of SRPT-*k* vs. offline OPT-*k* is

$$\frac{\mathbf{E}[T_{\text{SRPT-}k}]}{\mathbf{E}[T_{\text{OPT-}k}]} \leq O\left(\min\left\{\log \frac{\#\text{ jobs}}{k}, \log \frac{\text{max size}}{\text{min size}}\right\}\right)$$

with matching lower bound

How good is SRPT-*k*?

TCS [Leonardi & Raz, 2007]: not great, but best possible

Theorem: competitive ratio of SRPT-*k* vs. offline OPT-*k* is

$$\frac{\mathbf{E}[T_{\text{SRPT-}k}]}{\mathbf{E}[T_{\text{OPT-}k}]} \leq O\left(\min\left\{\log\frac{\#\text{ jobs}}{k}, \log\frac{\text{max size}}{\text{min size}}\right\}\right)$$

with matching lower bound

How good is SRPT-k?

TCS [Leonardi & Raz, 2007]: not great, but best possible

Theorem: competitive ratio of SRPT-k vs. offline OPT-k is

$$\frac{\mathbf{E}[T_{\text{SRPT-}k}]}{\mathbf{E}[T_{\text{OPT-}k}]} \leq O\left(\min\left\{\log\frac{\#\text{ jobs}}{k}, \log\frac{\text{max size}}{\text{min size}}\right\}\right)$$

with matching lower bound

How good is SRPT-k?

TCS [Leonardi & Raz, 2007]: not great, but best possible

Theorem: competitive ratio of SRPT-k vs. offline OPT-k is

$$\frac{\mathbf{E}[T_{\text{SRPT-}k}]}{\mathbf{E}[T_{\text{OPT-}k}]} \leq O\left(\min\left\{\log \frac{\#\text{ jobs}}{k}, \log \frac{\text{max size}}{\text{min size}}\right\}\right)$$

with matching lower bound

How good is SRPT-*k*?

TCS [Leonardi & Raz, 2007]: not great, but best possible

Theorem: competitive ratio of SRPT-*k* vs. offline OPT-*k* is

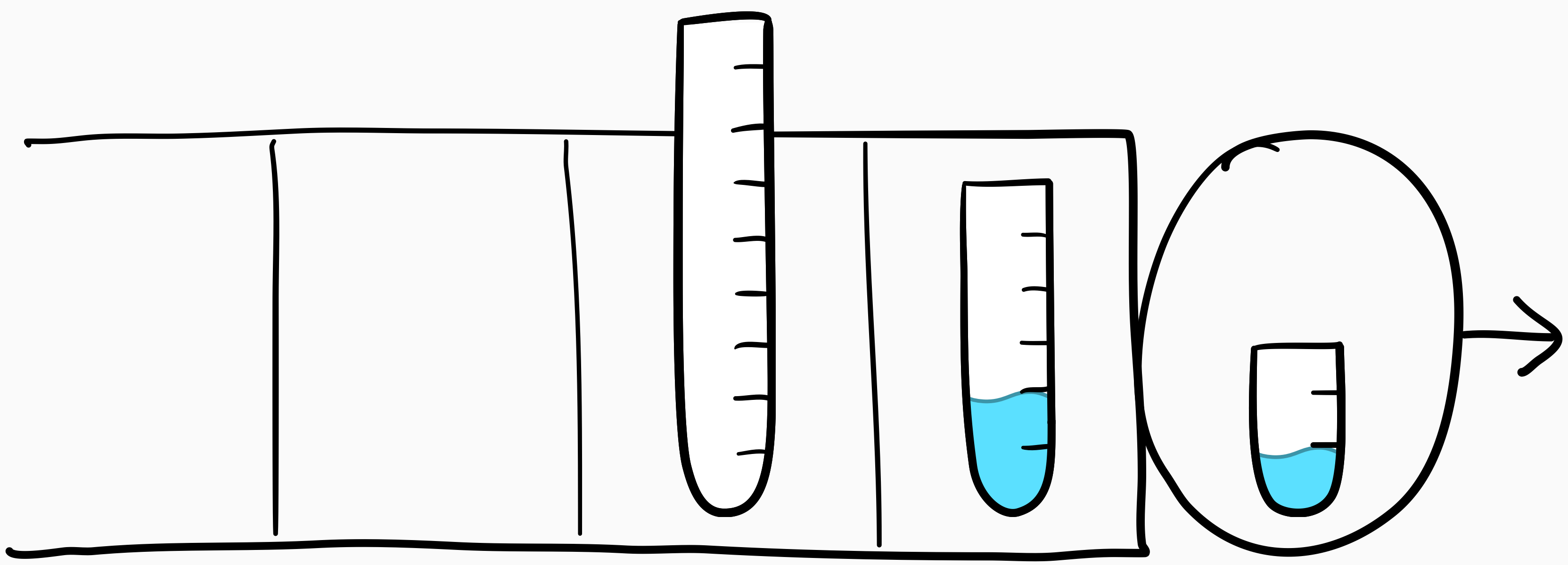
$$\frac{\mathbf{E}[T_{\text{SRPT-}k}]}{\mathbf{E}[T_{\text{OPT-}k}]} \leq O\left(\min\left\{\log \frac{\#\text{ jobs}}{k}, \log \frac{\text{max size}}{\text{min size}}\right\}\right)$$

with matching lower bound

Queueing: decades-old open problem!

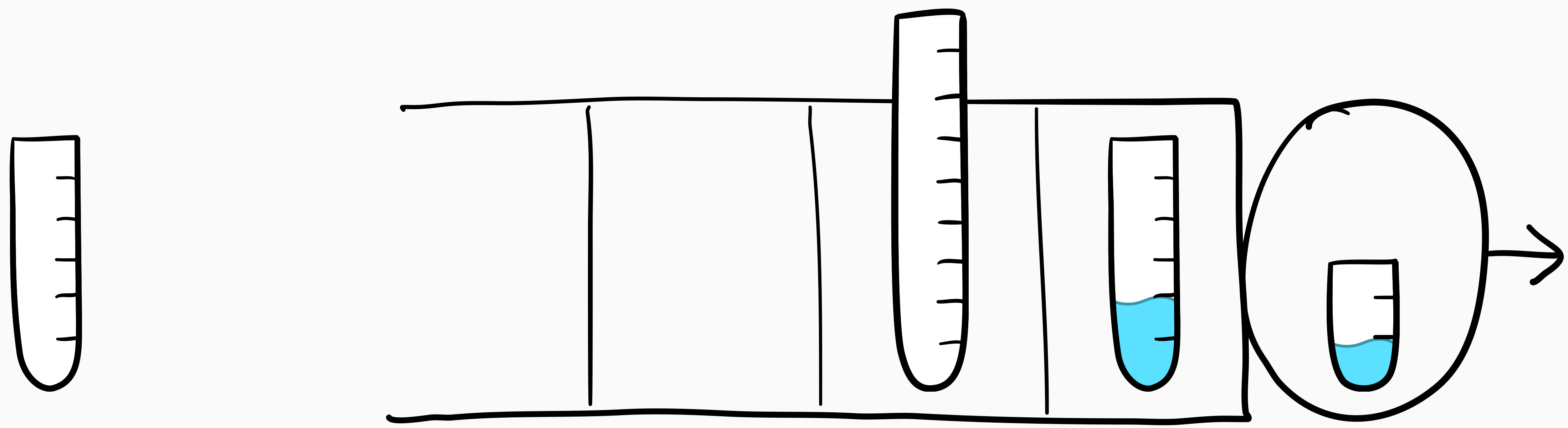
Queueing analysis of SRPT-1

[Schrage & Miller, 1966]



Queueing analysis of SRPT-1

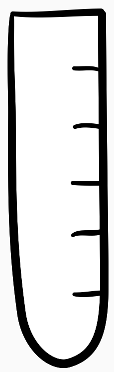
[Schrage & Miller, 1966]



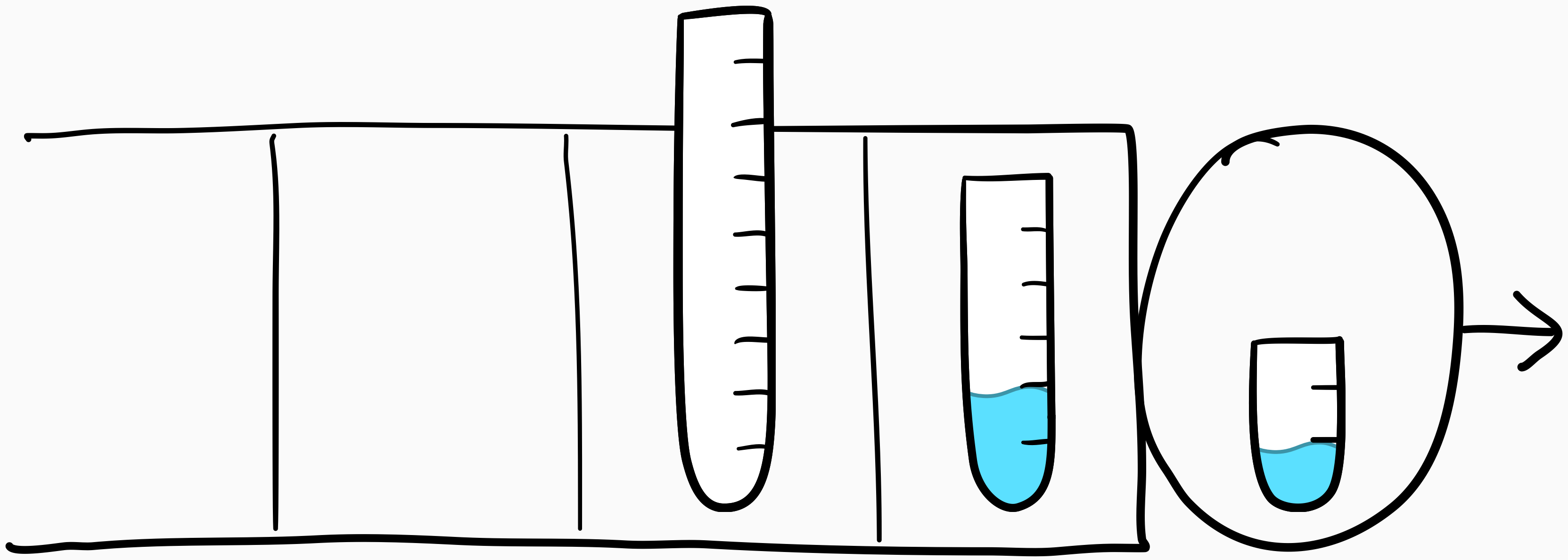
“tagged job”

Queueing analysis of SRPT-1

[Schrage & Miller, 1966]



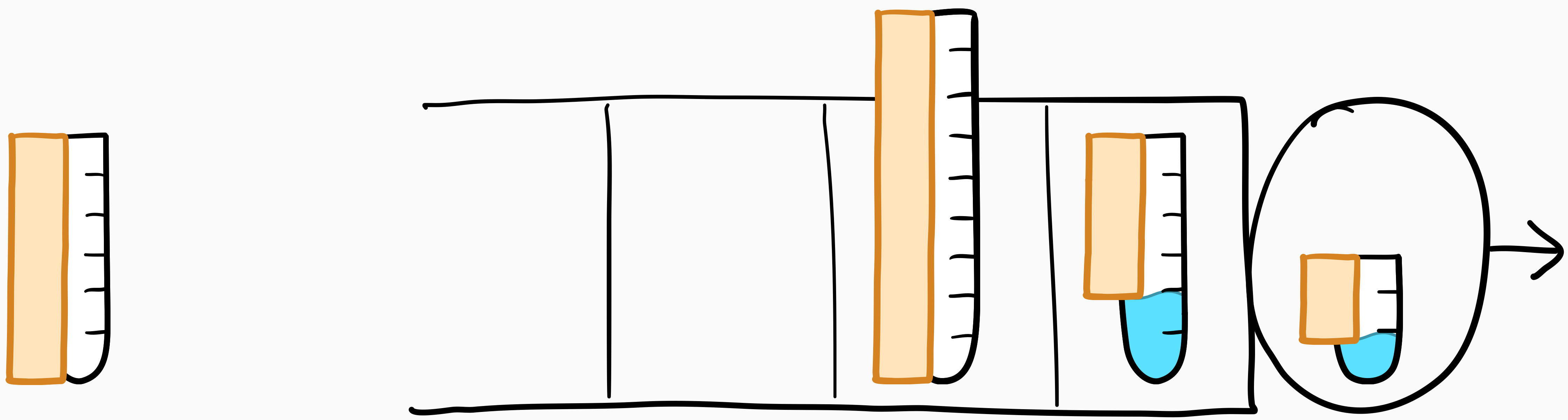
“tagged job”



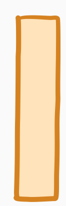
random system state

Queueing analysis of SRPT-1

[Schrage & Miller, 1966]



“tagged job”

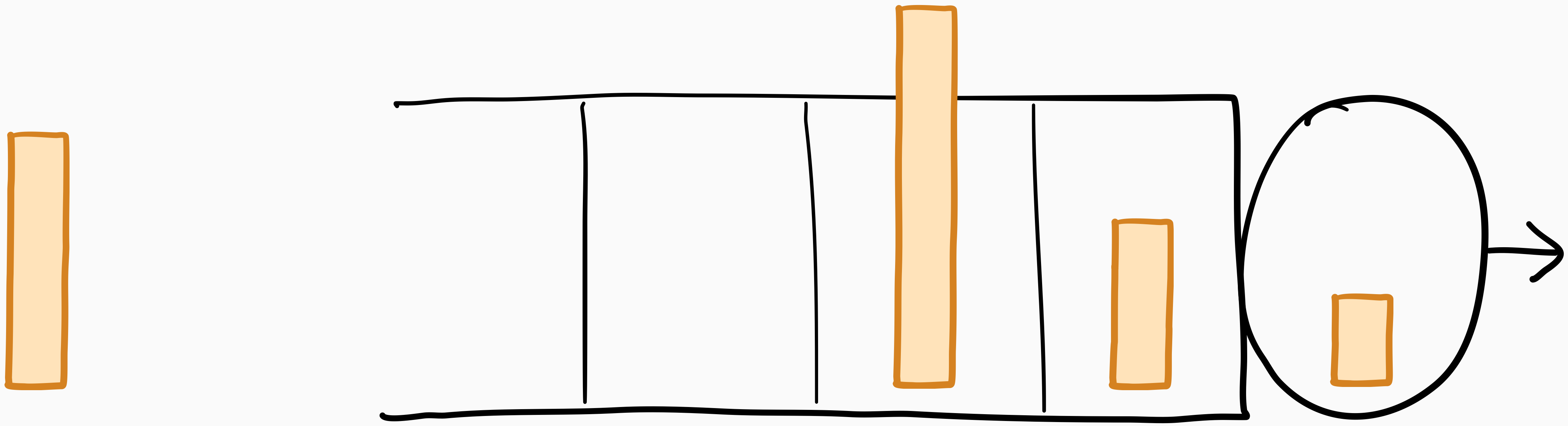
 = **rank**

random system state

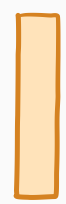
remaining size,
lower is better

Queueing analysis of SRPT-1

[Schrage & Miller, 1966]



“tagged job”

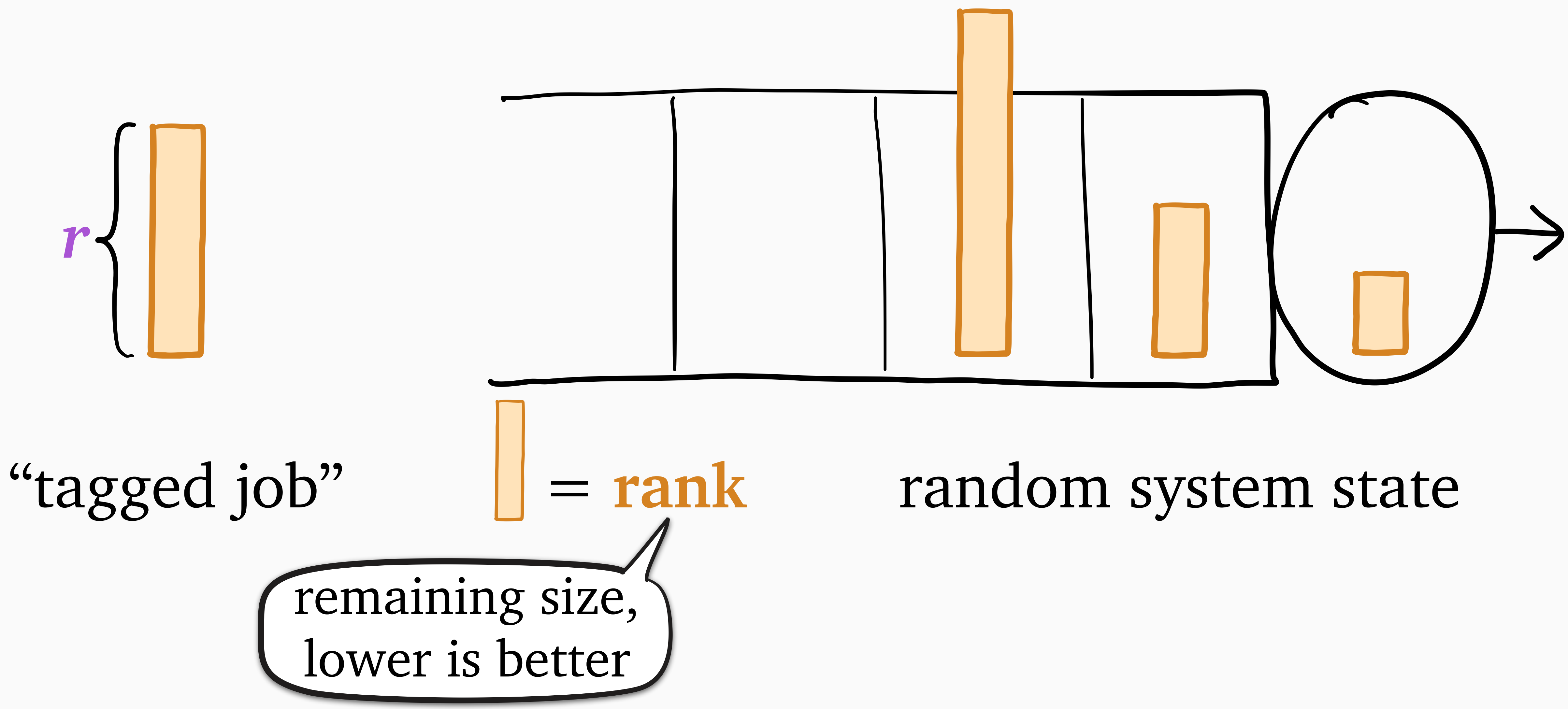
 = **rank**

remaining size,
lower is better

random system state

Queueing analysis of SRPT-1

[Schrage & Miller, 1966]

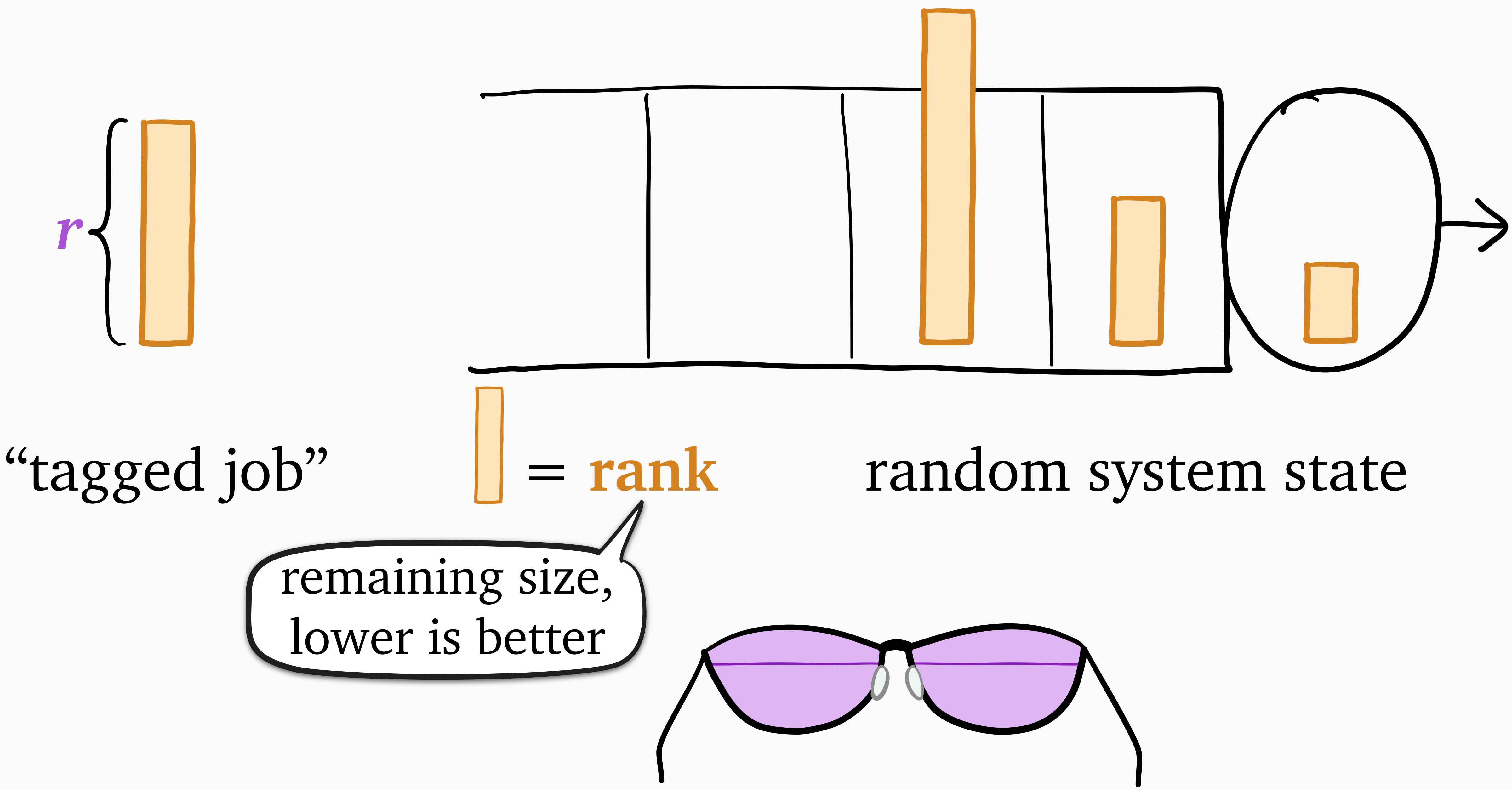


Key quantity:

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

Queueing analysis of SRPT-1

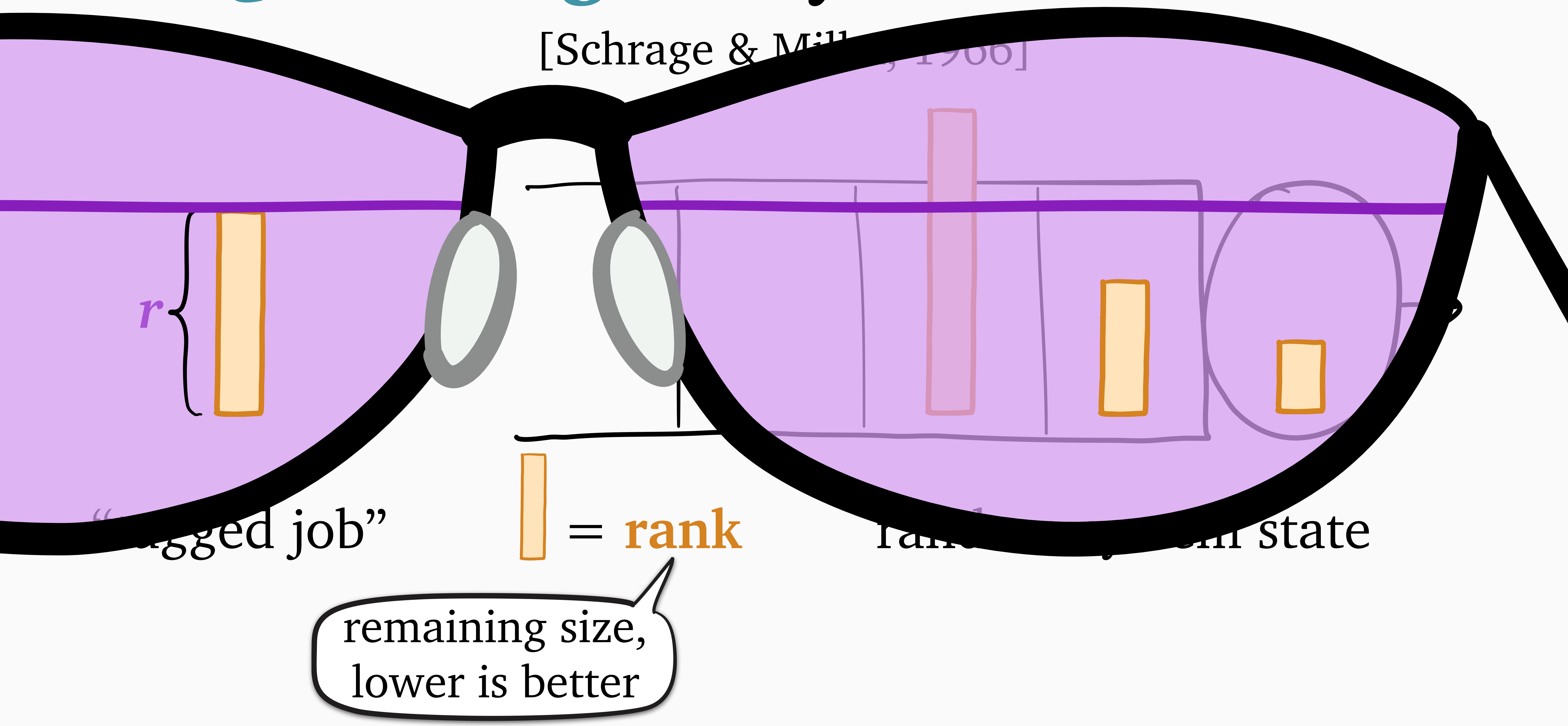
[Schrage & Miller, 1966]



Key quantity:

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

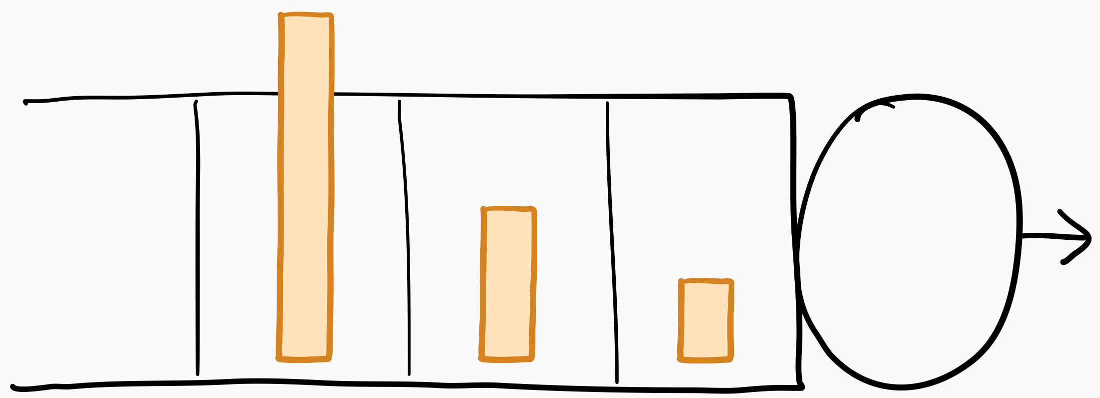
Queueing analysis of SRPT-1



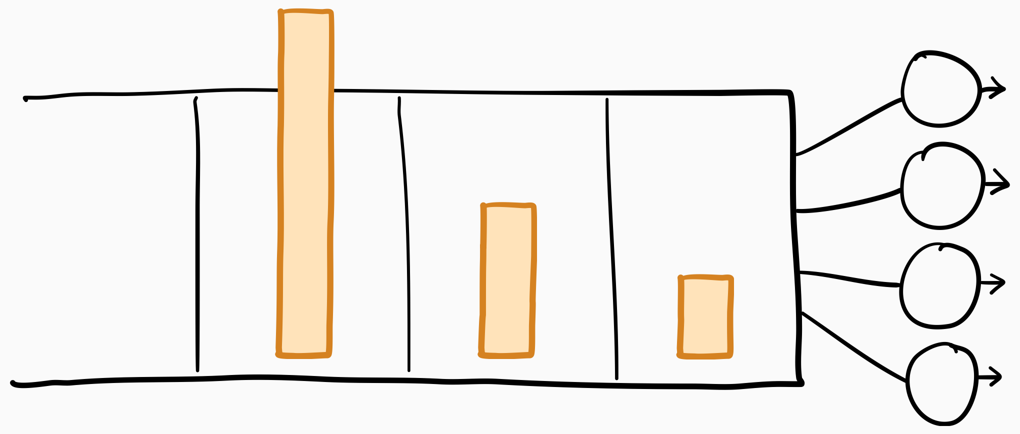
Key quantity:

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

Single-server system

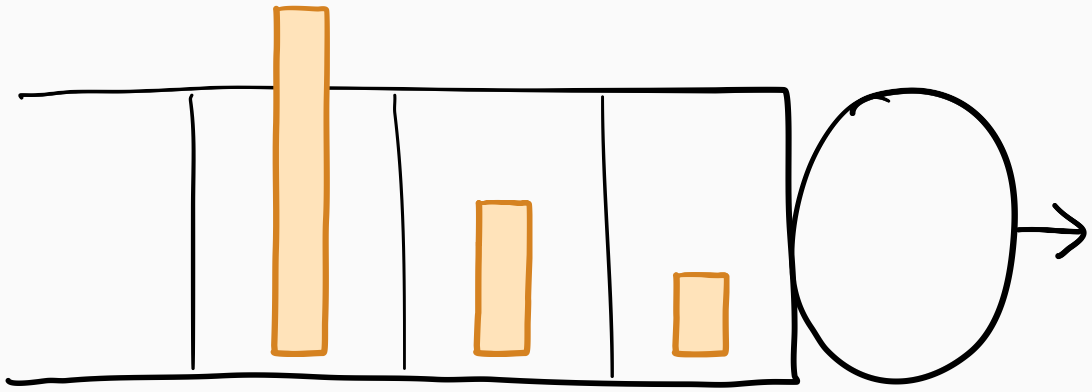


Multiserver system

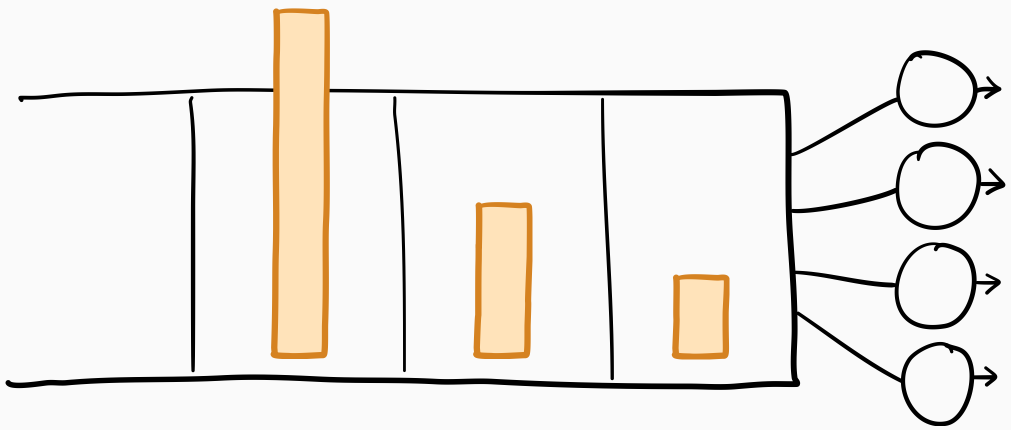


Single-server system

server is “choke point”



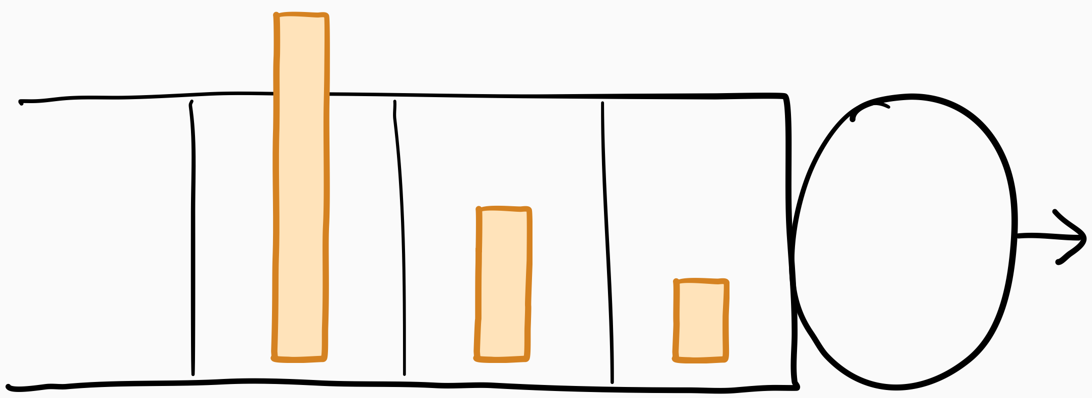
Multiserver system



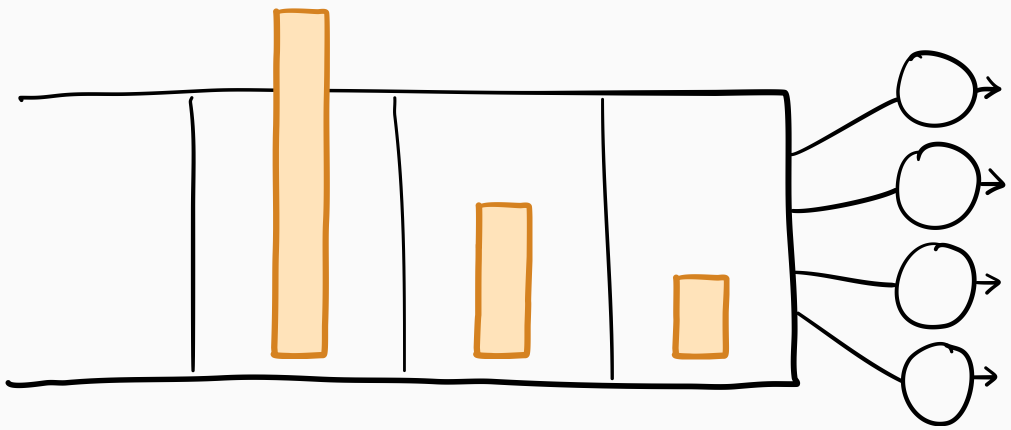
Single-server system

server is “choke point”

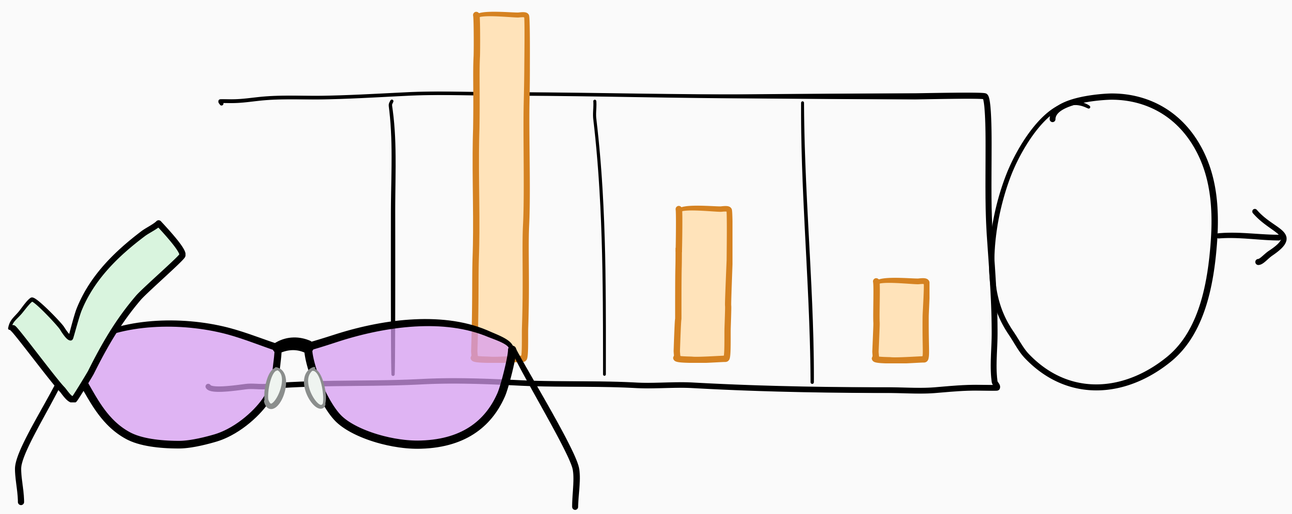
rank ordering absolute



Multiserver system



Single-server system



server is "choke point"

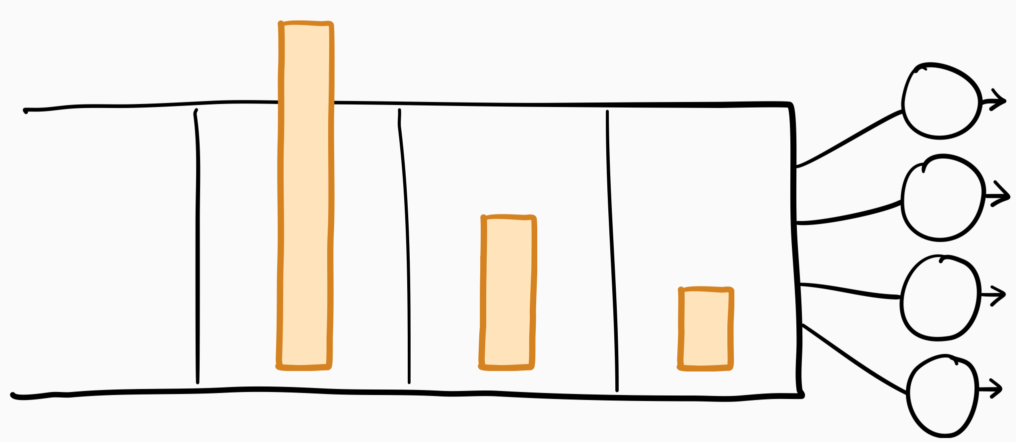
↓

rank ordering absolute

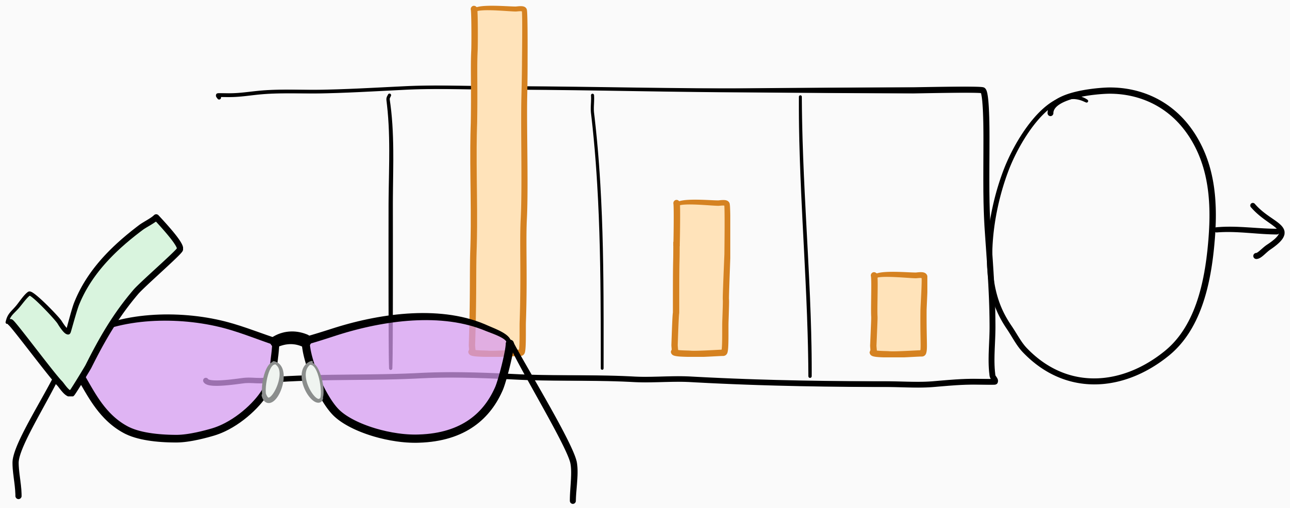
↓

observed r-work determines T

Multiserver system



Single-server system



server is "choke point"

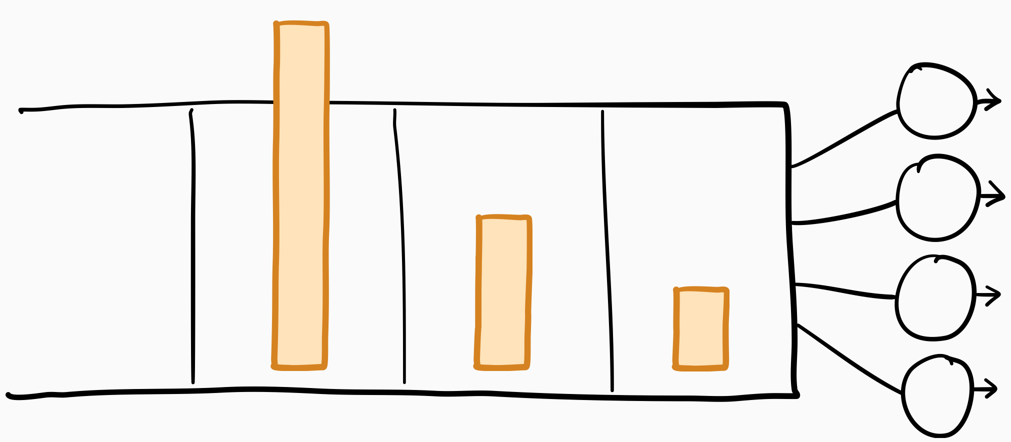
↓

rank ordering absolute

↓

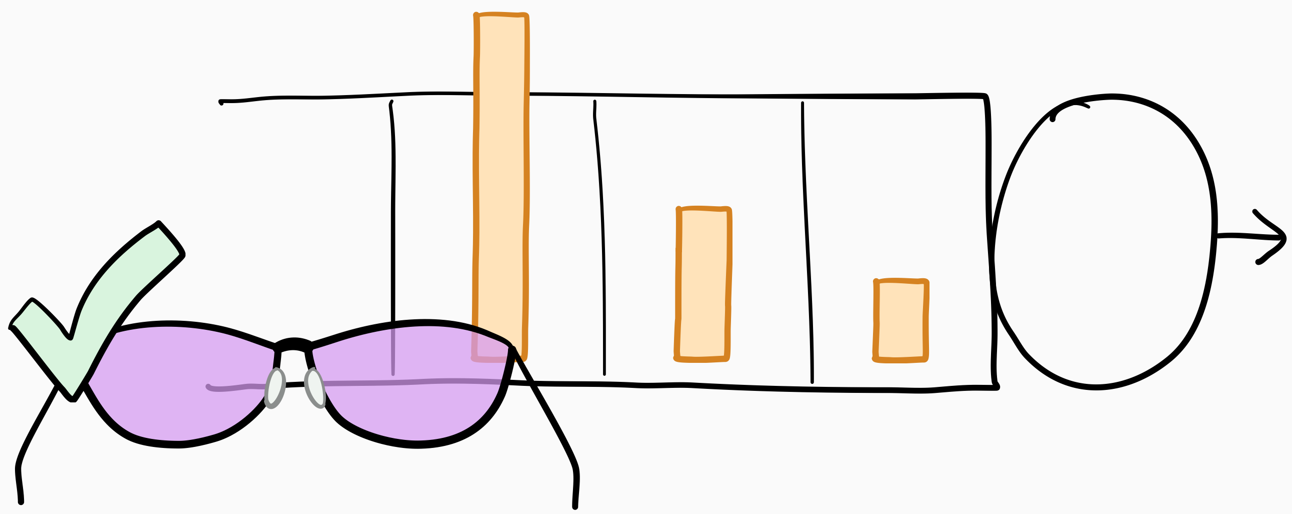
observed ***r***-work determines T

Multiserver system



no single "choke point"

Single-server system



server is “choke point”

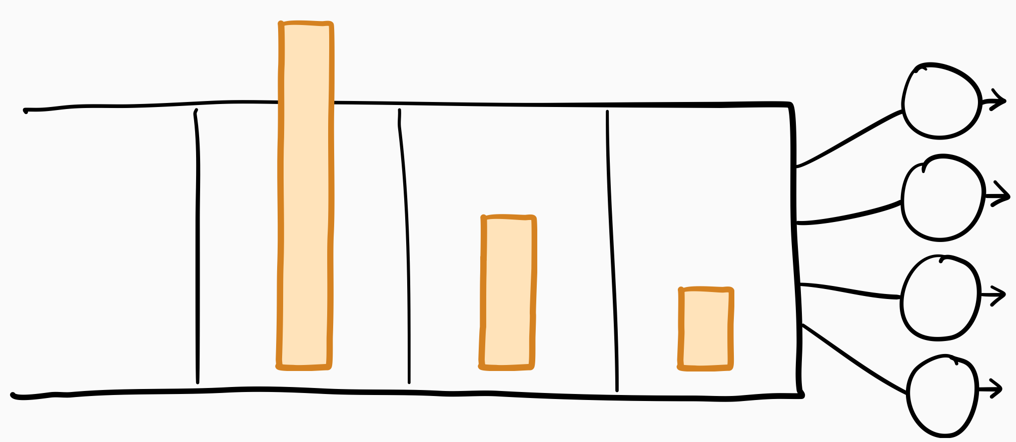
↓

rank ordering absolute

↓

observed ***r***-work determines T

Multiserver system

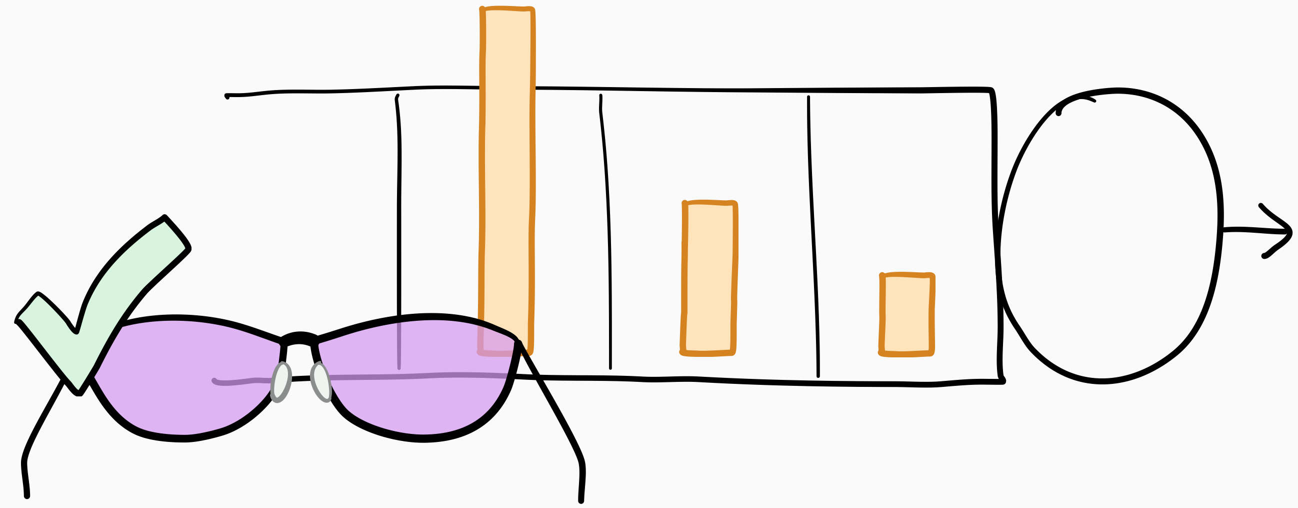


no single “choke point”

↓

rank ordering *not* absolute

Single-server system



server is "choke point"

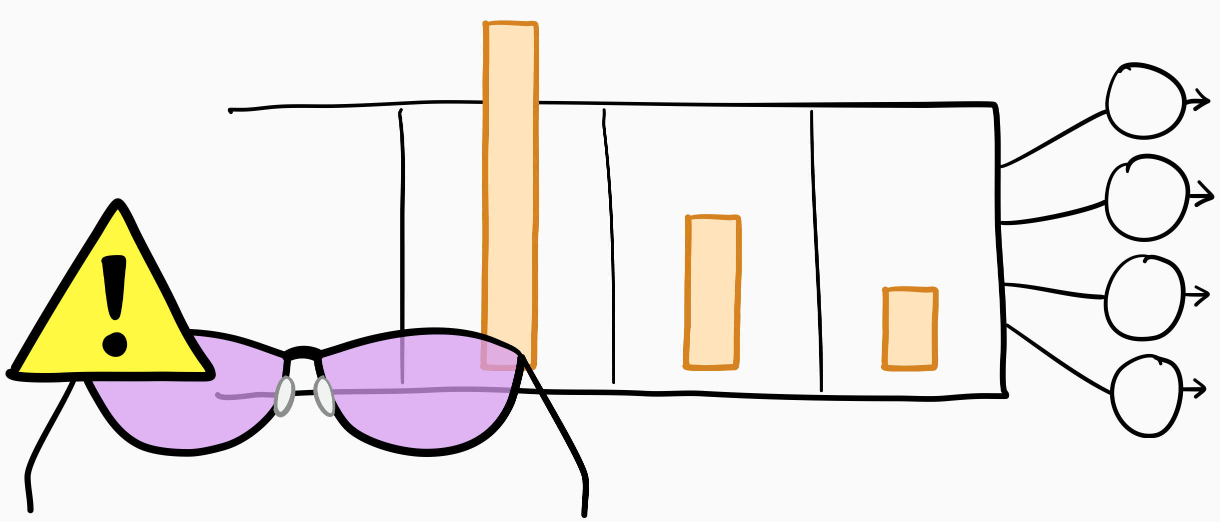
↓

rank ordering absolute

↓

observed *r*-work determines *T*

Multiserver system



no single "choke point"

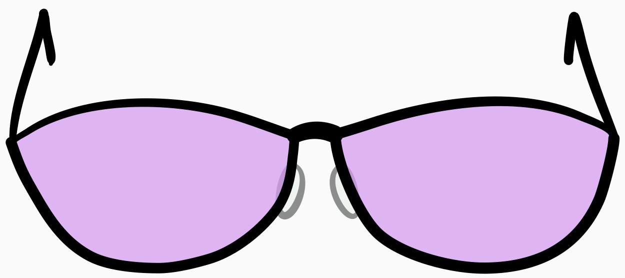
↓

rank ordering *not* absolute

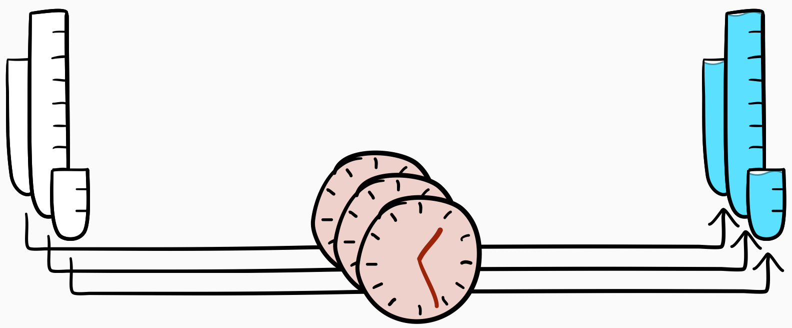
↓

observed *r*-work *not enough!*

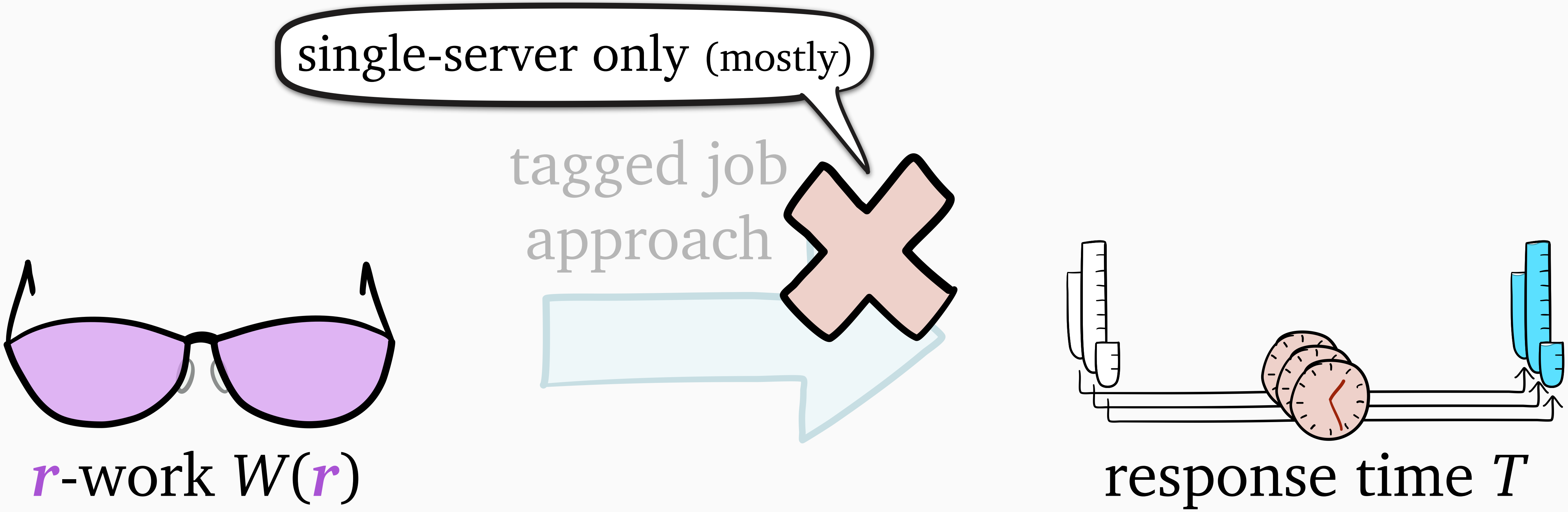
tagged job approach

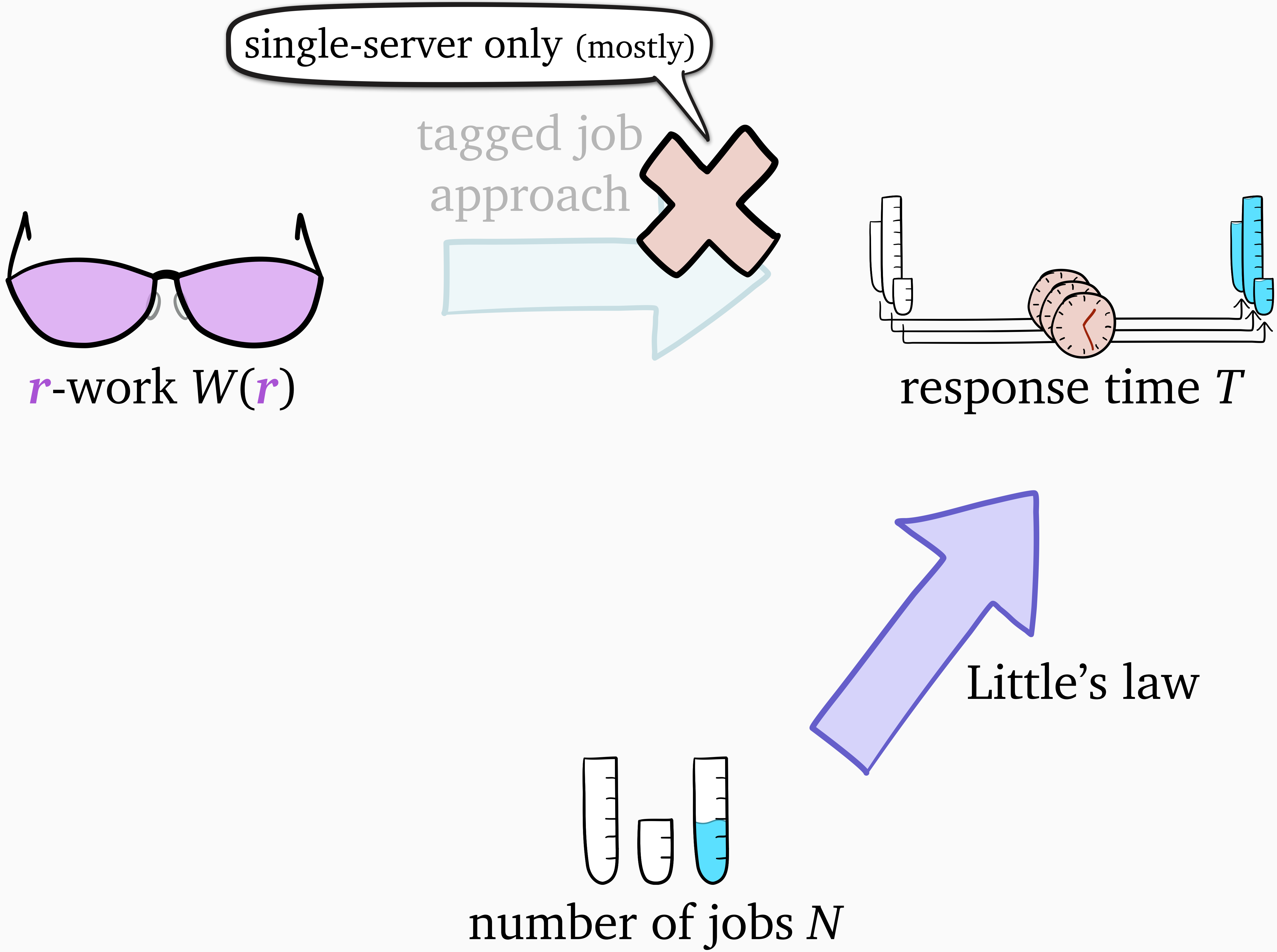


r-work $W(r)$

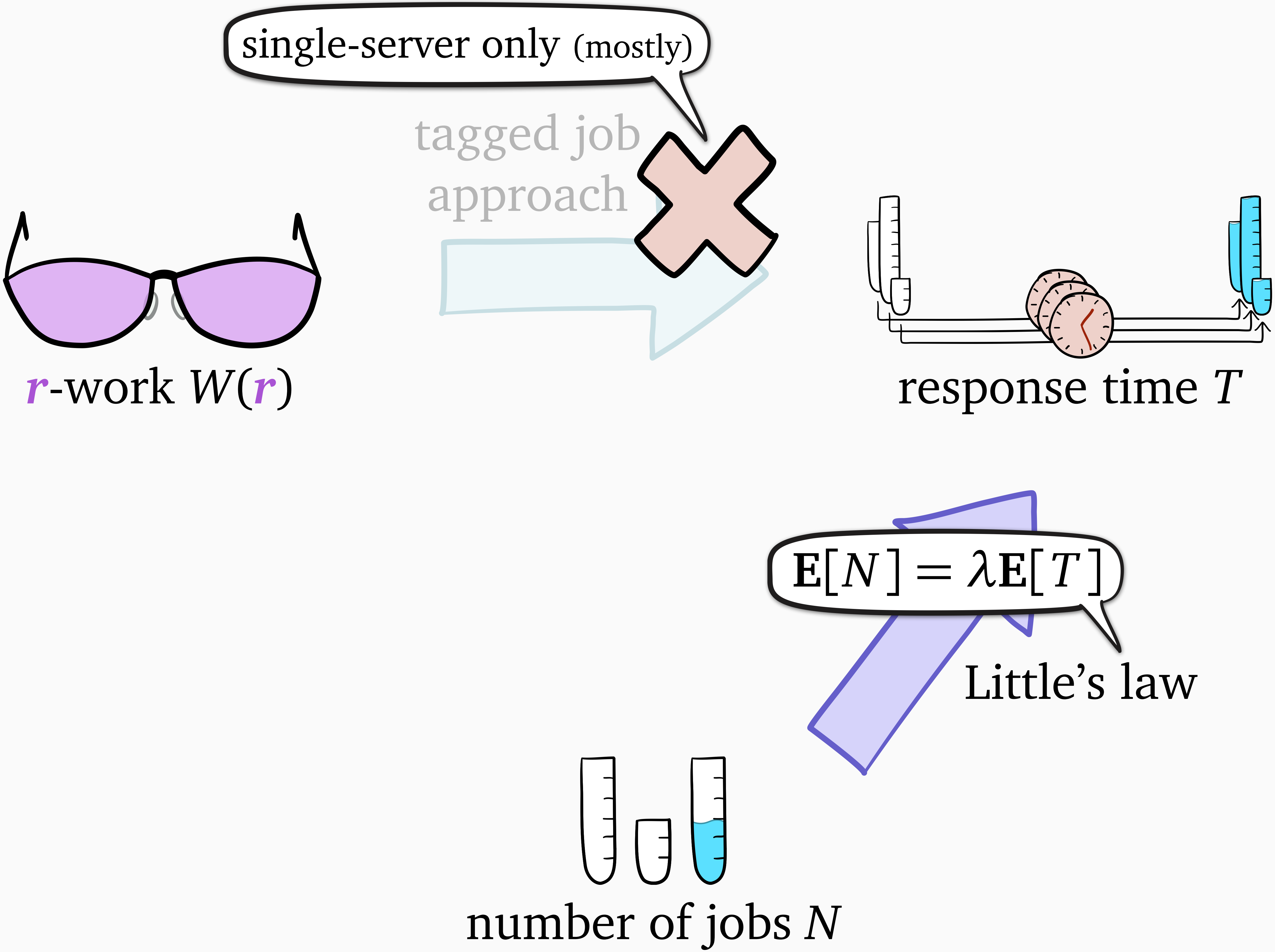


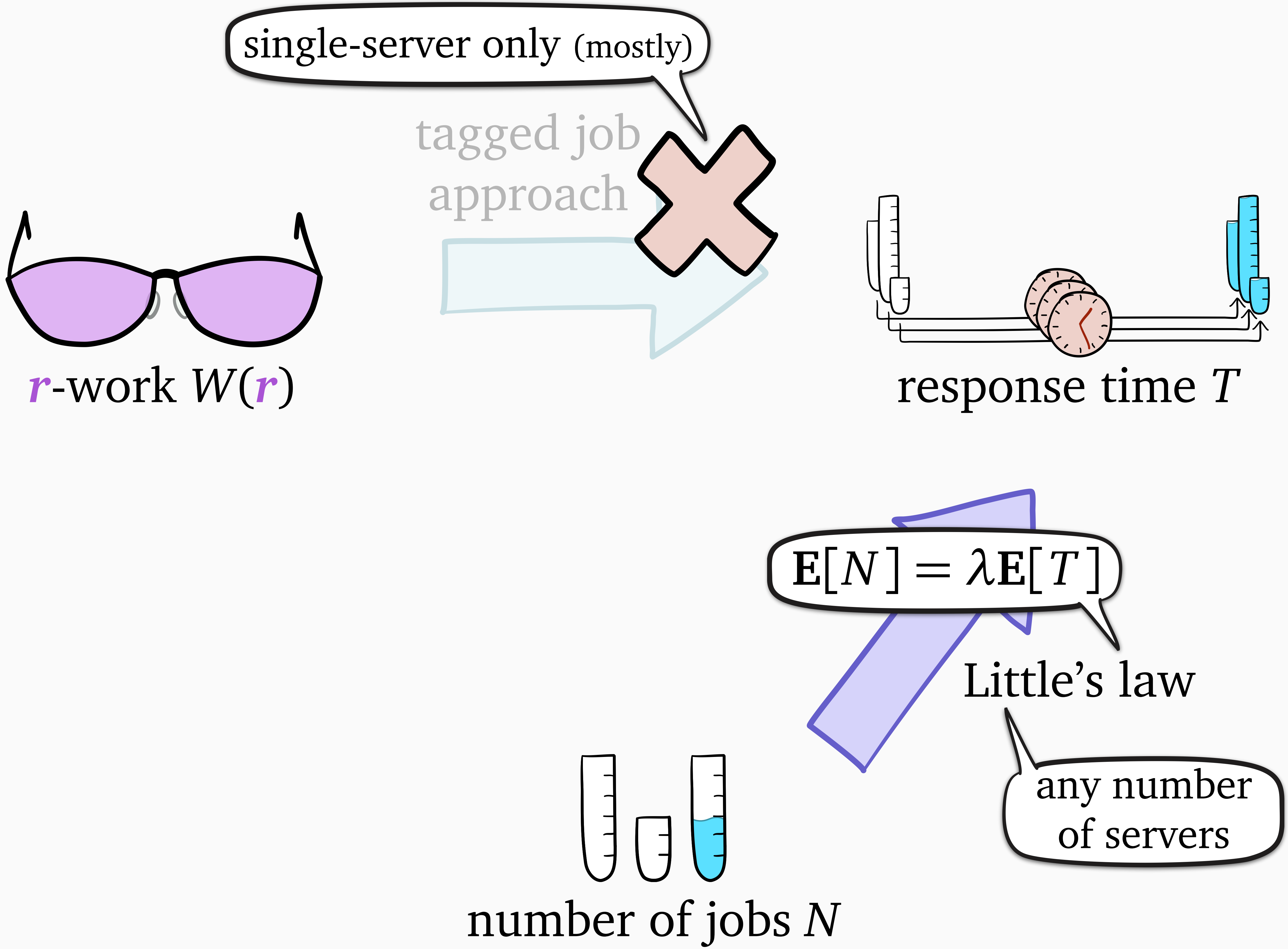
response time T


single-server only (mostly)
tagged job approach
r-work W(r)
response time T

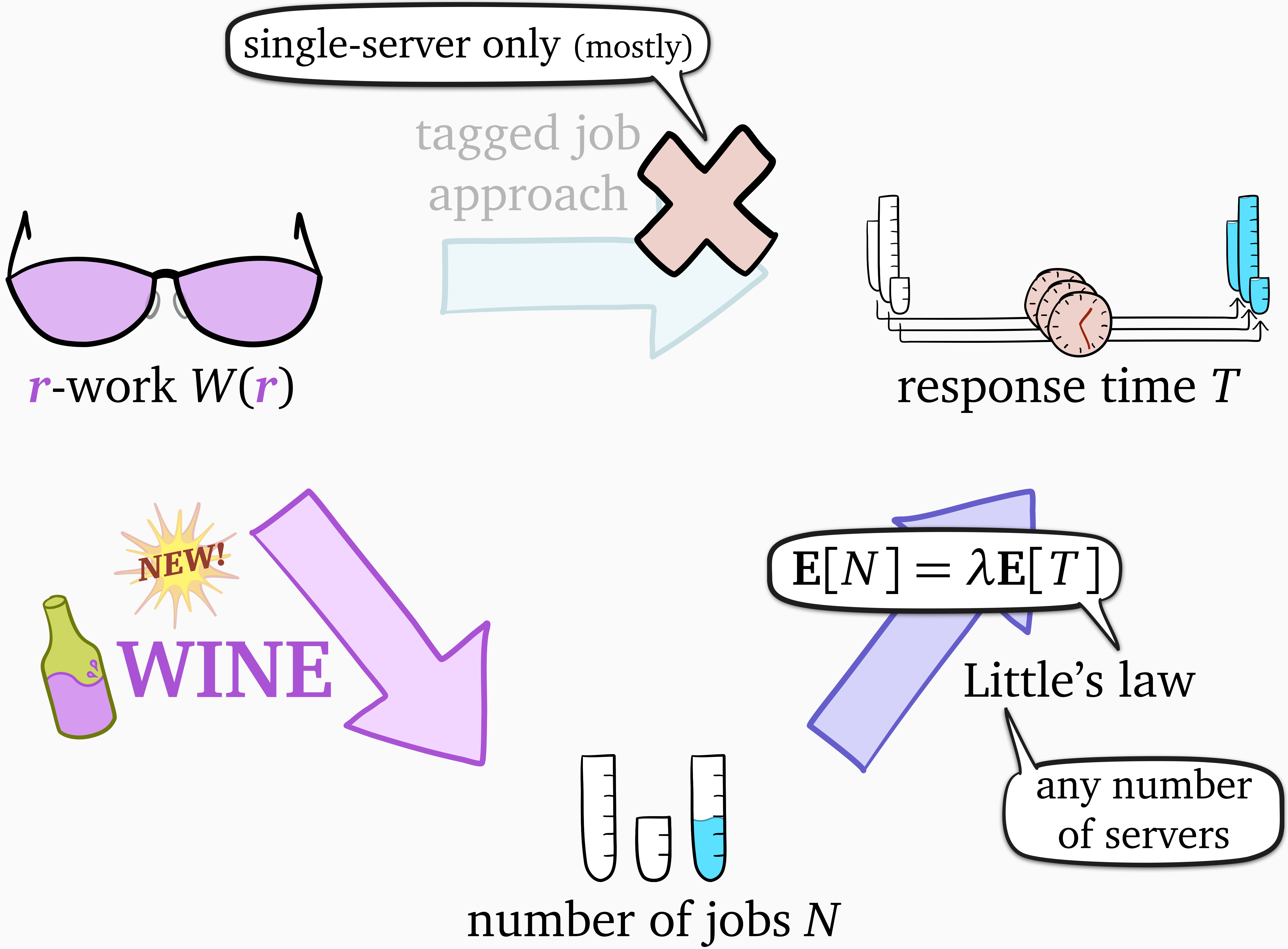


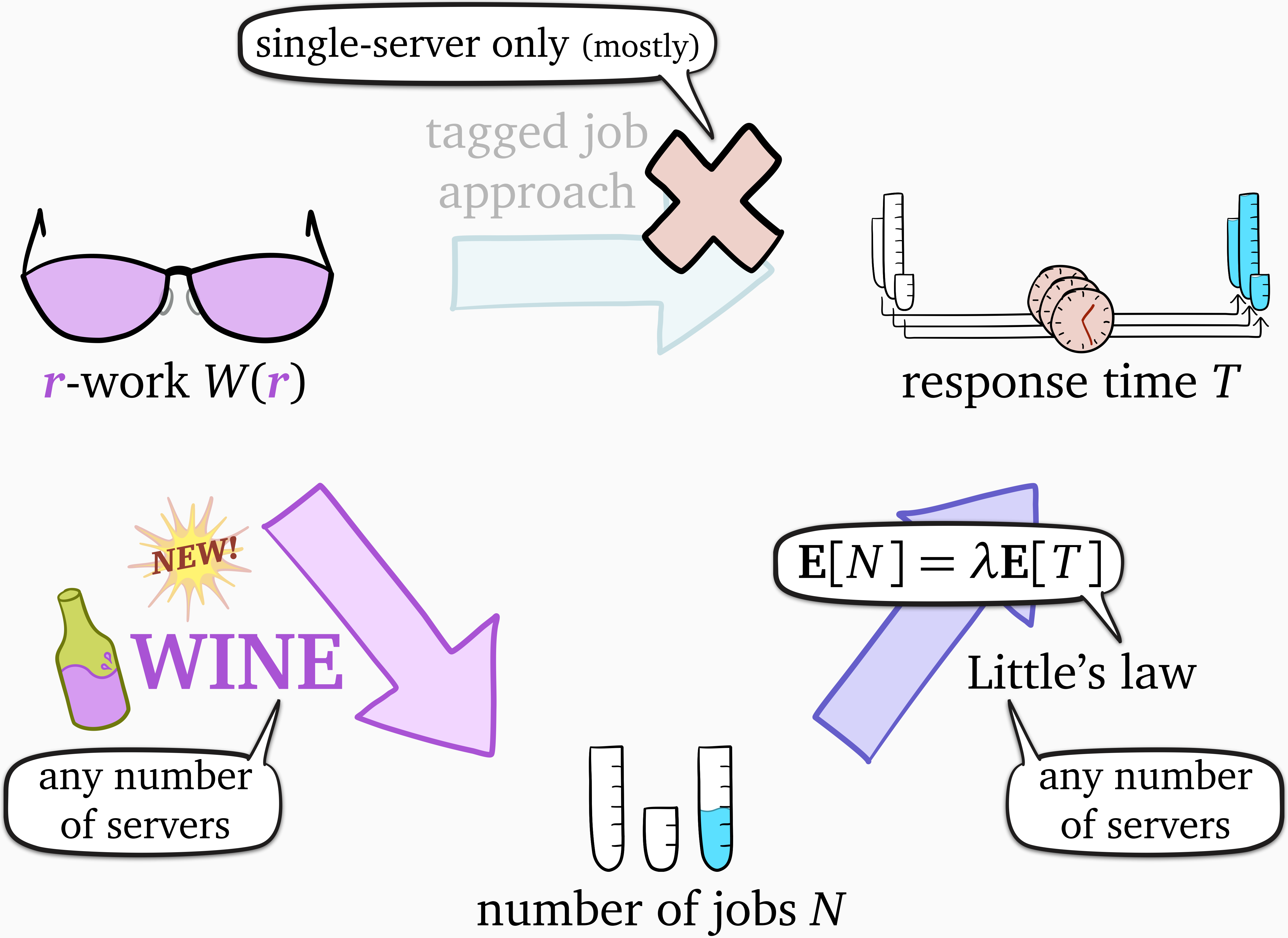
single-server only (mostly)
tagged job approach
r-work W(r)
response time T
Little's law
number of jobs N


single-server only (mostly)
tagged job approach
r-work $W(r)$
response time T
$\mathbf{E}[N] = \lambda \mathbf{E}[T]$
Little's law
number of jobs N



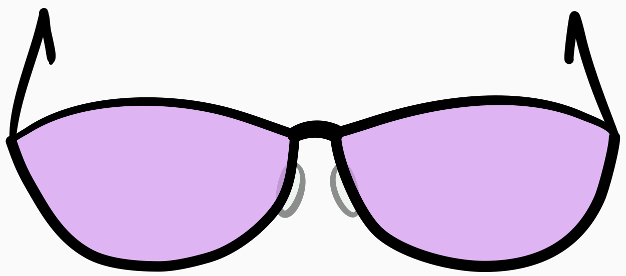
single-server only (mostly)
tagged job approach
r-work $W(r)$
response time T
$\mathbf{E}[N] = \lambda\mathbf{E}[T]$
Little's law
any number of servers
number of jobs N


single-server only (mostly)
tagged job approach
r-work W(r)
response time T
NEW!
WINE
E[N] = λE[T]
Little's law
any number of servers
number of jobs N

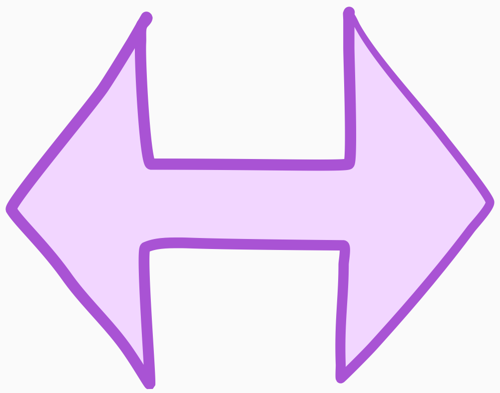

single-server only (mostly)
tagged job approach
r-work W(r)
response time T
NEW!
WINE
any number of servers
E[N] = λE[T]
Little's law
any number of servers
number of jobs N

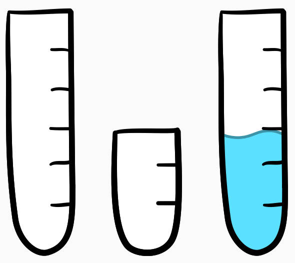
WINE

Work Integral Number Equality



r-work $W(r)$

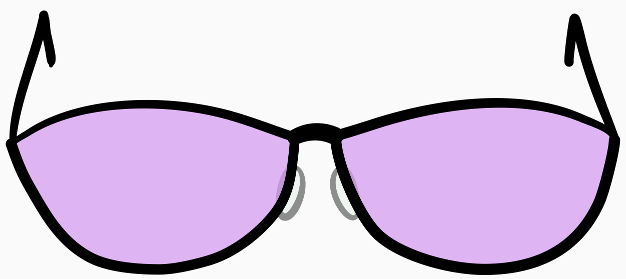




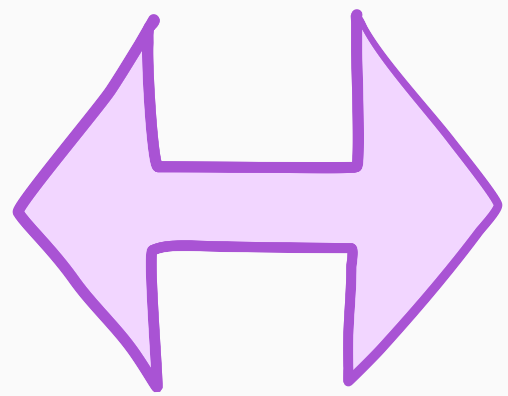
number of jobs N

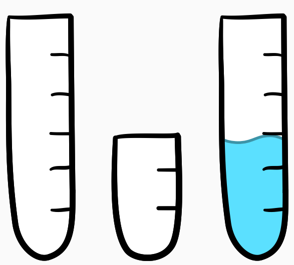
WINE

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



r-work $W(r)$



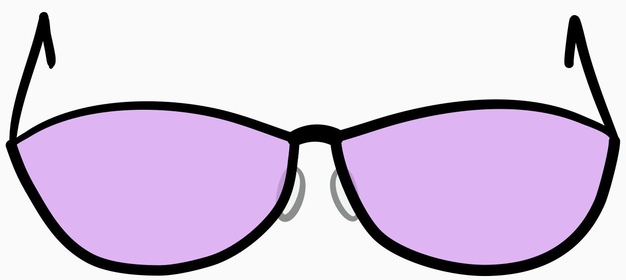


number of jobs N

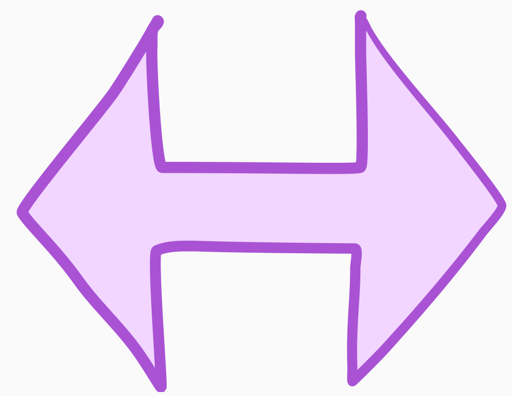
What is *r*-work?

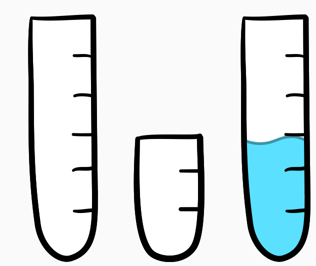
WINE

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



r-work $W(r)$





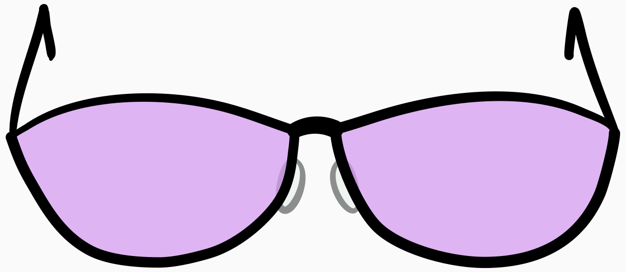
number of jobs N

What is r-work?

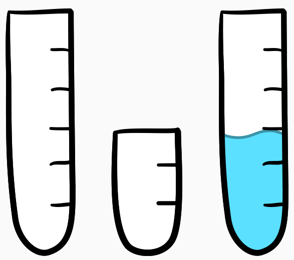
How do we get number of jobs from r-work?

WINE

Work Integral Number Equality



r-work $W(r)$



number of jobs N

What is r-work?

How do we get number of jobs from r-work?

How do we analyze r-work?

Defining r-work

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

Defining r-work

for SRPT

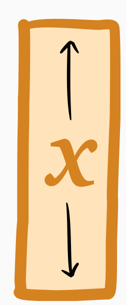
$W(r)$ = work relevant to **rank** r

Defining r-work

for SRPT

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

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

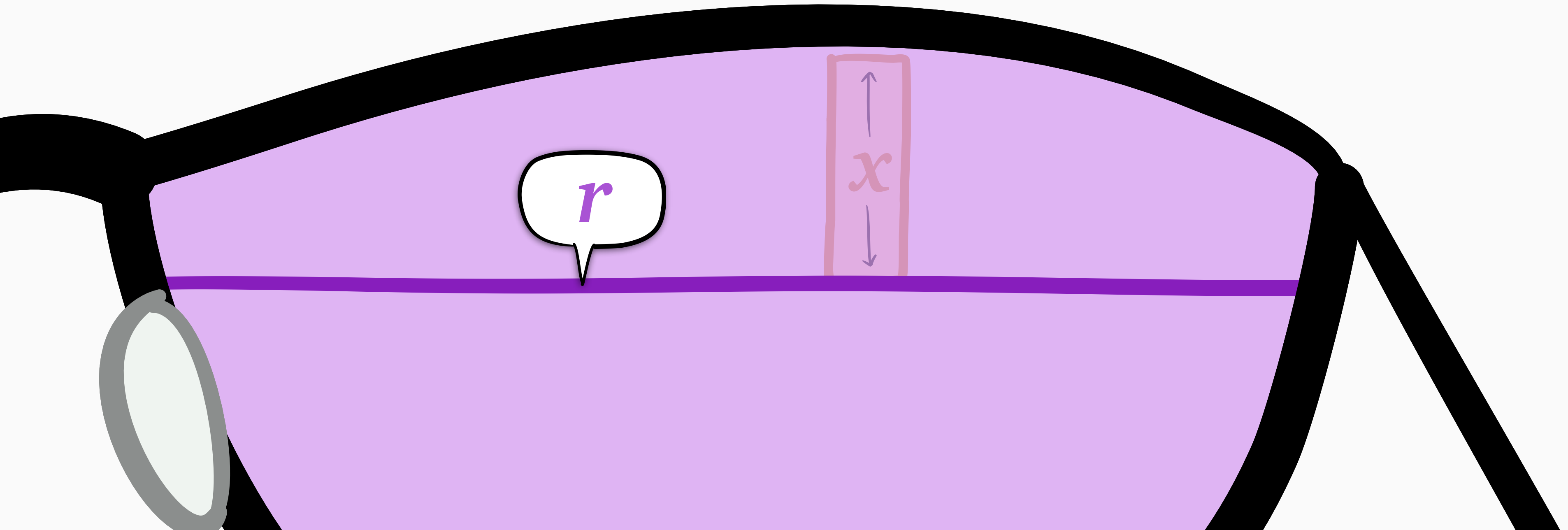


Defining r-work

for SRPT

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

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

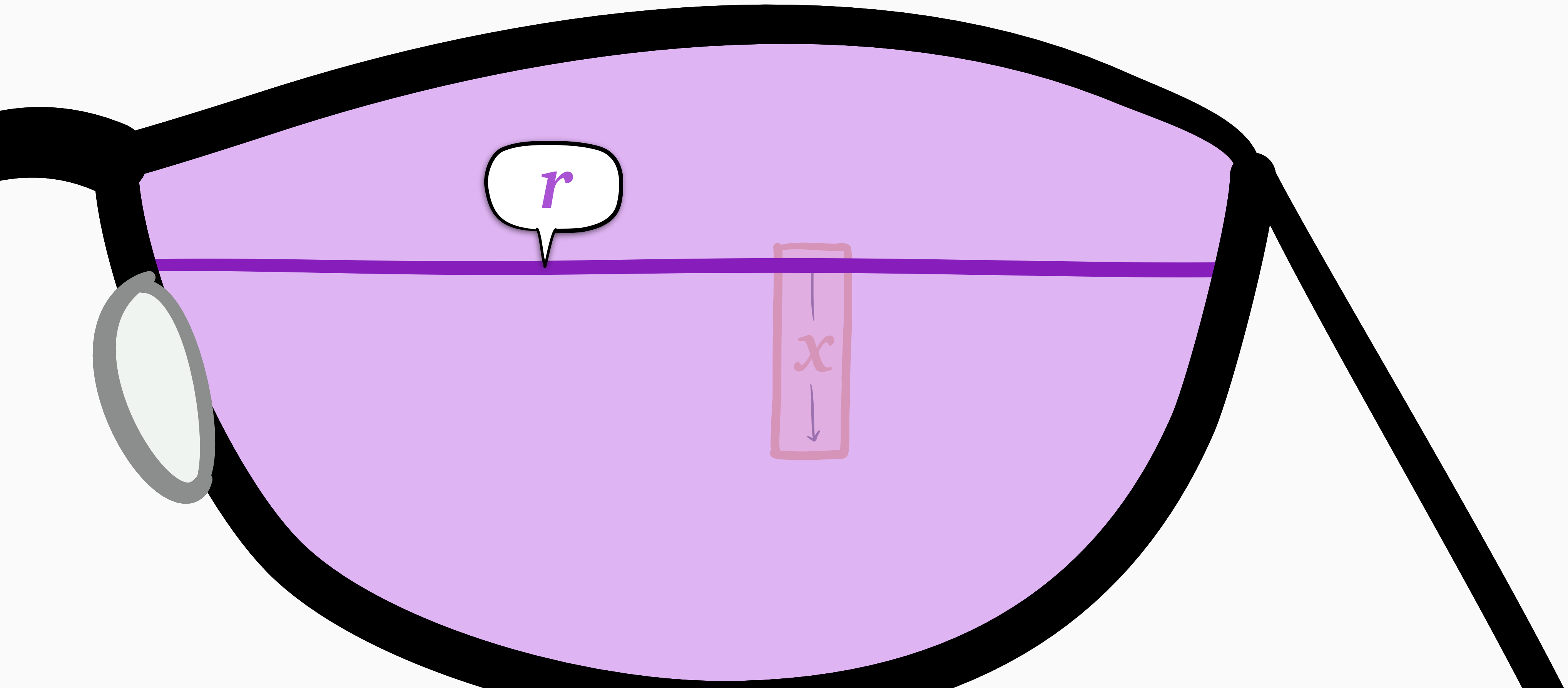


Defining r-work

for SRPT

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

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

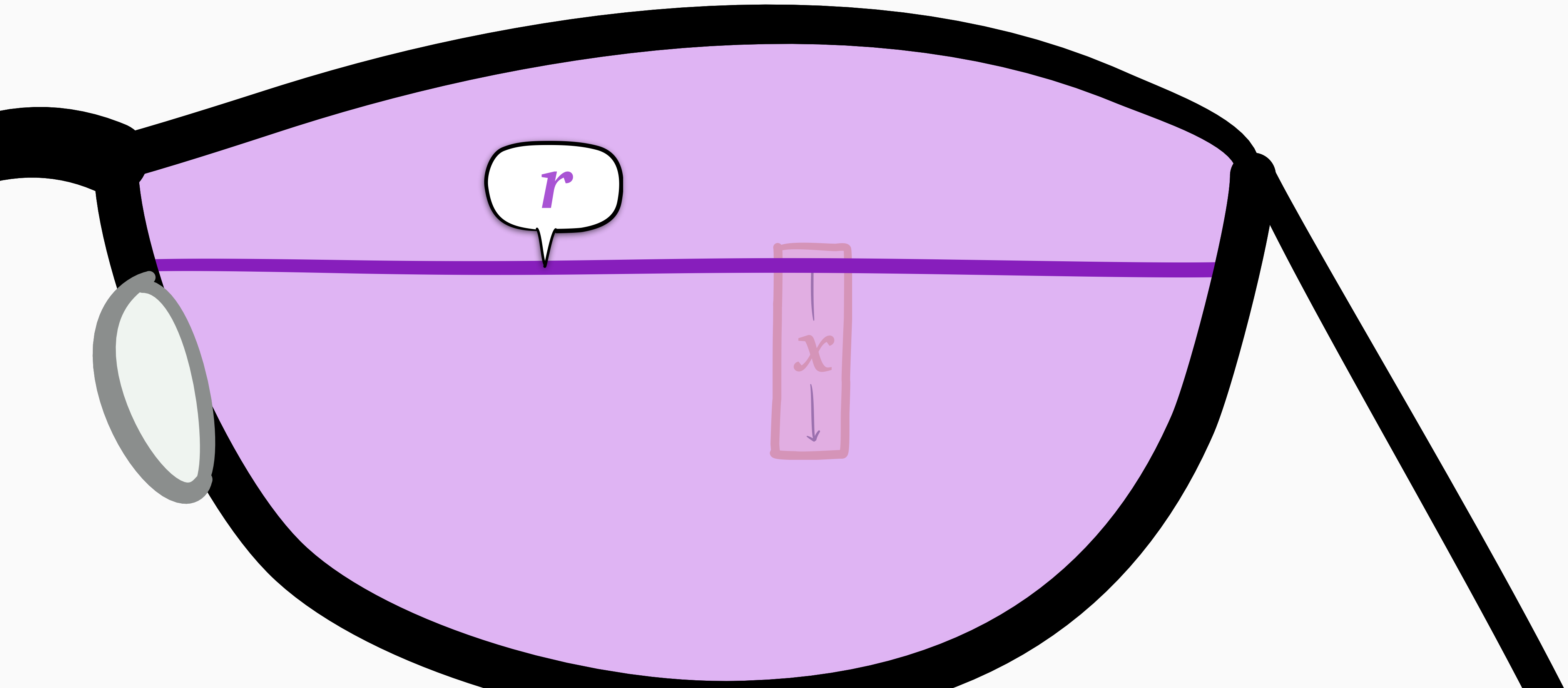


Defining r-work

for SRPT

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

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

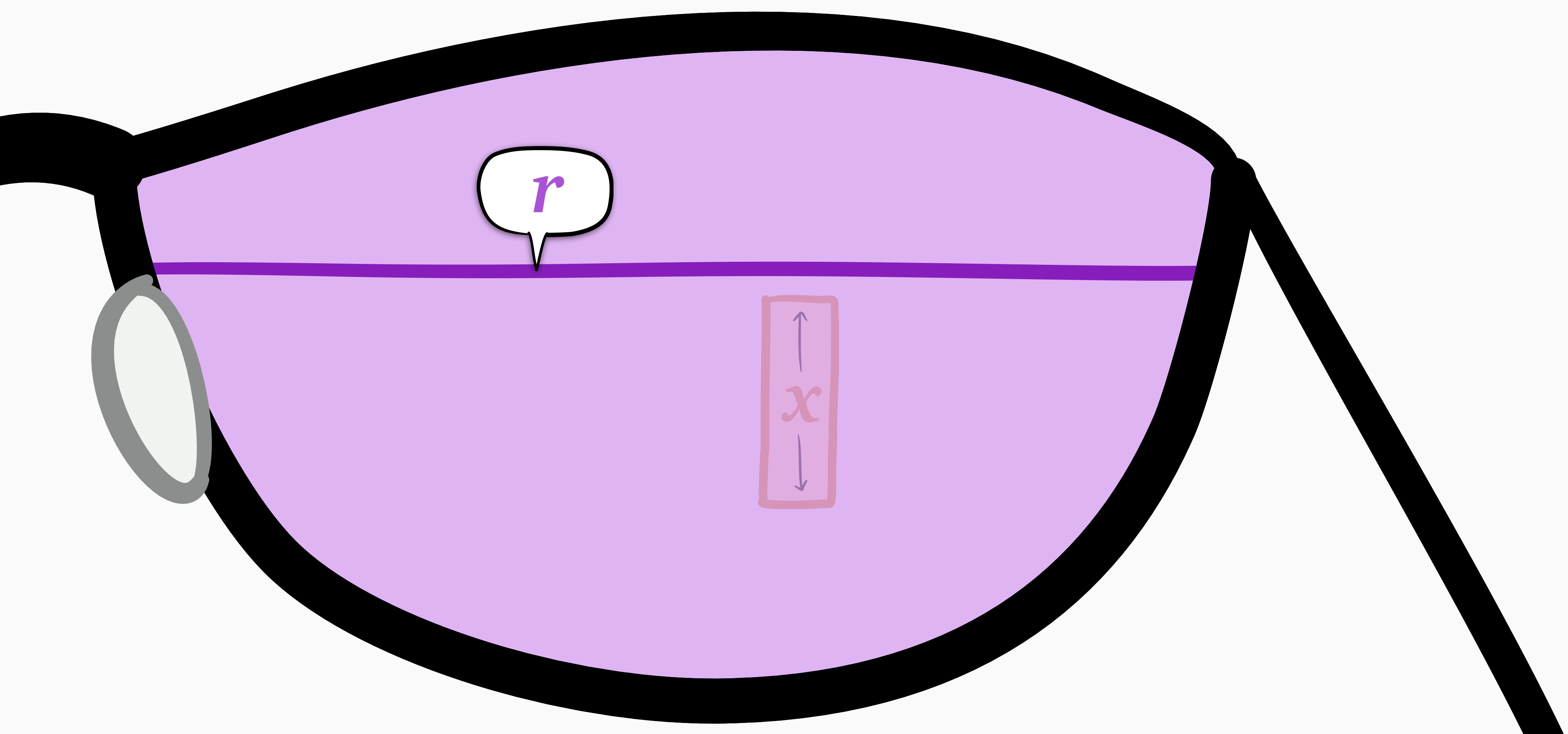


Defining r-work

for SRPT

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

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

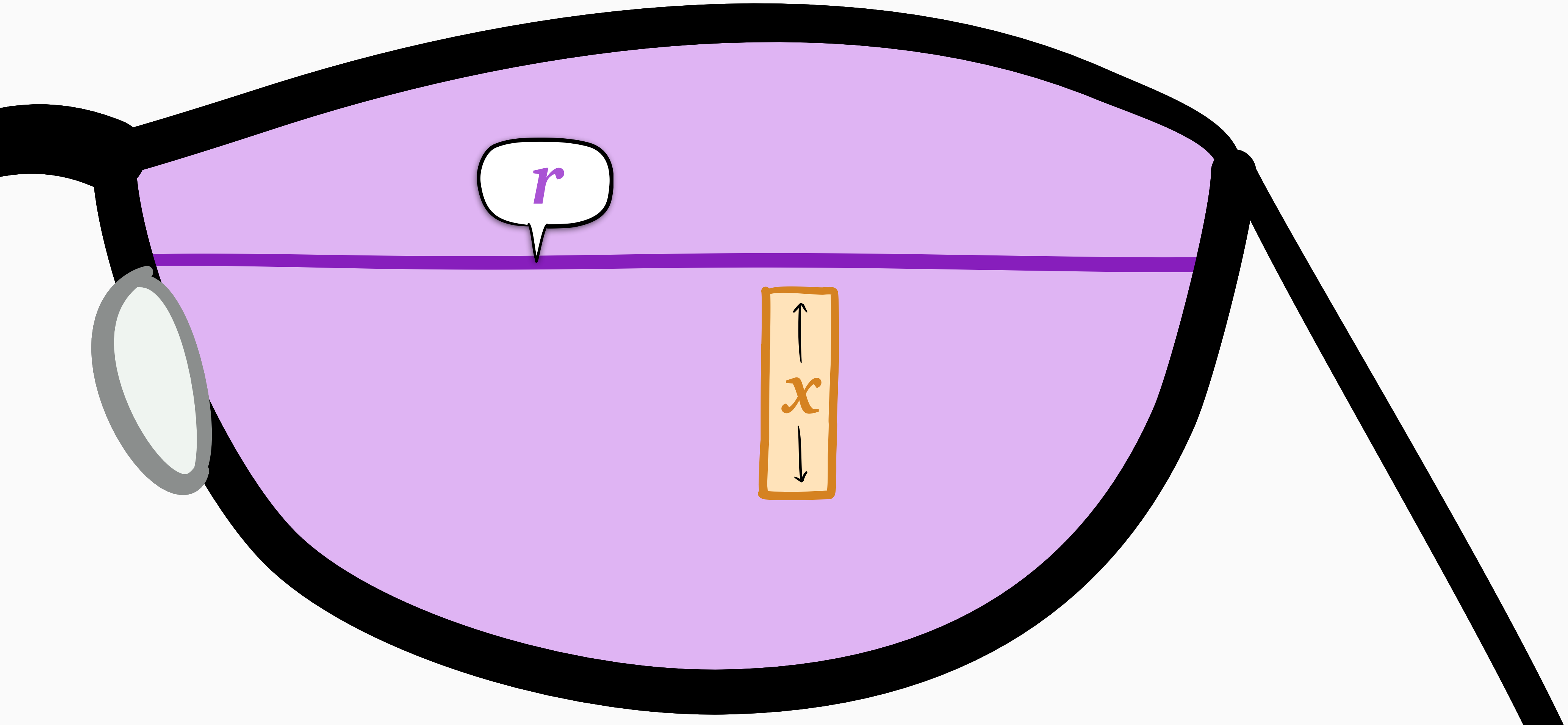


Defining r-work

for SRPT

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

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

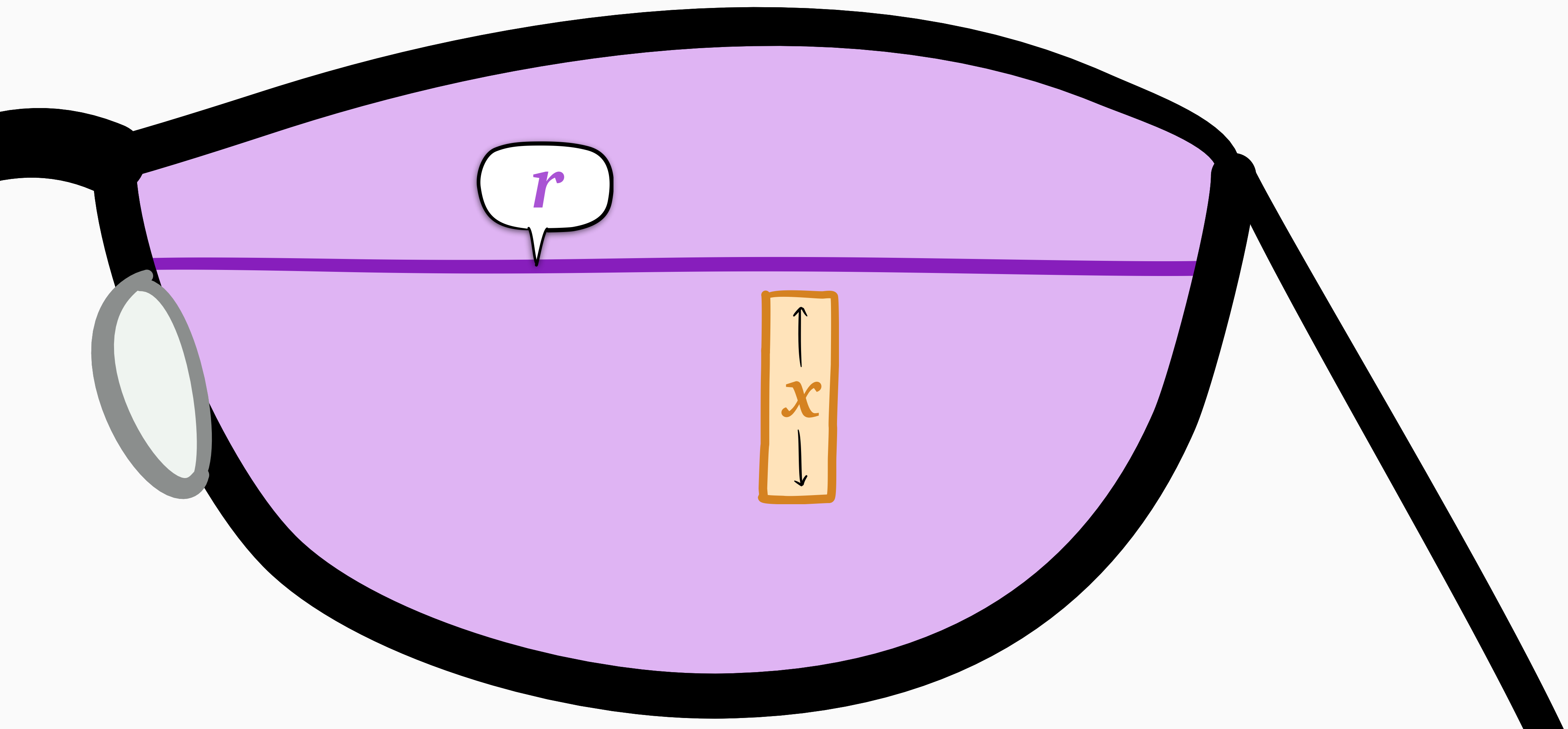


Defining r-work

for SRPT

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

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

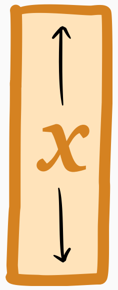


Defining r-work

for SRPT

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

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

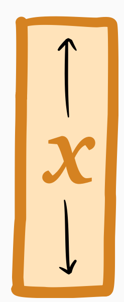


Defining r-work

for SRPT

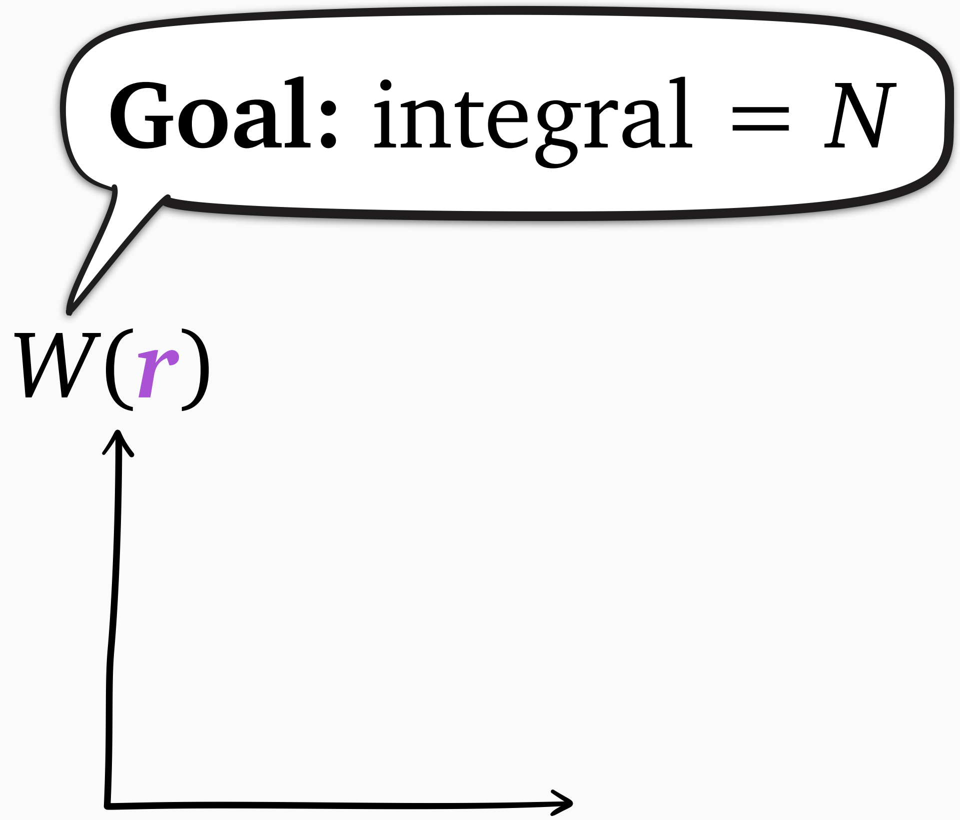
$$W(r) = \text{work relevant to } \textbf{rank } r$$
$$= \text{total } r\text{-work of all jobs}$$

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

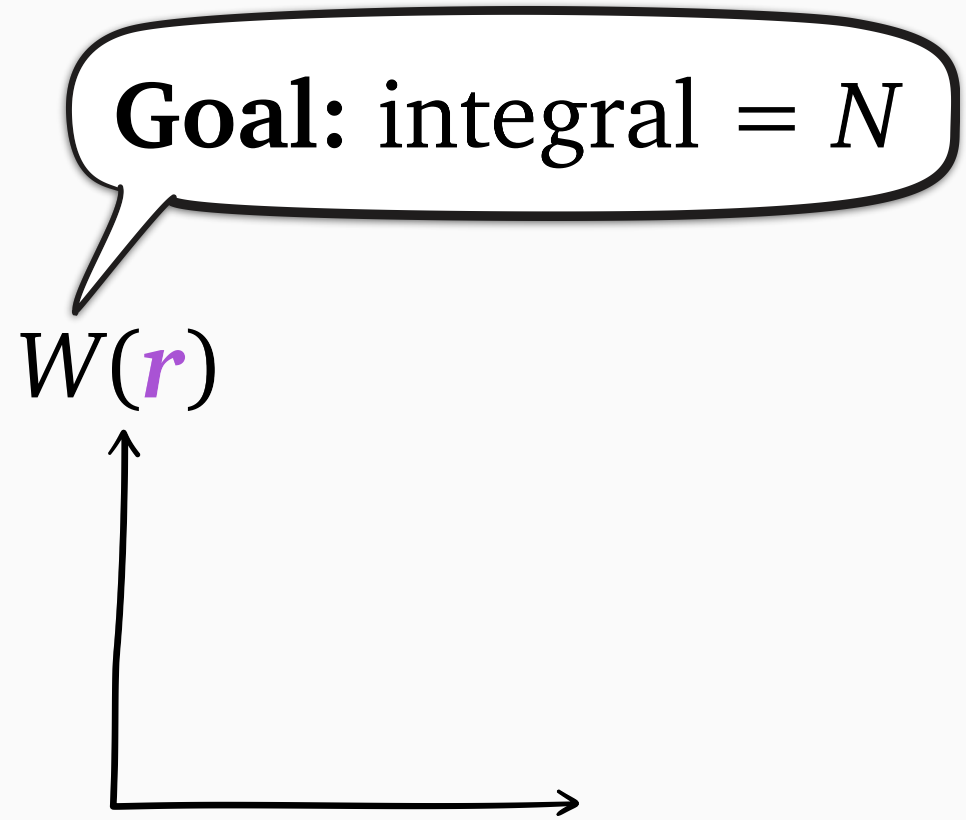


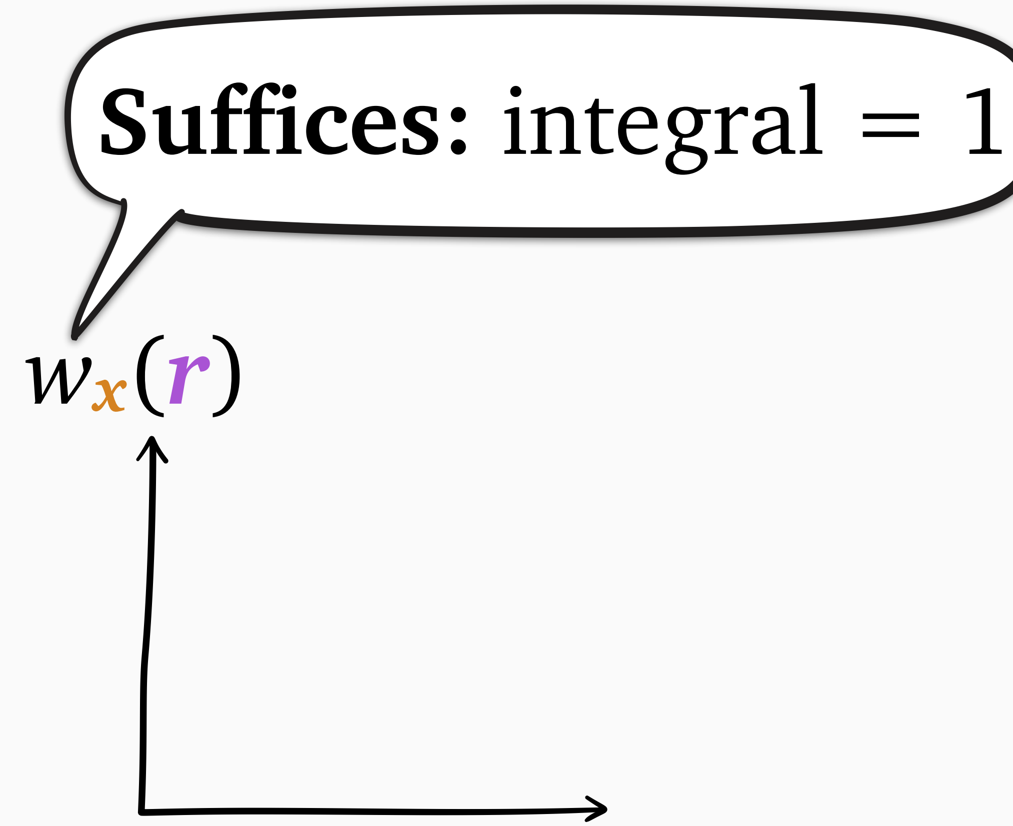
From r-work to number of jobs N

From r-work to number of jobs N



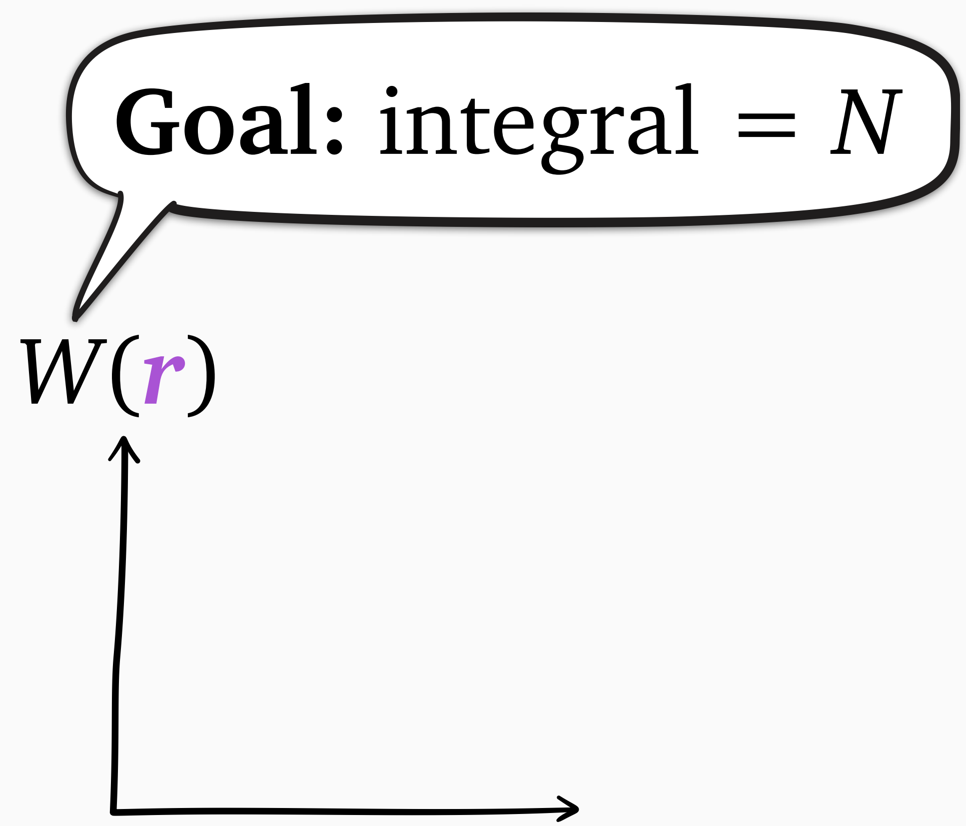
From r-work to number of jobs N

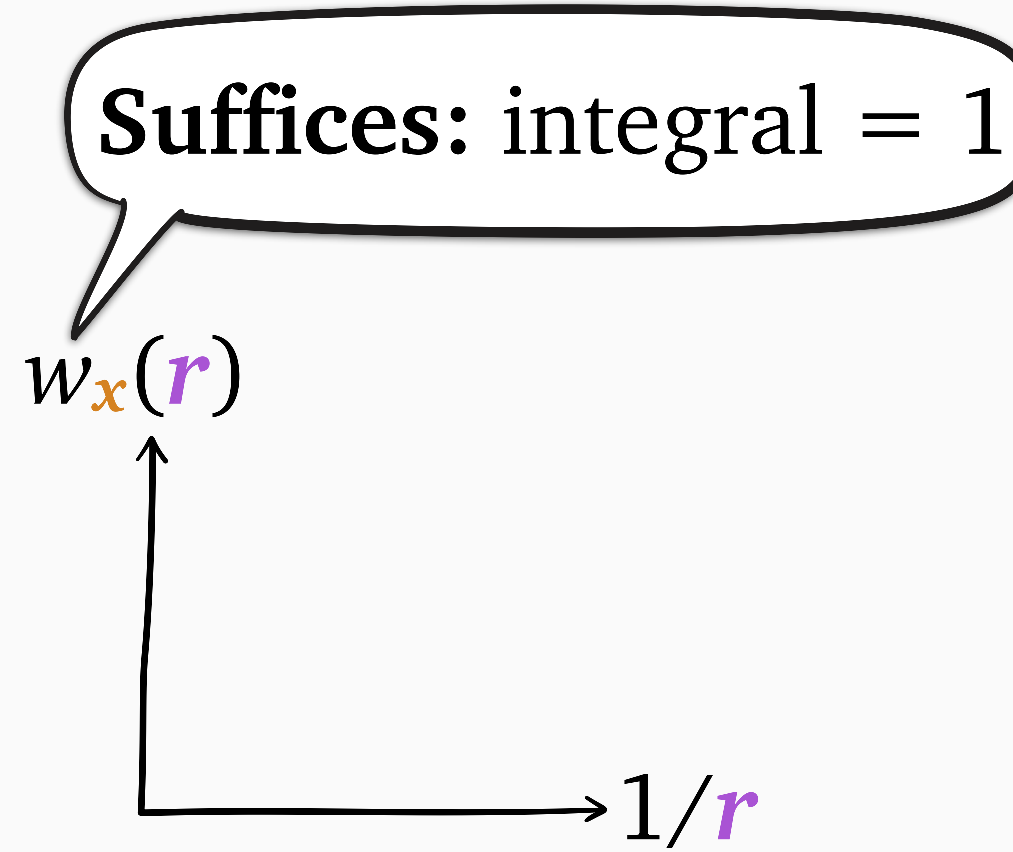




$$w_x(r) = r\text{-work of } \textit{job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$

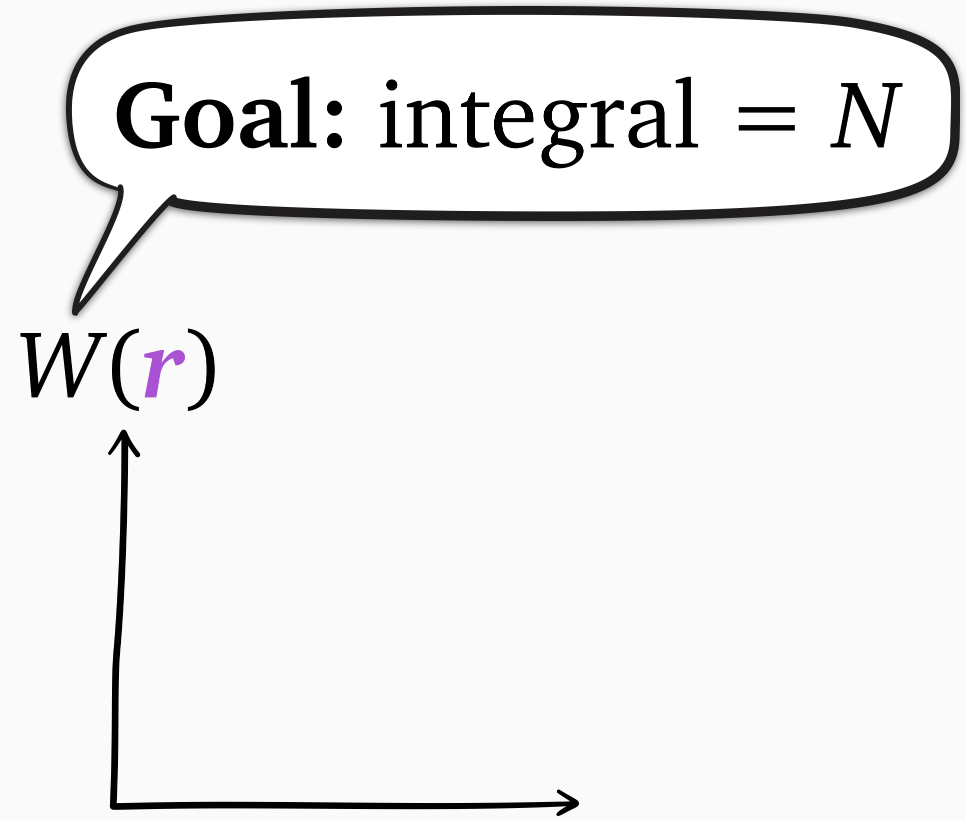
From r-work to number of jobs N

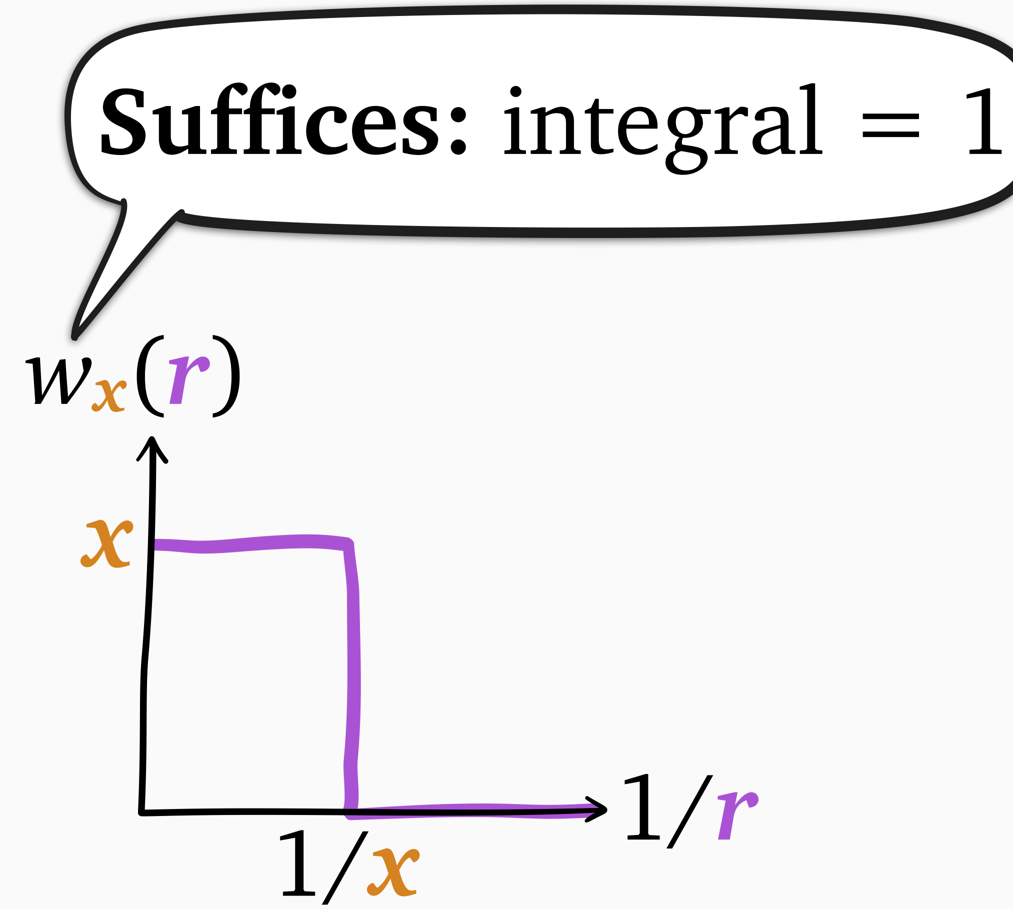




$$w_x(r) = r\text{-work of } \textit{job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$

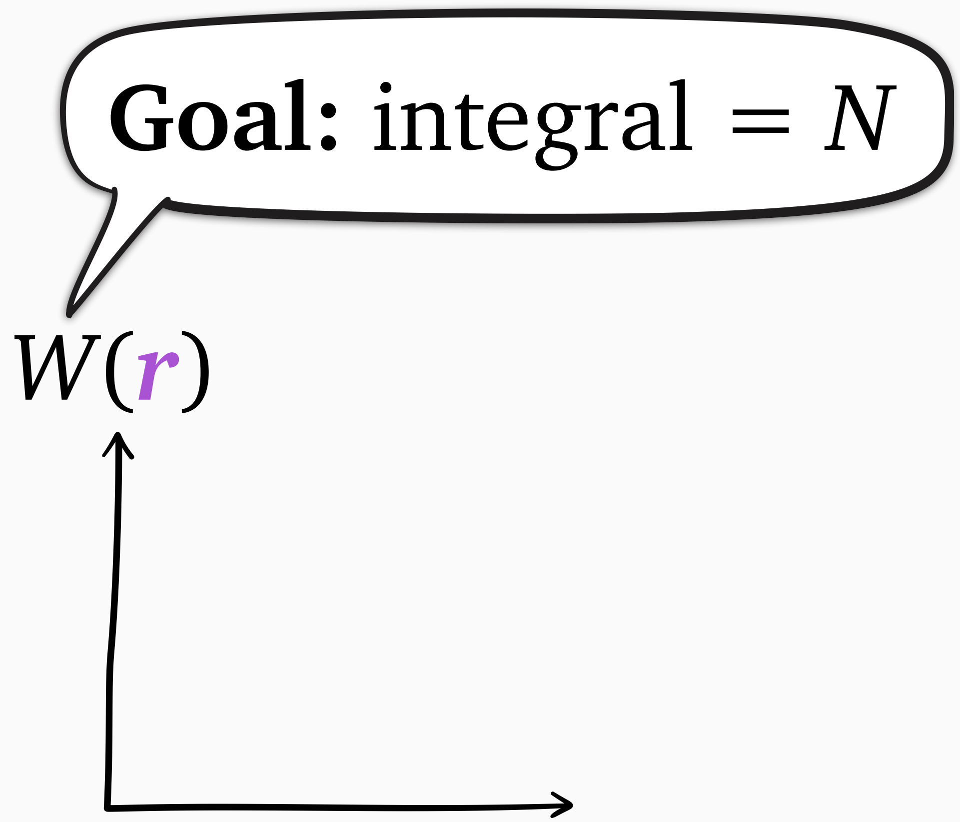
From r-work to number of jobs N

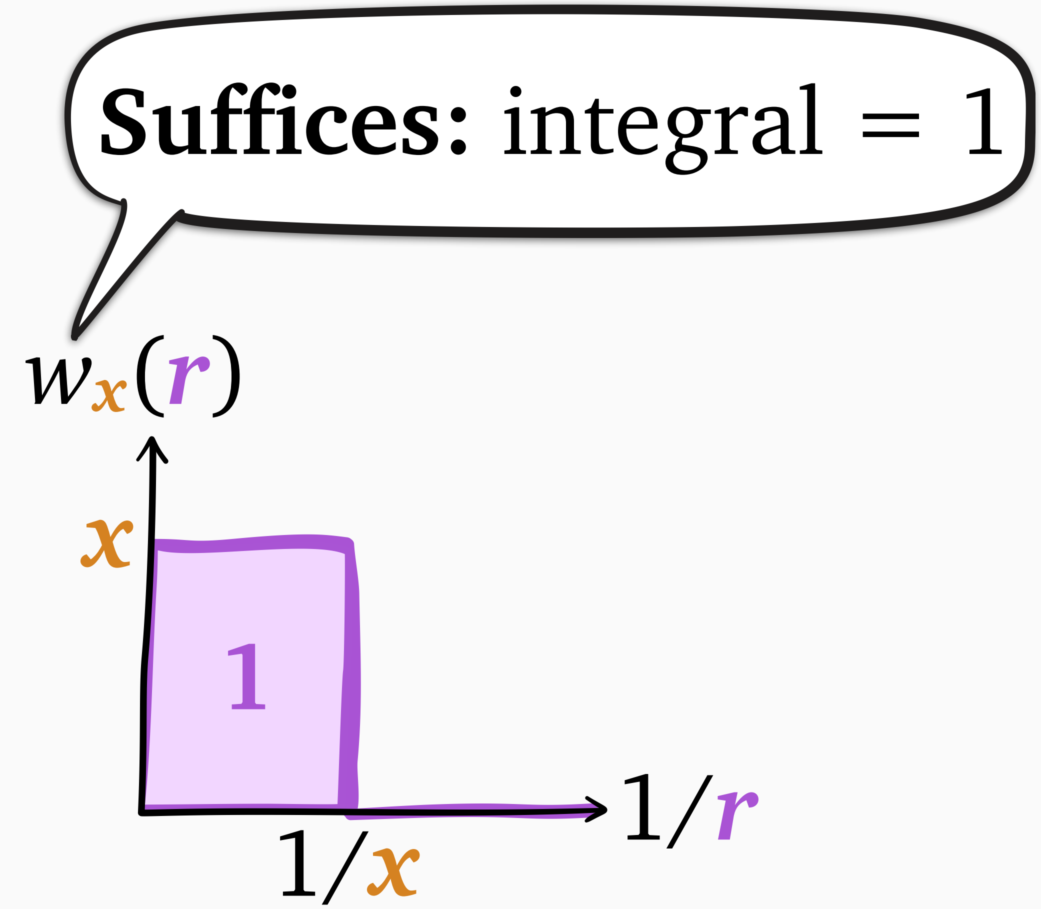




$$w_x(r) = r\text{-work of } \textit{job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$

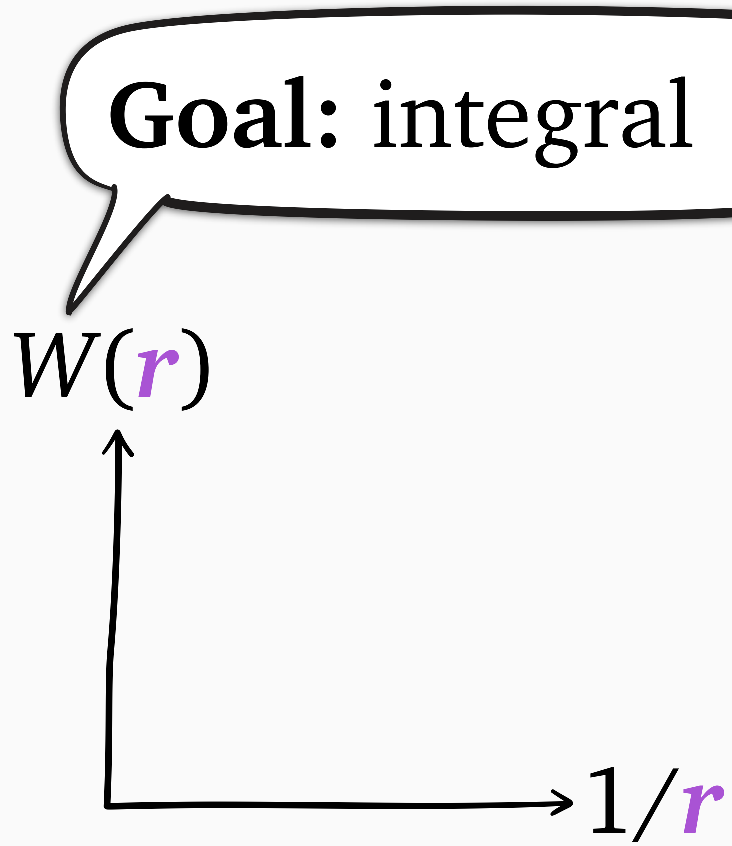
From r-work to number of jobs N

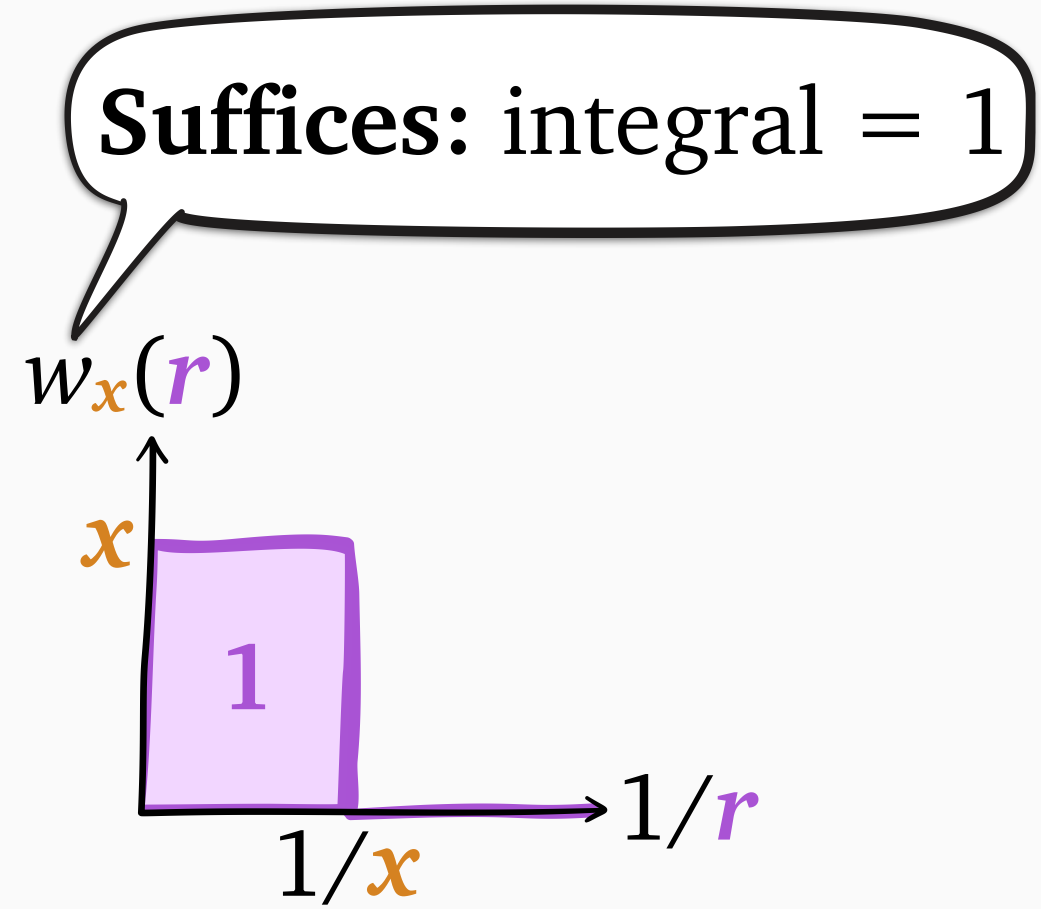




$$w_x(r) = r\text{-work of } \textit{job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$

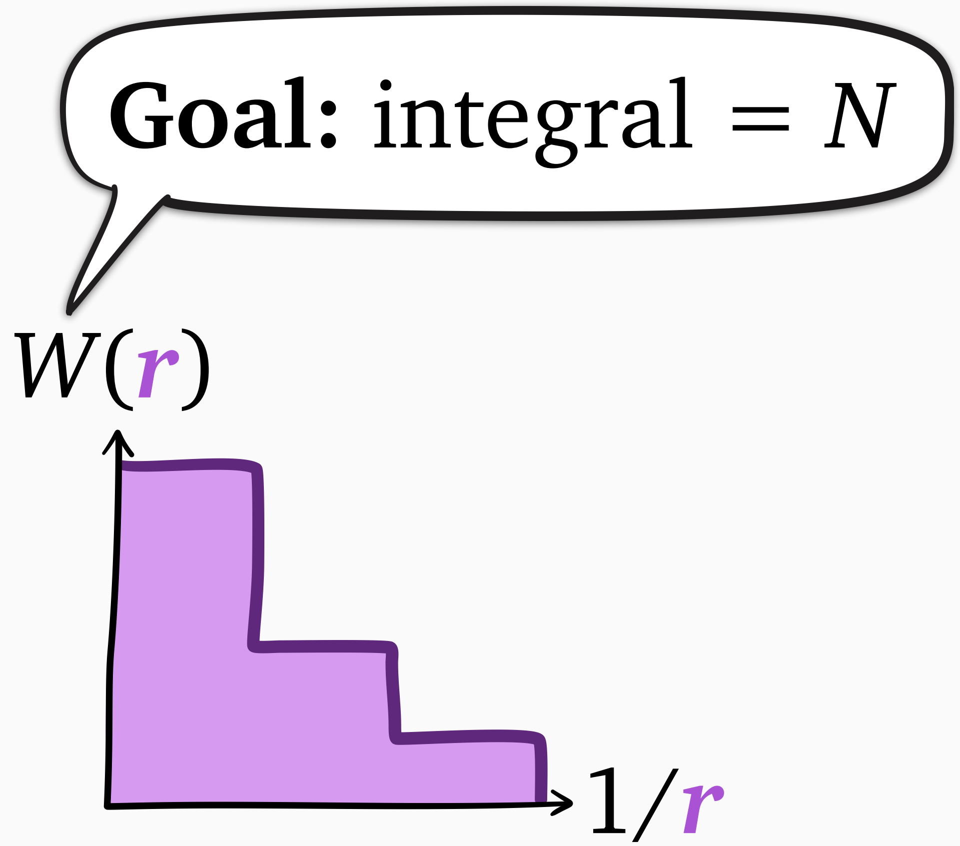
From r-work to number of jobs N

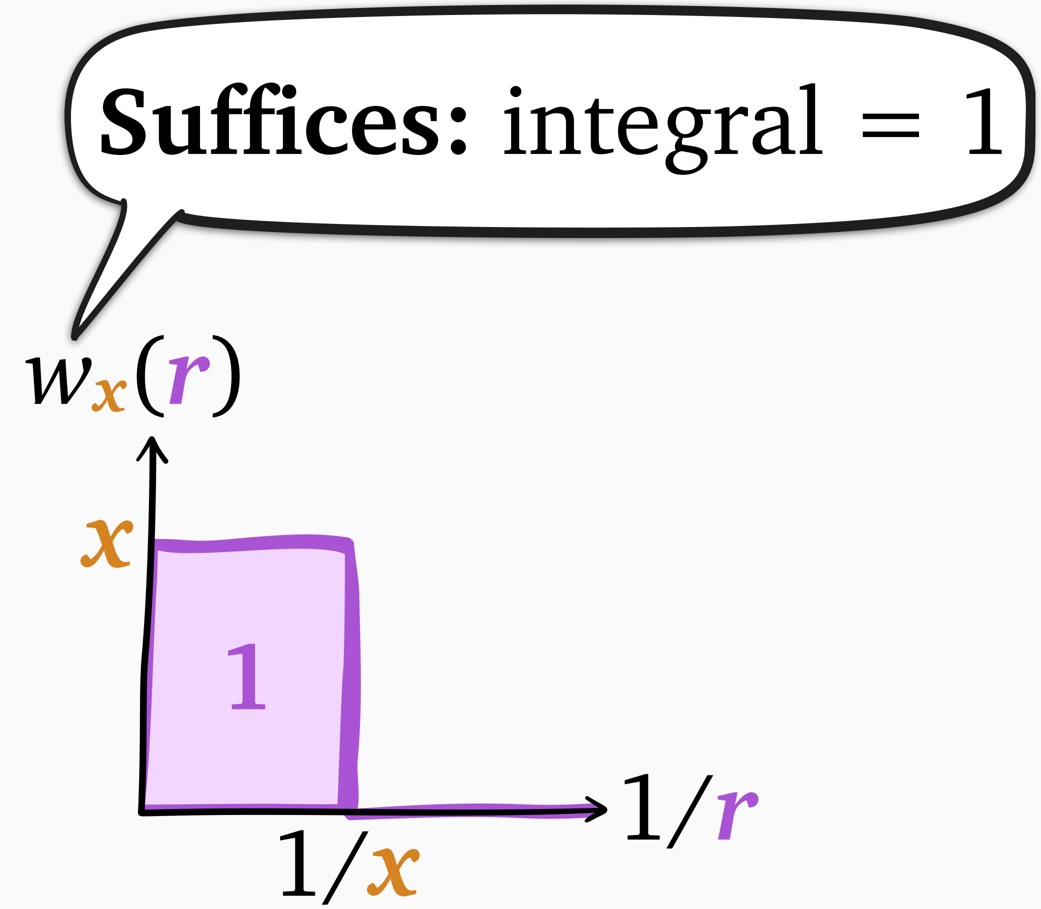




$$w_x(r) = r\text{-work of } \textit{job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$

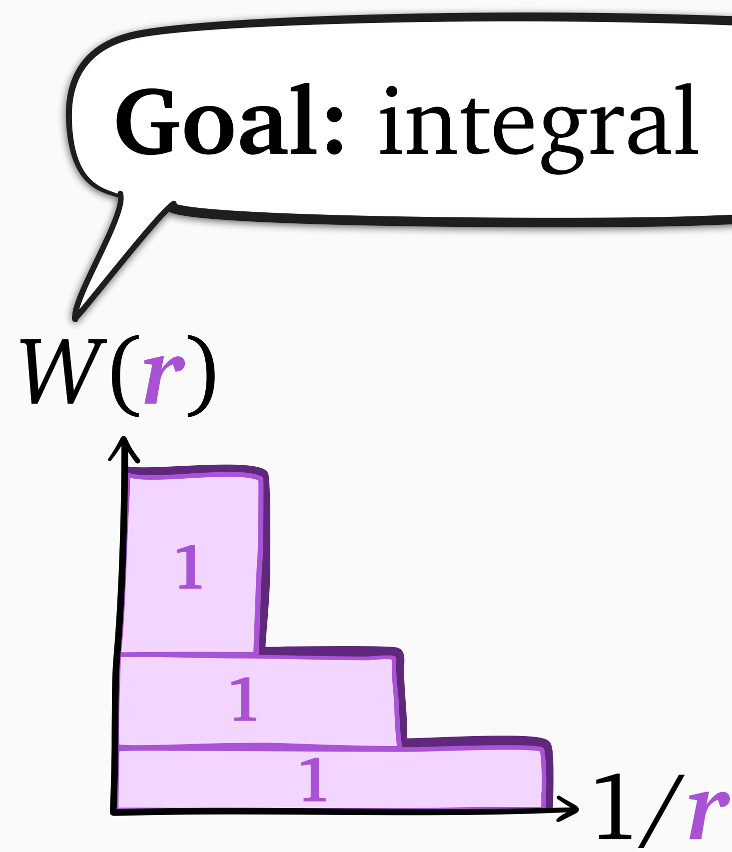
From r-work to number of jobs N

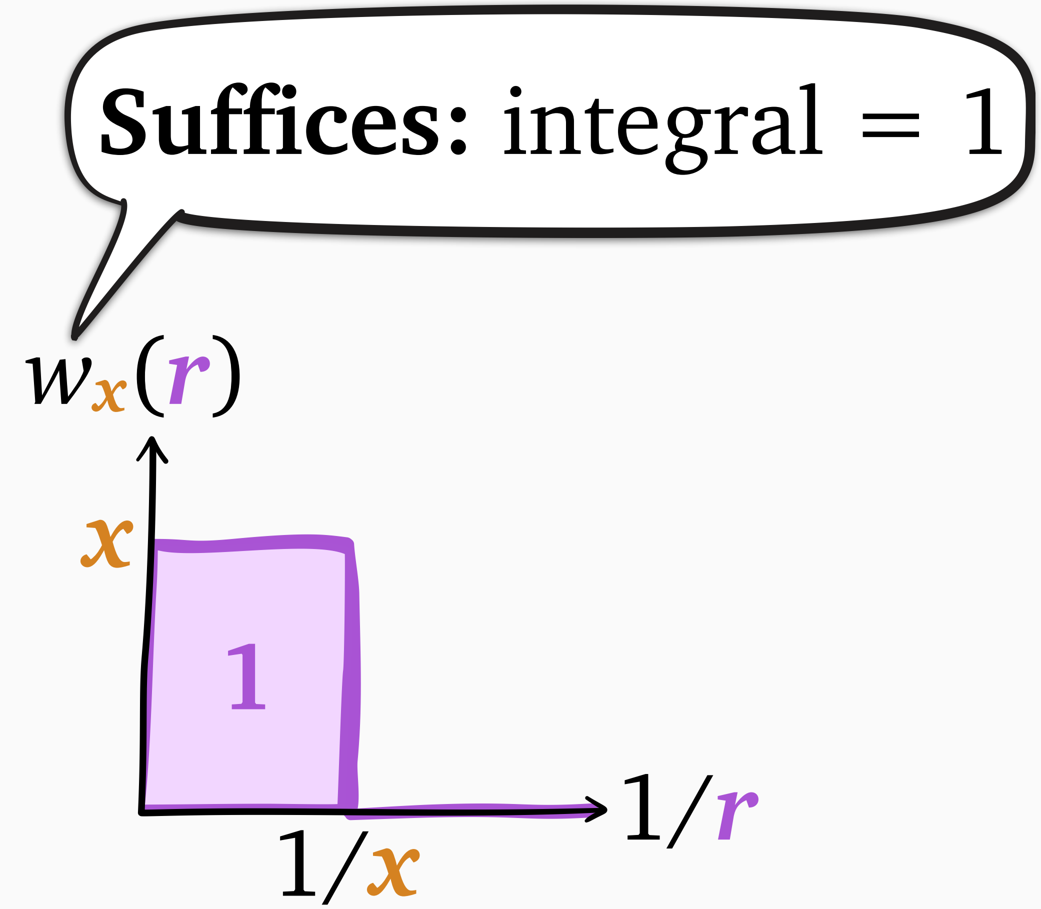




$$w_x(r) = r\text{-work of } \textit{job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$

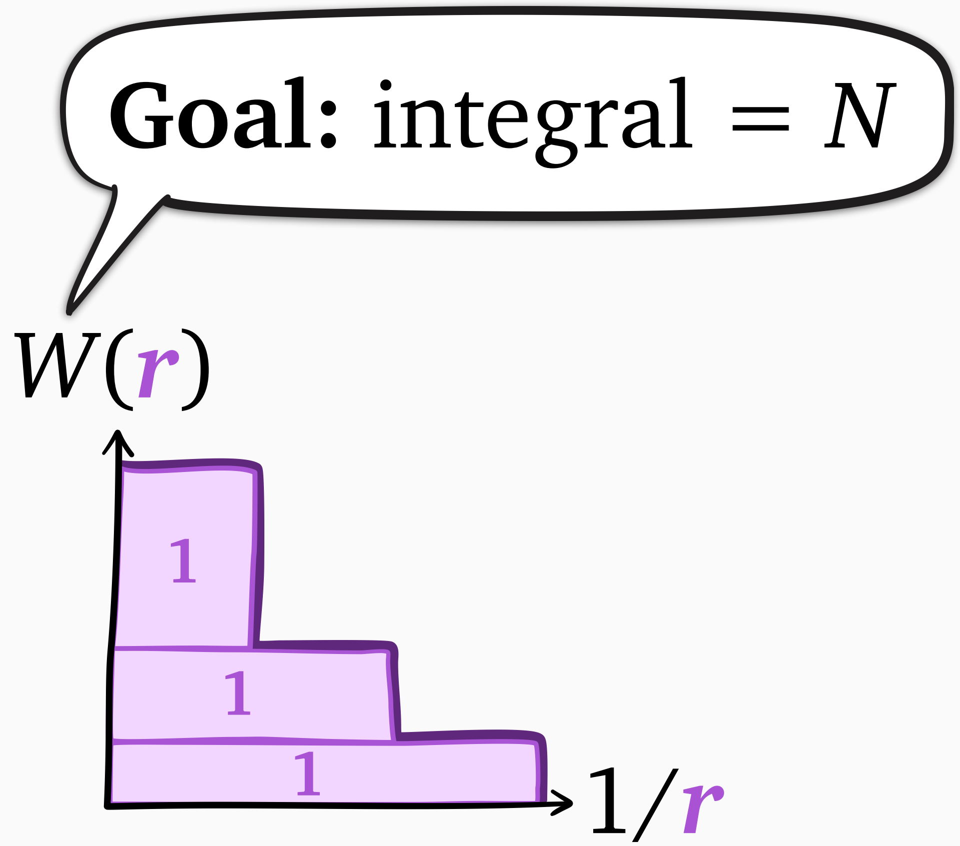
From r-work to number of jobs N

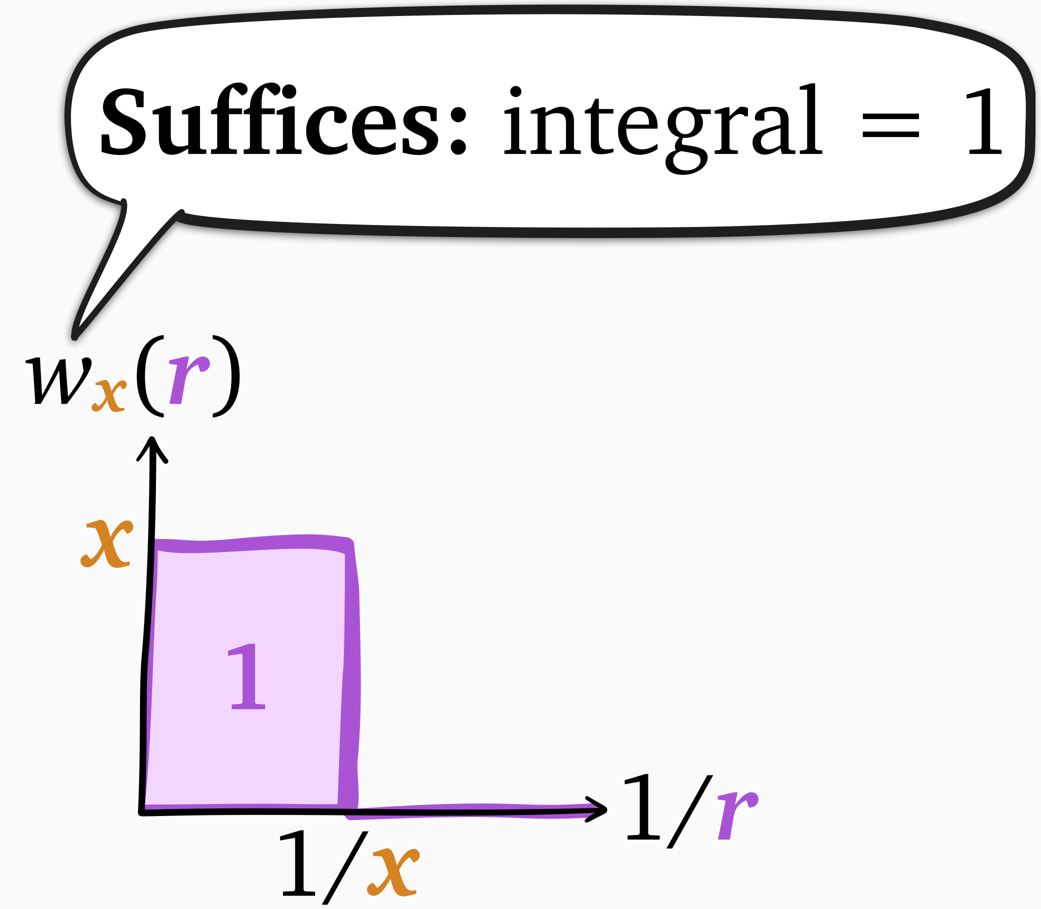




$$w_x(r) = r\text{-work of } \textit{job} \text{ of rem. size } x = \begin{cases} 0 & \text{if } r < x \\ x & \text{if } r \geq x \end{cases}$$

From r-work to number of jobs N





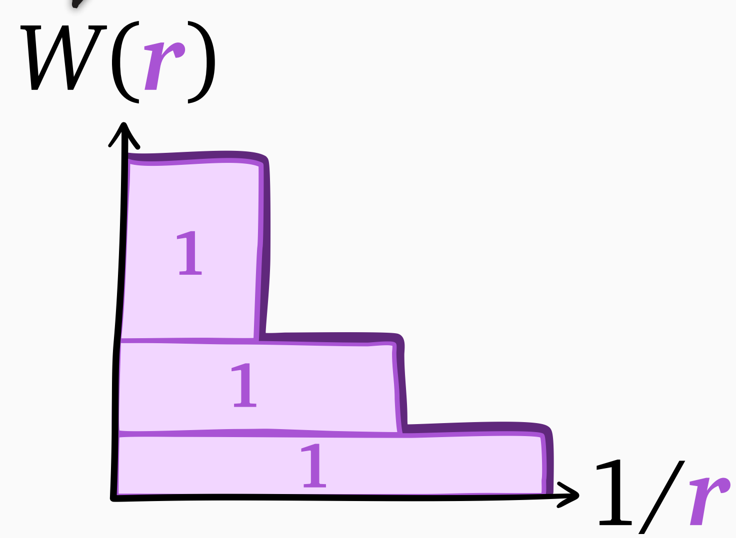
NEW!

Theorem:

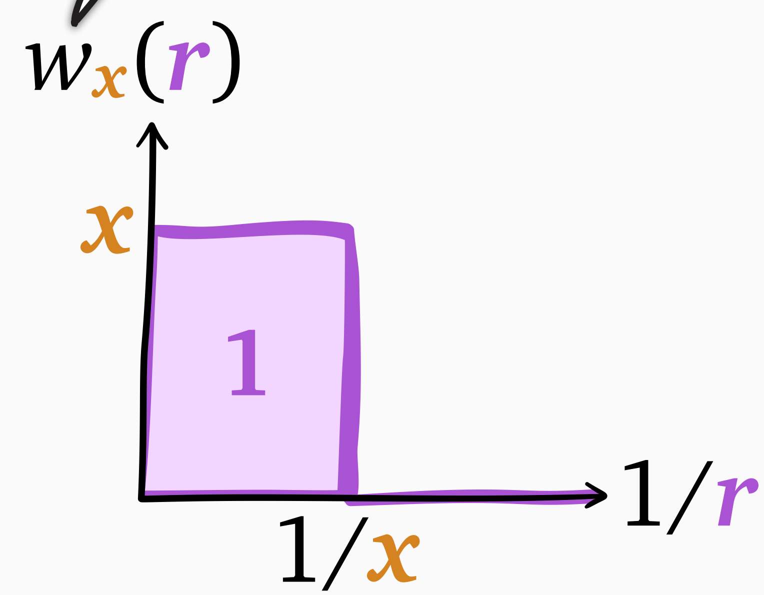
$$N = \int_0^\infty \frac{W(r)}{r^2}\,\mathrm{d}r$$

From r-work to number of jobs N

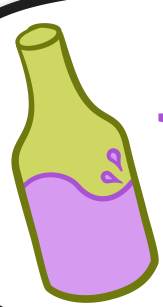
Goal: integral $= N$



Suffices: integral $= 1$



NEW!



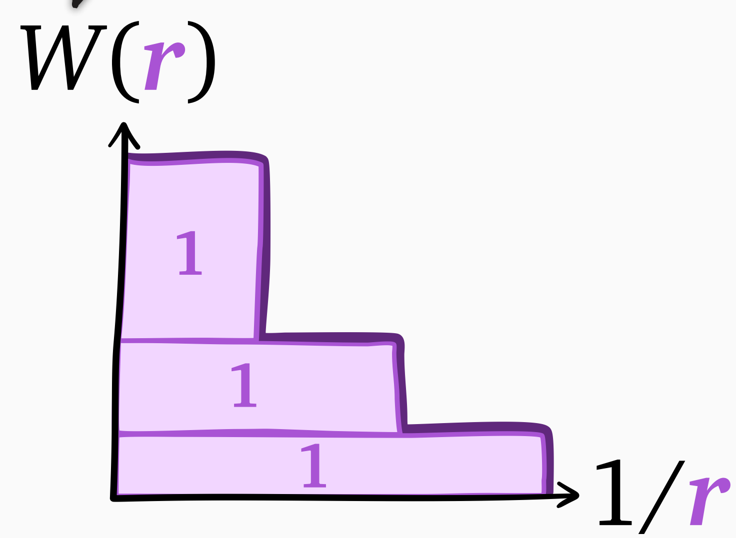
WINE

Theorem:

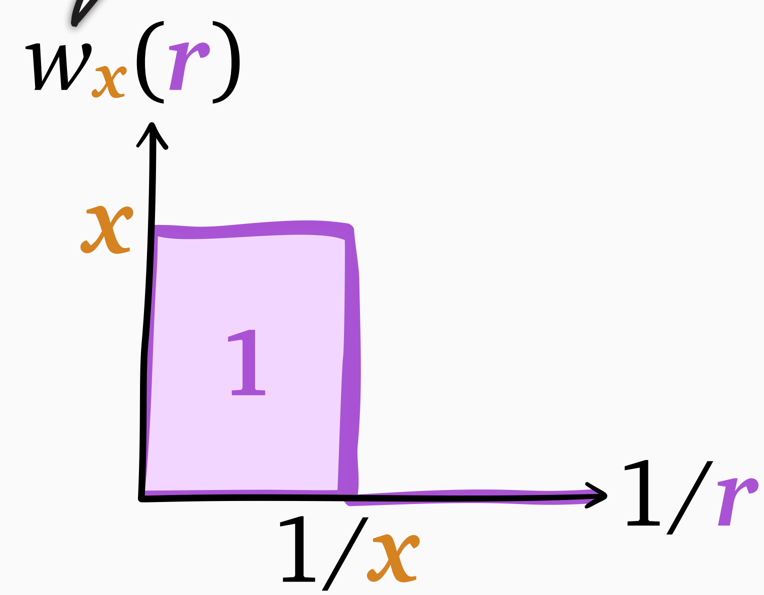
$$N = \int_0^\infty \frac{W(r)}{r^2}\,\mathrm{d}r$$

From r-work to number of jobs N

Goal: integral $= N$



Suffices: integral $= 1$

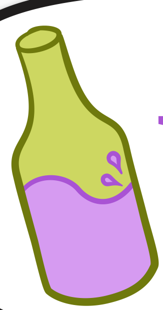


NEW!

Theorem:

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

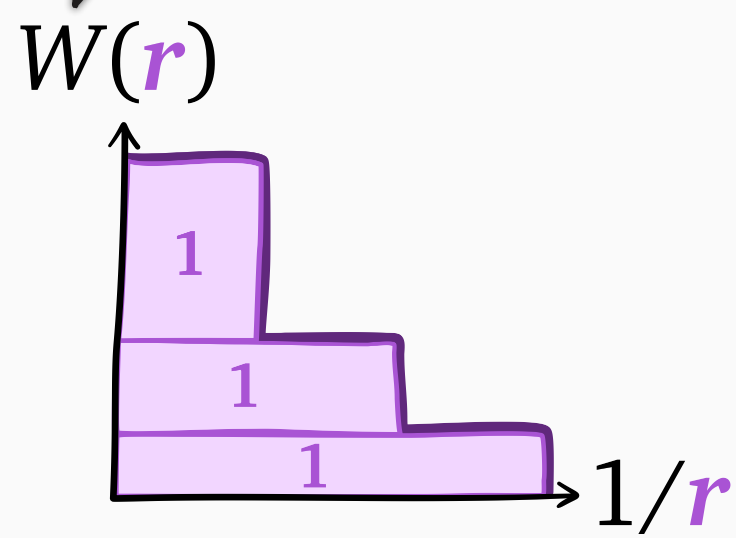
uses **rank** = rem. size



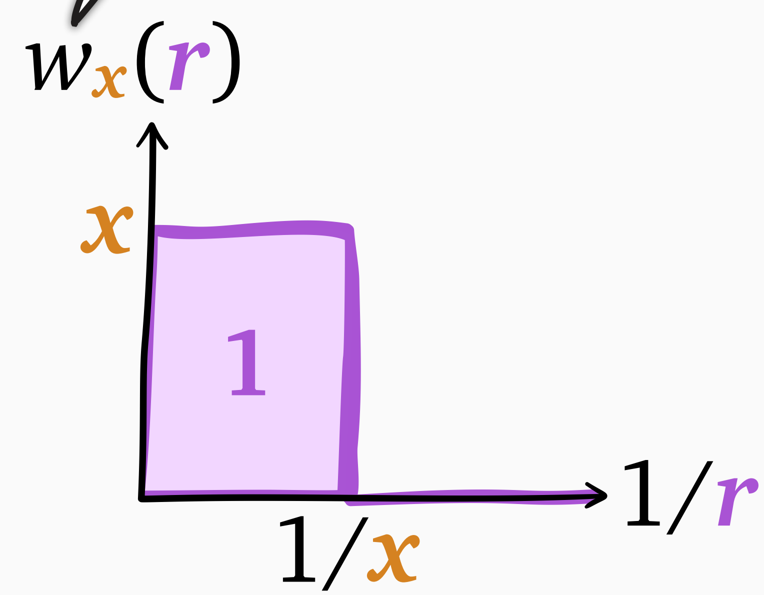
WINE

From r-work to number of jobs N

Goal: integral $= N$



Suffices: integral $= 1$



NEW!

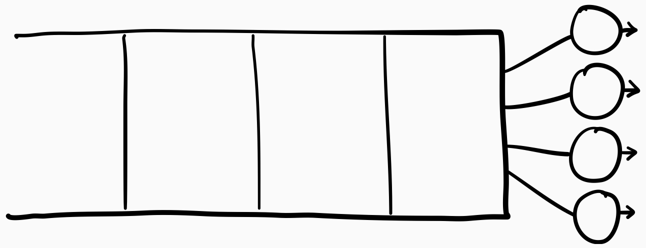
WINE

Theorem: under *any* policy,

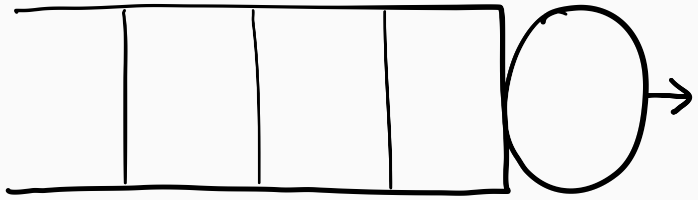
$$N = \int_0^\infty \frac{W(r)}{r^2}\,\mathrm{d}r$$

uses **rank** = rem. size

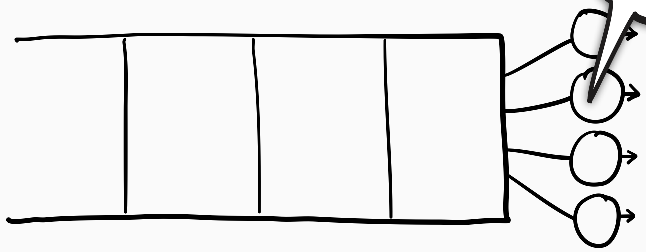
SRPT-k



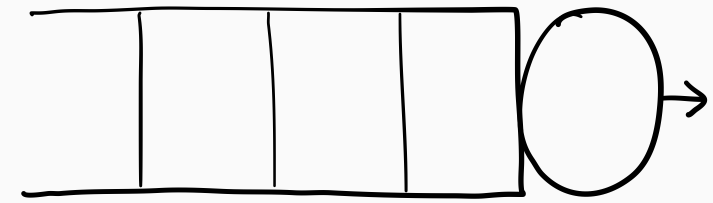
SRPT-1

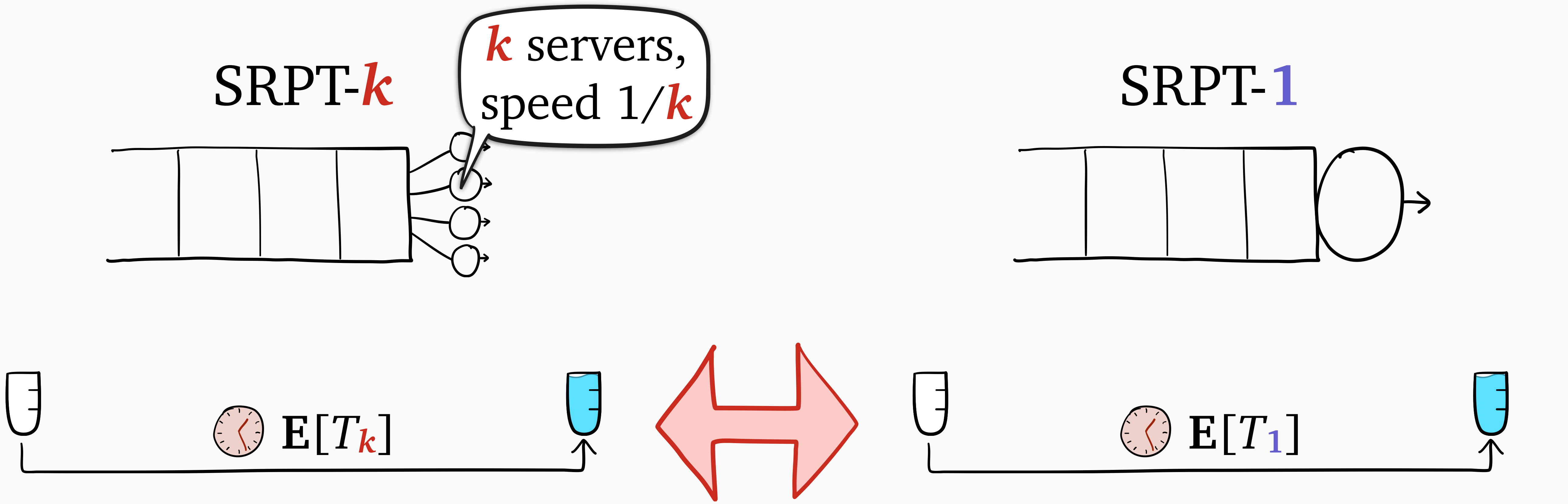


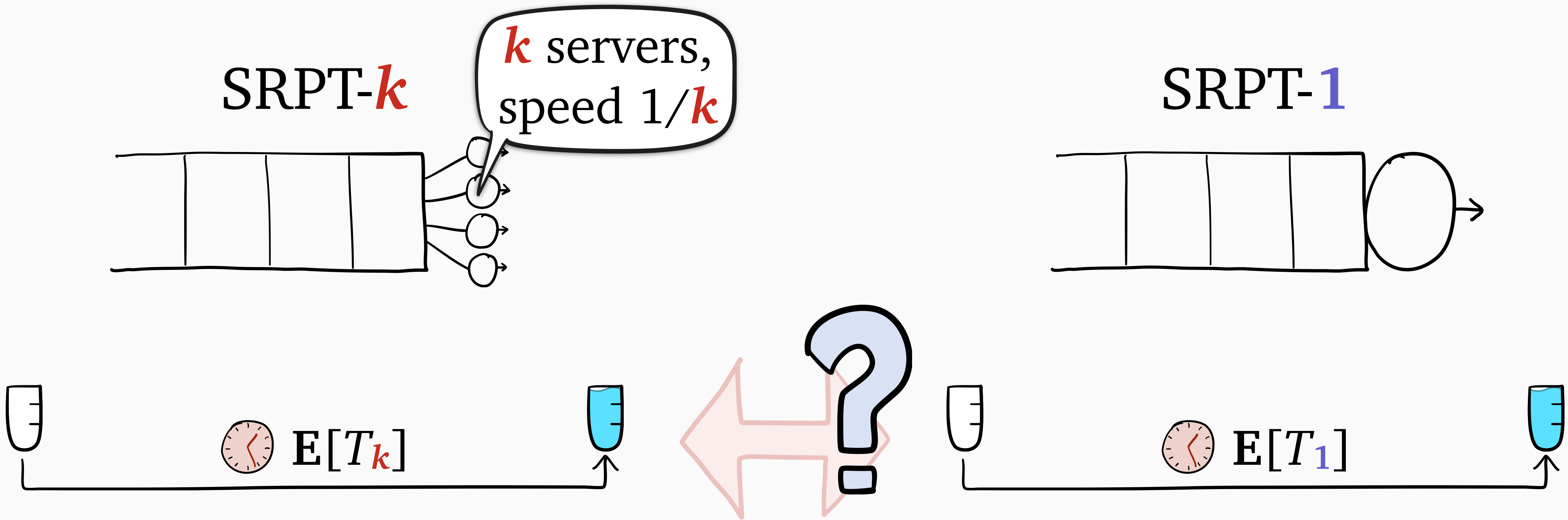
SRPT-k

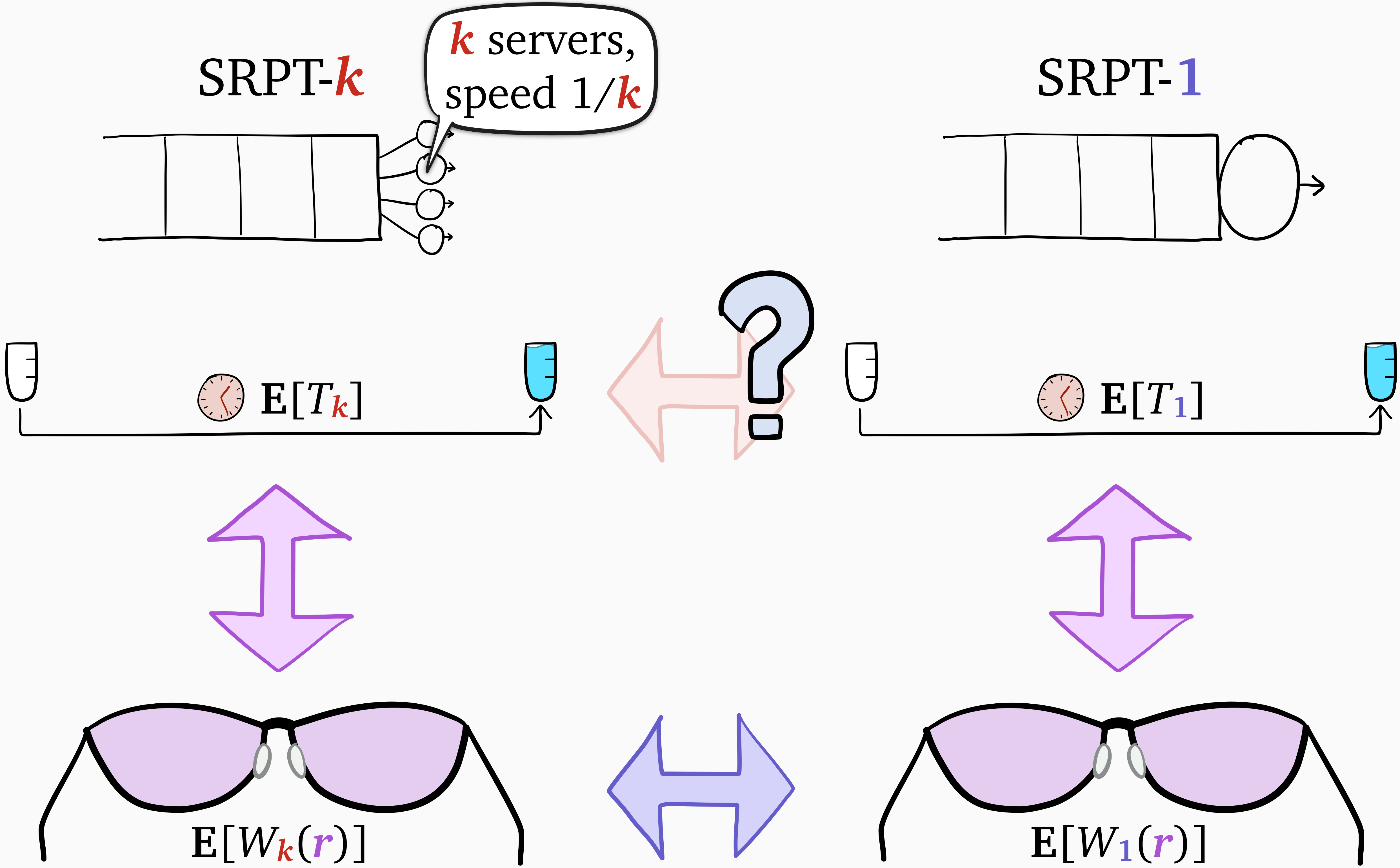


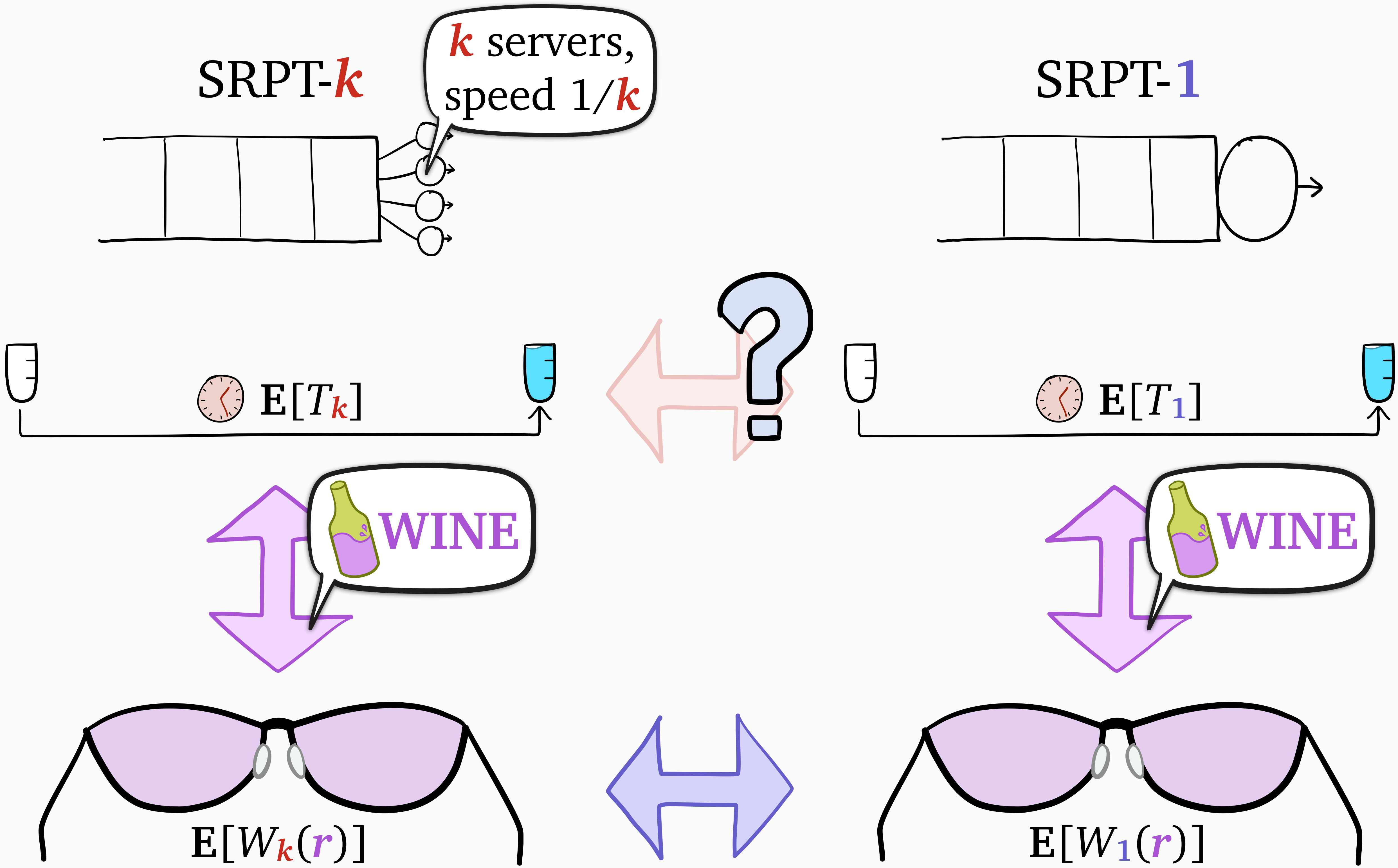
SRPT-1

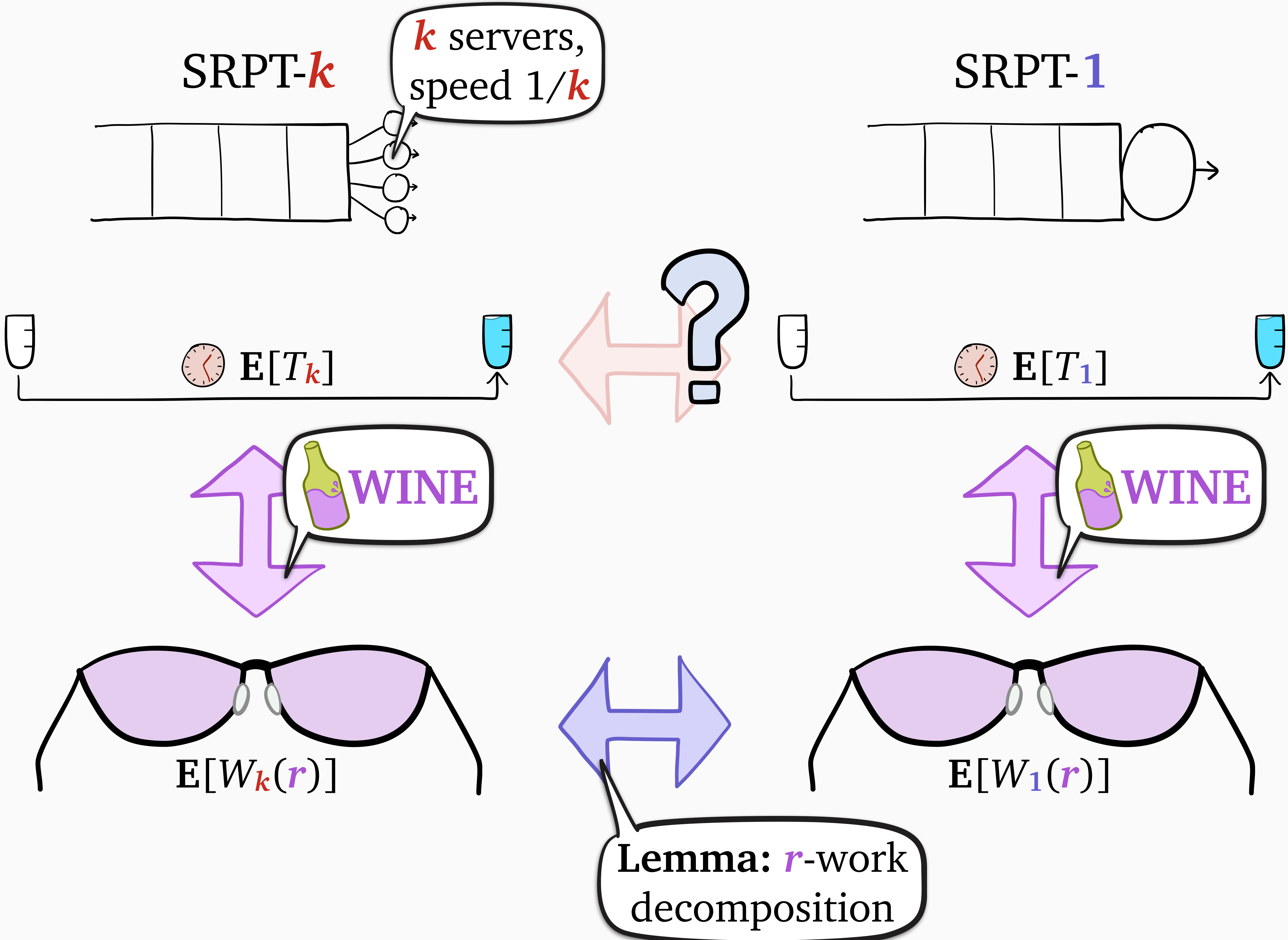


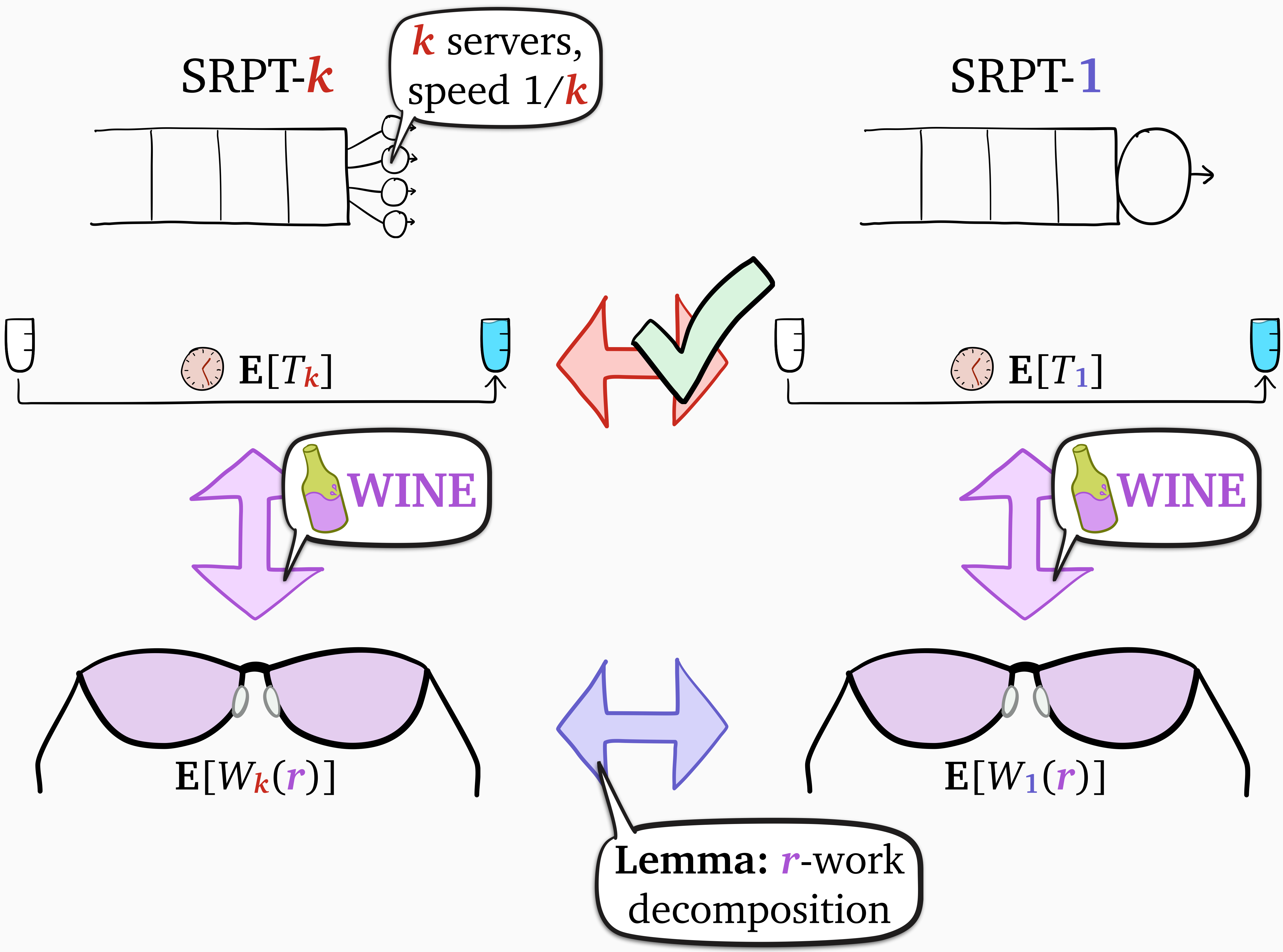

SRPT-k
k servers,
speed 1/k
SRPT-1
E[T_k]
E[T_1]

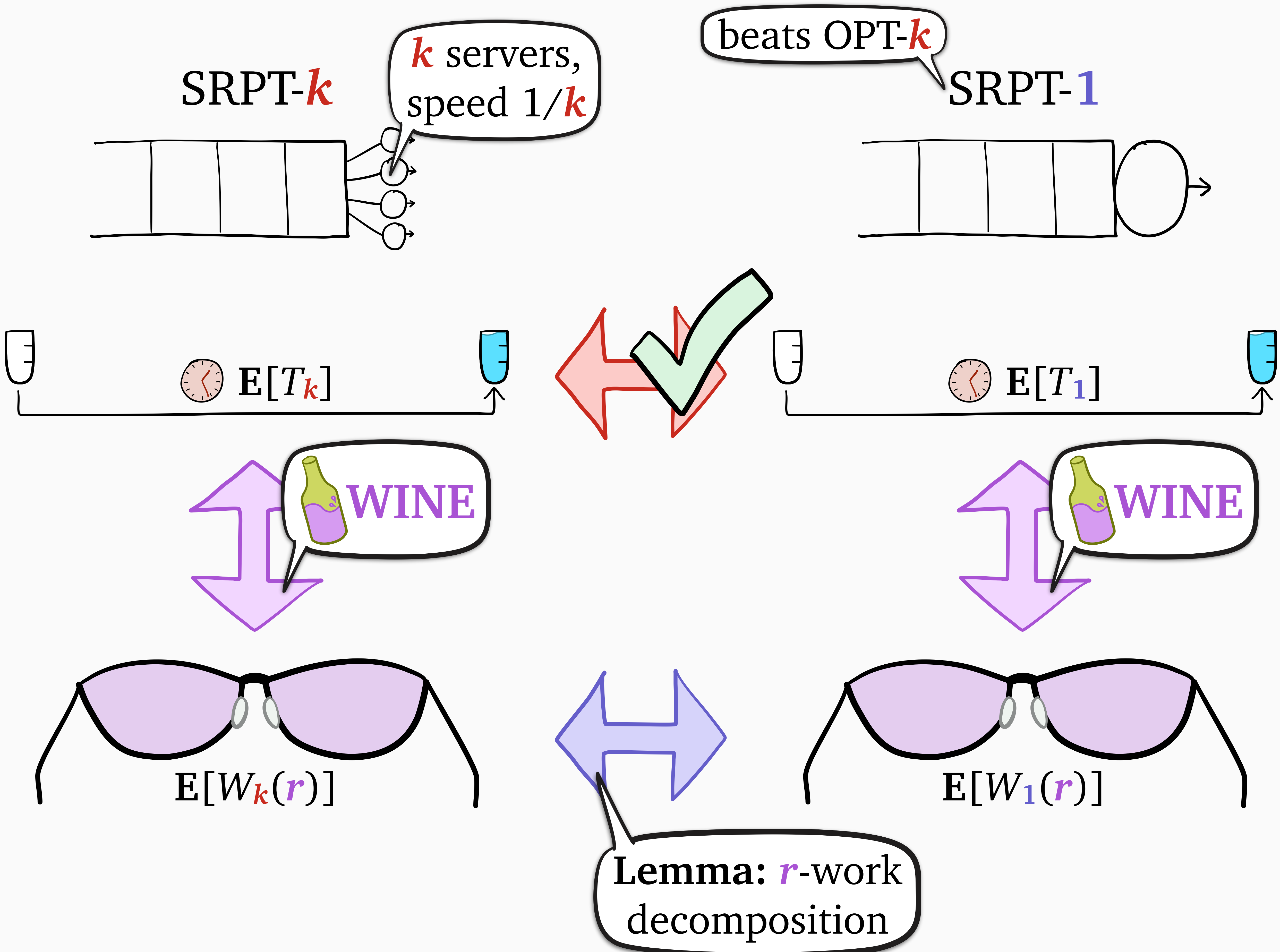

SRPT-k
k servers, speed 1/k
SRPT-1
E[T_k]
E[T_1]


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
E[T_k]
E[T_1]
WINE
WINE
E[W_k(r)]
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
$\mathbf{E}[W_k(r)]$
$\mathbf{E}[W_1(r)]$
Lemma: r-work decomposition



SRPT-k
k servers, speed $1/k$
beats OPT-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

Additive bounds for SRPT-*k*

Lemma:

$$\mathbf{E}[N_k] = \mathbf{E}[N_1] + \int_0^\infty \frac{\mathbf{E}[W_k(r)] - \mathbf{E}[W_1(r)]}{r^2}\, \mathrm{d}r$$

Additive bounds for SRPT-*k*

Lemma:

$$\mathbf{E}[N_k] = \mathbf{E}[N_1] + \int_0^\infty \frac{\mathbf{E}[W_k(r)] - \mathbf{E}[W_1(r)]}{r^2}\,\mathrm{d}r$$

Lemma: in **worst case**,

$$W_k(r) \leq W_1(r) + kr$$

Additive bounds for SRPT-k

Lemma:

$$\mathbf{E}[N_k] = \mathbf{E}[N_1] + \int_0^\infty \frac{\mathbf{E}[W_k(r)] - \mathbf{E}[W_1(r)]}{r^2}\,\mathrm{d}r$$

Lemma: in **worst case**,

$$W_k(r) \le W_1(r) + kr$$

TCS

$$\mathbf{E}[N_k] \le \mathbf{E}[N_1] + 2k + k \log \frac{\text{max size}}{\text{min size}}$$

Additive bounds for SRPT-k

Lemma:

$$\mathbf{E}[N_k] = \mathbf{E}[N_1] + \int_0^\infty \frac{\mathbf{E}[W_k(r)] - \mathbf{E}[W_1(r)]}{r^2}\,\mathrm{d}r$$

Lemma: in **worst case**,

$$W_k(r) \le W_1(r) + kr$$

TCS

$$\mathbf{E}[N_k] \le \mathbf{E}[N_1] + 2k + k\log\frac{\text{max size}}{\text{min size}}$$

NEW! **Queueing**

$$\mathbf{E}[N_k] \le \mathbf{E}[N_1] + 4(k-1)\log\frac{1}{1-\rho}$$

Additive bounds for SRPT-k

Lemma:

$$\mathbf{E}[N_k] = \mathbf{E}[N_1] + \int_0^\infty \frac{\mathbf{E}[W_k(r)] - \mathbf{E}[W_1(r)]}{r^2}\, \mathrm{d}r$$

Lemma: in **worst case**,

$$W_k(r) \leq W_1(r) + kr$$

TCS

$$\mathbf{E}[N_k] \leq \mathbf{E}[N_1] + 2k + k \log \frac{\text{max size}}{\text{min size}}$$

NEW! **Queueing**

$$\mathbf{E}[N_k] \leq \mathbf{E}[N_1] + 4(k-1)\log\frac{1}{1-\rho}$$

dominant term

Additive bounds for SRPT-k

Lemma:

$$\mathbf{E}[N_k] = \mathbf{E}[N_1] + \int_0^\infty \frac{\mathbf{E}[W_k(r)] - \mathbf{E}[W_1(r)]}{r^2}\,\mathrm{d}r$$

Lemma: in **worst case**,

$$W_k(r) \le W_1(r) + kr$$

uses stronger **stochastic** lemma

TCS

$$\mathbf{E}[N_k] \le \mathbf{E}[N_1] + 2k + k \log \frac{\text{max size}}{\text{min size}}$$

NEW! **Queueing**

$$\mathbf{E}[N_k] \le \mathbf{E}[N_1] + 4(k-1)\log \frac{1}{1-\rho}$$

dominant term

Today's talk

scheduling with

multiple servers

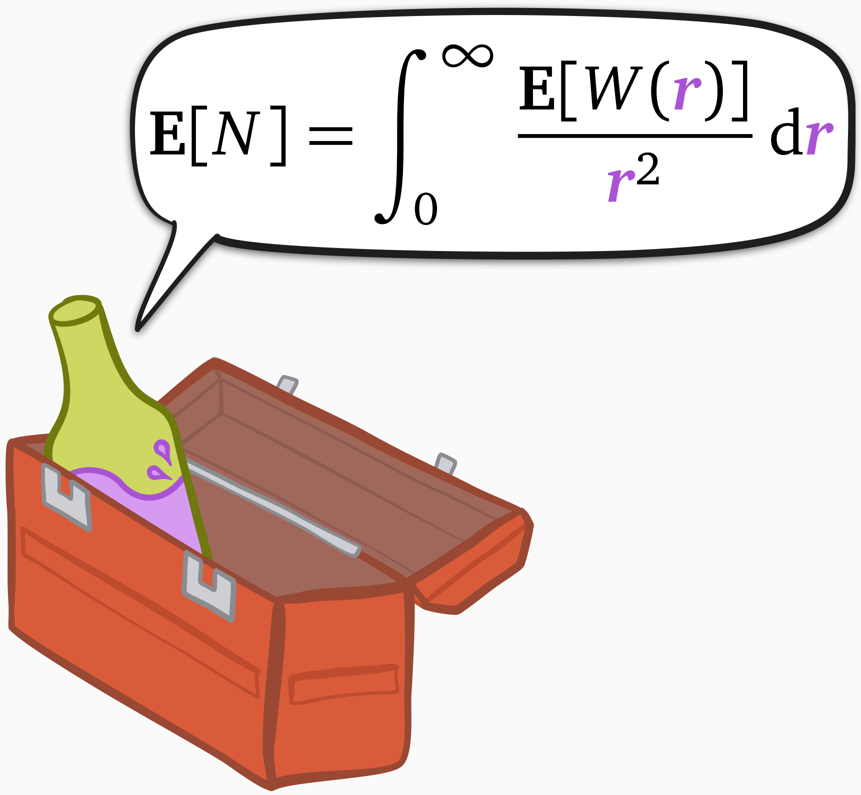
TCS **Queueing**

Today's talk

scheduling with

multiple servers

TCS **Queueing**

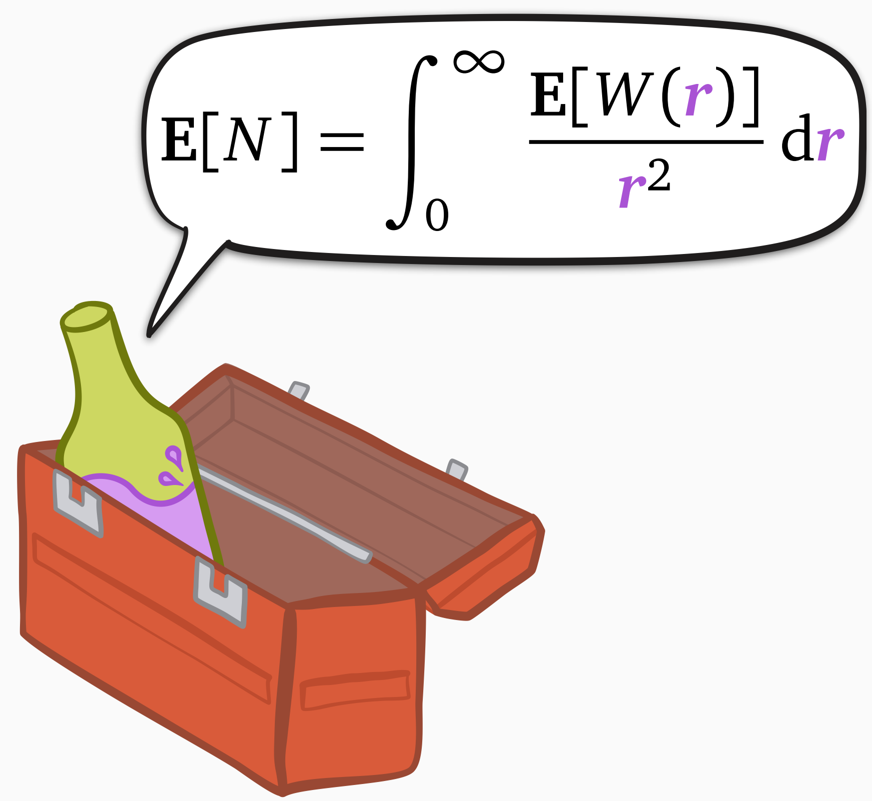


Today's talk

scheduling with

multiple servers

TCS 🤝 **Queueing**



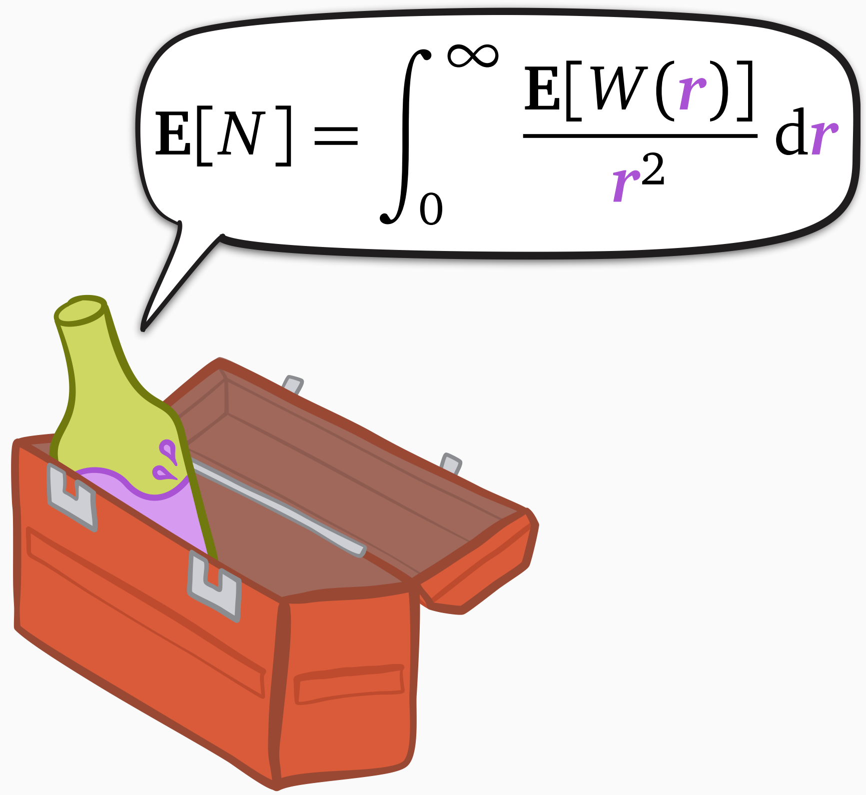
Today's talk

scheduling with

multiple servers

TCS 🤝 **Queueing**

- SRPT-k is good



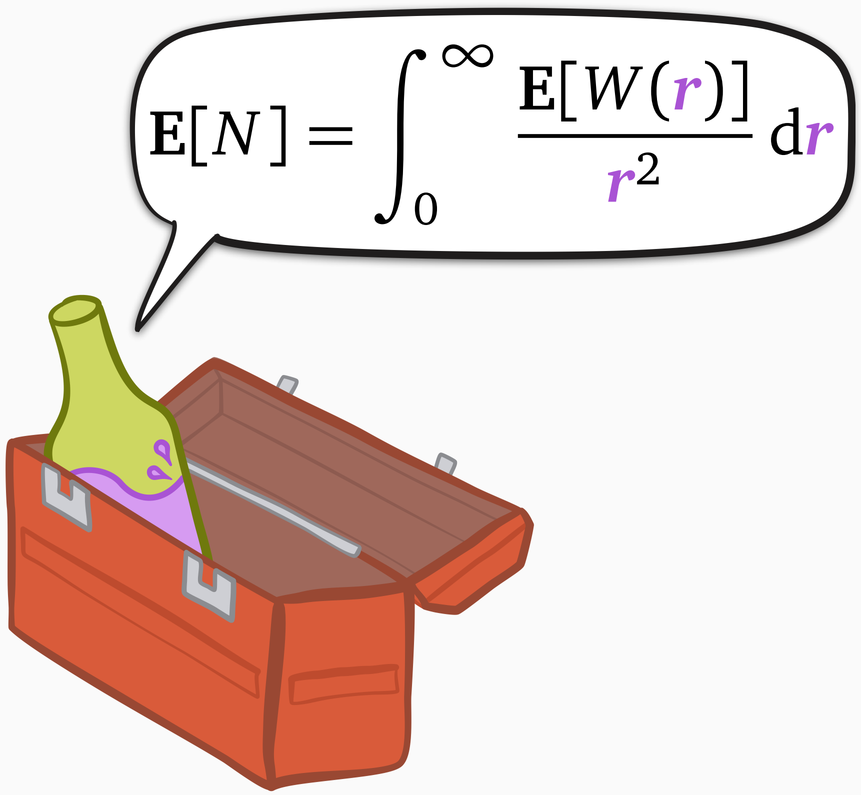
Today's talk

scheduling with

multiple servers

TCS 🤝 **Queueing**

- SRPT-*k* is good
- In general: good to adapt optimal single-server policy



Today's talk

scheduling with

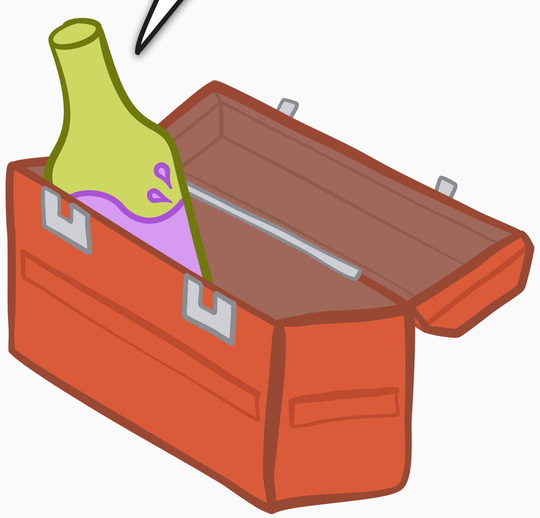
multiple servers

TCS 🤝 **Queueing**

- SRPT-*k* is good
- In general: good to adapt optimal single-server policy

TCS: RMLF
Queueing: **Gittins**

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



Today's talk

scheduling with

multiple servers

TCS 🤝 **Queueing**

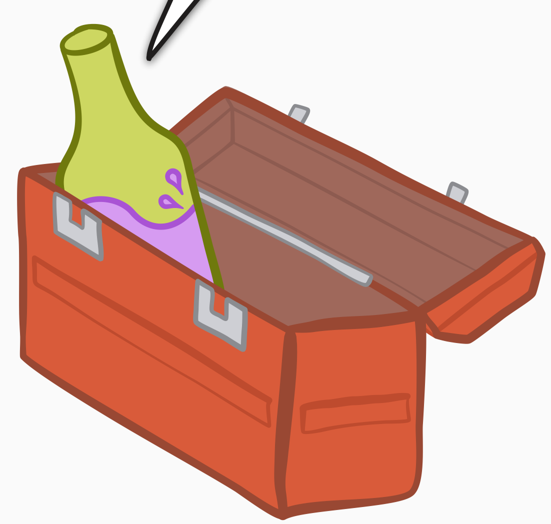
- SRPT-*k* is good
- In general: good to adapt optimal single-server policy

TCS: RMLF
Queueing: **Gittins**

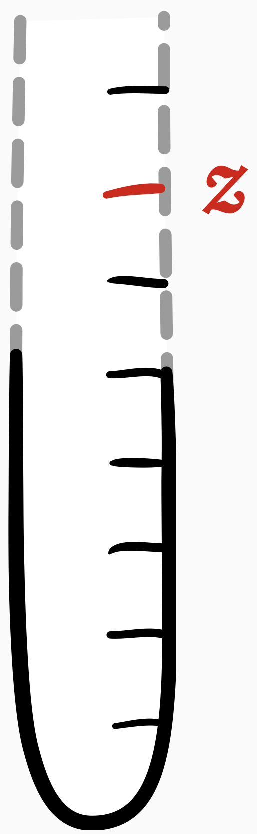
scheduling with

noisy predictions

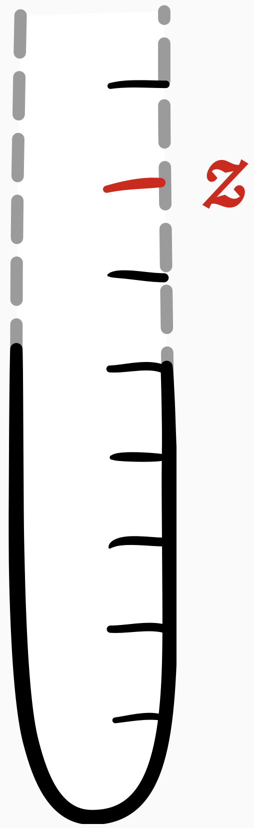
$$\mathbf{E}[N] = \int_0^\infty \frac{\mathbf{E}[W(r)]}{r^2}\,\mathrm{d}r$$



Noisily predicted job sizes



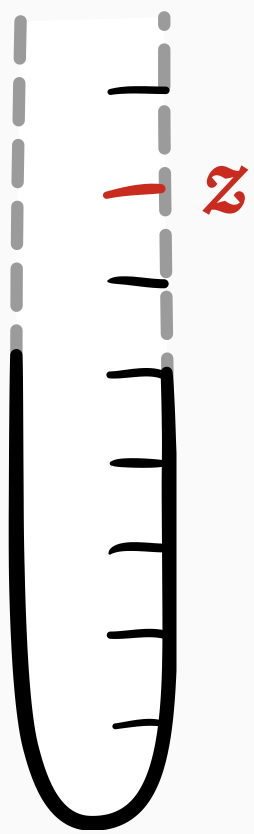
Noisily predicted job sizes



Model: (β, α)-bounded noise

true size s $\Rightarrow$ **estimated** size $\boldsymbol{z} \in [\beta s, \alpha s]$

Noisily predicted job sizes

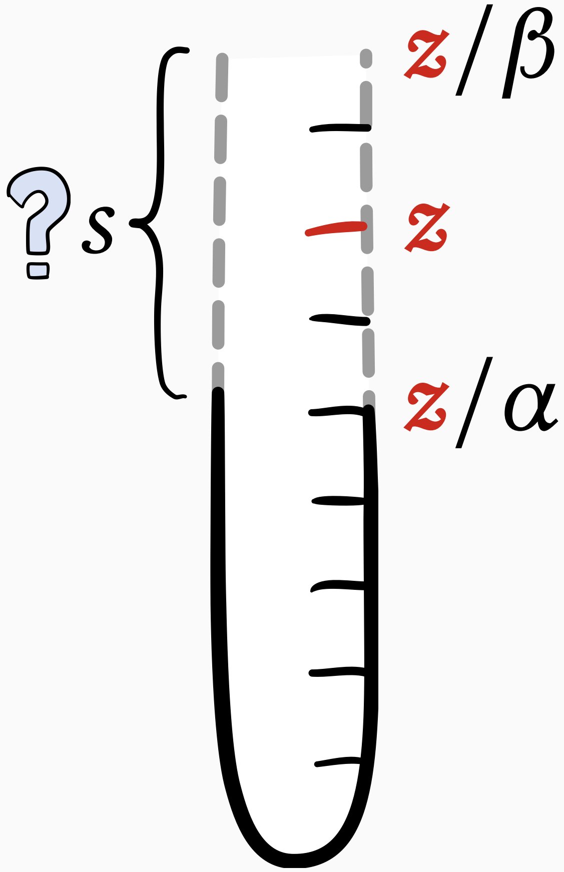


below above

Model: (β, α)-bounded noise

true size s $\Rightarrow$ **estimated** size $z \in [\beta s, \alpha s]$

Noisily predicted job sizes

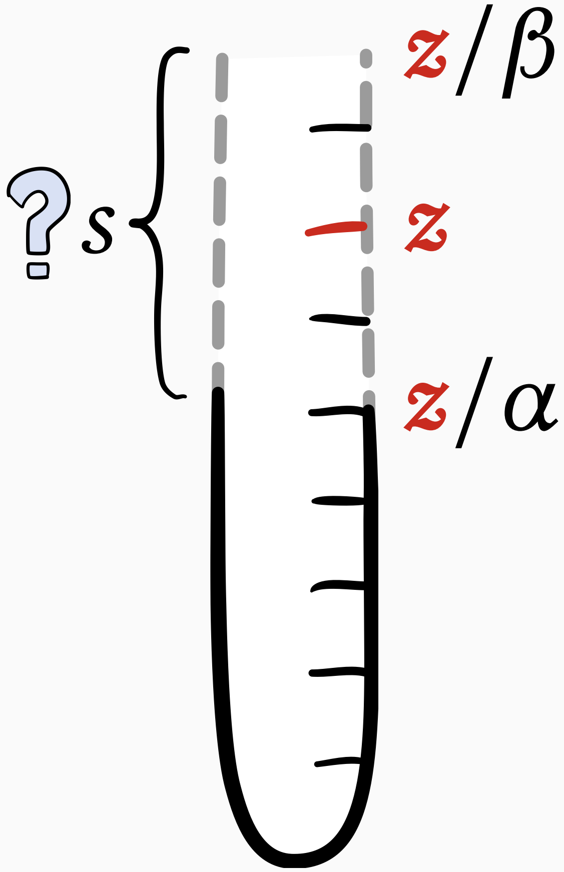


below above

Model: (β, α)-bounded noise

true size s $\Rightarrow$ **estimated** size $z \in [\beta s, \alpha s]$

Noisily predicted job sizes



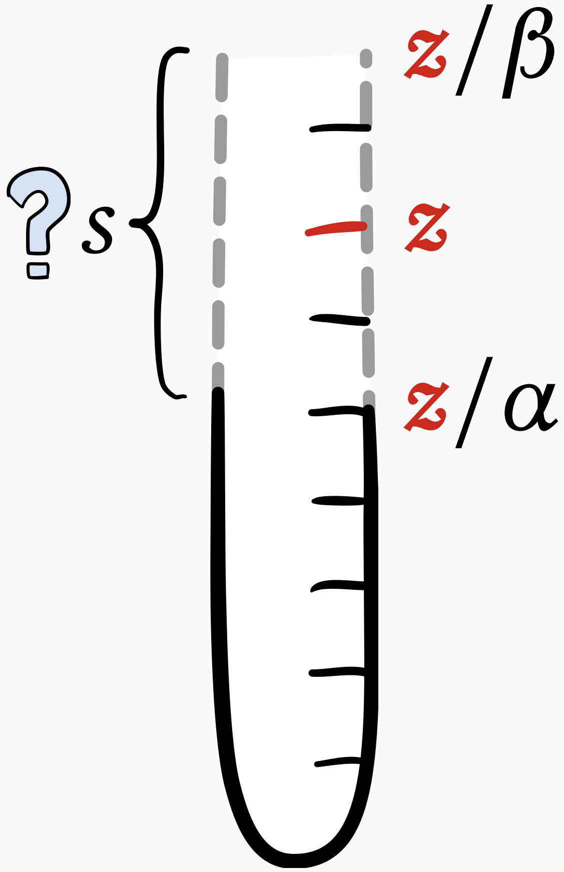
below above

Model: (β, α)-bounded noise

true size s $\Rightarrow$ **estimated** size $z \in [\beta s, \alpha s]$

Queueing: from *joint distribution* (S, Z)

Noisily predicted job sizes



below above

Model: (β, α)-bounded noise

true size s $\Rightarrow$ **estimated** size $z \in [\beta s, \alpha s]$

Queueing: from *joint distribution* (S, Z)

Goal: design a policy with "good" $\mathbf{E}[T]$ for

- *any* values of α, β
- *any* joint distribution (S, Z)

How well can we handle **noise**?

Definition: **distortion** is $\gamma = \frac{\alpha}{\beta}$

How well can we handle noise?

Definition: distortion is $\gamma = \frac{\alpha}{\beta}$

TCS [Azar, Leonardi, & Touitou, 2021 & 2022]:
pretty well, but need a sophisticated policy

$$\frac{\mathbf{E}[T_{\text{ZigZag}}]}{\mathbf{E}[T_{\text{SRPT}}]} \leq O(\gamma \log \gamma)$$

with nearly-matching $\Omega(\gamma)$ lower bound

How well can we handle **noise**?

Definition: **distortion** is $\gamma = \dfrac{\alpha}{\beta}$

TCS [Azar, Leonardi, & Touitou, 2021 & 2022]:
pretty well, but need a sophisticated policy

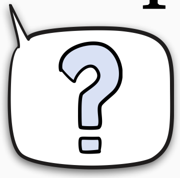
$$\frac{\mathbf{E}[T_{\text{ZigZag}}]}{\mathbf{E}[T_{\text{SRPT}}]} \le O(\gamma \log \gamma)$$

with nearly-matching $\Omega(\gamma)$ lower bound

How well can we handle **noise**?

Definition: **distortion** is $\gamma = \dfrac{\alpha}{\beta}$

TCS [Azar, Leonardi, & Touitou, 2021 & 2022]:
pretty well, but need a sophisticated policy



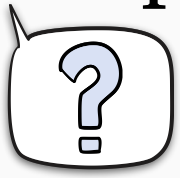
$$\frac{\mathbf{E}[T_{\text{ZigZag}}]}{\mathbf{E}[T_{\text{SRPT}}]} \le O(\gamma \log \gamma)$$

with nearly-matching $\Omega(\gamma)$ lower bound

How well can we handle **noise**?

Definition: **distortion** is $\gamma = \dfrac{\alpha}{\beta}$

TCS [Azar, Leonardi, & Touitou, 2021 & 2022]:
pretty well, but need a sophisticated policy

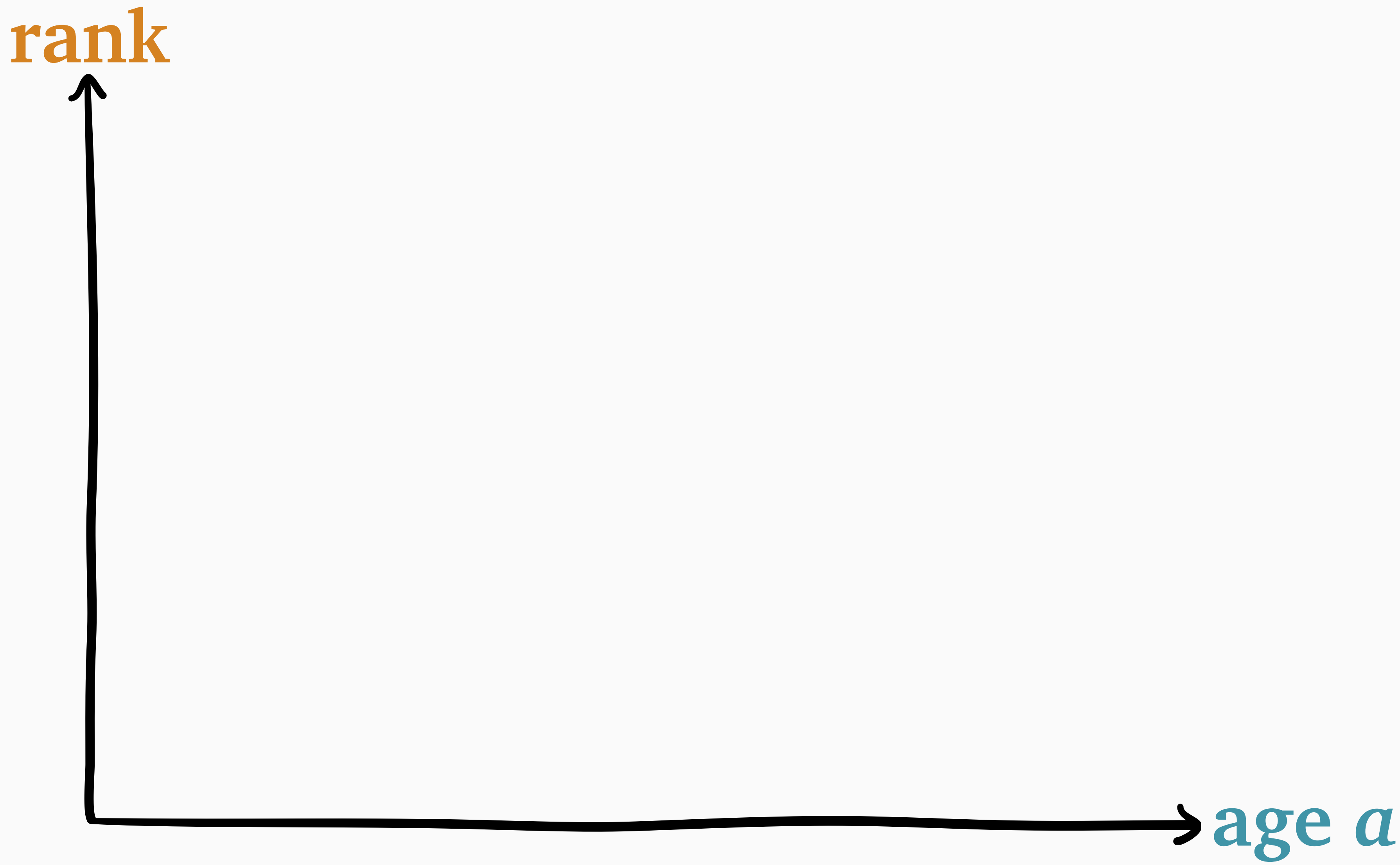


$$\frac{\mathbf{E}[T_{\mathrm{ZigZag}}]}{\mathbf{E}[T_{\mathrm{SRPT}}]} \leq O(\gamma \log \gamma)$$

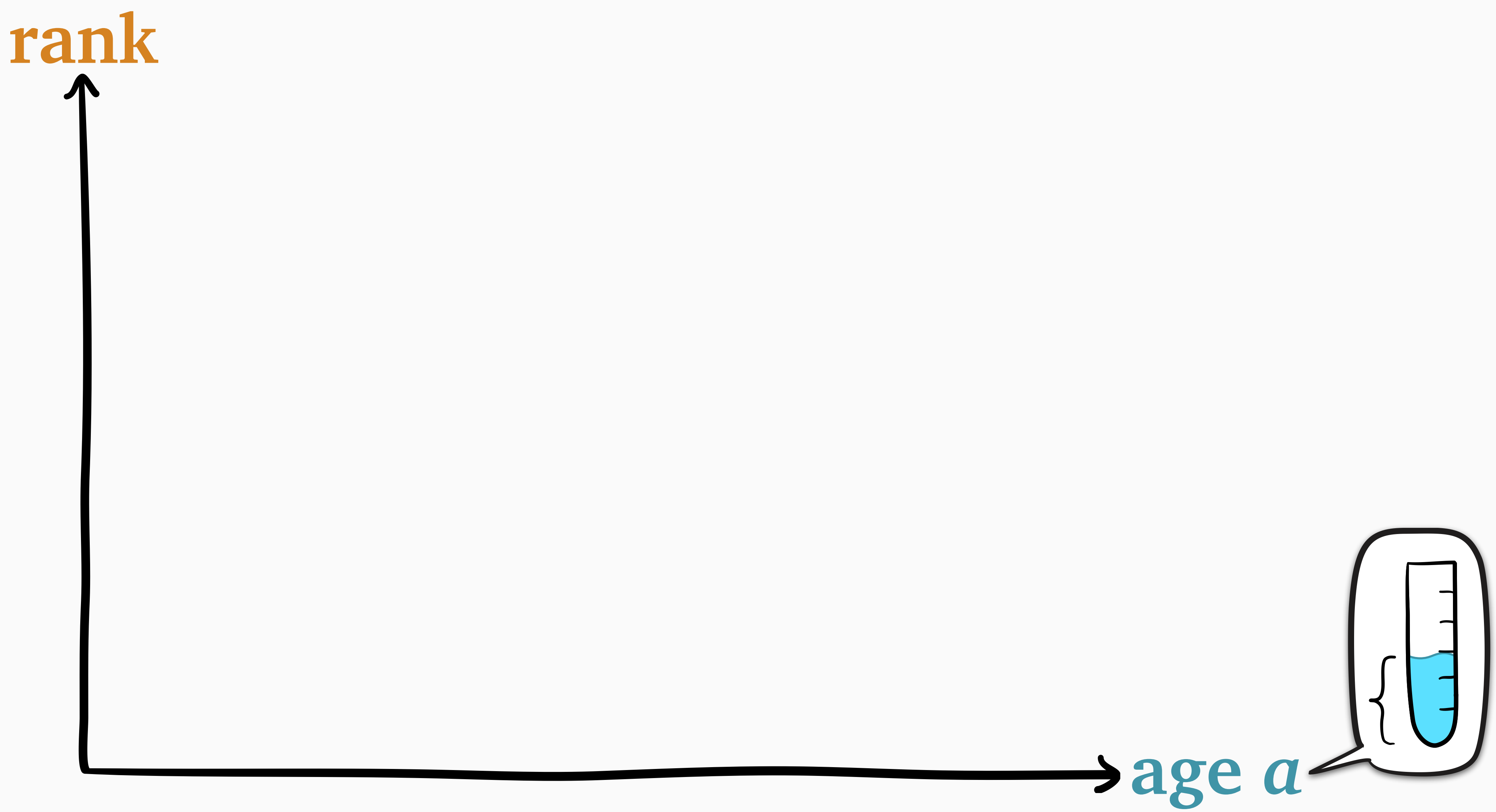
with nearly-matching $\Omega(\gamma)$ lower bound

Queueing: can we do better with simpler policy?

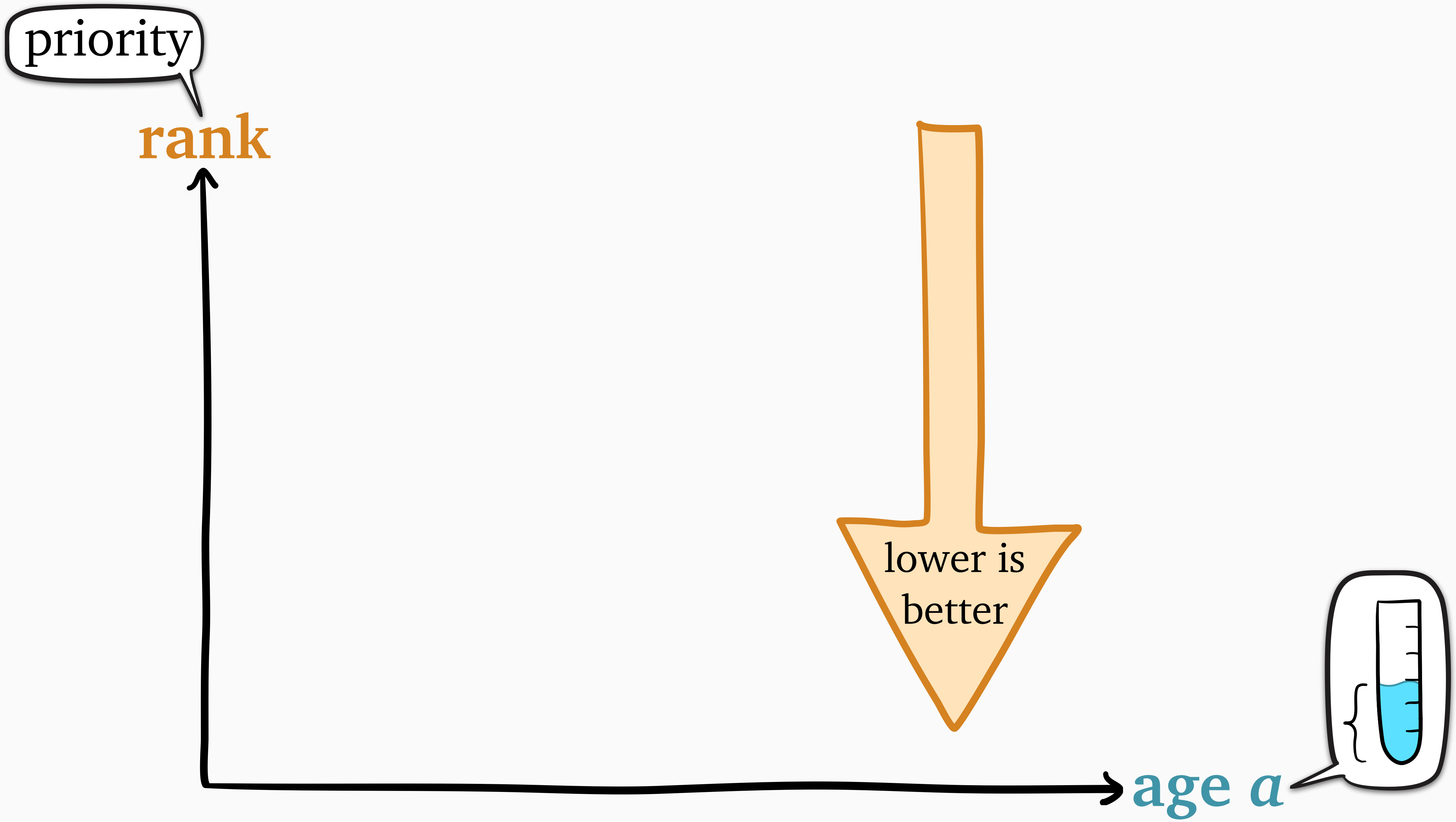
Scheduling with **rank** functions



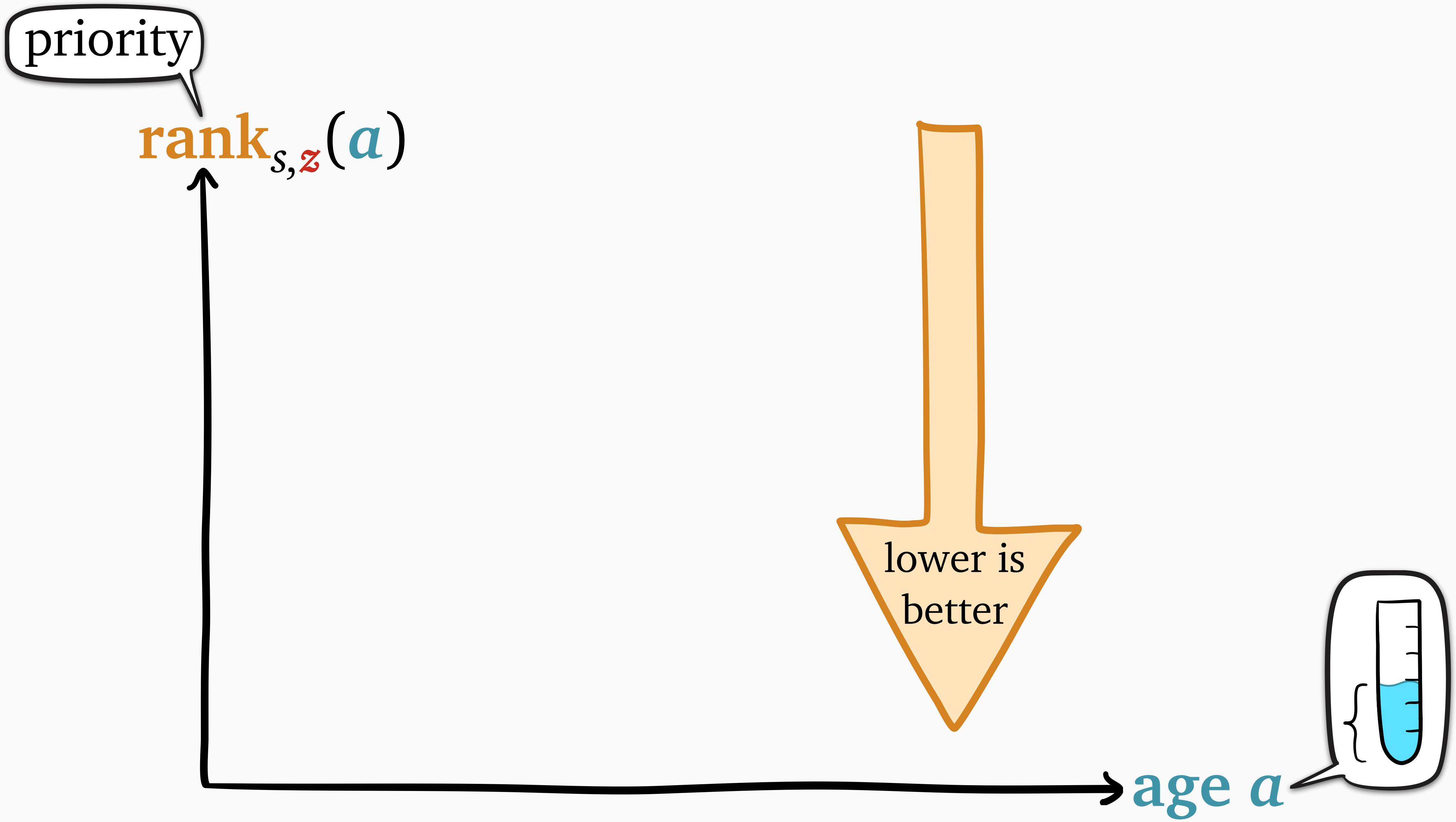
Scheduling with **rank** functions



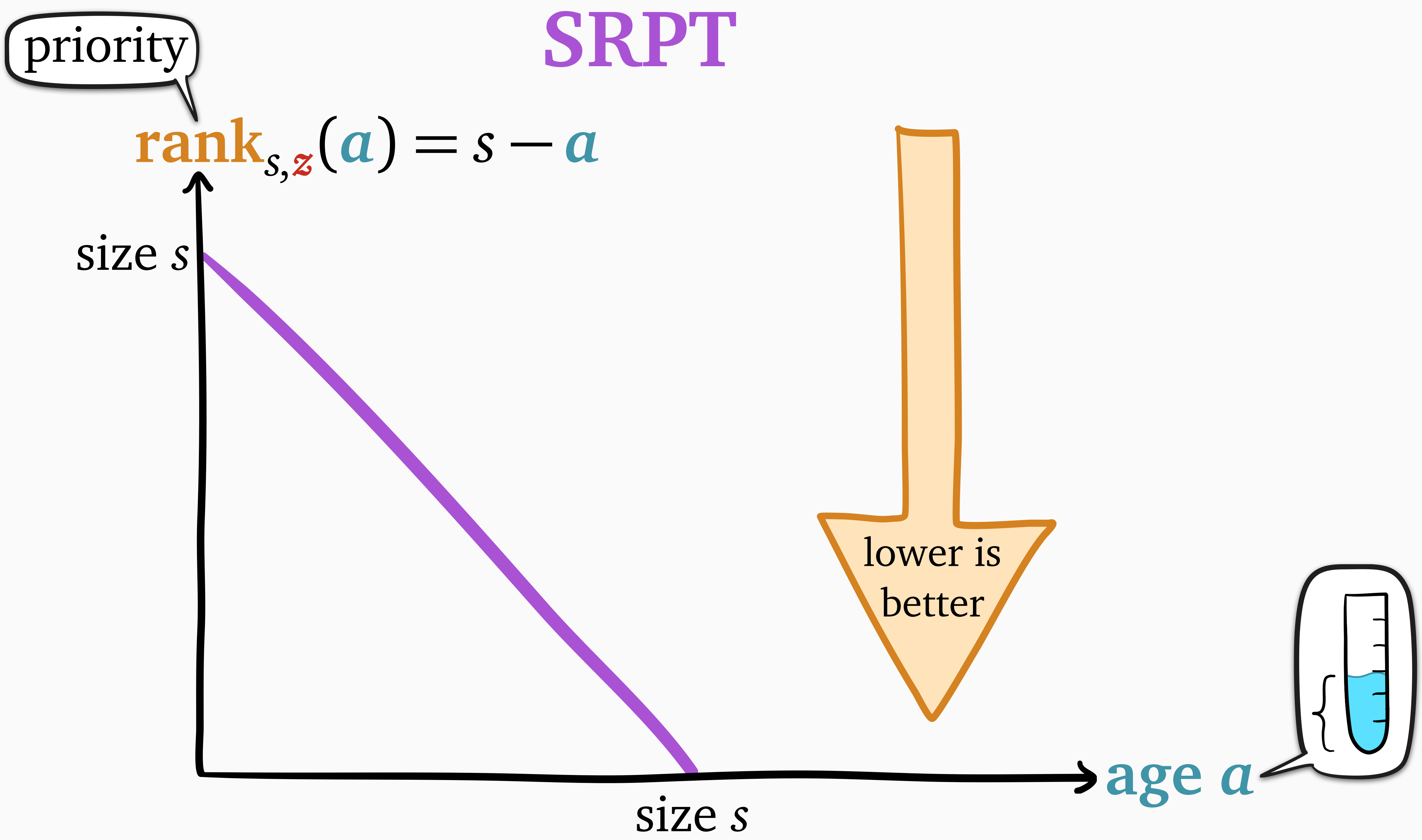
Scheduling with **rank** functions



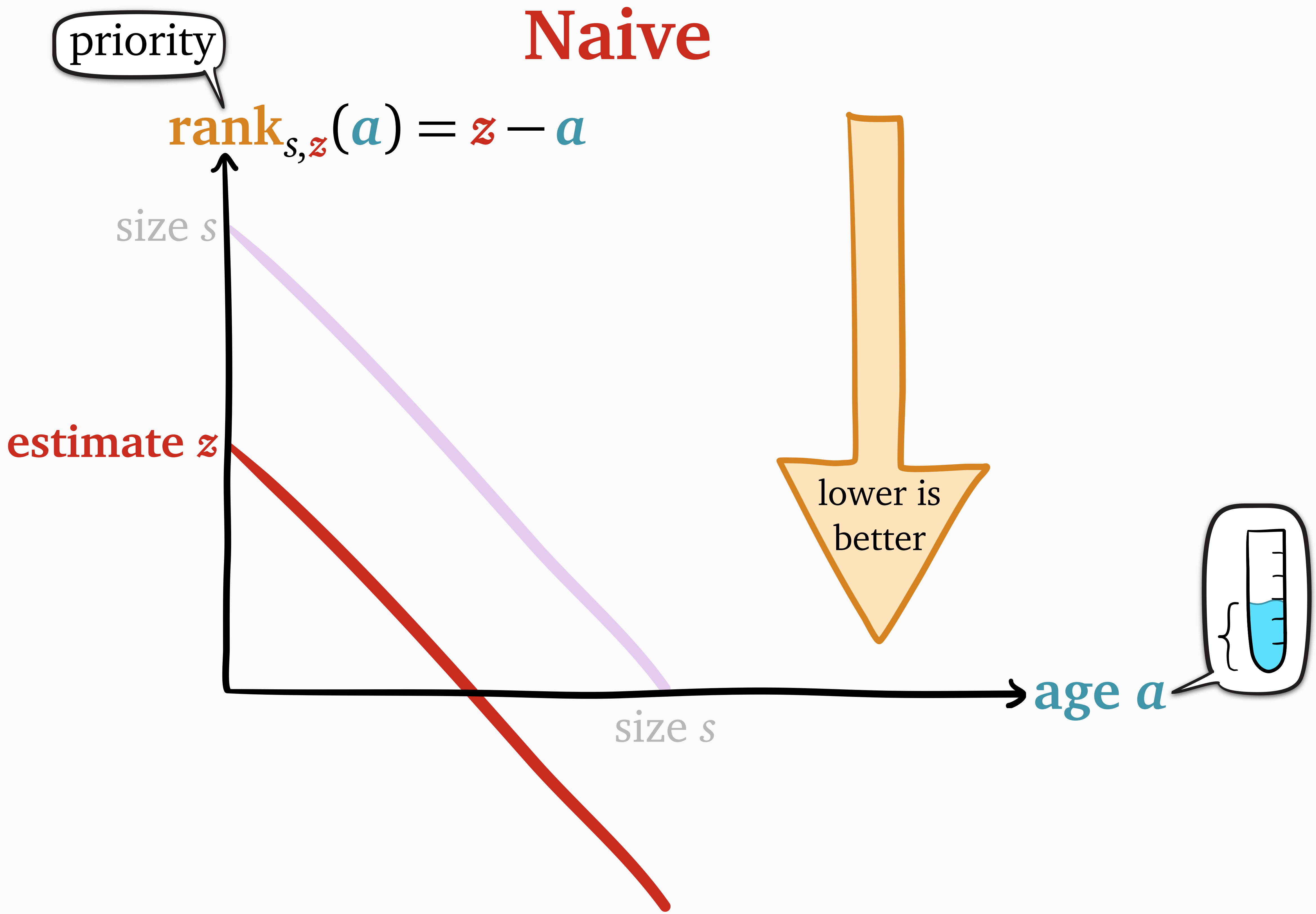
Scheduling with **rank** functions

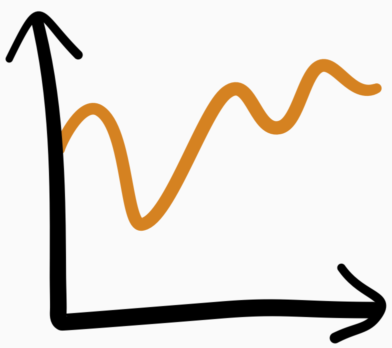


Scheduling with **rank** functions



Scheduling with **rank** functions

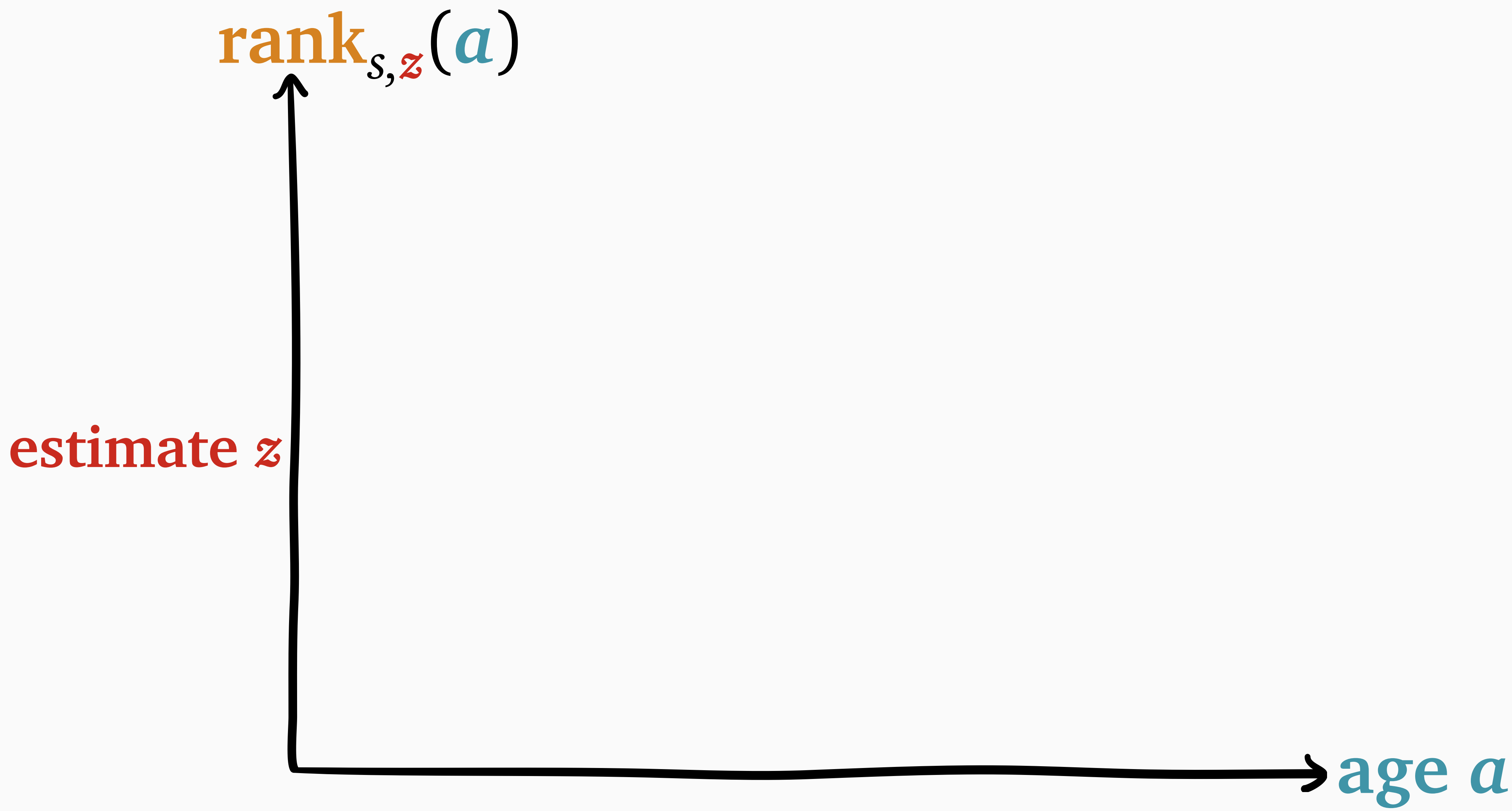




Policy design space:

rank functions

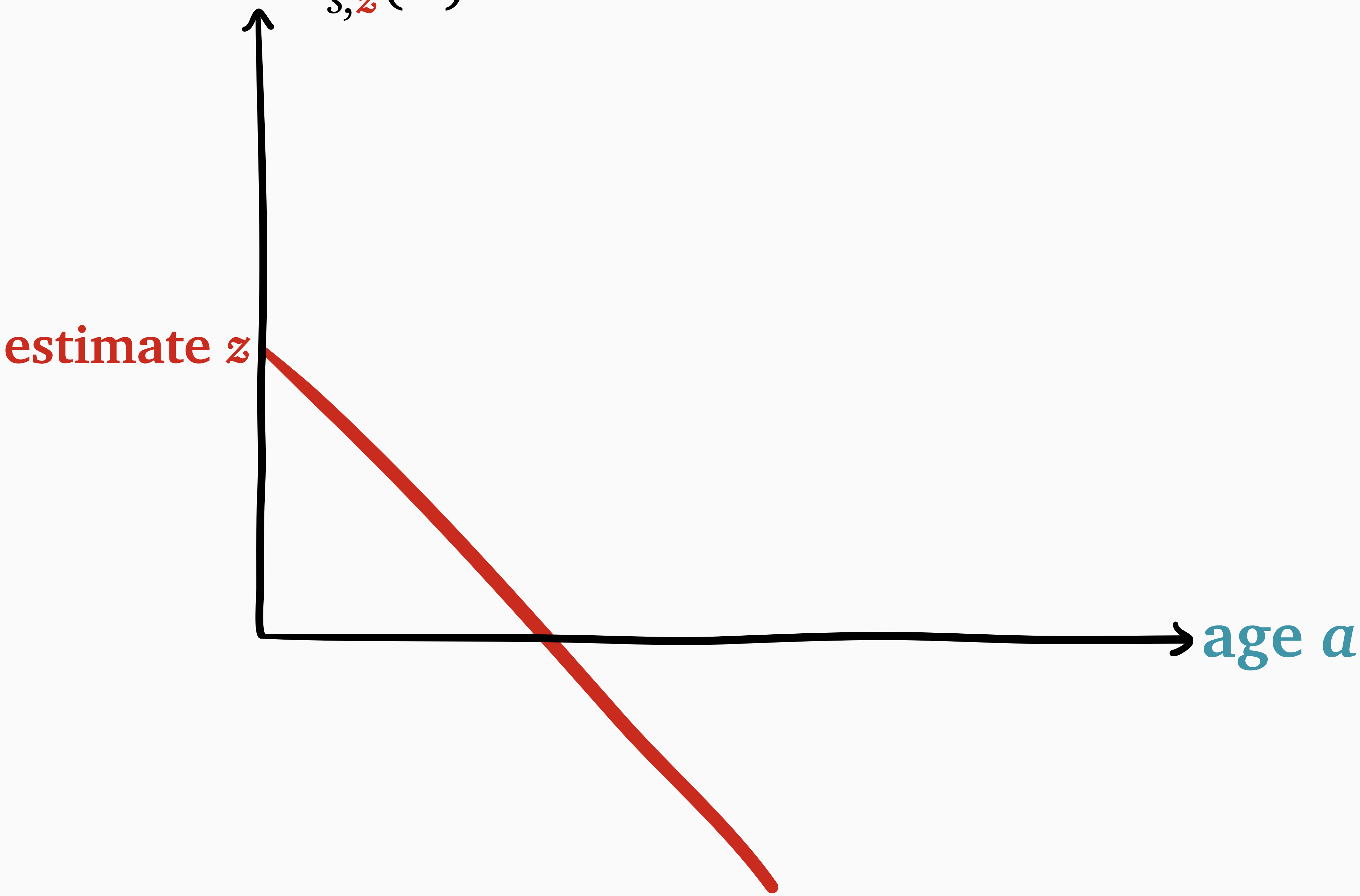
What's the right **rank** function?



What's the right rank function?

Naive

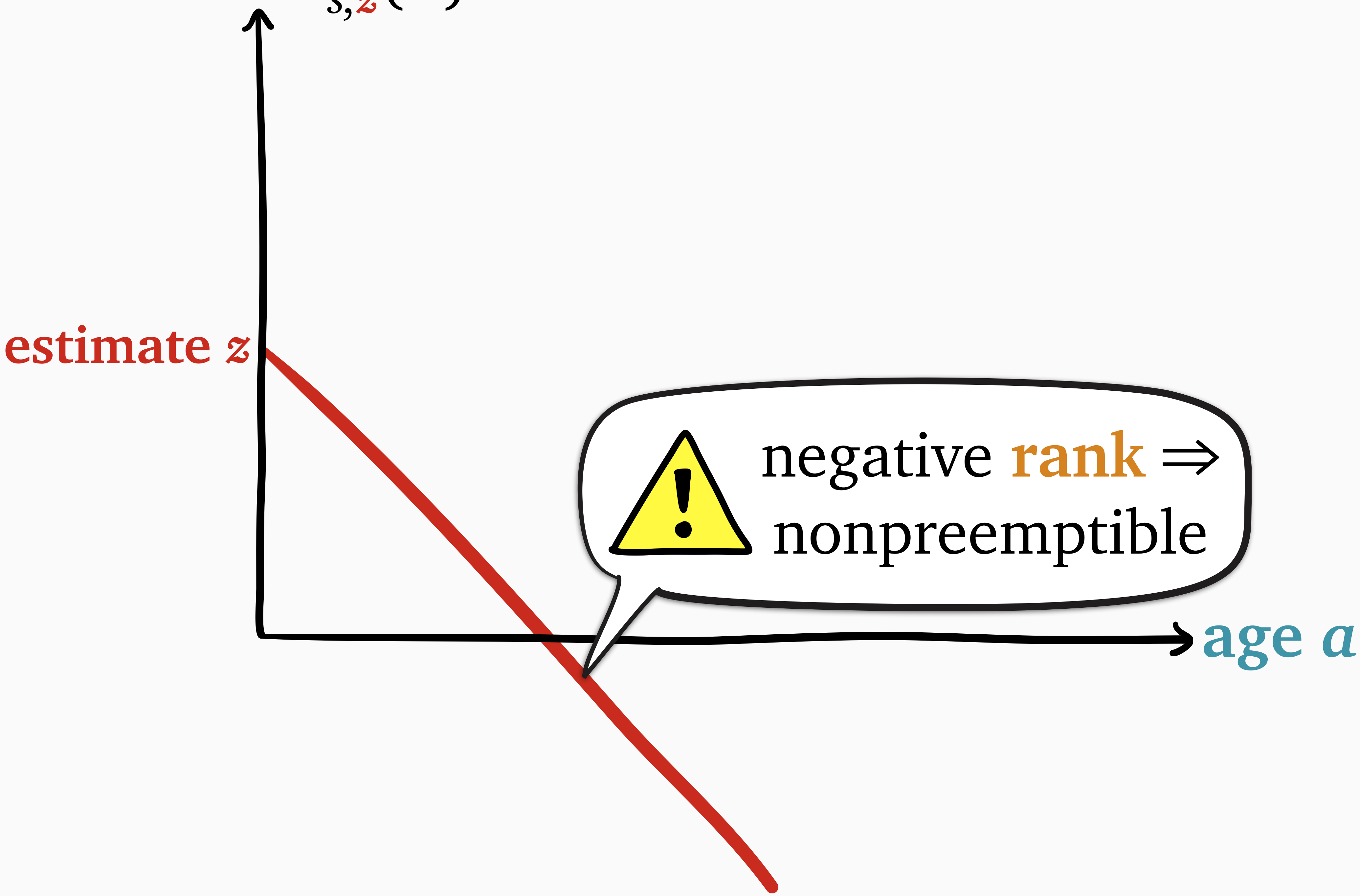
$\text{rank}_{s,z}(a) = z - a$



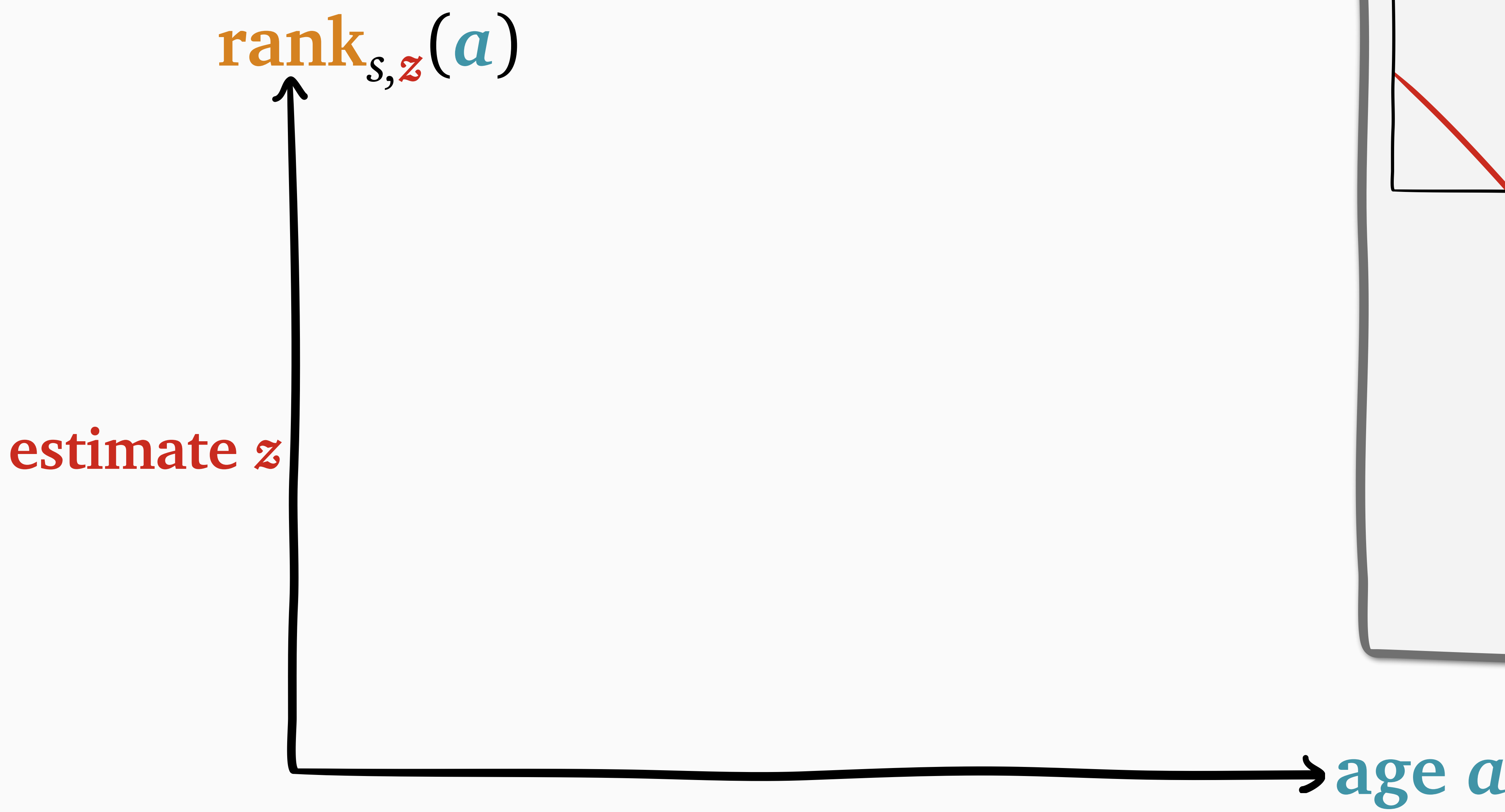
What's the right rank function?

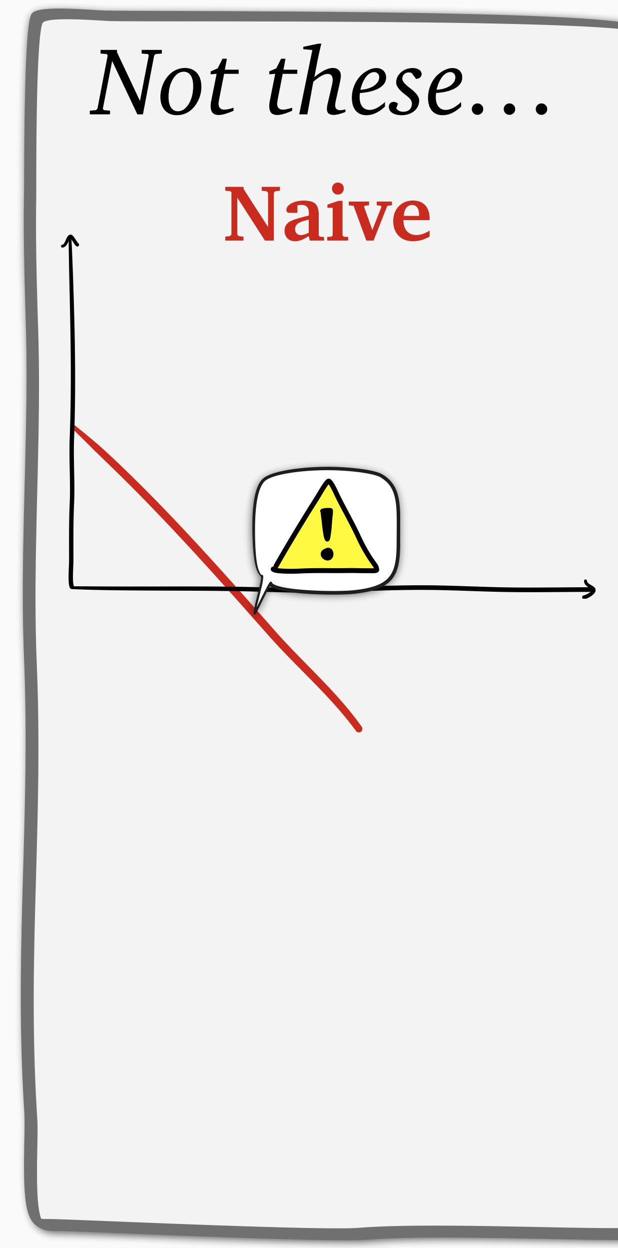
Naive

$\text{rank}_{s,z}(a) = z - a$



What's the right rank function?

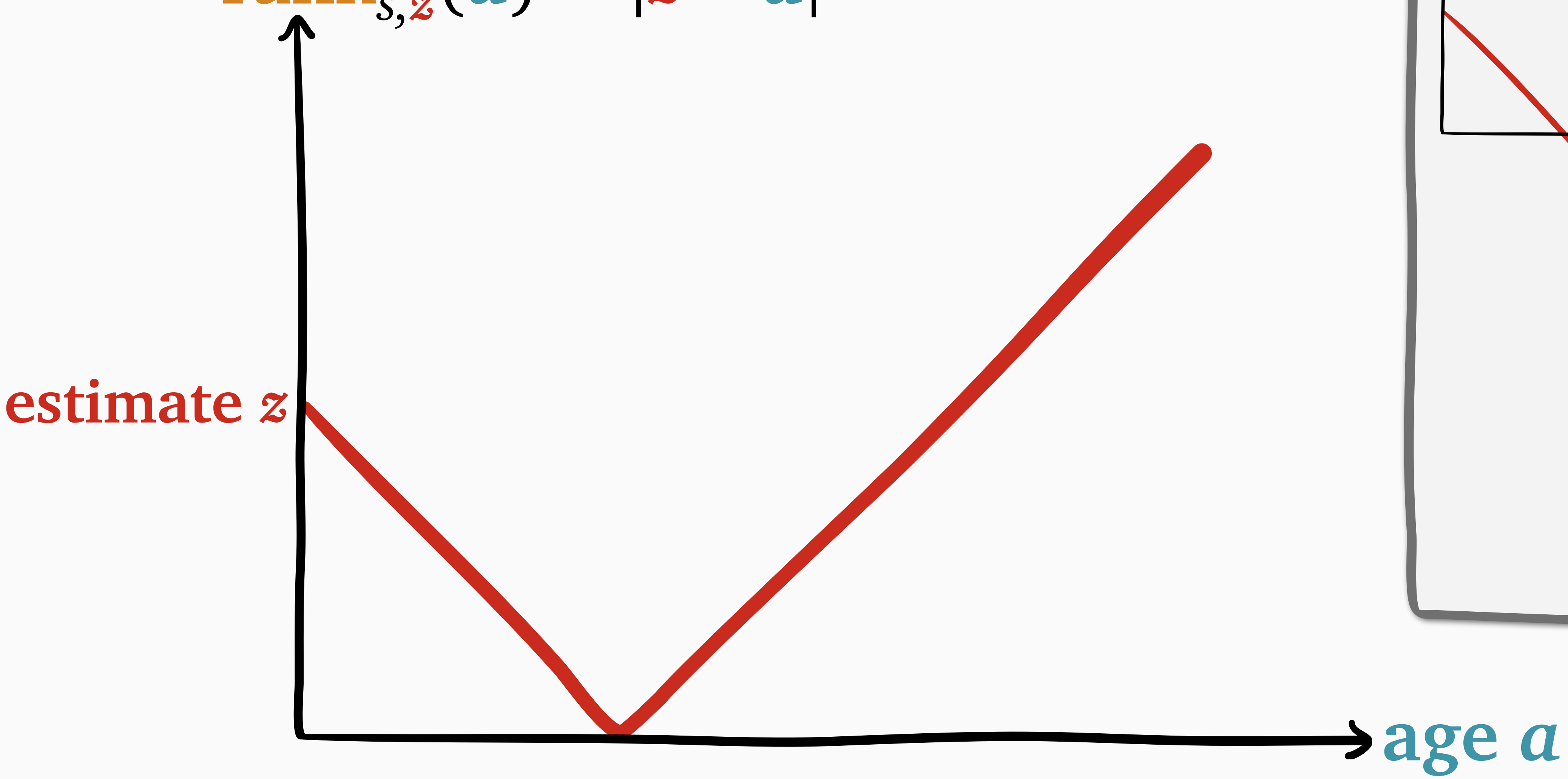




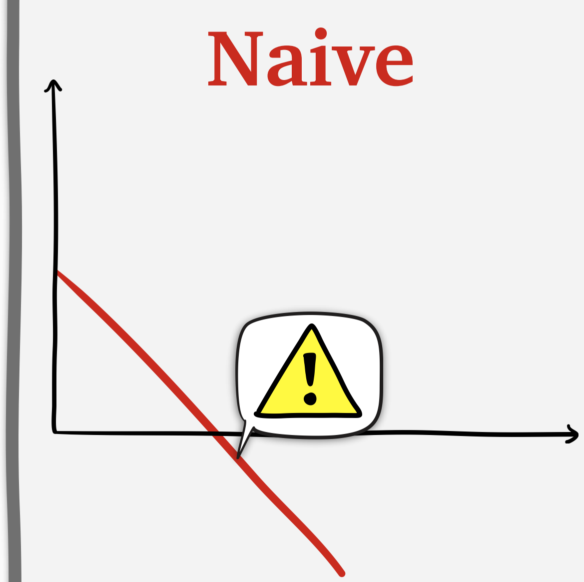
What's the right rank function?

Checkmark

$\text{rank}_{s,z}(a) = |z - a|$



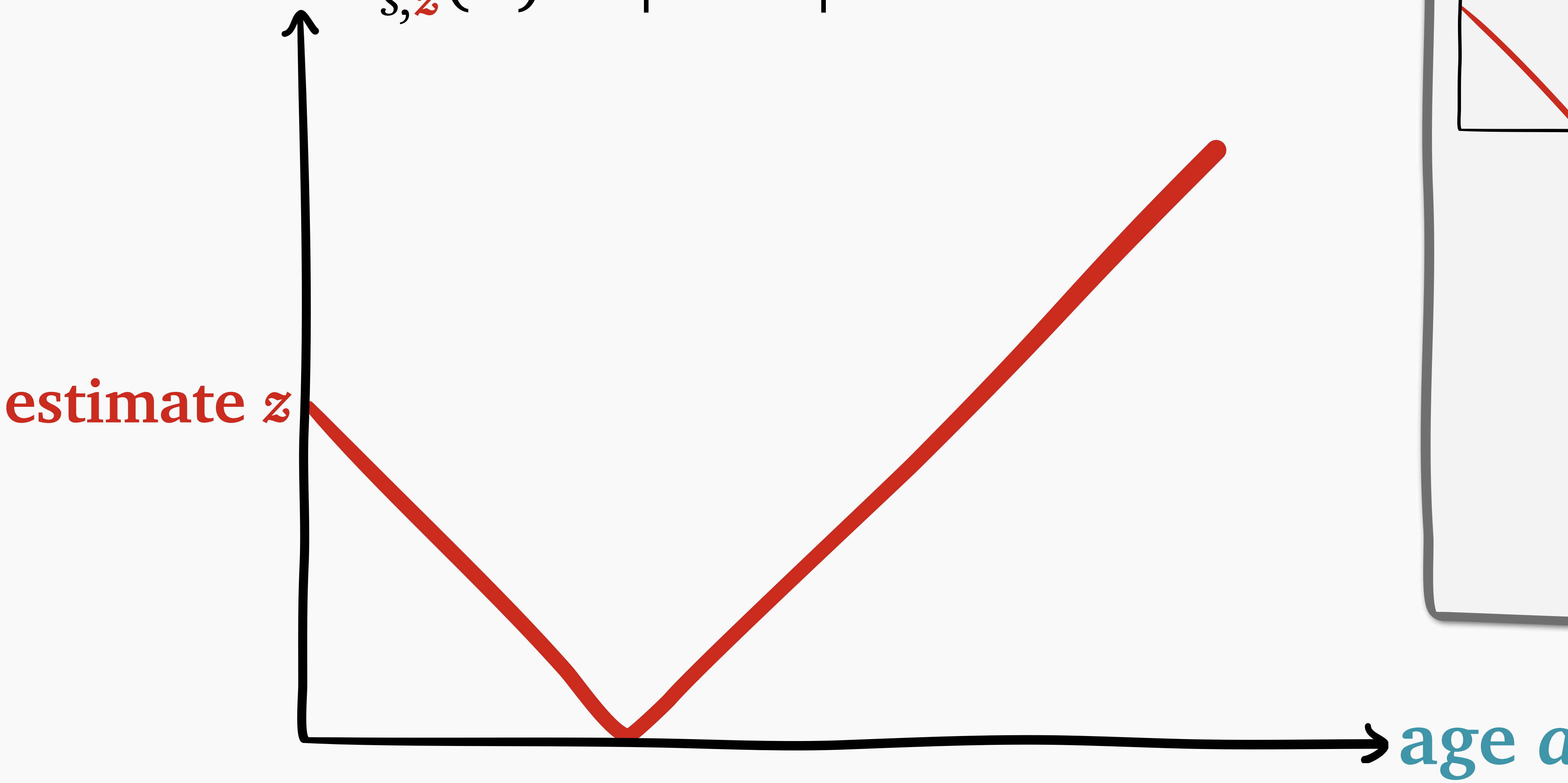
Not these…

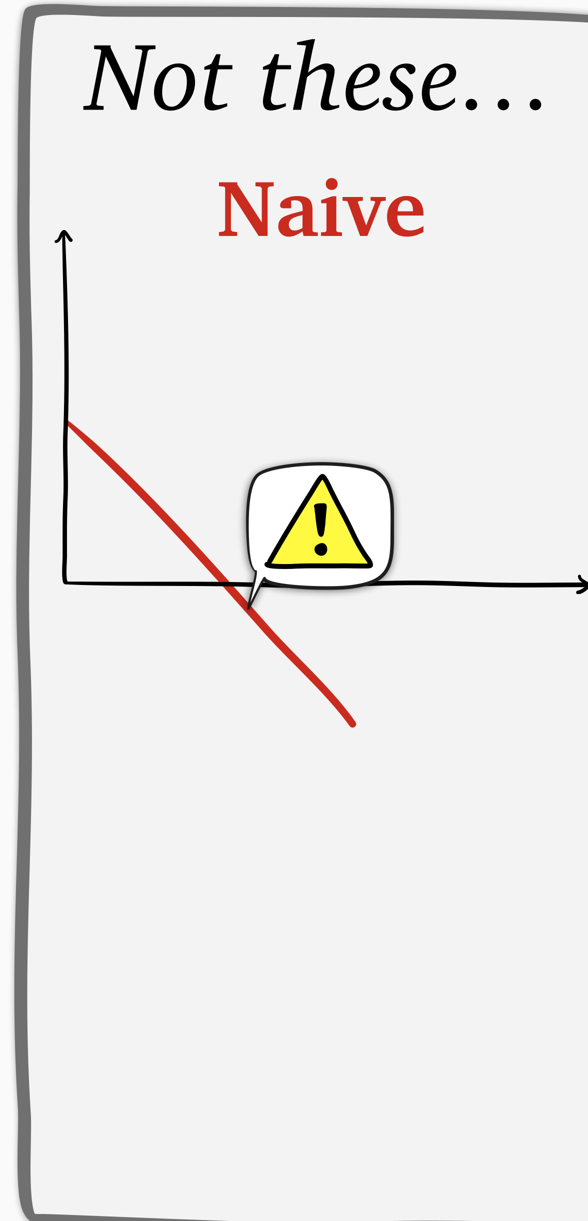


What's the right **rank** function?

Checkmark

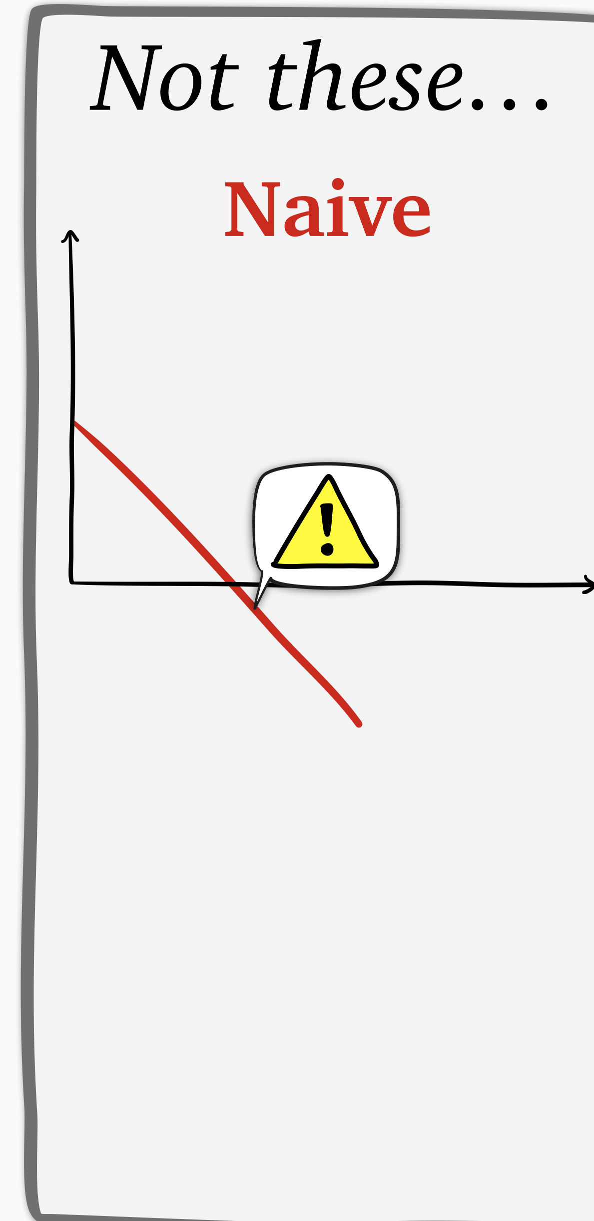
$$\text{rank}_{s,z}(a) = |z - a|$$





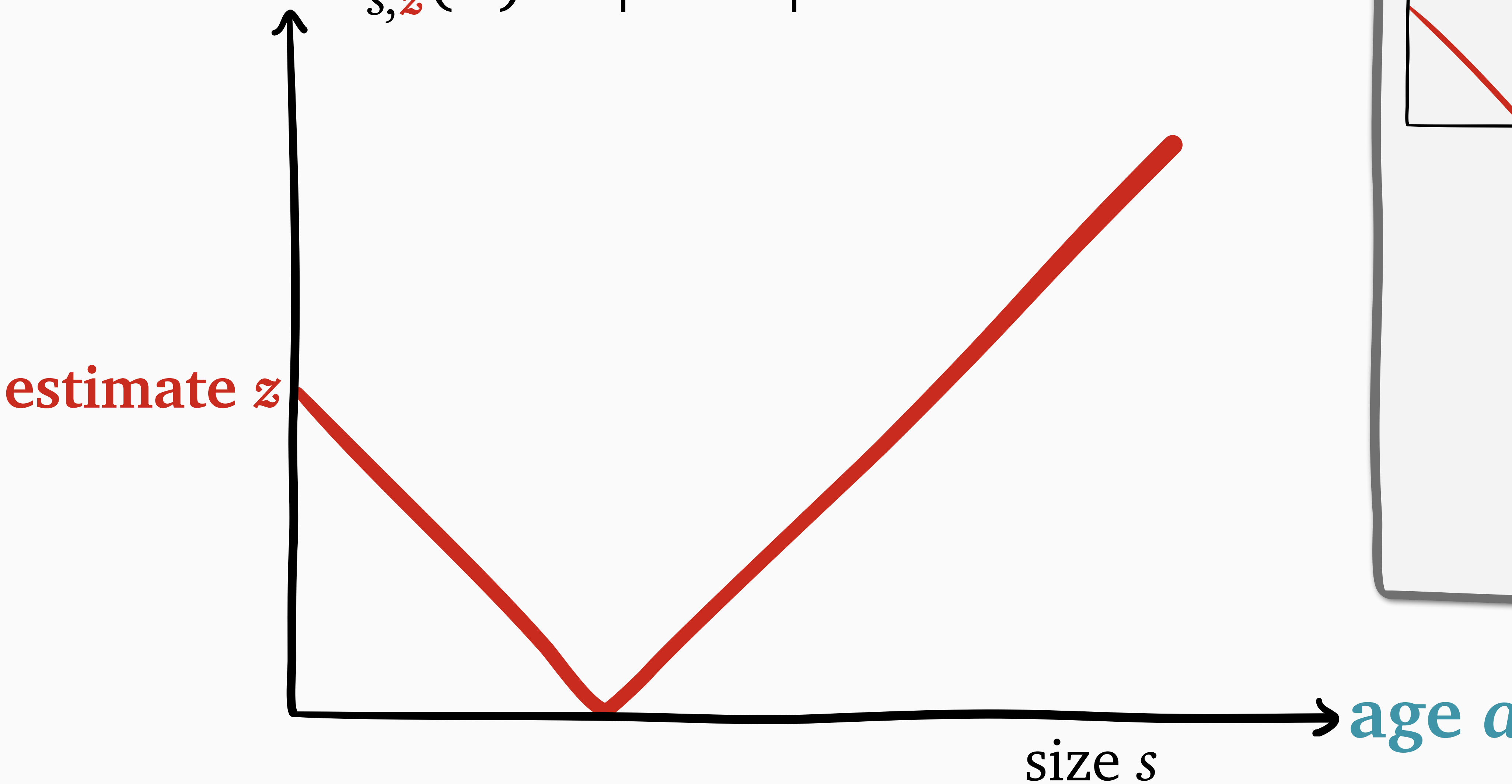
What if $\beta < \frac{1}{2}$?

What's the right **rank** function?



Checkmark

$$\text{rank}_{s,z}(a) = |z - a|$$

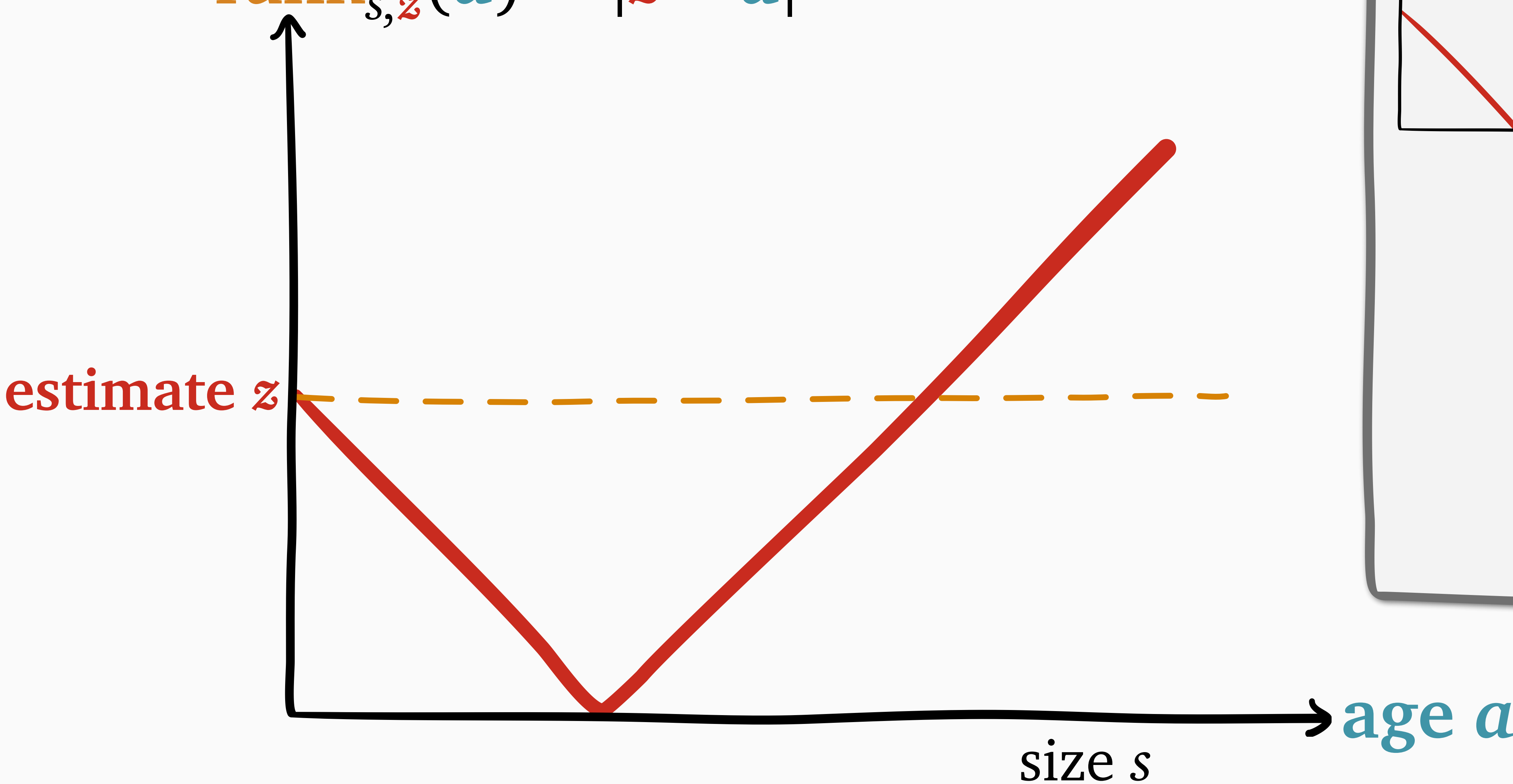


What if $\beta < \frac{1}{2}$?

What's the right rank function?

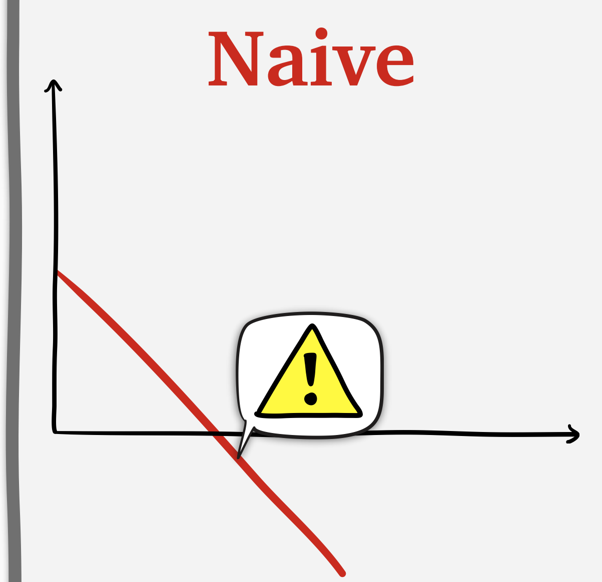
Checkmark

$\text{rank}_{s,z}(a) = |z - a|$



What if $\beta < \frac{1}{2}$?

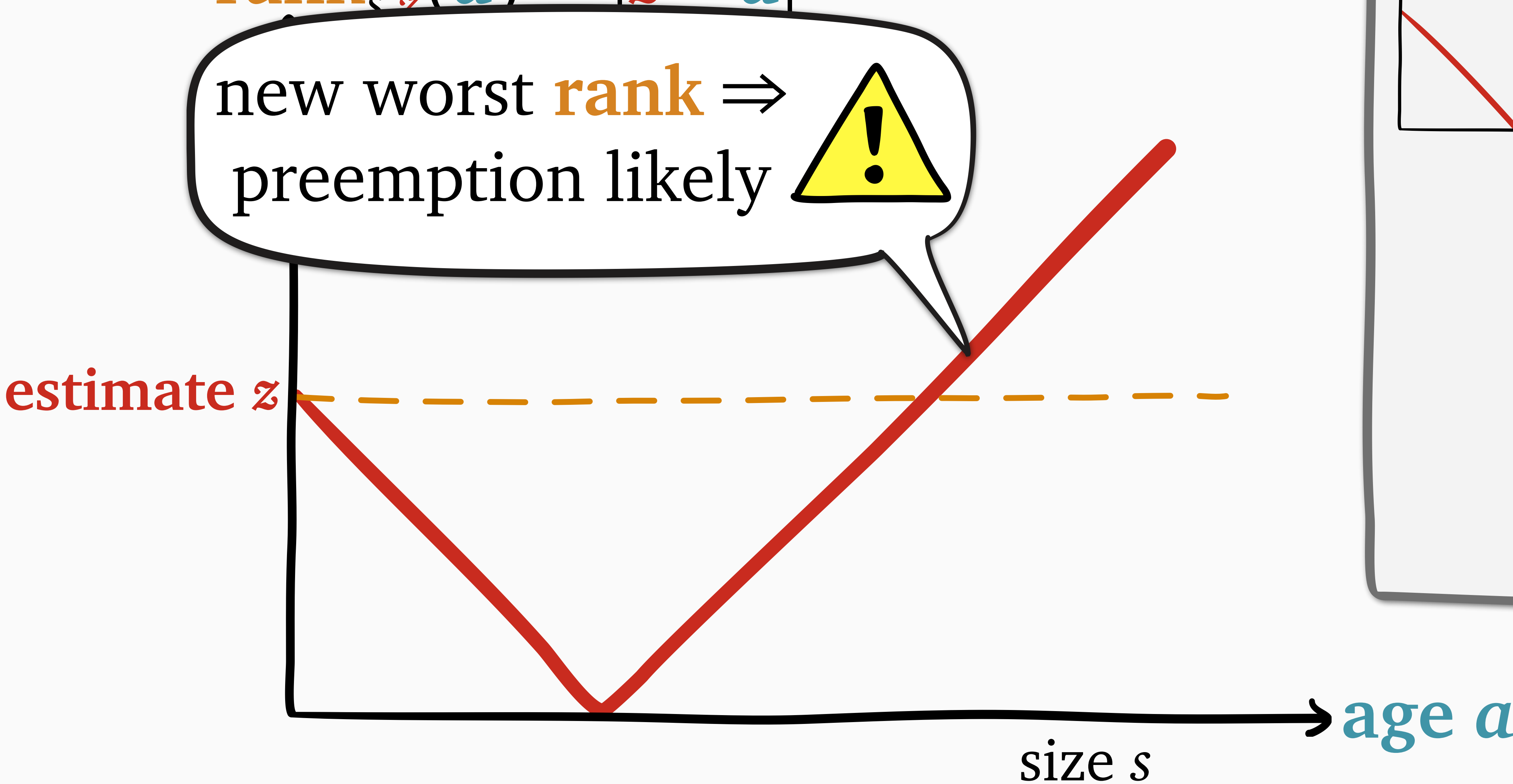
Not these…

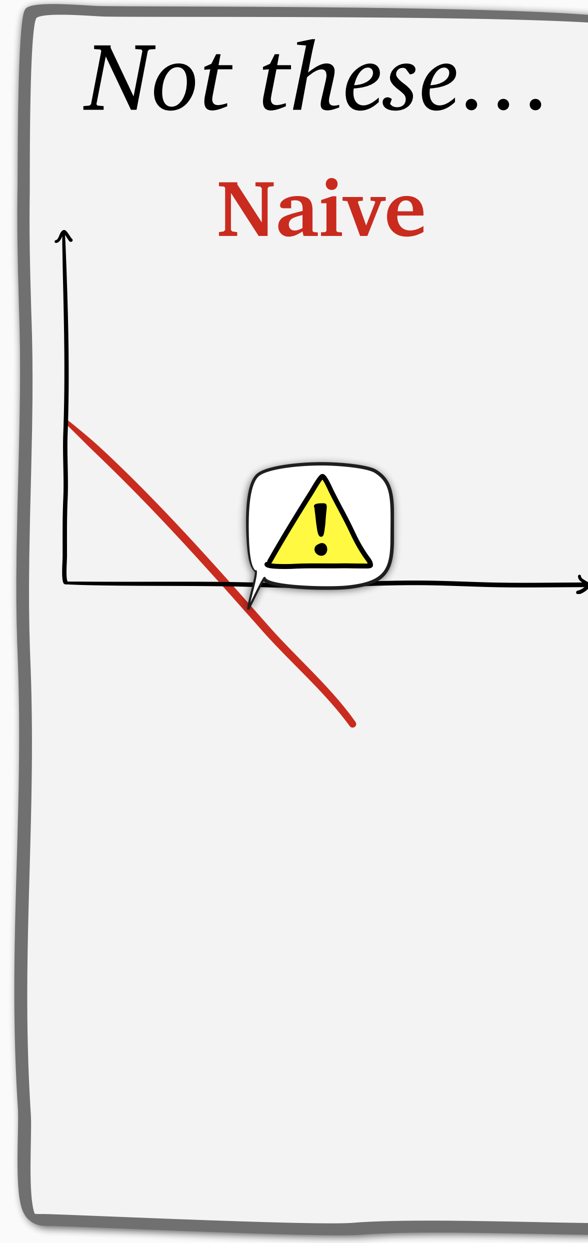


What's the right rank function?

Checkmark

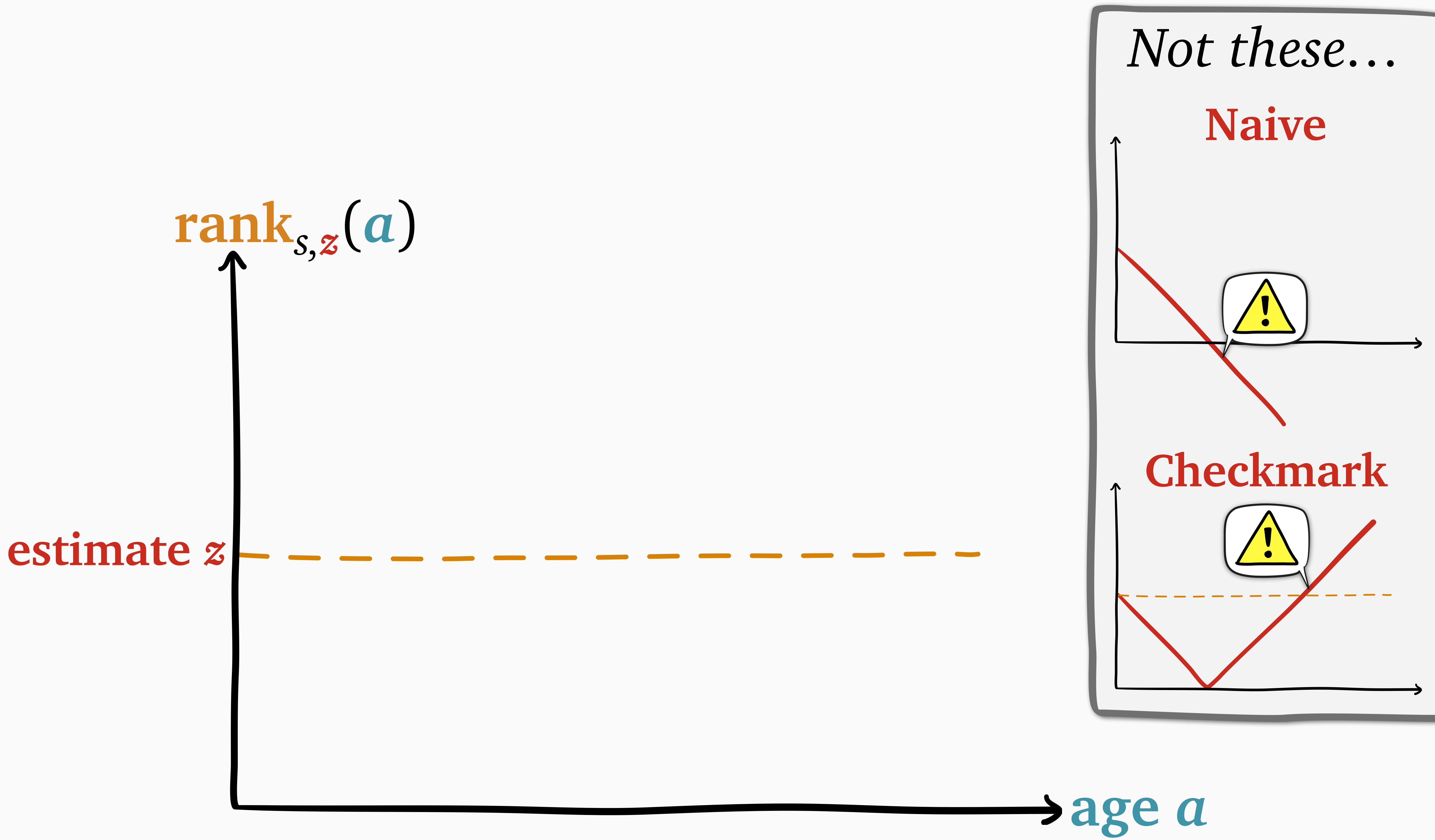
$\text{rank}_{s,z}(a) = |z - a|$





What if $\beta < \frac{1}{2}$?

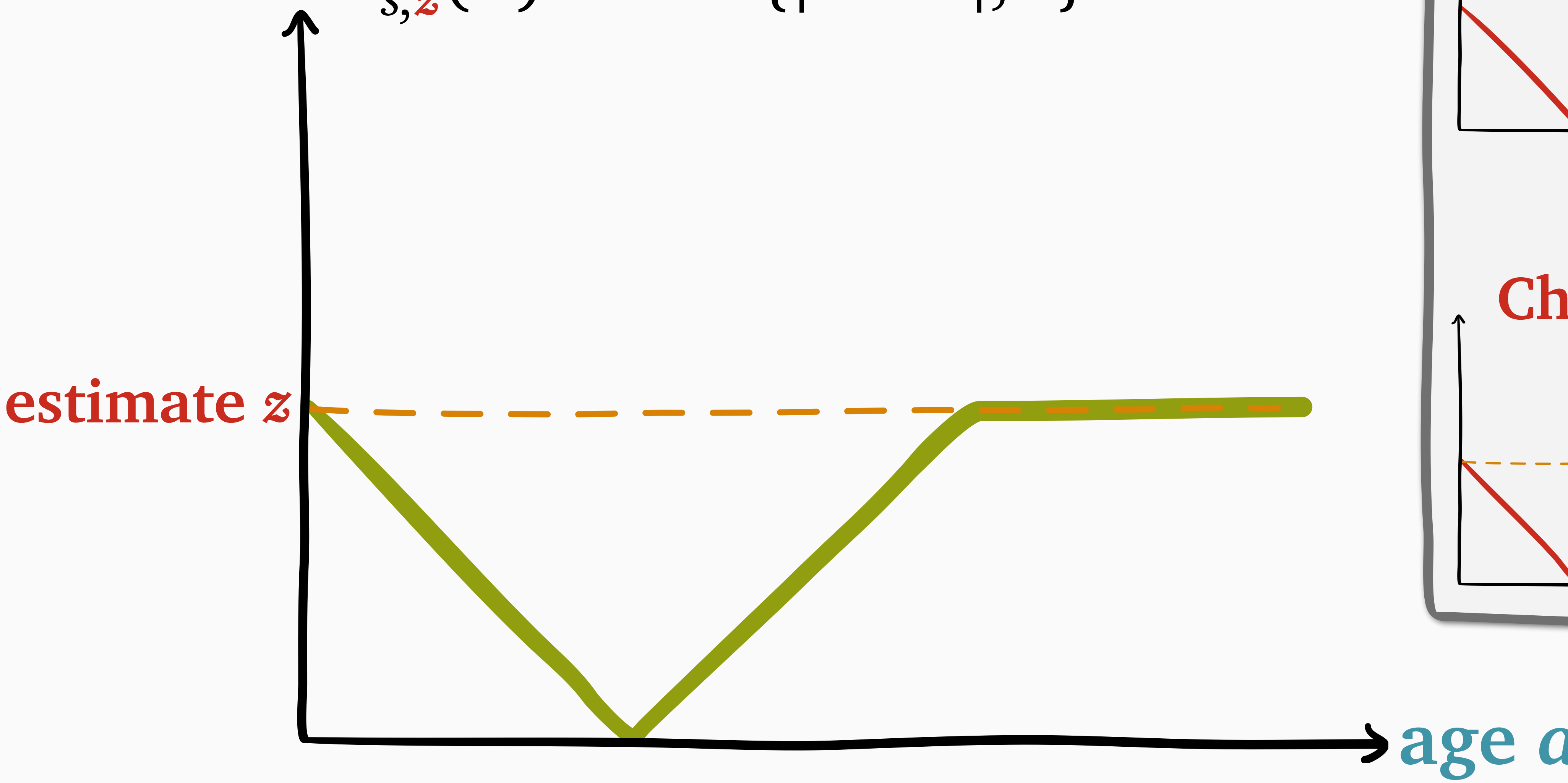
What's the right **rank** function?

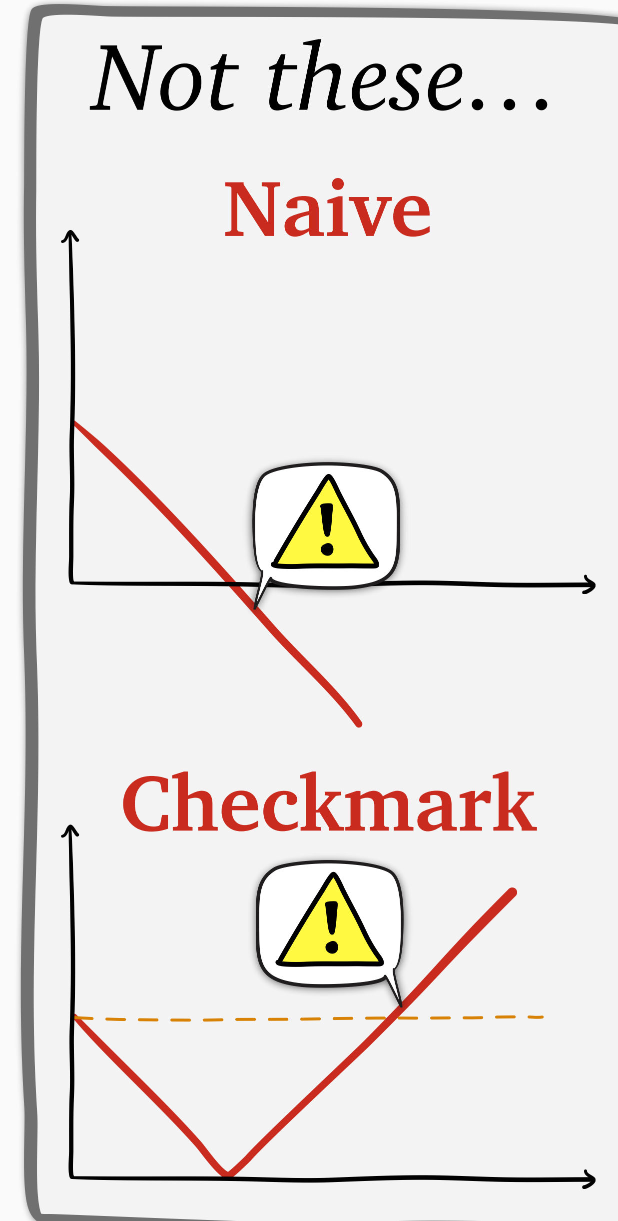


What's the right rank function?

Radical

$$\mathrm{rank}_{s,z}(a) = \min\{|z - a|, z\}$$

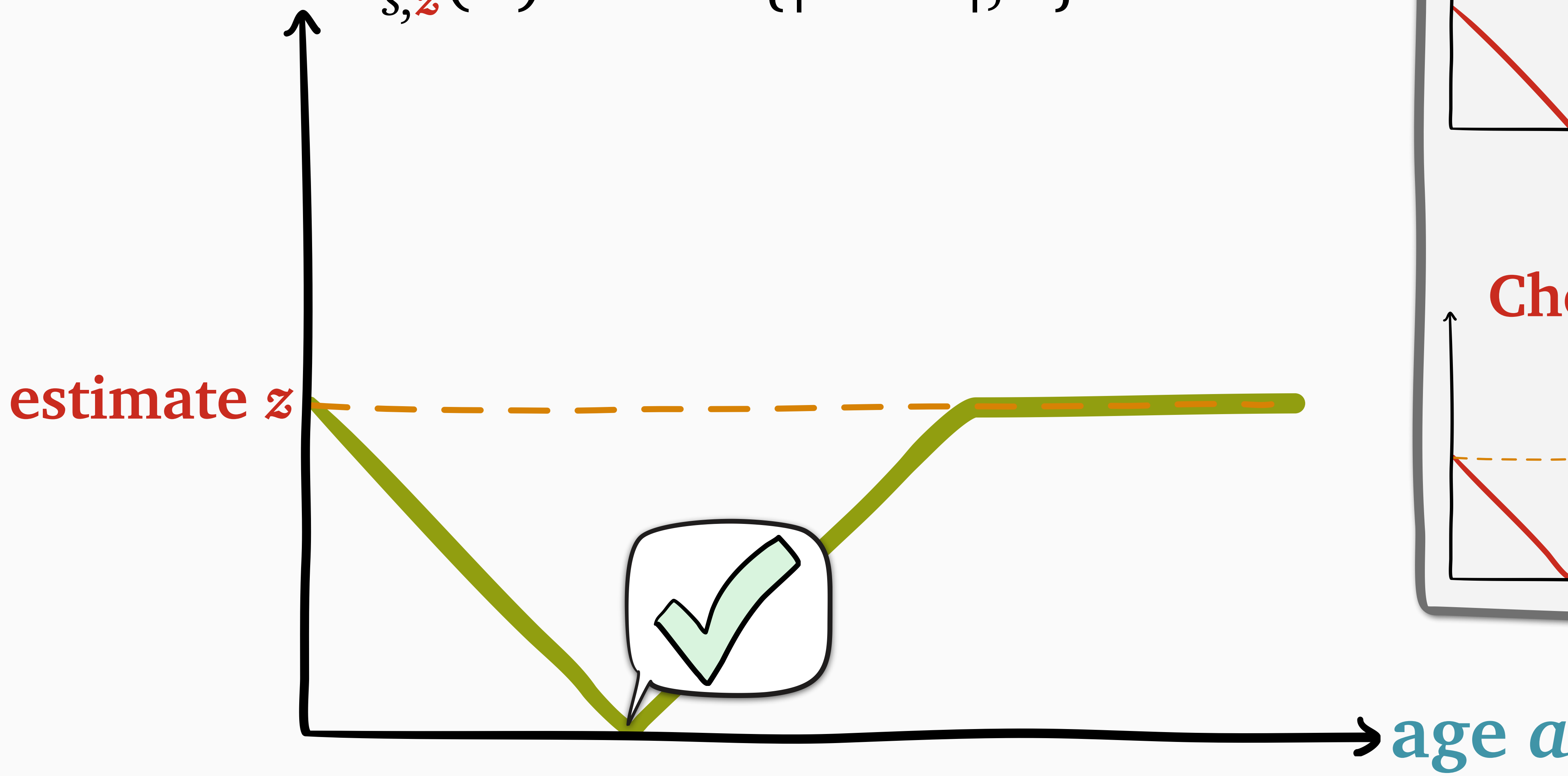


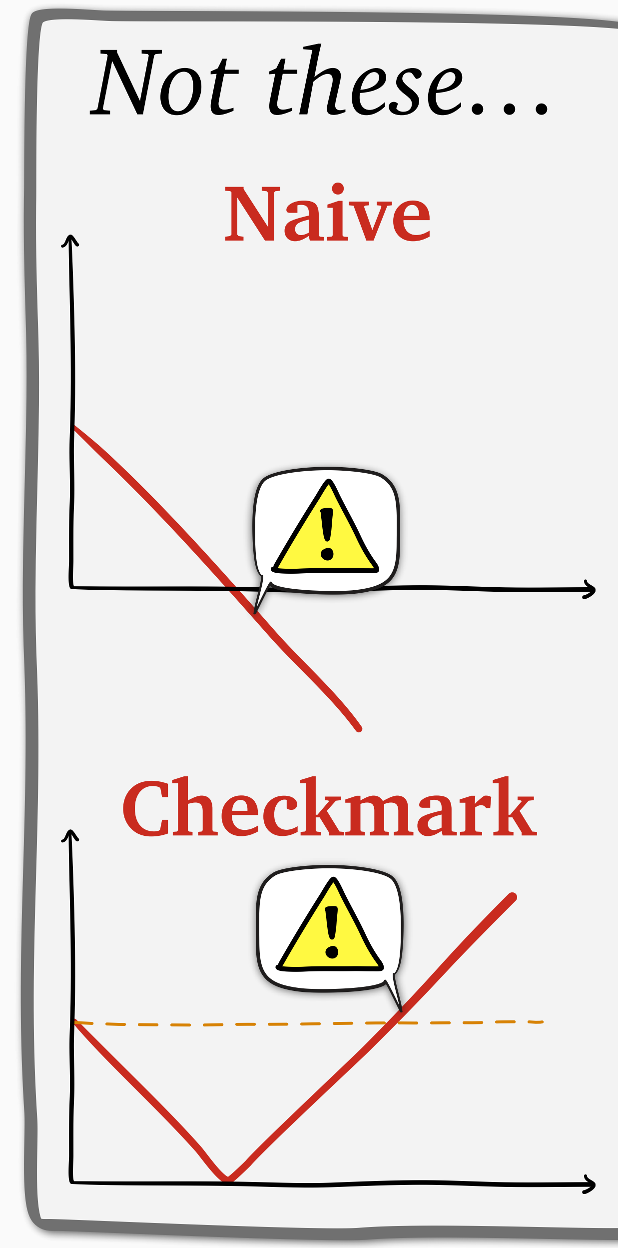


What's the right rank function?

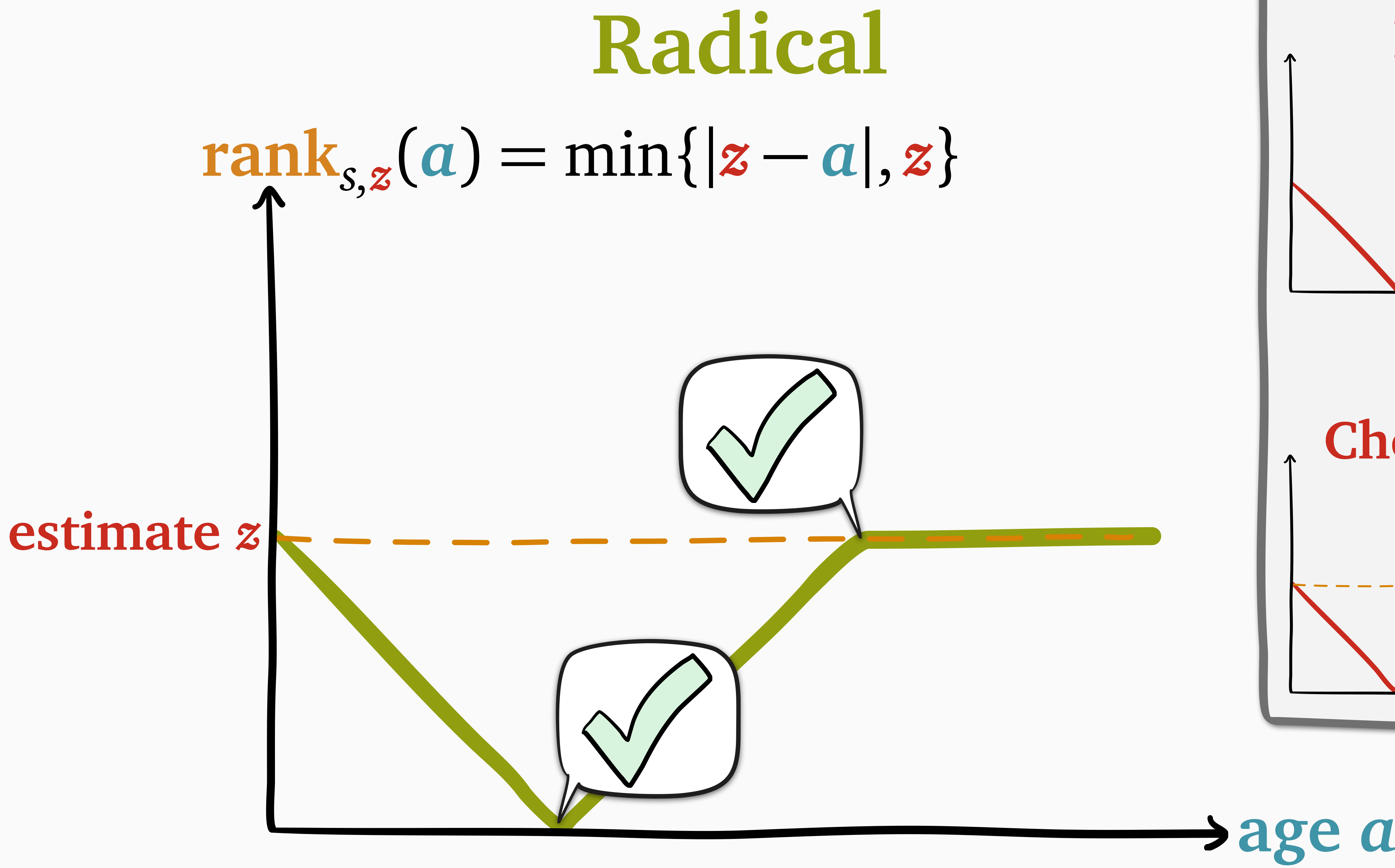
Radical

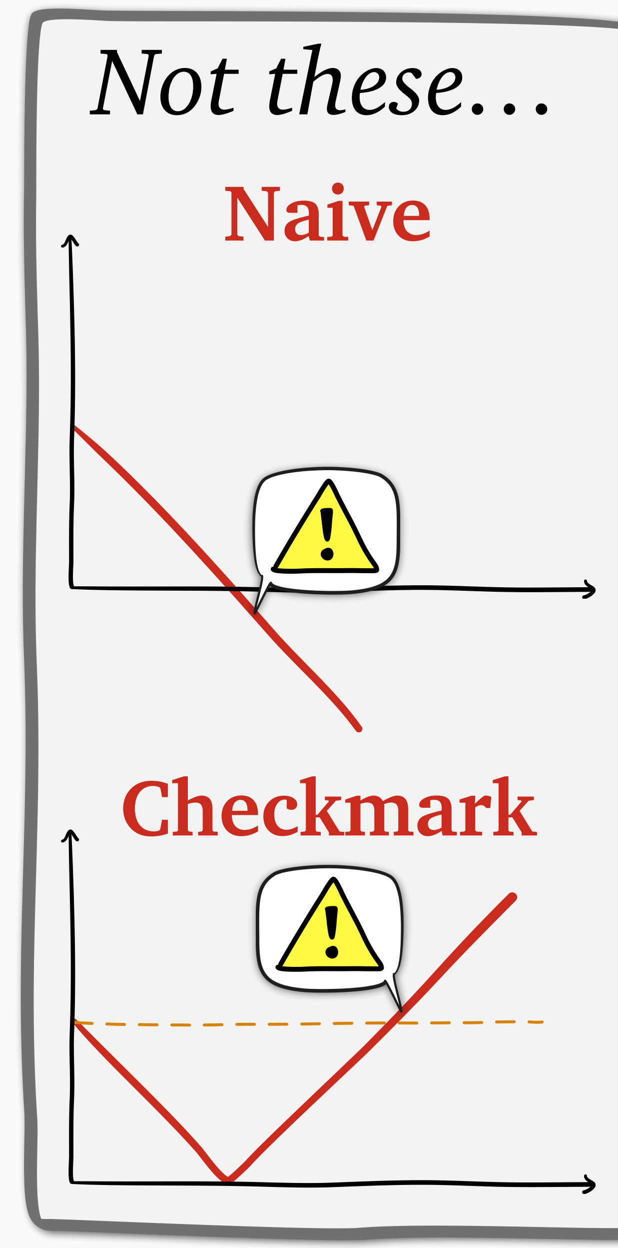
$$\text{rank}_{s,z}(a) = \min\{|z - a|, z\}$$

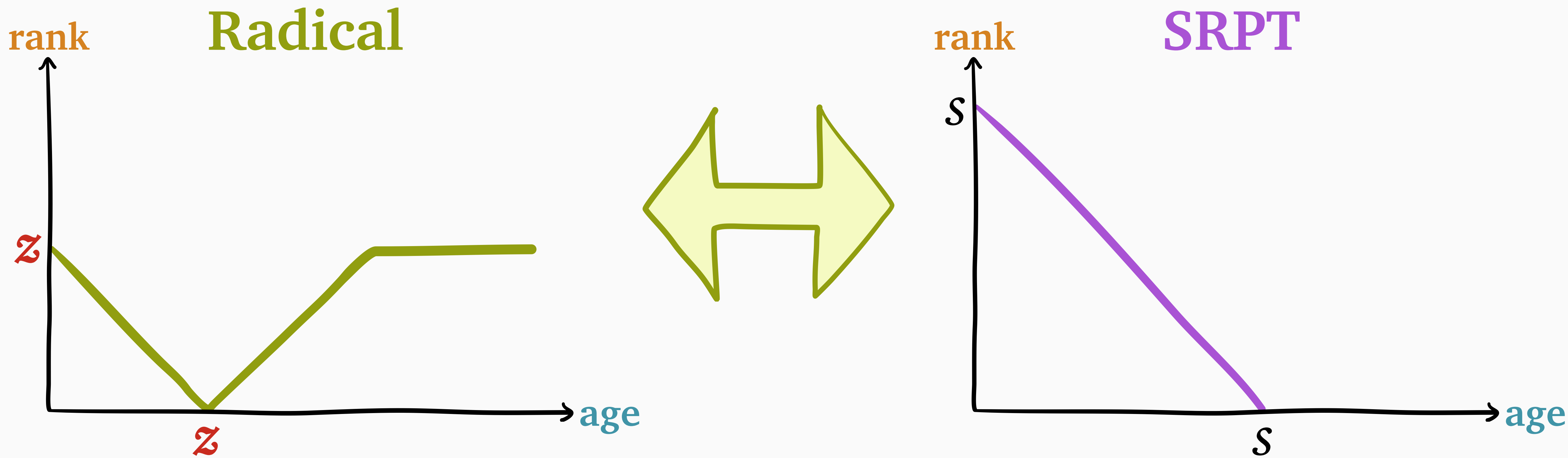




What's the right **rank** function?







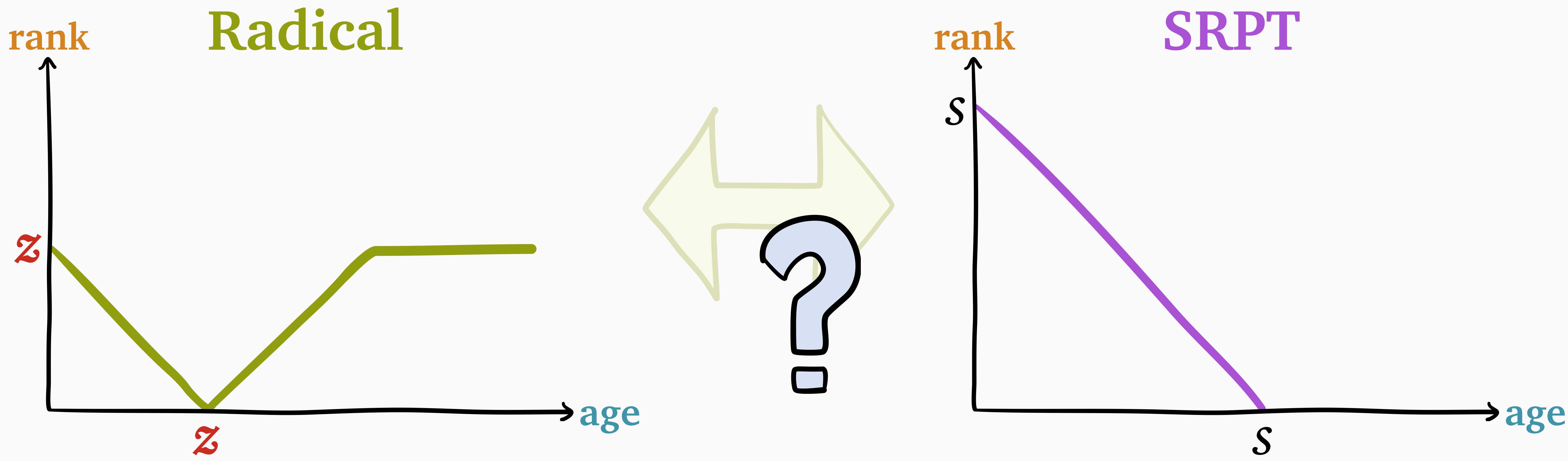
Theorem: in queueing model,

$$\frac{\mathbf{E}[T_{\text{Radical}}]}{\mathbf{E}[T_{\text{SRPT}}]} \le C_{\alpha,\beta} \cdot \gamma$$

where

$$C_{\alpha,\beta} \le 3.5$$

$$C_{\alpha,\beta} \to 1 \quad \text{as } \alpha, \beta \to 1$$



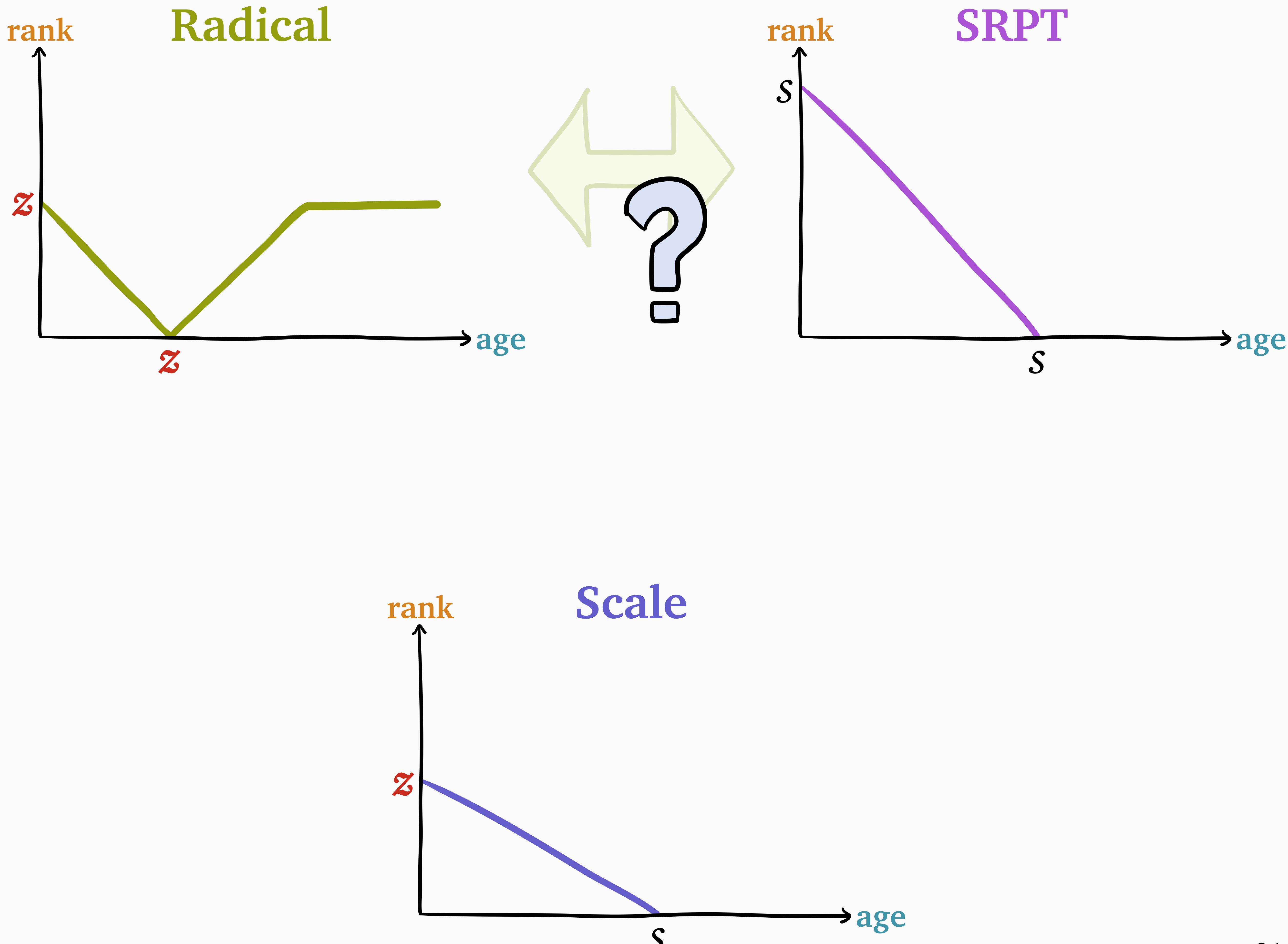
Theorem: in queueing model,

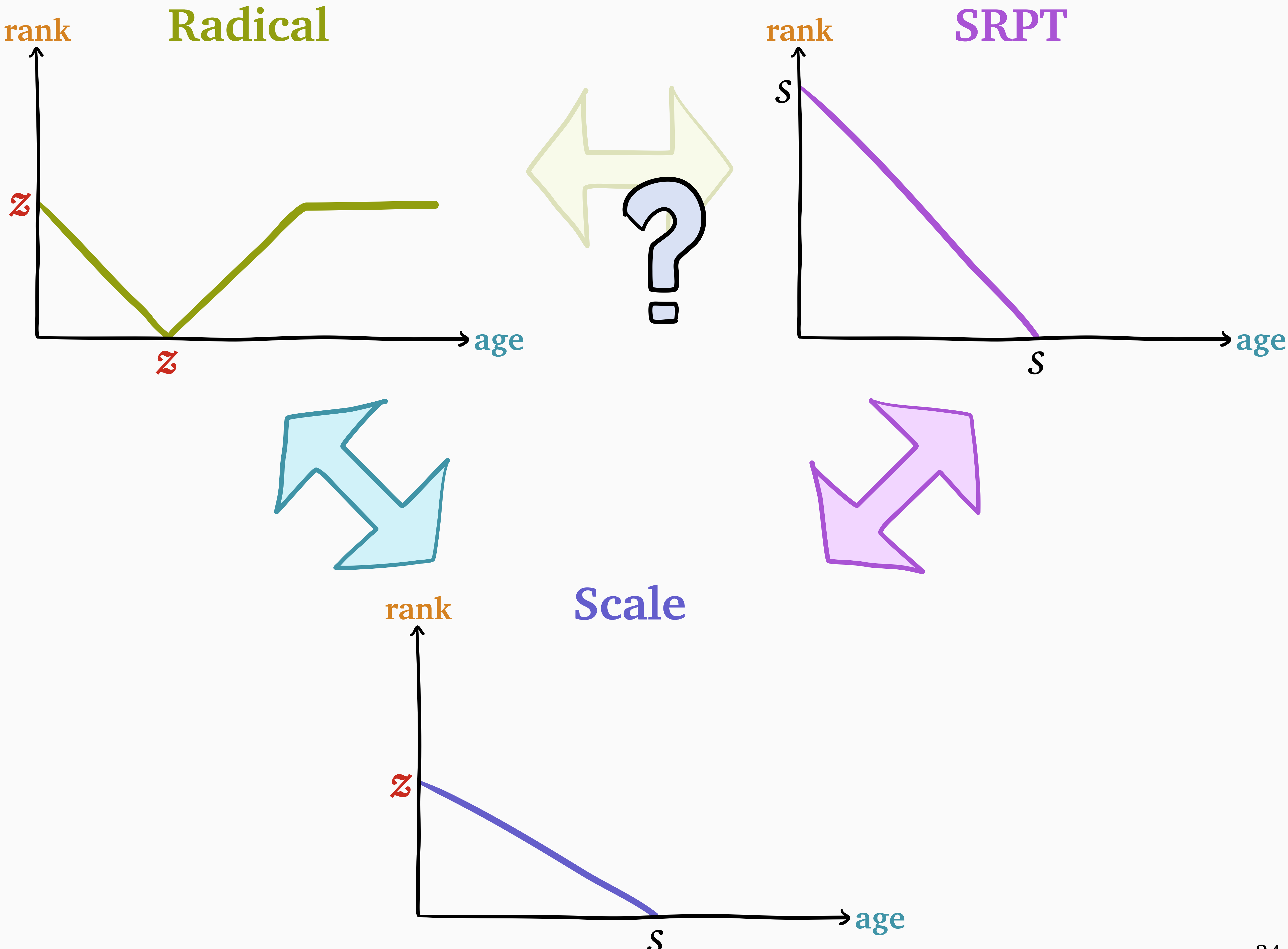
$$\frac{\mathbf{E}[T_{\text{Radical}}]}{\mathbf{E}[T_{\text{SRPT}}]} \leq C_{\alpha,\beta} \cdot \gamma$$

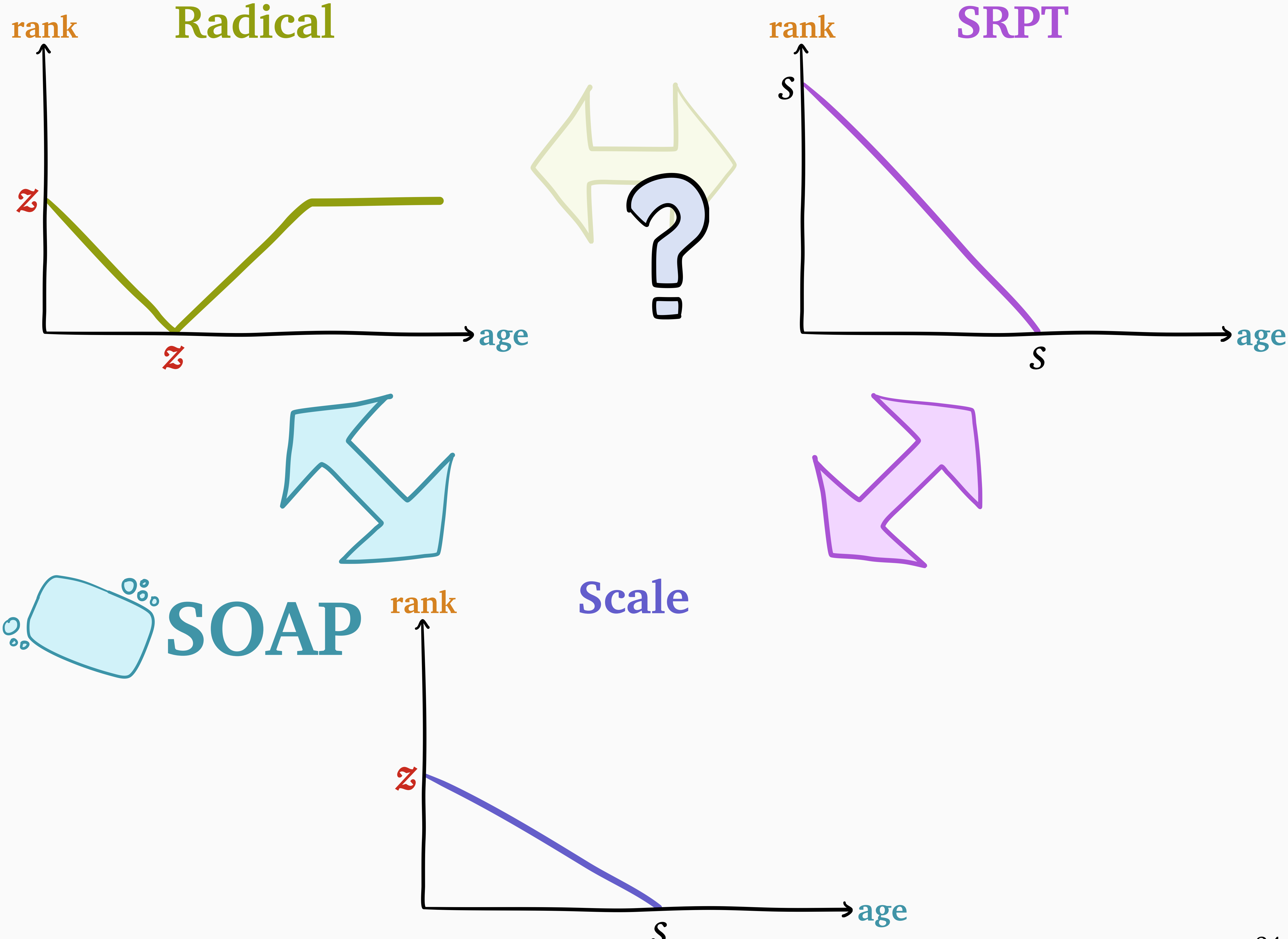
where

$$C_{\alpha,\beta} \leq 3.5$$

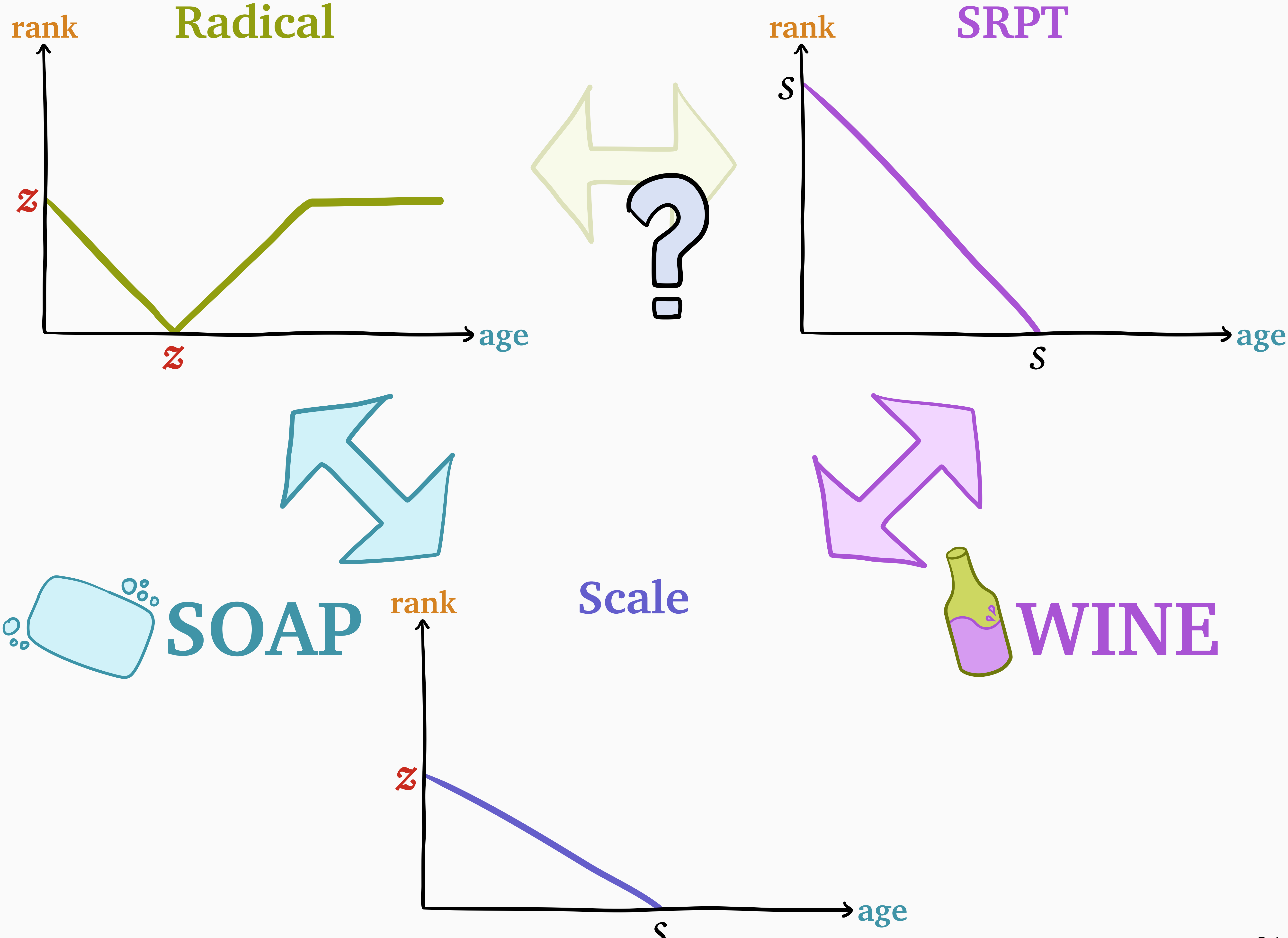
$$C_{\alpha,\beta} \to 1 \quad \text{as } \alpha, \beta \to 1$$


Radical
rank
z
z
age
SRPT
rank
s
s
age
Scale
rank
z
s
age


Radical
rank
z
z
age
SRPT
rank
s
s
age
Scale
rank
z
s
age



Radical
rank
z
z
age
SRPT
rank
s
s
age
?
SOAP
Scale
rank
z
age
s


rank
Radical
z
z
age
rank
SRPT
s
s
age
SOAP
rank
Scale
z
s
age
WINE

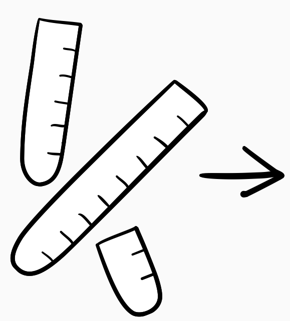
SOAP

Schedule **O**rdered by **A**ge-based **P**riority

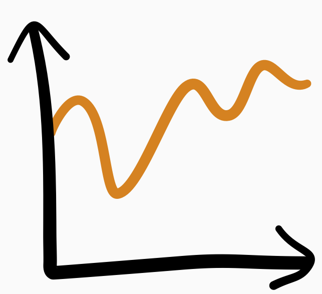
SOAP

Schedule **O**rdered by **A**ge-based **P**riority

stochastic arrival process λ, $(S, \mathbf{Z})$



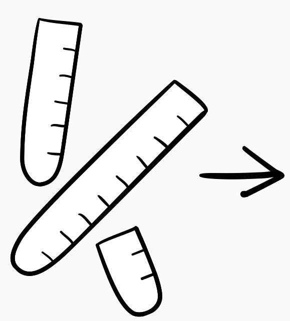
any **rank** function



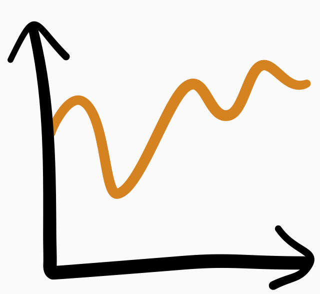
SOAP

Schedule **O**rdered by **A**ge-based **P**riority

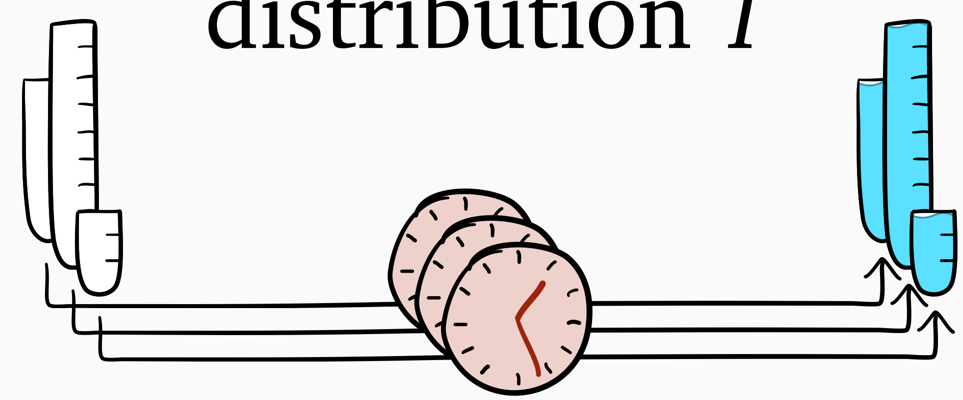
stochastic arrival process λ, (S, Z)

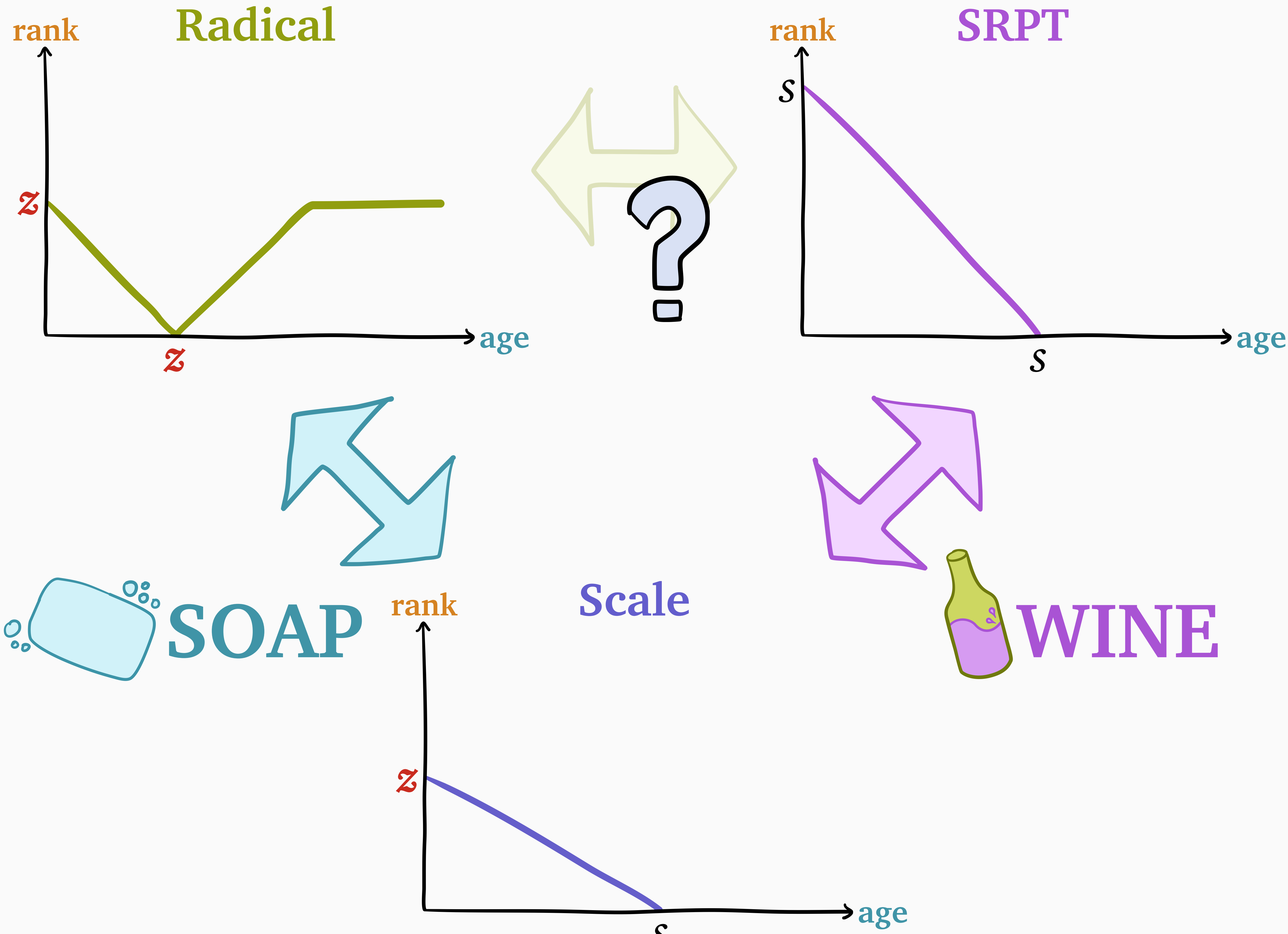


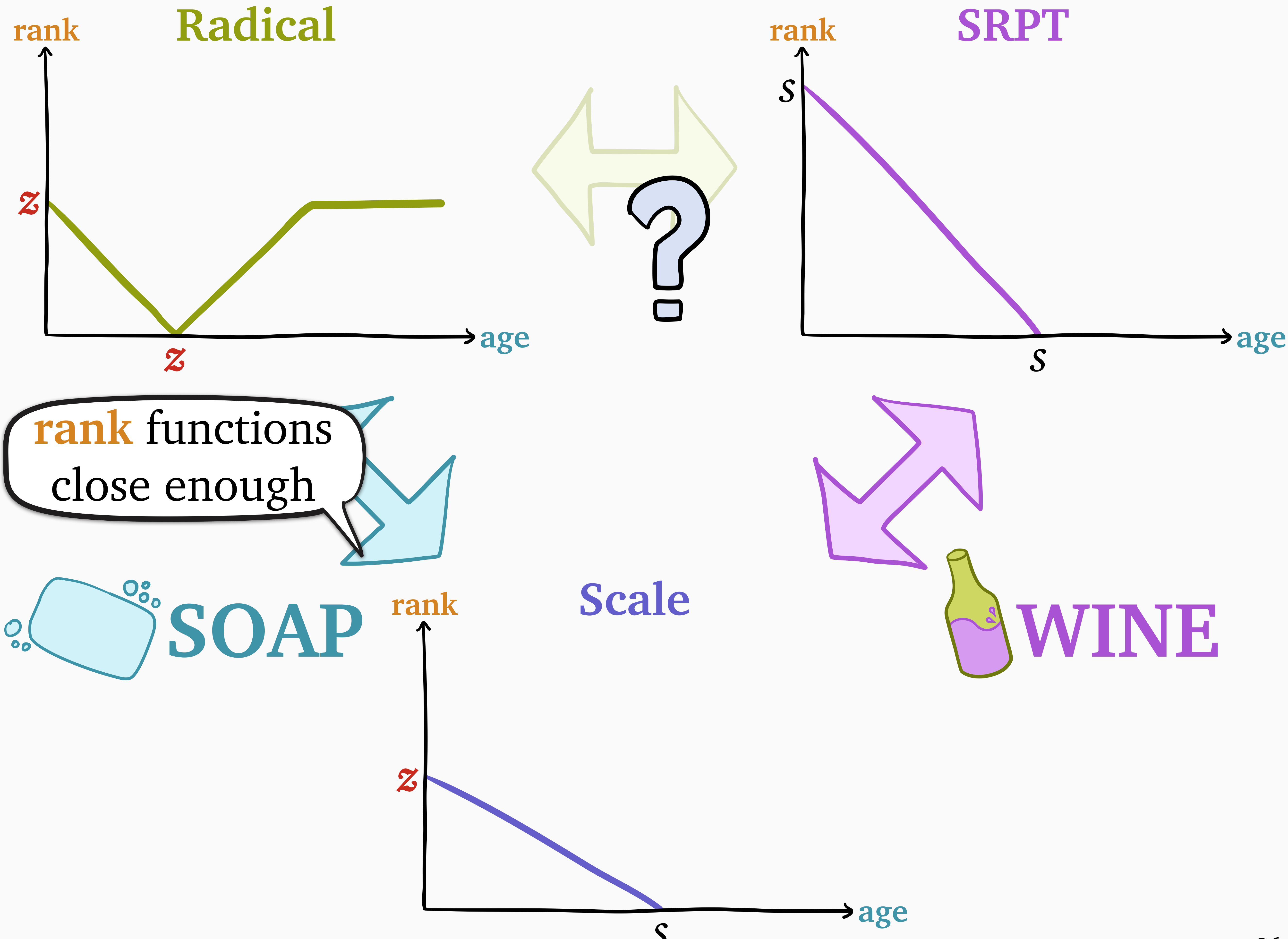
any **rank** function



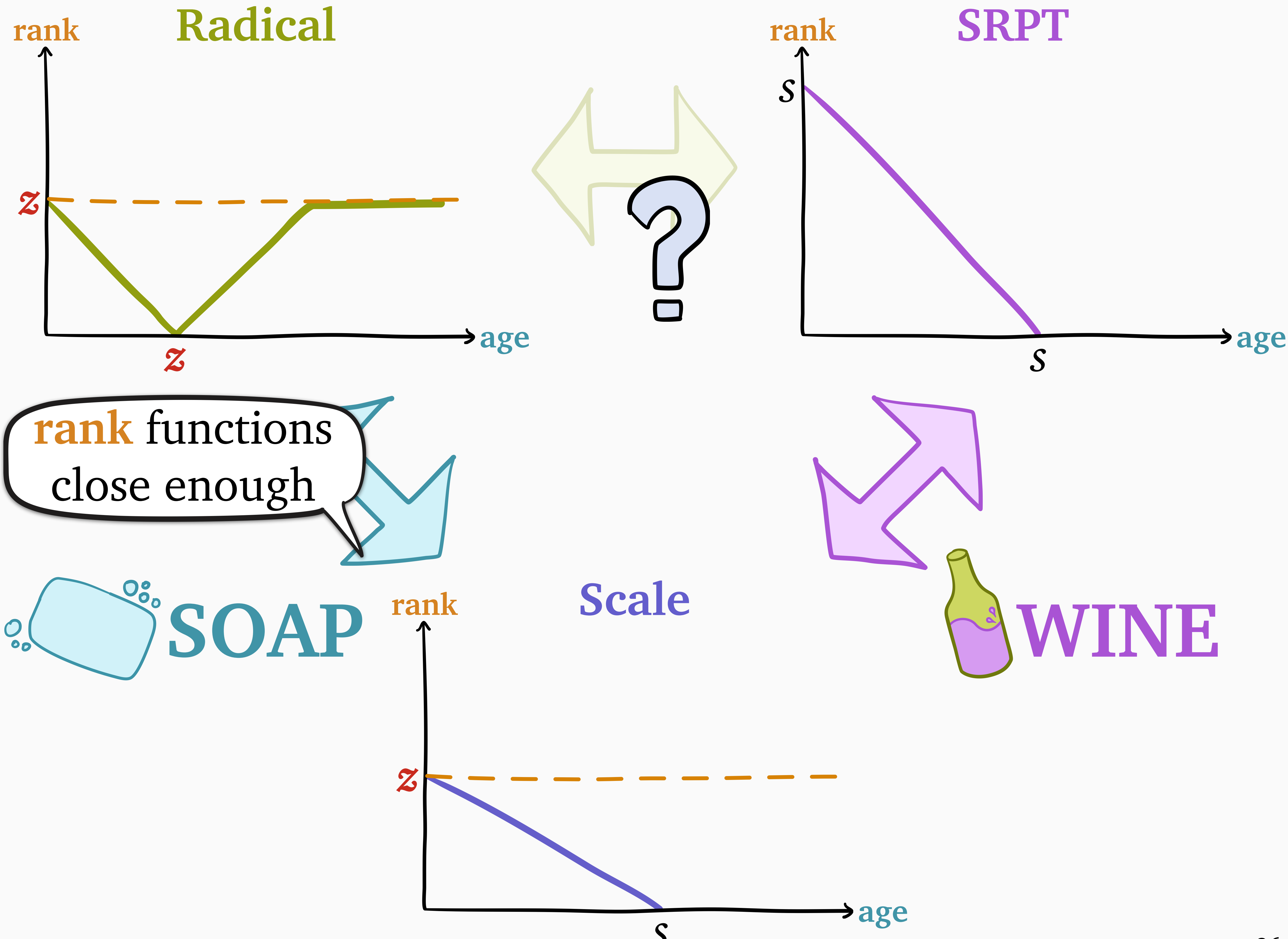
response time distribution T




Radical
rank
z
z
age
SRPT
rank
s
s
age
?
SOAP
WINE
Scale
rank
z
s
age



Radical
rank
z
z
age
SRPT
rank
s
s
age
rank functions close enough
SOAP
WINE
Scale
rank
z
s
age


Radical
rank
z
z
age
SRPT
rank
s
s
age
rank functions close enough
SOAP
WINE
Scale
rank
z
s
age

Comparing r-work

Lemma:

$$\mathbf{E}[W_{\mathrm{SRPT}}(r)] \le \mathbf{E}[W_{\mathrm{Scale}}(r)] \le \mathbf{E}[W_{\mathrm{SRPT}}(\gamma r)]$$

Comparing r-work

Lemma:

$$\mathbf{E}[W_{\mathrm{SRPT}}(r)] \le \mathbf{E}[W_{\mathrm{Scale}}(r)] \le \mathbf{E}[W_{\mathrm{SRPT}}(\gamma r)]$$

Key steps:

Comparing r-work

Lemma:

$$\mathbf{E}[W_{\mathbf{SRPT}}(r)] \le \mathbf{E}[W_{\mathbf{Scale}}(r)] \le \mathbf{E}[W_{\mathbf{SRPT}}(\gamma r)]$$

Key steps:

- **SRPT** minimizes mean r-work

Comparing r-work

Lemma:

$$\mathbf{E}[W_{\mathrm{SRPT}}(r)] \le \mathbf{E}[W_{\mathrm{Scale}}(r)] \le \mathbf{E}[W_{\mathrm{SRPT}}(\gamma r)]$$

Key steps:

- SRPT minimizes mean r-work

Comparing r-work

Lemma:

$$\mathbf{E}[W_{\mathrm{SRPT}}(r)] \le \mathbf{E}[W_{\mathrm{Scale}}(r)] \le \mathbf{E}[W_{\mathrm{SRPT}}(\gamma r)]$$

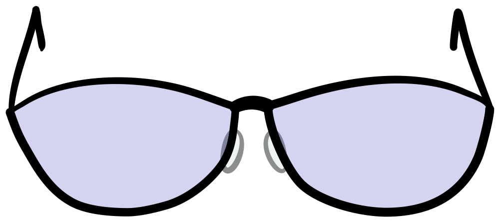
Key steps:

- **SRPT** minimizes mean r-work
- **Scale** minimizes mean *Scale-flavored* r-work

Comparing r-work

Lemma:

$$\mathbf{E}[W_{\text{SRPT}}(r)] \le \mathbf{E}[W_{\text{Scale}}(r)] \le \mathbf{E}[W_{\text{SRPT}}(\gamma r)]$$



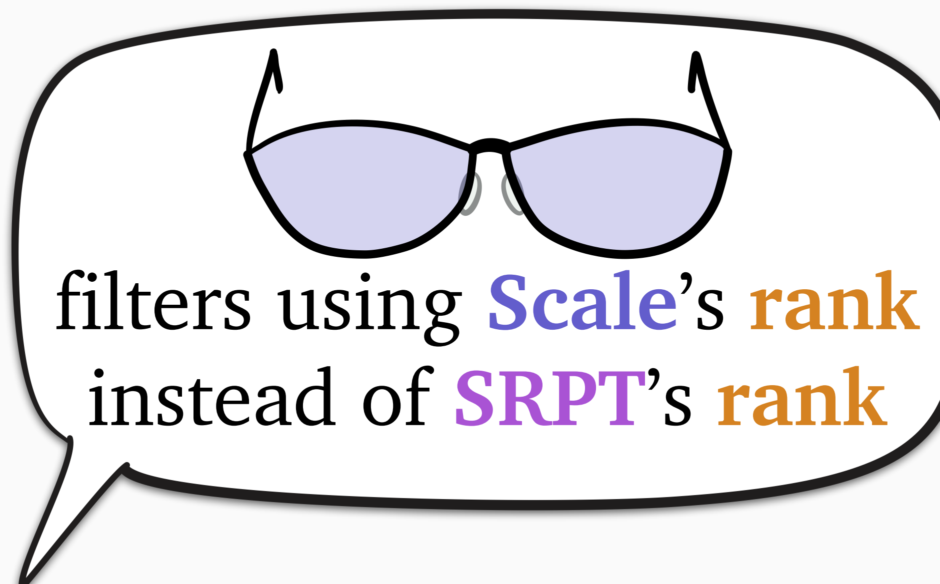
Key steps:

- SRPT minimizes mean r-work
- Scale minimizes mean *Scale-flavored* r-work

Comparing r-work

Lemma:

$$\mathbf{E}[W_{\mathrm{SRPT}}(r)] \le \mathbf{E}[W_{\mathrm{Scale}}(r)] \le \mathbf{E}[W_{\mathrm{SRPT}}(\gamma r)]$$



Key steps:

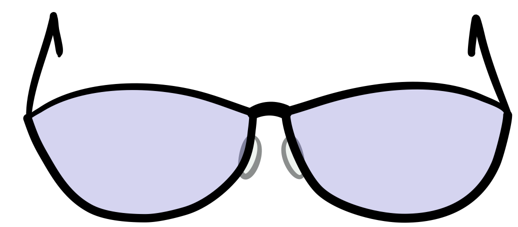
- SRPT minimizes mean r-work
- Scale minimizes mean *Scale-flavored* r-work
- Under any policy,

  r-work ≤ Scale-flavored αr-work ≤ $\frac{\alpha}{\beta} r$-work

Comparing r-work

Lemma:

$$\mathbf{E}[W_{\text{SRPT}}(r)] \le \mathbf{E}[W_{\text{Scale}}(r)] \le \mathbf{E}[W_{\text{SRPT}}(\gamma r)]$$



filters using **Scale**'s **rank**
instead of **SRPT**'s **rank**

Key steps:

- **SRPT** minimizes mean r-work
- **Scale** minimizes mean *Scale-flavored* r-work
- Under any policy,

  r-work $\le$ **Scale**-flavored αr-work $\le \frac{\alpha}{\beta} r$-work

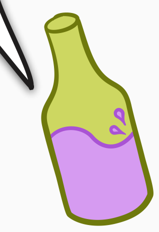
Comparing r-work

Lemma:

$$\mathbf{E}[W_{\text{SRPT}}(r)] \le \mathbf{E}[W_{\text{Scale}}(r)] \le \mathbf{E}[W_{\text{SRPT}}(\gamma r)]$$

Key steps:

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



- SRPT minimizes mean r-work
- Scale minimizes mean *Scale-flavored* r-work
- Under any policy,

  r-work $\le$ Scale-flavored αr-work $\le \frac{\alpha}{\beta} r$-work

Comparing r-work

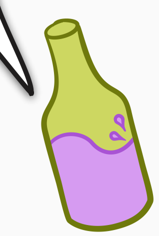
Lemma:

$$\mathbf{E}[W_{\mathrm{SRPT}}(r)] \le \mathbf{E}[W_{\mathrm{Scale}}(r)] \le \mathbf{E}[W_{\mathrm{SRPT}}(\gamma r)]$$

⇩

$$\mathbf{E}[N_{\mathrm{SRPT}}] \le \mathbf{E}[N_{\mathrm{Scale}}] \le \gamma \mathbf{E}[N_{\mathrm{SRPT}}]$$

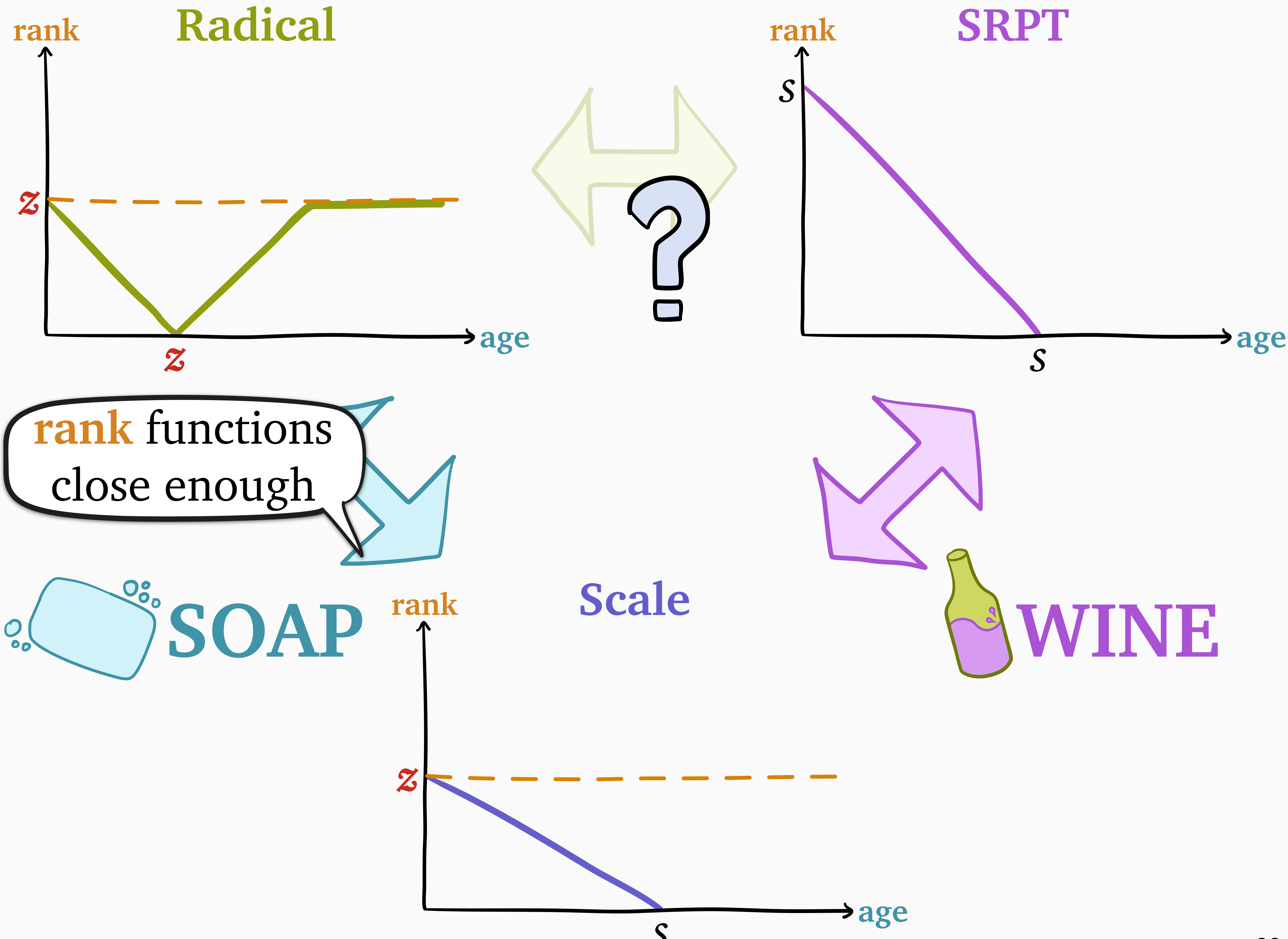
$$\mathbf{E}[N] = \int_0^\infty \frac{\mathbf{E}[W(r)]}{r^2} \, \mathrm{d}r$$

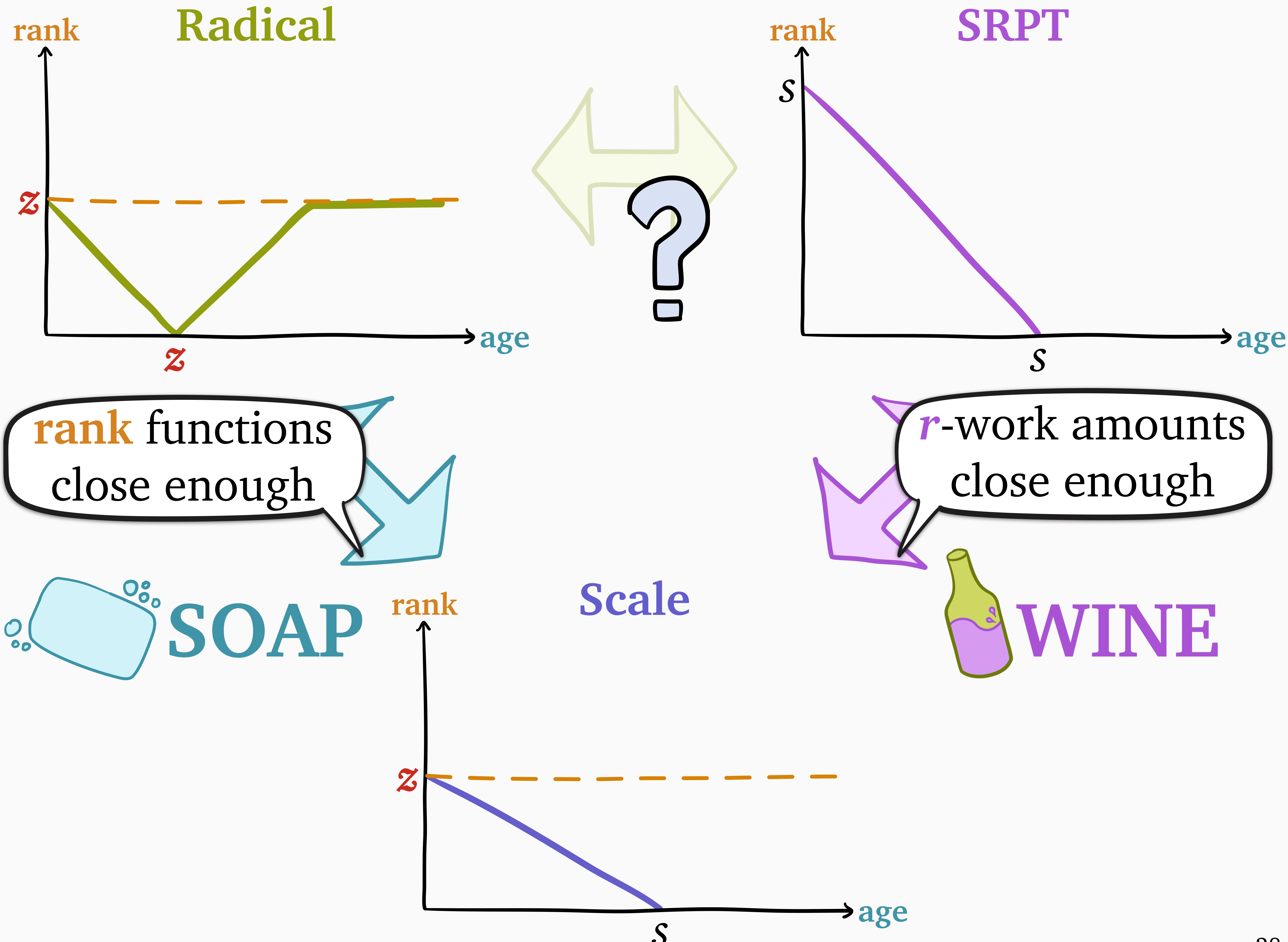


Key steps:

- SRPT minimizes mean r-work
- Scale minimizes mean *Scale-flavored* r-work
- Under any policy,

  r-work $\le$ Scale-flavored αr-work $\le \frac{\alpha}{\beta} r$-work


rank
Radical
age
z
z
rank
SRPT
s
s
age
rank functions
close enough
SOAP
rank
Scale
z
s
age
WINE


Radical
rank
age
z
z
SRPT
rank
age
s
s
rank functions close enough
r-work amounts close enough
SOAP
WINE
Scale
rank
age
z
s

Today's talk

scheduling with

noisy predictions

TCS 🤺🤺 **Queueing**

Today's talk

scheduling with

noisy predictions

TCS 🤺🤺 **Queueing**

- **TCS**: need to be careful

Today's talk

scheduling with

noisy predictions

TCS 🤺🤺 **Queueing**

- **TCS**: need to be careful
- **Queueing**: simple **rank**-based policy suffices

Today's talk

scheduling with

multiple servers

TCS 🤝 **Queueing**

- SRPT-k is good
- In general: good to adapt optimal single-server policy

scheduling with

noisy predictions

TCS 🤺🤺 **Queueing**

- **TCS**: need to be careful
- **Queueing**: simple **rank**-based policy suffices

Today's talk

scheduling with
multiple servers

TCS 🤝 **Queueing**

- SRPT-*k* is good
- In general: good to adapt optimal single-server policy

scheduling with
noisy predictions

TCS 🤺🤺 **Queueing**

- **TCS**: need to be careful
- **Queueing**: simple **rank**-based policy suffices

New tools:

Today's talk

scheduling with

multiple servers

TCS 🤝 **Queueing**

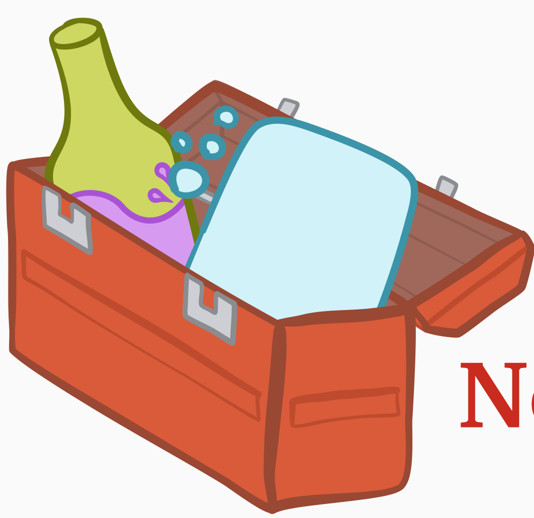
- SRPT-*k* is good
- In general: good to adapt optimal single-server policy

scheduling with

noisy predictions

TCS 🤺🤺 **Queueing**

- **TCS**: need to be careful
- **Queueing**: simple **rank**-based policy suffices



New tools: **WINE** and **SOAP**

Today's talk

scheduling with
multiple servers

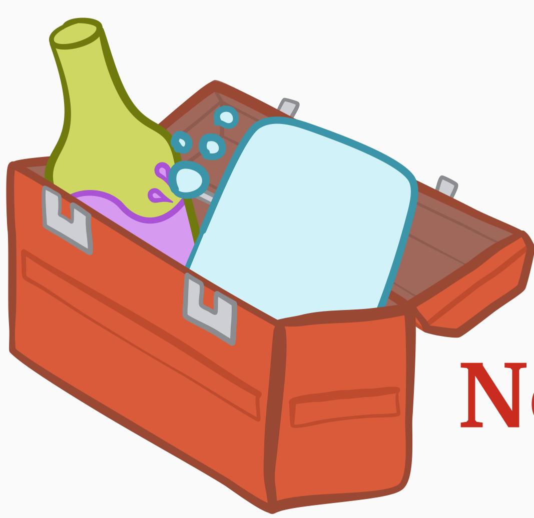
TCS 🤝 **Queueing**

- SRPT-*k* is good
- In general: good to adapt optimal single-server policy

scheduling with
noisy predictions

TCS 🤺🤺 **Queueing**

- **TCS**: need to be careful
- **Queueing**: simple **rank**-based policy suffices



potential TCS use?

New tools: **WINE** and **SOAP**

Today's talk

scheduling with

multiple servers

TCS 🤝 **Queueing**

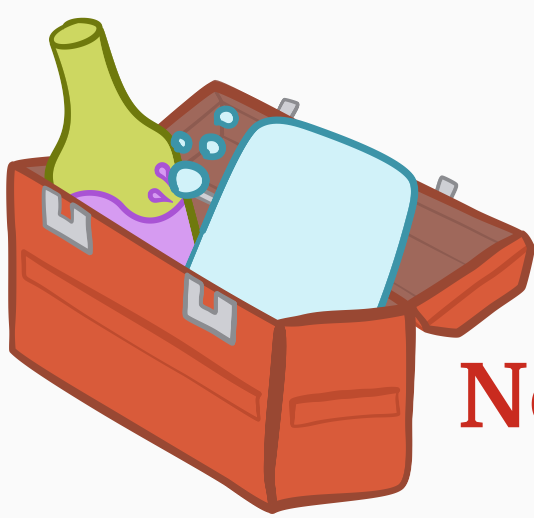
- SRPT-*k* is good
- In general: good to adapt optimal single-server policy

scheduling with

noisy predictions

TCS 🤺🤺 **Queueing**

- **TCS**: need to be careful
- **Queueing**: simple **rank**-based policy suffices



potential TCS use?

detailed system modeling?

New tools: **WINE** and **SOAP**

References

Scully, Harchol-Balter, and Scheller-Wolf (2018). "SOAP: One Clean Analysis of All Age-Based Scheduling Policies." *Proc. ACM Meas. Anal. Comput. Syst.* (SIGMETRICS 2018).

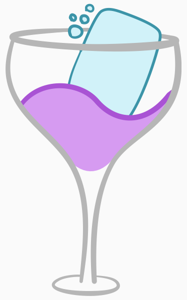
- Introduces **rank** functions and the general SOAP analysis
- **Finalist: 2019 INFORMS APS Best Student Paper Prize**

Grosof, Scully, and Harchol-Balter (2018). "SRPT for Multiserver Systems." *Perform. Eval.* (PERFORMANCE 2018).

- First **queueing** analysis of SRPT-k
- Uses tagged job method plus **worst-case** r-work decomposition
- **Winner: PERFORMANCE 2018 Best Student Paper Award**

Scully, Grosof, and Harchol-Balter (2020). "The Gittins Policy is Nearly Optimal in the M/G/k under Extremely General Conditions." *Proc. ACM Meas. Anal. Comput. Syst.* (SIGMETRICS 2021).

- First **queueing** analysis of **Gittins**-k
- Introduces WINE
- **Winner: 2022 INFORMS George Nicholson Student Paper Competition**



Scully, Grosof, and Mitzenmacher (2022). "Uniform Bounds for Scheduling with Job Size Estimates." *13th Innovations in Theoretical Computer Science Conference* (ITCS 2022).

- First **queueing** competitive ratios for **noisy predictions**
- Uses both SOAP and WINE