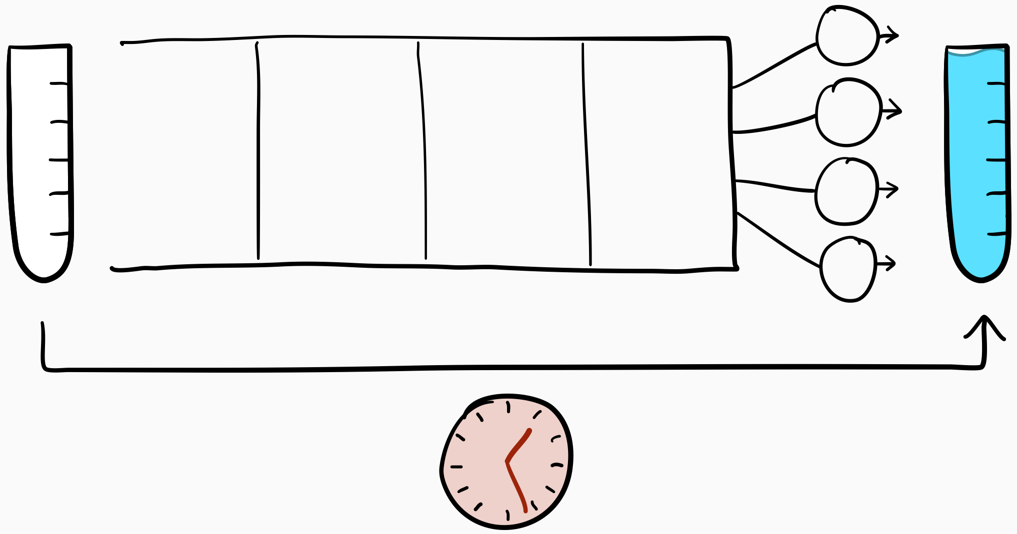
The Gittins Policy Is Nearly Optimal in the M/G/*k*

under Extremely General Conditions

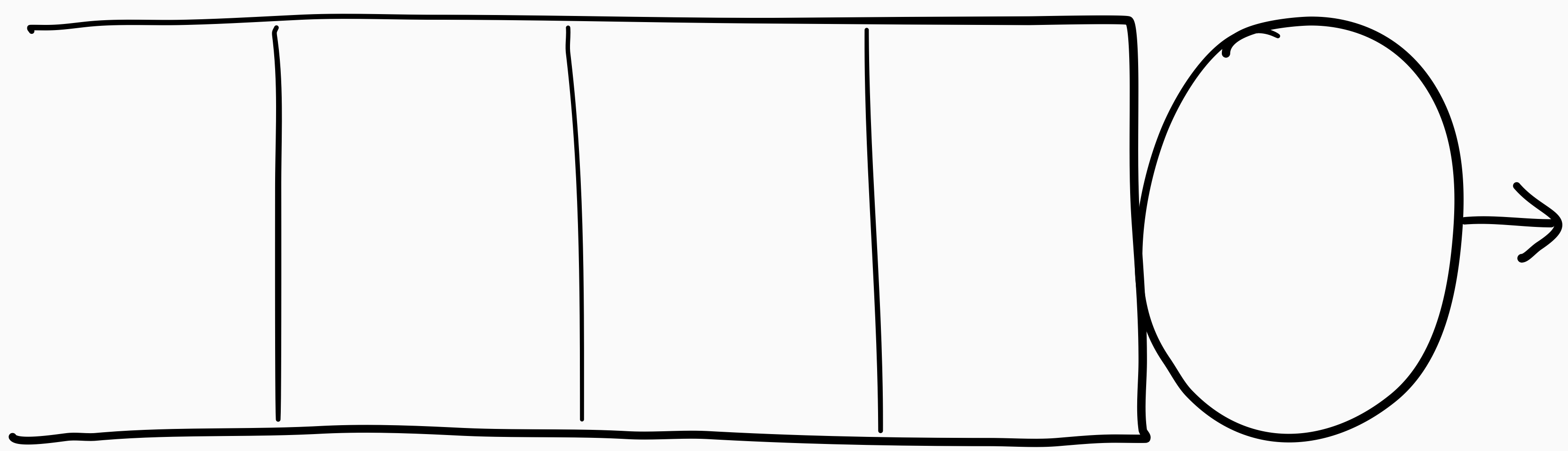
Ziv Scully
Isaac Grosof
Mor Harchol-Balter

Carnegie Mellon University

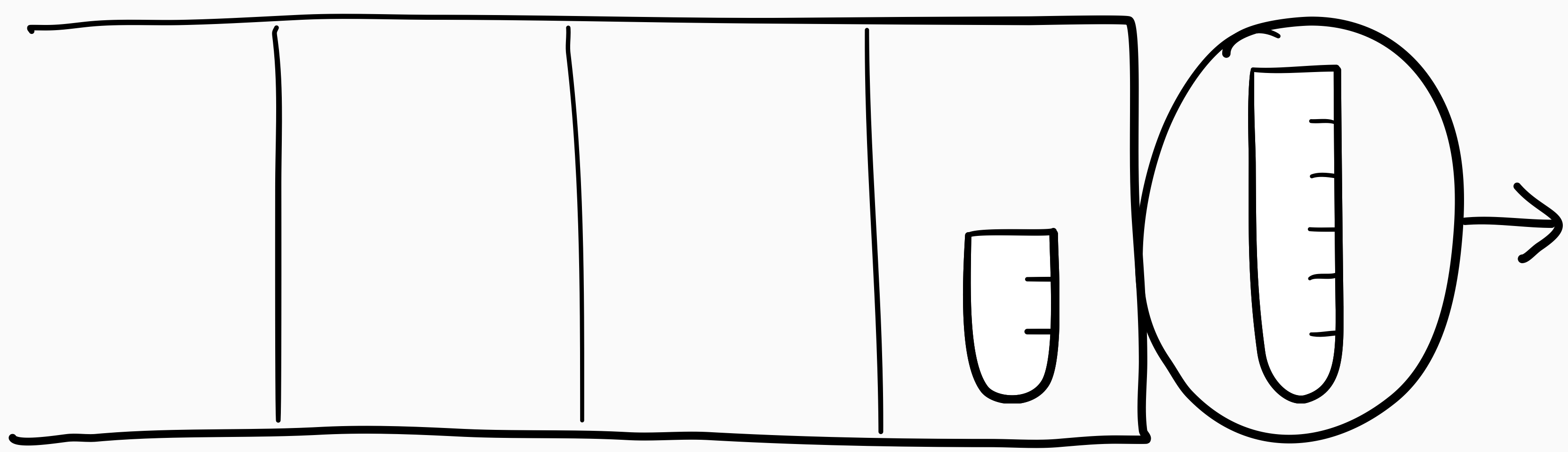


How should we schedule jobs to minimize delay?

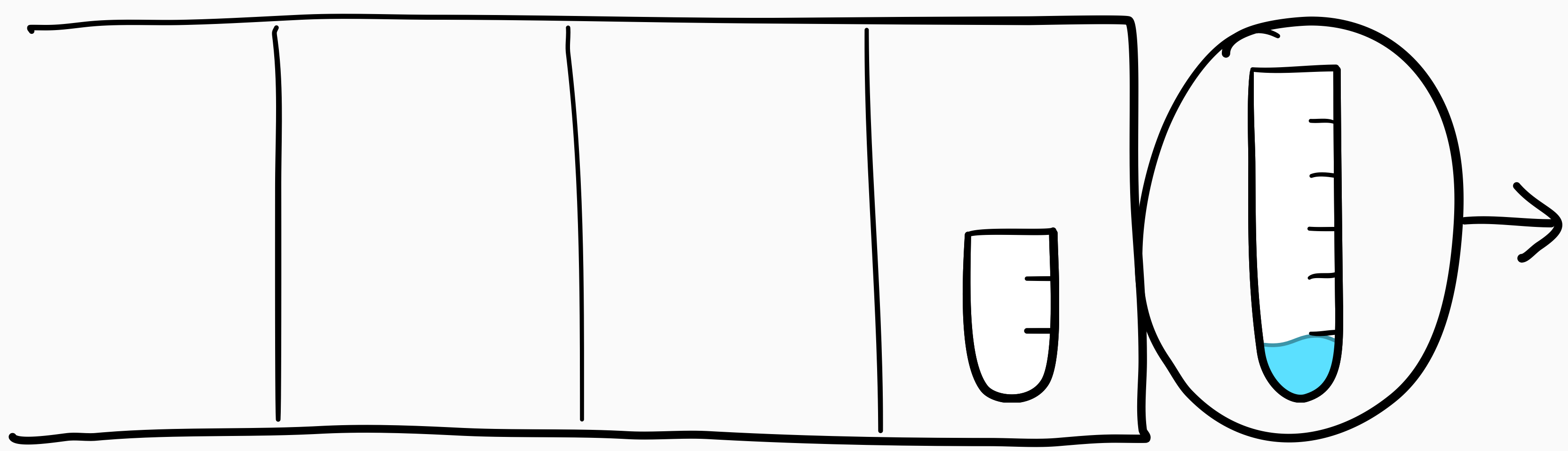
How should we schedule jobs to minimize delay?



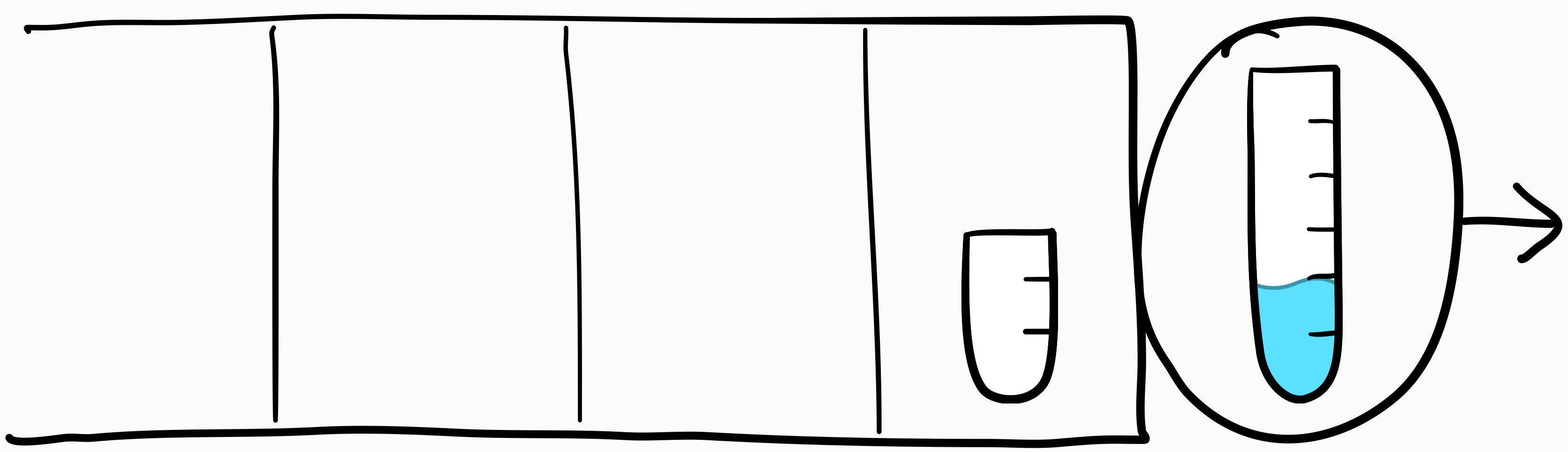
How should we schedule jobs to minimize delay?



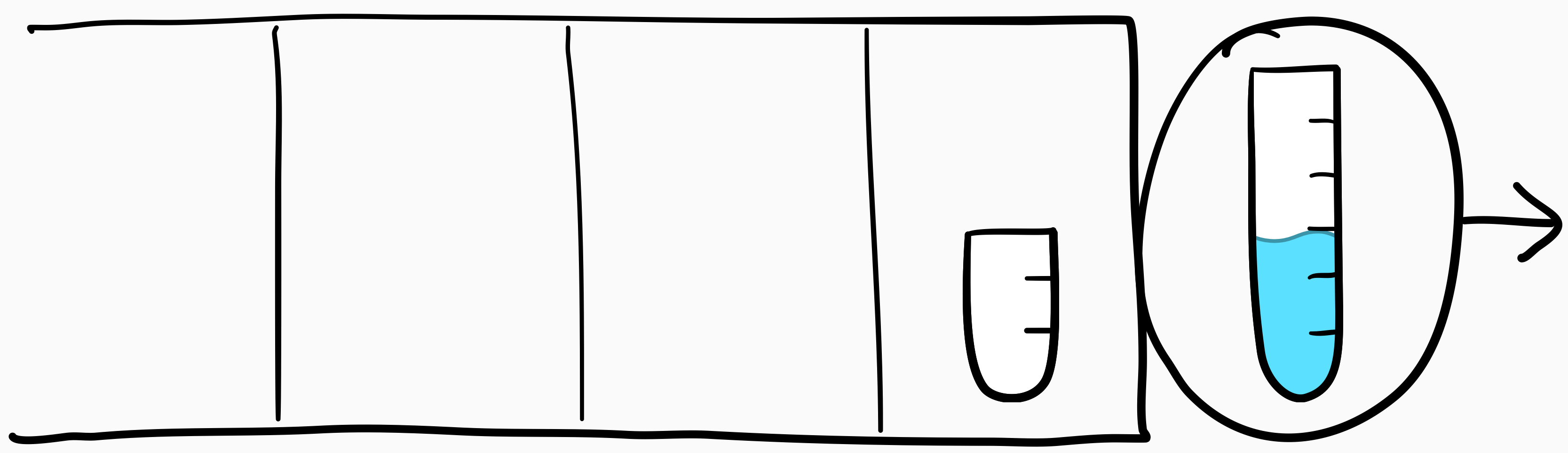
How should we schedule jobs to minimize delay?



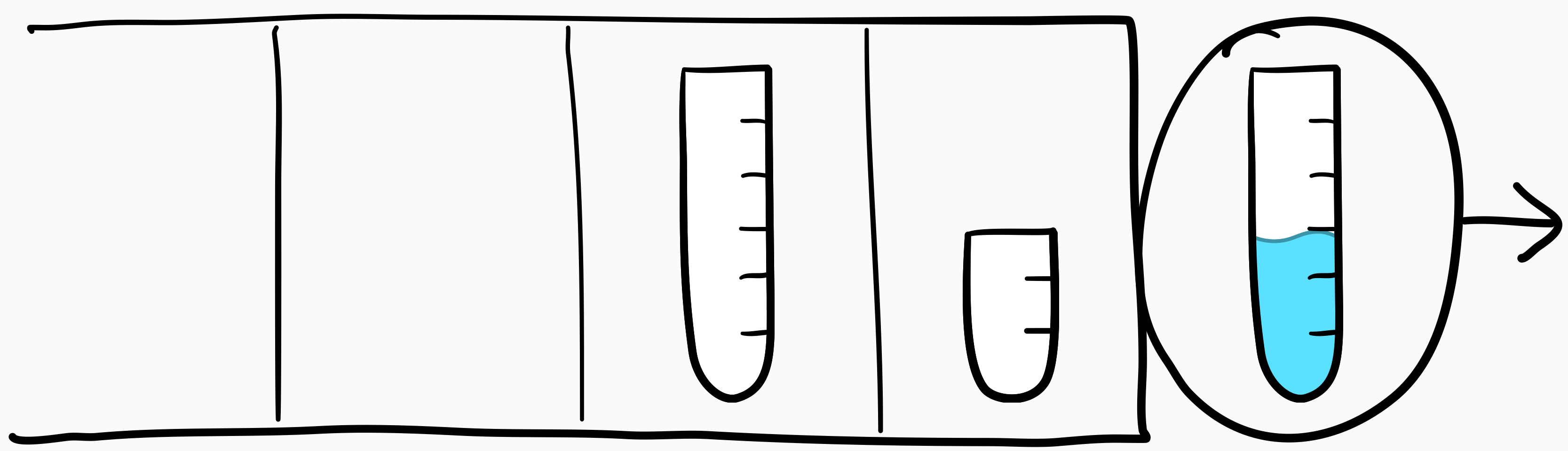
How should we schedule jobs to minimize delay?



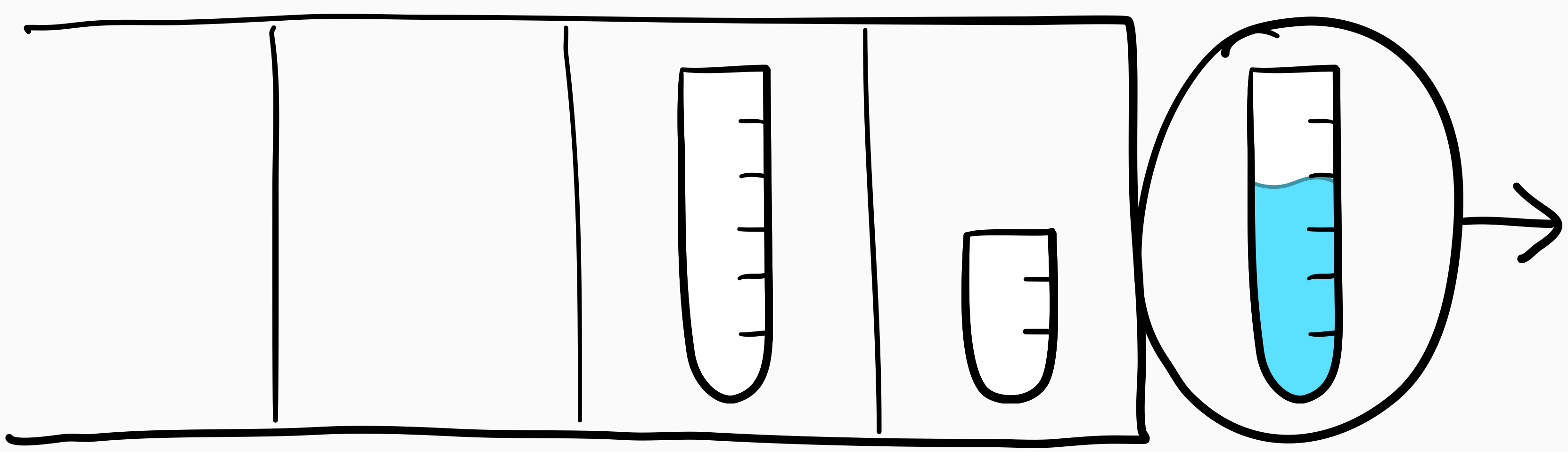
How should we schedule jobs to minimize delay?



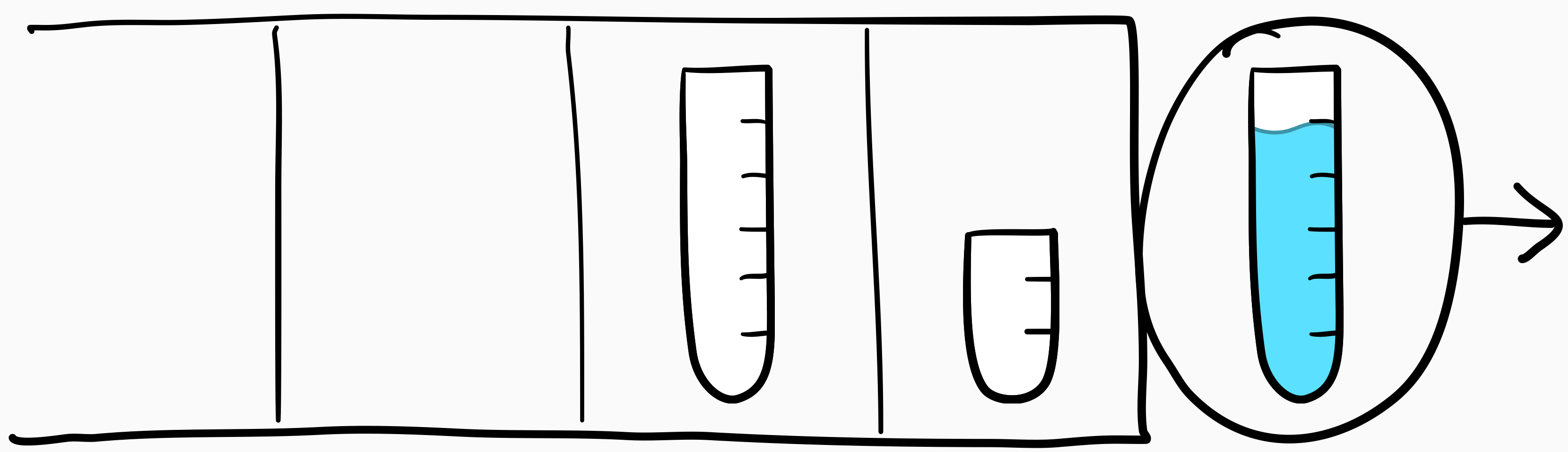
How should we schedule jobs to minimize delay?



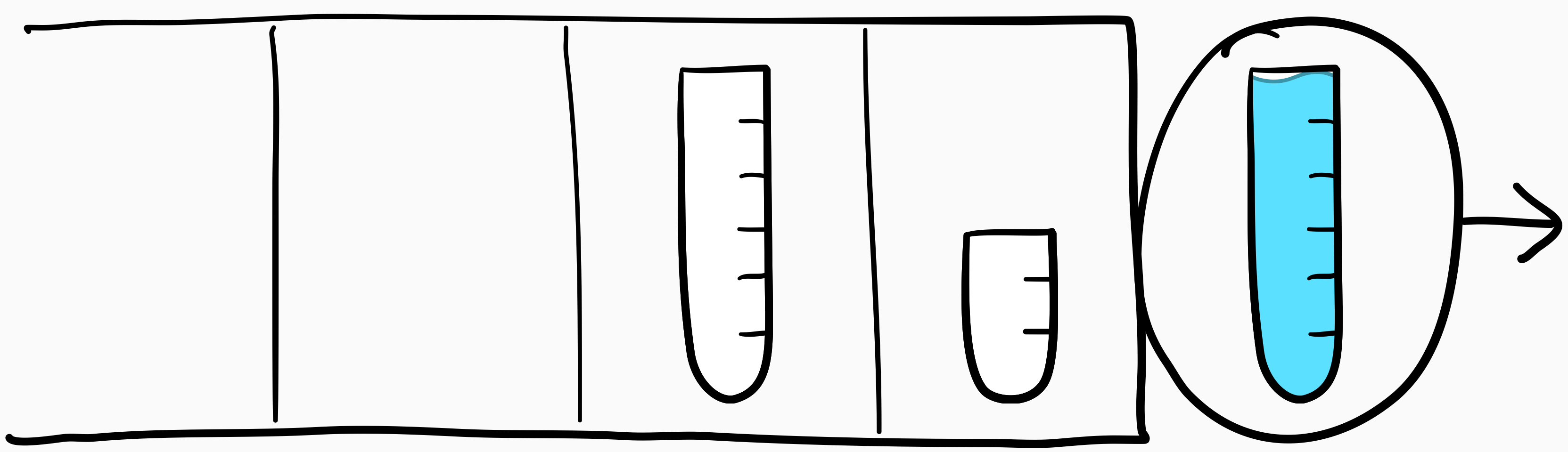
How should we schedule jobs to minimize delay?



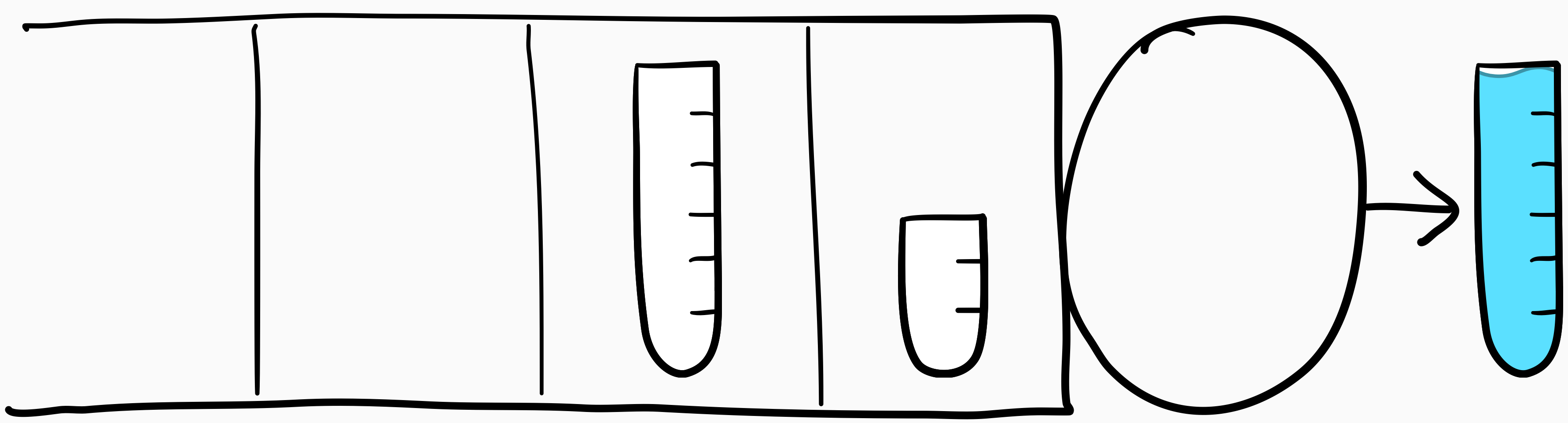
How should we schedule jobs to minimize delay?



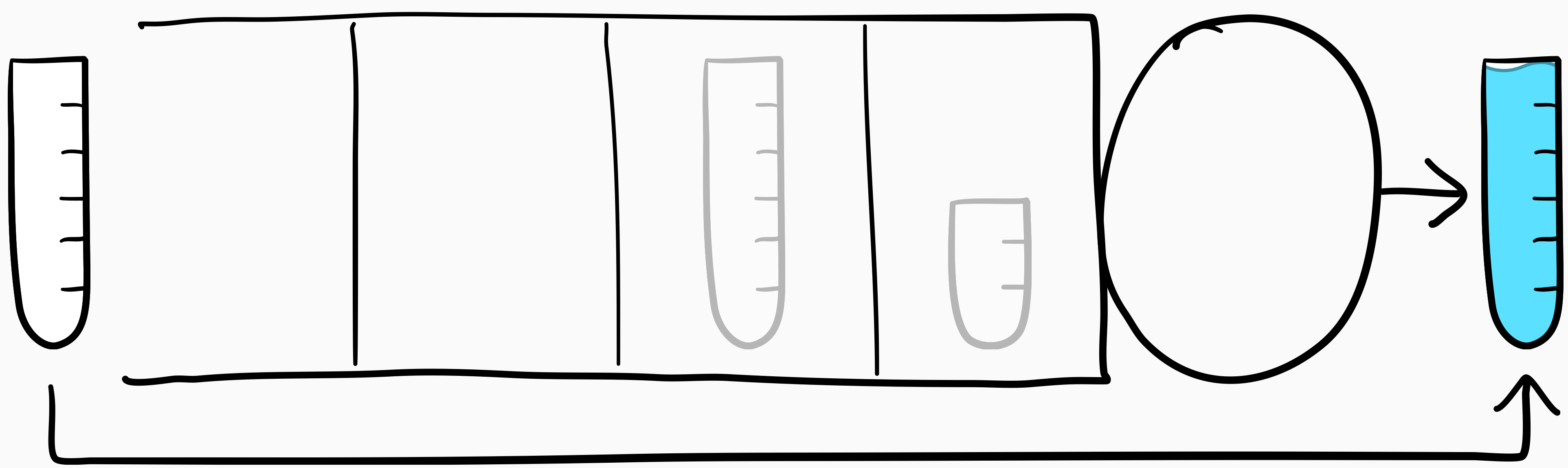
How should we schedule jobs to minimize delay?

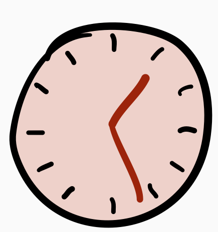


How should we schedule jobs to minimize delay?



How should we schedule jobs to minimize delay?

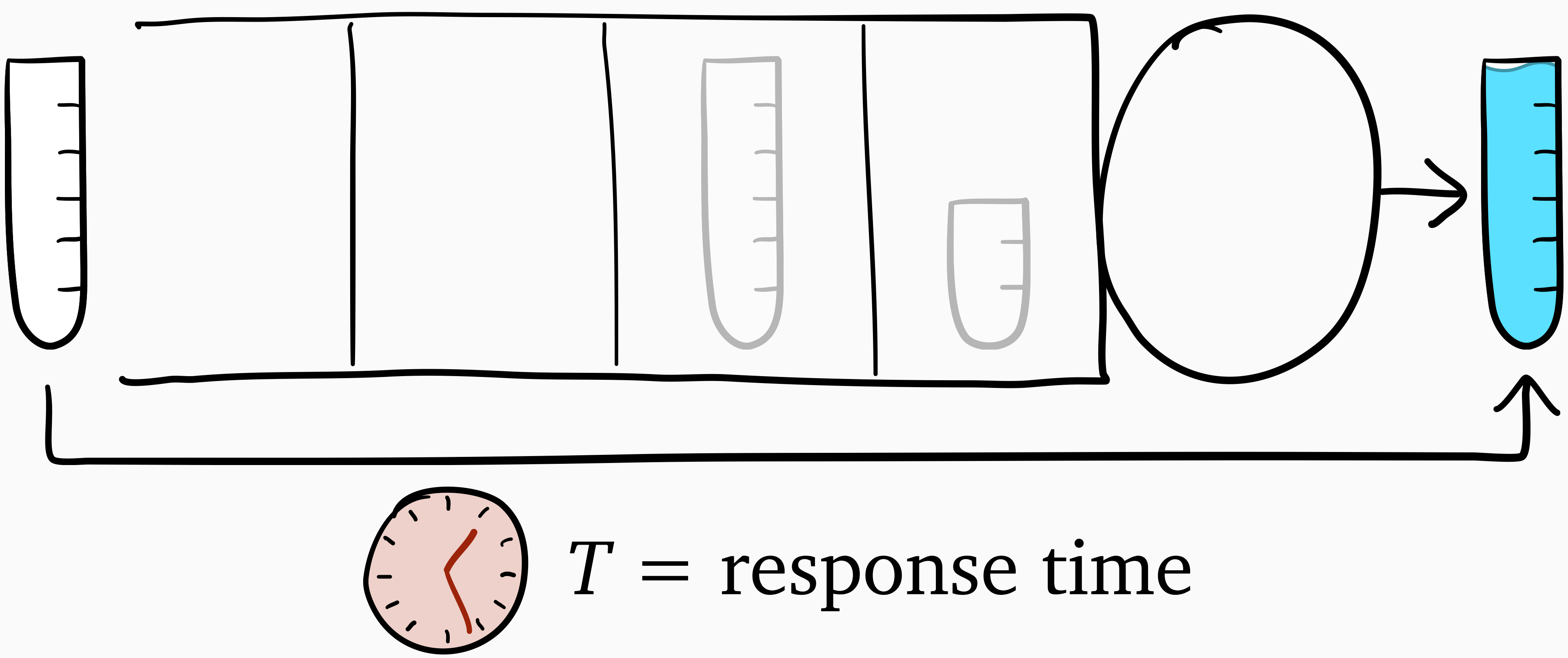




T = response time

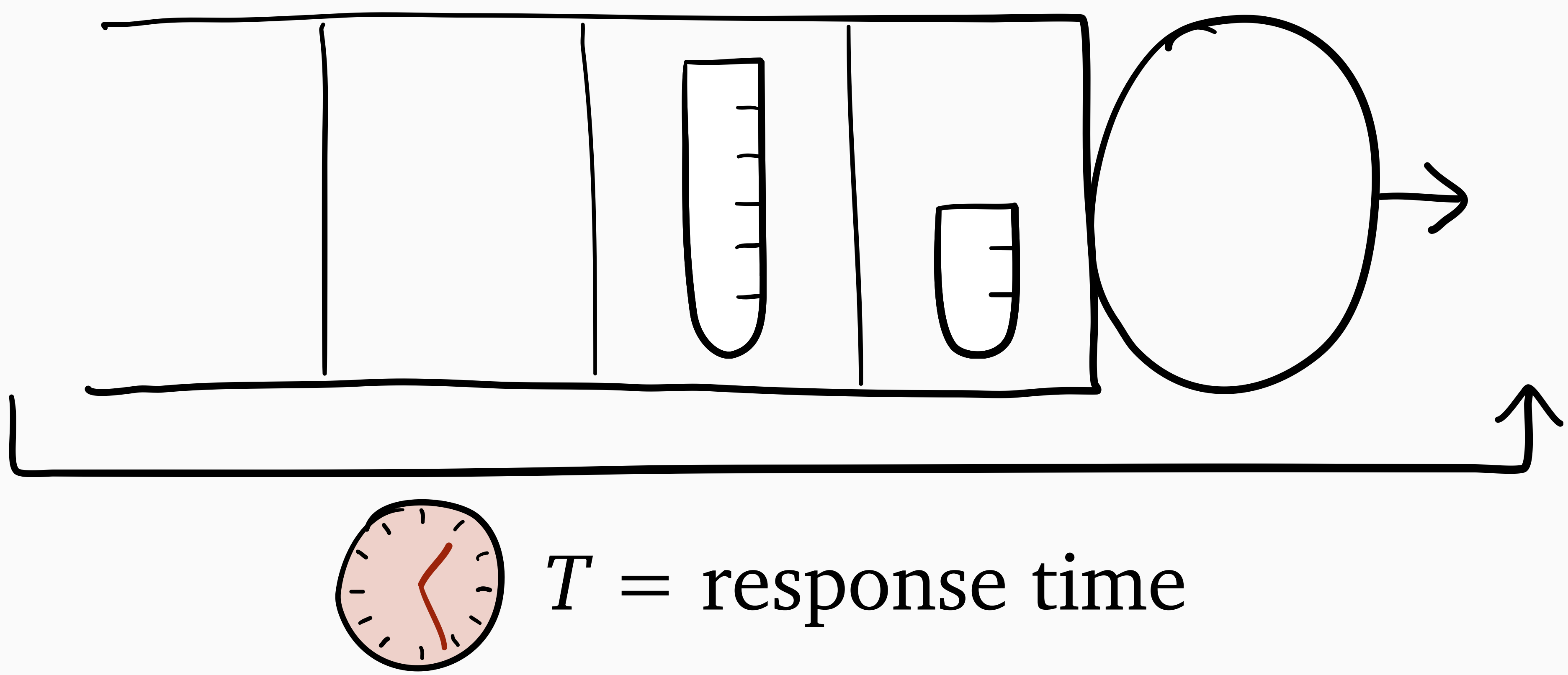
How should we schedule jobs to minimize delay?

$\mathbf{E}[T]$



How should we schedule jobs to minimize delay?

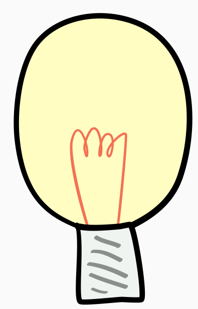
$\mathbf{E}[T]$



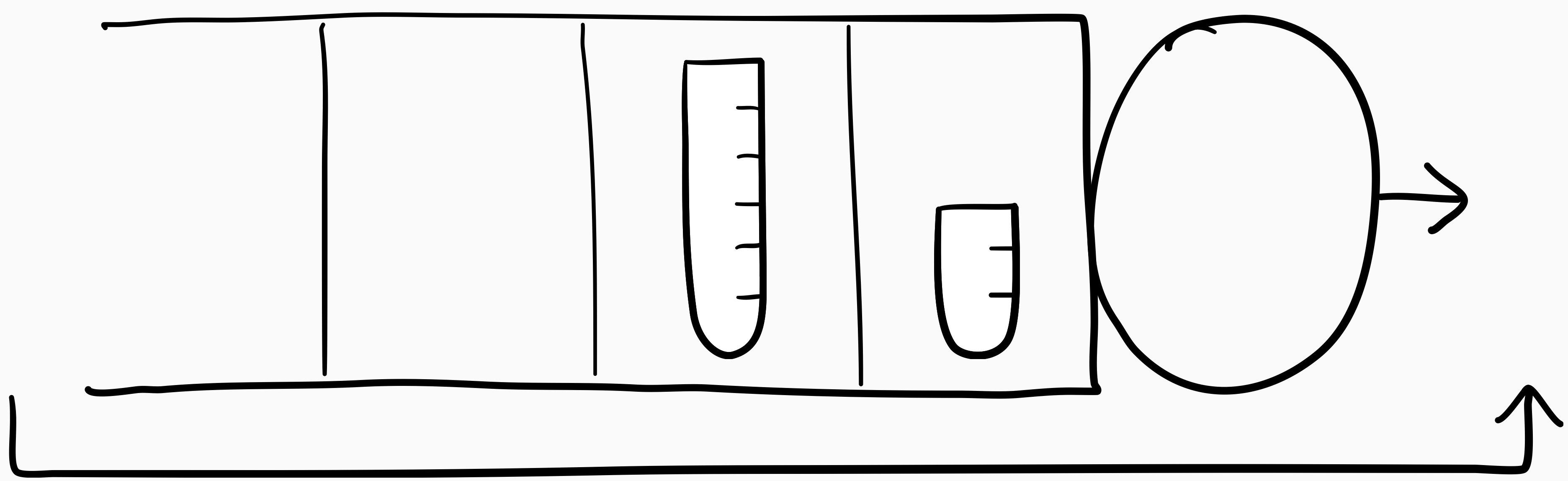
T = response time

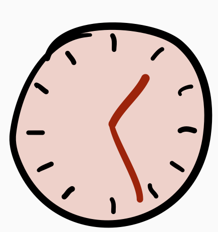
How should we schedule jobs to minimize delay?

$\mathbf{E}[T]$



Serve short jobs before long jobs

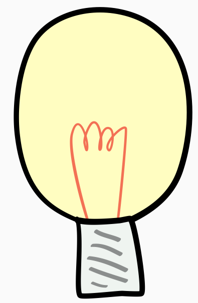




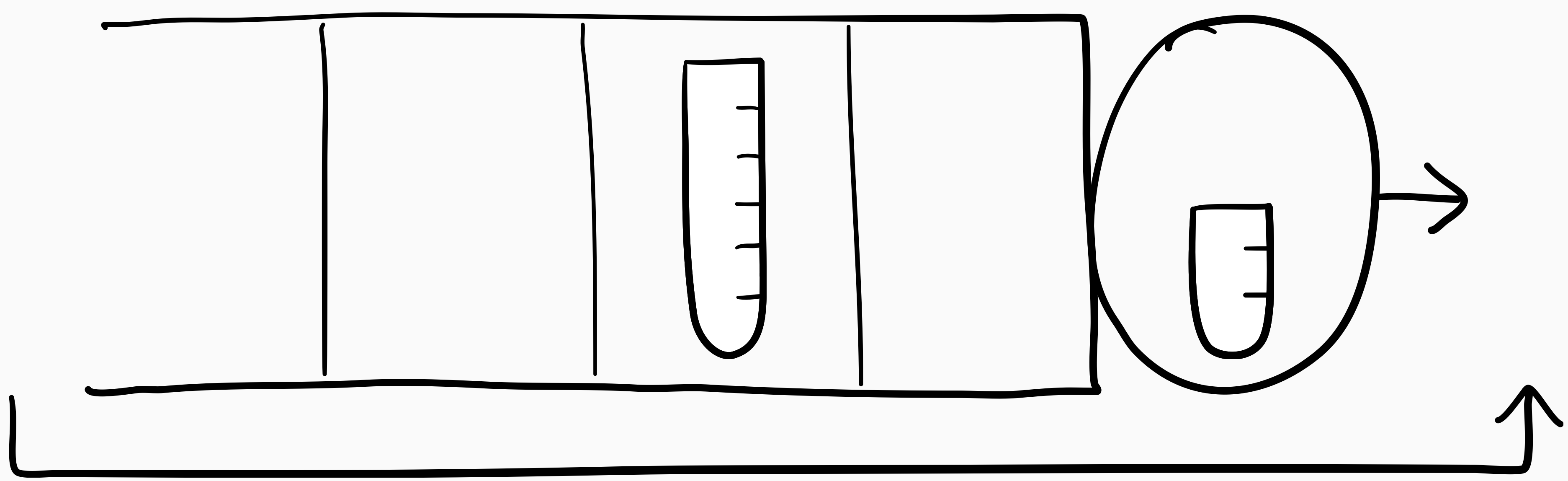
T = response time

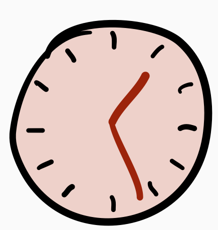
How should we schedule jobs to minimize delay?

$\mathbf{E}[T]$



Serve short jobs before long jobs

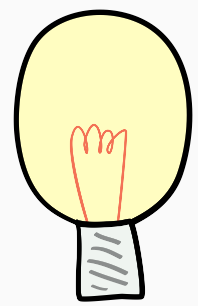




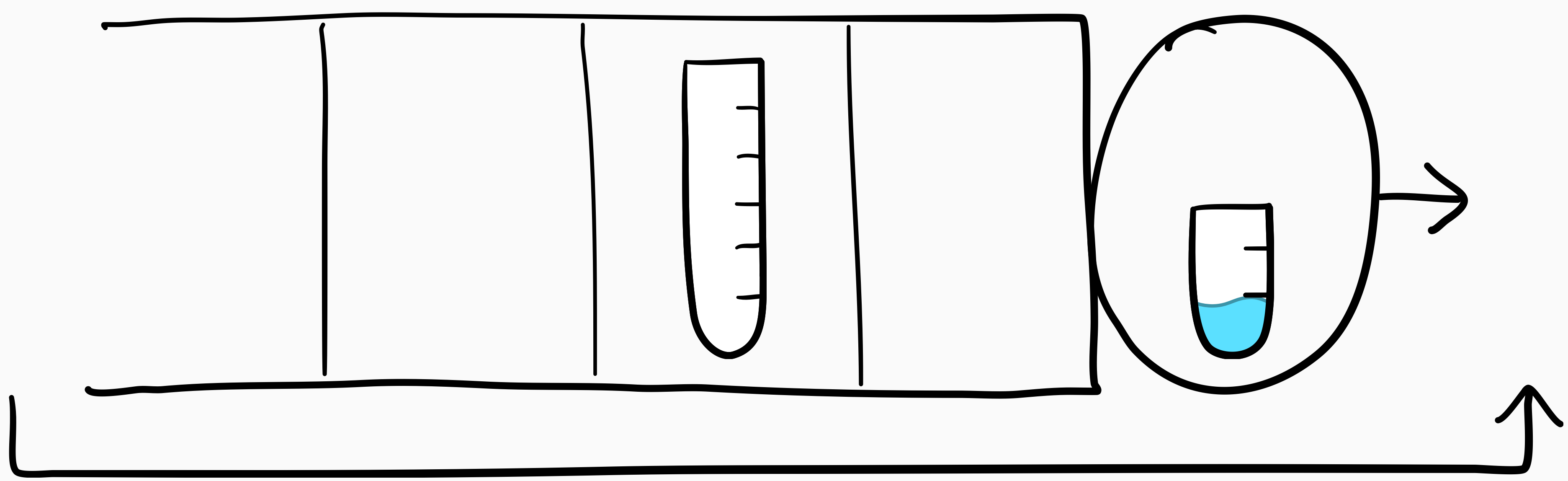
T = response time

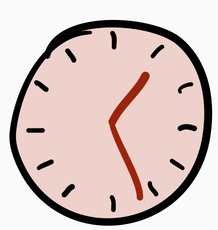
How should we schedule jobs to minimize delay?

$\mathbf{E}[T]$



Serve short jobs before long jobs

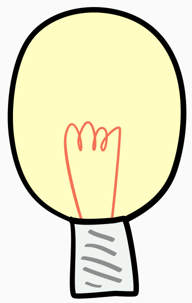




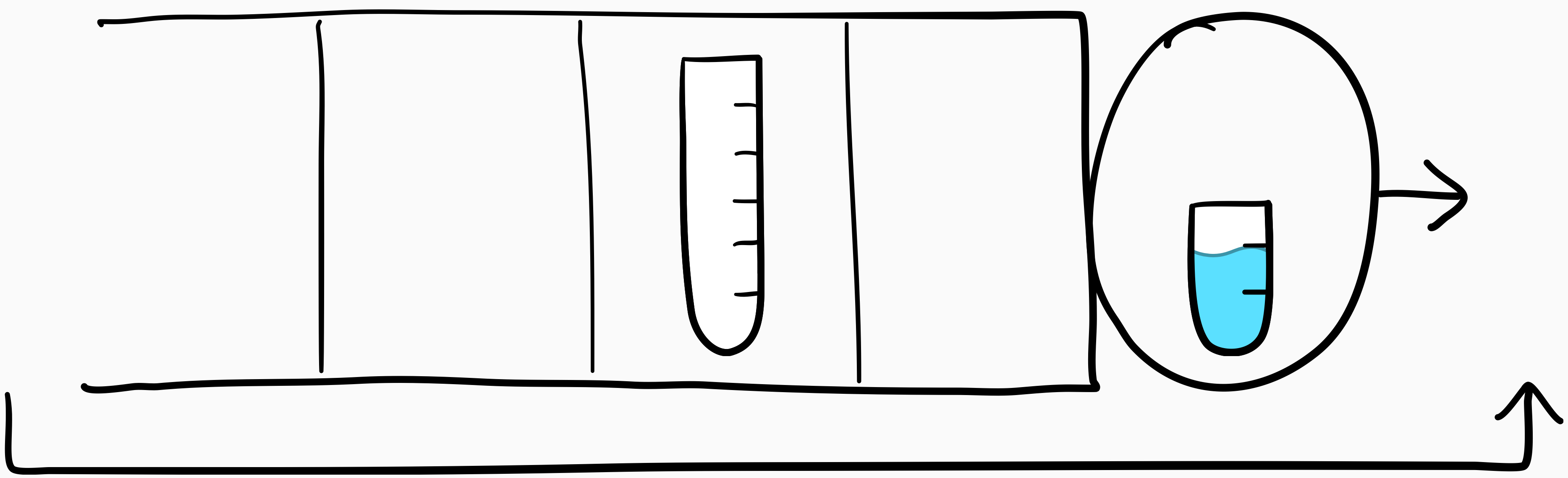
T = response time

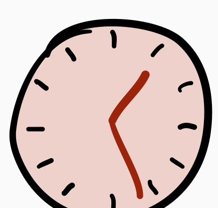
How should we schedule jobs to minimize delay?

$\mathbf{E}[T]$



Serve short jobs before long jobs

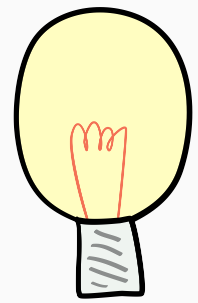




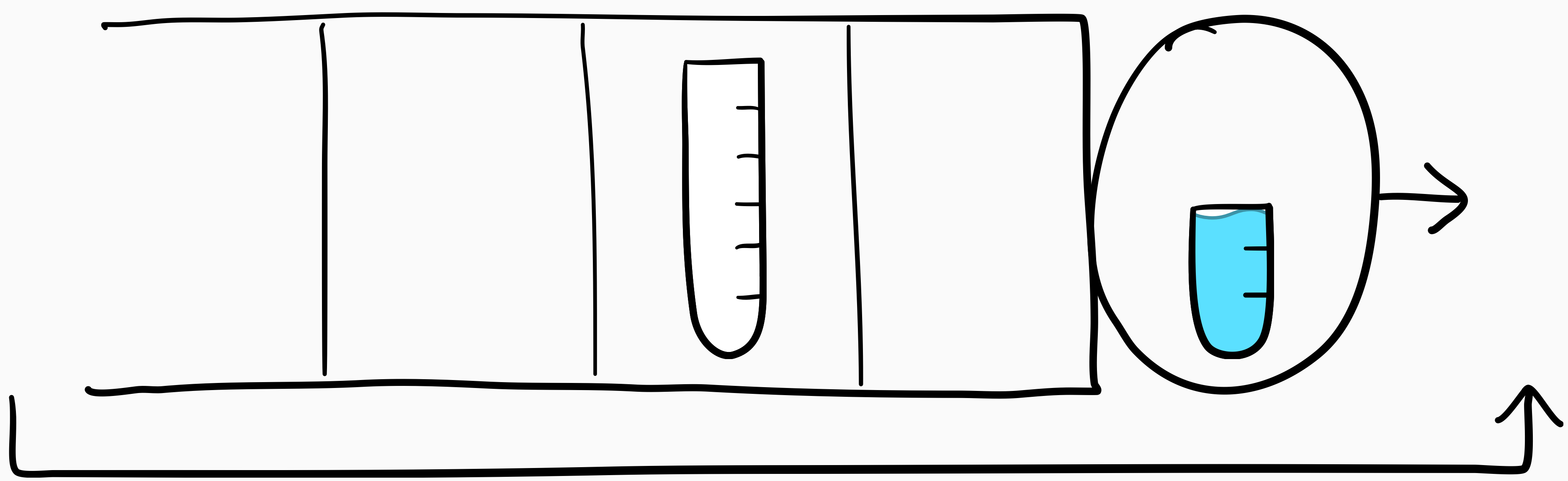
T = response time

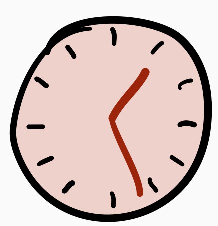
How should we schedule jobs to minimize delay?

$\mathbf{E}[T]$



Serve short jobs before long jobs

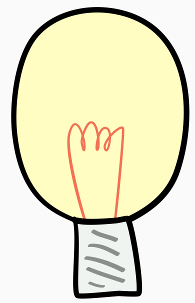




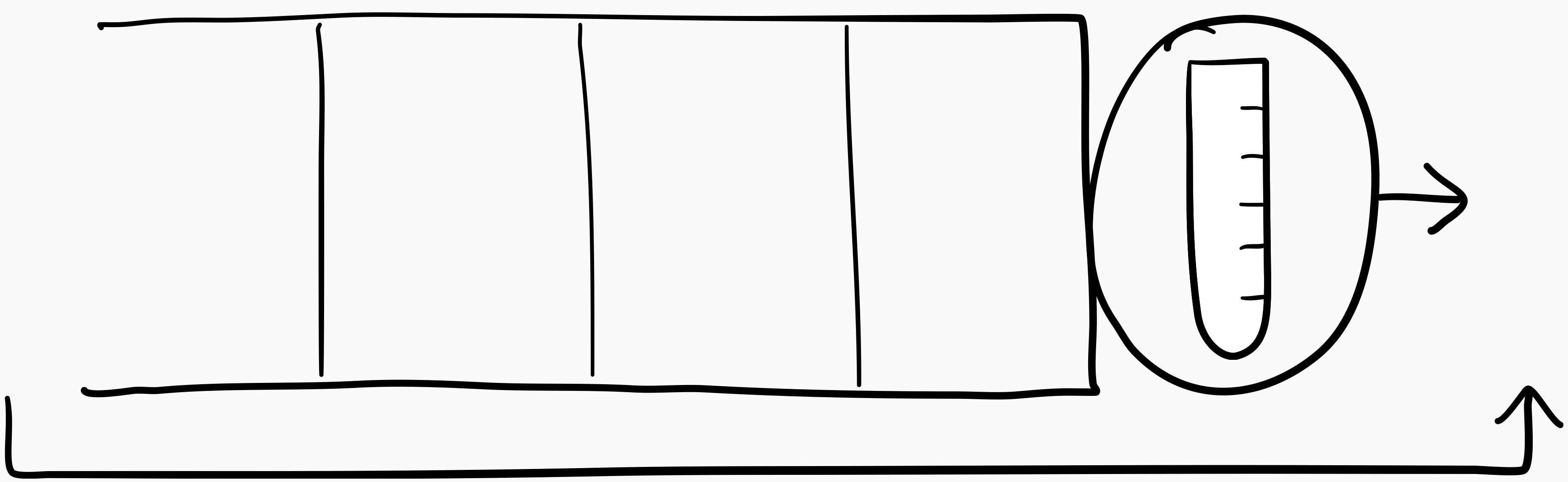
T = response time

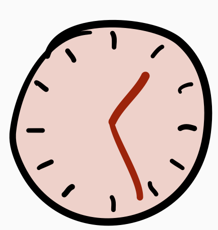
How should we schedule jobs to minimize delay?

$\mathbf{E}[T]$



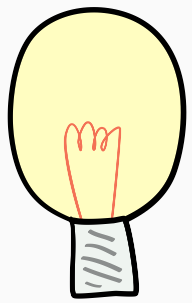
Serve short jobs
before long jobs



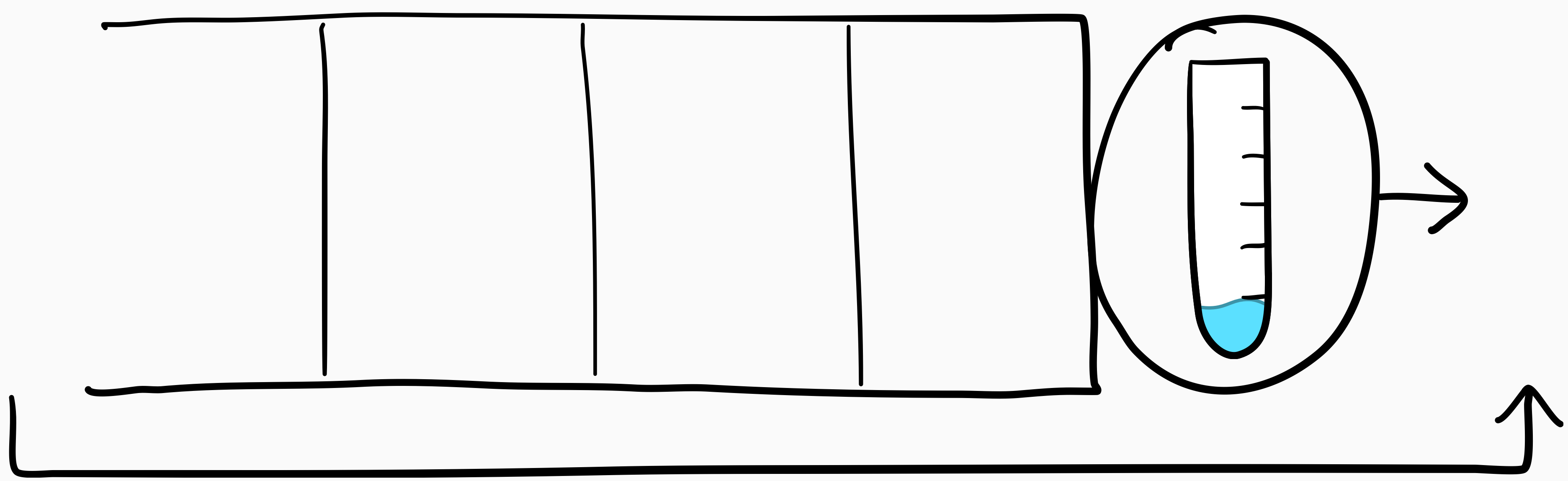


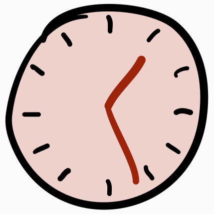
T = response time

How should we schedule jobs to minimize delay?



Serve short jobs before long jobs

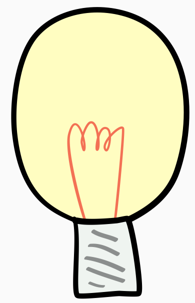




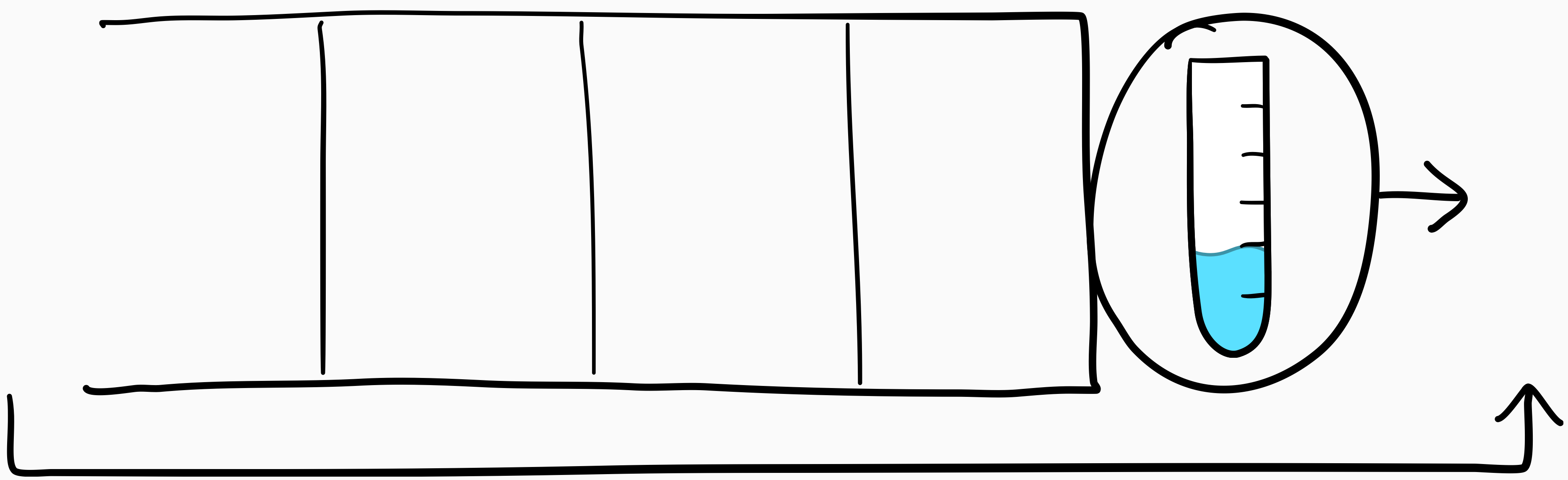
T = response time

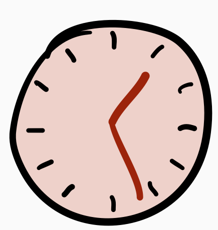
How should we schedule jobs to minimize delay?

$\mathbf{E}[T]$



Serve short jobs before long jobs

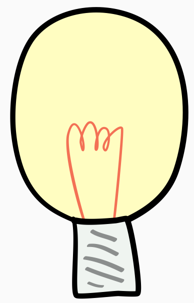




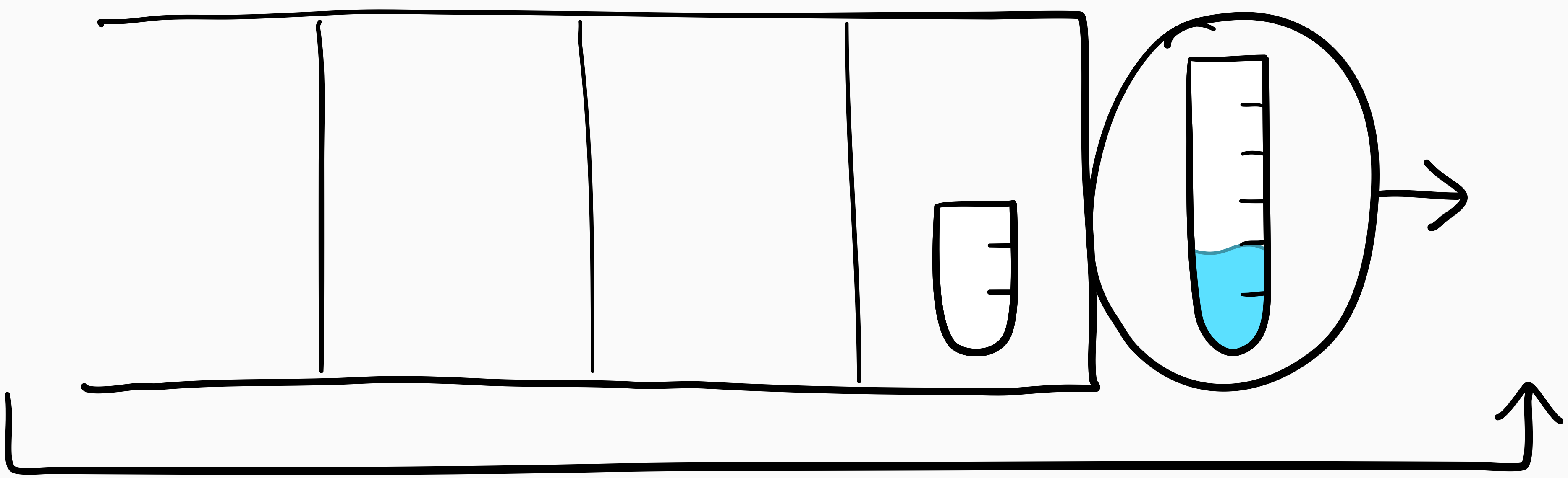
T = response time

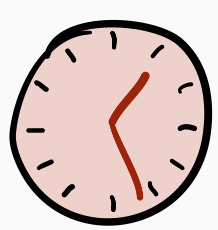
How should we schedule jobs to minimize delay?

$\mathbf{E}[T]$



Serve short jobs before long jobs

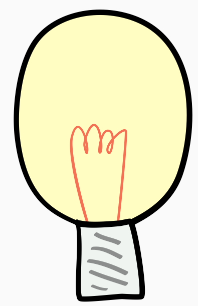




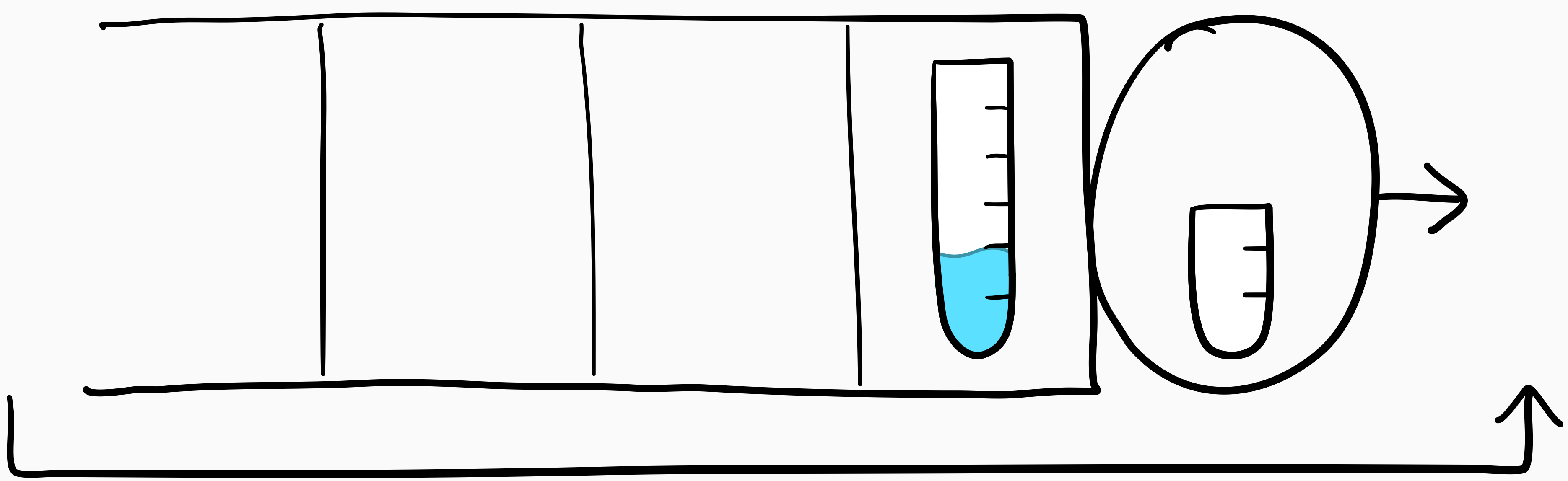
T = response time

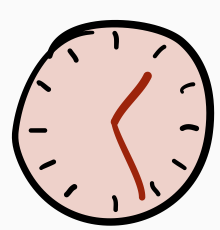
How should we schedule jobs to minimize delay?

$\mathbf{E}[T]$



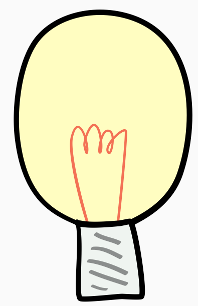
Serve short jobs before long jobs



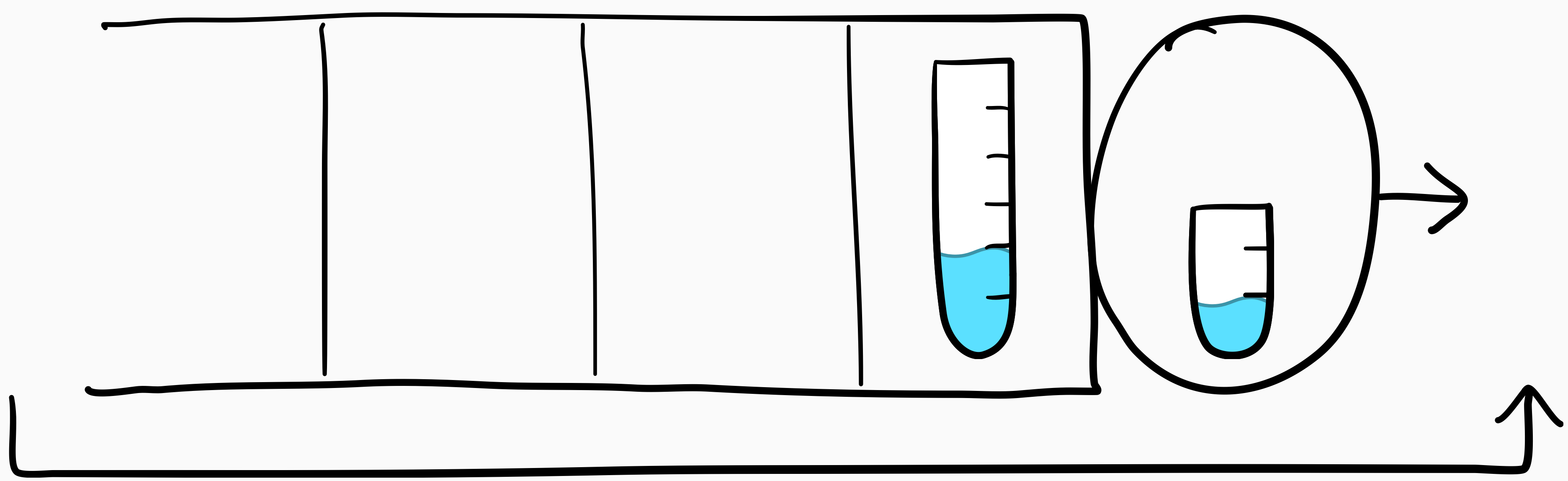


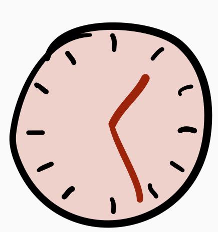
T = response time

How should we schedule jobs to minimize delay?



Serve short jobs
before long jobs

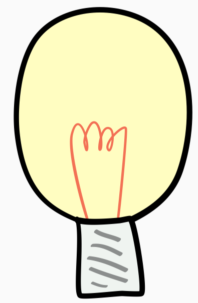




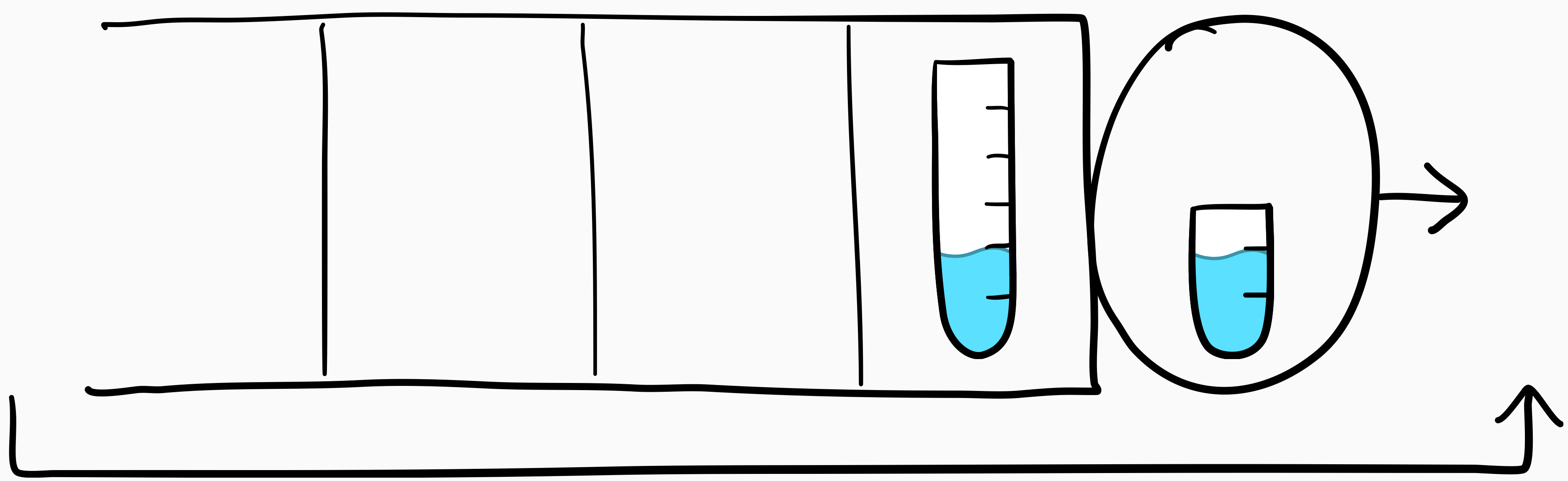
T = response time

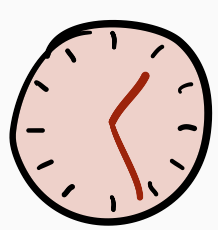
How should we schedule jobs to minimize delay?

$\mathbf{E}[T]$



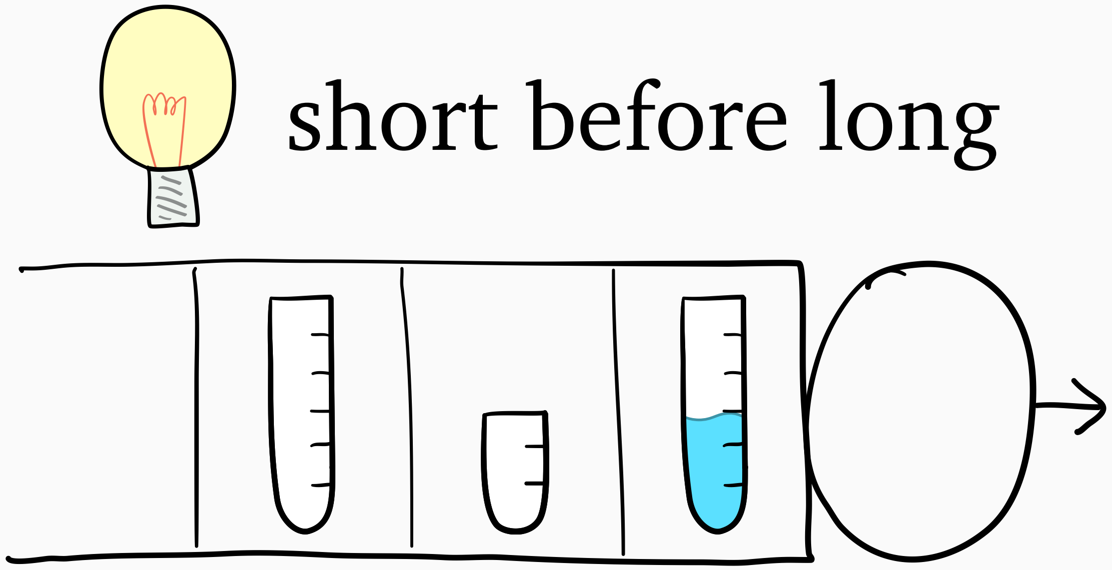
Serve short jobs before long jobs



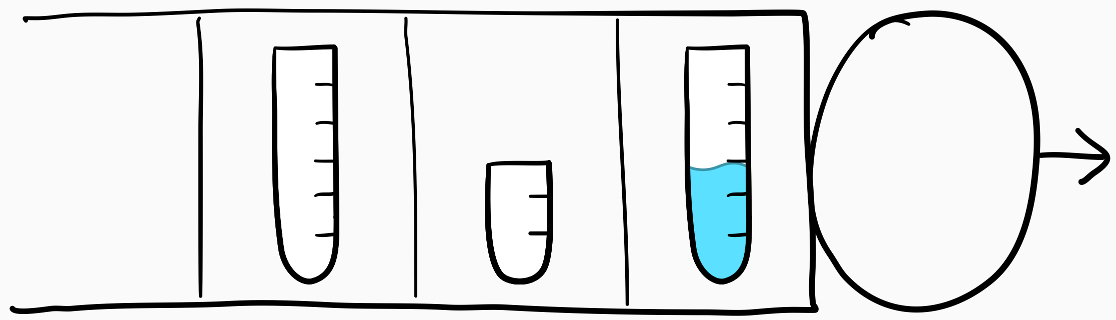


T = response time

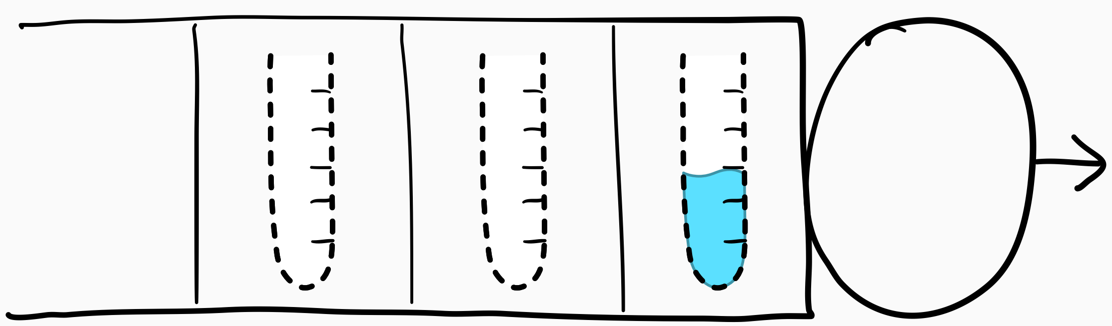
short before long



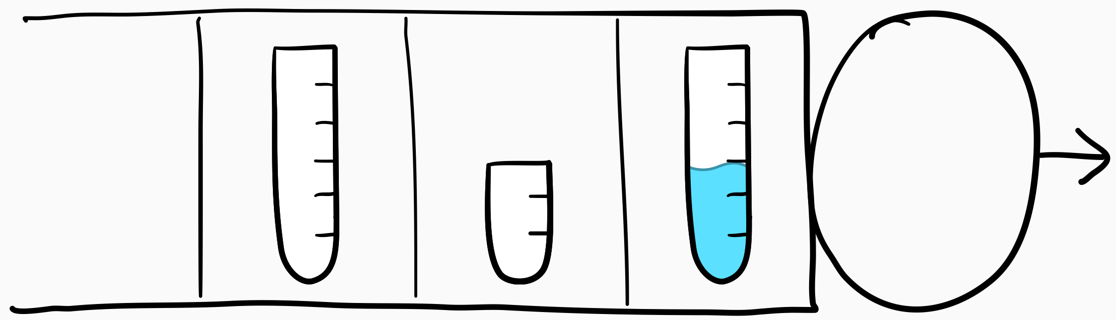
short before long



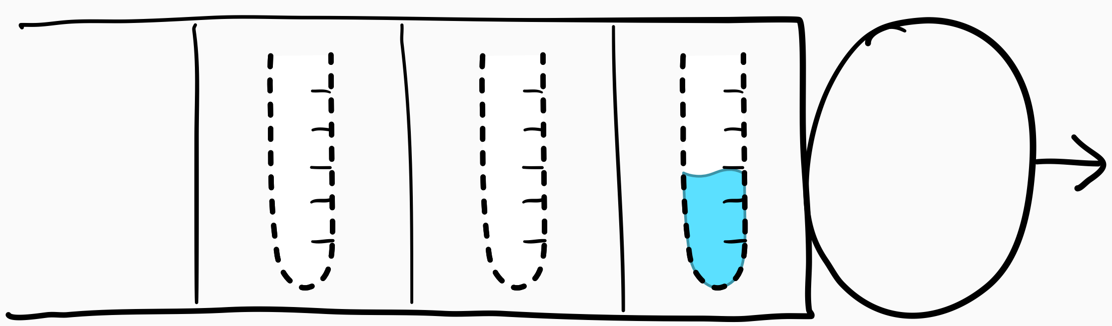
unknown sizes



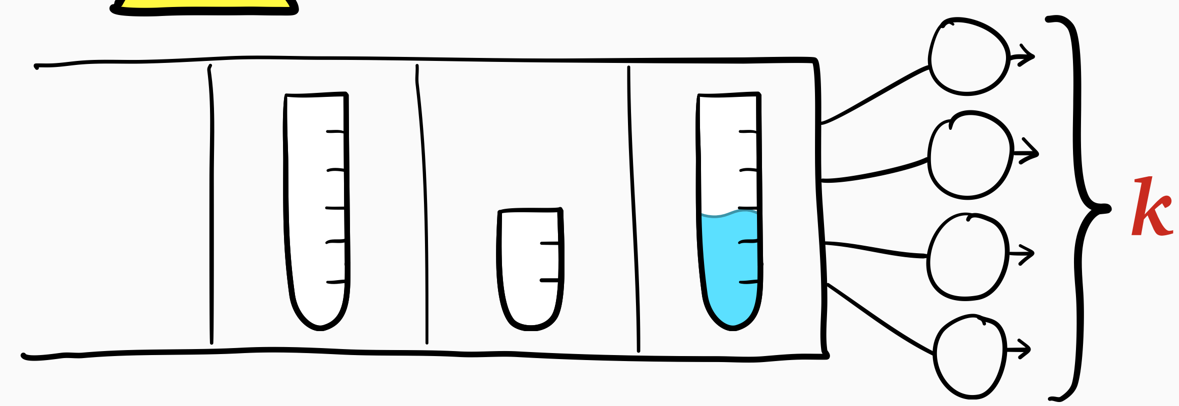
short before long



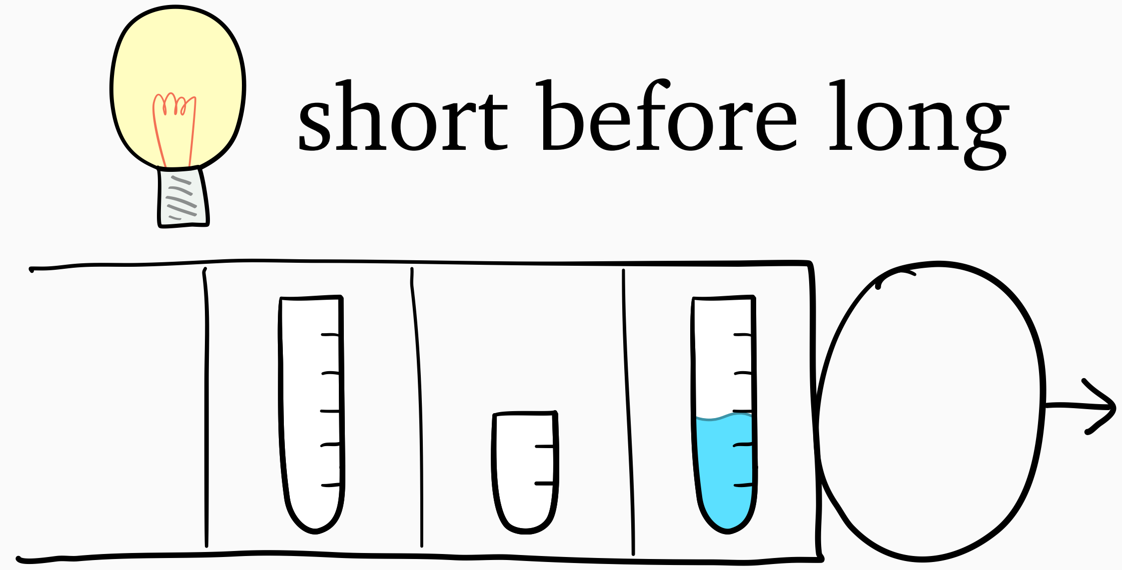
unknown sizes



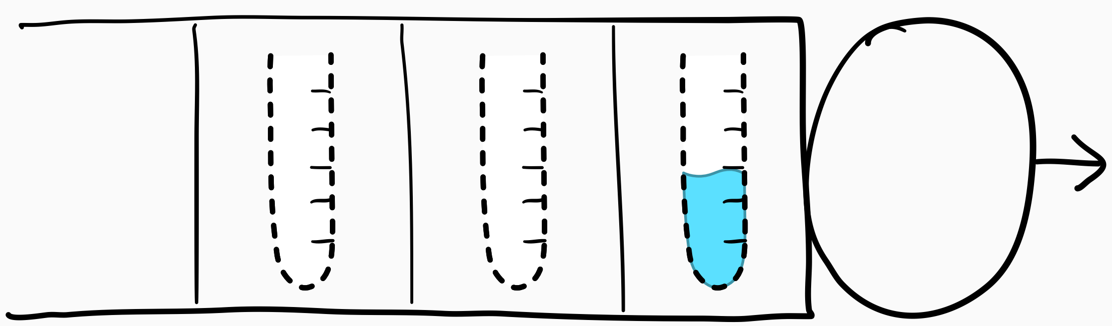
multiple servers



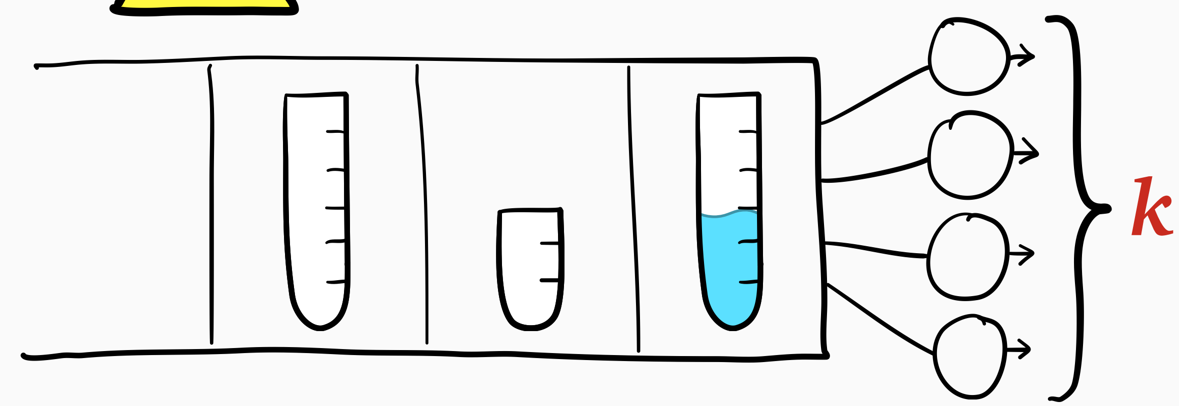
short before long



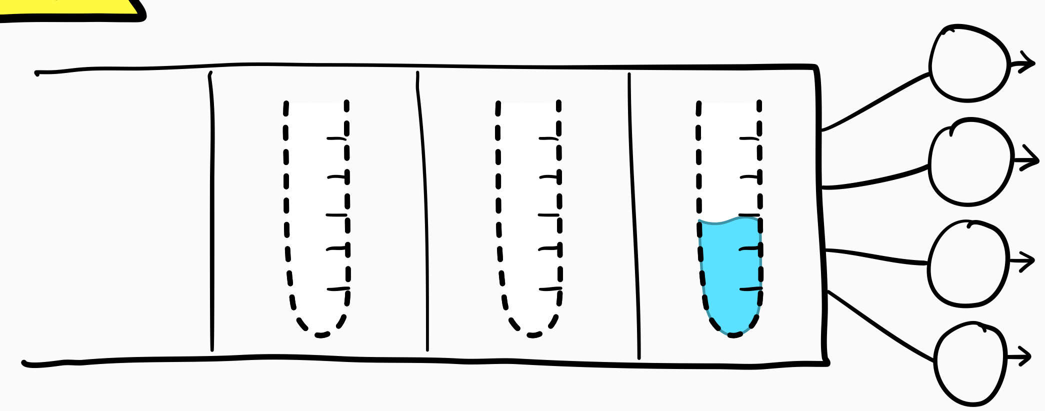
unknown sizes



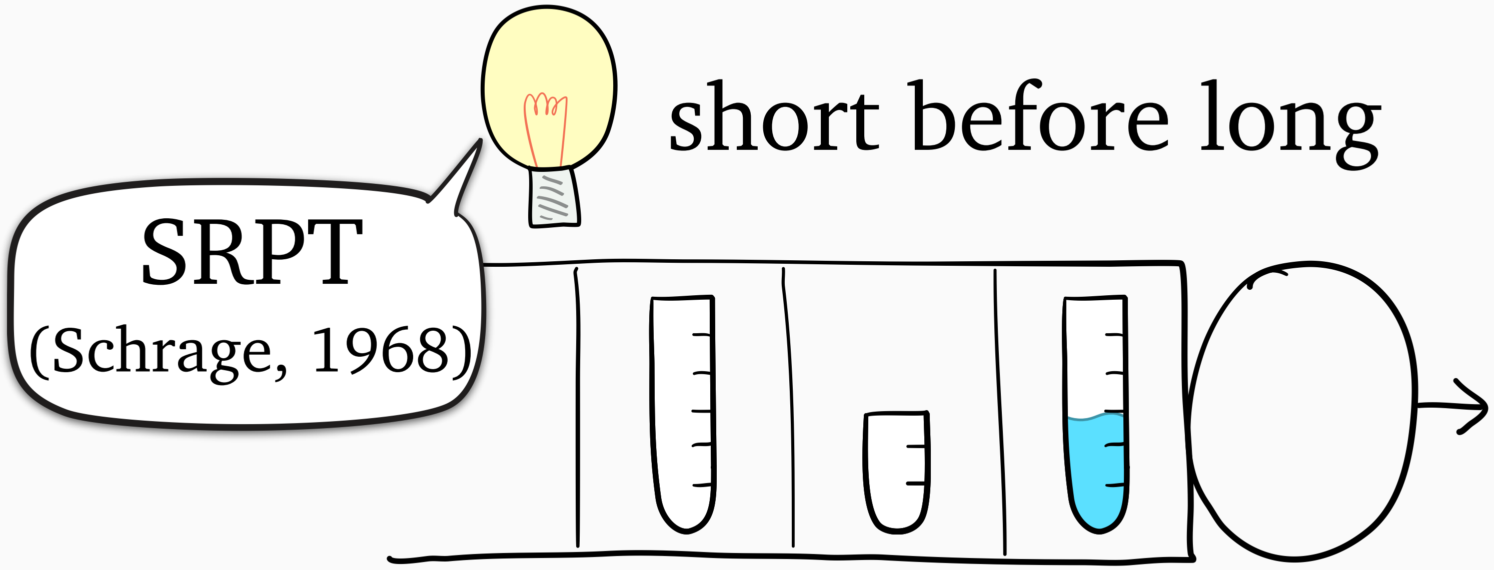
multiple servers



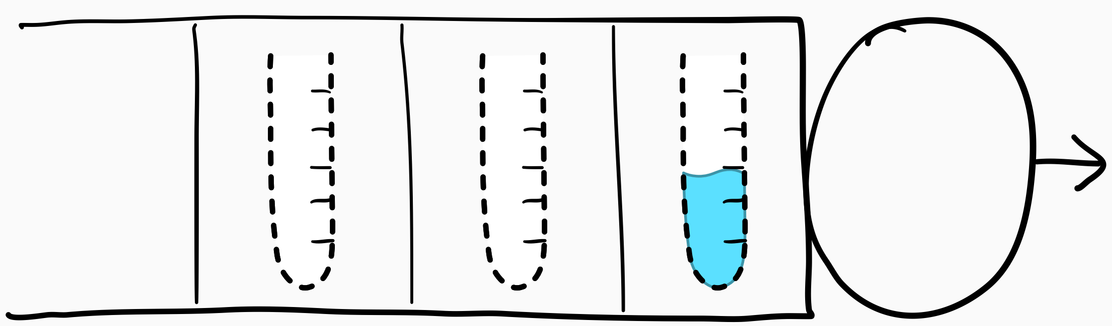
This work: both at once!



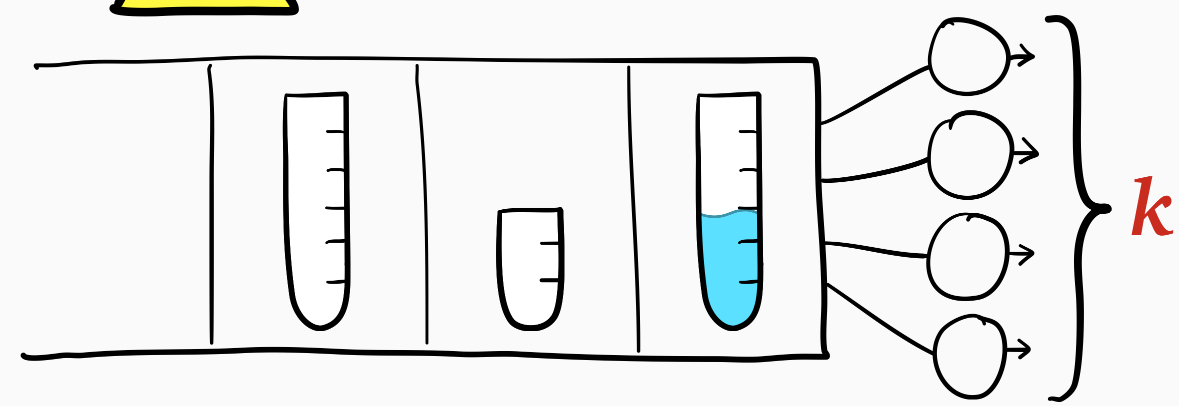
short before long



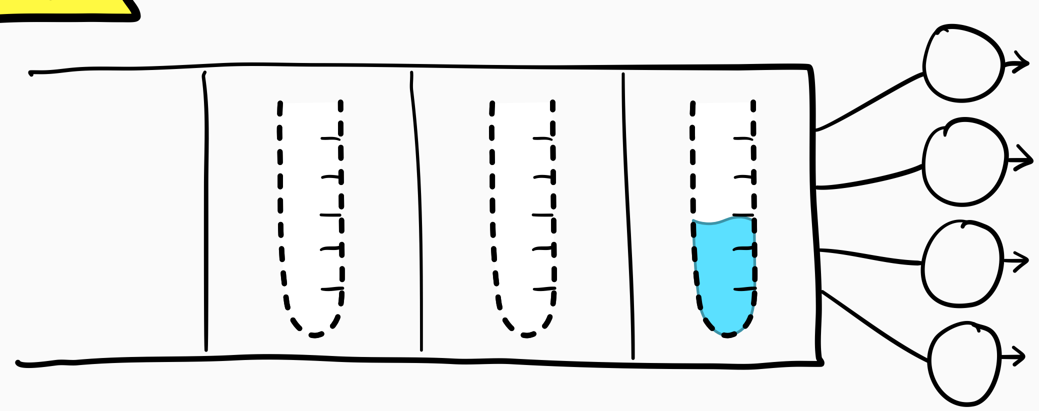
unknown sizes

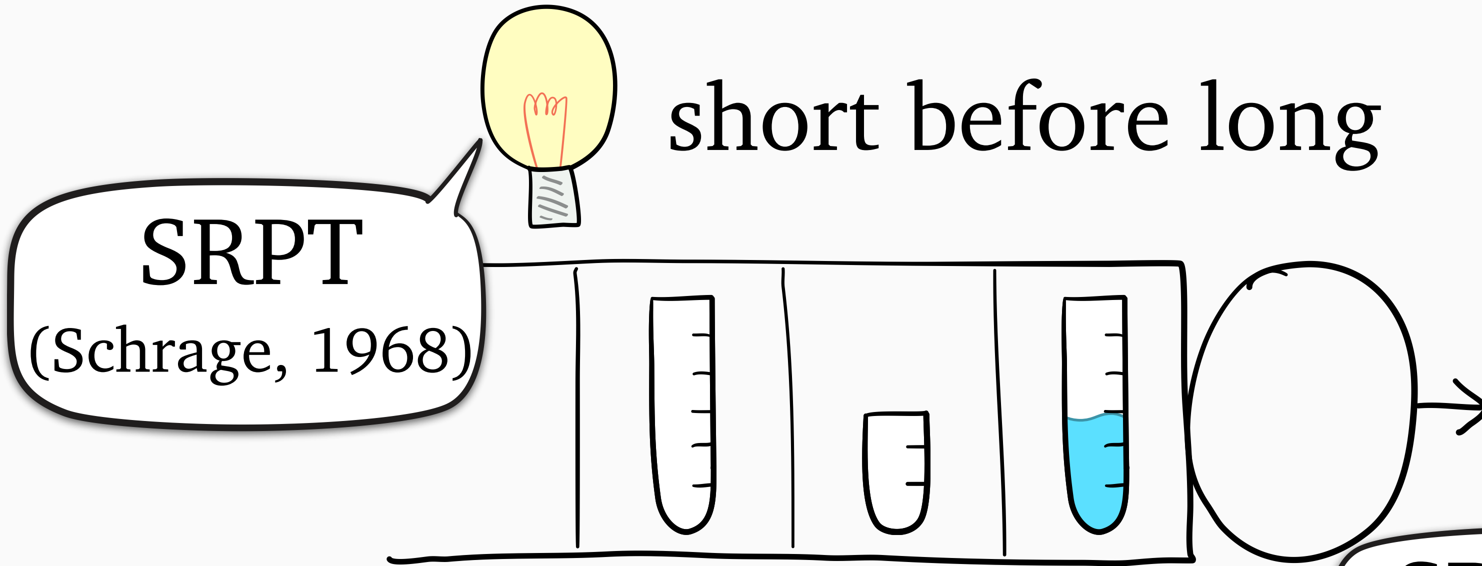


multiple servers

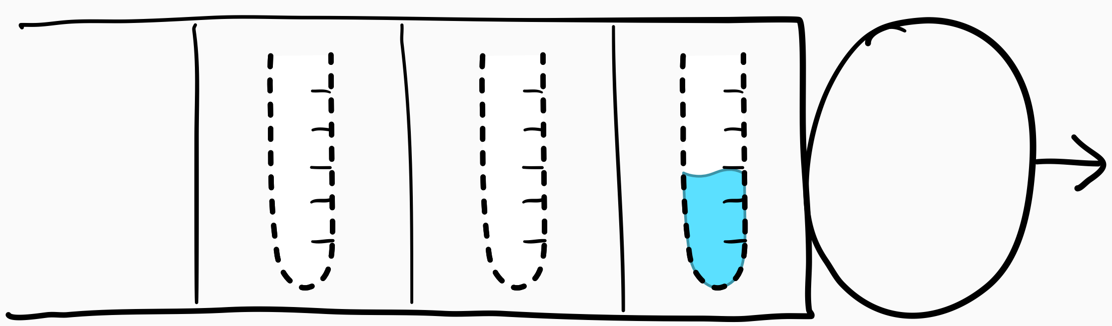


This work: both at once!

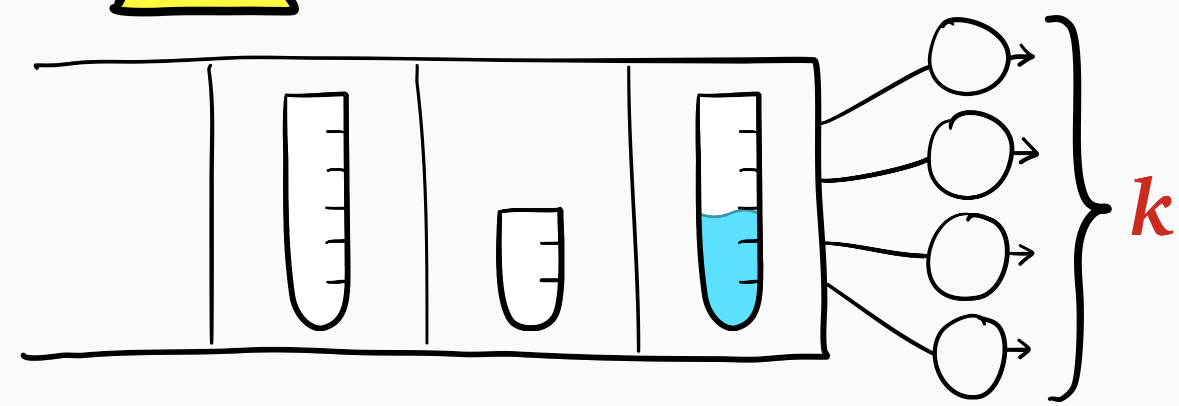




short before long

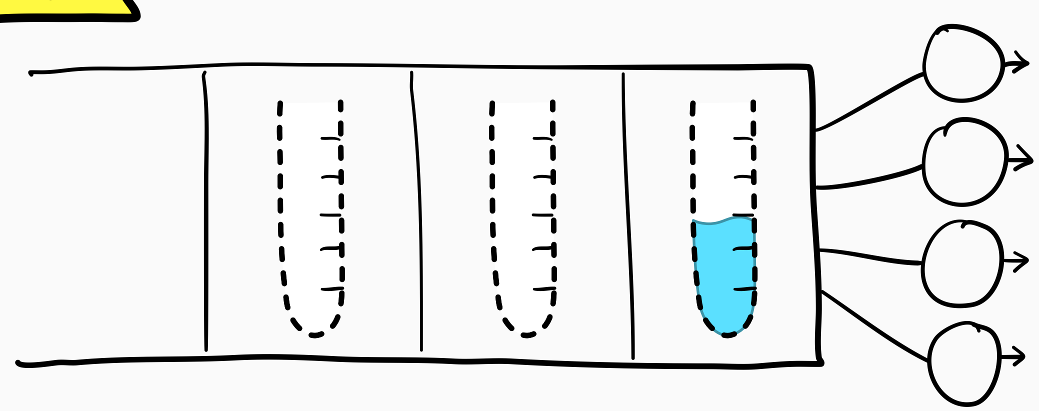


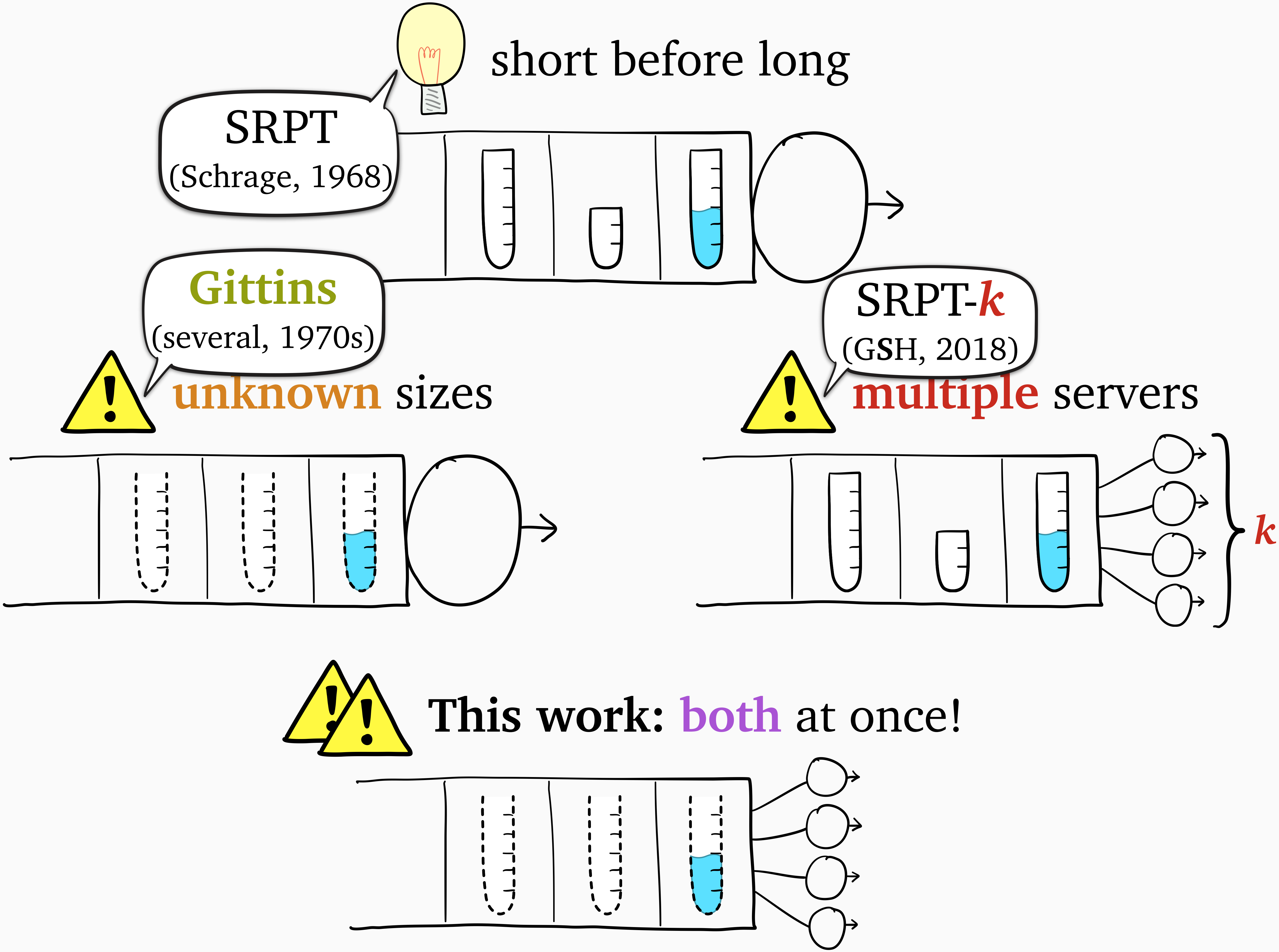
unknown sizes

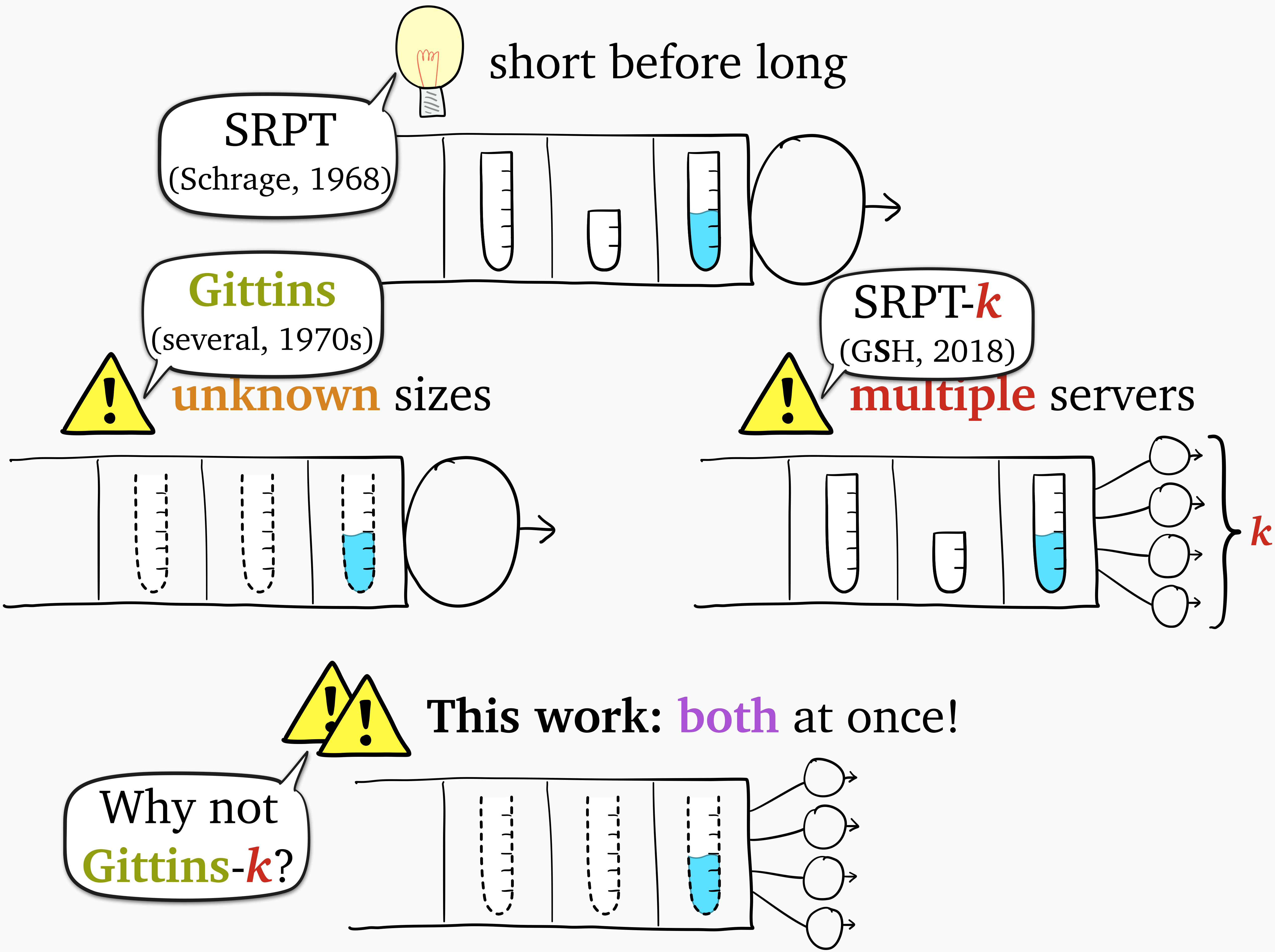


multiple servers

This work: both at once!

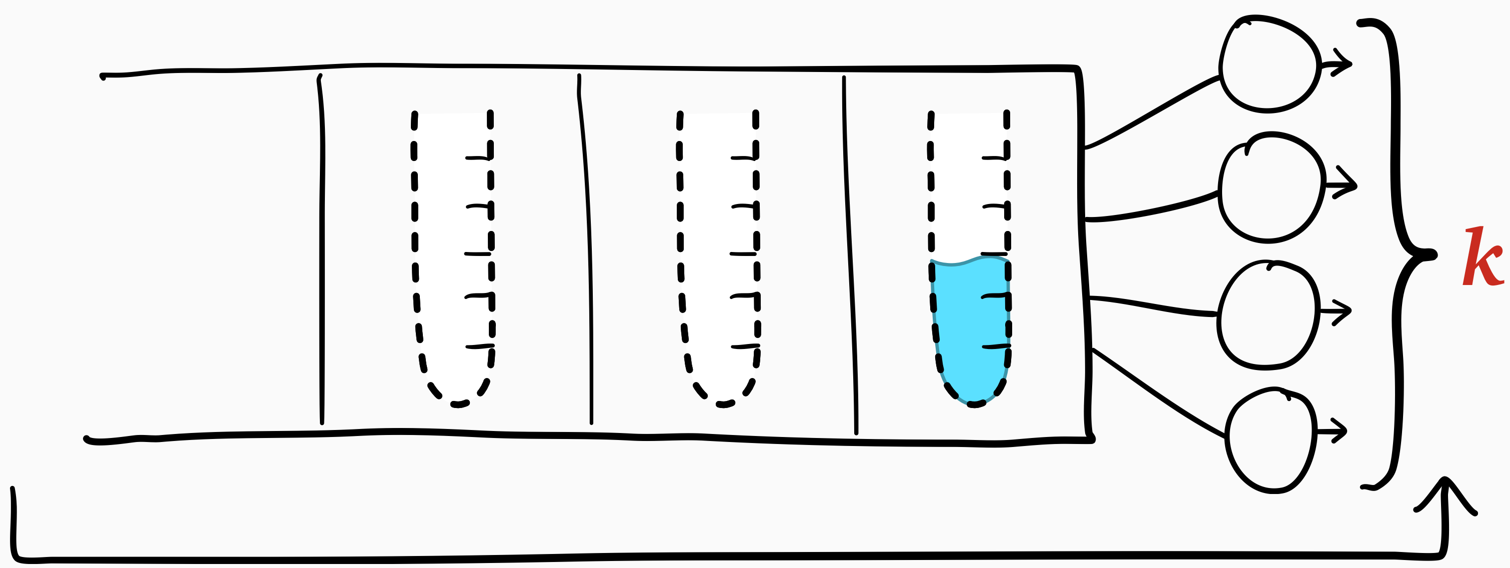


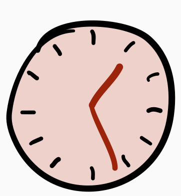

short before long
SRPT
(Schrage, 1968)
Gittins
(several, 1970s)
unknown sizes
SRPT-k
(GSH, 2018)
multiple servers
k
This work: both at once!


short before long
SRPT
(Schrage, 1968)
Gittins
(several, 1970s)
unknown sizes
SRPT-k
(GSH, 2018)
multiple servers
k
This work: both at once!
Why not Gittins-k?

Main result

Theorem: **Gittins**-k has "near-optimal" $\mathbf{E}[T]$ in the M/G/k with **unknown** job sizes

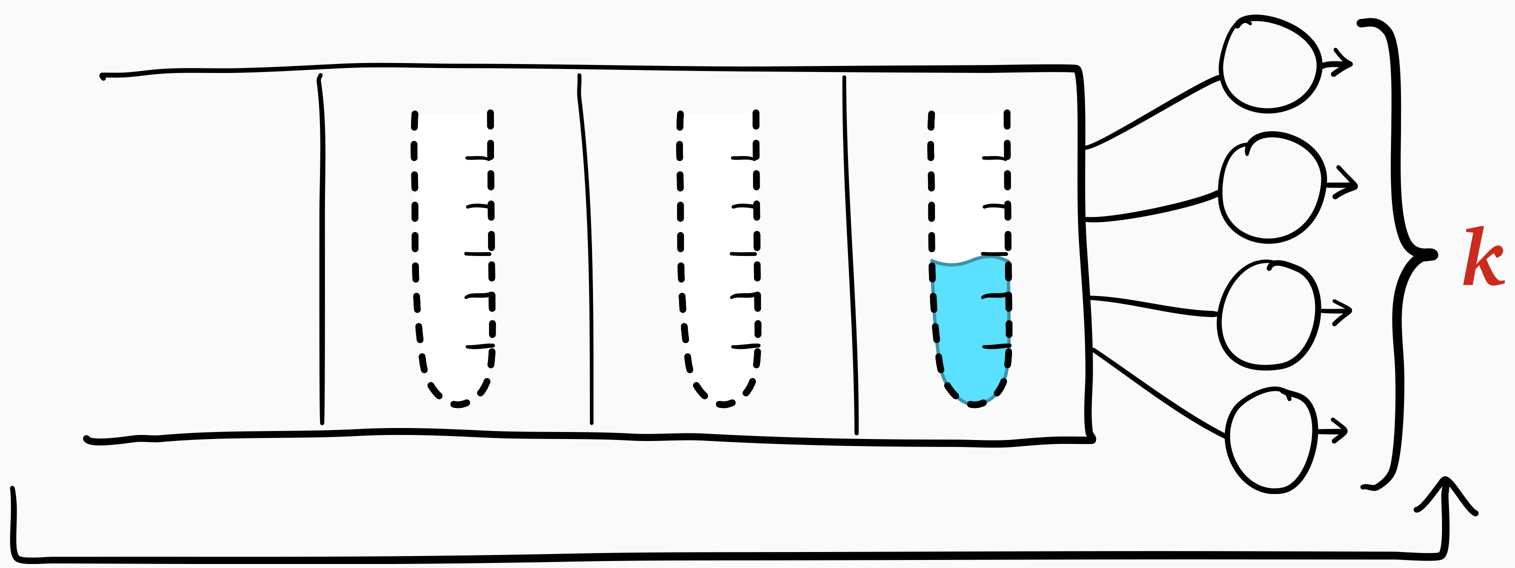


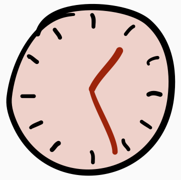

T = response time

Main result

Theorem: **Gittins**-$\boldsymbol{k}$ has "near-optimal" $\mathbf{E}[T]$ in the M/G/$\boldsymbol{k}$ with **unknown** job sizes

$$\mathbf{E}[T_{\text{Gittins-}k}] \leq \mathbf{E}[T_{\text{Opt-}k}] + \text{"small"}$$

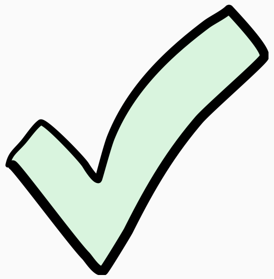



T = response time

Main result

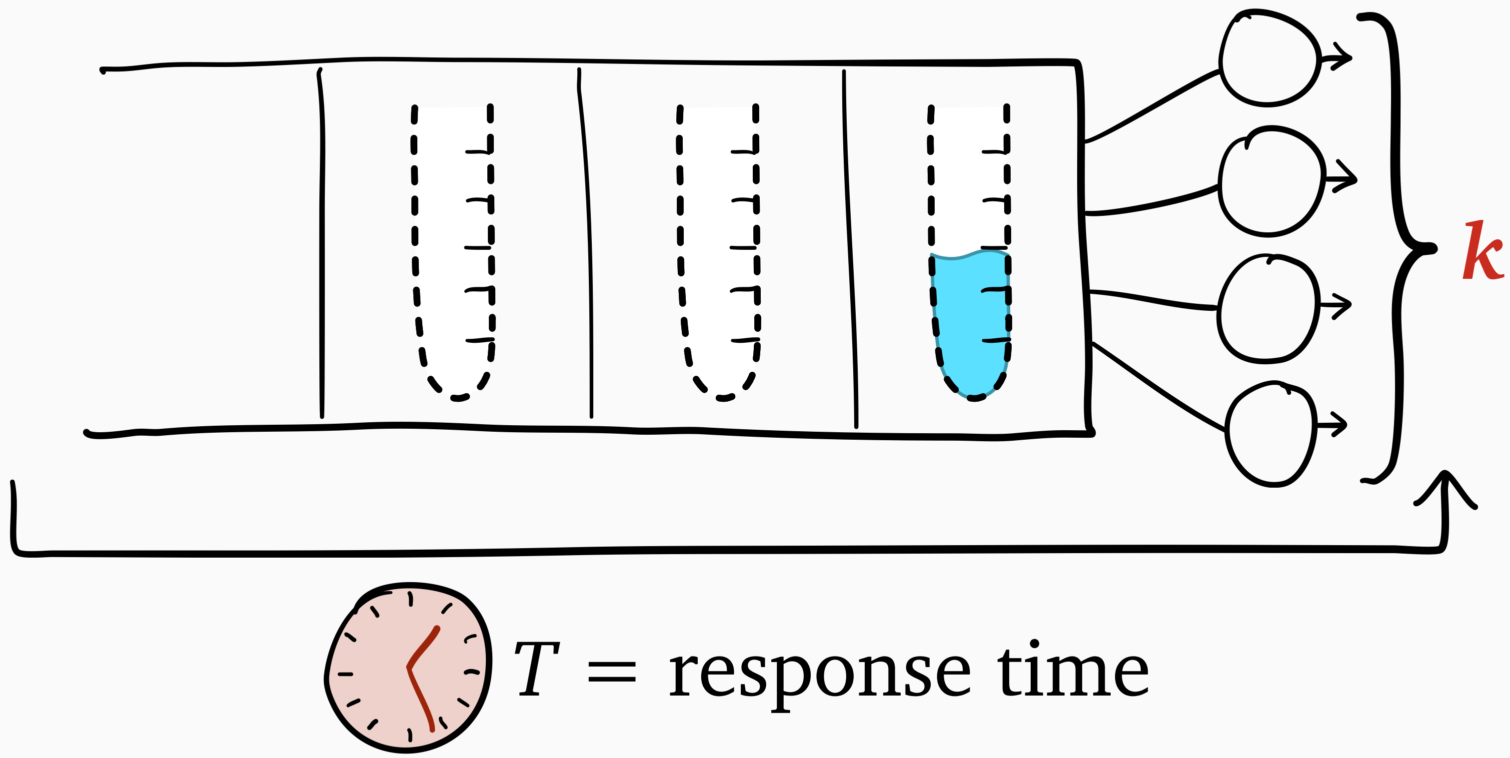
Theorem: **Gittins**-k has "near-optimal" $\mathbf{E}[T]$ in the M/G/k with **unknown** job sizes

$$\mathbf{E}[T_{\text{Gittins-}k}] \le \mathbf{E}[T_{\text{Opt-}k}] + \text{"small"}$$



Q: How to schedule with k servers?

A: Use **Gittins**



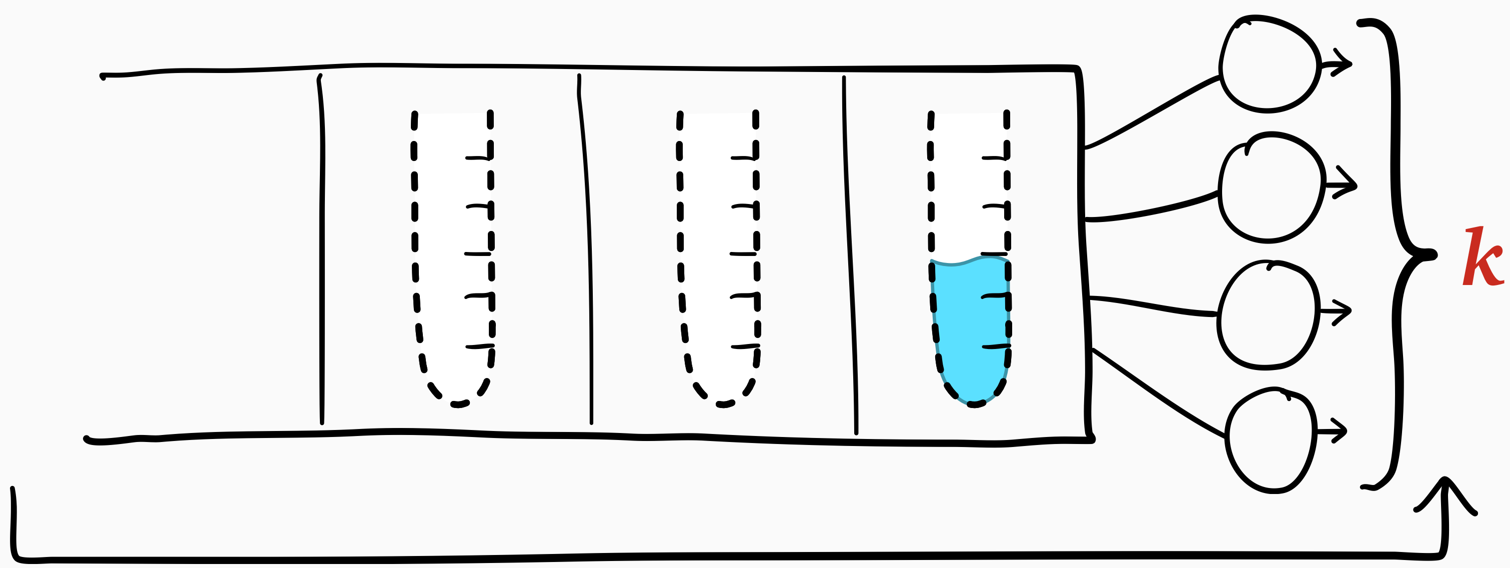
Main result

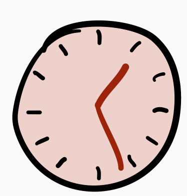
Theorem: Gittins-k has "near-optimal" $\mathbf{E}[T]$ in the M/G/k with **unknown** job sizes

$$\mathbf{E}[T_{\text{Gittins-}k}] \leq \mathbf{E}[T_{\text{Opt-}k}] + \text{“small”}$$

Q: How to schedule with k servers?

A: Use **Gittins**



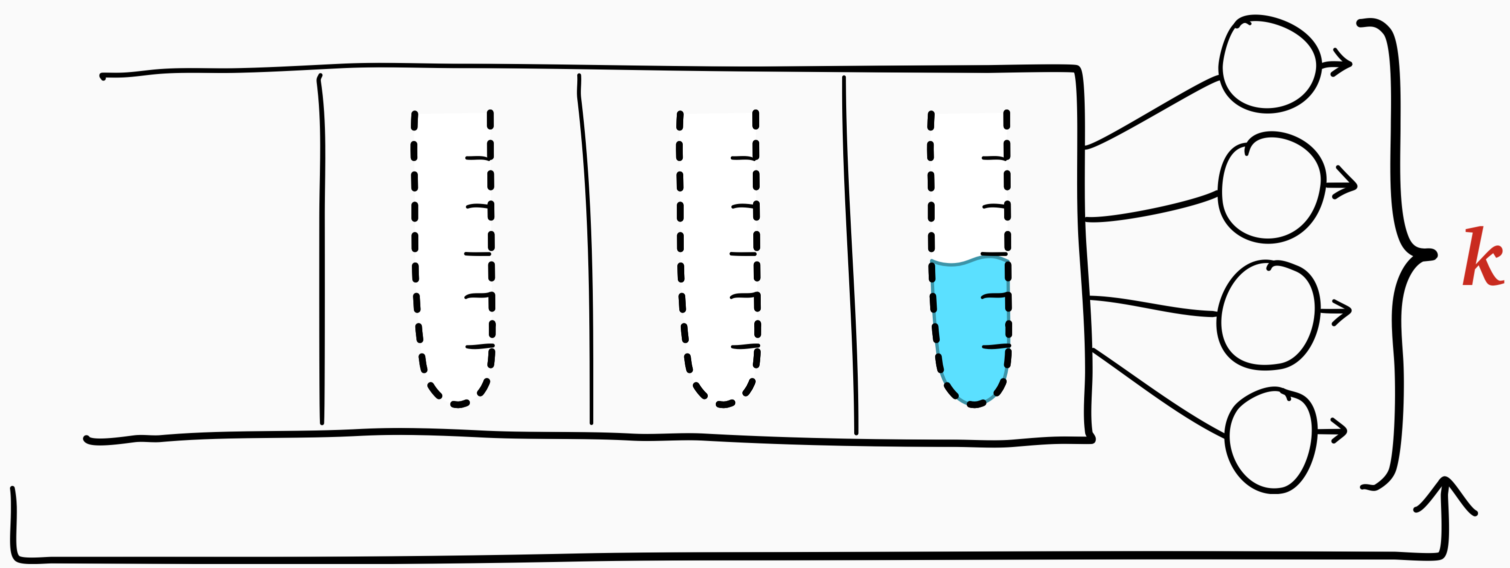

T = response time

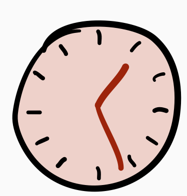
Main result

Theorem: Gittins-k has "near-optimal" $\mathbf{E}[T]$ in the M/G/k with **unknown** job sizes

$$\mathbf{E}[T_{\text{Gittins-}k}] \leq \mathbf{E}[T_{\text{Opt-}k}] + \text{"small"}$$

Q: How to schedule with k servers?
A: Use **Gittins**



 T = response time

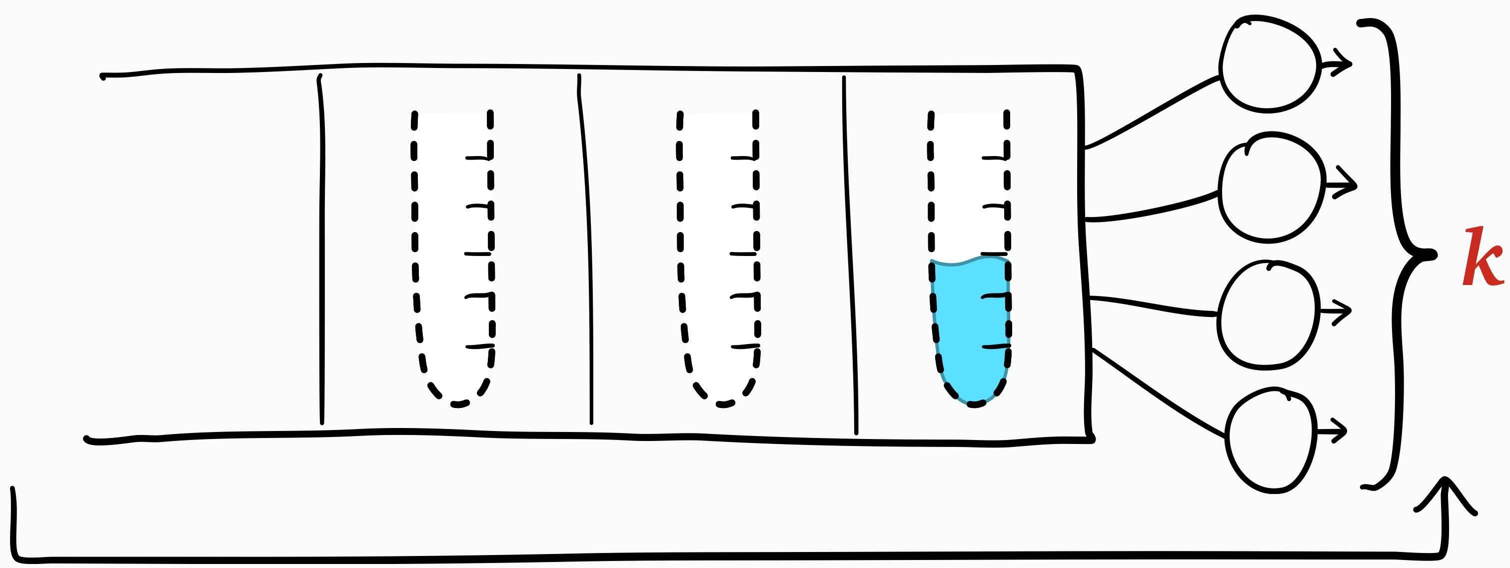
Main result

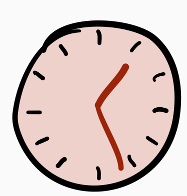
Theorem: Gittins-k has "near-optimal" $\mathbf{E}[T]$ in the M/G/k with **unknown** job sizes

$$\mathbf{E}[T_{\text{Gittins-}k}] \le \mathbf{E}[T_{\text{Opt-}k}] + \text{“small”}$$

Q: How to schedule with k servers?

A: Use **Gittins**



 T = response time

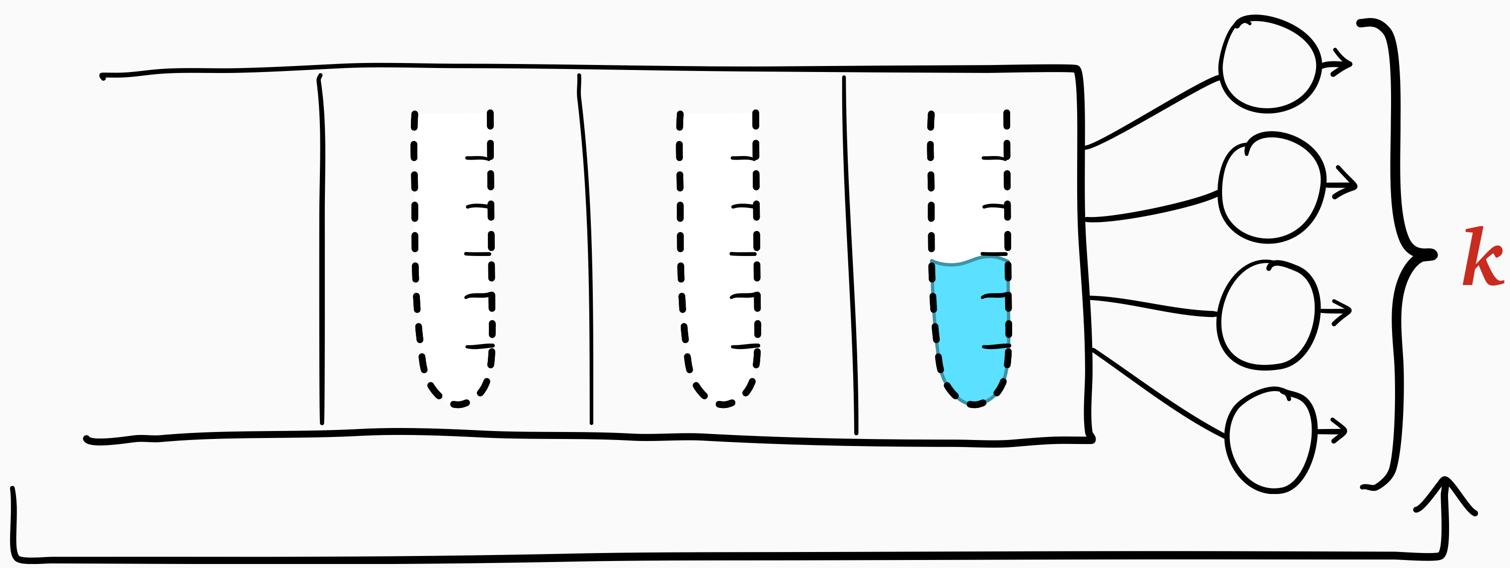
Main result

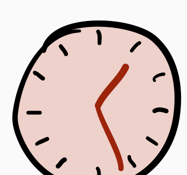
Theorem: **Gittins**-***k*** has "near-optimal" $\mathbf{E}[T]$ in the M/G/***k*** with **unknown** job sizes

$$\mathbf{E}[T_{\text{Gittins-}k}] \le \mathbf{E}[T_{\text{Opt-}k}] + \text{“small”}$$

Q: How to schedule with ***k*** servers?

A: Use **Gittins**





T = response time

Main result

Theorem: **Gittins**-***k*** has "near-optimal" $\mathbf{E}[T]$ in the M/G/***k*** with **unknown** job sizes

$$\mathbf{E}[T_{\text{Gittins-}k}] \le \mathbf{E}[T_{\text{Opt-}k}] + \text{"small"}$$

Q: How to schedule with ***k*** servers?

A: Use **Gittins**

Main result

Theorem: **Gittins**-***k*** has "near-optimal" $\mathbf{E}[T]$ in the M/G/***k*** with **unknown** job sizes

$$\mathbf{E}[T_{\text{Gittins-}k}] \le \mathbf{E}[T_{\text{Opt-}k}] + \text{“small”}$$

Q: How to schedule with ***k*** servers?
A: Use **Gittins**

Q: How do we analyze $\mathbf{E}[T]$?

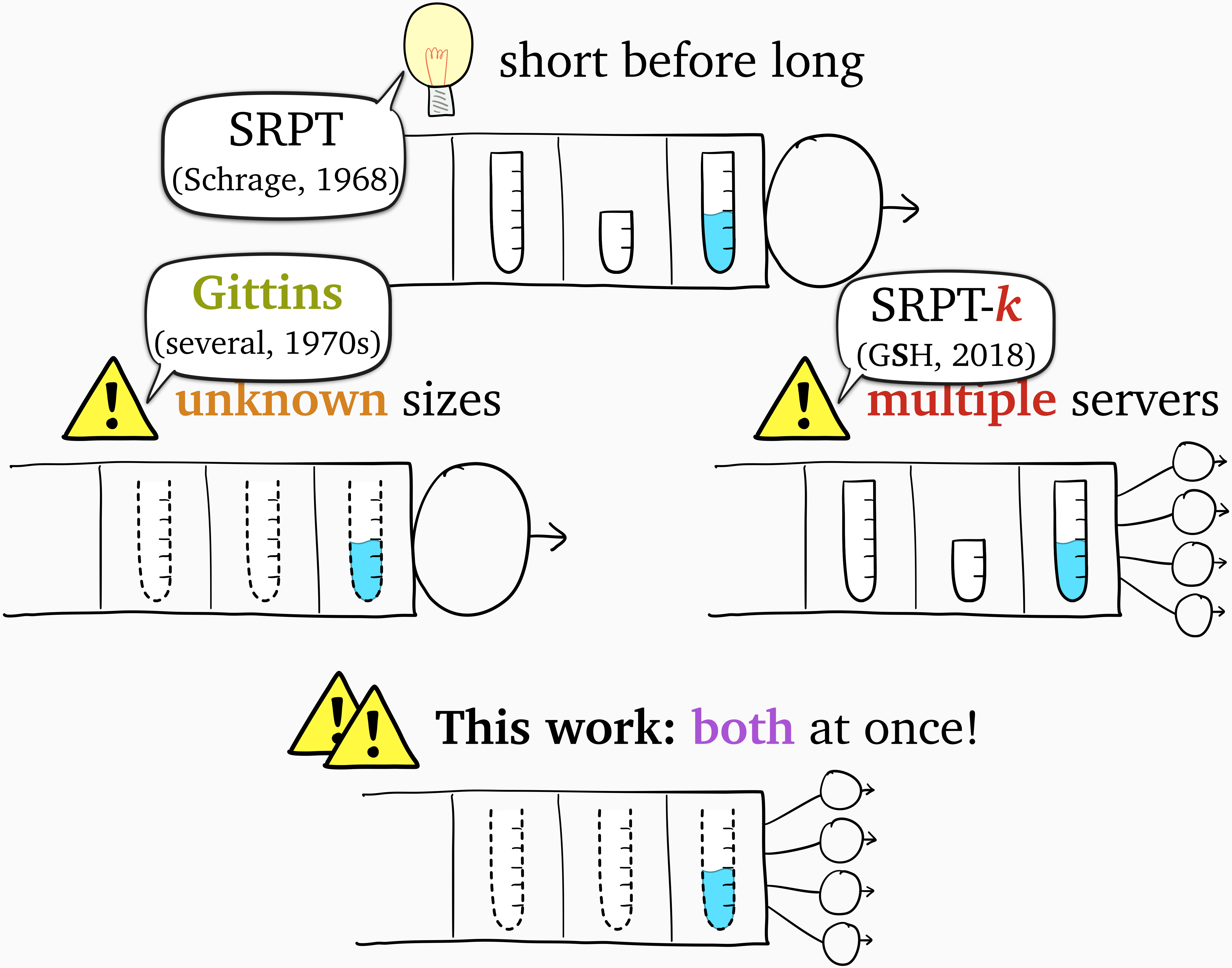
Main result

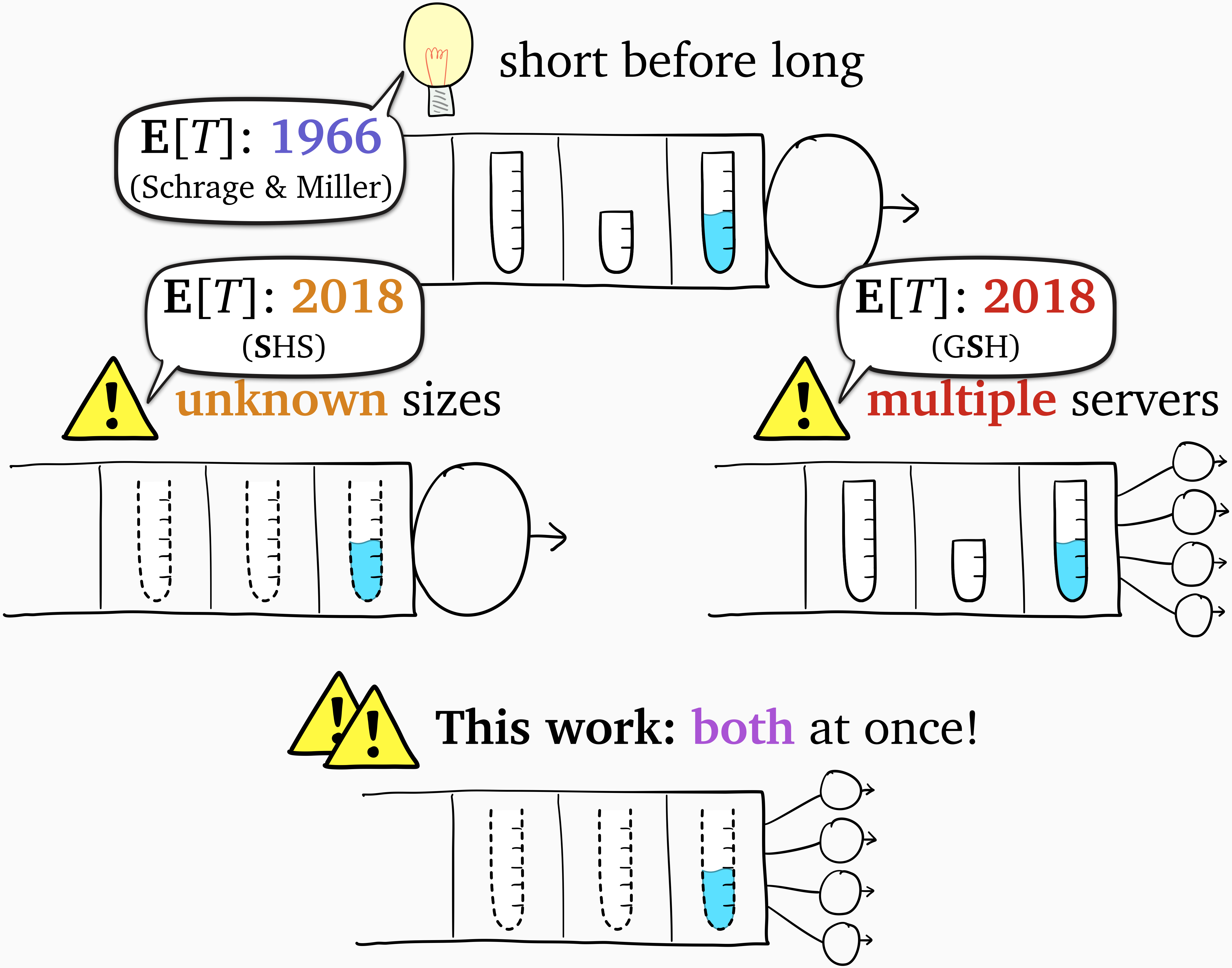
Theorem: **Gittins**-$\boldsymbol{k}$ has "near-optimal" $\mathbf{E}[T]$ in the M/G/$\boldsymbol{k}$ with **unknown** job sizes

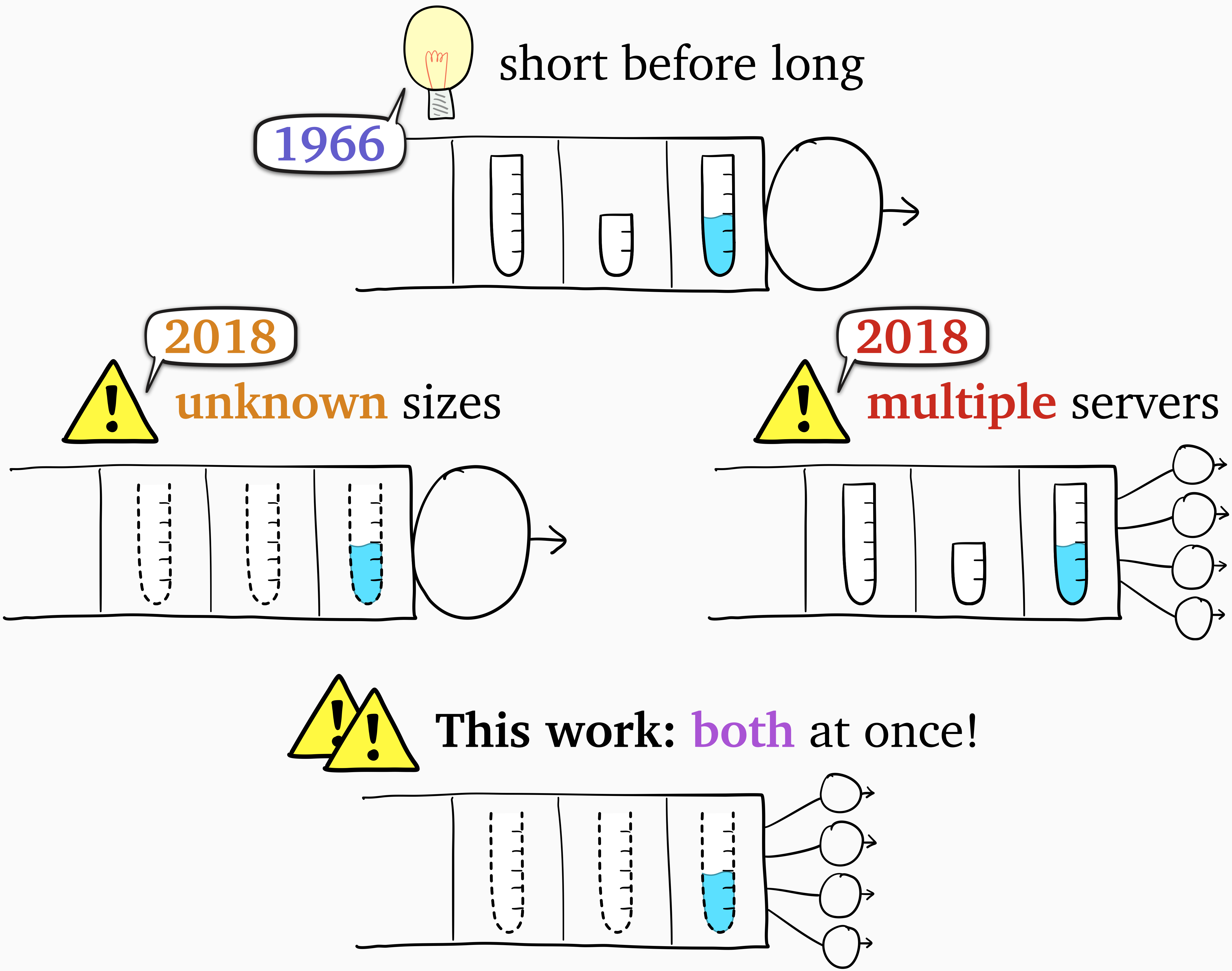
$$\mathbf{E}[T_{\text{Gittins-}k}] \leq \mathbf{E}[T_{\text{Opt-}k}] + \text{"small"}$$

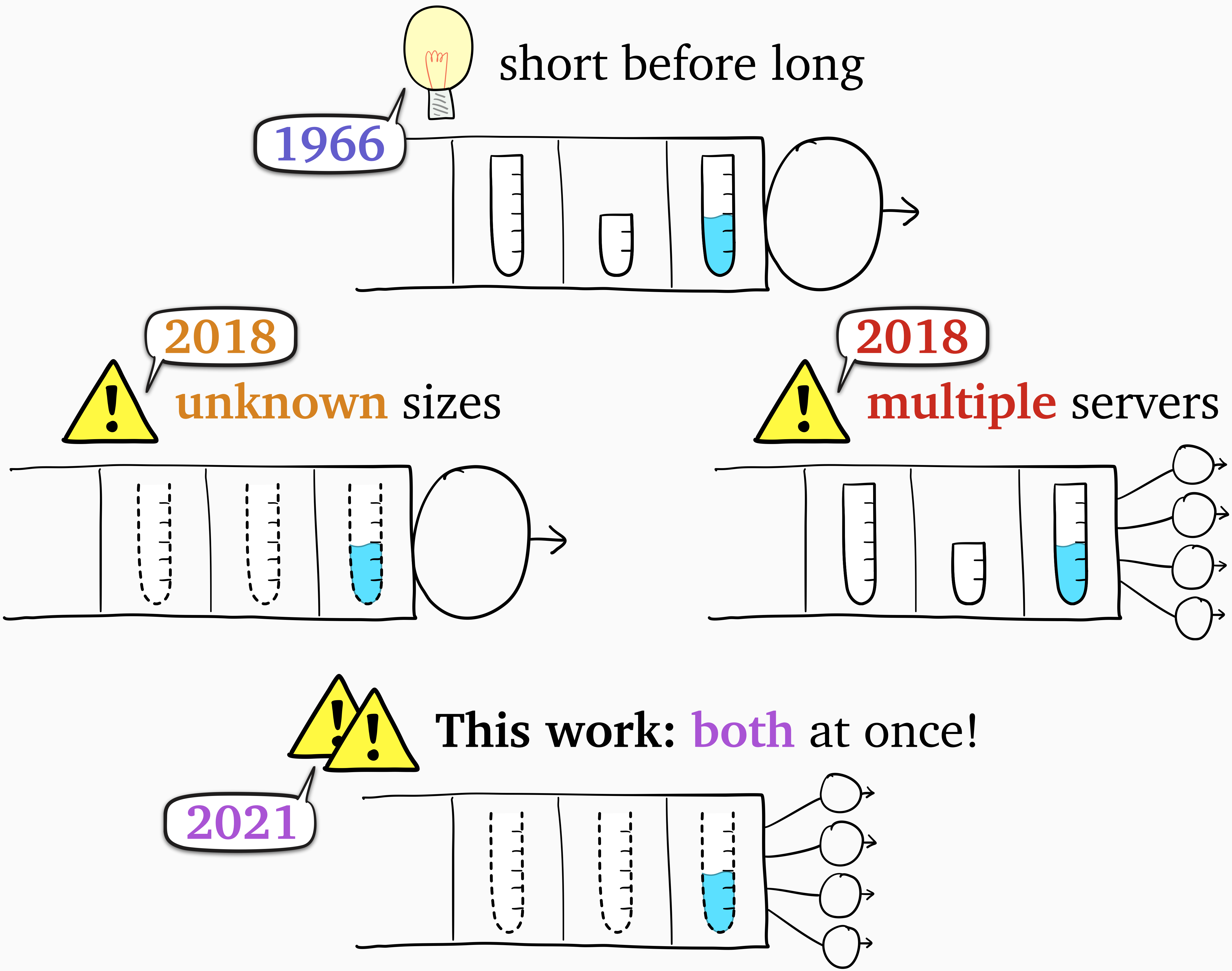
Q: How to schedule with $\boldsymbol{k}$ servers?
A: Use **Gittins**

Q: How do we analyze $\mathbf{E}[T]$?
Q: Why is our approach significant?

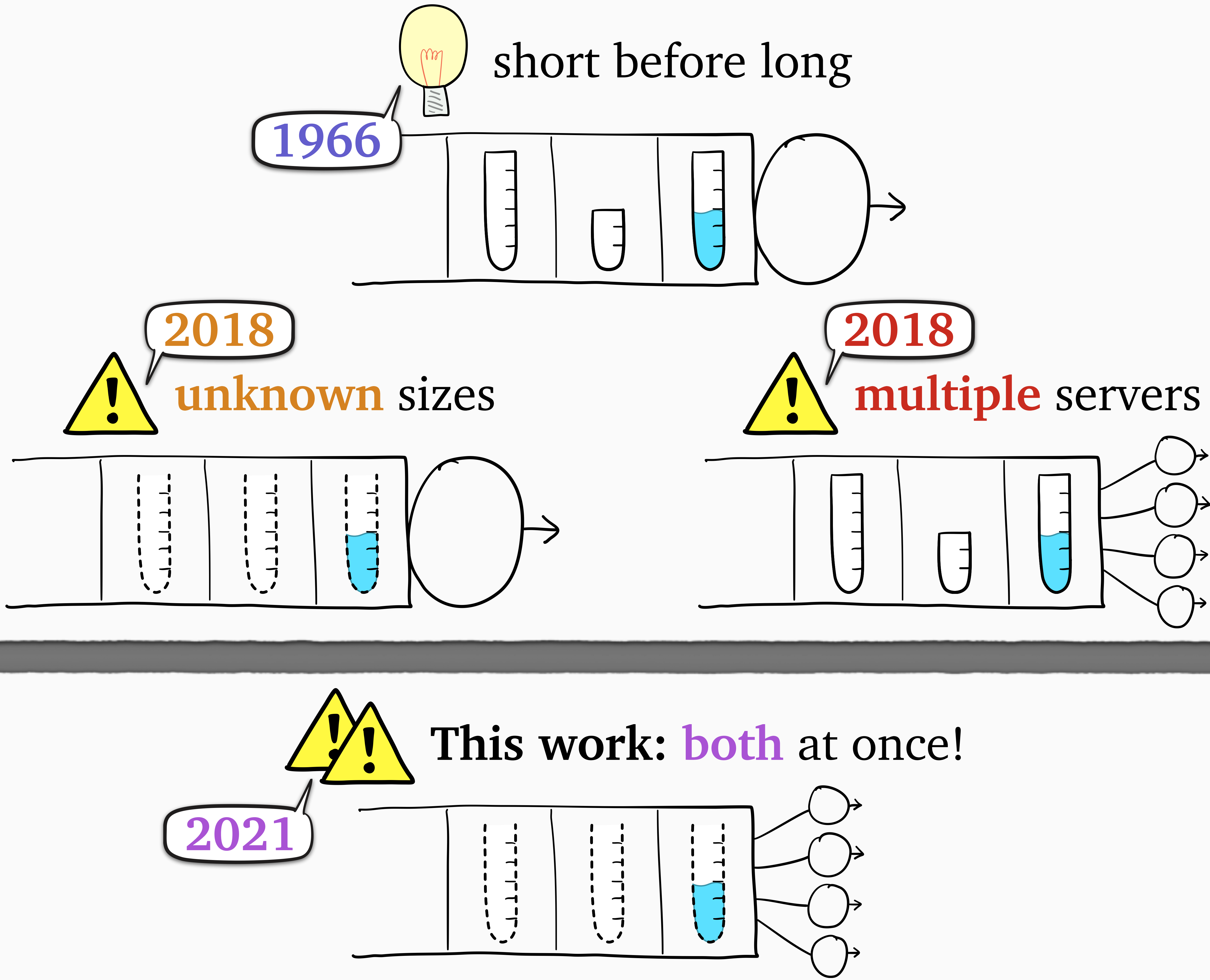

short before long
SRPT
(Schrage, 1968)
Gittins
(several, 1970s)
unknown sizes
SRPT-k
(GSH, 2018)
multiple servers
This work: both at once!

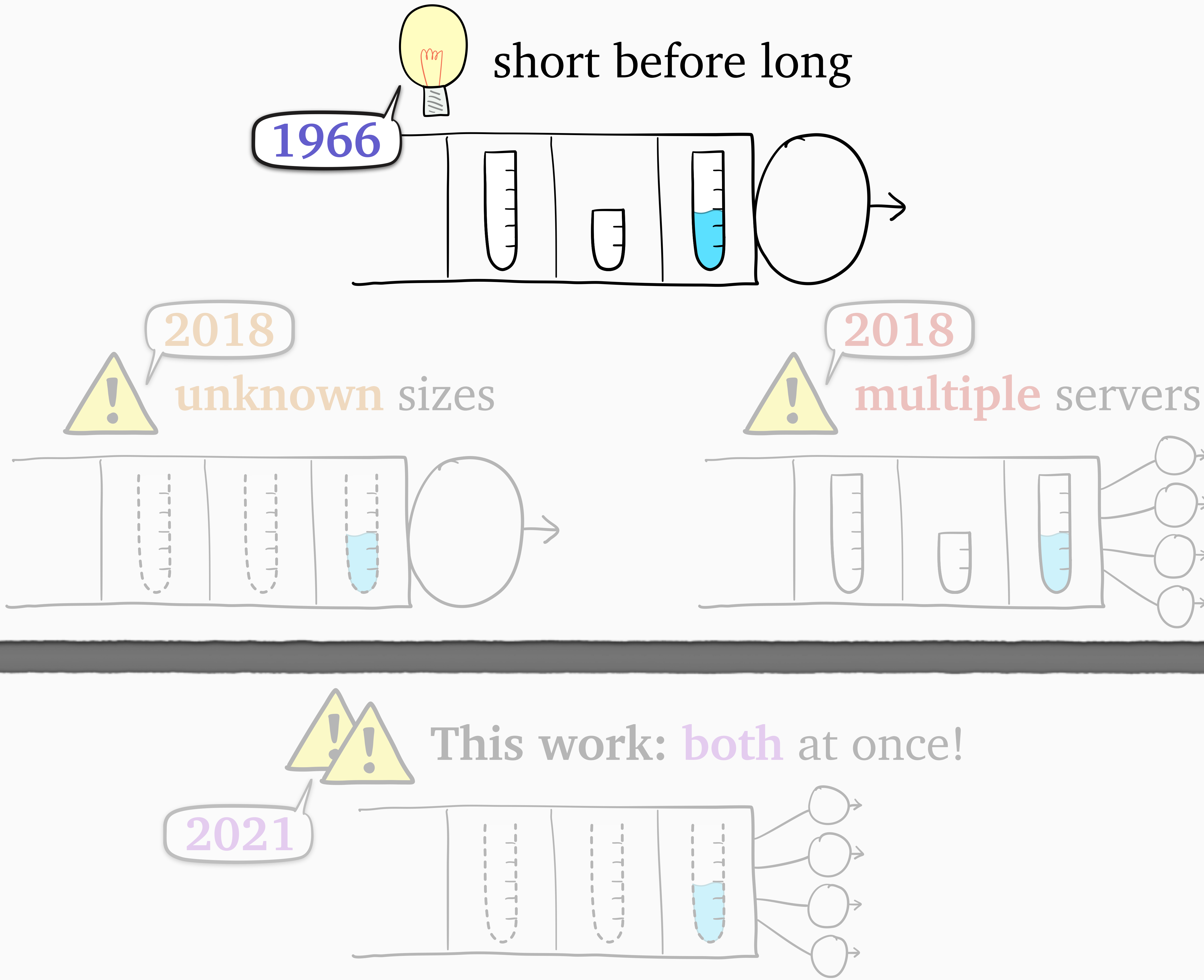

short before long
E[T]: 1966
(Schrage & Miller)
E[T]: 2018
(SHS)
unknown sizes
E[T]: 2018
(GSH)
multiple servers
This work: both at once!


short before long
1966
2018
unknown sizes
2018
multiple servers
This work: both at once!

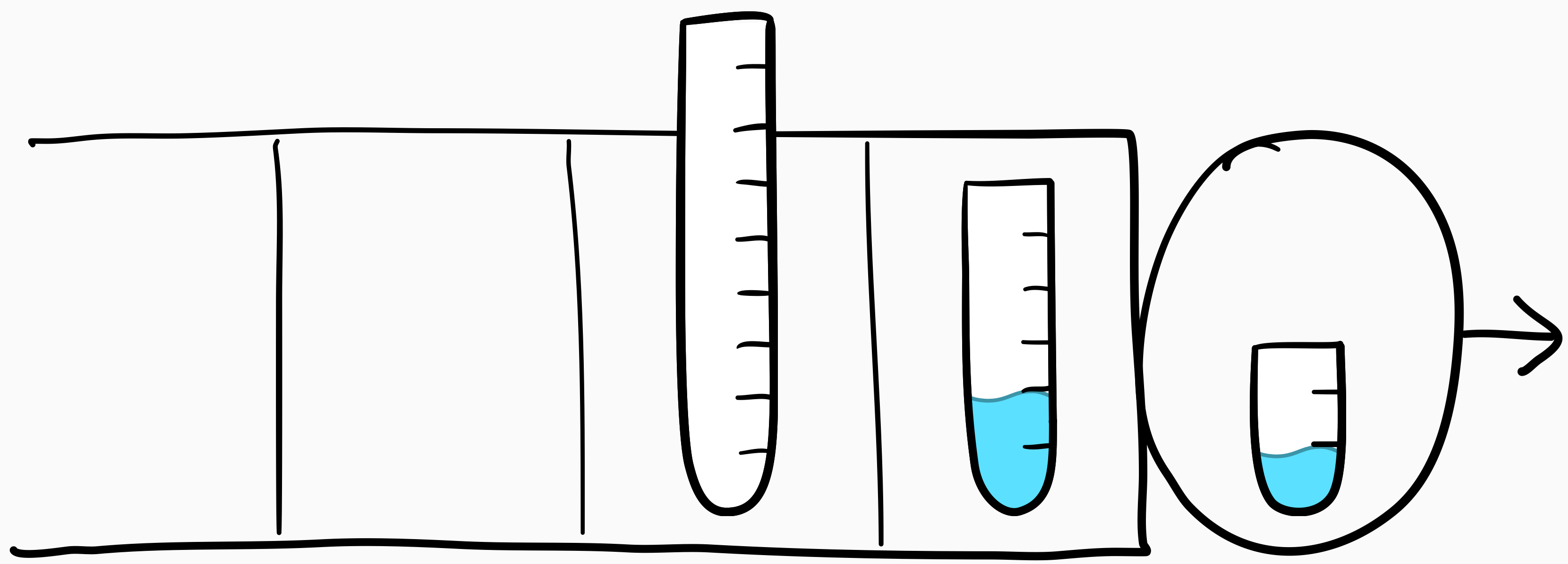


short before long
1966
2018
unknown sizes
2018
multiple servers
This work: both at once!
2021

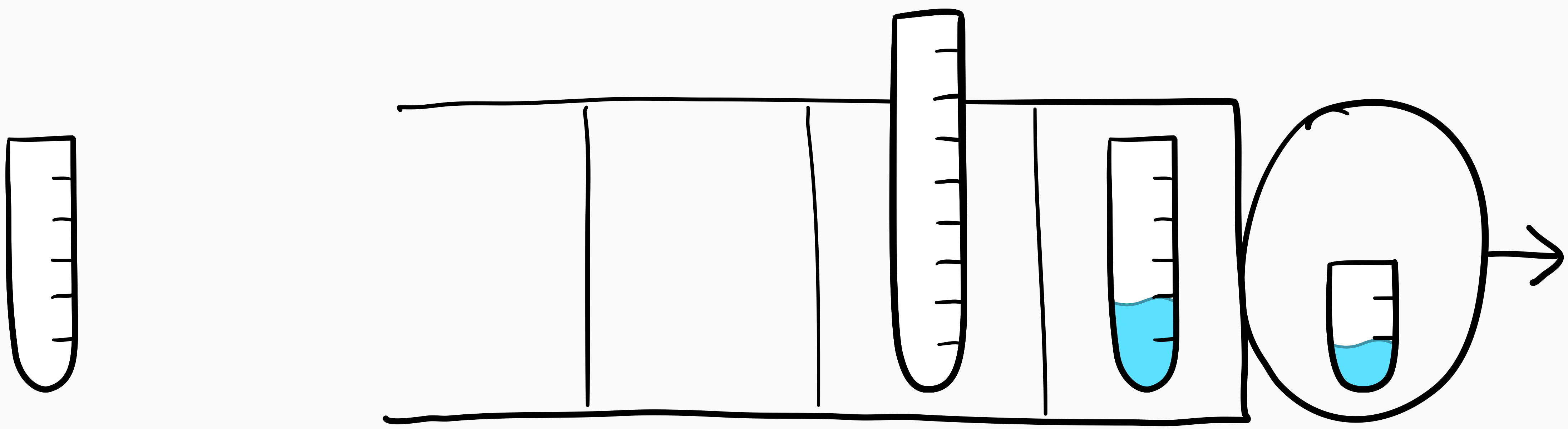

short before long
1966
2018
unknown sizes
2018
multiple servers
This work: both at once!
2021


short before long
1966
2018
unknown sizes
2018
multiple servers
This work: both at once!
2021

$\mathbf{E}[T]$ of SRPT

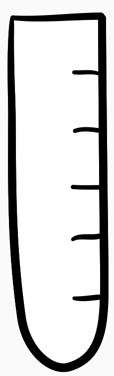


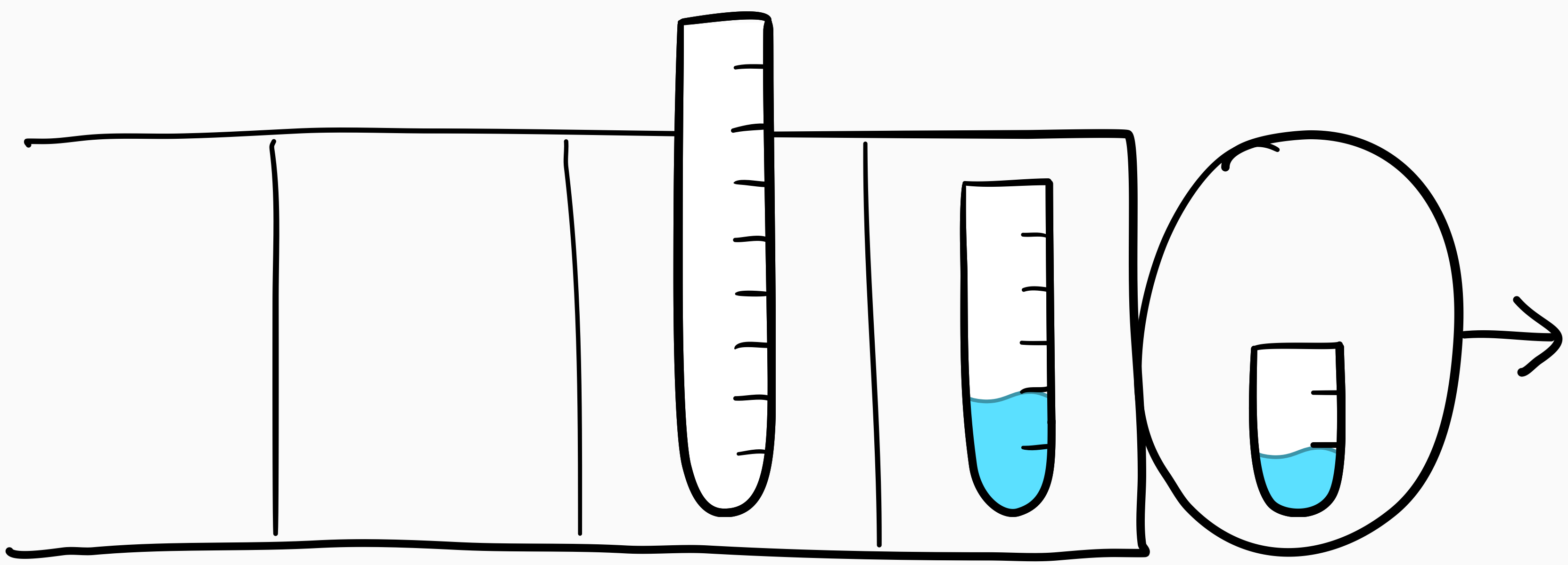
$\mathbf{E}[T]$ of SRPT



tagged job

$\mathbf{E}[T]$ of SRPT

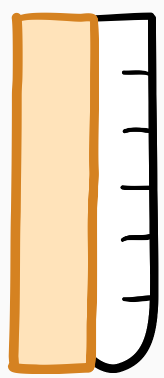


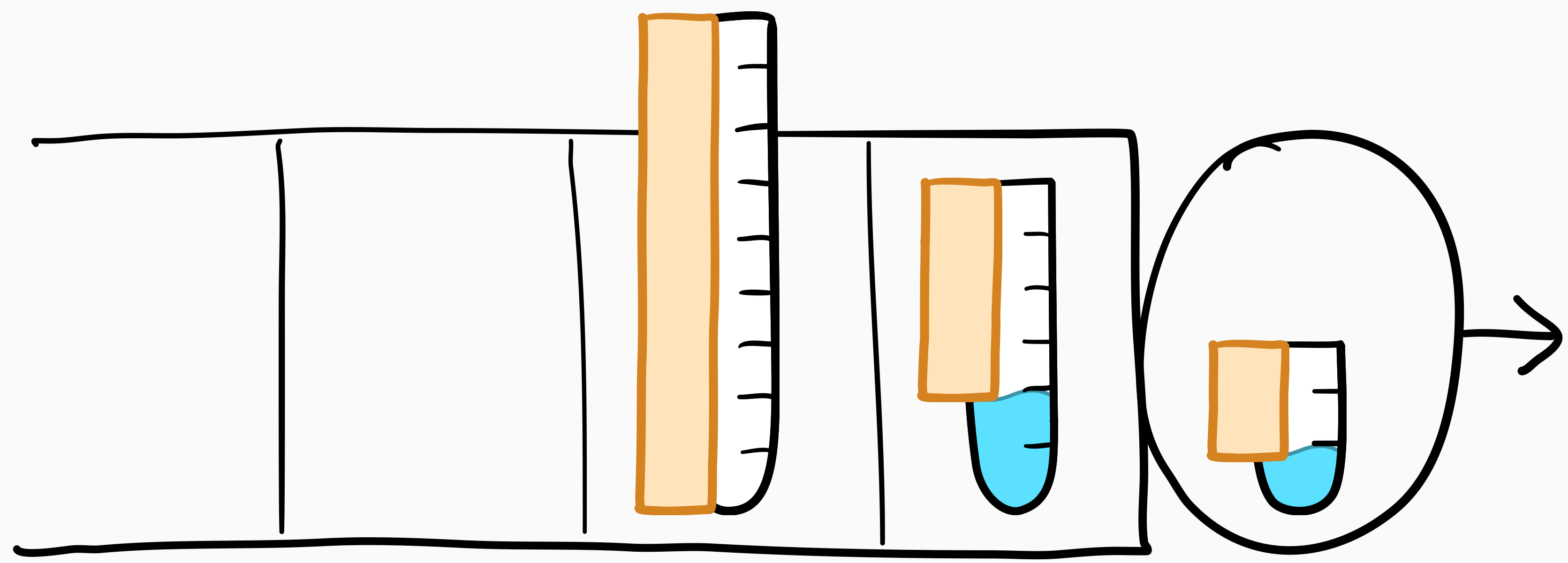


tagged job

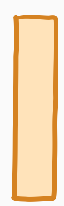
random system state

$\mathbf{E}[T]$ of SRPT





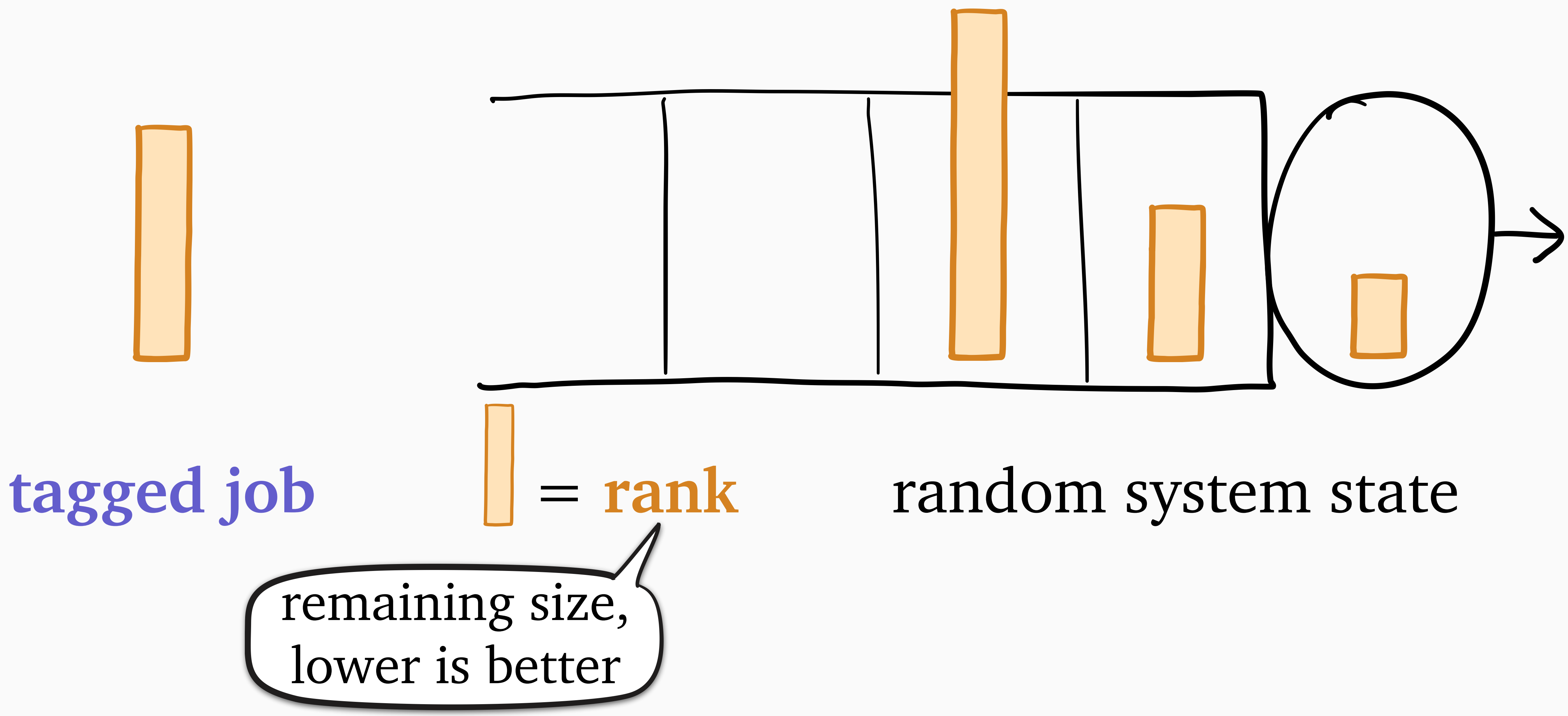
tagged job

 = rank

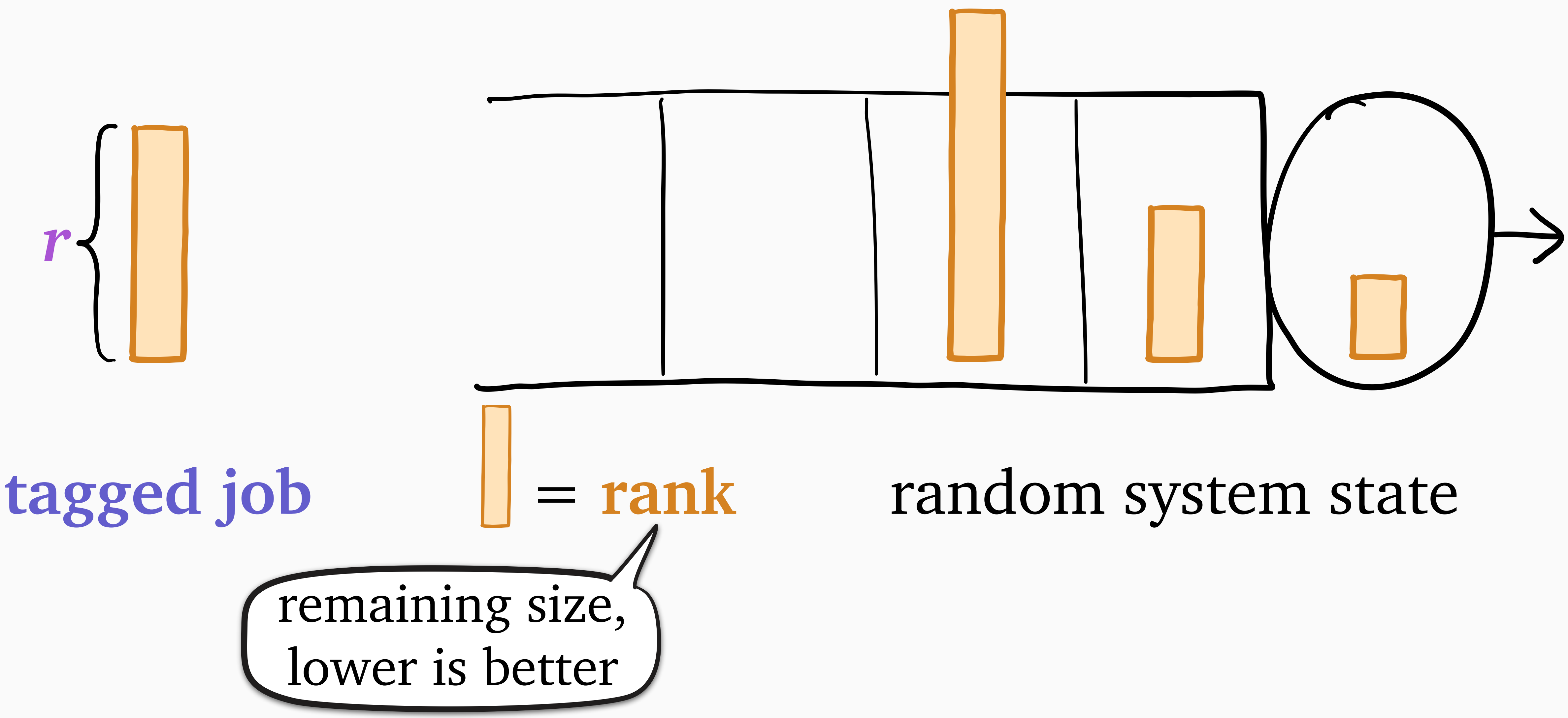
random system state

remaining size,
lower is better

E[T] of SRPT


tagged job
= rank
random system state
remaining size,
lower is better

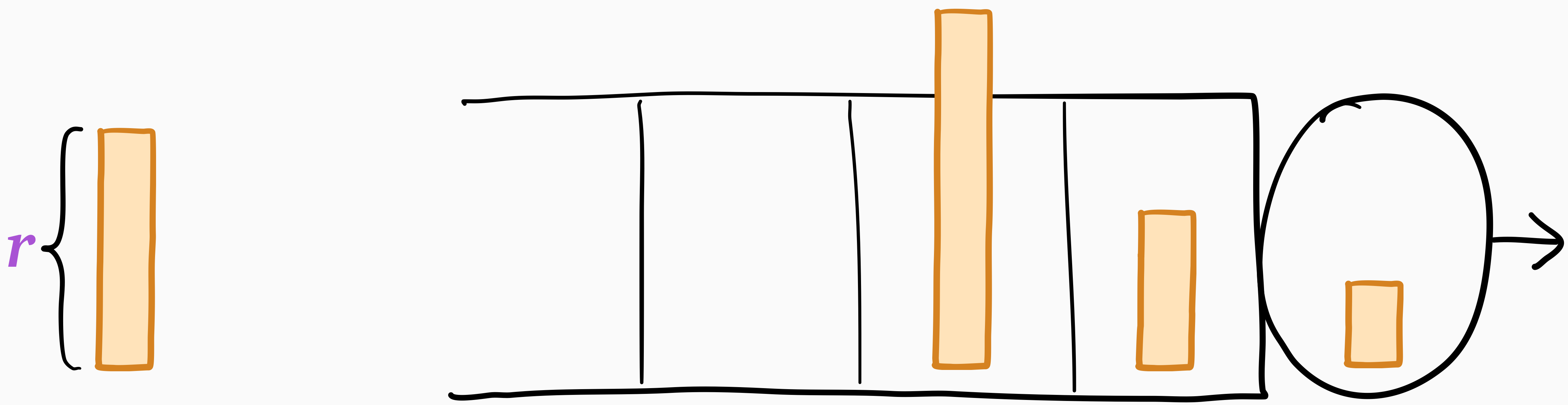
$\mathbf{E}[T]$ of SRPT



Key quantity:

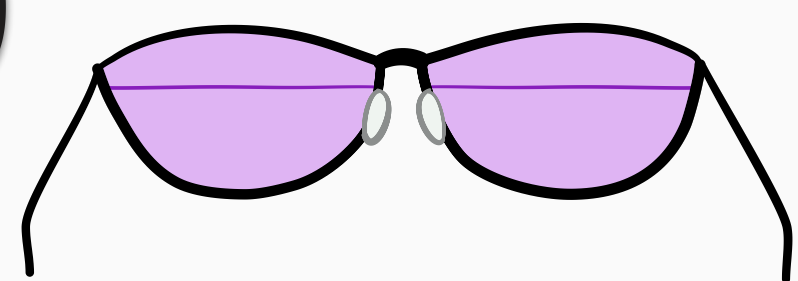
$W(r)$ = "r-work" = work relevant to job of **rank** r

$\mathbf{E}[T]$ of SRPT



tagged job | = **rank** random system state

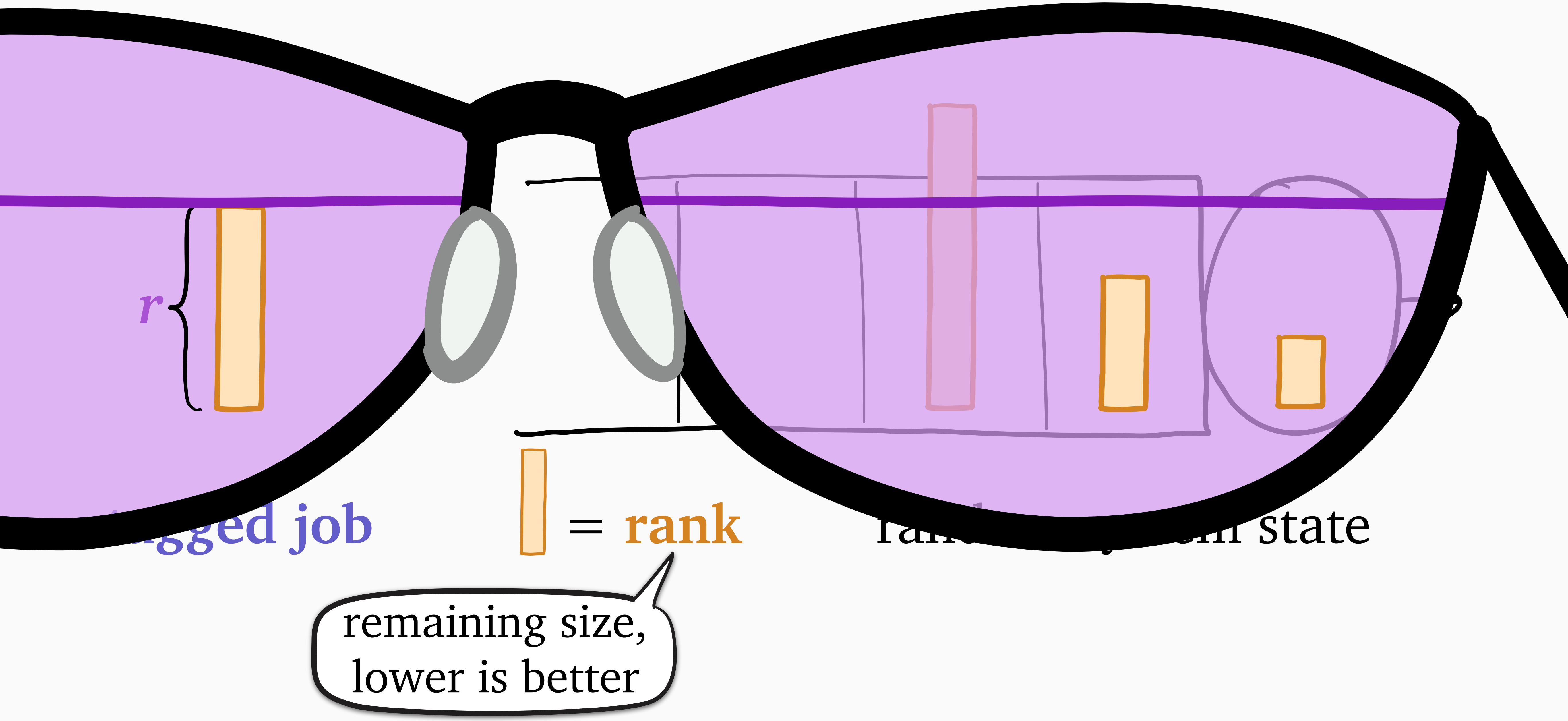
remaining size, lower is better



Key quantity:

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

$\mathbf{E}[T]$ of SRPT

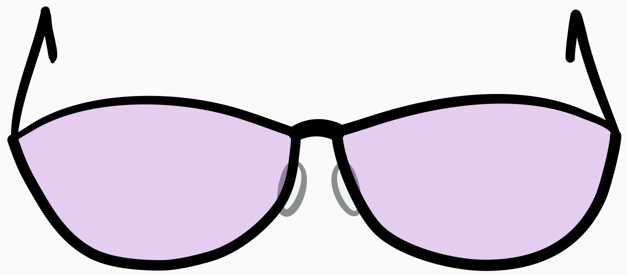


Key quantity:

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

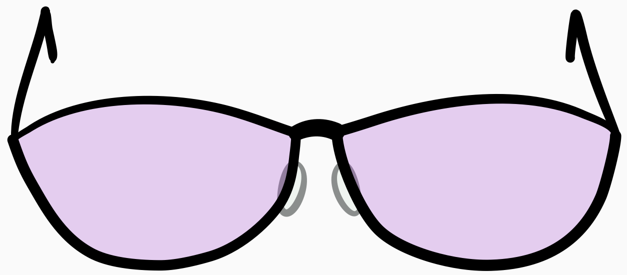
Two-step $\mathbf{E}[T]$ analysis

Two-step $\mathbf{E}[T]$ analysis

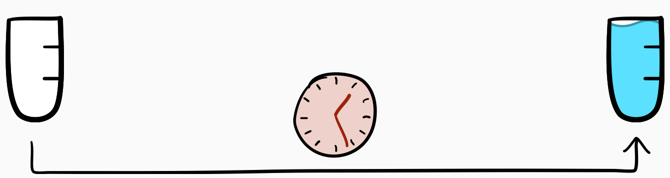


Step 1: compute $\mathbf{E}[W(\boldsymbol{r})]$

Two-step $\mathbf{E}[T]$ analysis

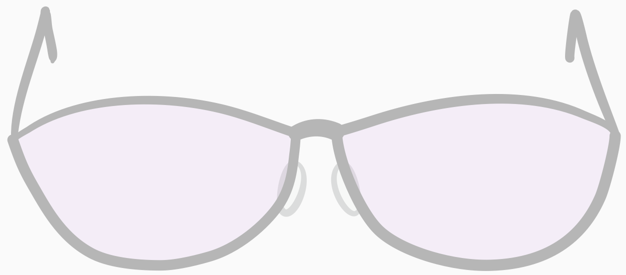


Step 1: compute $\mathbf{E}[W(\boldsymbol{r})]$

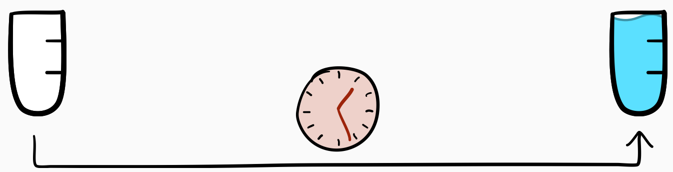


Step 2: $\mathbf{E}[W(\boldsymbol{r})]$ to $\mathbf{E}[T]$

Two-step $\mathbf{E}[T]$ analysis

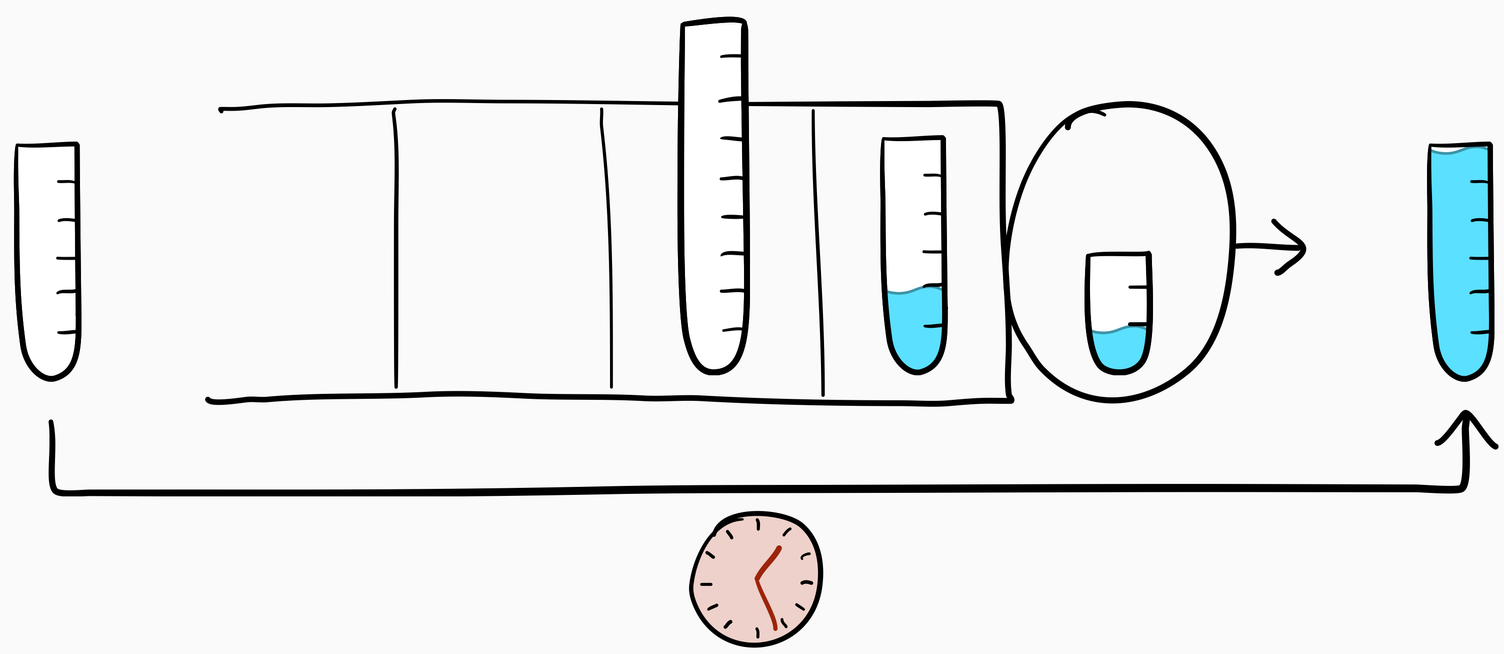


Step 1: compute $\mathbf{E}[W(r)]$

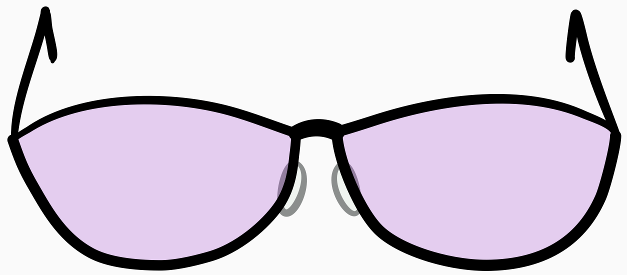


Step 2: $\mathbf{E}[W(r)]$ to $\mathbf{E}[T]$

Tagged job method

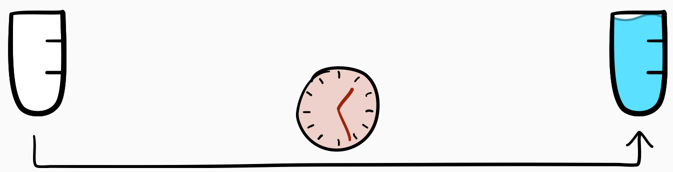


Two-step $\mathbf{E}[T]$ analysis



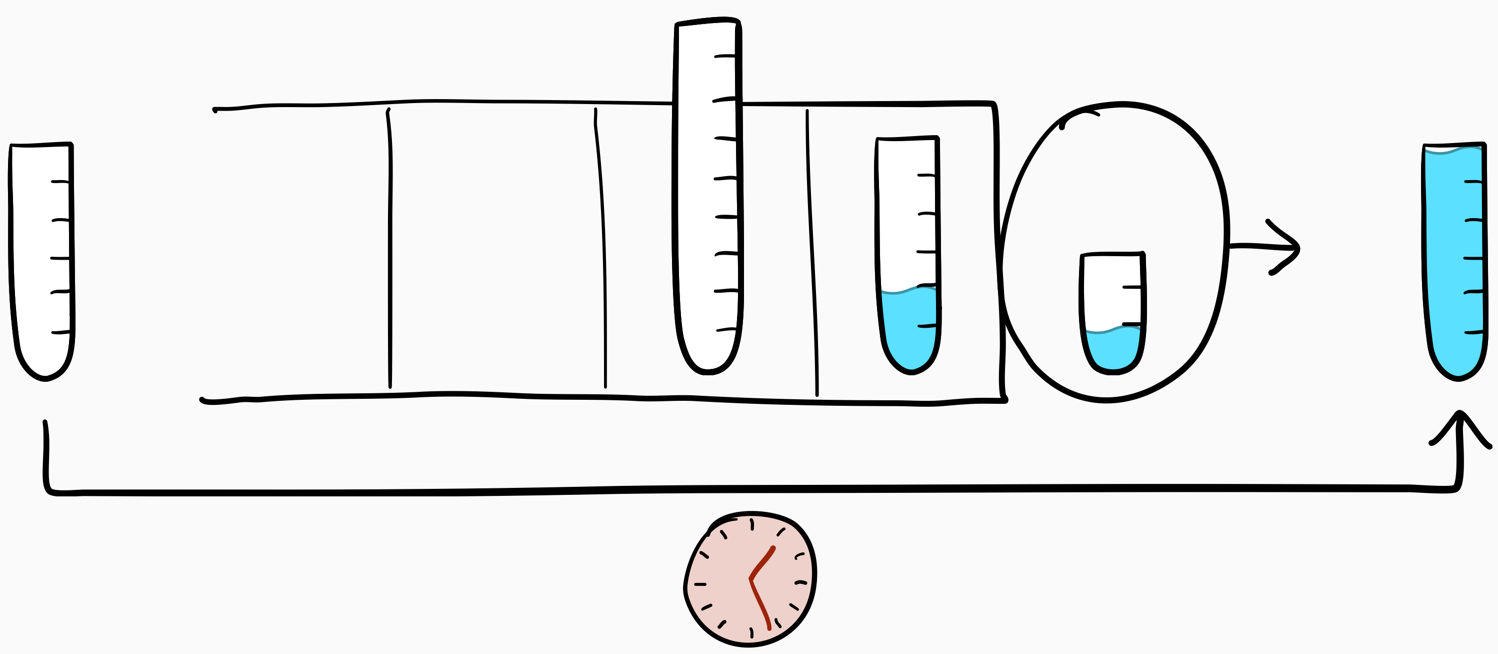
Step 1: compute $\mathbf{E}[W(r)]$

standard queueing

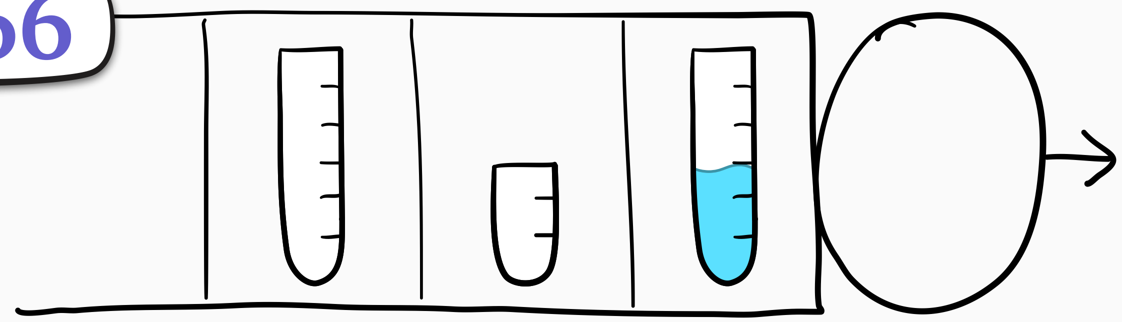


Step 2: $\mathbf{E}[W(r)]$ to $\mathbf{E}[T]$

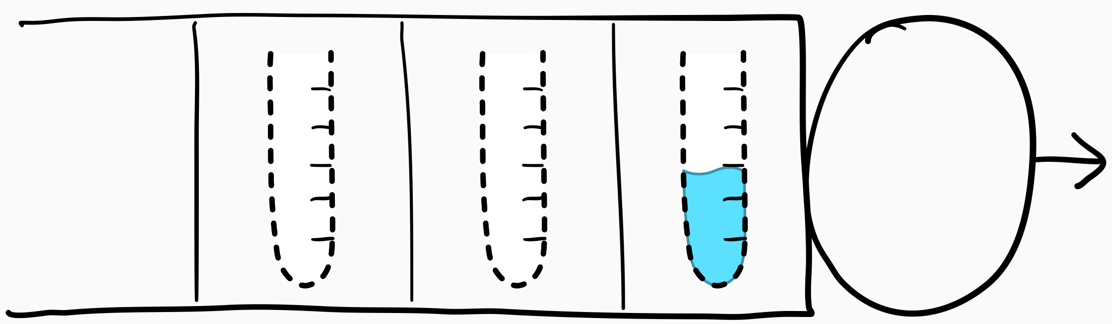
Tagged job method



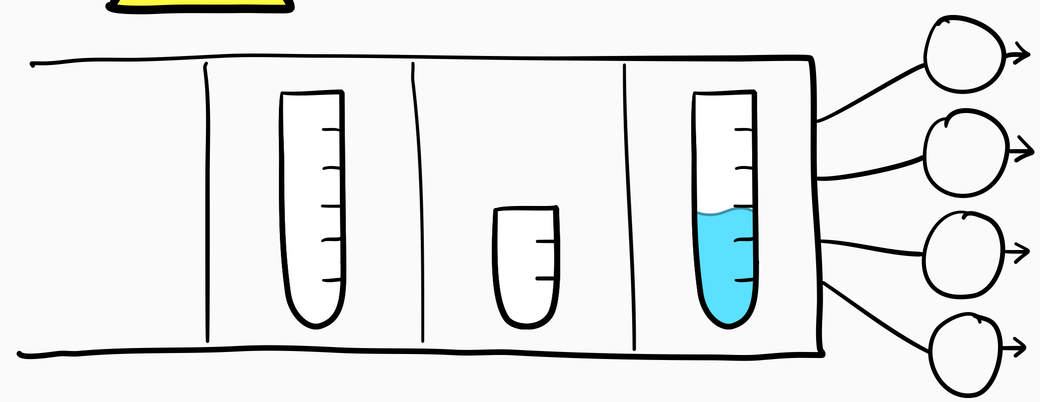
1966 short before long



2018 unknown sizes

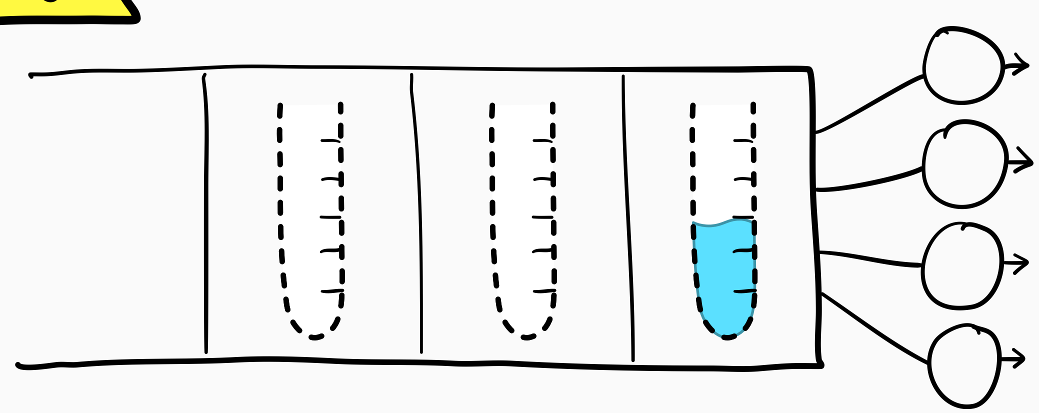


2018 multiple servers

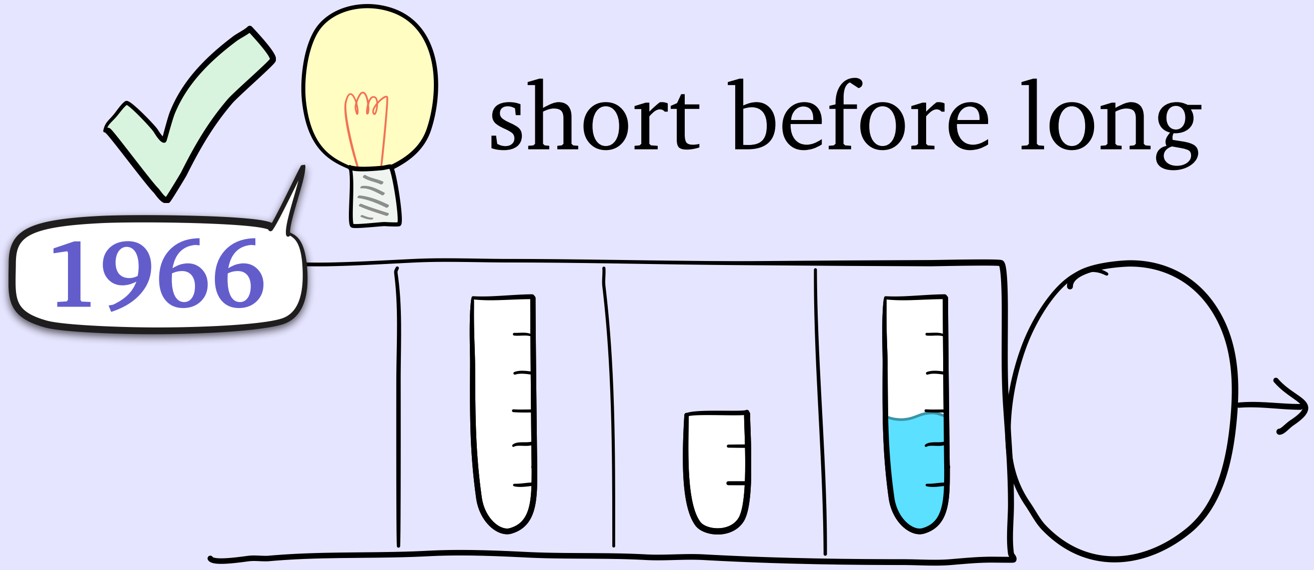


This work: both at once!

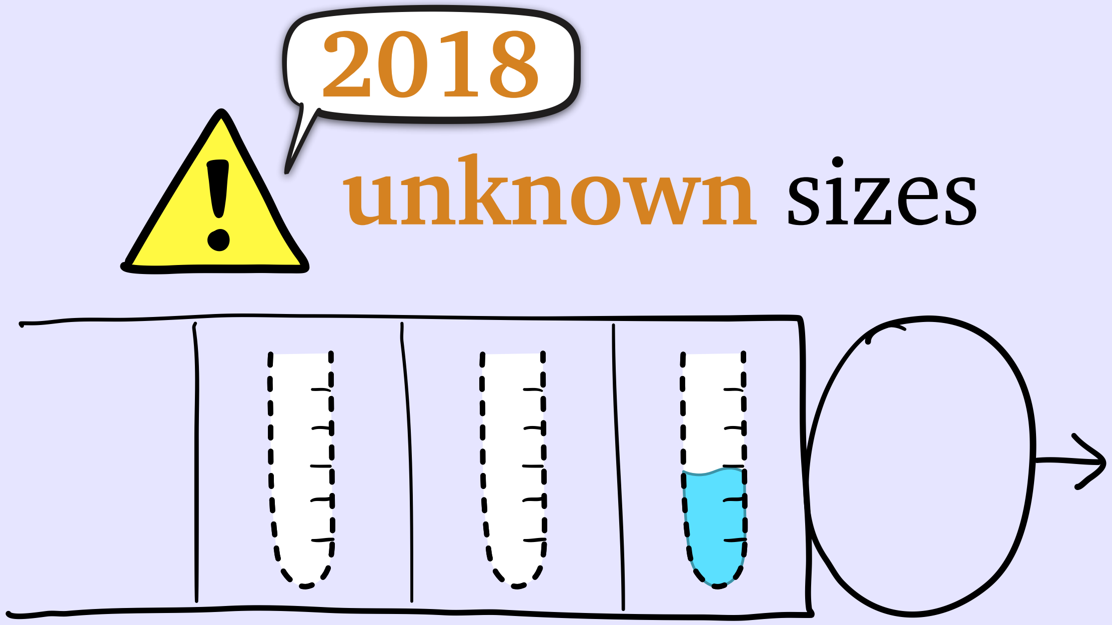
2021



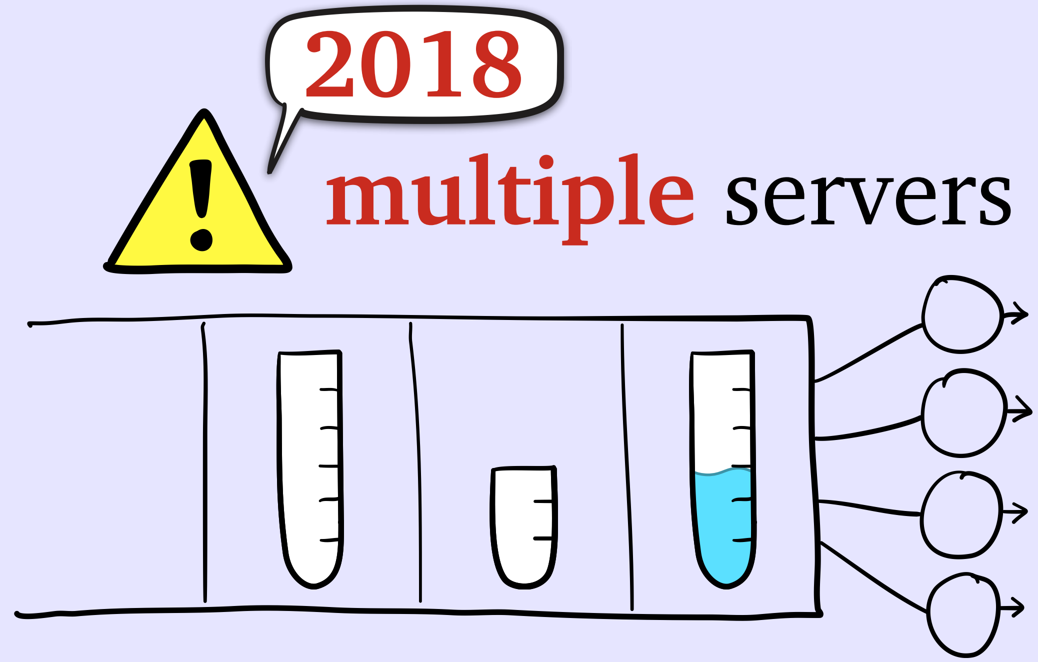
Tagged job methods



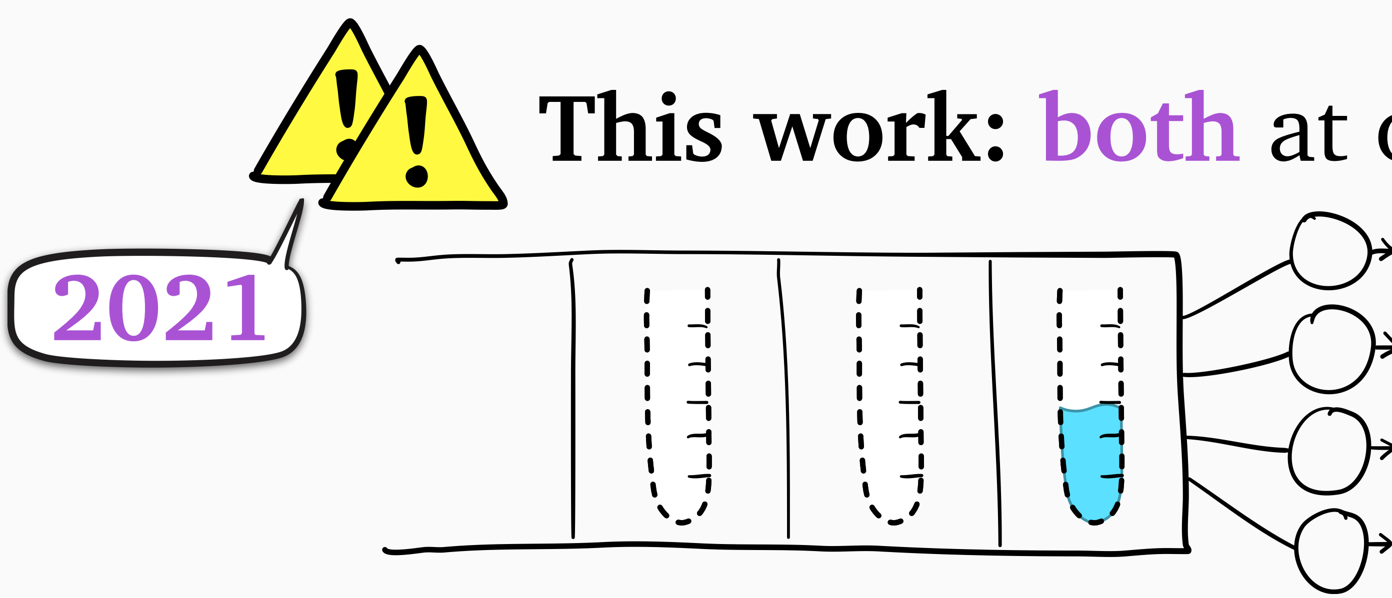
short before long



unknown sizes

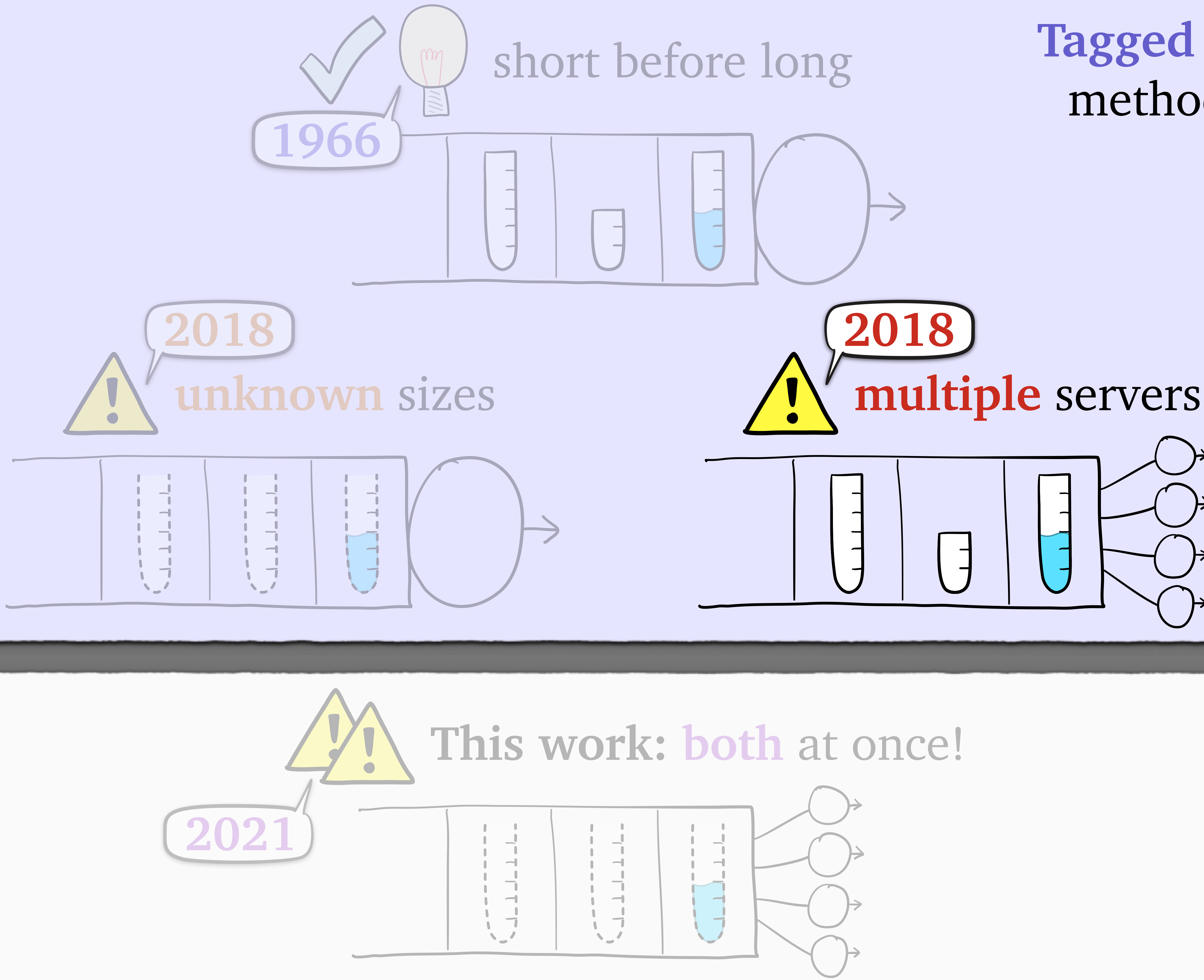


multiple servers

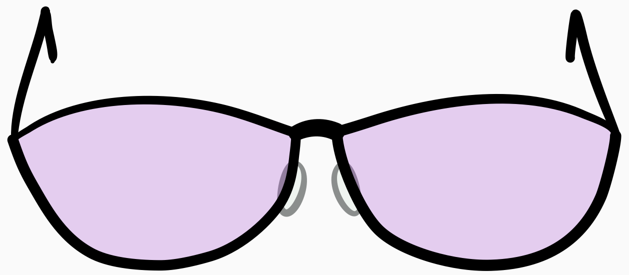


This work: both at once!

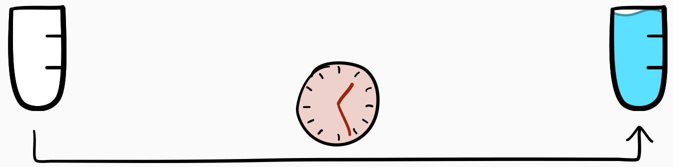
Tagged job methods



$\mathbf{E}[T]$ with **multiple** servers

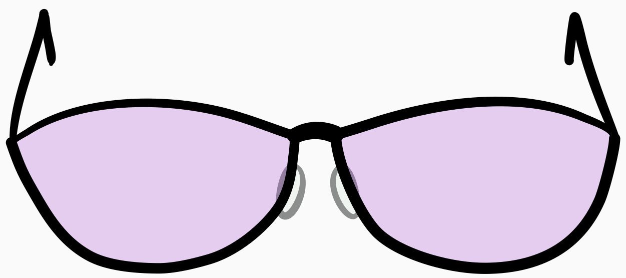


Step 1: compute $\mathbf{E}[W(\boldsymbol{r})]$



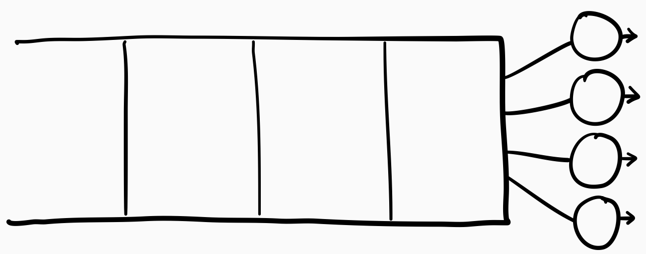
Step 2: $\mathbf{E}[W(\boldsymbol{r})]$ to $\mathbf{E}[T]$

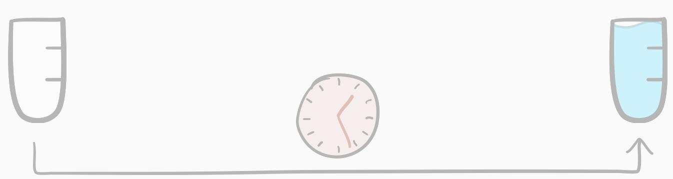
$\mathbf{E}[T]$ with multiple servers



Step 1: compute $\mathbf{E}[W(r)]$

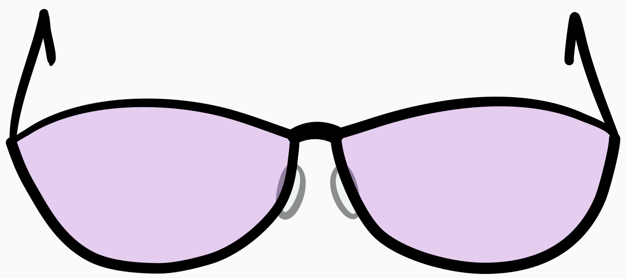
SRPT-k



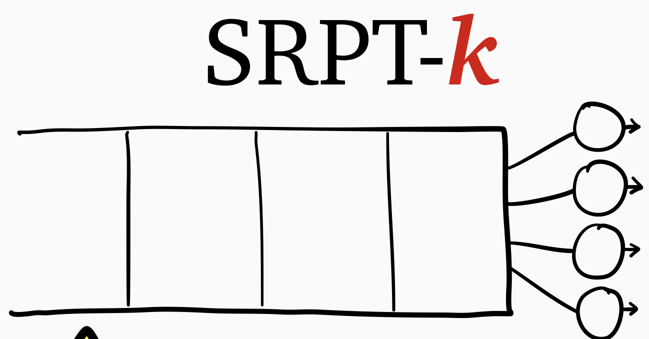


Step 2: $\mathbf{E}[W(r)]$ to $\mathbf{E}[T]$

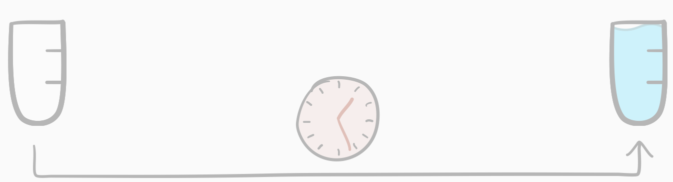
$\mathbf{E}[T]$ with **multiple** servers



Step 1: compute $\mathbf{E}[W(r)]$

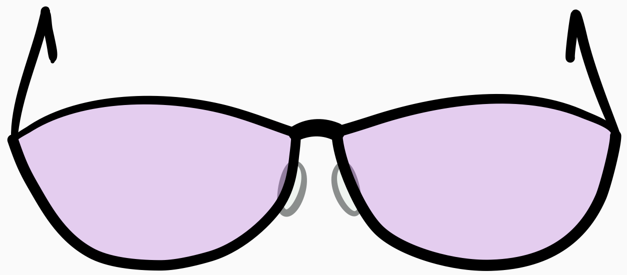


intractable

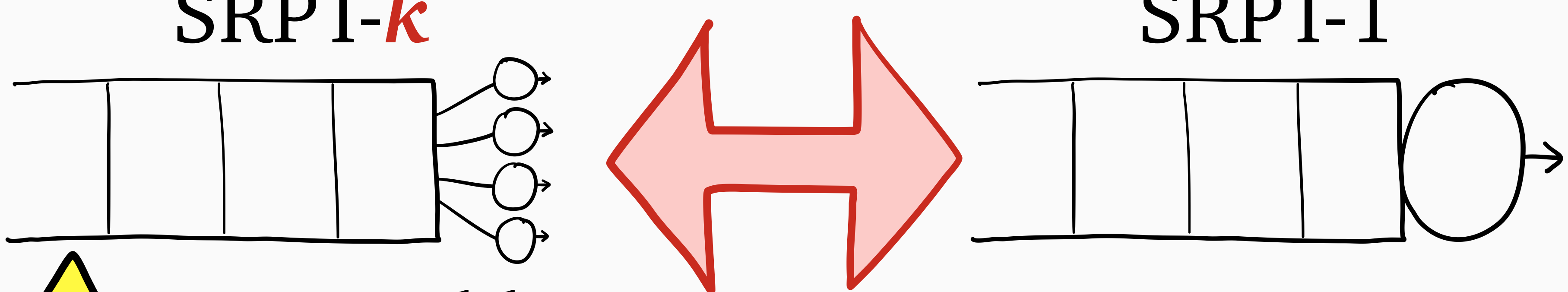


Step 2: $\mathbf{E}[W(r)]$ to $\mathbf{E}[T]$

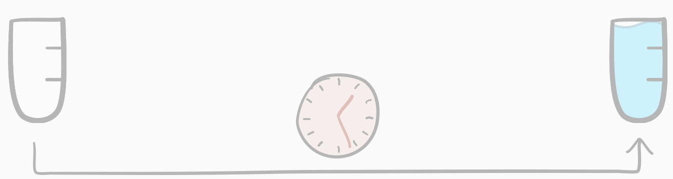
$\mathbf{E}[T]$ with **multiple** servers



Step 1: compute $\mathbf{E}[W(r)]$

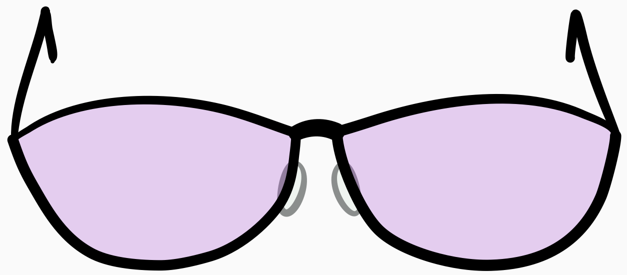


intractable

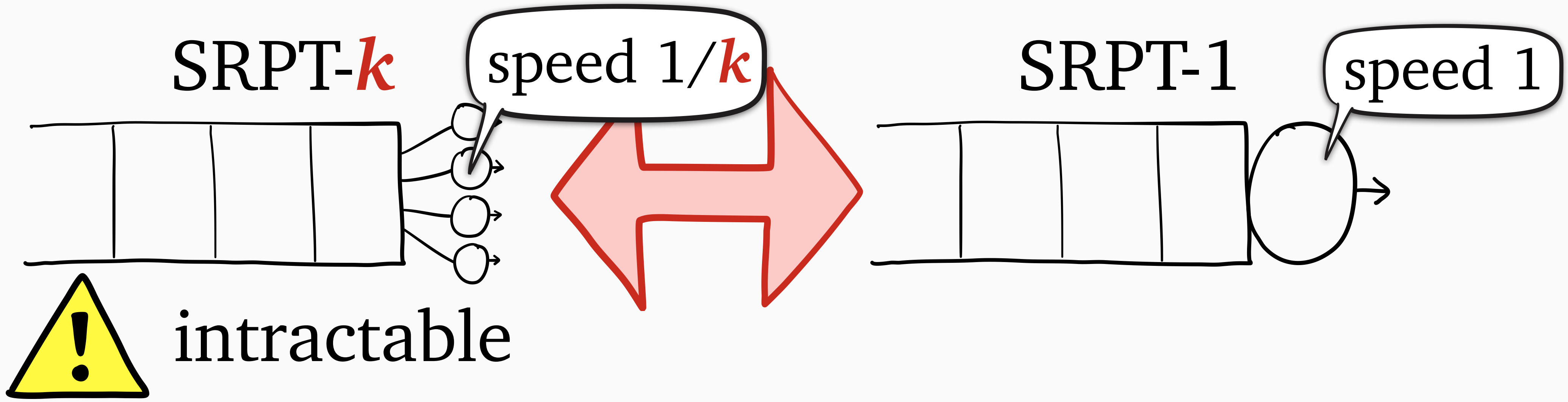


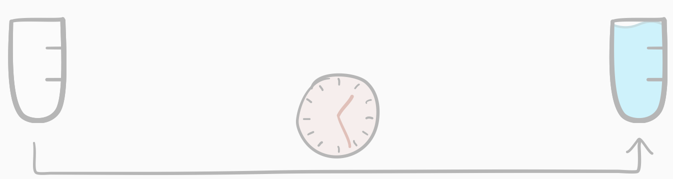
Step 2: $\mathbf{E}[W(r)]$ to $\mathbf{E}[T]$

E[T] with **multiple** servers



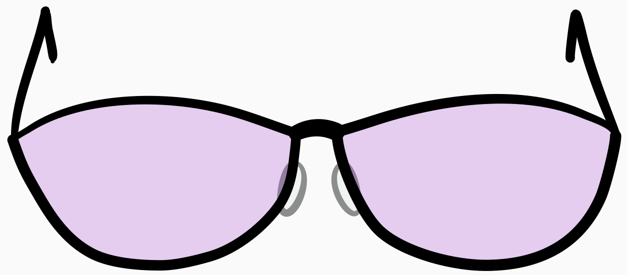
Step 1: compute **E**[$W(r)$]



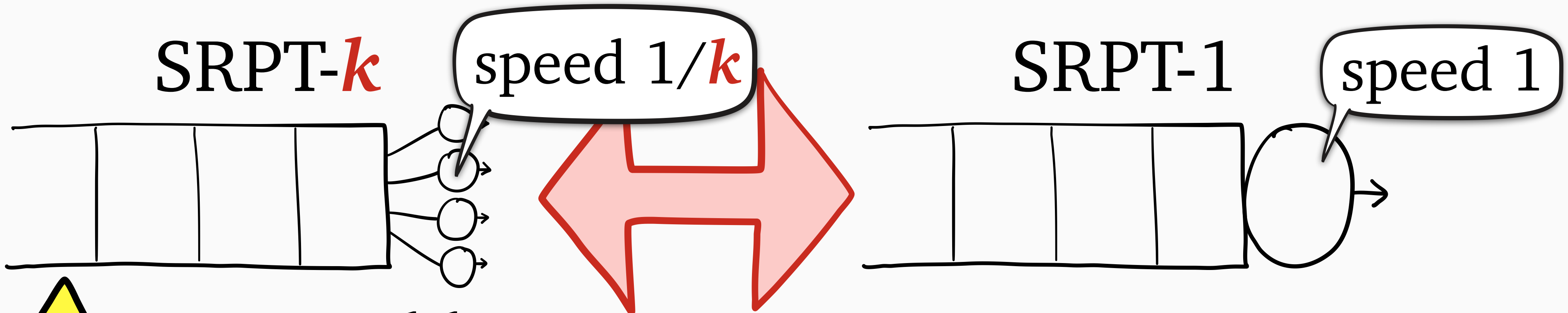


Step 2: E[$W(r)$] to **E**[T]

$\mathbf{E}[T]$ with **multiple** servers



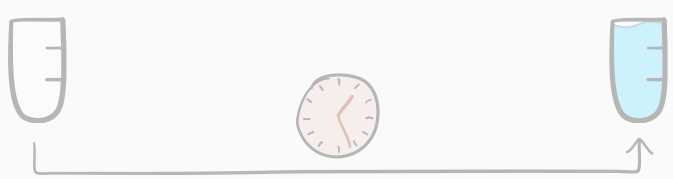
Step 1: compute $\mathbf{E}[W(r)]$



intractable

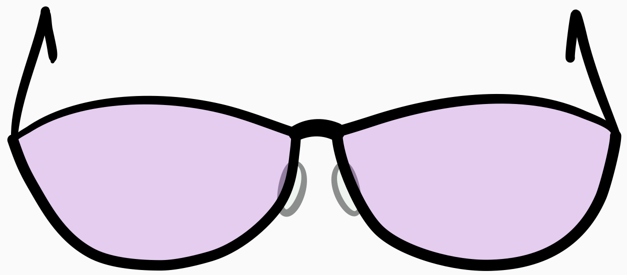
Worst-case gap:

$$W_{\text{SRPT-}k}(r) \le W_{\text{SRPT-}1}(r) + kr$$

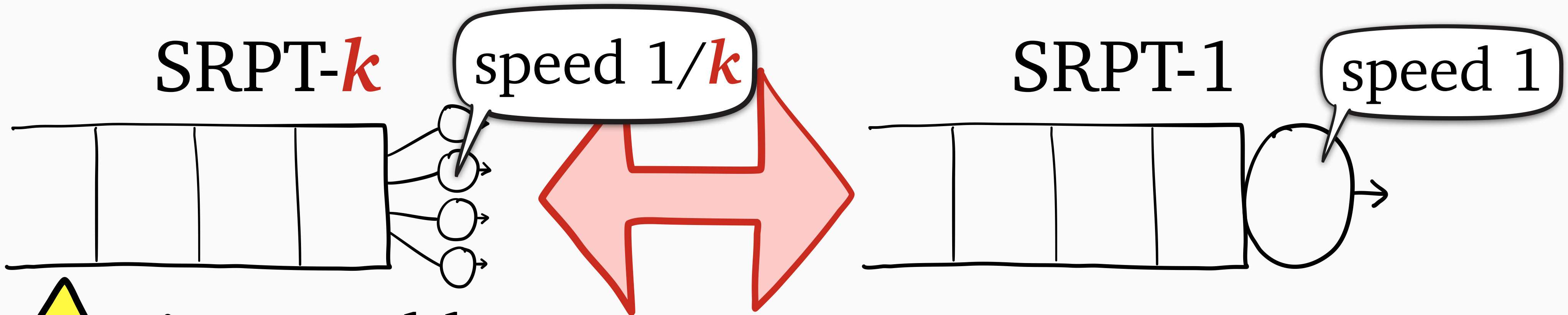


Step 2: $\mathbf{E}[W(r)]$ to $\mathbf{E}[T]$

$\mathbf{E}[T]$ with **multiple** servers



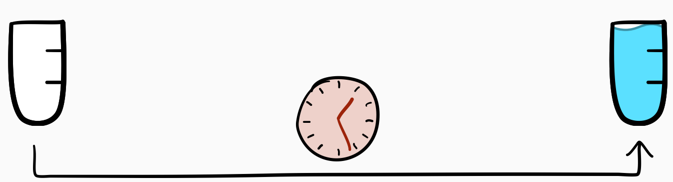
Step 1: compute $\mathbf{E}[W(r)]$



intractable

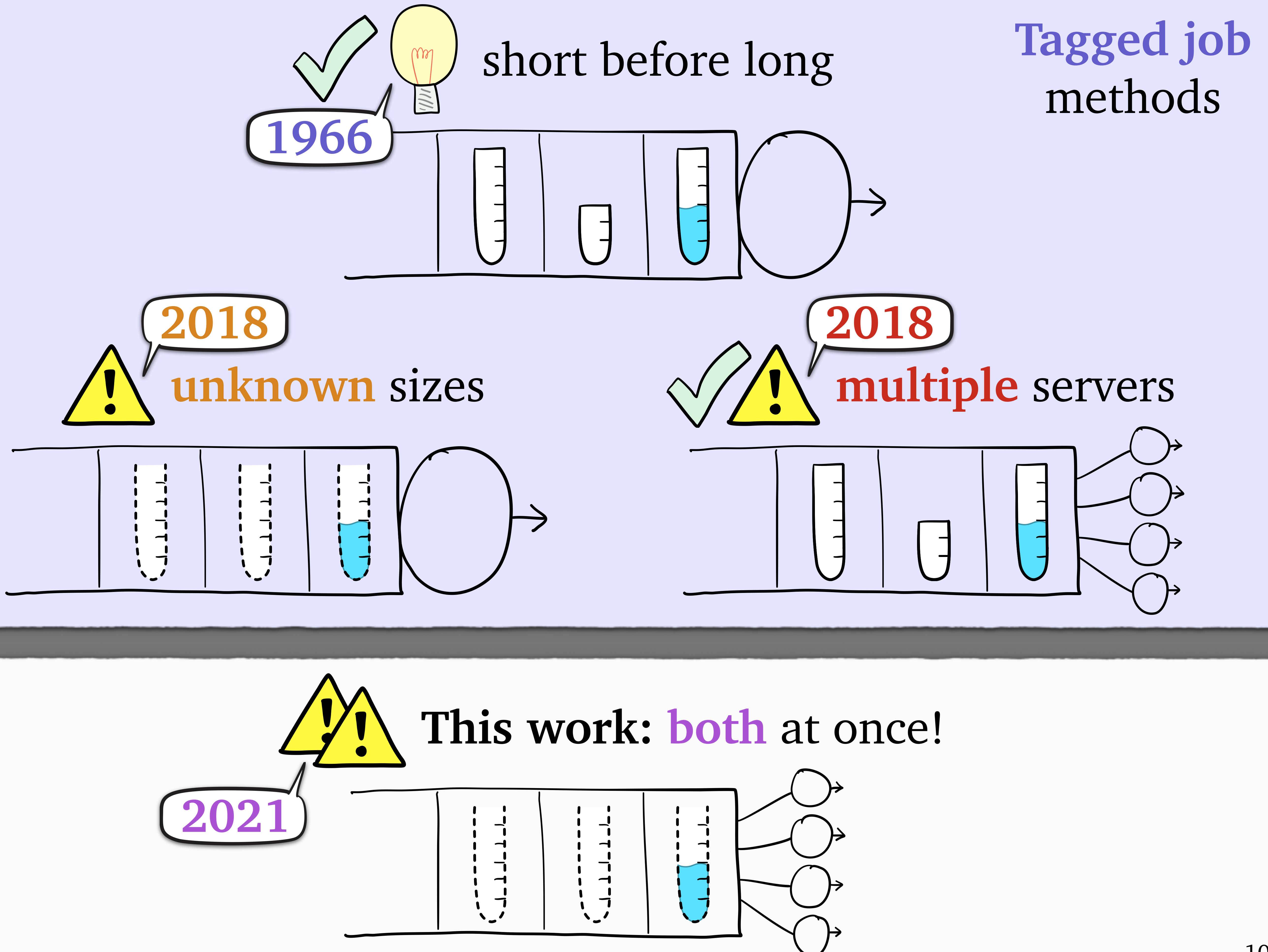
Worst-case gap:

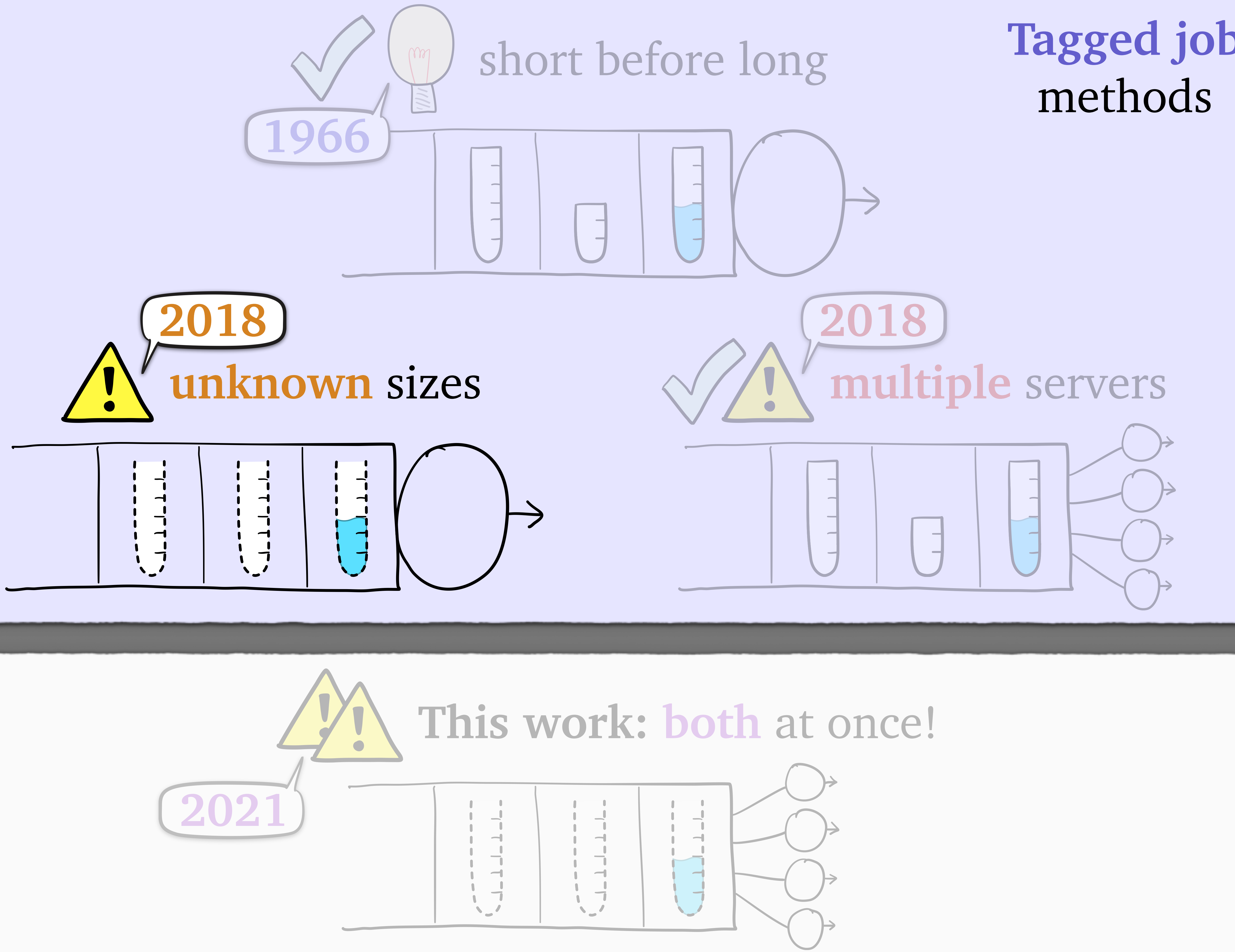
$$W_{\text{SRPT-}k}(r) \leq W_{\text{SRPT-1}}(r) + kr$$



Step 2: $\mathbf{E}[W(r)]$ to $\mathbf{E}[T]$

tagged job
+
worst-case

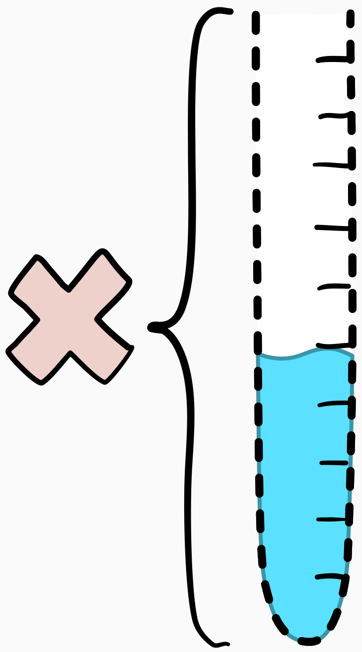

Tagged job methods
1966
short before long
2018
unknown sizes
2018
multiple servers
This work: both at once!
2021


Tagged job
methods
1966
short before long
2018
unknown sizes
2018
multiple servers
2021
This work: both at once!

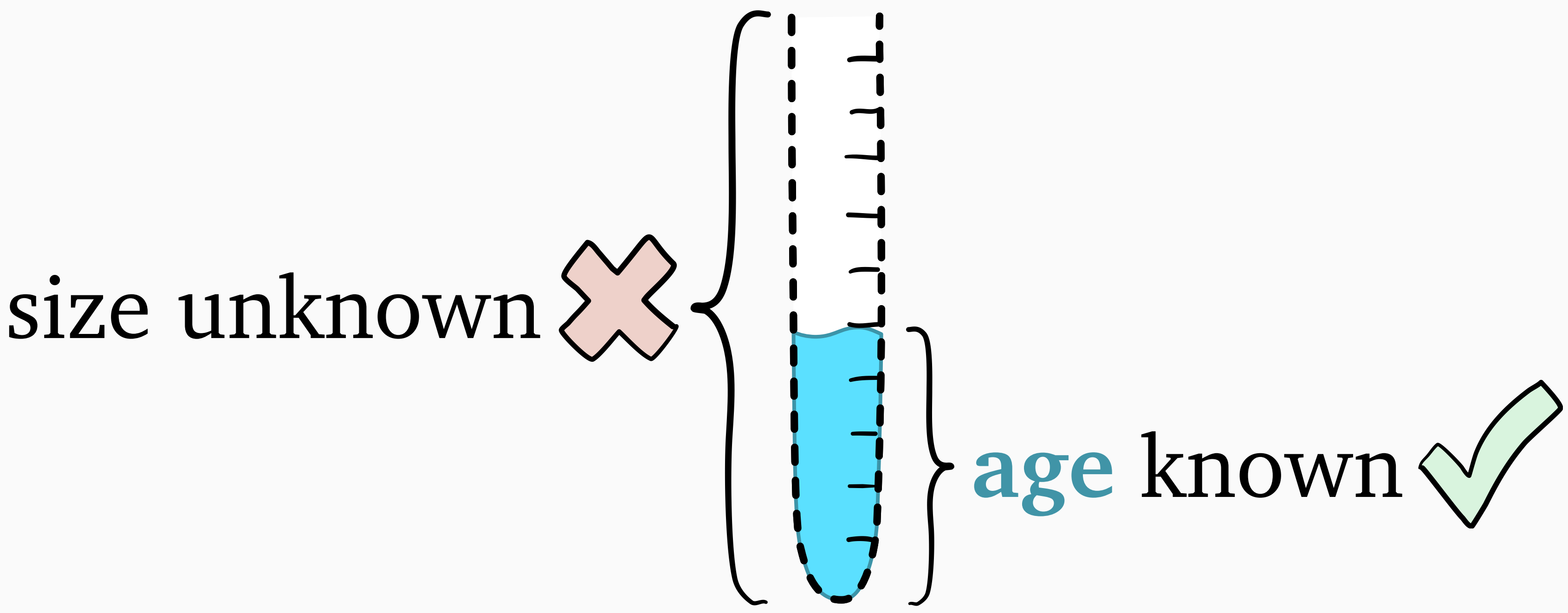
Scheduling with **unknown** sizes

Scheduling with unknown sizes

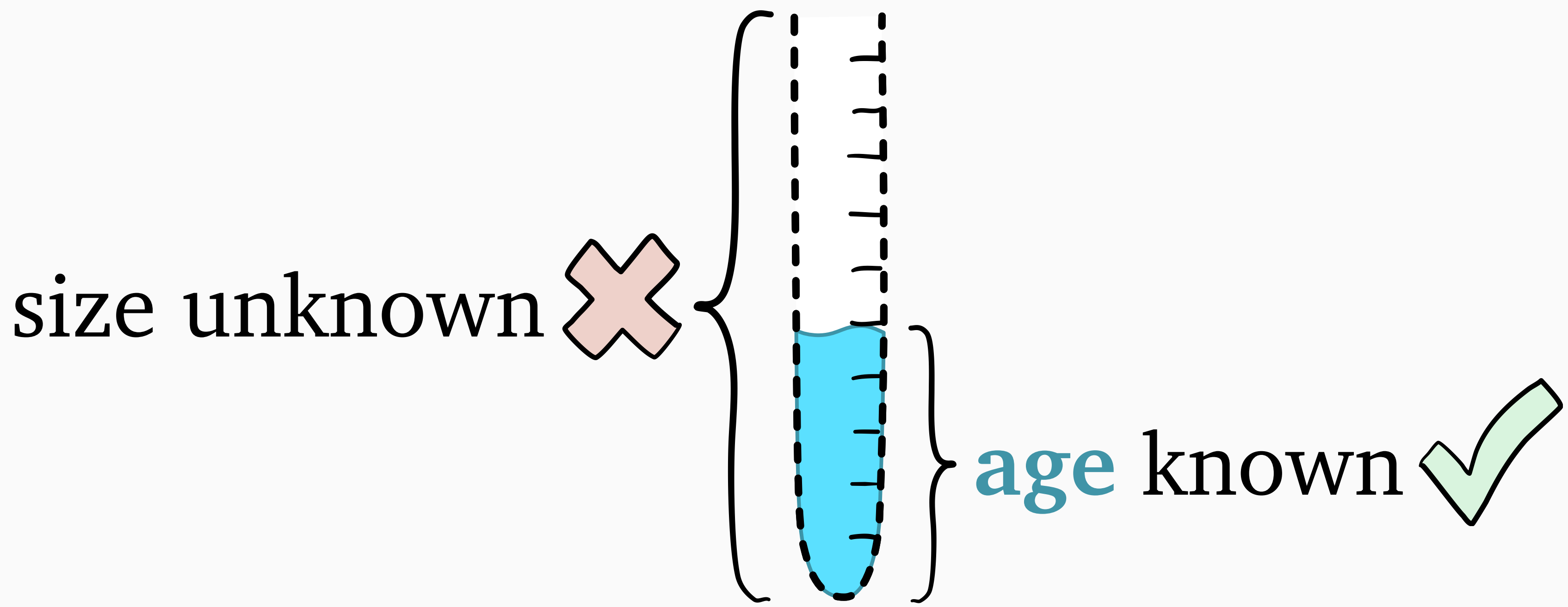
size unknown

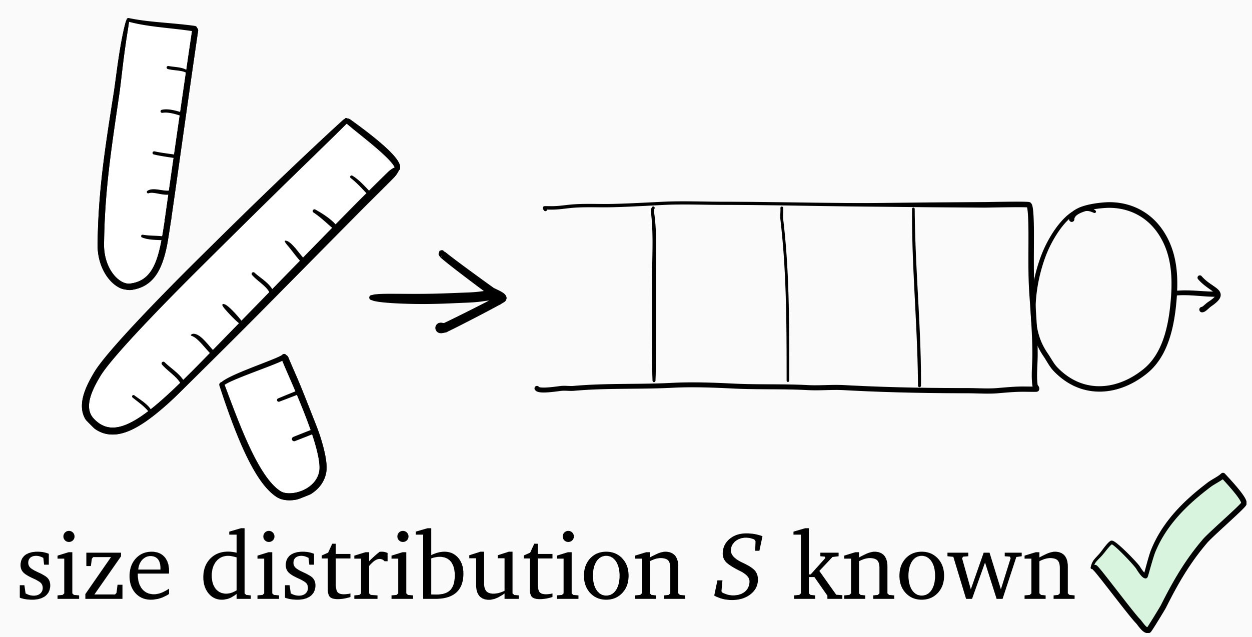


Scheduling with **unknown** sizes

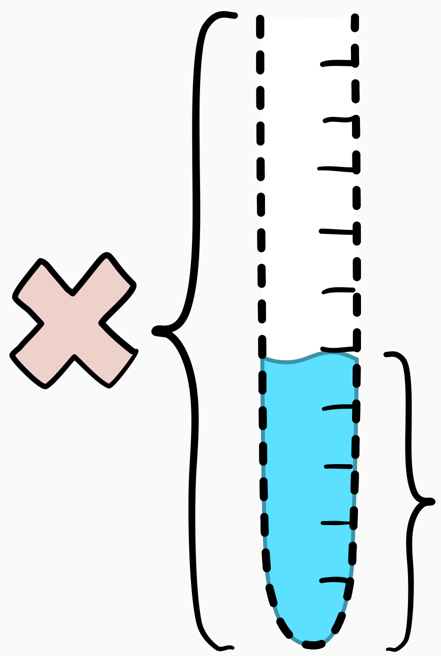


Scheduling with **unknown** sizes



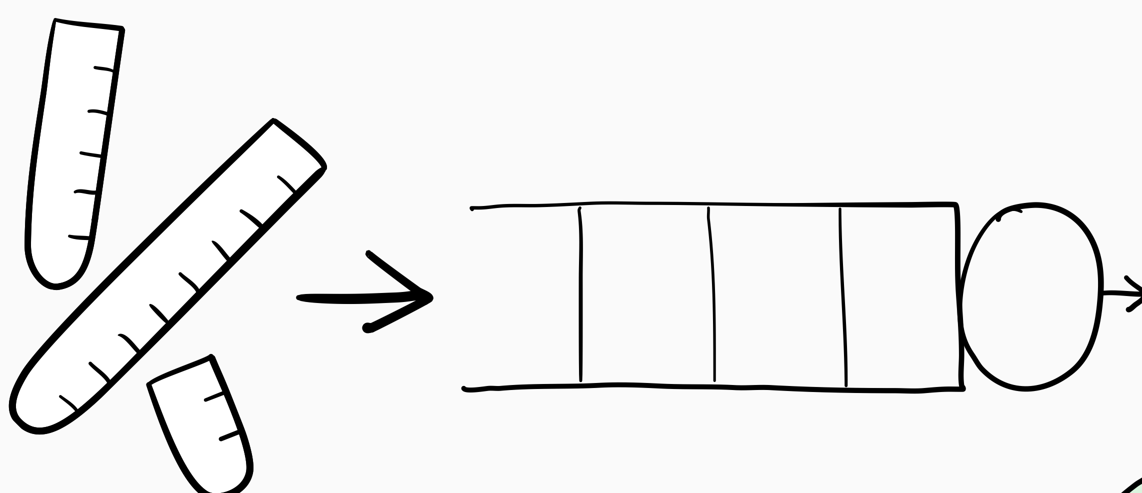


Scheduling with **unknown** sizes



size unknown

age known

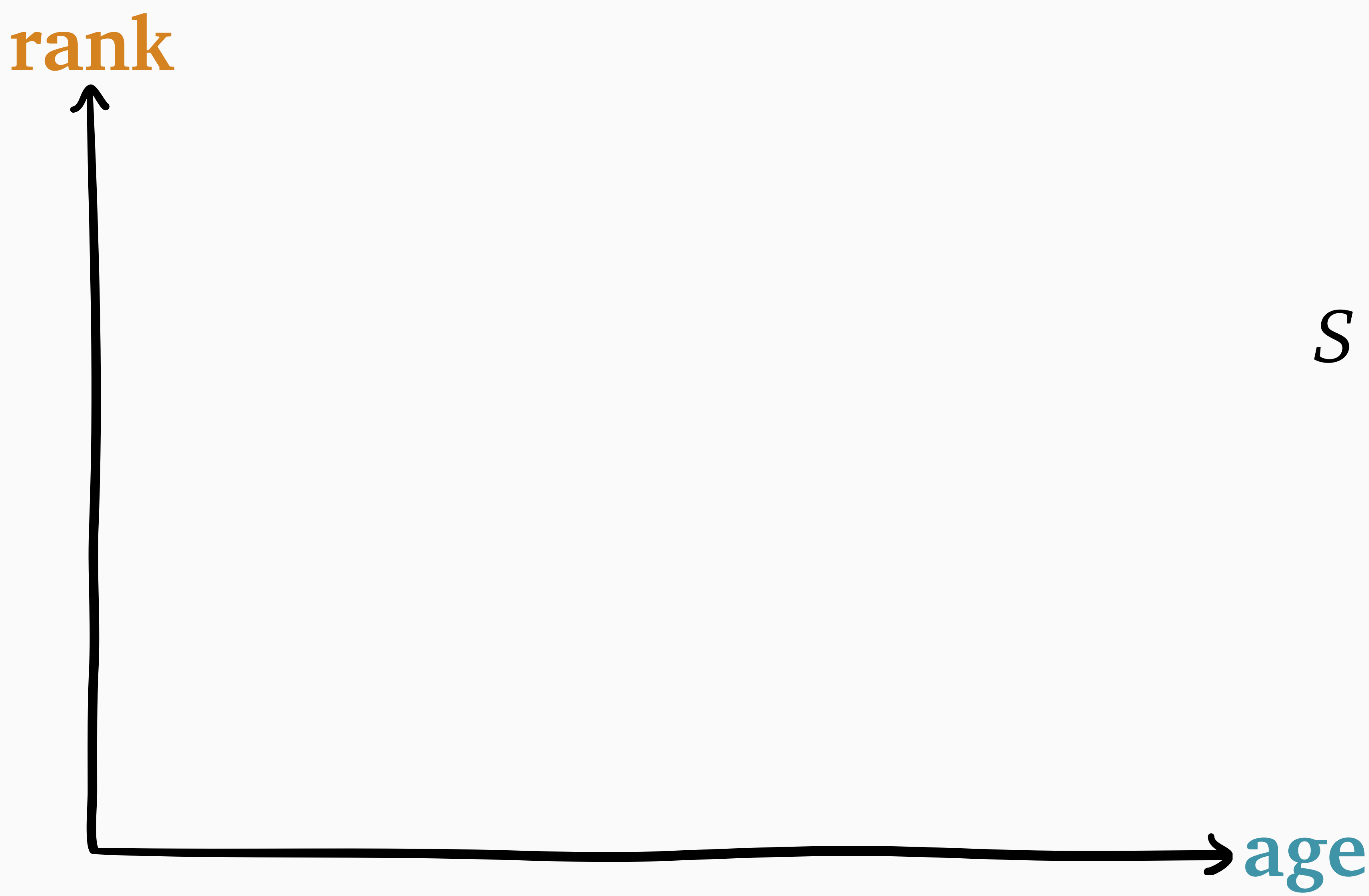


size distribution S known

Example:

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

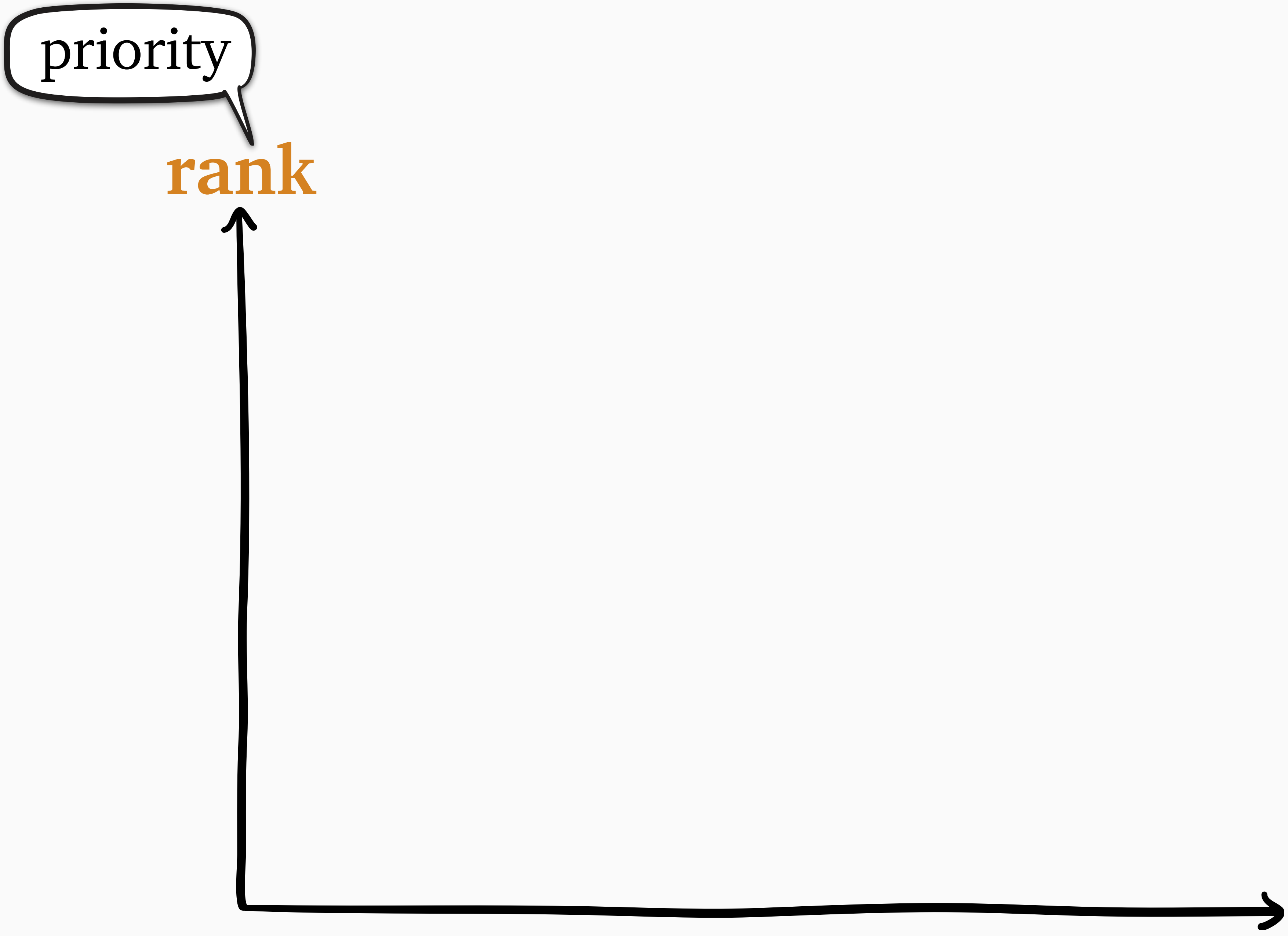
Scheduling with **unknown** sizes

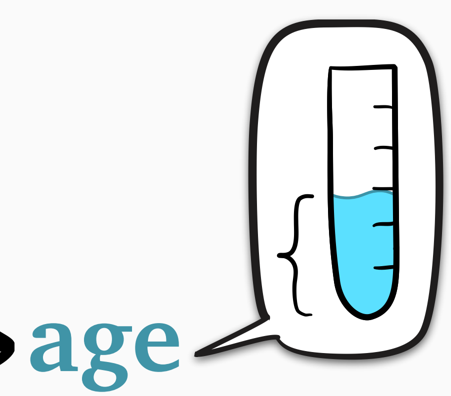


Example:

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

Scheduling with **unknown** sizes

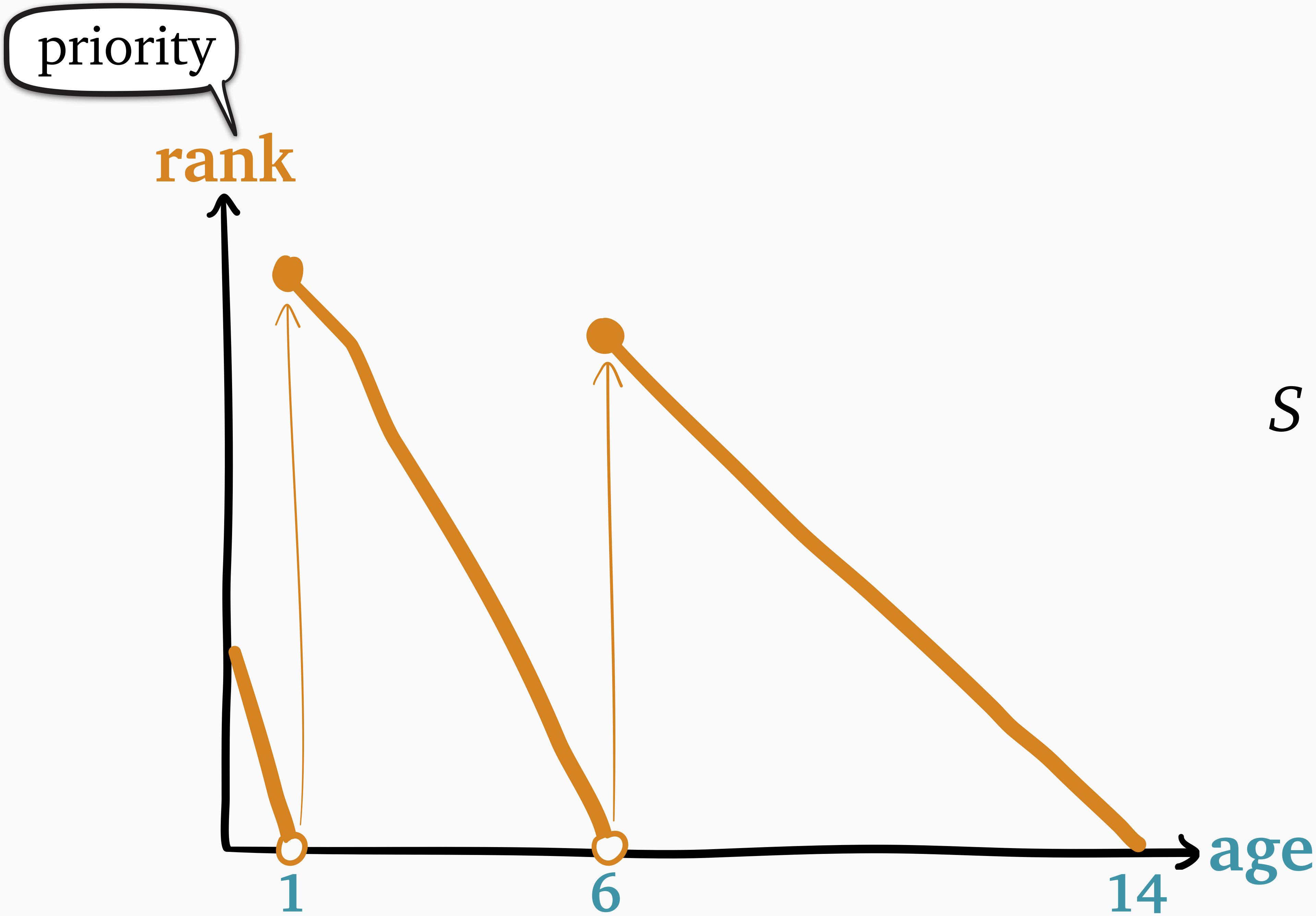




Example:

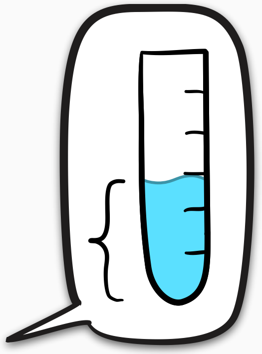
$$S = \begin{cases} 1 & \text{w.p. } \frac{1}{3} \\ 6 & \text{w.p. } \frac{1}{3} \\ 14 & \text{w.p. } \frac{1}{3} \end{cases}$$

Scheduling with **unknown** sizes



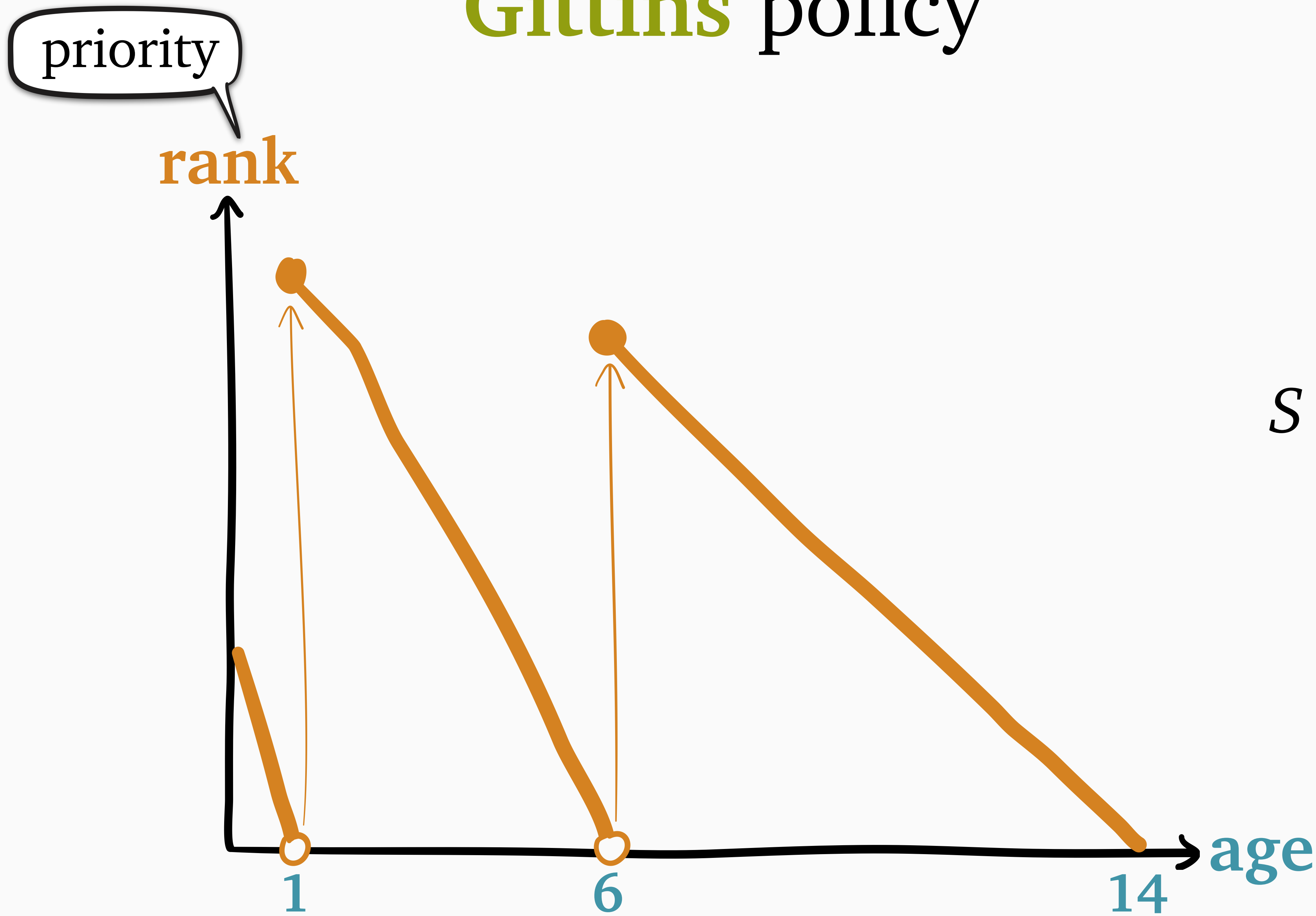
Example:

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



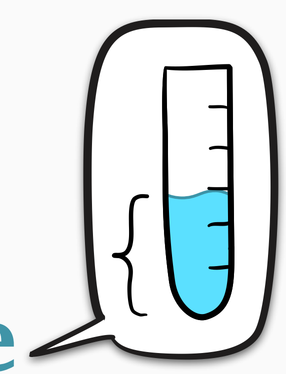
Scheduling with **unknown** sizes

Gittins policy



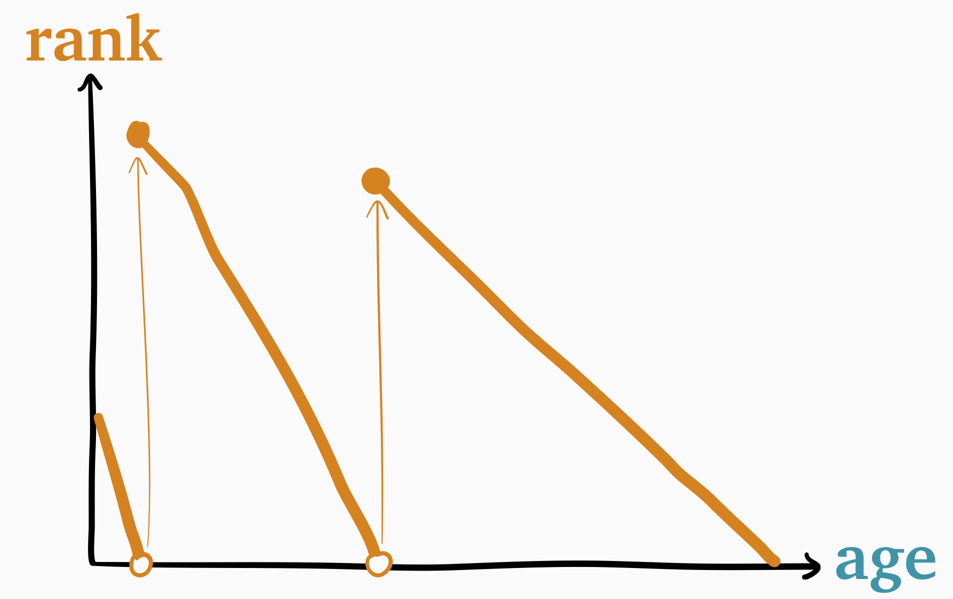
Example:

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



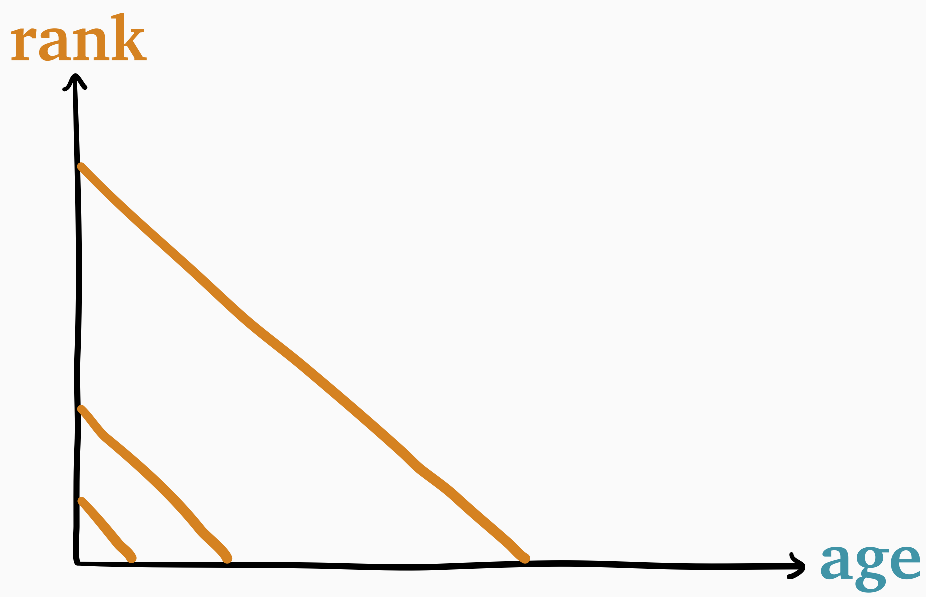
Analyzing *nonmonotonic* **rank**

Gittins: *nonmonotonic*

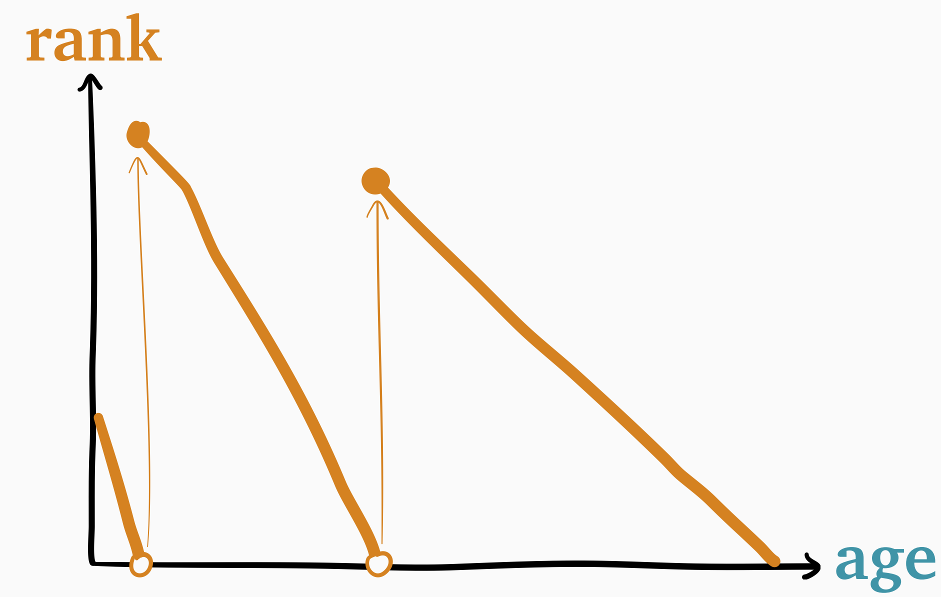


Analyzing *nonmonotonic* **rank**

SRPT: monotonic

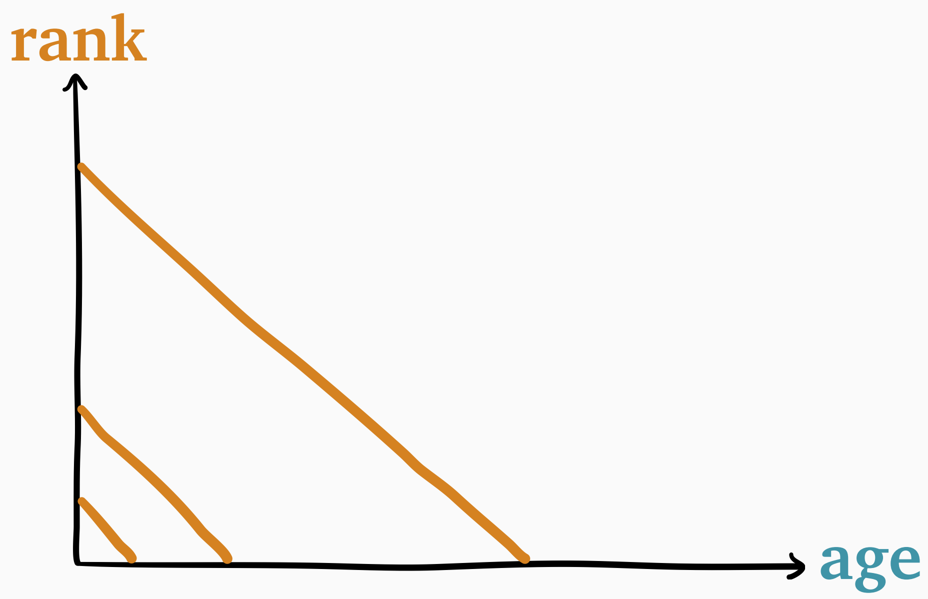


Gittins: *nonmonotonic*

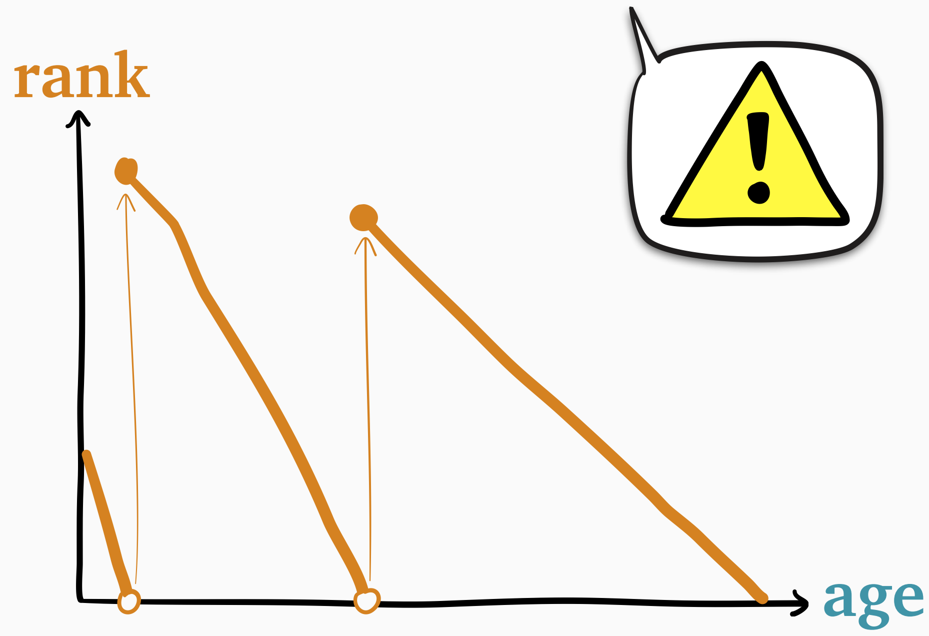


Analyzing *nonmonotonic* **rank**

SRPT: monotonic

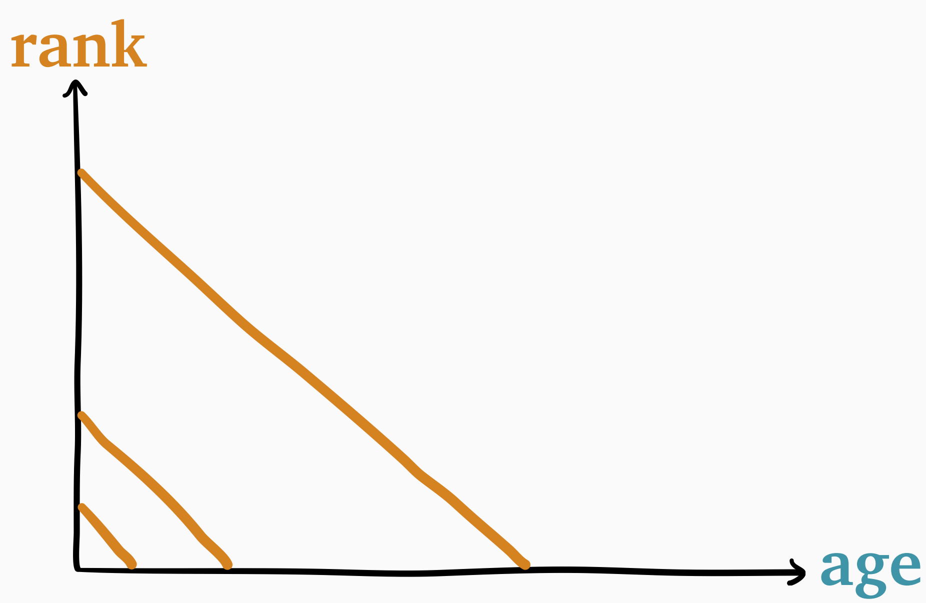


Gittins: *nonmonotonic*

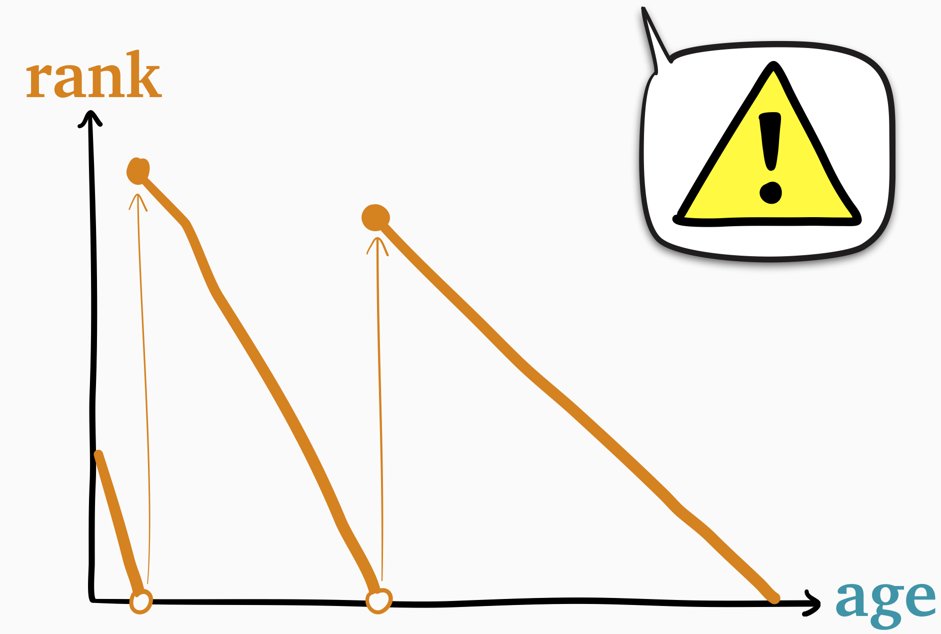


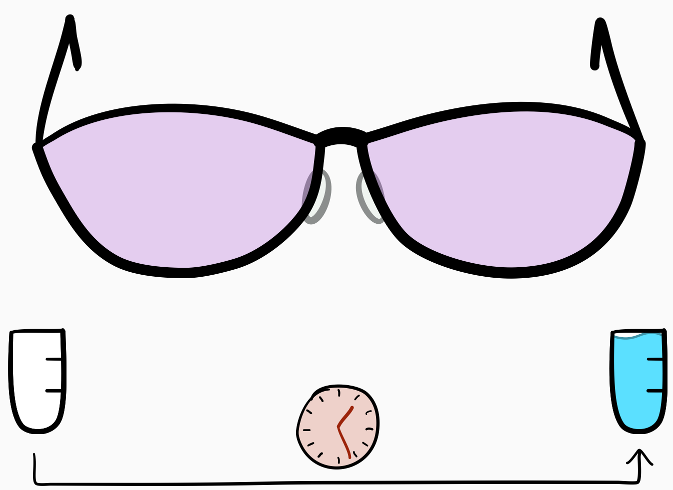
Analyzing *nonmonotonic* **rank**

SRPT: monotonic



Gittins: *nonmonotonic*



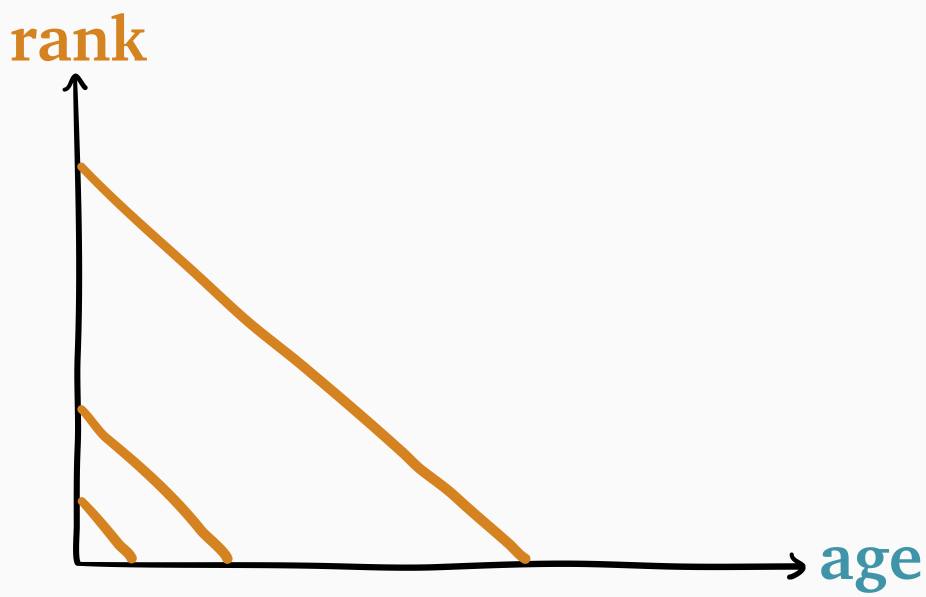


Step 1: compute $\mathbf{E}[W(r)]$

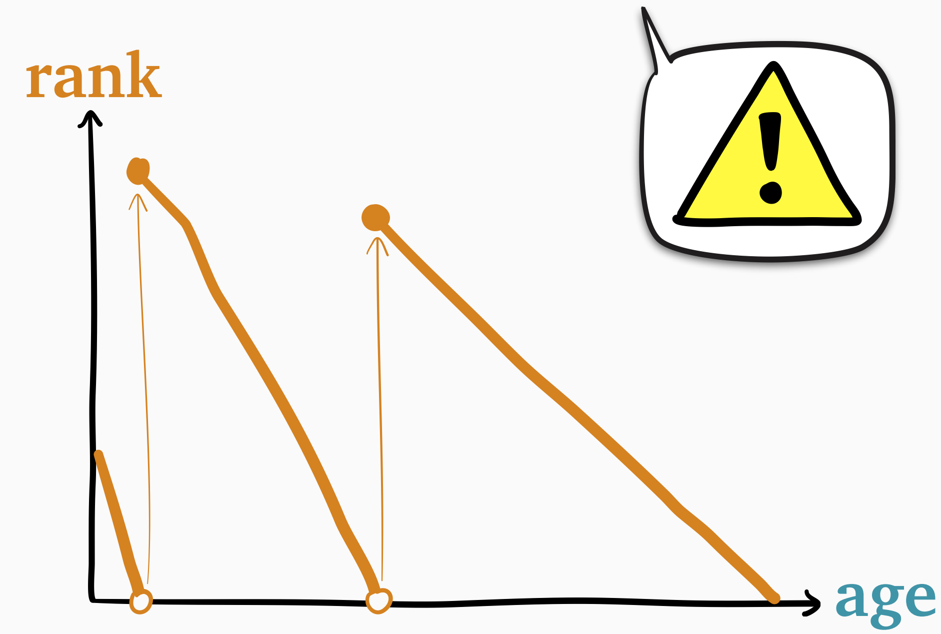
Step 2: $\mathbf{E}[W(r)]$ to $\mathbf{E}[T]$

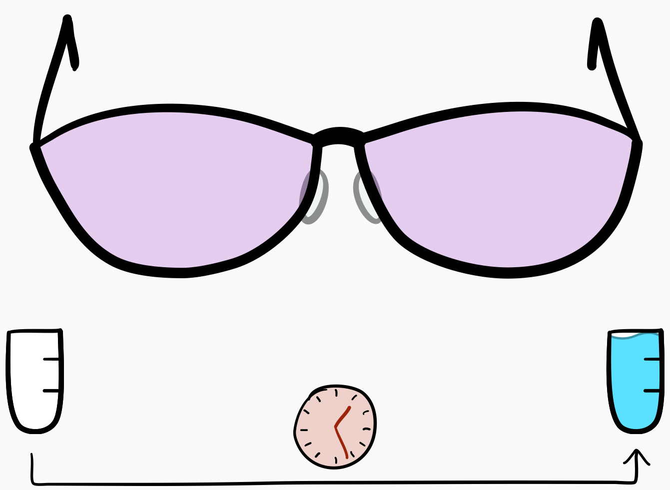
Analyzing *nonmonotonic* **rank**

SRPT: monotonic



Gittins: *nonmonotonic*

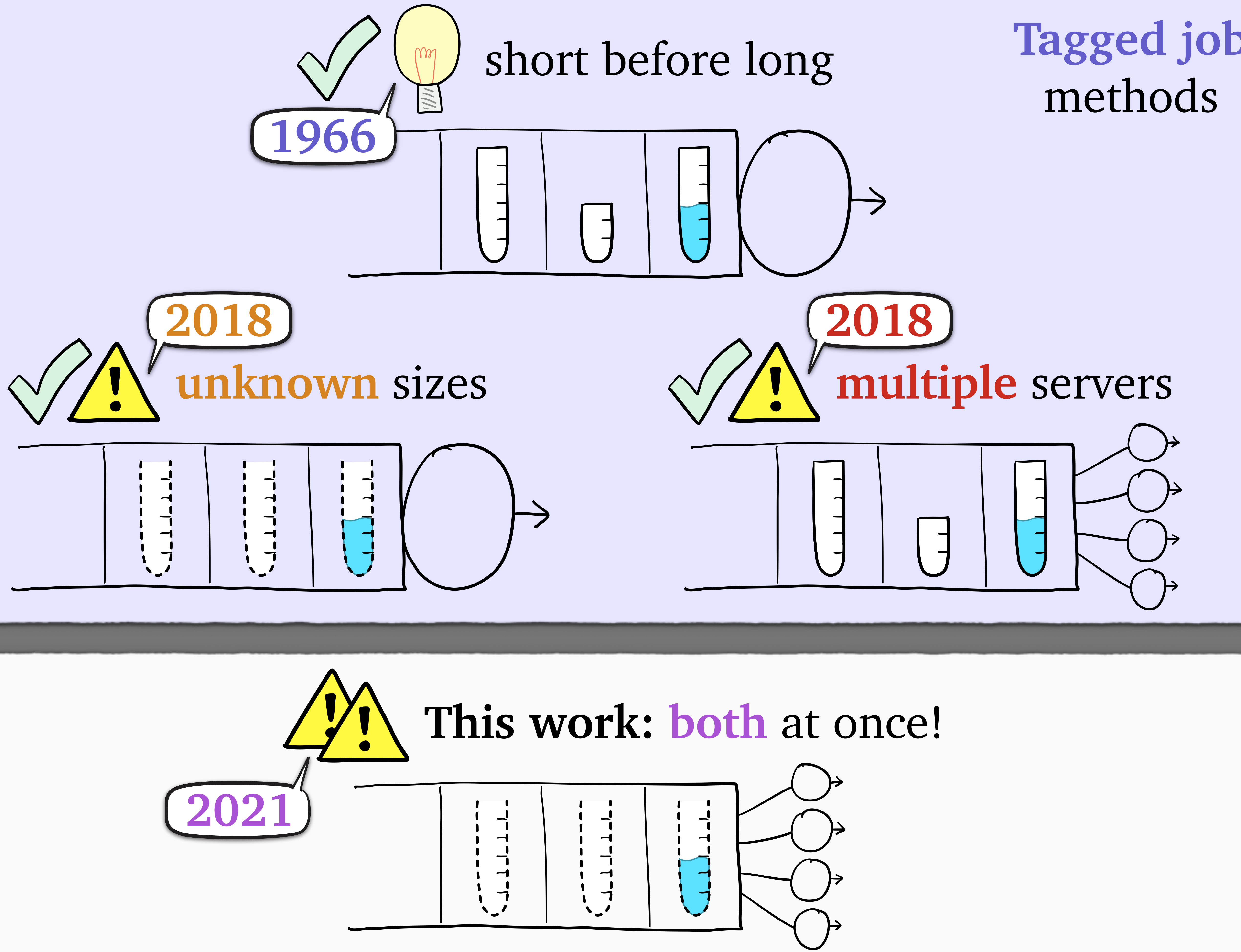


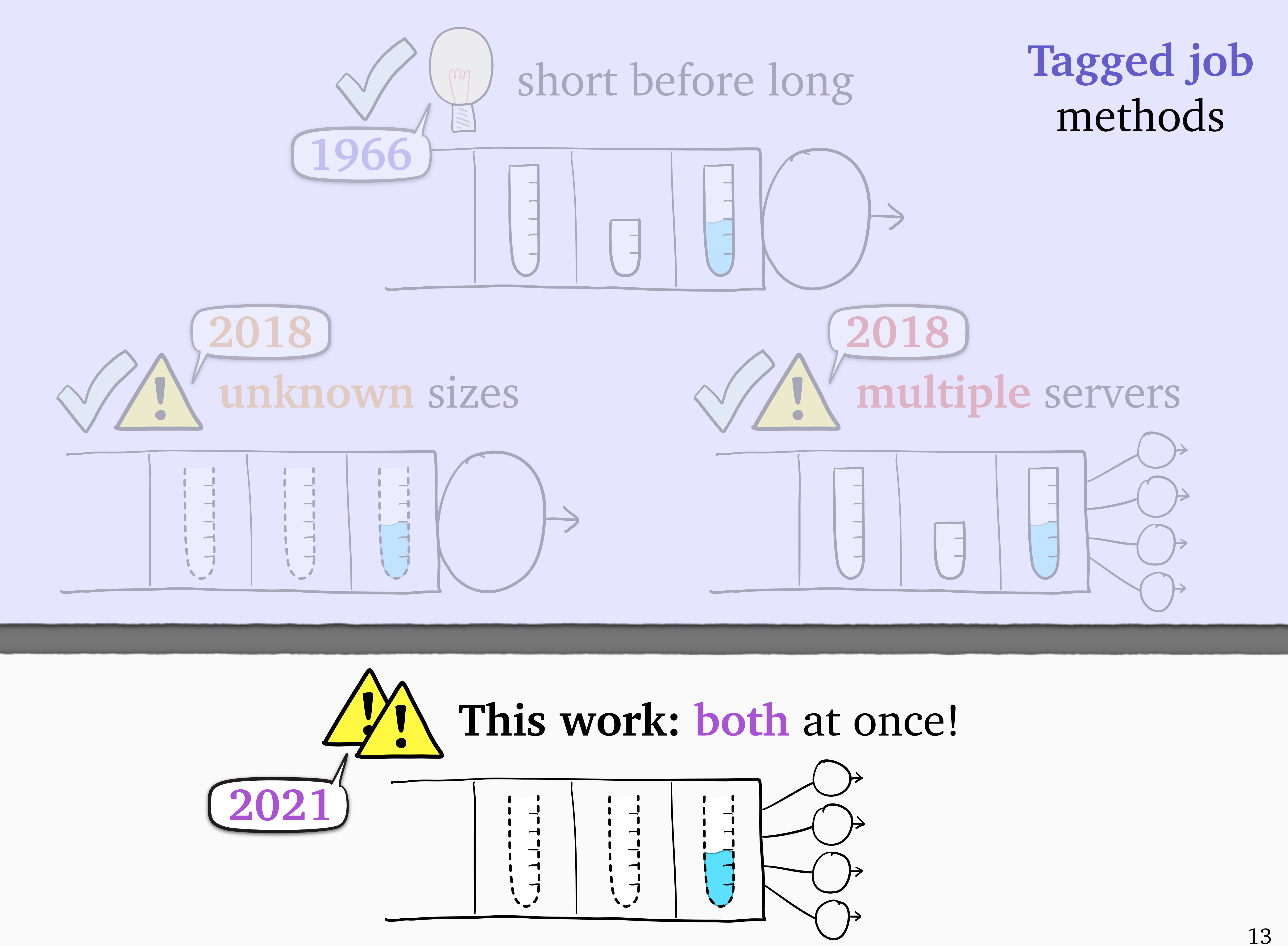


Step 1: compute $\mathbf{E}[W(r)]$

Step 2: $\mathbf{E}[W(r)]$ to $\mathbf{E}[T]$

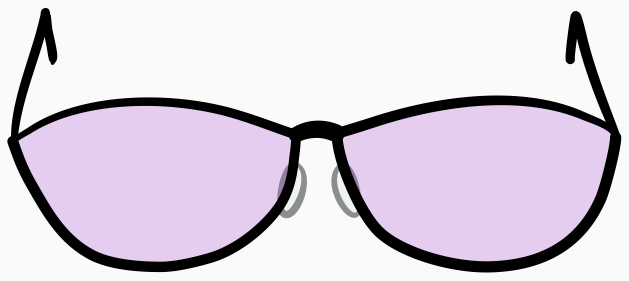
fancy
tagged job


Tagged job methods
1966
short before long
2018
unknown sizes
2018
multiple servers
This work: both at once!
2021


Tagged job
methods
1966
short before long
2018
unknown sizes
2018
multiple servers
This work: both at once!
2021

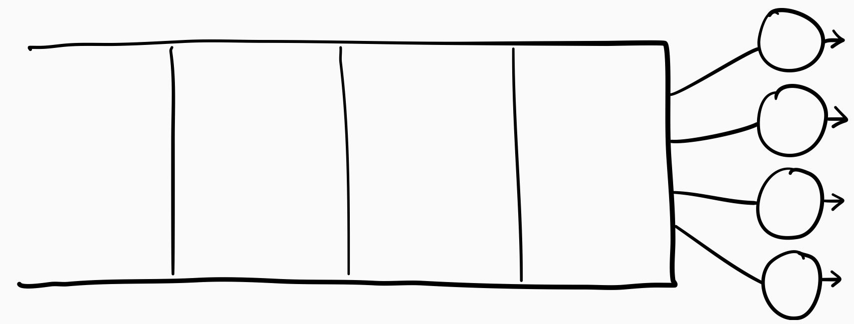
unknown sizes + **multiple** servers

unknown sizes + **multiple** servers

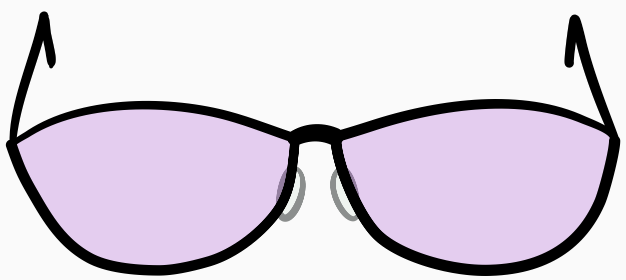


Step 1: compute $\mathbf{E}[W(r)]$

k servers: *intractable*

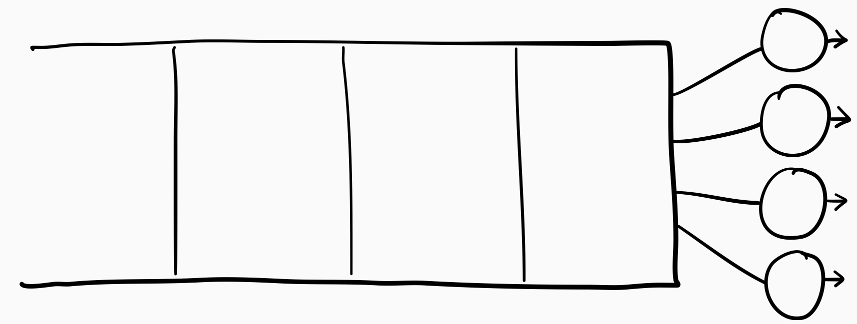


unknown sizes + **multiple** servers

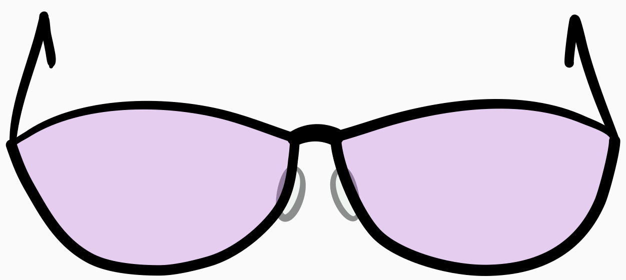


Step 1: compute $\mathbf{E}[W(r)]$

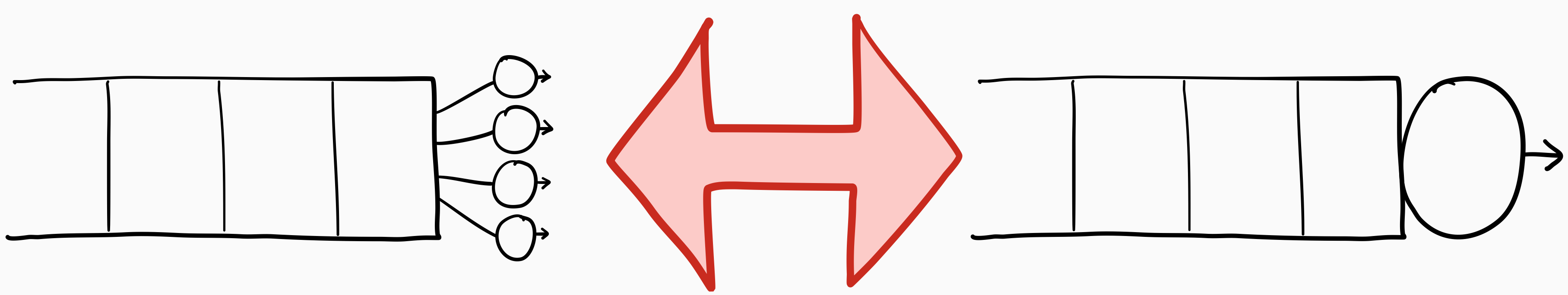
k servers:
intractable



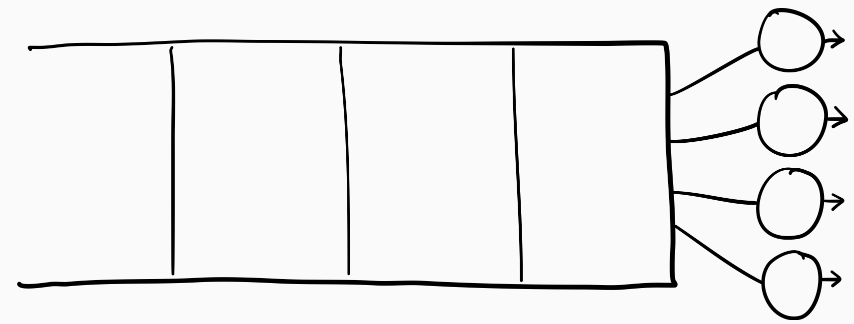
unknown sizes + multiple servers



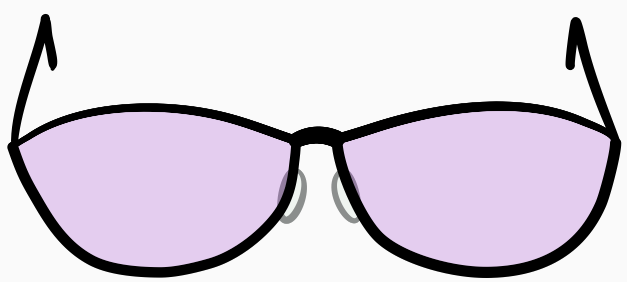
Step 1: compute $\mathbf{E}[W(r)]$



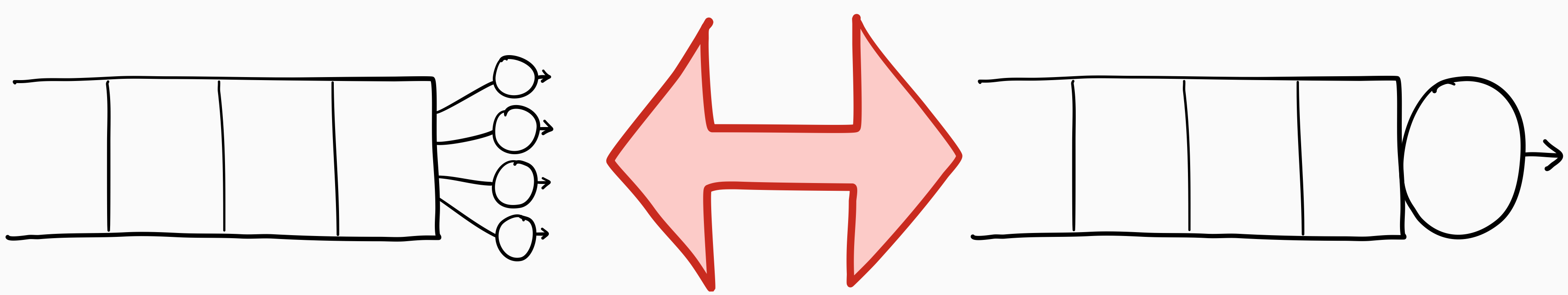
k servers:
intractable



unknown sizes + **multiple** servers

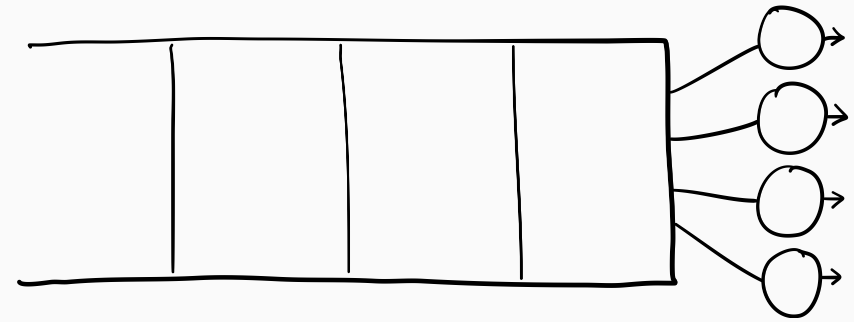


Step 1: compute $\mathbf{E}[W(r)]$

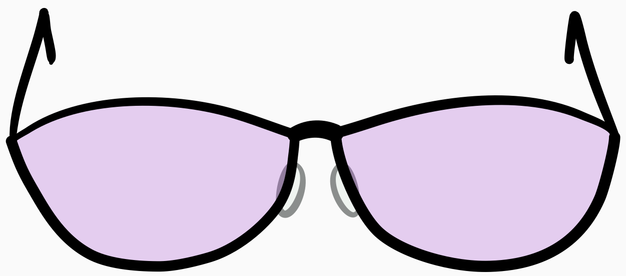


SRPT's $W(r)$ gap $\le kr$

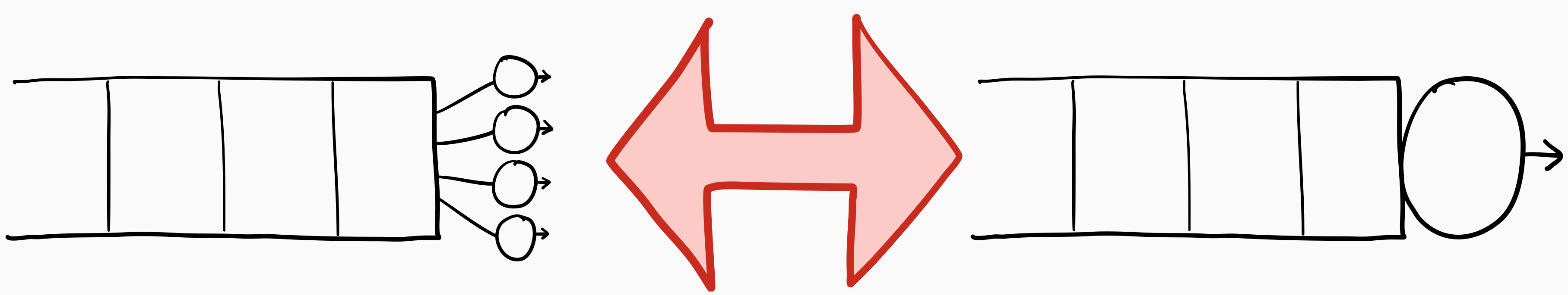
k servers:
intractable



unknown sizes + multiple servers

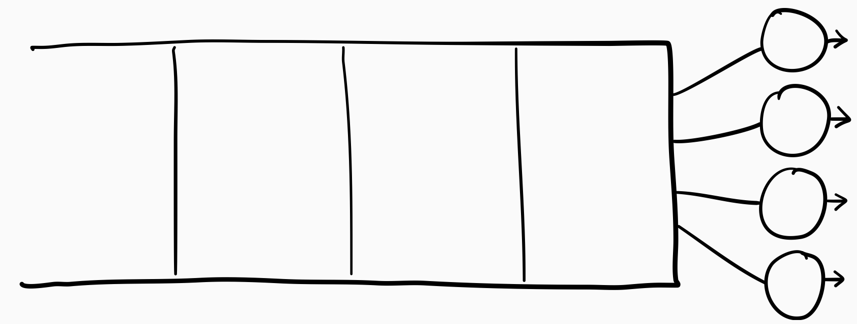


Step 1: compute $\mathbf{E}[W(r)]$

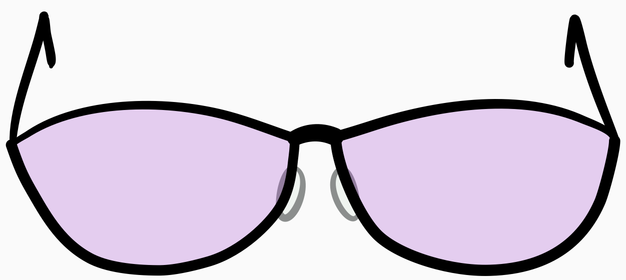


SRPT's $W(r)$ gap $\le kr$
Gittins's $W(r)$ gap $\le k\infty$

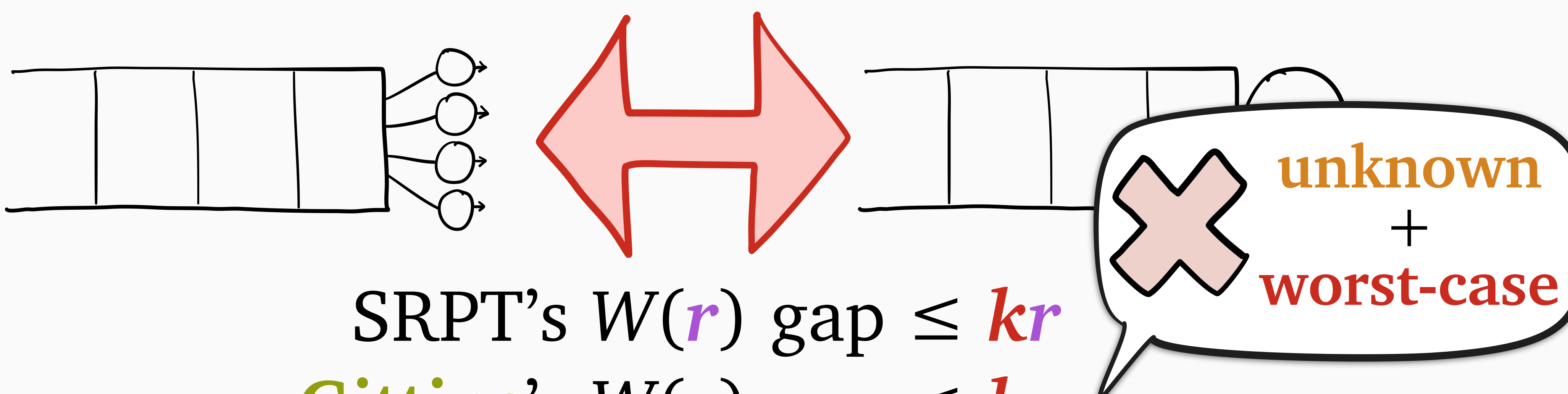
k servers:
intractable



unknown sizes + multiple servers

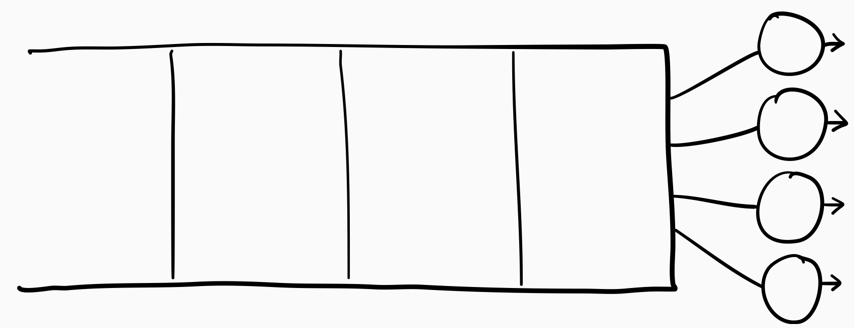


Step 1: compute $\mathbf{E}[W(r)]$

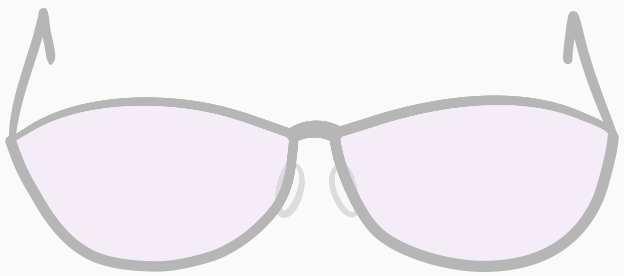


SRPT's $W(r)$ gap $\le kr$
Gittins's $W(r)$ gap $\le k\infty$

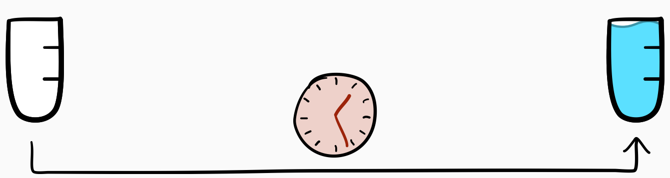
k servers:
intractable



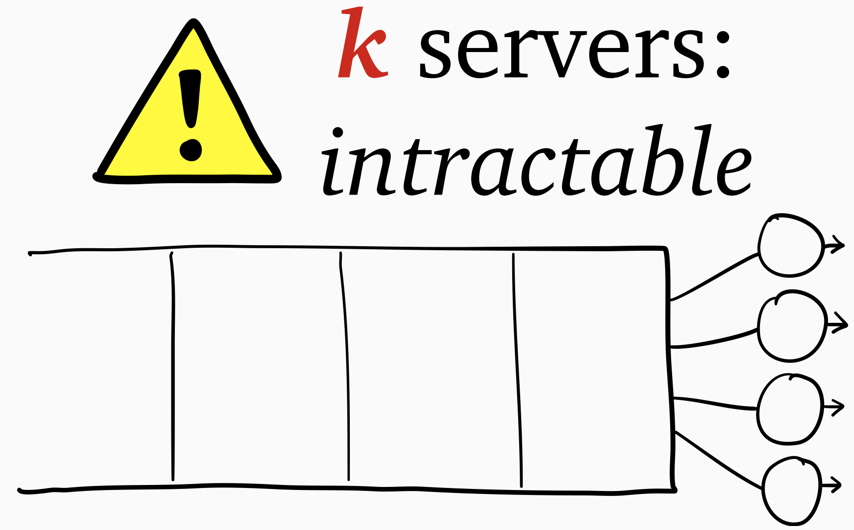
unknown sizes + multiple servers



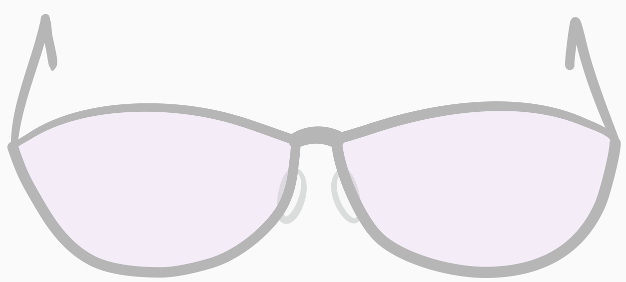
Step 1: compute $\mathbf{E}[W(r)]$



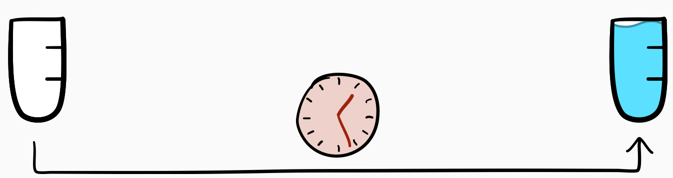
Step 2: $\mathbf{E}[W(r)]$ to $\mathbf{E}[T]$



unknown sizes + multiple servers



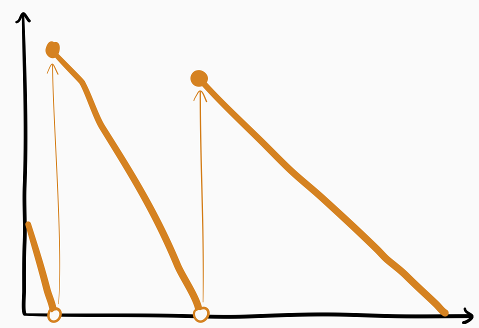
Step 1: compute $\mathbf{E}[W(r)]$



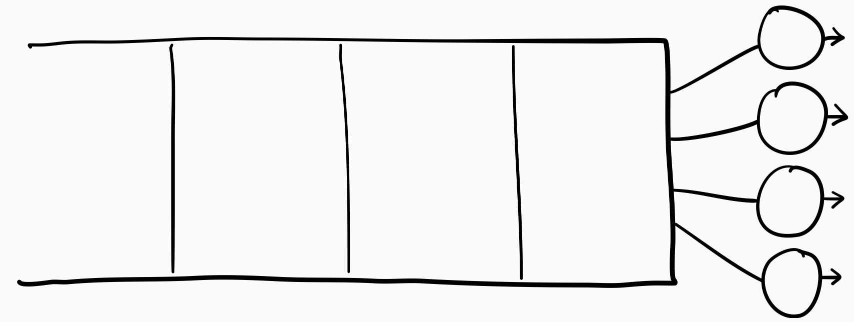
Step 2: $\mathbf{E}[W(r)]$ to $\mathbf{E}[T]$

tagged job
+
worst-case

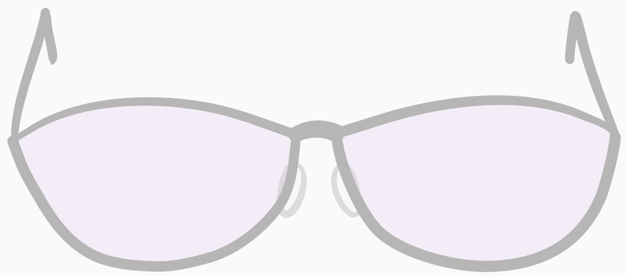
Gittins rank: *nonmonotonic*



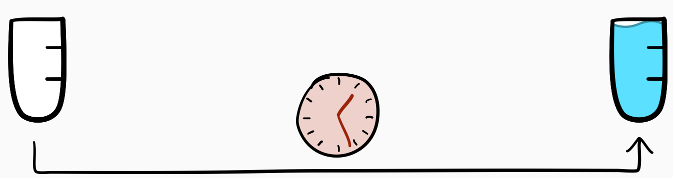
k servers: *intractable*



unknown sizes + **multiple** servers



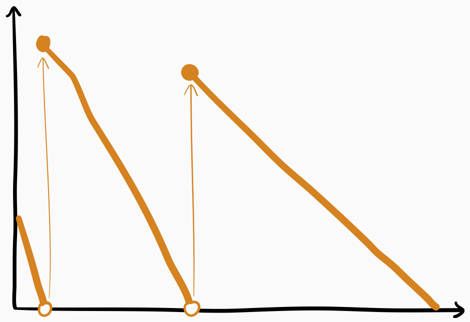
Step 1: compute $\mathbf{E}[W(r)]$



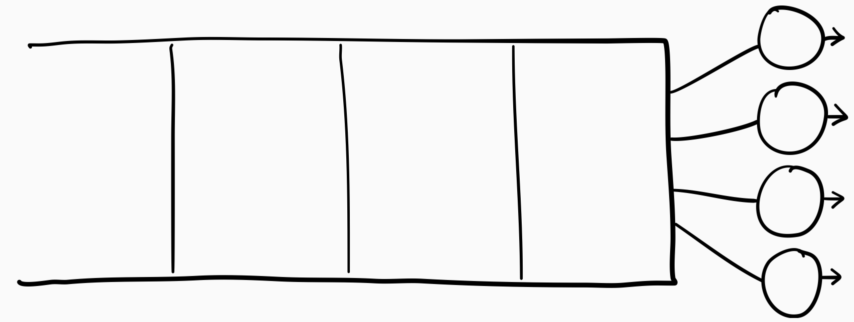
Step 2: $\mathbf{E}[W(r)]$ to $\mathbf{E}[T]$

tagged job
+
worst-case

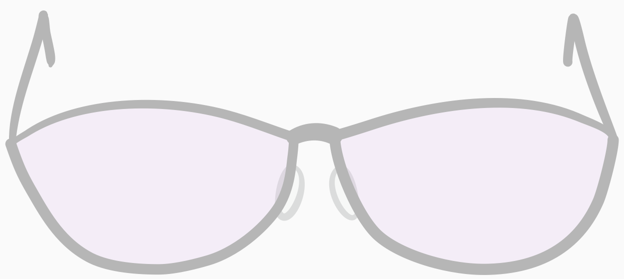
Gittins rank:
nonmonotonic



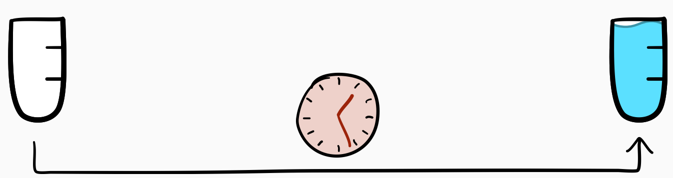
k **servers:**
intractable



unknown sizes + **multiple** servers



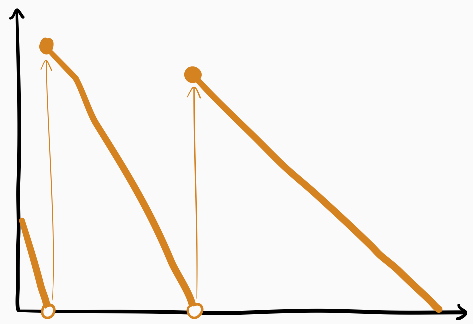
Step 1: compute $\mathbf{E}[W(r)]$



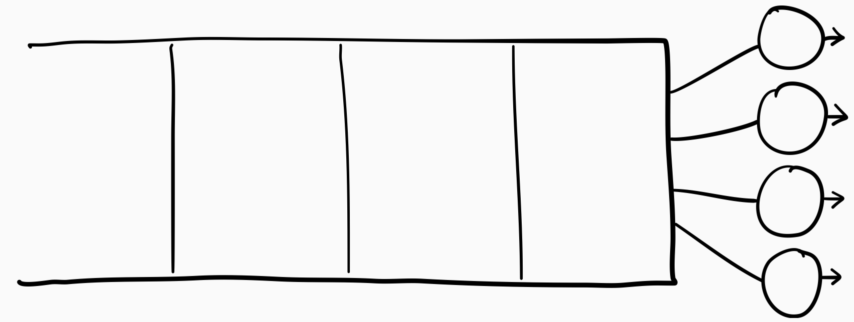
Step 2: $\mathbf{E}[W(r)]$ to $\mathbf{E}[T]$

fancy **tagged job**
\+
worst-case

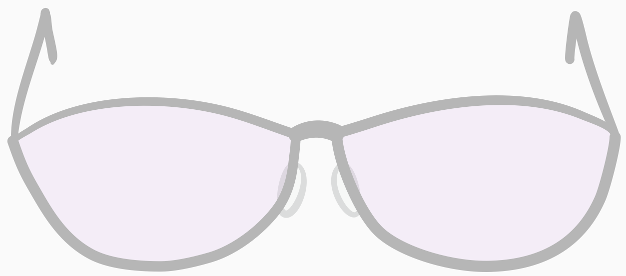
Gittins rank: *nonmonotonic*



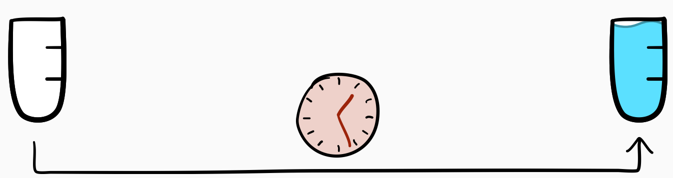
k servers: *intractable*



unknown sizes + multiple servers

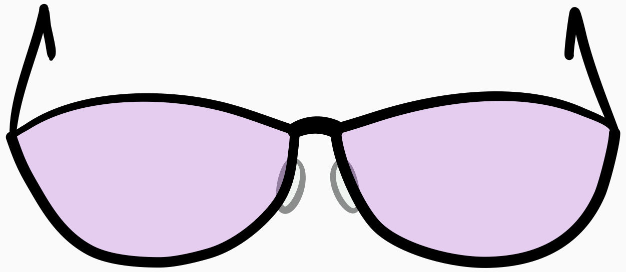


Step 1: compute $\mathbf{E}[W(r)]$

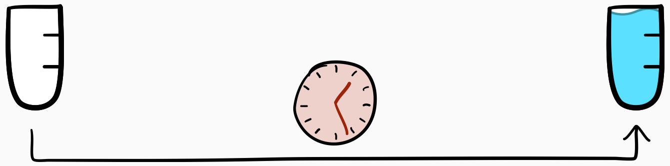


Step 2: $\mathbf{E}[W(r)]$ to $\mathbf{E}[T]$

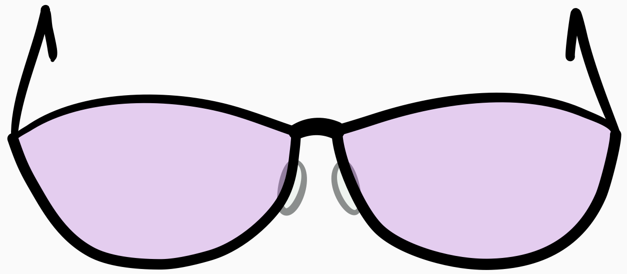
fancy **tagged job**
+
worst-case



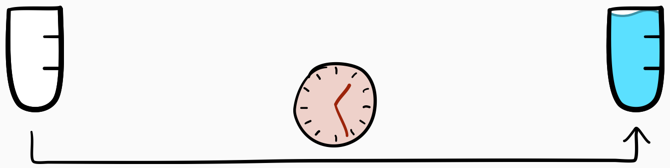
Step 1: compute $\mathbf{E}[W(\boldsymbol{r})]$



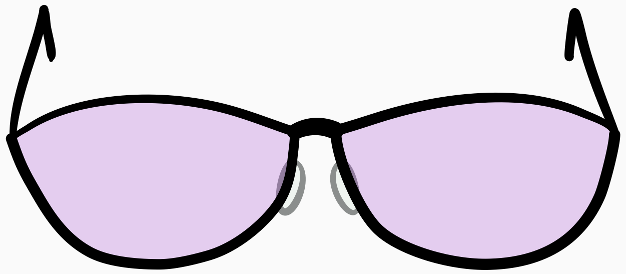
Step 2: $\mathbf{E}[W(\boldsymbol{r})]$ to $\mathbf{E}[T]$



Step 1: compute $\mathbf{E}[W(r)]$
*without **worst-case** steps*

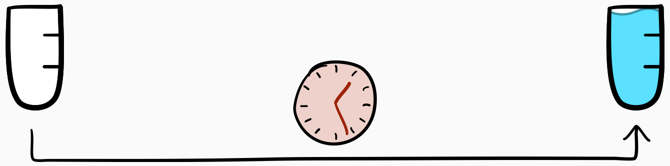


Step 2: $\mathbf{E}[W(r)]$ to $\mathbf{E}[T]$



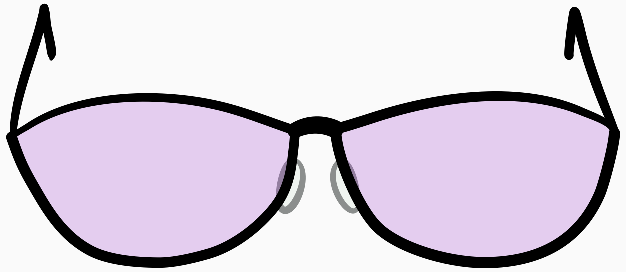
Step 1: compute $\mathbf{E}[W(r)]$

*without **worst-case** steps*



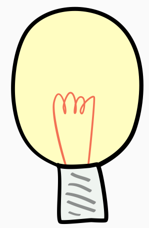
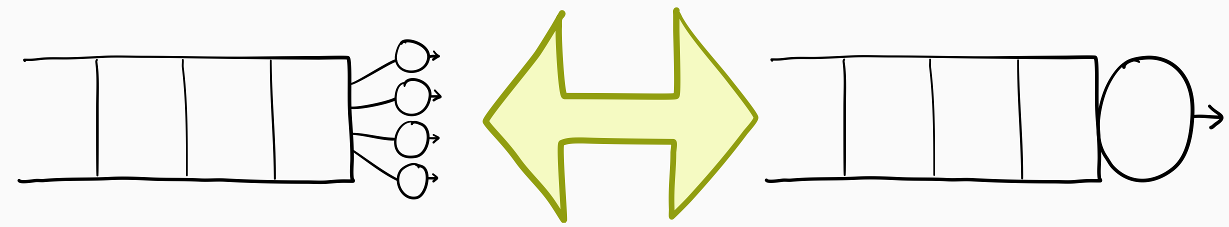
Step 2: $\mathbf{E}[W(r)]$ to $\mathbf{E}[T]$

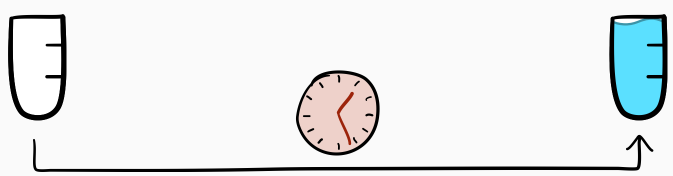
*without **tagged job** method*



Step 1: compute $\mathbf{E}[W(r)]$

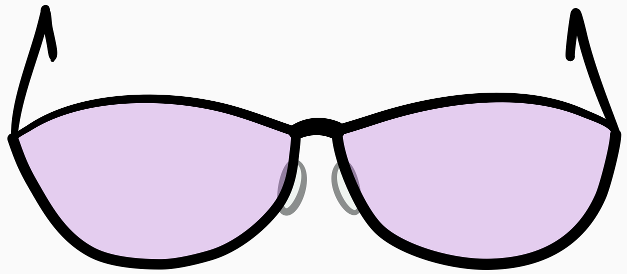
*without **worst-case** steps*

 **Idea:** 



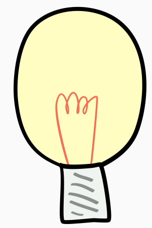
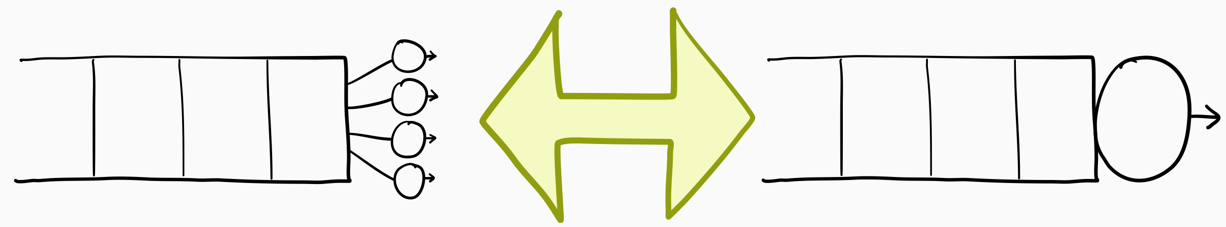
Step 2: $\mathbf{E}[W(r)]$ to $\mathbf{E}[T]$

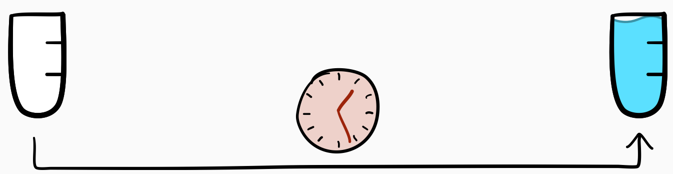
*without **tagged job** method*



Step 1: compute $\mathbf{E}[W(r)]$

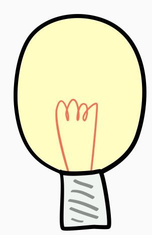
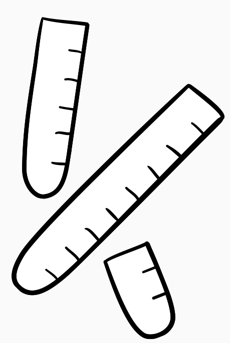
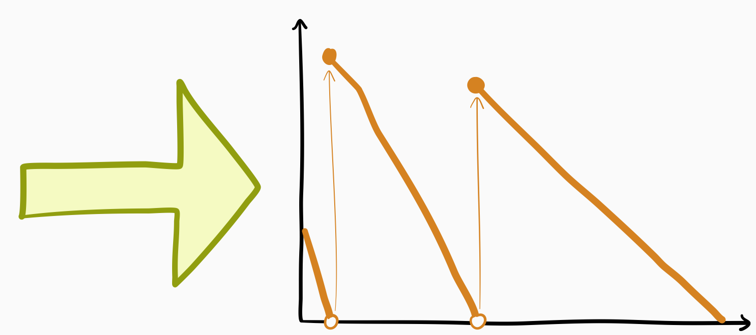
without ***worst-case*** *steps*

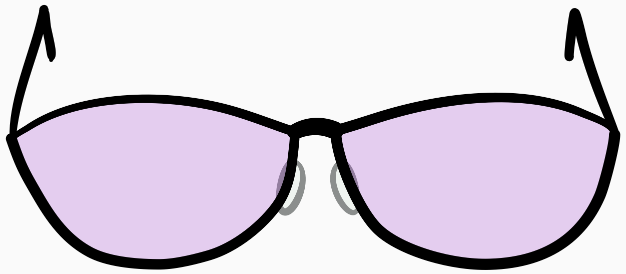
 **Idea:** 



Step 2: $\mathbf{E}[W(r)]$ to $\mathbf{E}[T]$

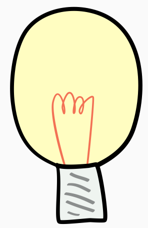
without ***tagged job*** *method*

 **Idea:**  s 

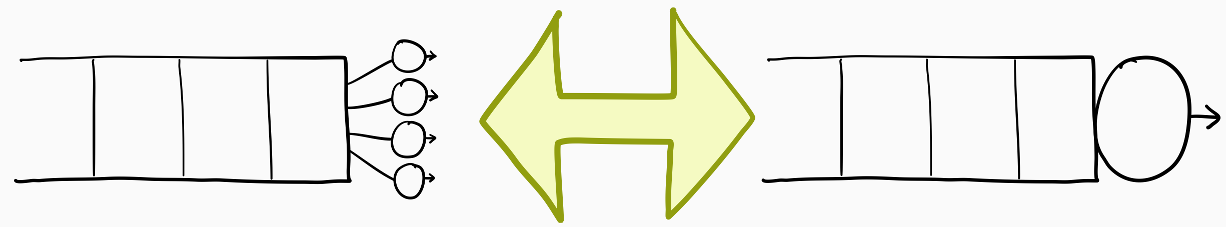


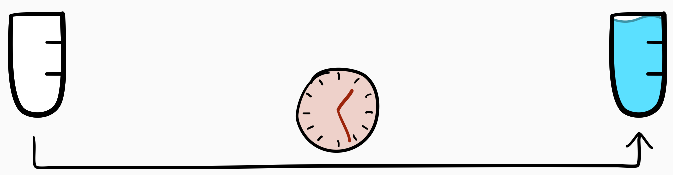
Step 1: compute $\mathbf{E}[W(r)]$

*without **worst-case** steps*



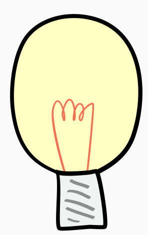
Idea:



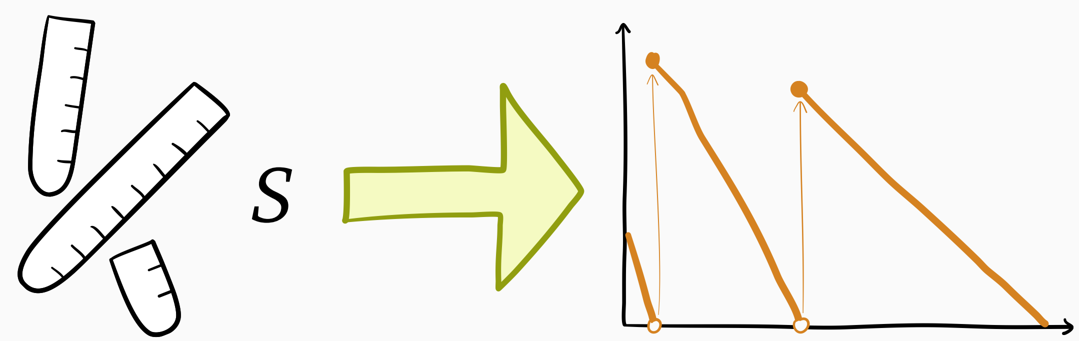


Step 2: $\mathbf{E}[W(r)]$ to $\mathbf{E}[T]$

*without **tagged job** method*



Idea:



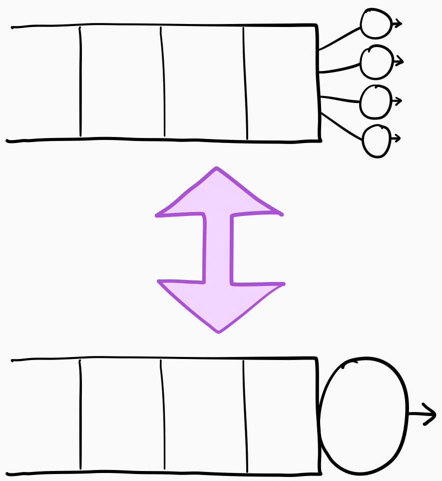
Our contribution:

new **exact formulas** for both steps

Step 1: compute $\mathbf{E}[W(\boldsymbol{r})]$

Step 2: $\mathbf{E}[W(\boldsymbol{r})]$ to $\mathbf{E}[T]$

Step 1: compute $\mathbf{E}[W(r)]$

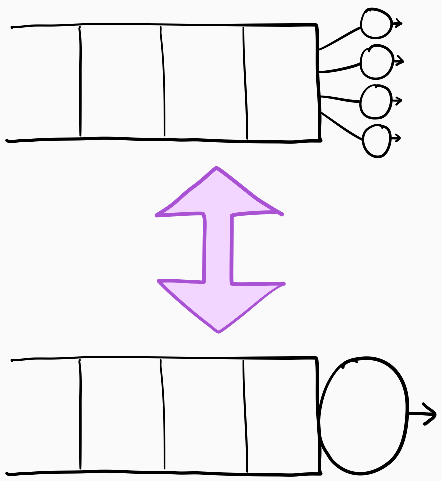


Work Decomposition Law

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

Step 2: $\mathbf{E}[W(r)]$ to $\mathbf{E}[T]$

Step 1: compute $\mathbf{E}[W(r)]$



Work Decomposition Law

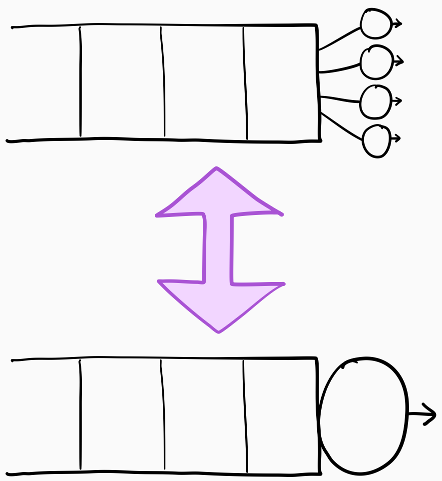
$$\mathbf{E}[W_k(r)] = \mathbf{E}[W_1(r)] + \mathbf{E}[\text{“}< k \text{ jobs' } r\text{-work”}]$$

Step 2: $\mathbf{E}[W(r)]$ to $\mathbf{E}[T]$

WINE

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

Step 1: compute $\mathbf{E}[W(r)]$



Work Decomposition Law

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

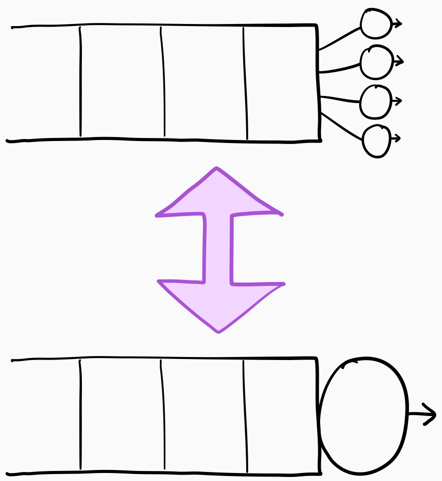
Step 2: $\mathbf{E}[W(r)]$ to $\mathbf{E}[T]$

WINE

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

specific to
Gittins's rank

Step 1: compute $\mathbf{E}[W(r)]$



Work Decomposition Law

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

Step 2: $\mathbf{E}[W(r)]$ to $\mathbf{E}[T]$

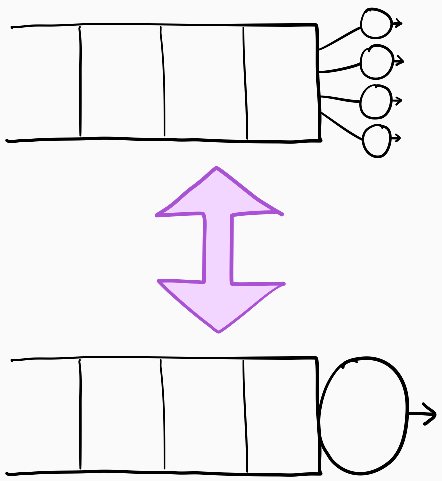
WINE

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

specific to Gittins's rank

Impact 1:
first bound on $\mathbf{E}[T_{\text{Gittins-}k}]$

Step 1: compute $\mathbf{E}[W(r)]$



Work Decomposition Law

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

Step 2: $\mathbf{E}[W(r)]$ to $\mathbf{E}[T]$

WINE

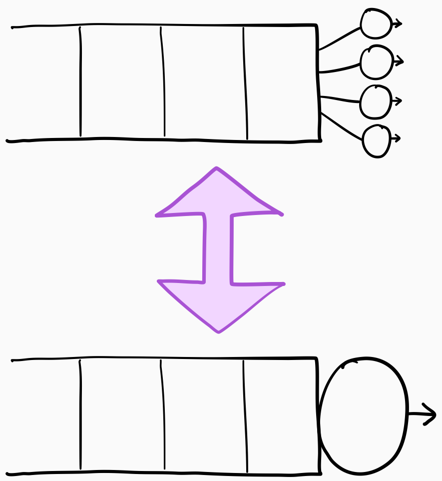
specific to Gittins's rank

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

Impact 1:
first bound on $\mathbf{E}[T_{\text{Gittins-}k}]$

Impact 2:
both generalize beyond M/G/k

Step 1: compute $\mathbf{E}[W(r)]$



Work Decomposition Law

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

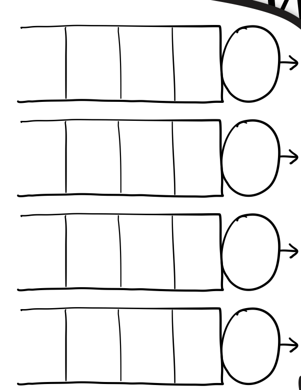
Step 2: $\mathbf{E}[W(r)]$ to $\mathbf{E}[T]$

WINE

$\lambda \mathbf{E}$ … $\int^{\infty}$ … $\mathbf{E}[W(r)]$ … $\mathrm{d}r$

specific to Gittins's rank

Example:
load-balancing
+
scheduling



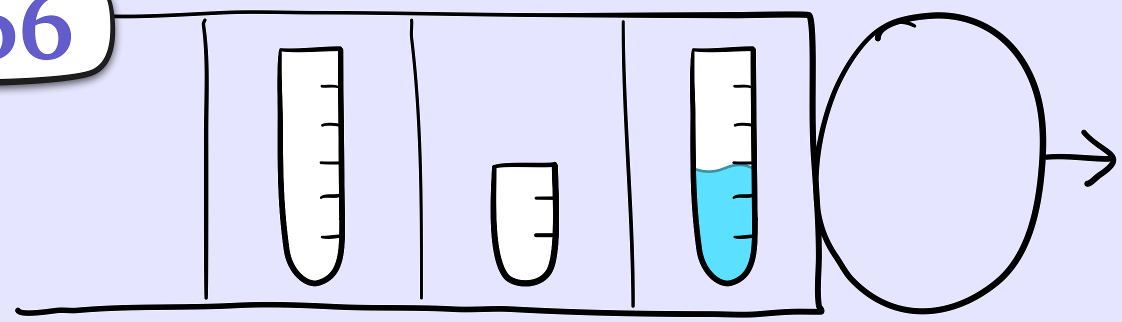
Impact 1:
first bound on $\mathbf{E}[T_{\text{Gittins-}k}]$

Impact 2:
both generalize beyond M/G/k

Tagged job methods

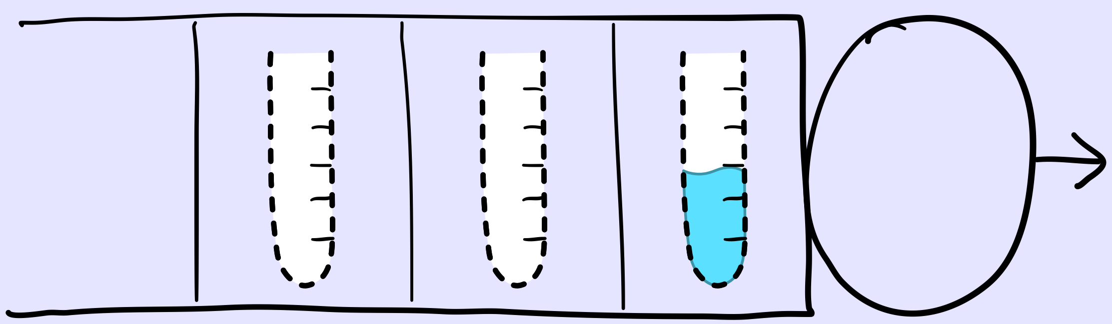
1966

short before long



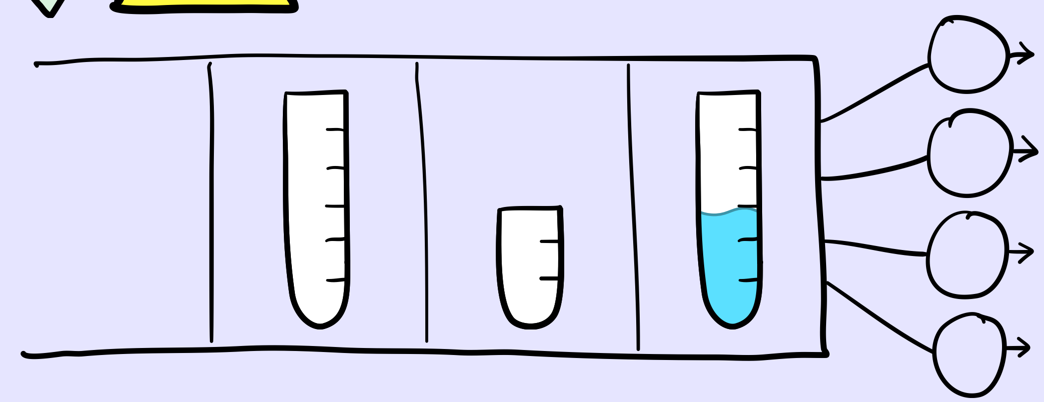
2018

unknown sizes



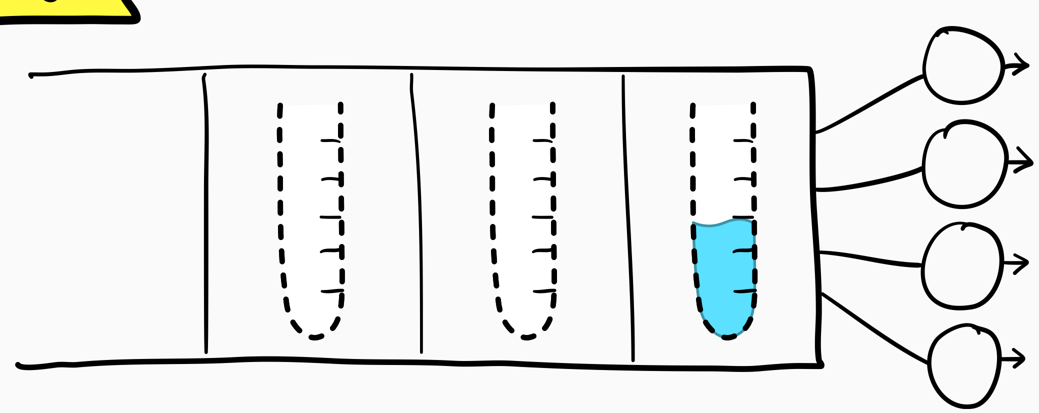
2018

multiple servers



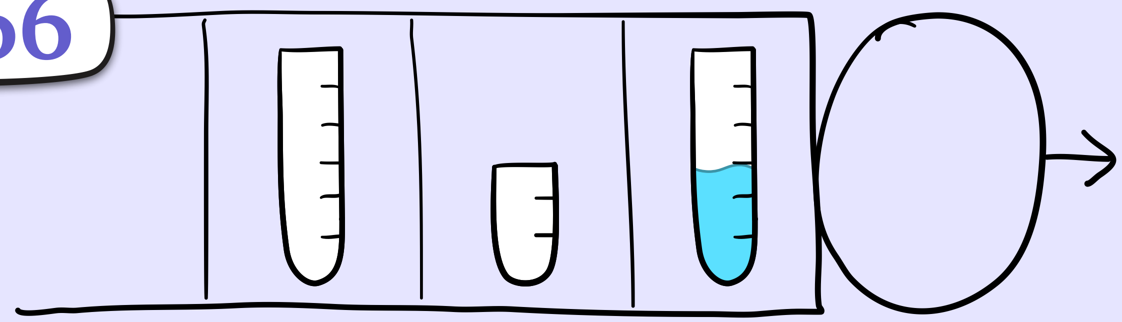
This work: both at once!

SIGMETRICS 2021

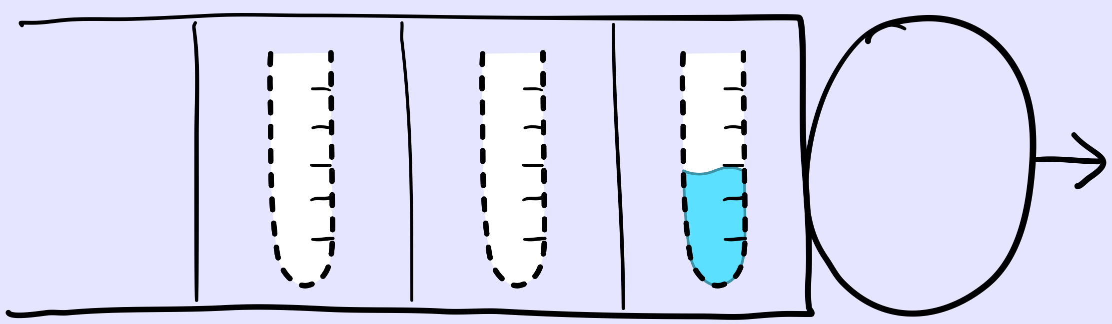


Tagged job methods

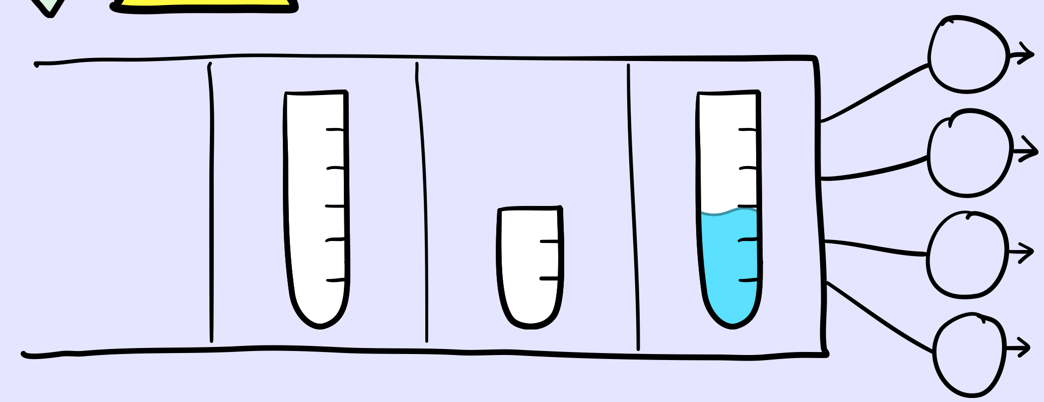
1966 short before long



2018 **unknown** sizes

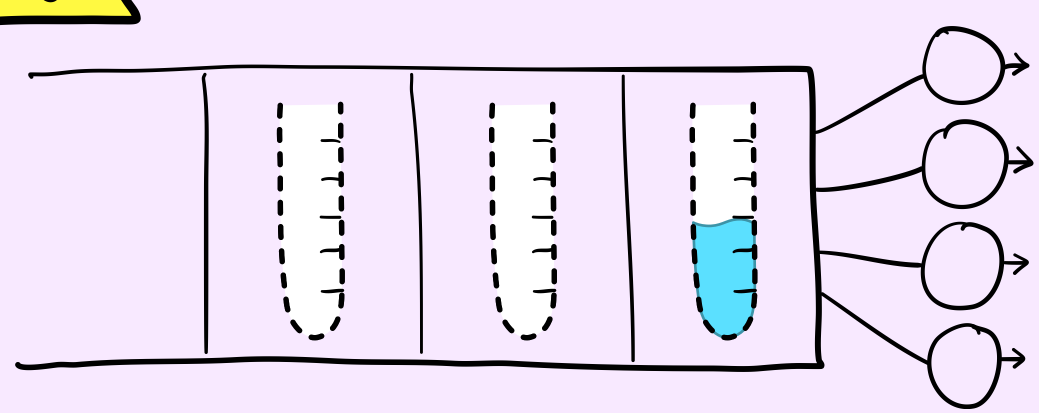


2018 **multiple** servers

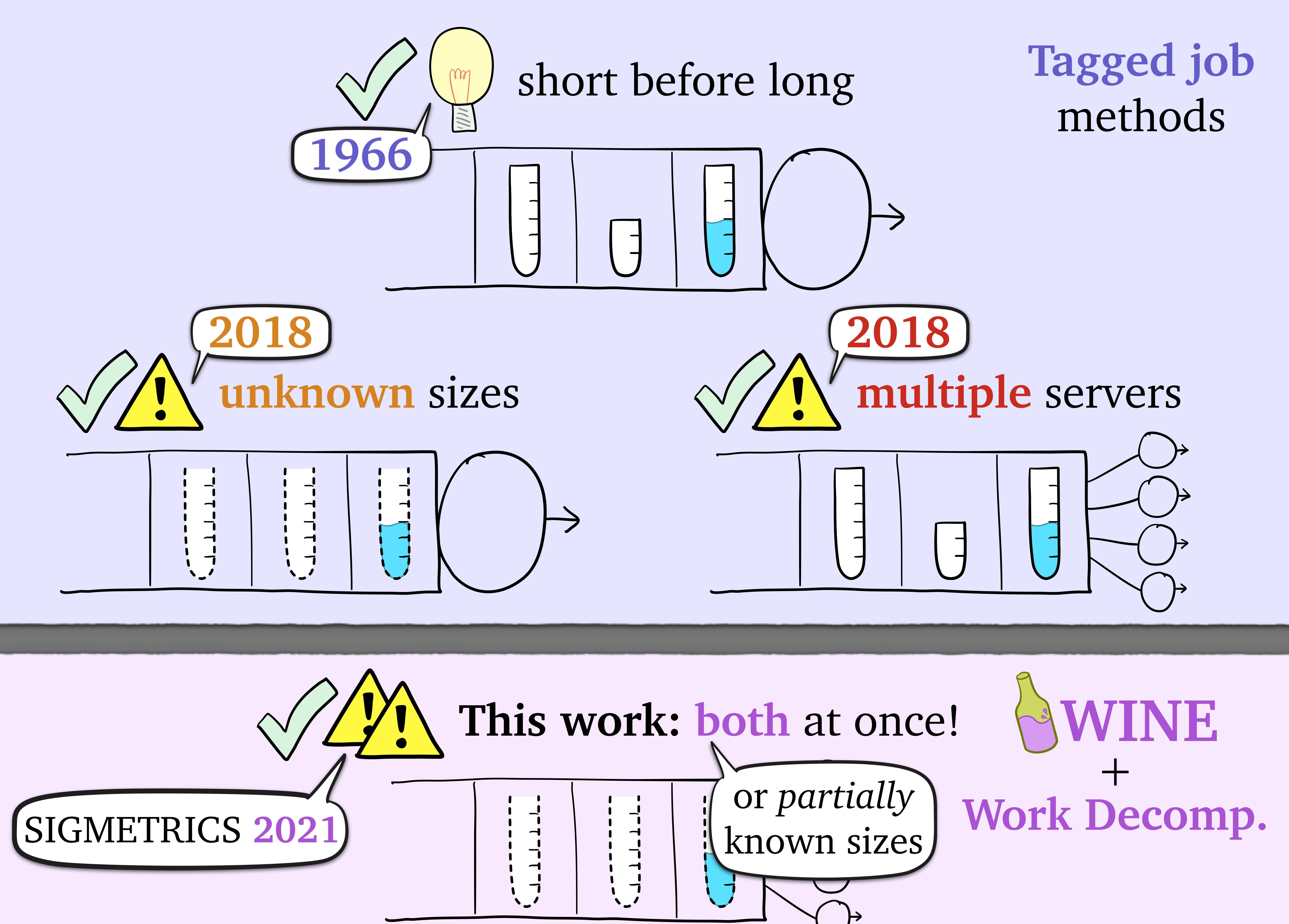


This work: both at once!

SIGMETRICS 2021



WINE + **Work Decomp.**



Tagged job methods
1966
short before long
2018
unknown sizes
2018
multiple servers
This work: both at once!
SIGMETRICS 2021
or partially known sizes
WINE
+
Work Decomp.