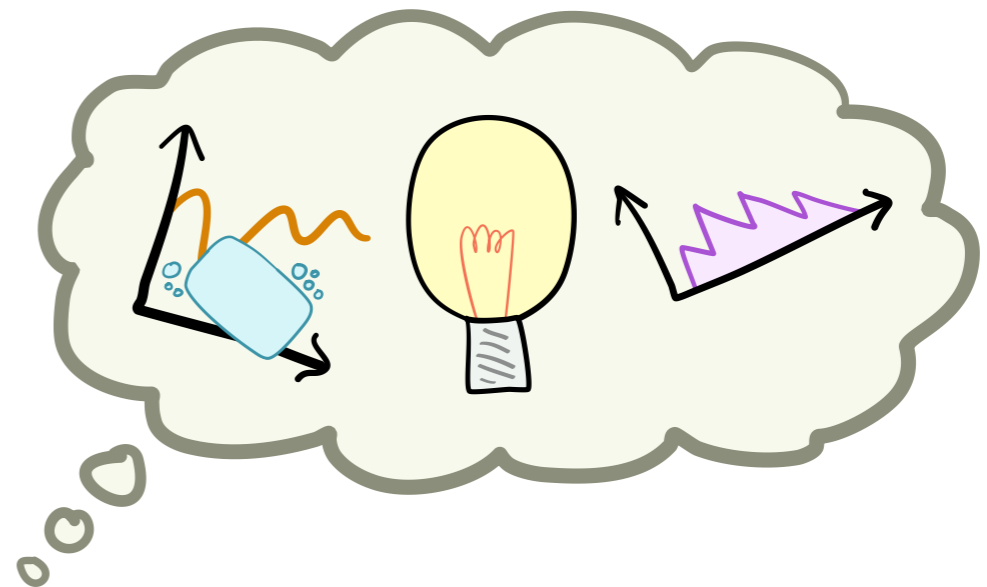


Near-Optimal Scheduling: *Towards a Unified Theory*

Ziv Scully
Carnegie Mellon University



Collaborators



Mor Harchol-Balter (CMU)



Alan Scheller-Wolf (CMU)



Isaac Grosf (CMU)

Adam Wierman (Caltech)

Onno Boxma (TU/e)

Jan-Pieter Dorsman (UvA)

Lucas van Kreveld (UvA)

Queues in Computer Systems

Queues in Computer Systems

↖ Queueing system: *jobs* waiting for *service*

Queues in Computer Systems

 Queueing system: *jobs* waiting for *service*

- File servers
 - *Jobs*: file requests
 - *Service*: load and send contents

Queues in Computer Systems

 Queueing system: *jobs* waiting for *service*

- File servers
 - *Jobs*: file requests
 - *Service*: load and send contents
- Databases
 - *Jobs*: SQL queries
 - *Service*: execute and send result

Queues in Computer Systems

 Queueing system: *jobs* waiting for *service*

- File servers
 - *Jobs*: file requests
 - *Service*: load and send contents
- Databases
 - *Jobs*: SQL queries
 - *Service*: execute and send result
- Network switches
 - *Jobs*: packet flows
 - *Service*: transmit all packets

Queues in Computer Systems

 Queueing system: *jobs* waiting for *service*

- File servers
 - *Jobs*: file requests
 - *Service*: load and send contents
- Databases
 - *Jobs*: SQL queries
 - *Service*: execute and send result
- Network switches
 - *Jobs*: packet flows
 - *Service*: transmit all packets
- Operating systems
 - *Jobs*: threads
 - *Service*: run on a CPU core

Queues in Computer Systems

↖ Queueing system: *jobs* waiting for *service*

- File servers
 - *Jobs*: file requests
 - *Service*: load and send contents
- Databases
 - *Jobs*: SQL queries
 - *Service*: execute and send result
- Network switches
 - *Jobs*: packet flows
 - *Service*: transmit all packets
- Operating systems
 - *Jobs*: threads
 - *Service*: run on a CPU core

Queueing theory: studies the *mathematical essence* of queueing systems

Queues in Computer Systems

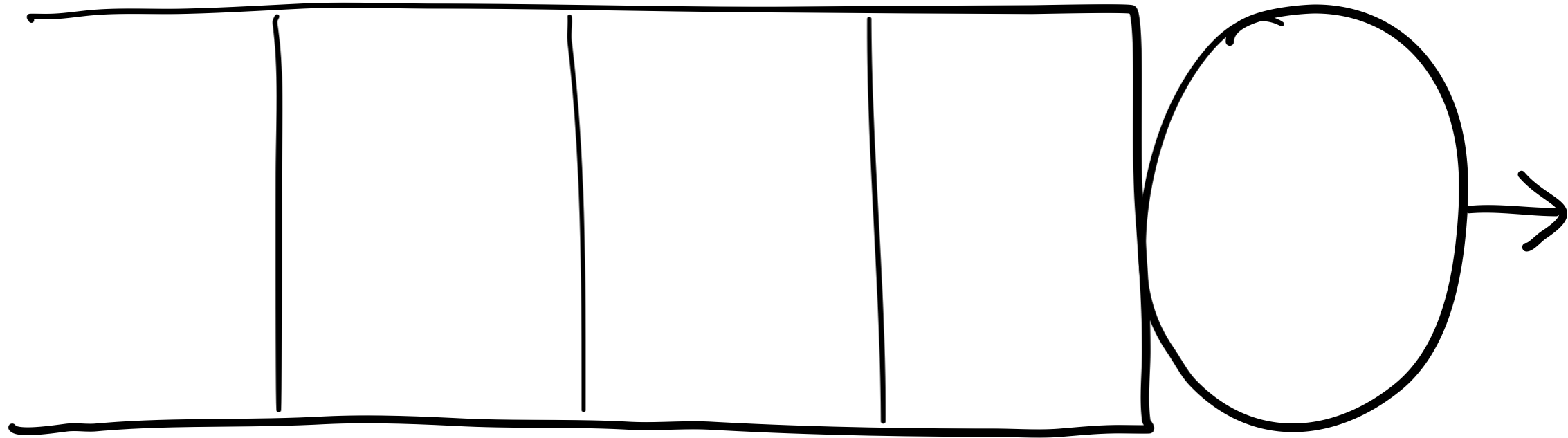
↖ Queueing system: *jobs* waiting for *service*

- File servers
 - *Jobs*: file requests
 - *Service*: load and send contents
- Databases
 - *Jobs*: SQL queries
 - *Service*: execute and send result
- Network switches
 - *Jobs*: packet flows
 - *Service*: transmit all packets
- Operating systems
 - *Jobs*: threads
 - *Service*: run on a CPU core

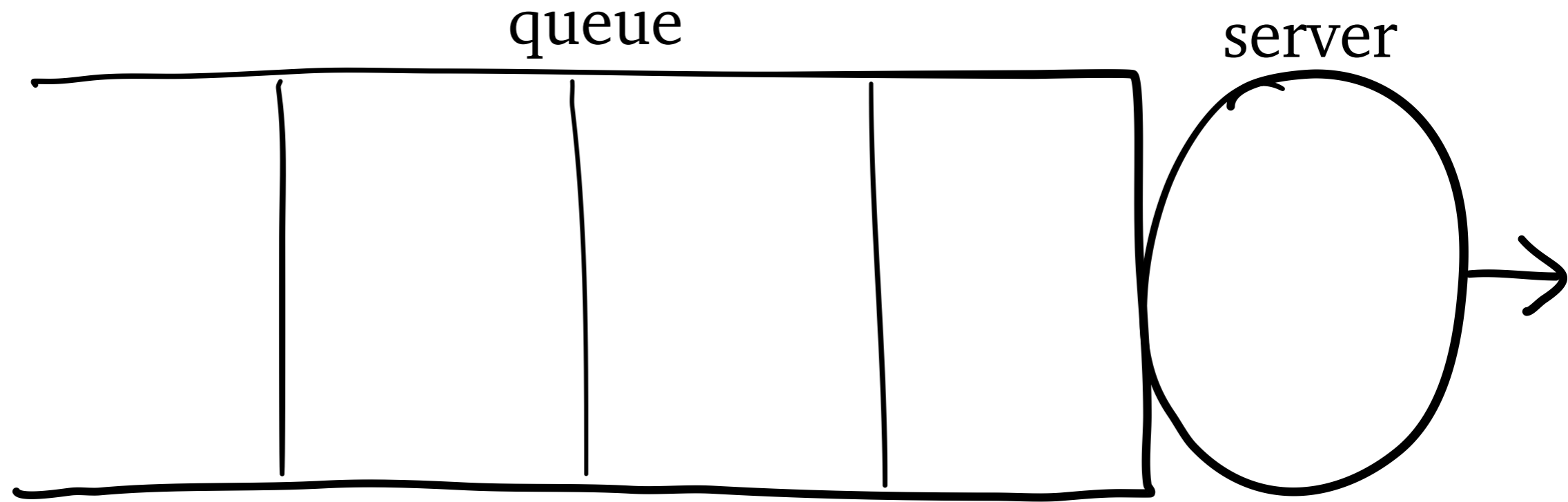
Queueing theory: studies the *mathematical essence* of queueing systems

↙ ↘
jobs service

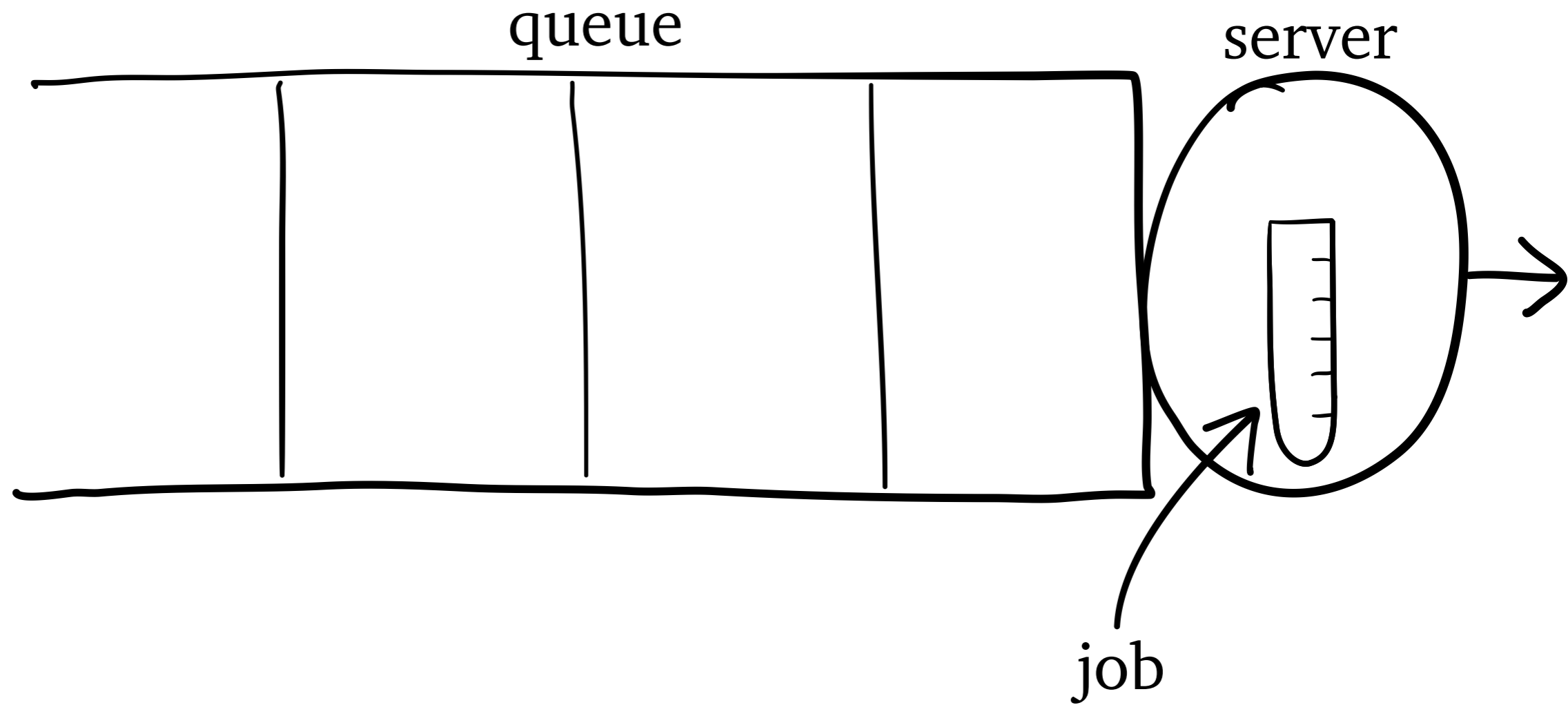
M/G/1 Queueing Model



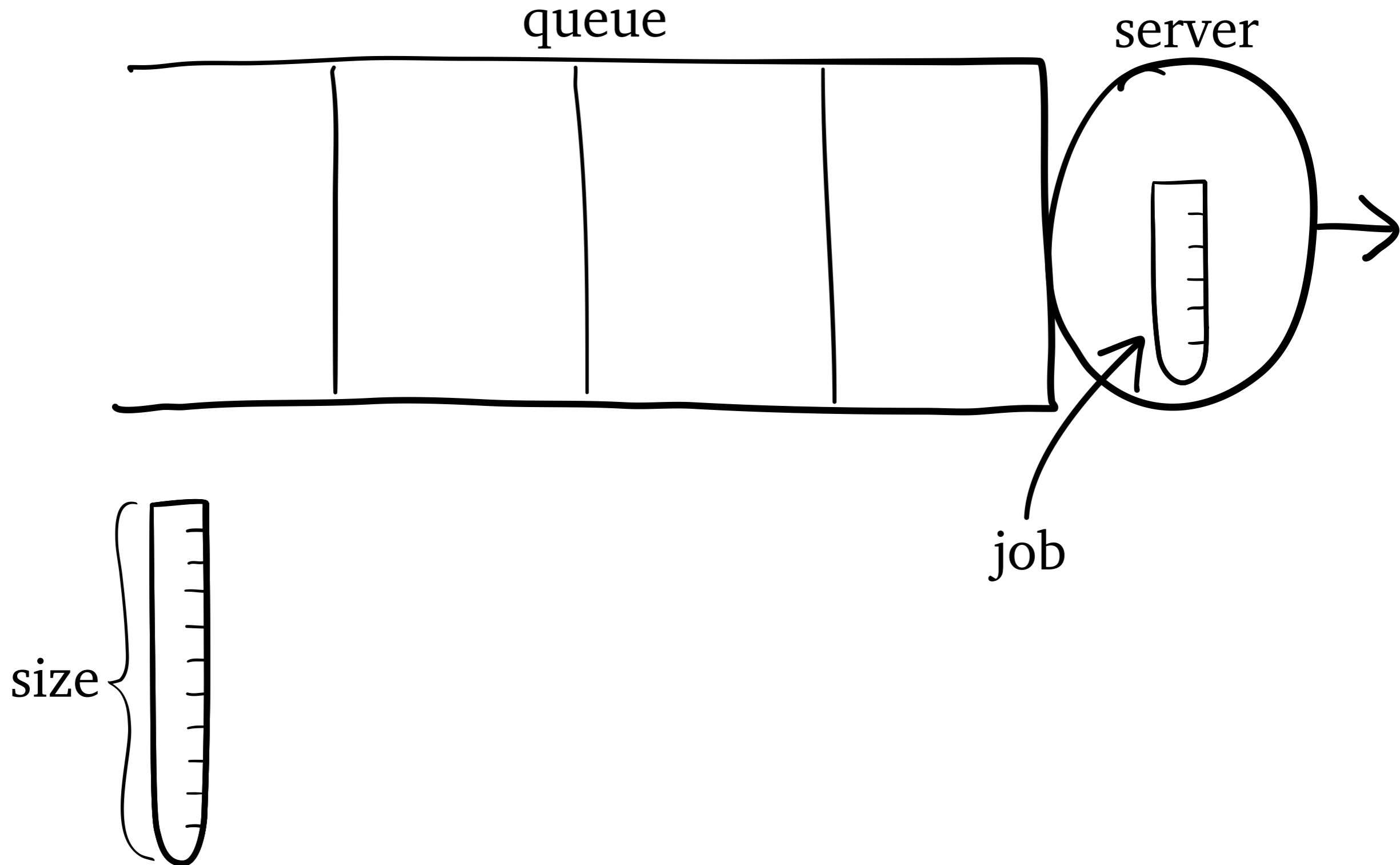
M/G/1 Queueing Model



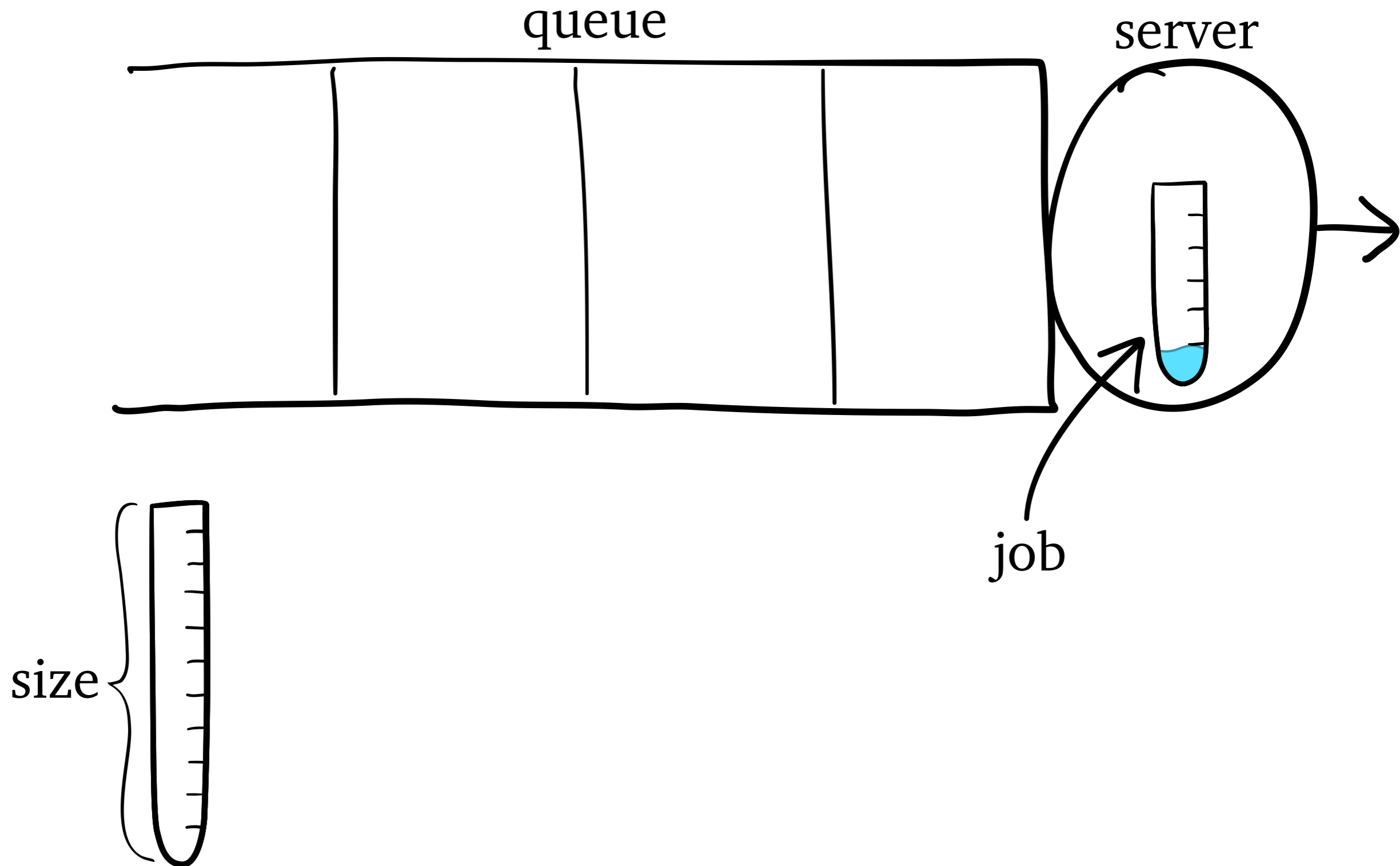
M/G/1 Queueing Model



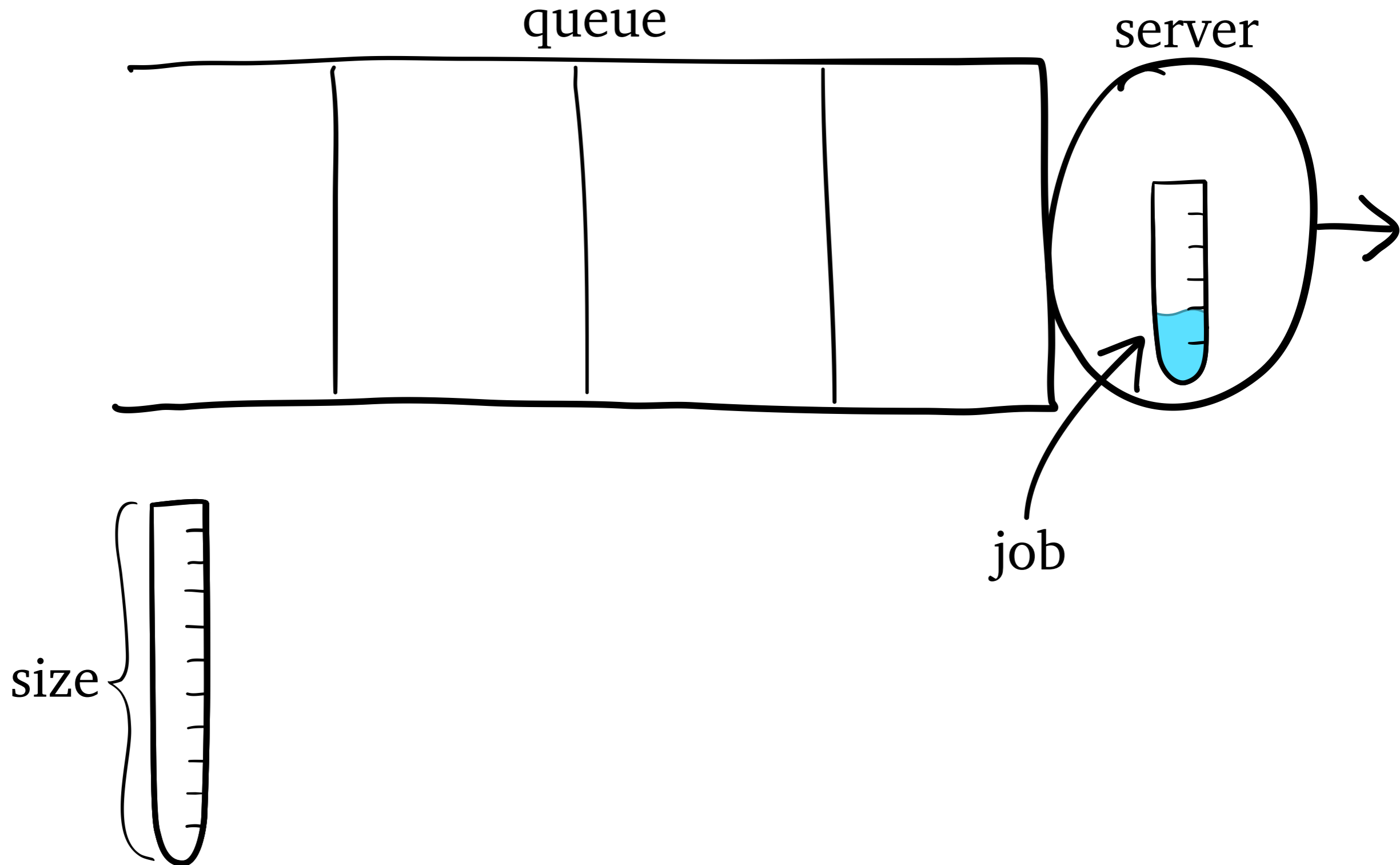
M/G/1 Queueing Model



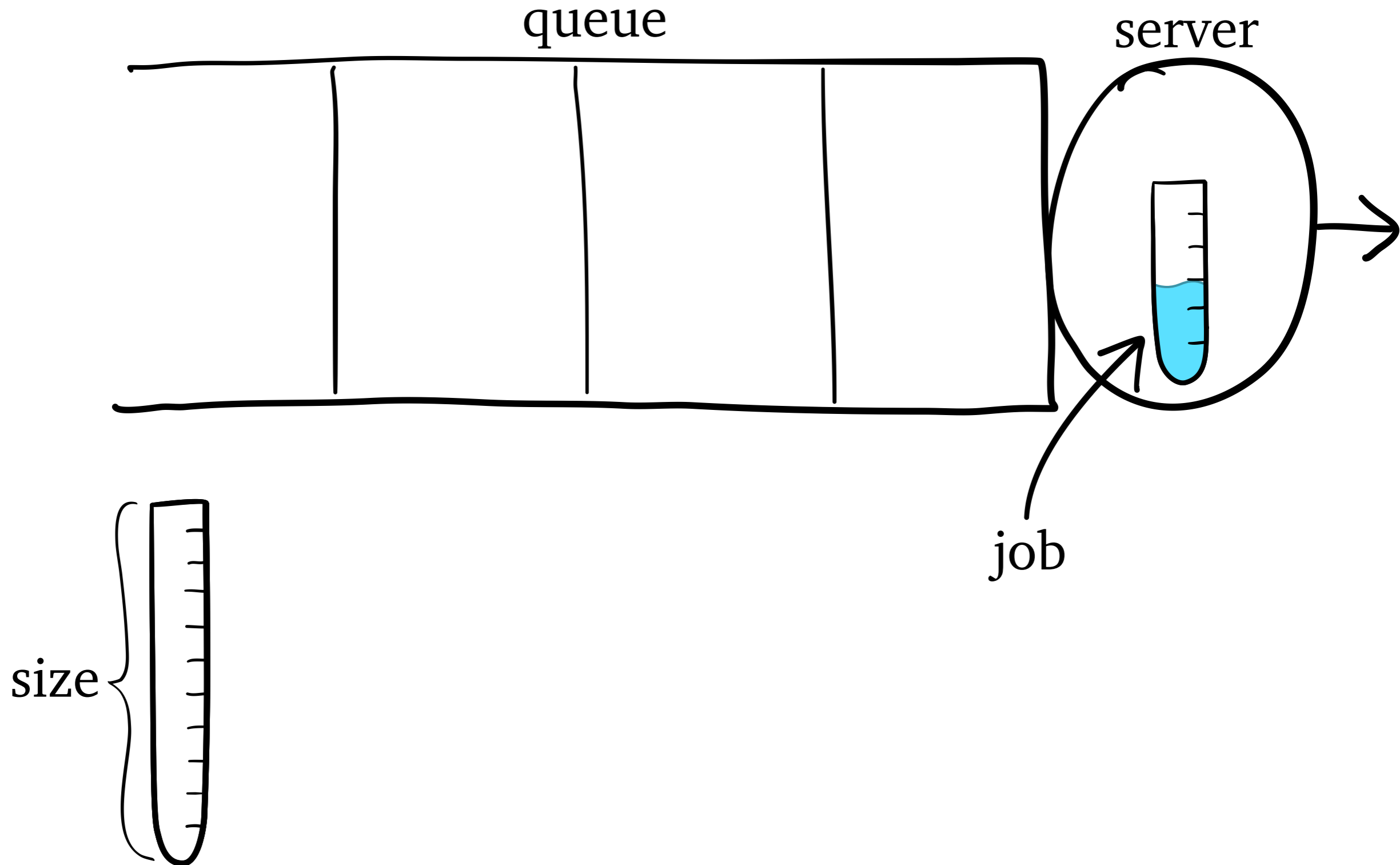
M/G/1 Queueing Model



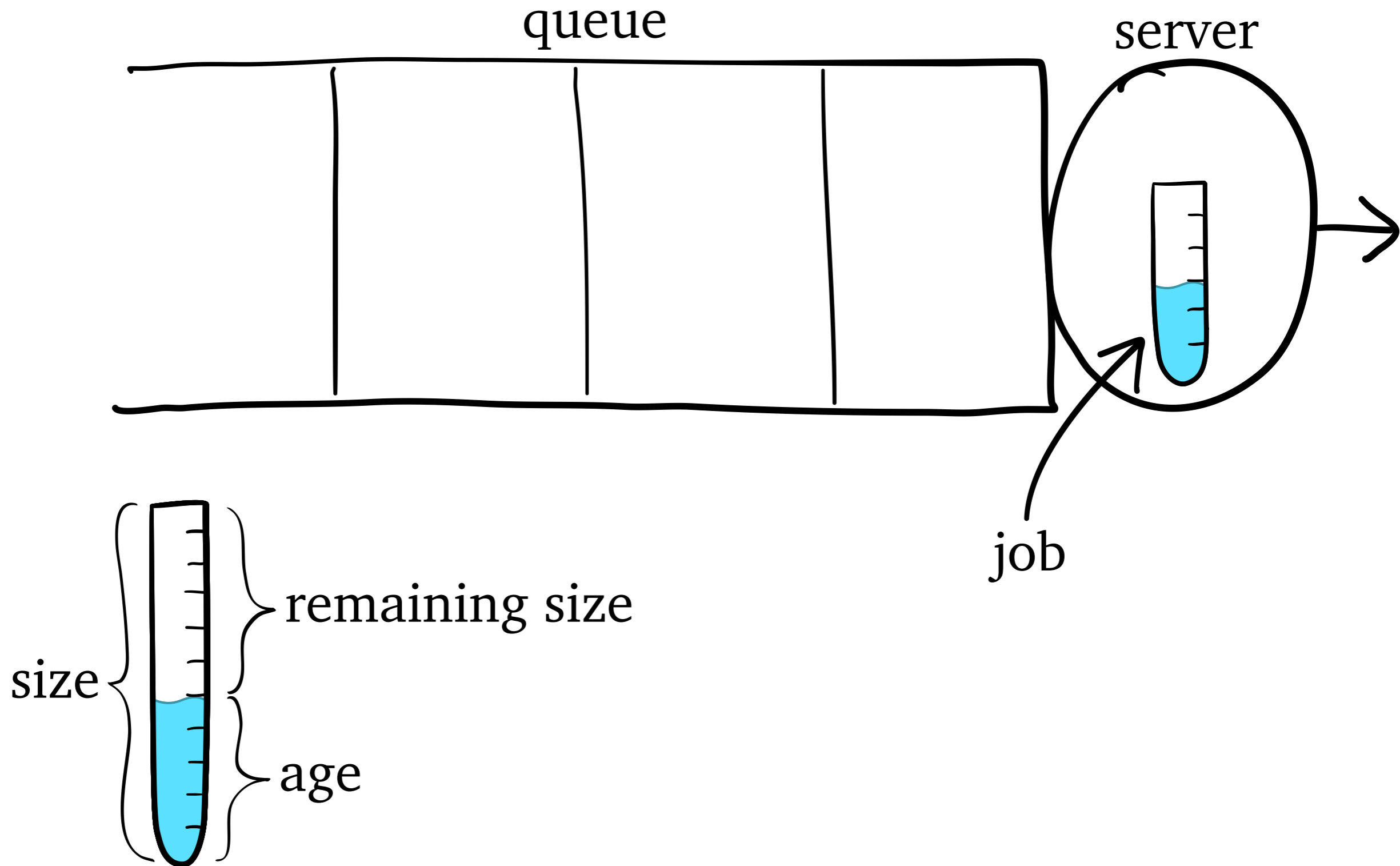
M/G/1 Queueing Model



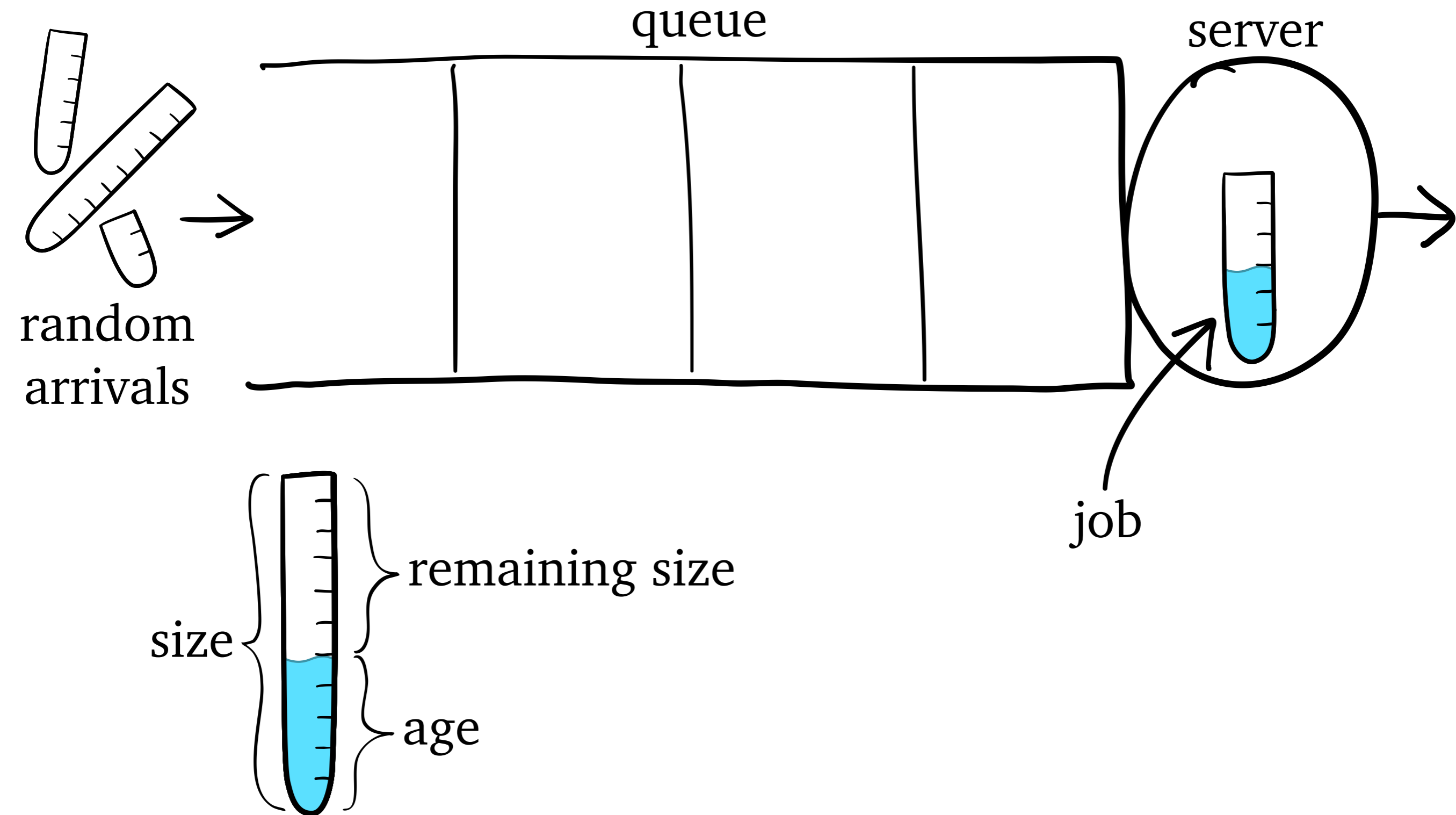
M/G/1 Queueing Model



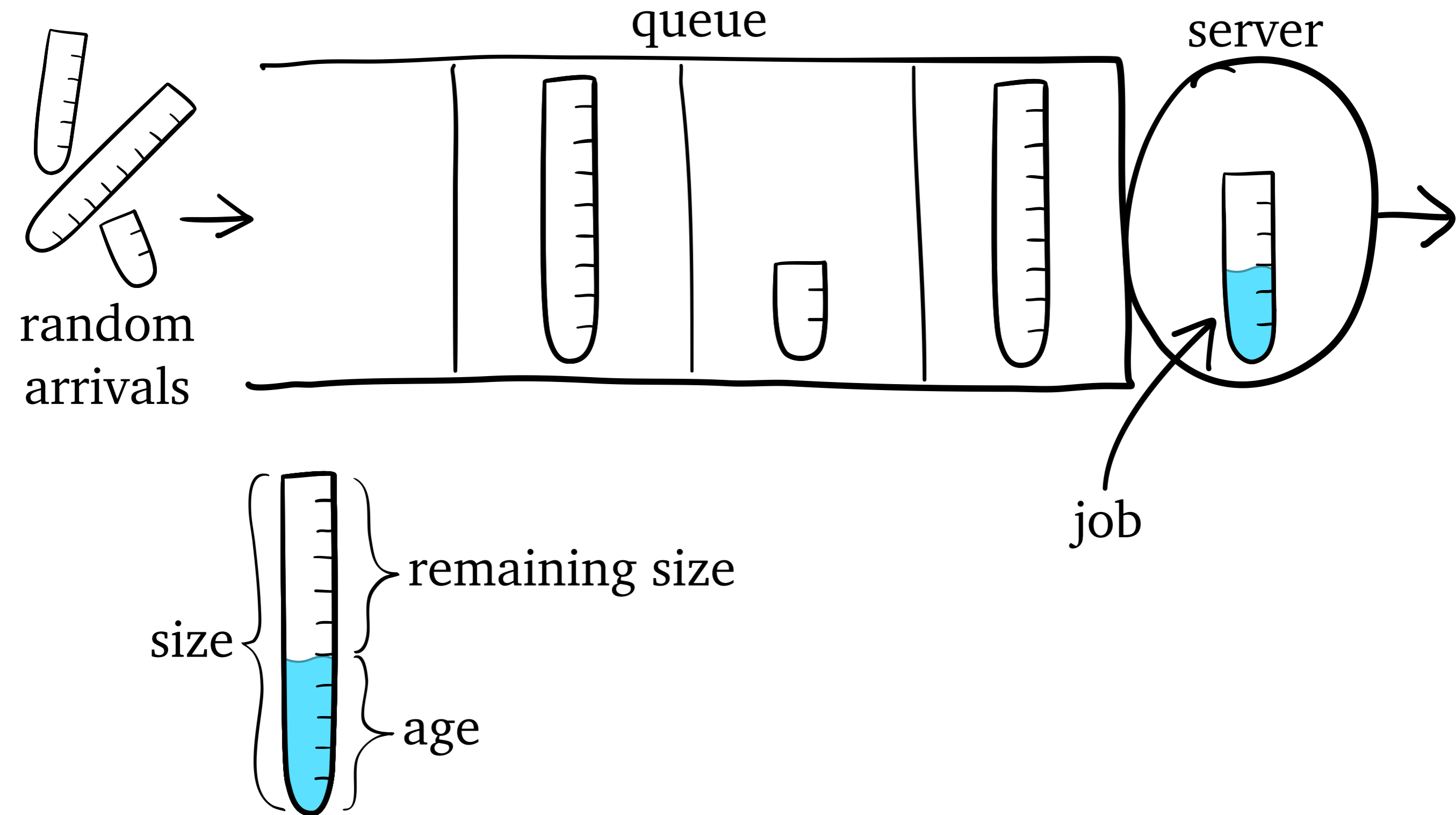
M/G/1 Queueing Model



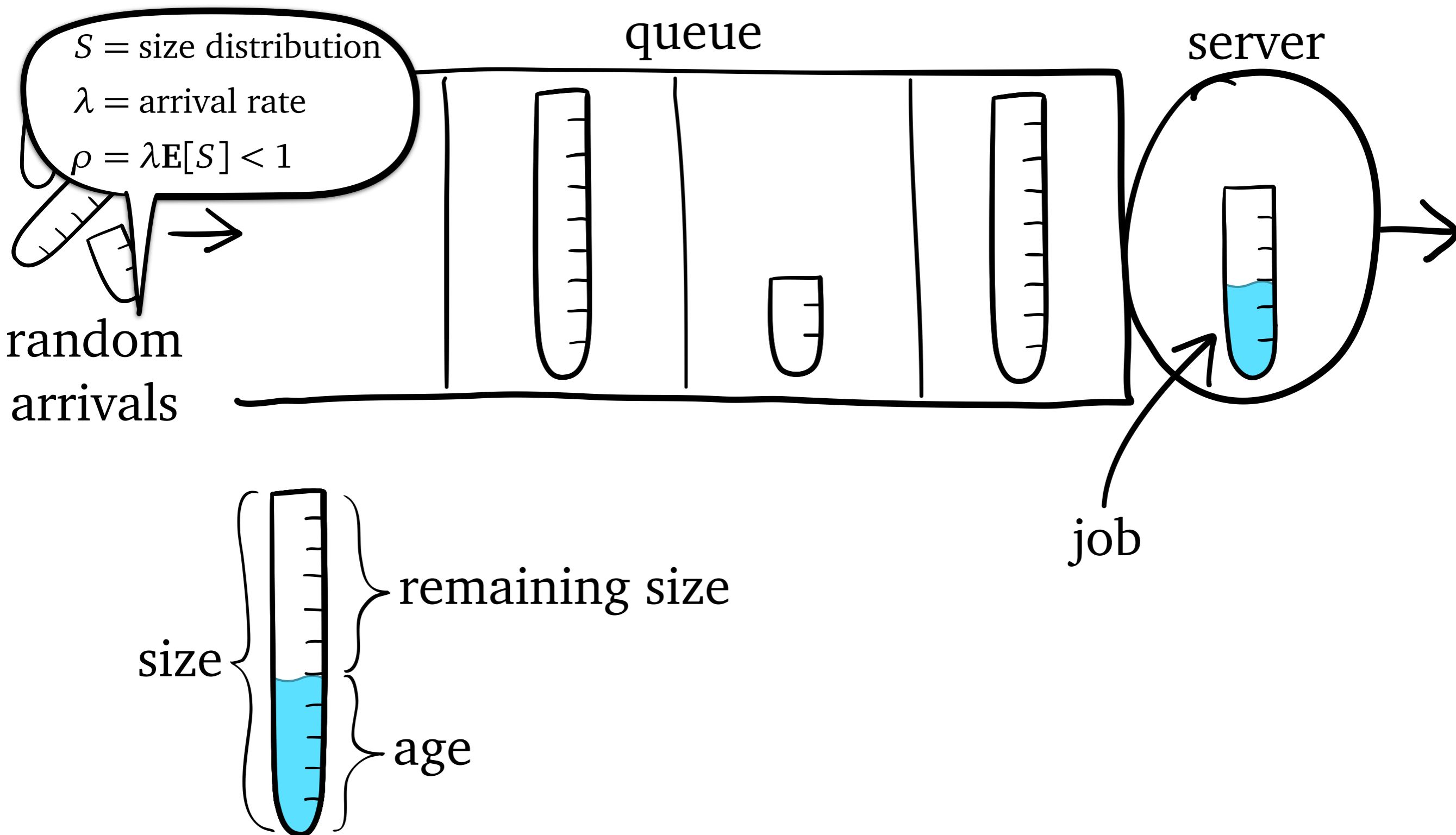
M/G/1 Queueing Model



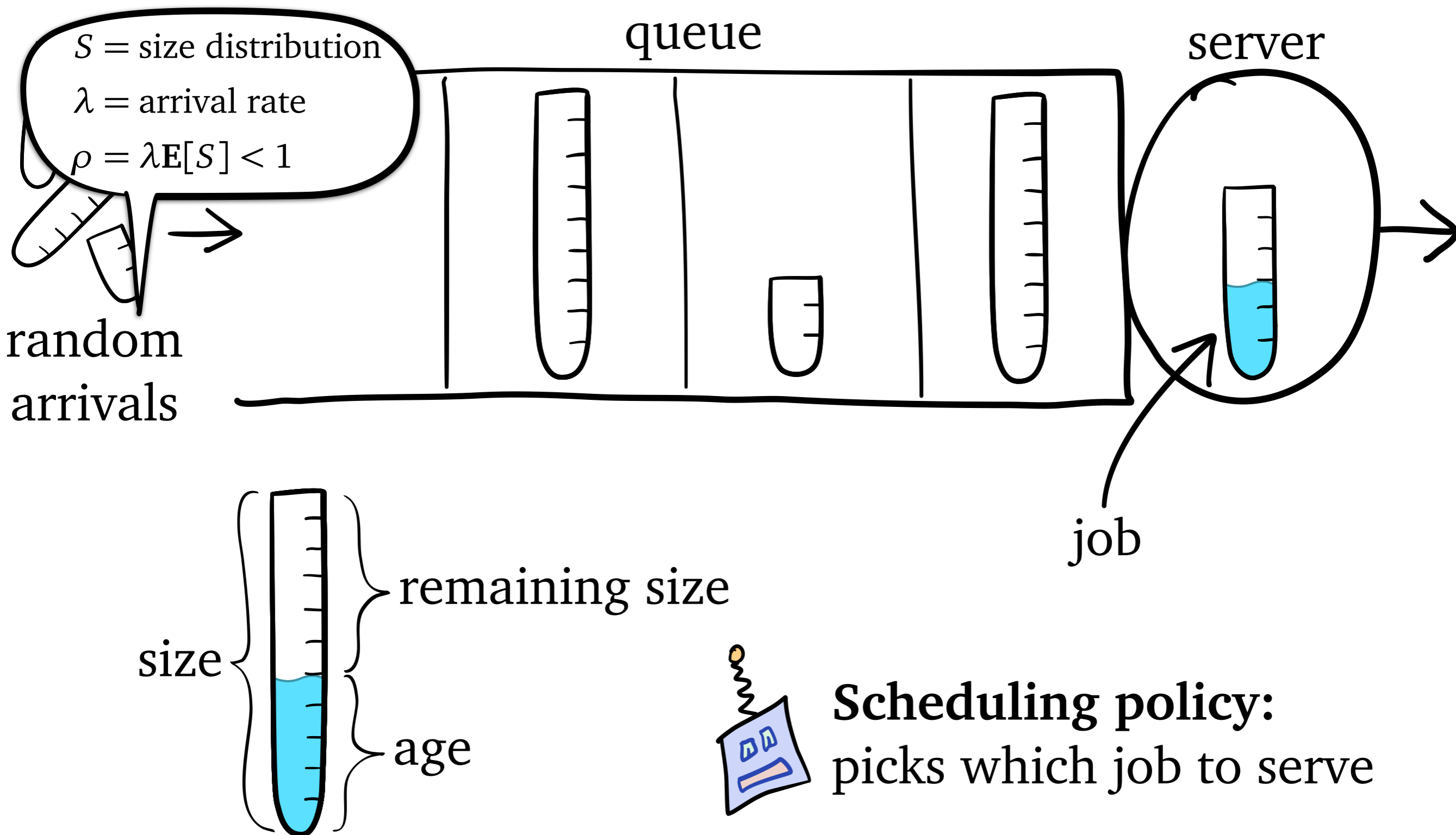
M/G/1 Queueing Model



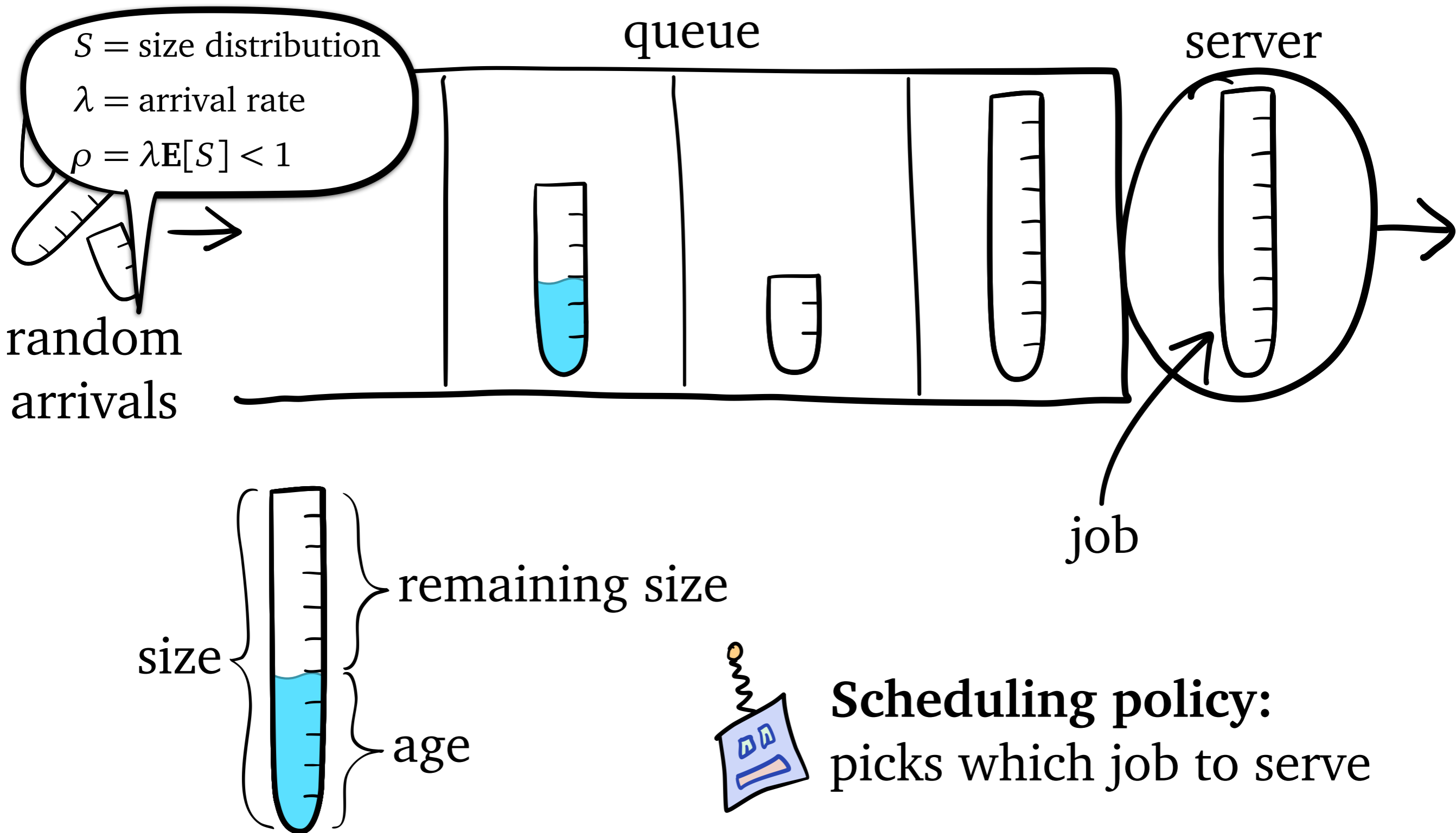
M/G/1 Queueing Model



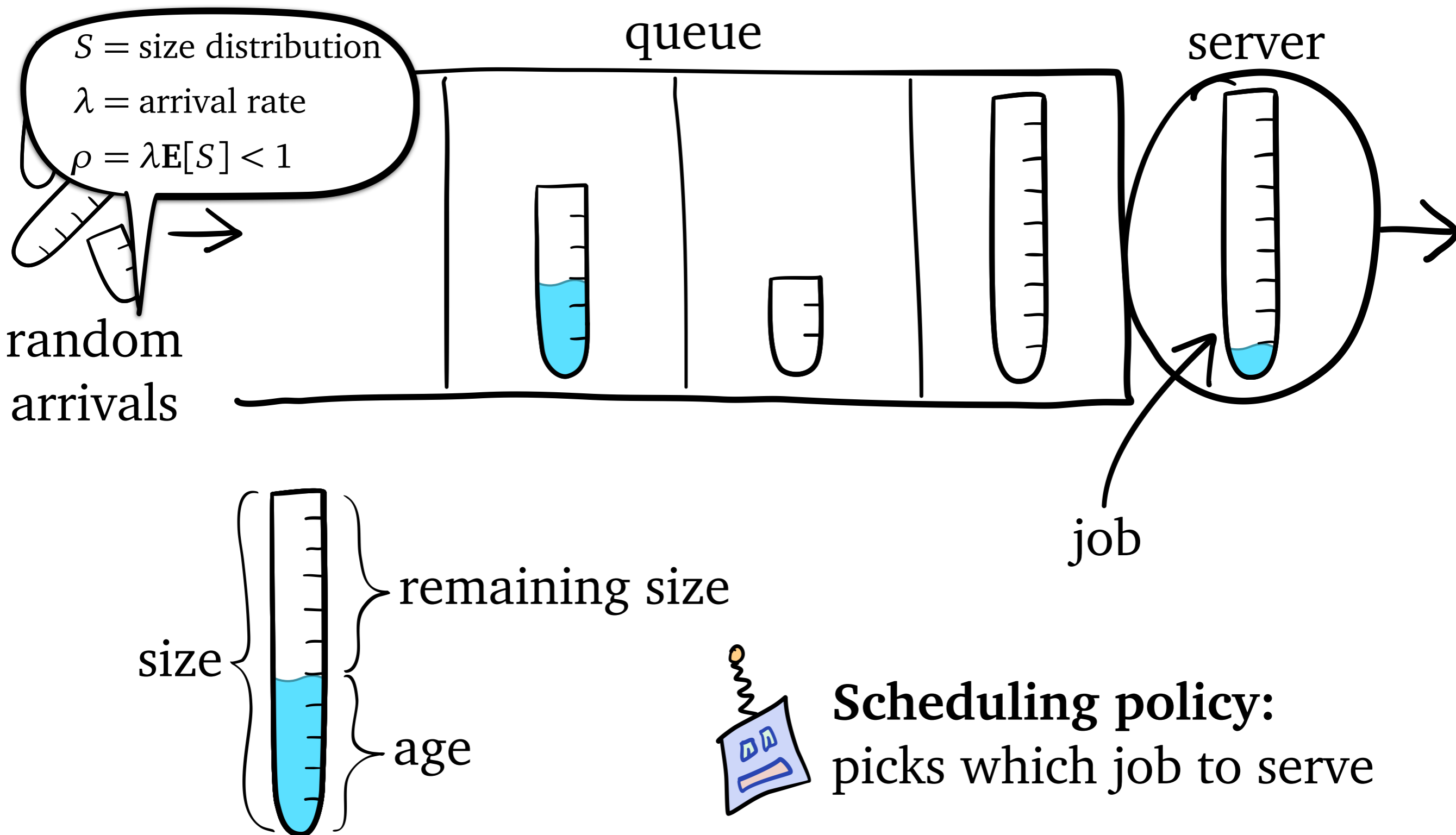
M/G/1 Queueing Model



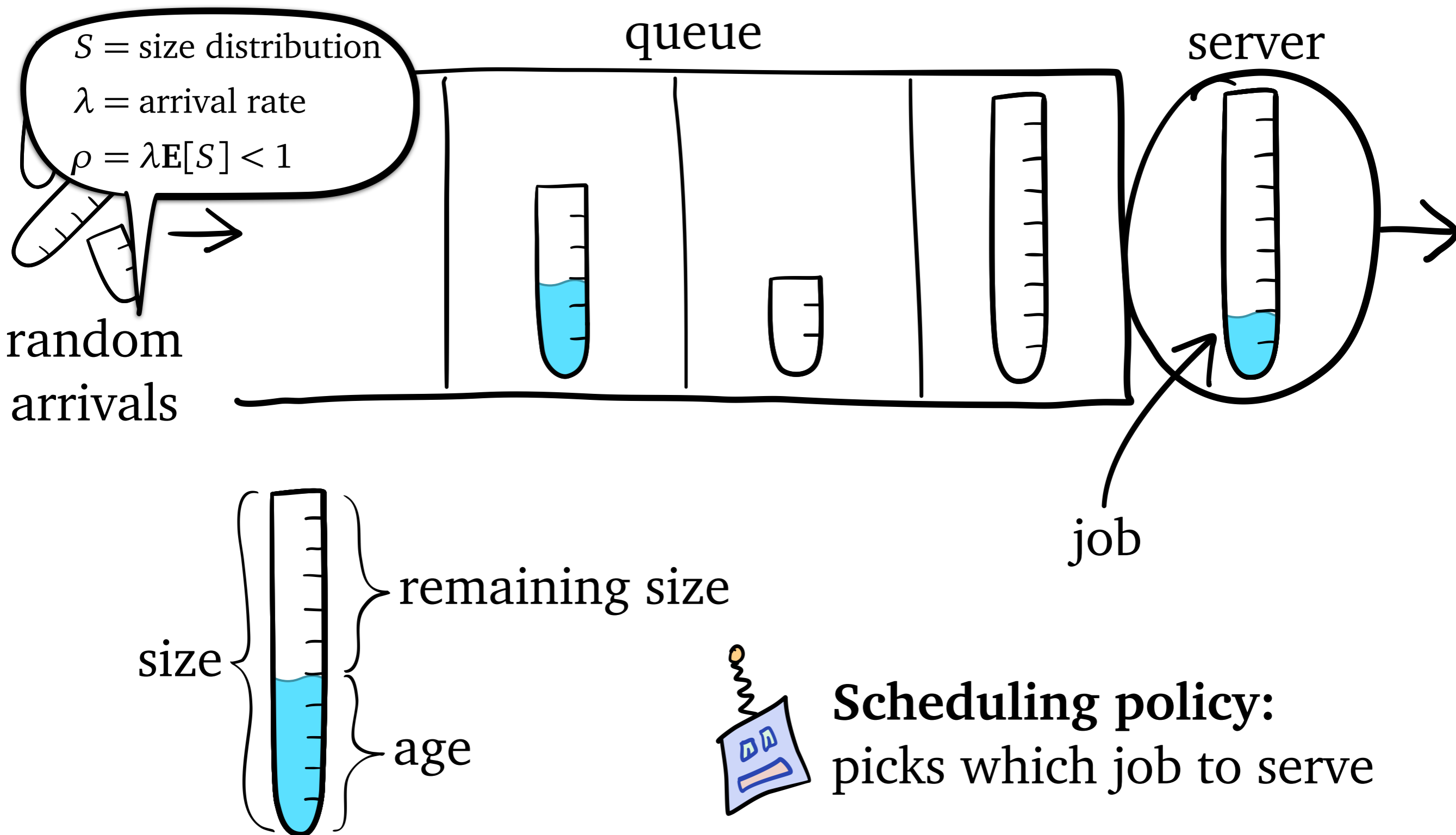
M/G/1 Queueing Model



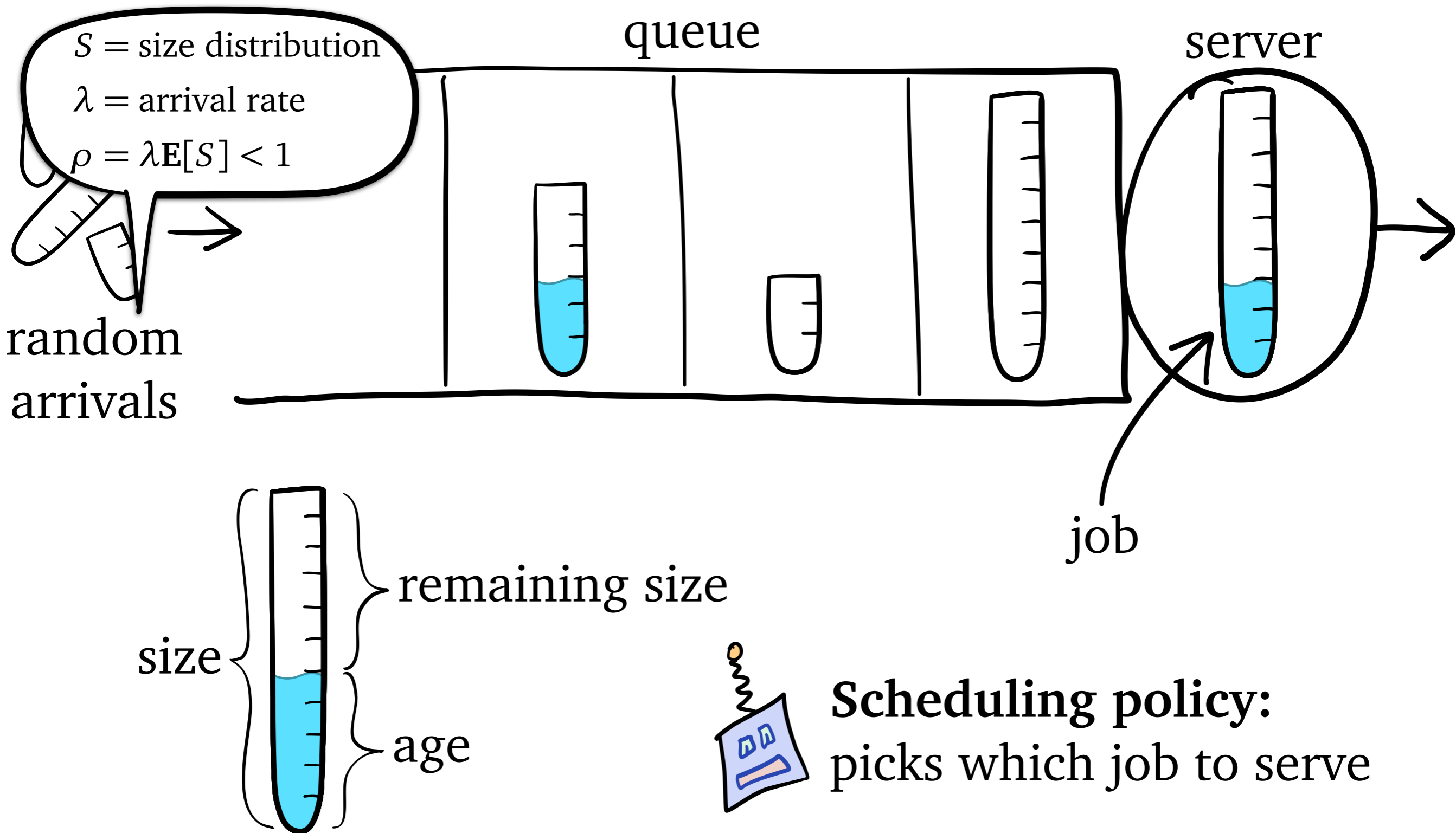
M/G/1 Queueing Model



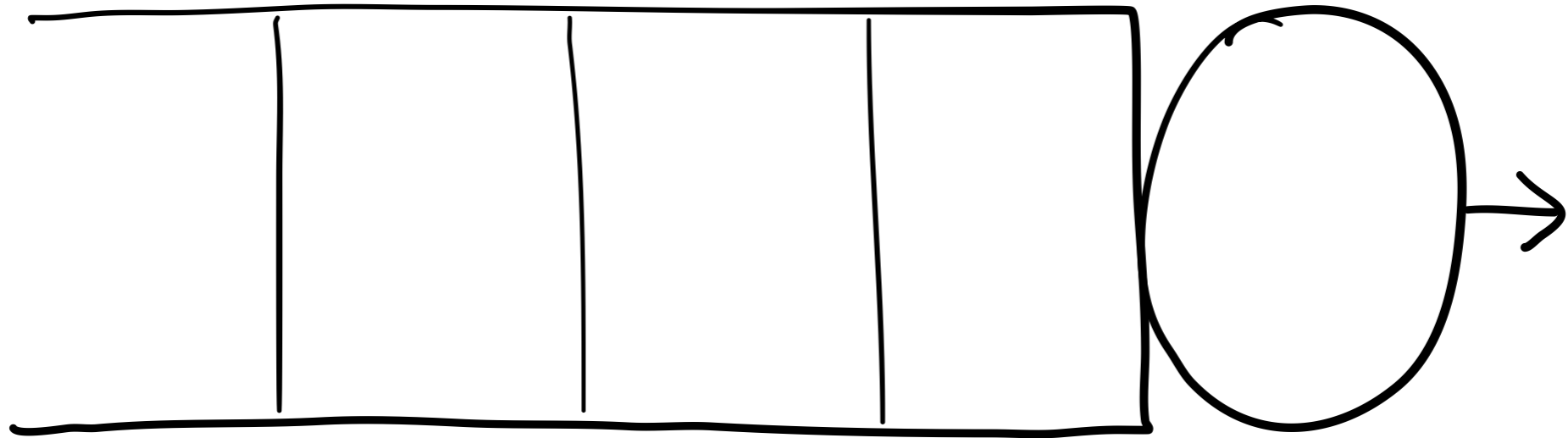
M/G/1 Queueing Model



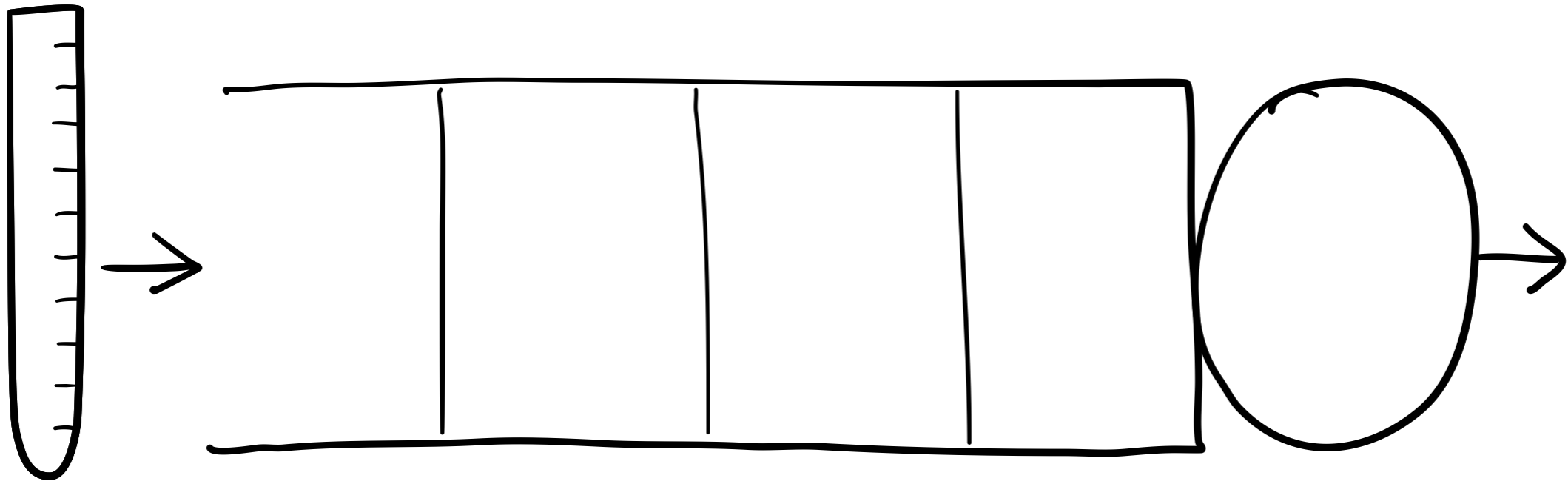
M/G/1 Queueing Model



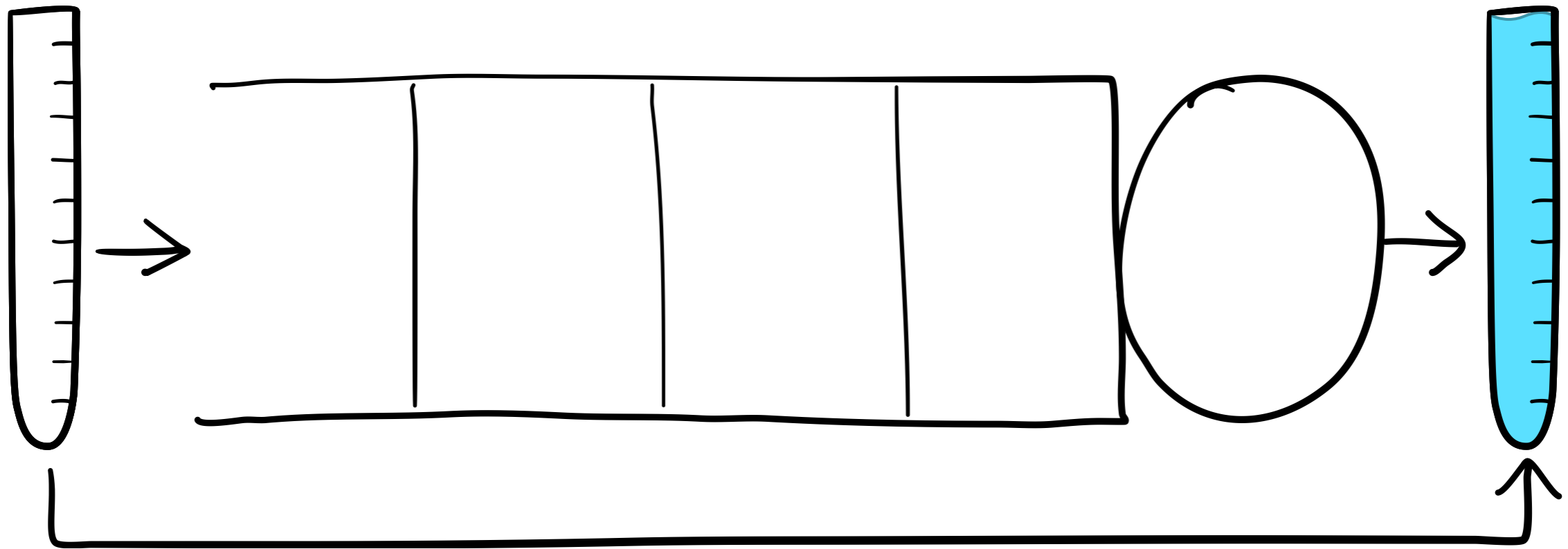
Response Time




Response Time

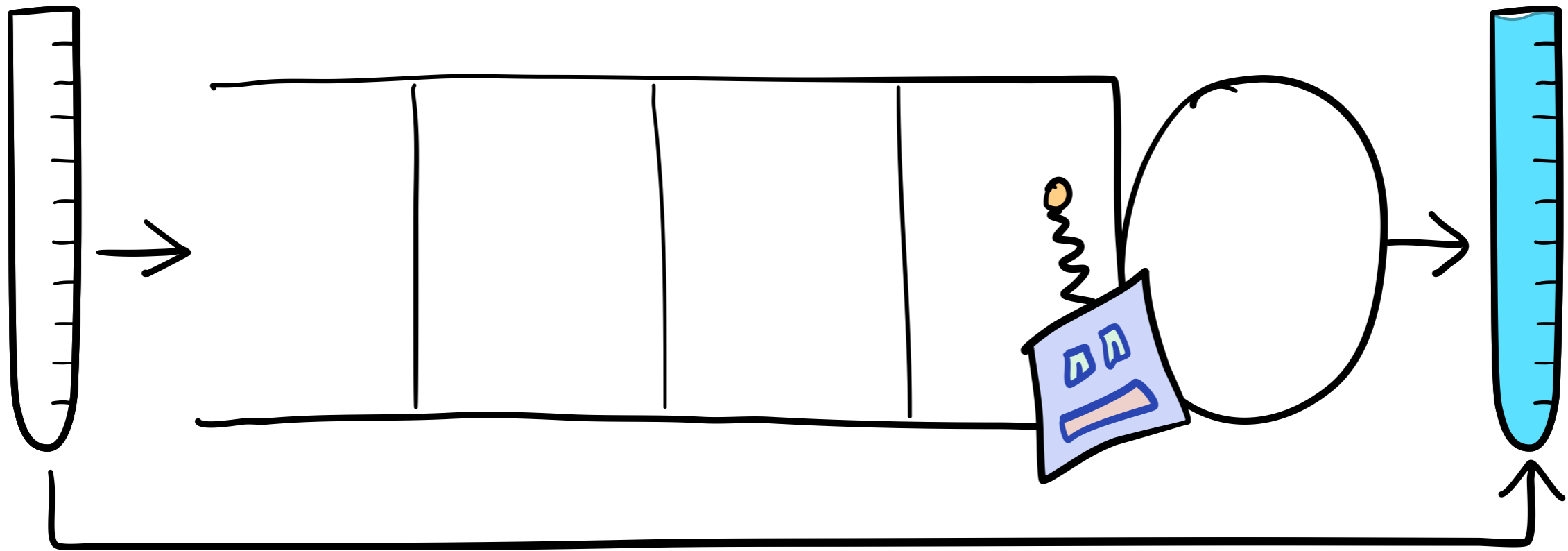



Response Time



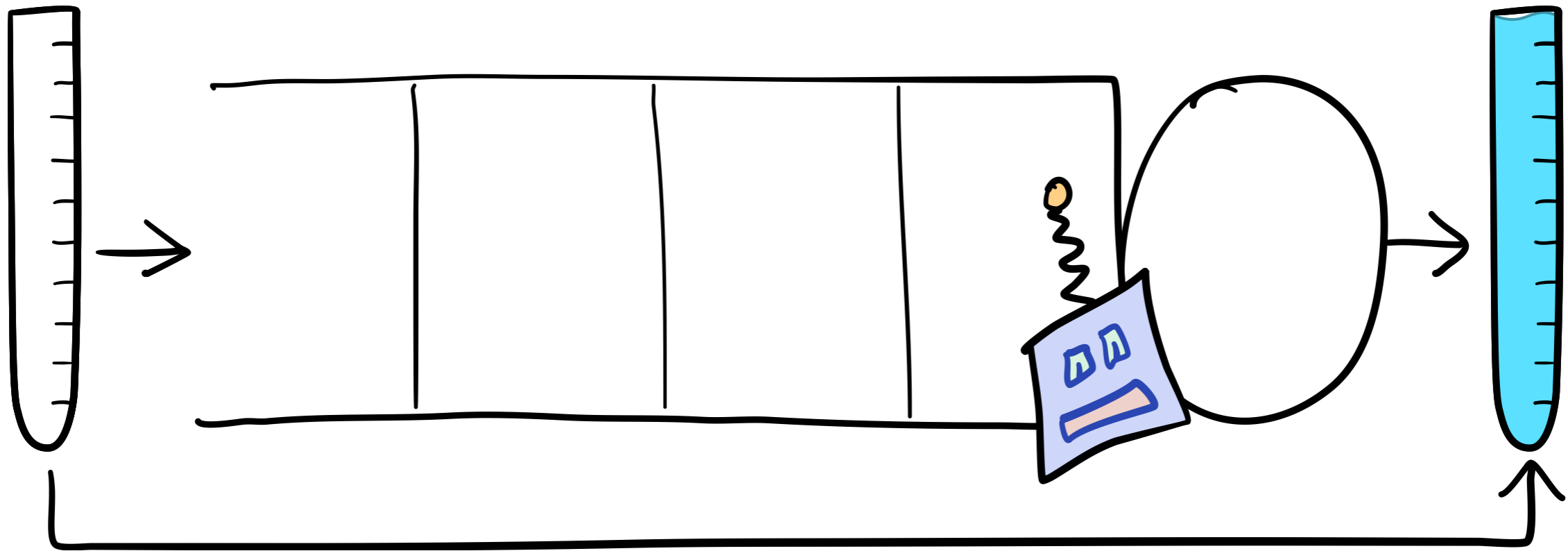
 = T = *response time*


Response Time



 = T = *response time*

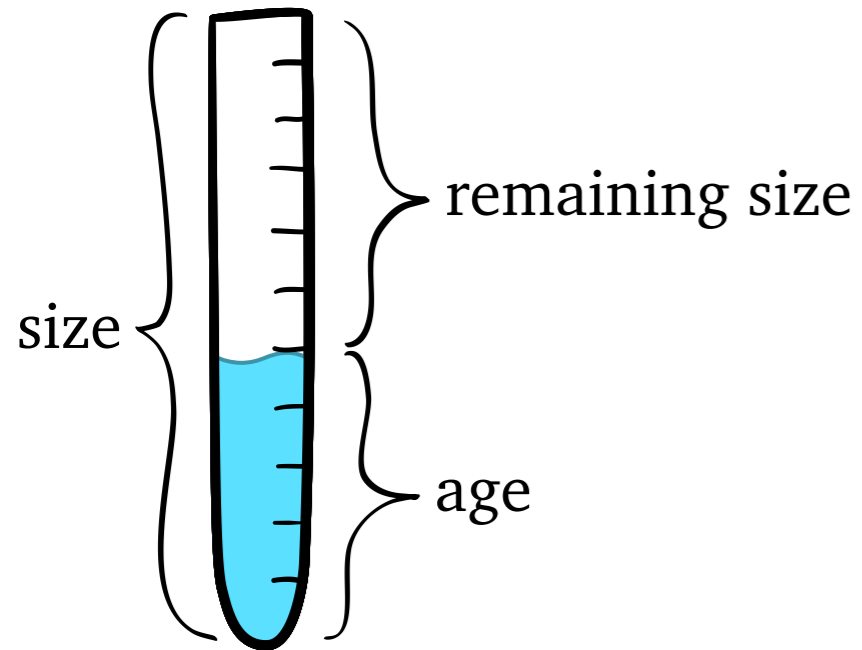
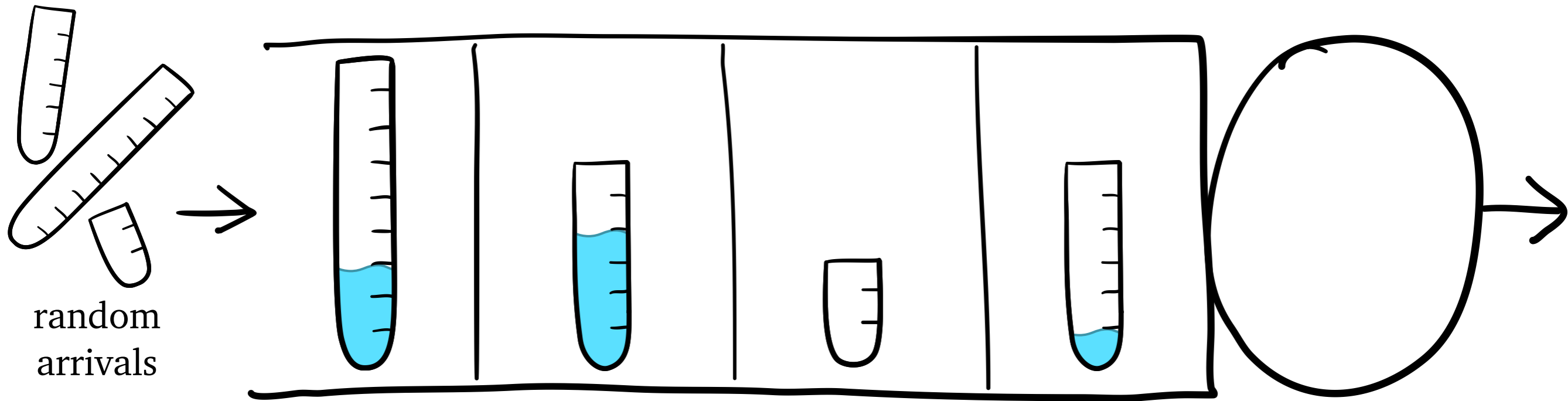
Response Time



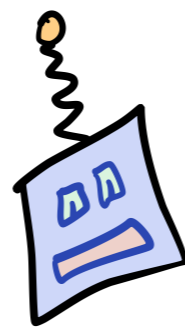
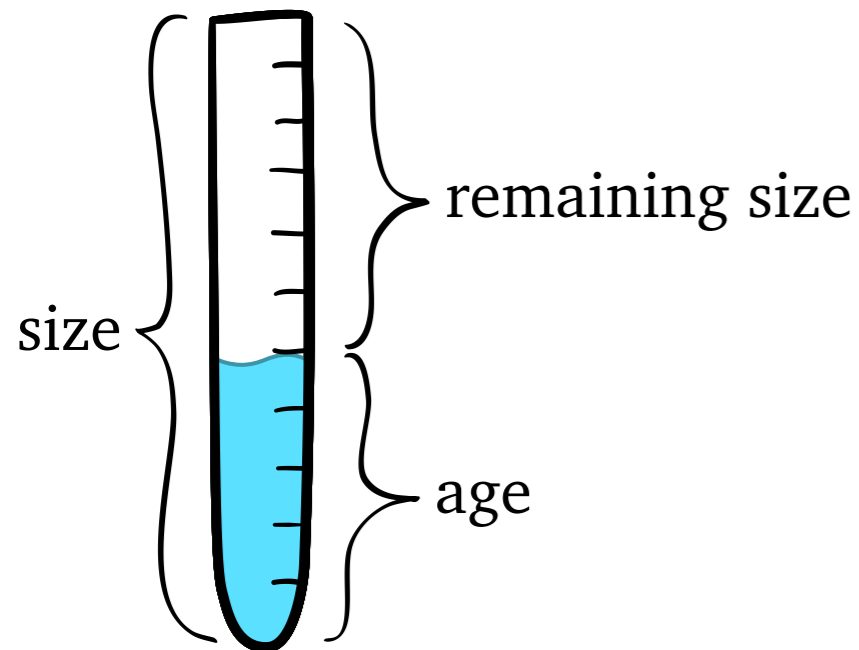
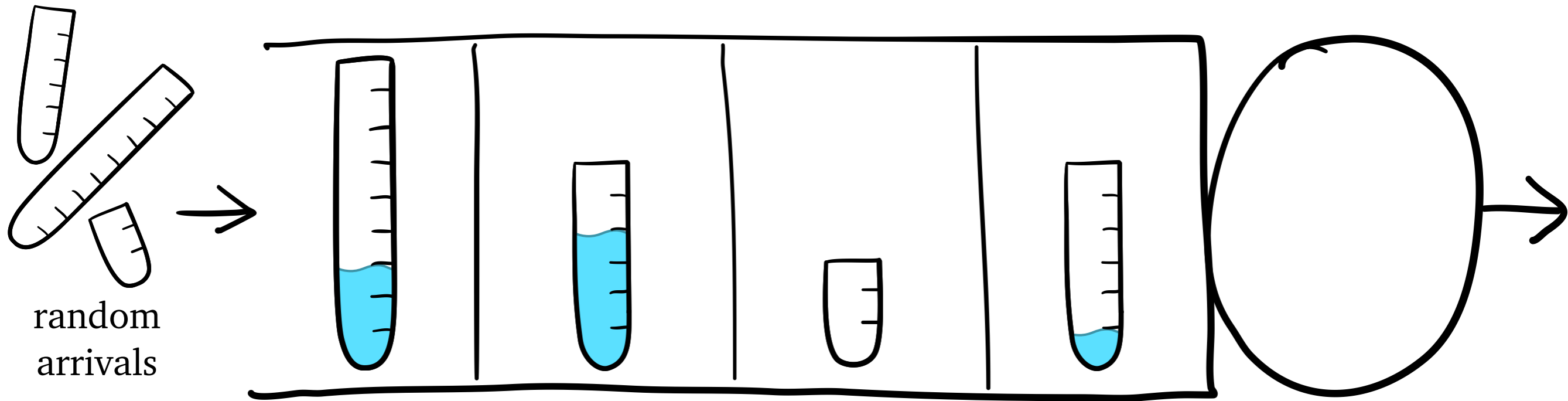
 = T = *response time*

Goal: schedule to minimize *mean response time* $E[T]$ and other metrics

How to Schedule?

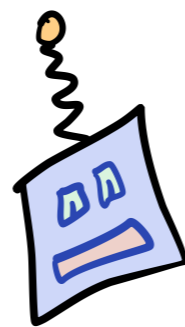
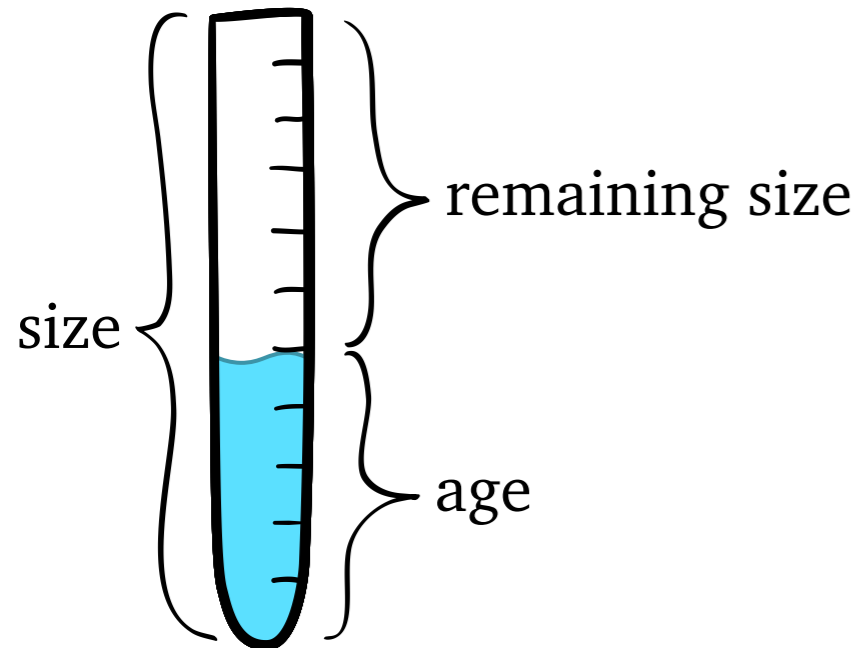
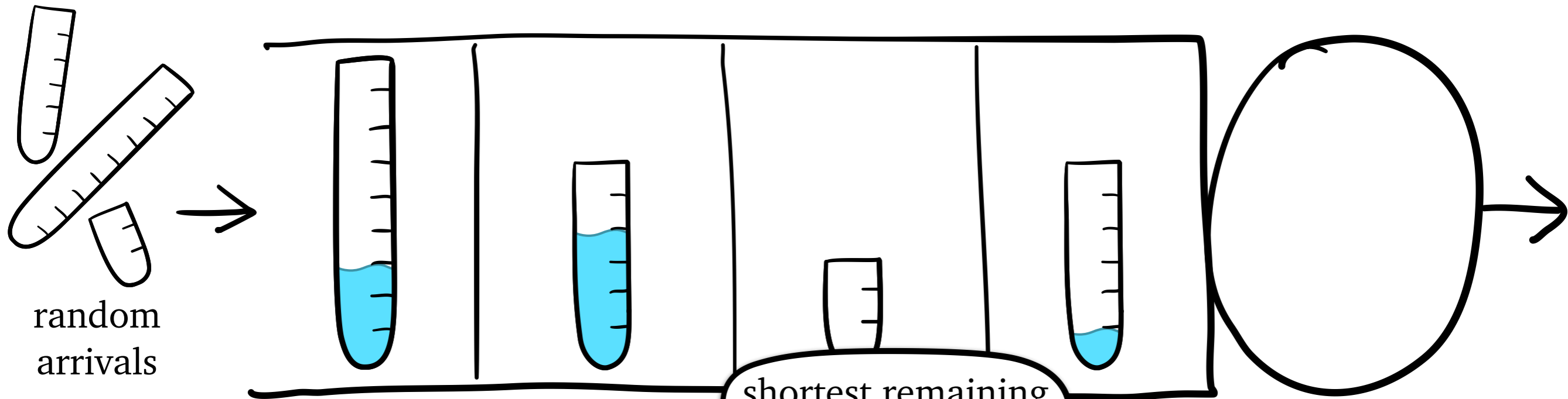


How to Schedule?



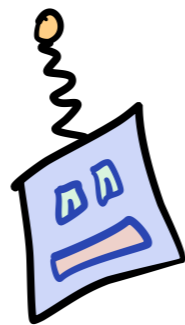
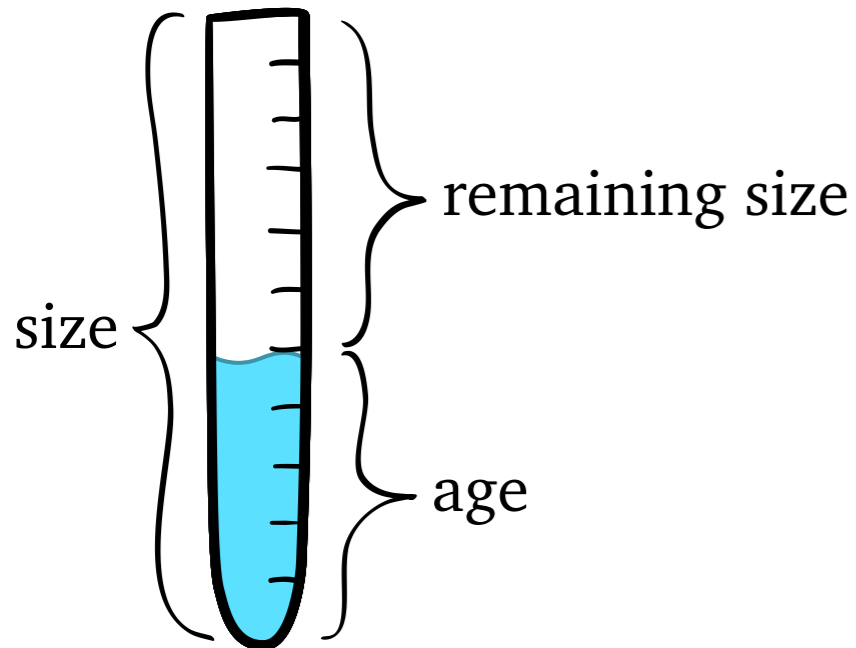
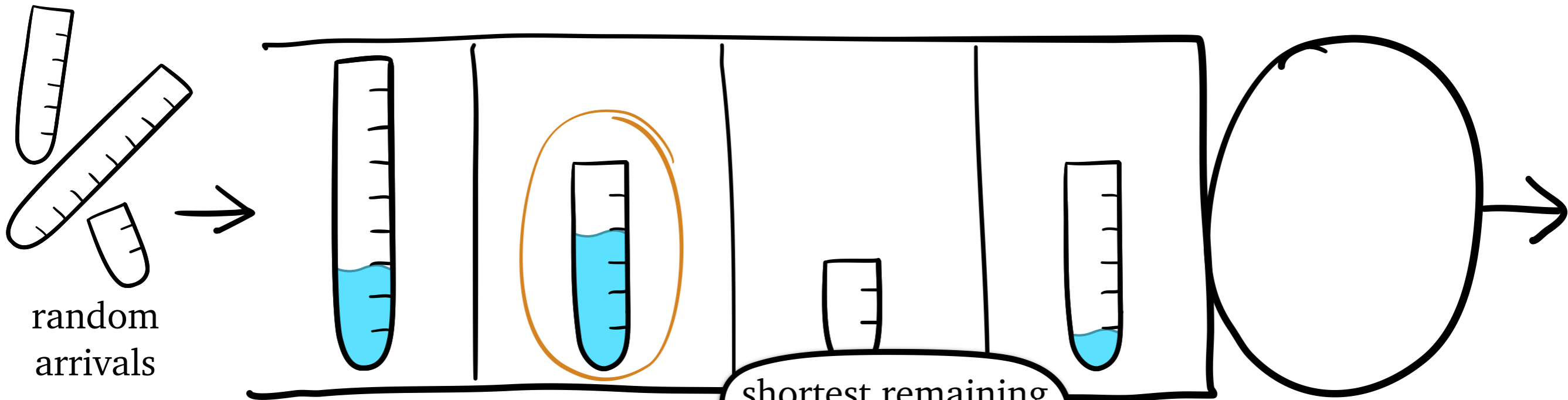
SRPT: always serve job of *least remaining size*

How to Schedule?



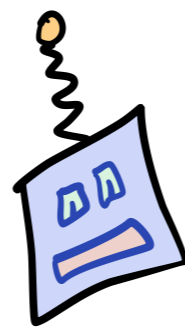
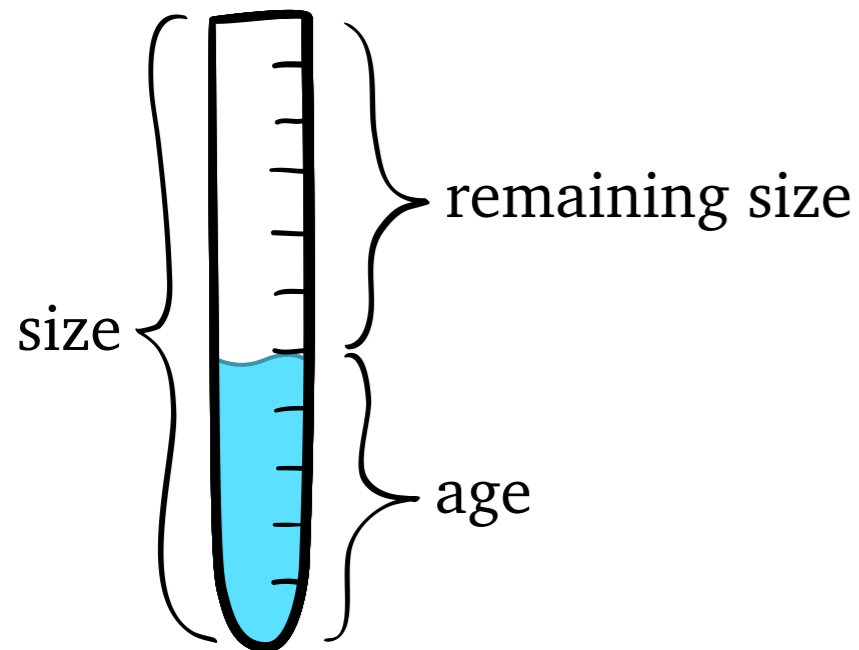
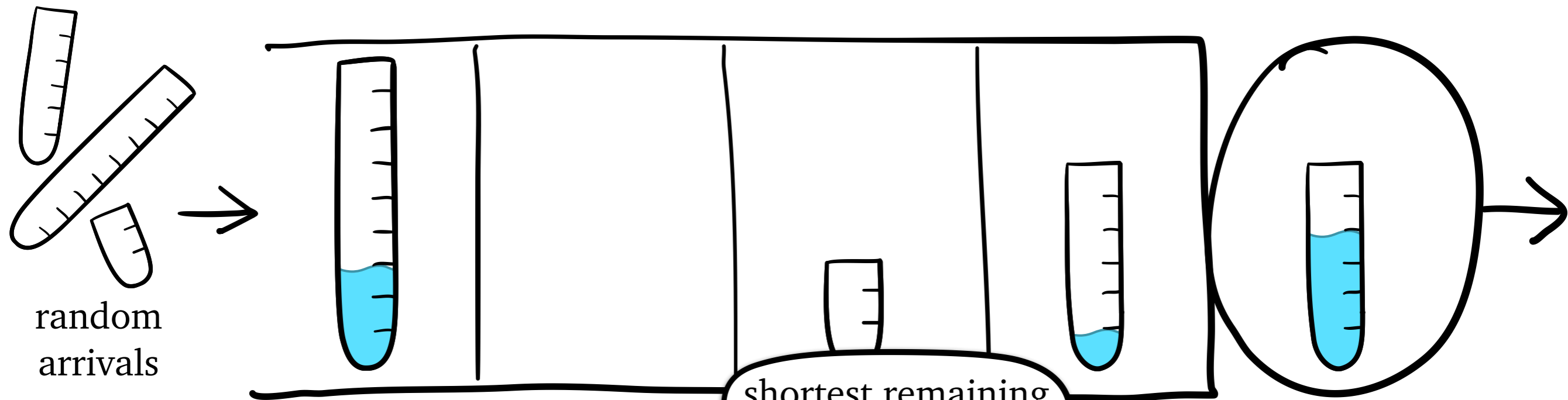
SRPT: always serve job of *least remaining size*

How to Schedule?



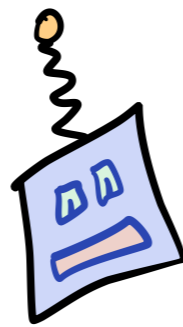
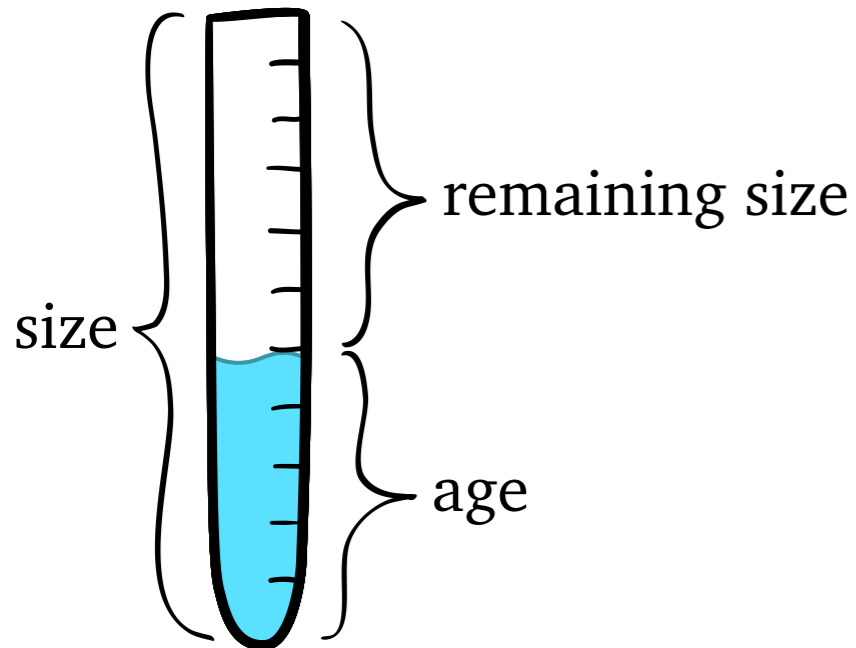
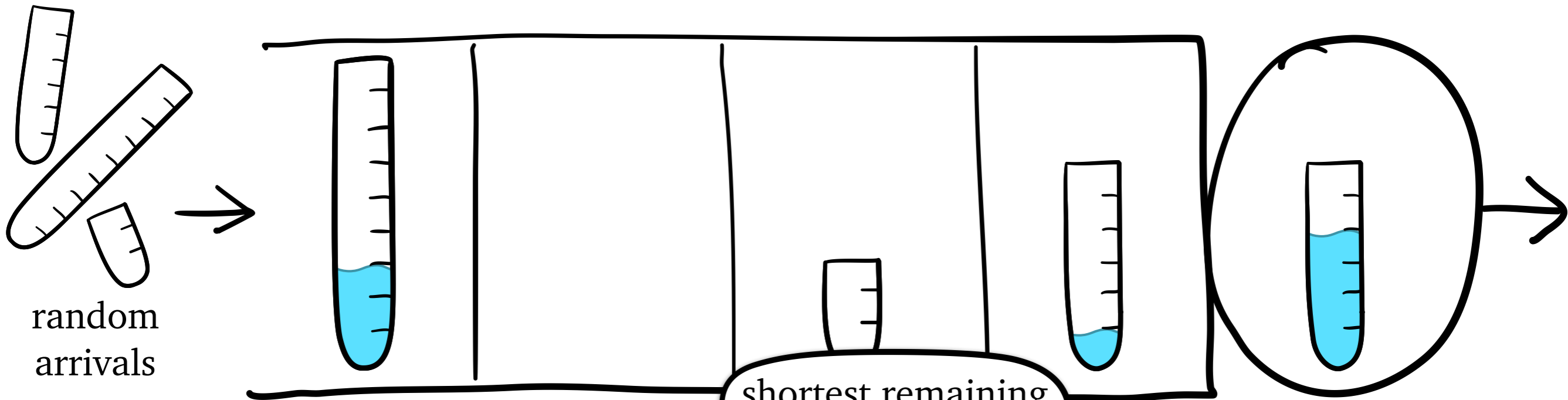
SRPT: always serve job of *least remaining size*

How to Schedule?



SRPT: always serve job of *least remaining size*

How to Schedule?

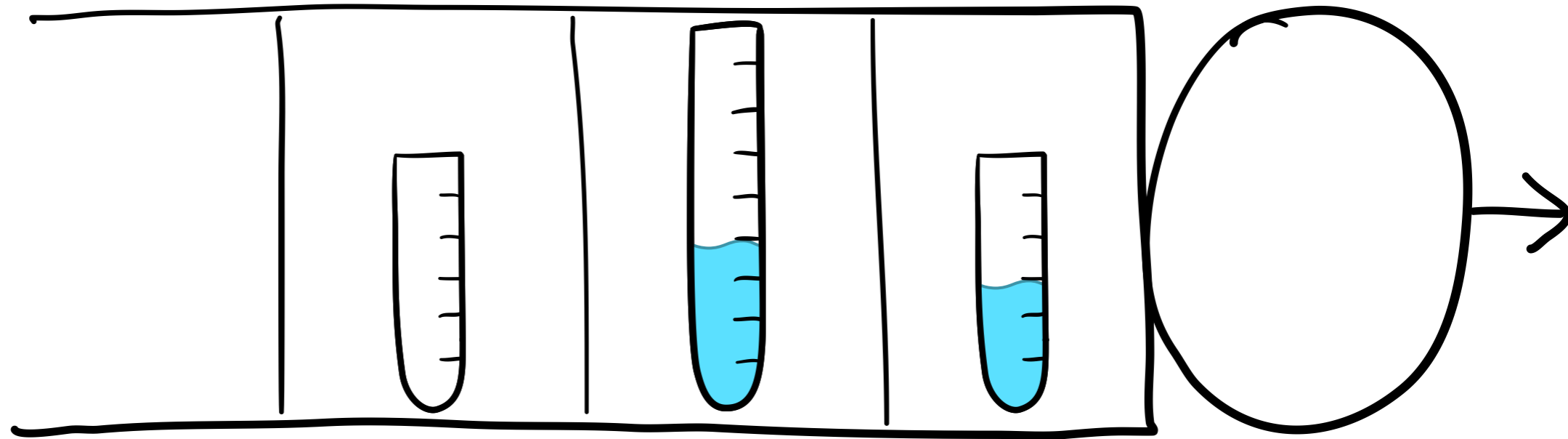


SRPT: always serve job of *least remaining size*

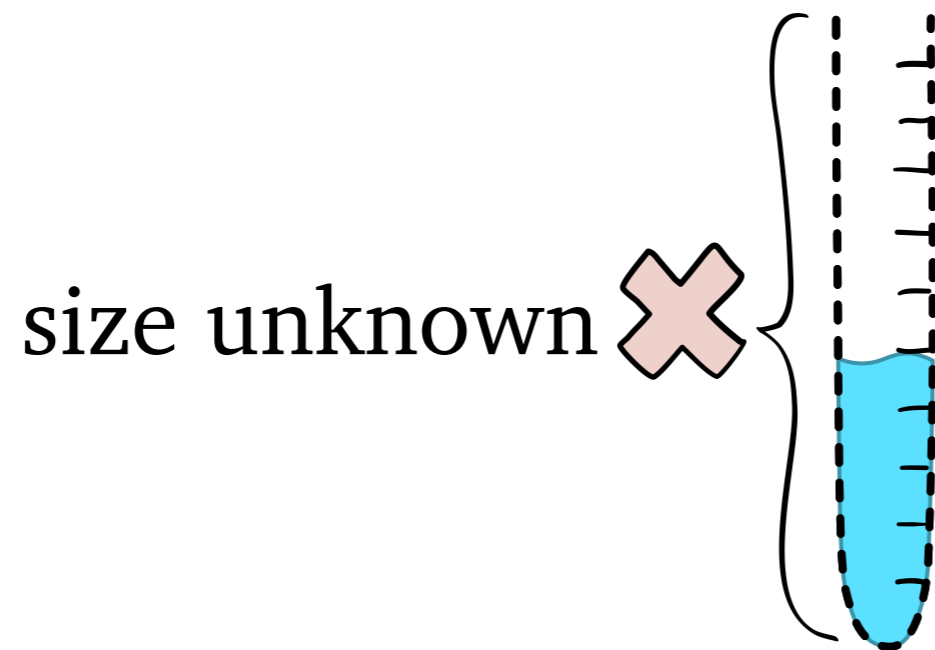
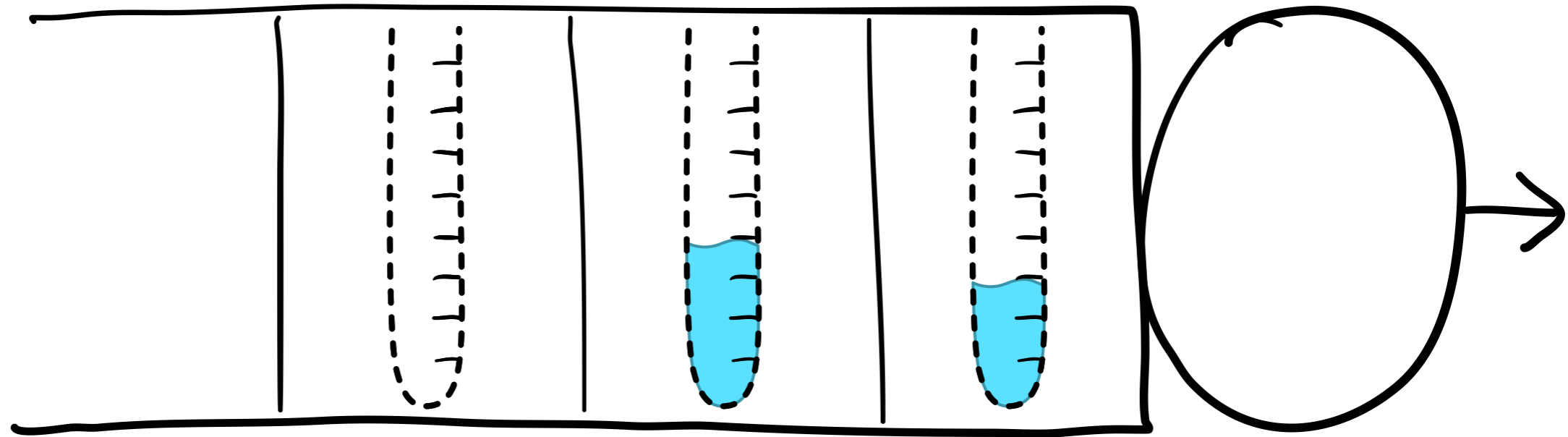


SRPT minimizes $E[T]$

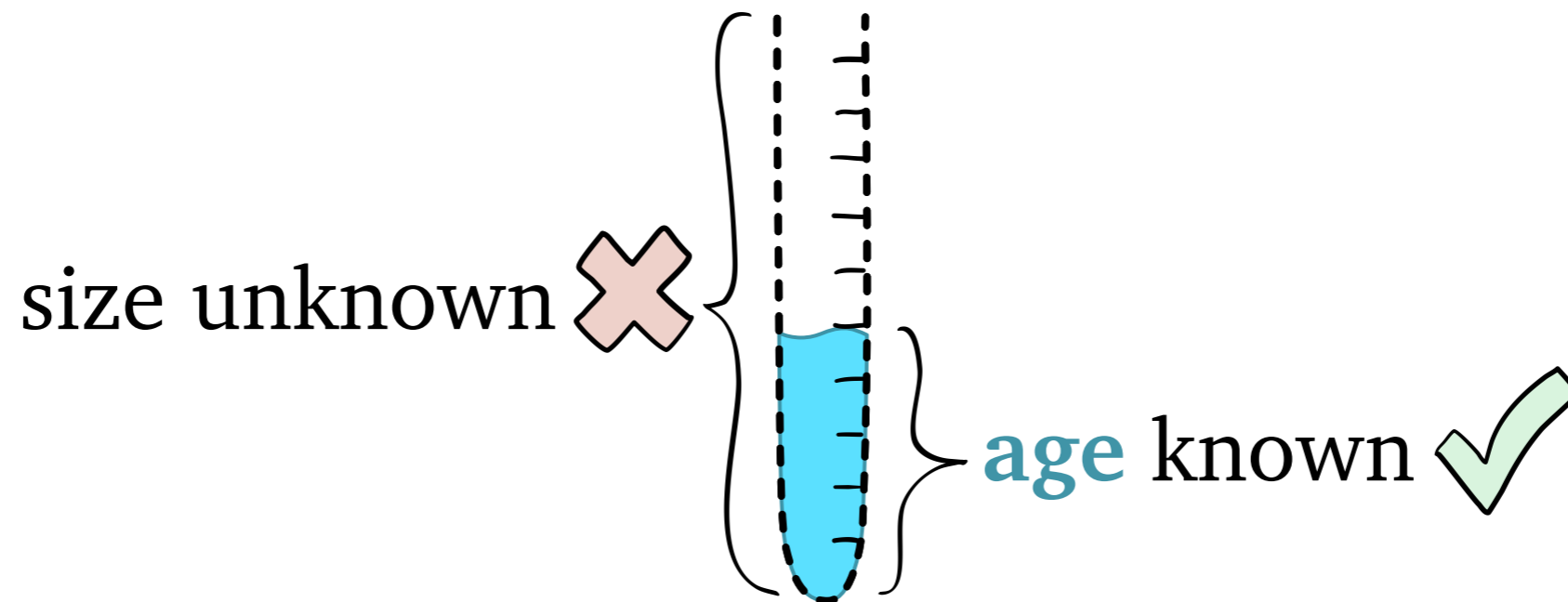
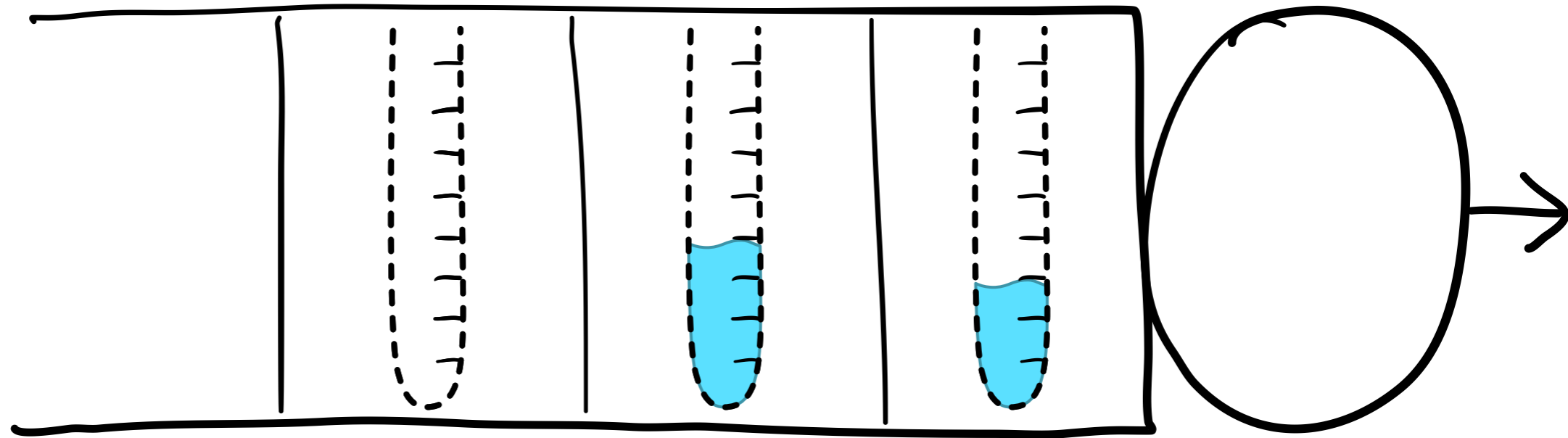
Unknown Job Sizes



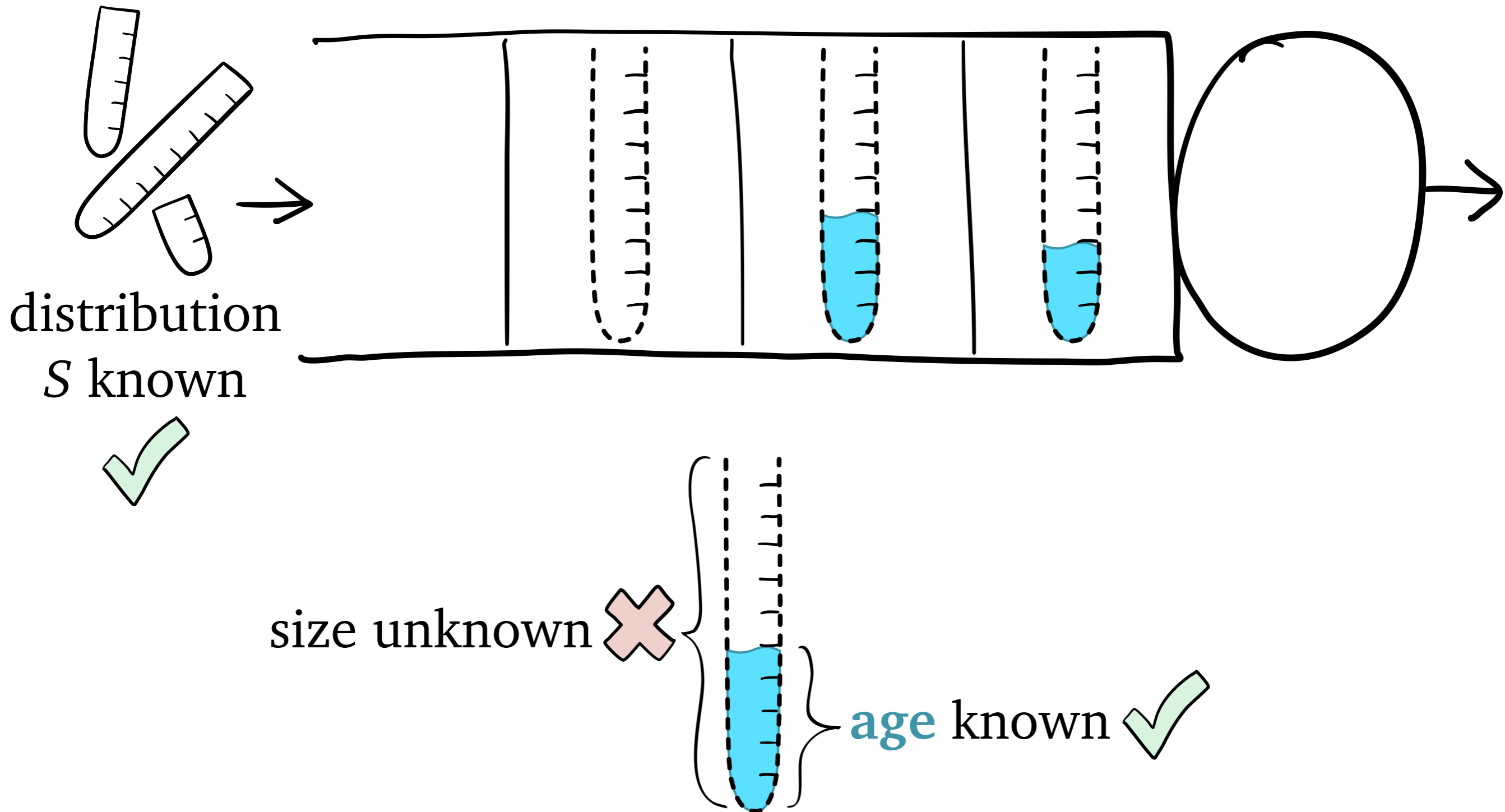
Unknown Job Sizes



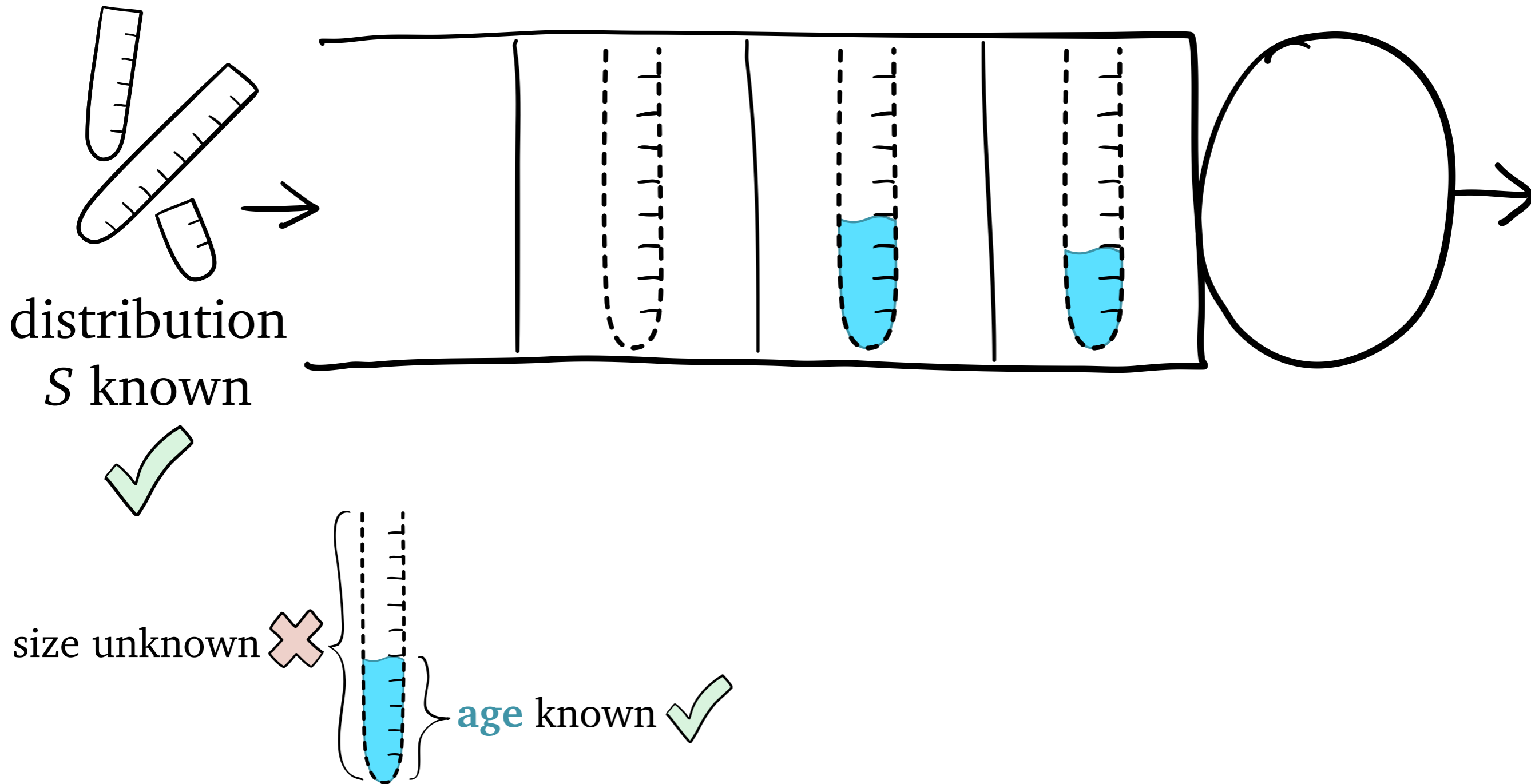
Unknown Job Sizes



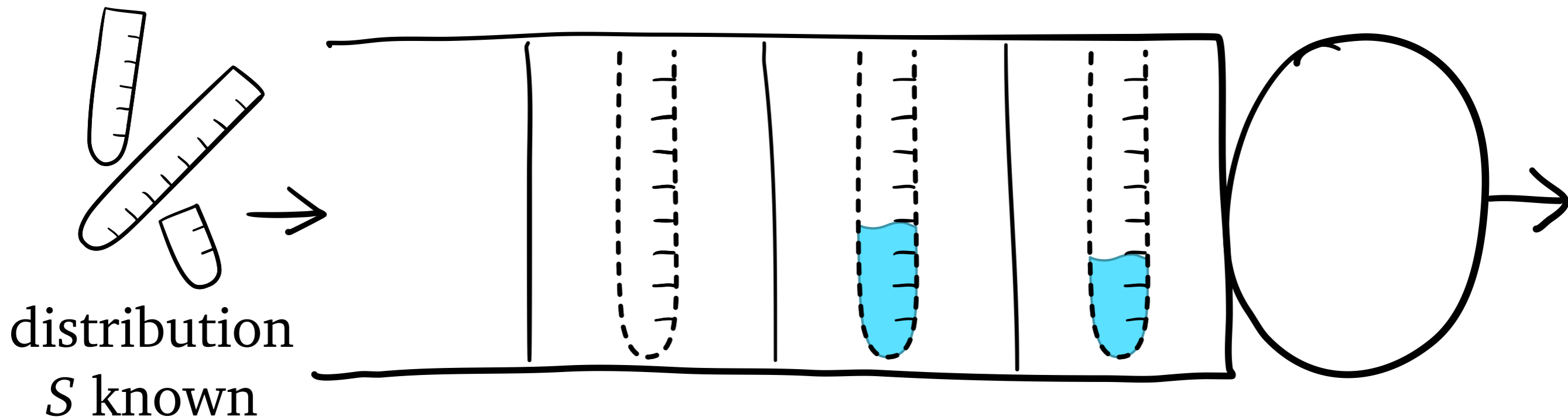
Unknown Job Sizes



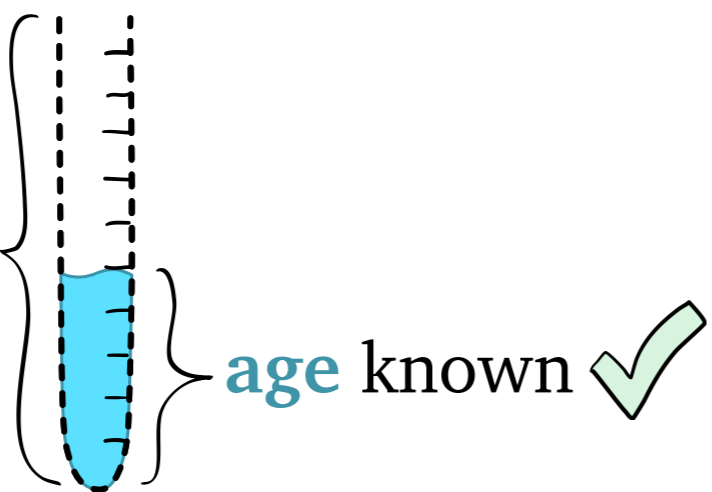
Unknown Job Sizes



Unknown Job Sizes

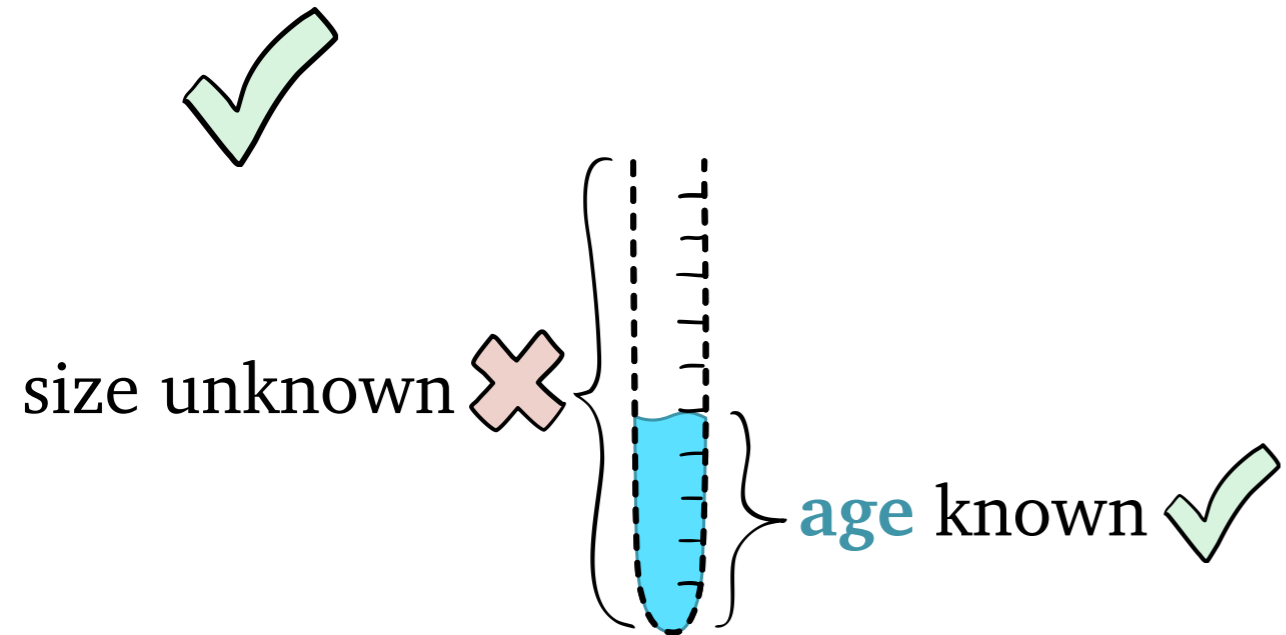
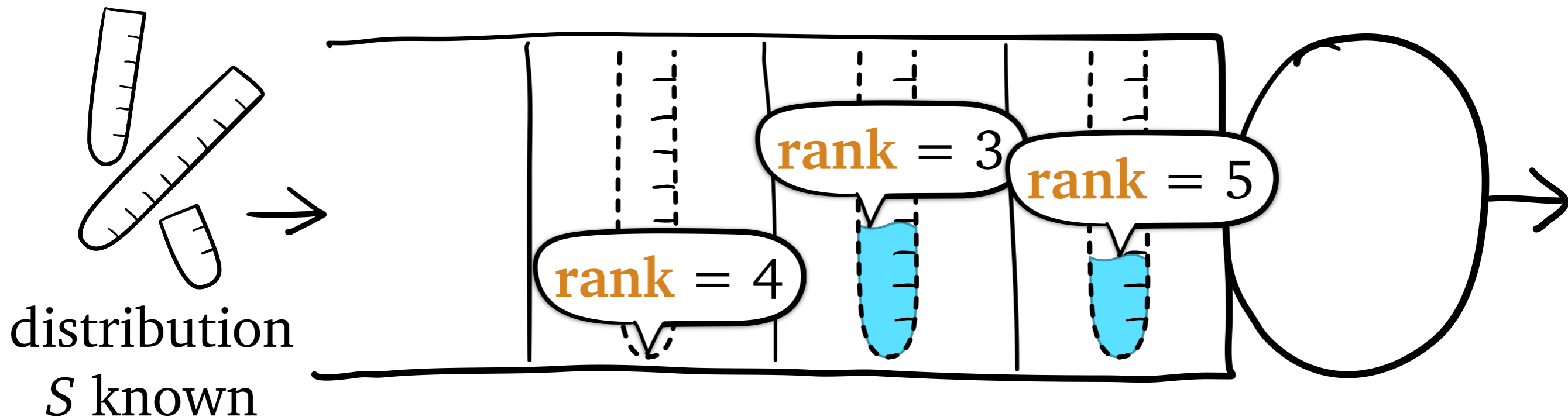



size unknown ✗



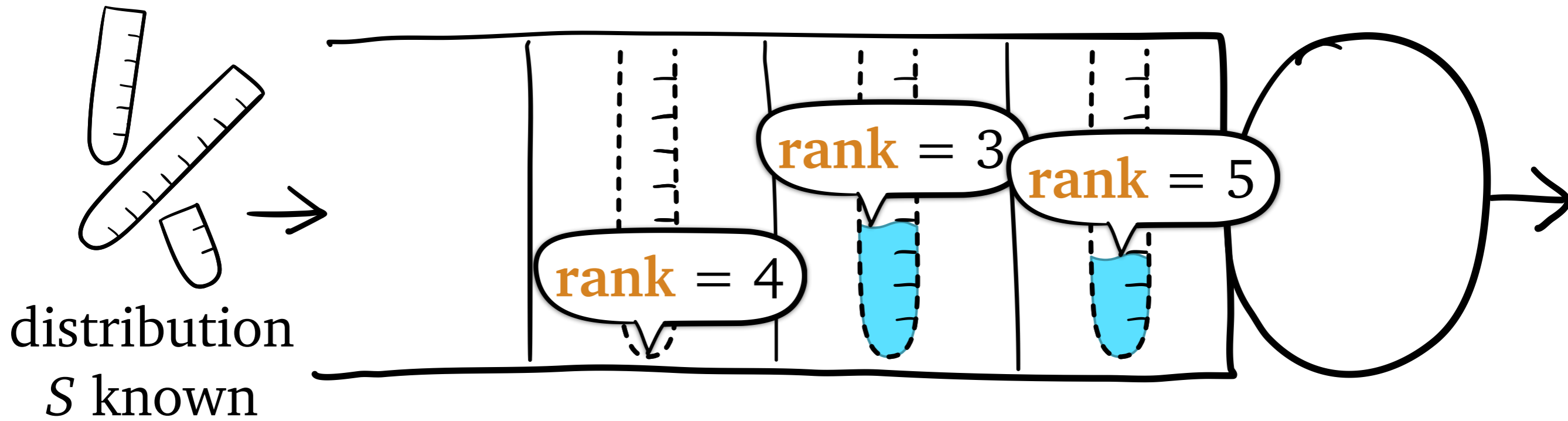
Gittins: assign each job a **rank** based on **age** and S
(lower is better)

Unknown Job Sizes

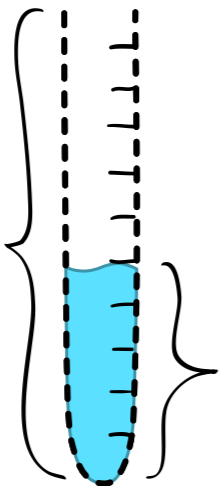
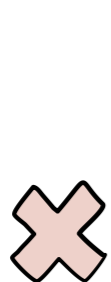


 **Gittins:** assign each job a **rank** based on **age** and S (lower is better)

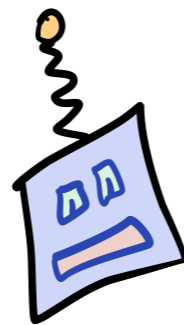
Unknown Job Sizes



size unknown 



age known



Gittins: assign each job a **rank** based on **age** and S (lower is better)



Gittins minimizes $E[T]$

SRPT and Gittins
minimize $E[T]$

known job sizes

SRPT and Gittins
minimize $E[T]$

known job sizes

sizes unknown,
partially known, known
(subsumes SRPT), ...

SRPT and Gittins
minimize $E[T]$

known job sizes

sizes unknown,
partially known, known
(subsumes SRPT), ...

SRPT and Gittins
minimize $E[T]$

Why *not* use Gittins?

Gittins Assumption

Computer System Reality

Gittins Assumption

Computer System Reality

Single server

Gittins Assumption

Single server

Computer System Reality

Multiple servers

Gittins Assumption

Single server

Complicated implementation
not a problem

Computer System Reality

Multiple servers

Gittins Assumption

Single server

Complicated implementation
not a problem

Computer System Reality

Multiple servers

Simple implementation
preferred

Gittins Assumption

Single server

Complicated implementation
not a problem

Preemption *unrestricted*
with *no cost*

Computer System Reality

Multiple servers

Simple implementation
preferred

Gittins Assumption

Single server

Complicated implementation
not a problem

Preemption *unrestricted*
with *no cost*

Computer System Reality

Multiple servers

Simple implementation
preferred

Preemption *restricted*
and/or *costly*

Gittins Assumption

Single server

Complicated implementation
not a problem

Preemption *unrestricted*
with *no cost*

Arbitrarily many
priority levels

Computer System Reality

Multiple servers

Simple implementation
preferred

Preemption *restricted*
and/or *costly*

Gittins Assumption

Single server

Complicated implementation
not a problem

Preemption *unrestricted*
with *no cost*

Arbitrarily many
priority levels

Computer System Reality

Multiple servers

Simple implementation
preferred

Preemption *restricted*
and/or *costly*

Limited number
of priority levels

Gittins Assumption

Single server

Complicated implementation
not a problem

Preemption *unrestricted*
with *no cost*

Arbitrarily many
priority levels

Goal is minimizing
mean response time

Computer System Reality

Multiple servers

Simple implementation
preferred

Preemption *restricted*
and/or *costly*

Limited number
of priority levels

Gittins Assumption

Single server

Complicated implementation
not a problem

Preemption *unrestricted*
with *no cost*

Arbitrarily many
priority levels

Goal is minimizing
mean response time

Computer System Reality

Multiple servers

Simple implementation
preferred

Preemption *restricted*
and/or *costly*

Limited number
of priority levels

Want to optimize *other*
response time metrics

Gittins Assumption

Single server

*Complicated implementation
not a problem*

*Preemption unrestricted
with no cost*

*Arbitrarily many
priority levels*

*Goal is minimizing
mean response time*

Computer System Reality

Multiple servers

*Simple implementation
preferred*

*Preemption restricted
and/or costly*

*Limited number
of priority levels*

*Want to optimize other
response time metrics*

Gittins Assumption

Single server

*Complicated implementation
not a problem*

*Preemption unrestricted
with no cost*

*Arbitrarily many
priority levels*

*Goal is minimizing
mean response time*

Easy

Computer System Reality

Multiple servers

*Simple implementation
preferred*

*Preemption restricted
and/or costly*

*Limited number
of priority levels*

*Want to optimize other
response time metrics*

Gittins Assumption

Single server

*Complicated implementation
not a problem*

*Preemption unrestricted
with no cost*

*Arbitrarily many
priority levels*

*Goal is minimizing
mean response time*

Easy

Computer System Reality

Multiple servers

*Simple implementation
preferred*

*Preemption restricted
and/or costly*

*Limited number
of priority levels*

*Want to optimize other
response time metrics*

Hard

Gittins Assumption

Single server

*Complicated implementation
not a problem*

*Preemption unrestricted
with no cost*

*Arbitrarily many
priority levels*

*Goal is minimizing
mean response time*

Easy

Computer System Reality

Multiple servers

*Simple implementation
preferred*

*Preemption restricted
and/or costly*

*Limited number
of priority levels*

*Want to optimize other
response time metrics*

Hard

... and in this talk!

I work on inventing

new queueing-theoretic tools

for solving

practical scheduling problems

Overview

Goals

Multiple servers

Simple implementation preferred

Preemption restricted and/or costly

Limited number of priority levels

Want to optimize other response time metrics

Overview

New Tools

Goals

Multiple servers

Simple implementation preferred

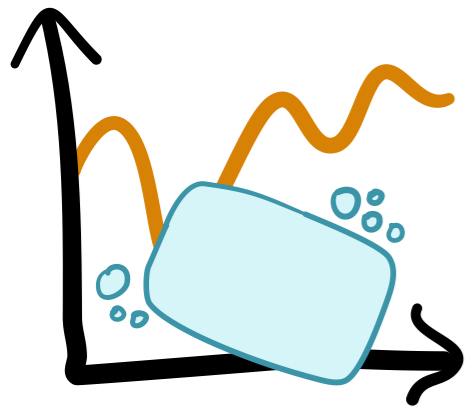
Preemption restricted and/or costly

Limited number of priority levels

Want to optimize other response time metrics

Overview

New Tools



SOAP

*analyzes a huge variety
of scheduling heuristics*

Goals

Multiple servers

*Simple implementation
preferred*

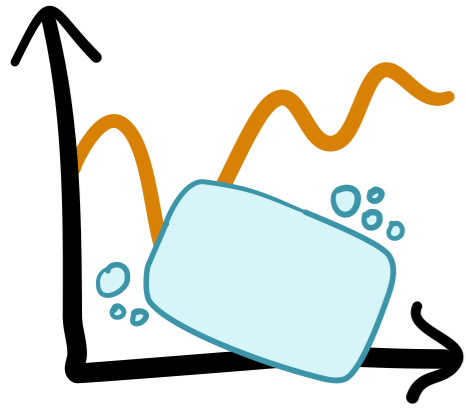
*Preemption restricted
and/or costly*

*Limited number
of priority levels*

*Want to optimize other
response time metrics*

Overview

New Tools



SOAP

analyzes a huge variety of scheduling heuristics

r-Work

provides a new, deeper understanding of Gittins

Goals

Multiple servers

Simple implementation preferred

Preemption restricted and/or costly

Limited number of priority levels

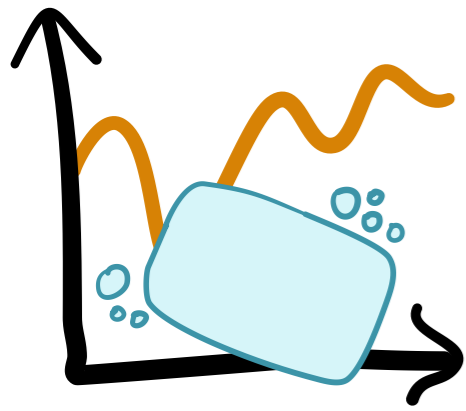
Want to optimize other response time metrics

Overview

New Tools

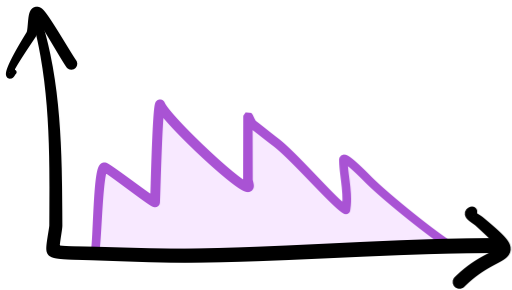
SOAP

analyzes a huge variety of scheduling heuristics



r-Work

provides a new, deeper understanding of Gittins



Goals

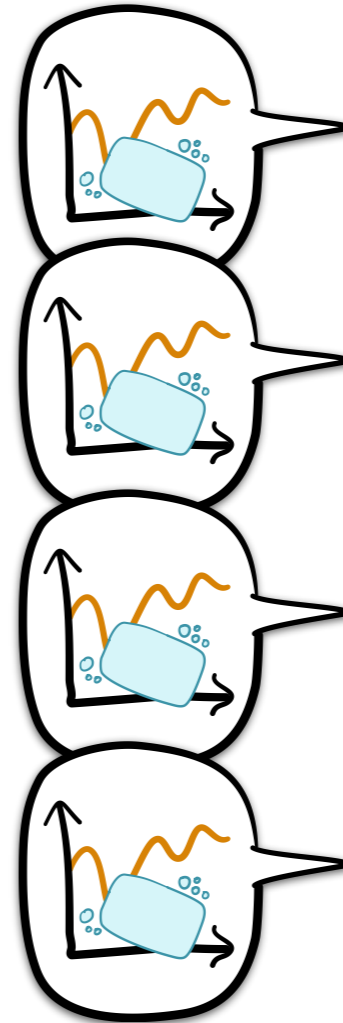
Multiple servers

Simple implementation preferred

Preemption restricted and/or costly

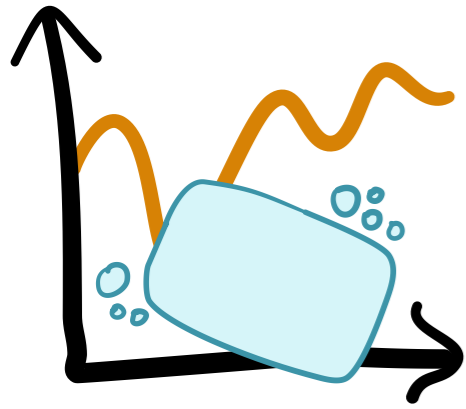
Limited number of priority levels

Want to optimize other response time metrics



Overview

New Tools



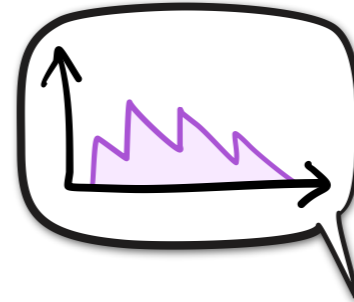
SOAP

analyzes a huge variety of scheduling heuristics

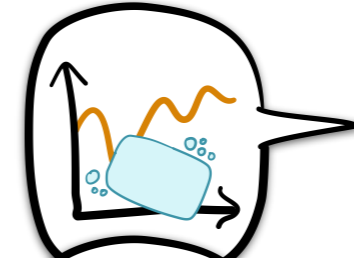
r-Work

provides a new, deeper understanding of Gittins

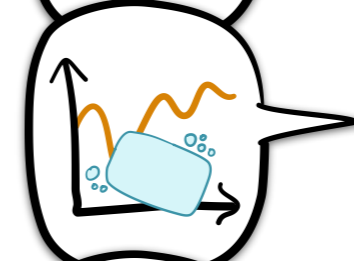
Goals



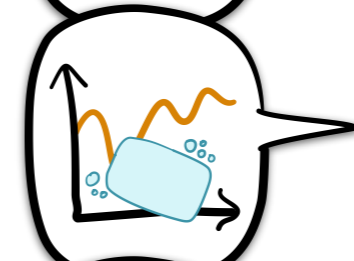
Multiple servers



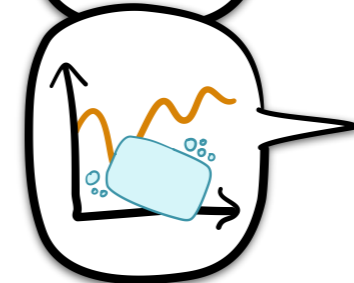
Simple implementation preferred



Preemption restricted and/or costly



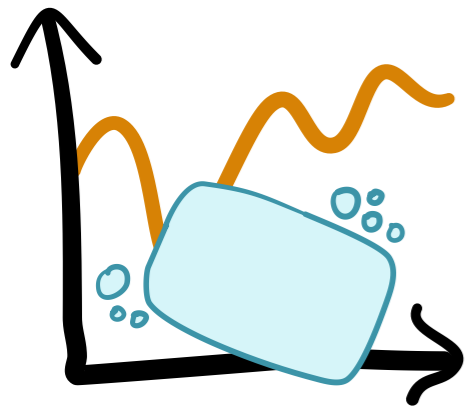
Limited number of priority levels



Want to optimize other response time metrics

Overview

New Tools



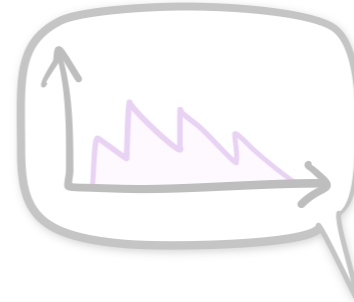
SOAP

analyzes a huge variety of scheduling heuristics

r-Work

provides a new, deeper understanding of Gittins

Goals



Multiple servers



Simple implementation preferred



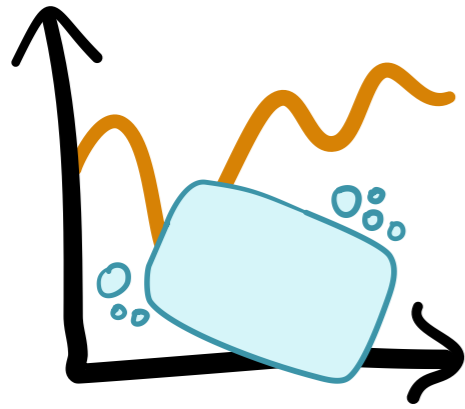
Preemption restricted and/or costly



Limited number of priority levels

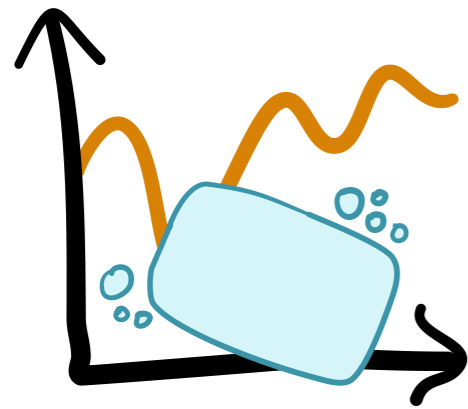


Want to optimize other response time metrics



SOAP

Schedule Ordered by Age-based Priority

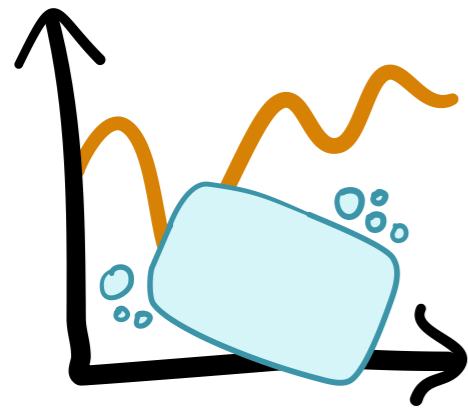


SOAP

Schedule Ordered by Age-based Priority

SOAP policies:

broad class of scheduling policies



SOAP

Schedule Ordered by Age-based Priority

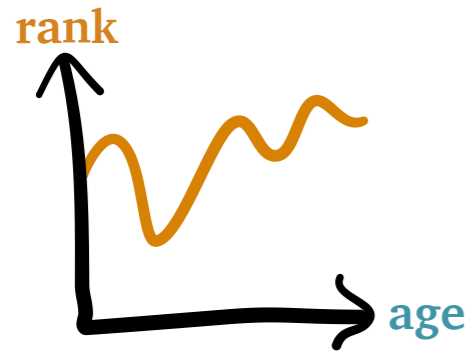
SOAP policies:

broad class of scheduling policies

SOAP analysis:

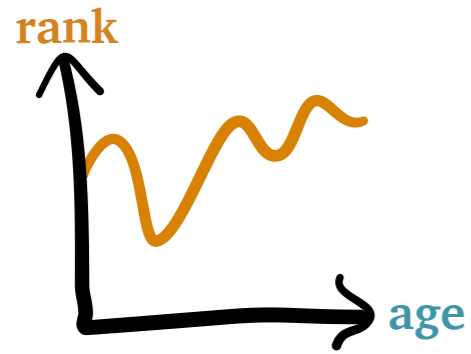
analyze response time of *any* **SOAP** policy

SOAP Policies



SOAP policy: any scheduling policy where a job's **rank** is a function of its **age**

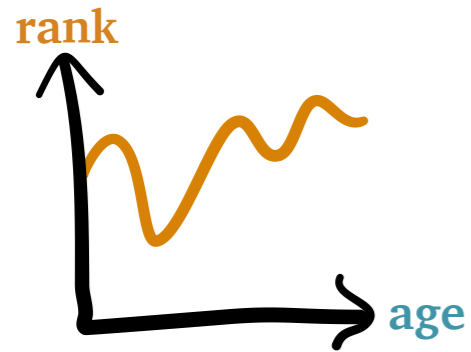
SOAP Policies



SOAP policy: any scheduling policy where a job's **rank** is a function of its **age**

priority, lower is better

SOAP Policies

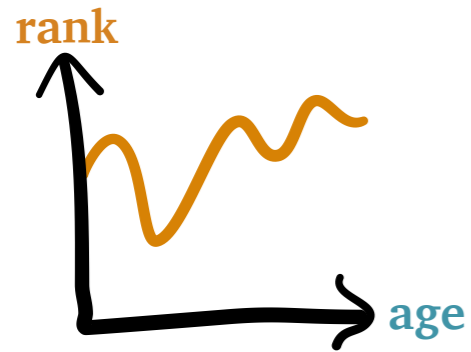


SOAP policy: any scheduling policy where a job's **rank** is a function of its **age**

priority, lower is better

service so far

SOAP Policies

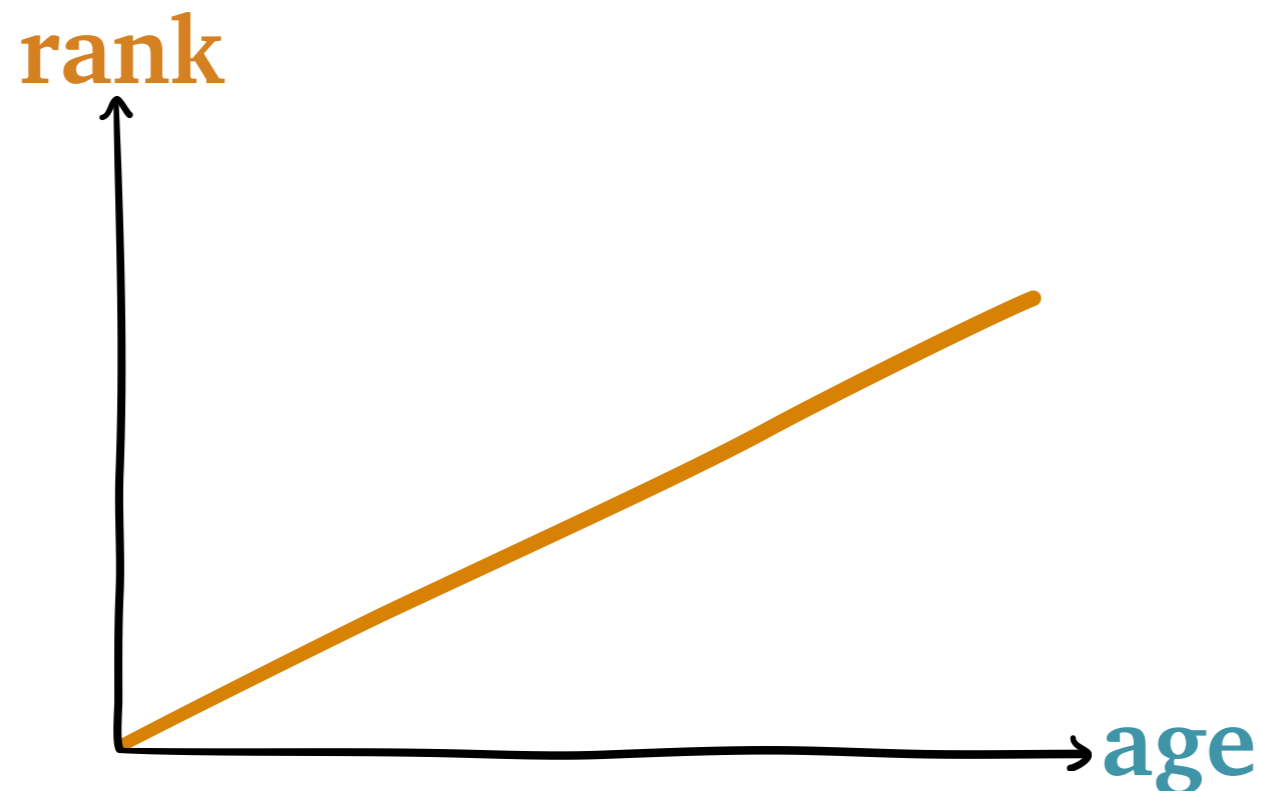


SOAP policy: any scheduling policy where a job's **rank** is a function of its **age**

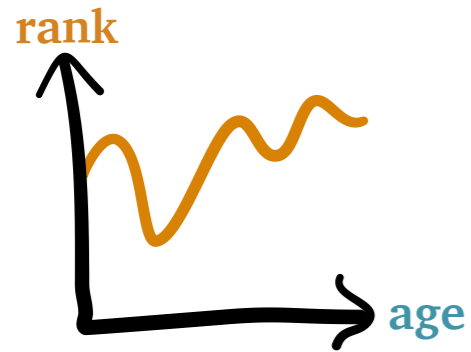
priority, lower is better

service so far

Foreground-Background (FB)



SOAP Policies

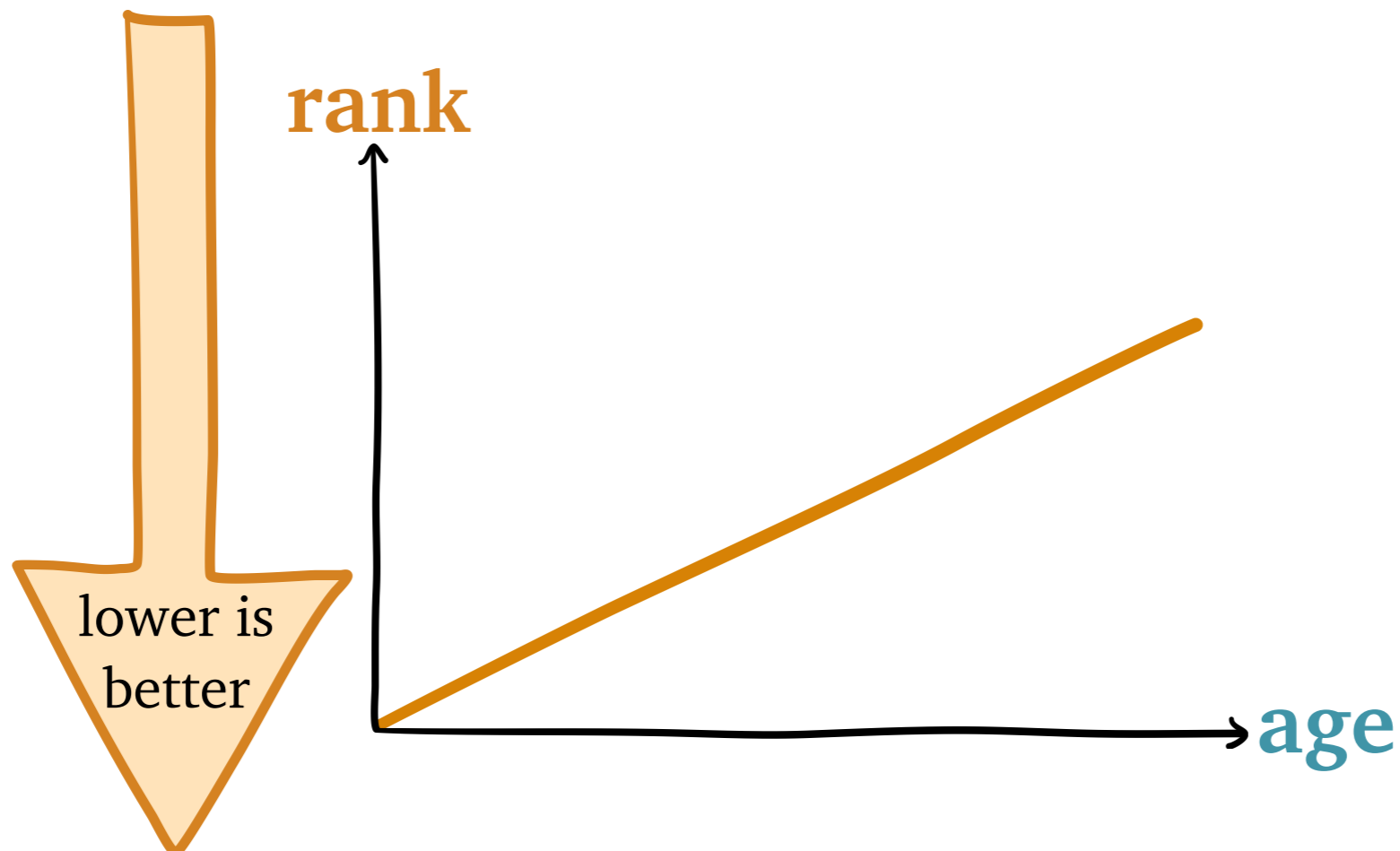


SOAP policy: any scheduling policy where a job's **rank** is a function of its **age**

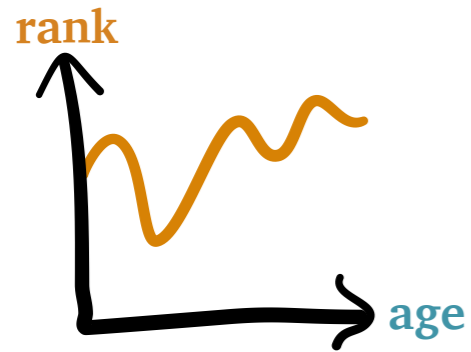
priority, lower is better

service so far

Foreground-Background (FB)



SOAP Policies

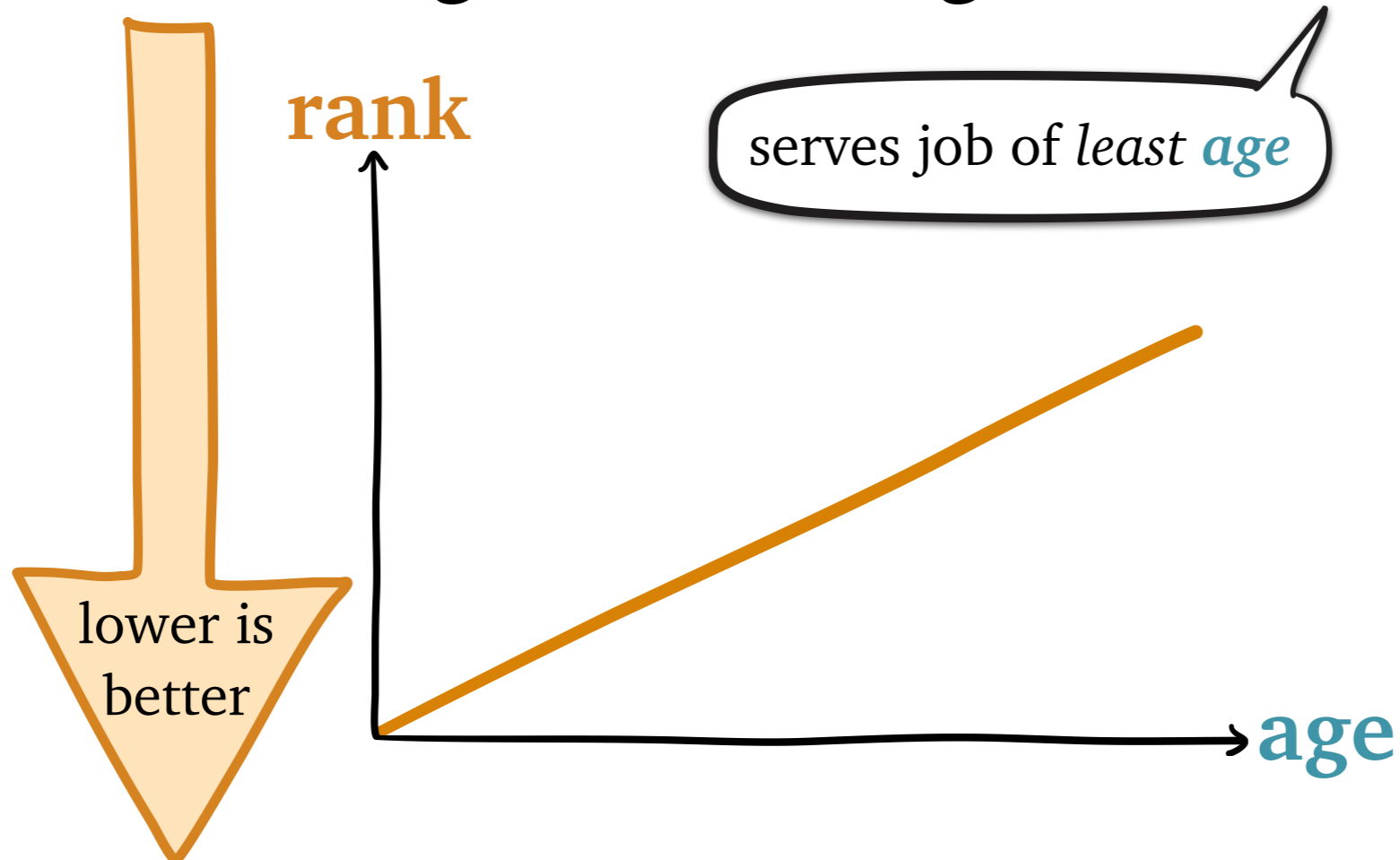


SOAP policy: any scheduling policy where a job's **rank** is a function of its **age**

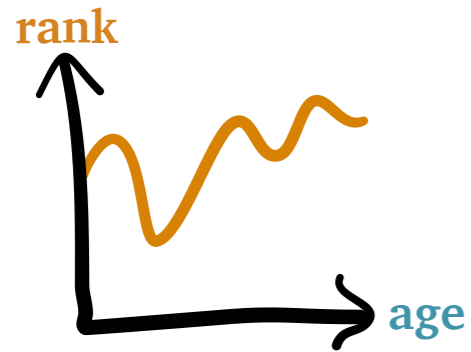
priority, lower is better

service so far

Foreground-Background (FB)



SOAP Policies



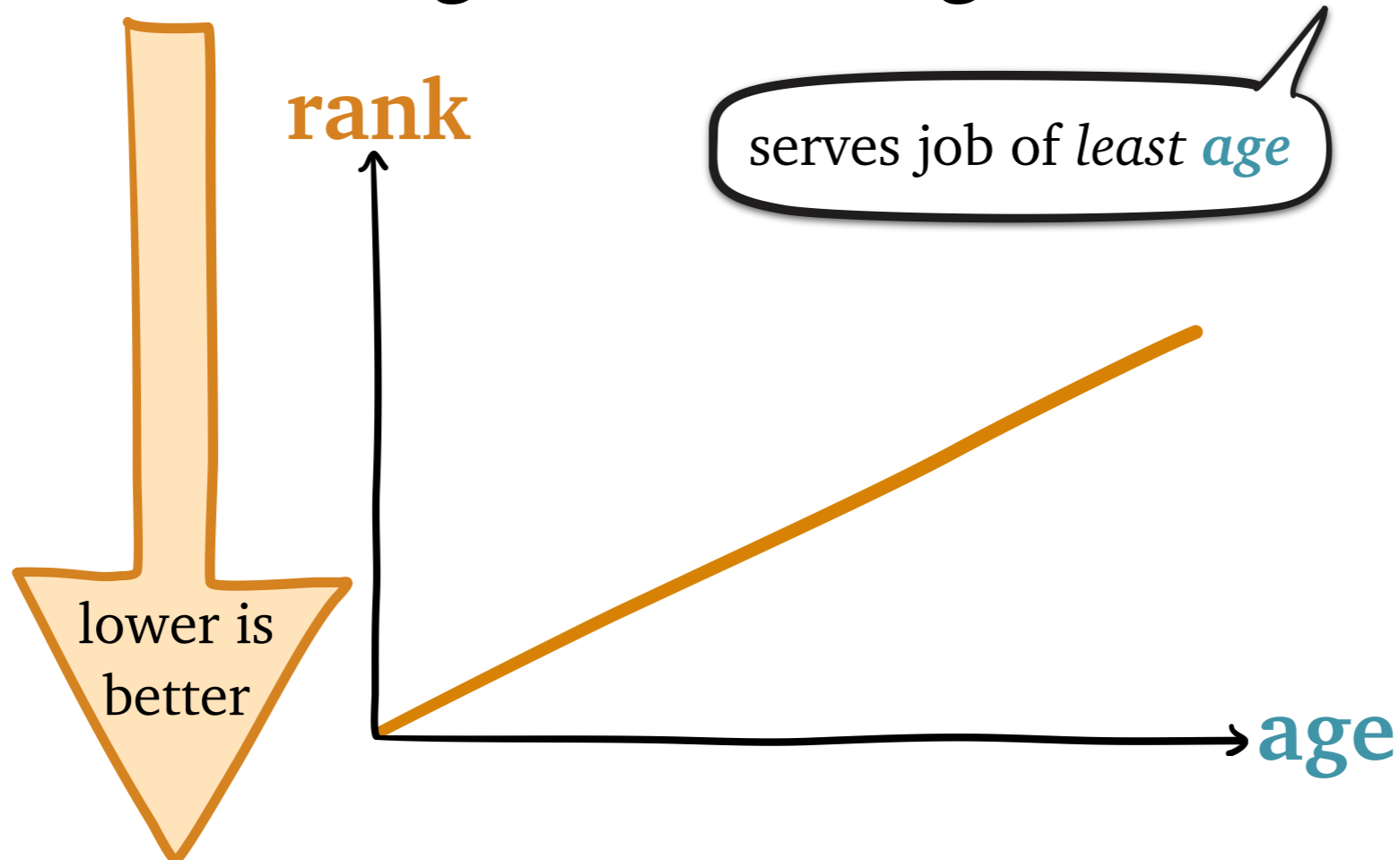
SOAP policy: any scheduling policy where a job's **rank** is a function of its **age**

priority, lower is better

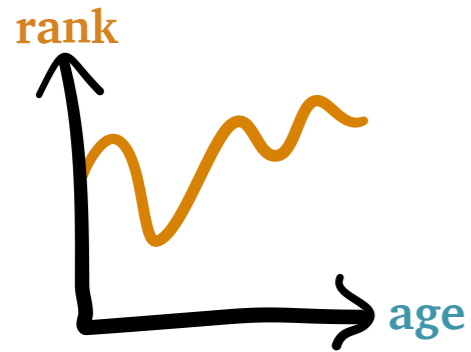
and other "static" info

service so far

Foreground-Background (FB)



SOAP Policies



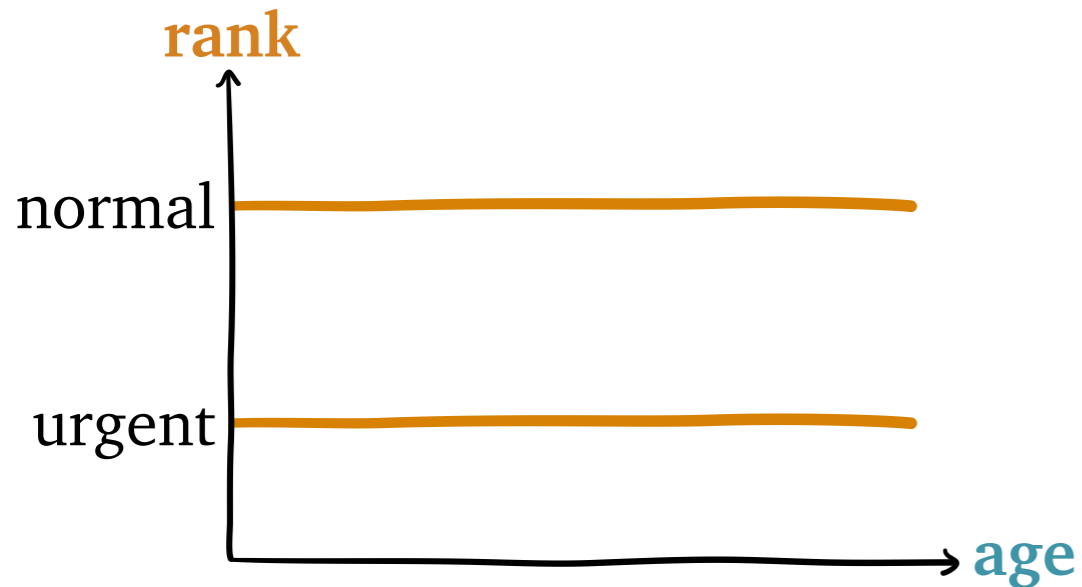
SOAP policy: any scheduling policy where a job's **rank** is a function of its **age**

priority, lower is better

and other "static" info

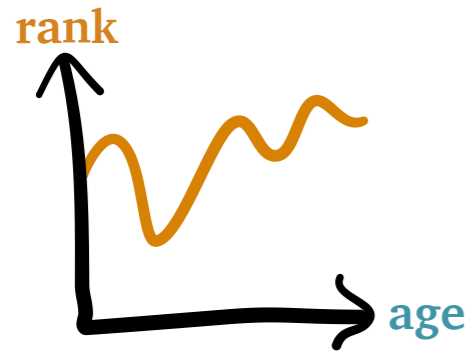
service so far

Preemptive Priority



lower is better

SOAP Policies



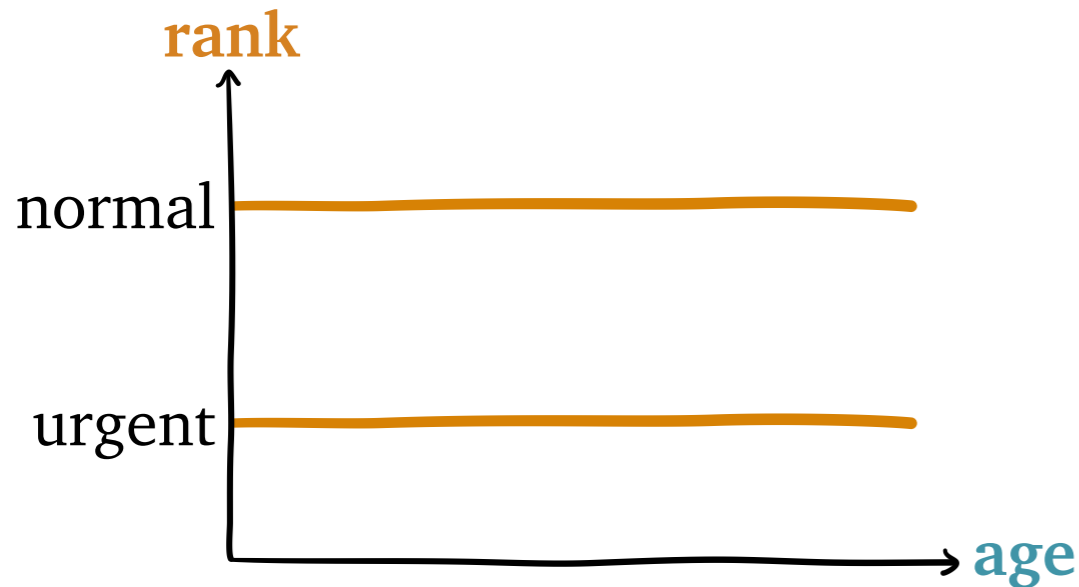
SOAP policy: any scheduling policy where a job's **rank** is a function of its **age**

priority, lower is better

and other "static" info

service so far

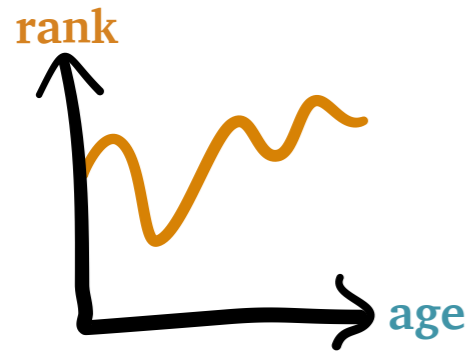
Preemptive Priority



break ties FCFS

lower is better

SOAP Policies



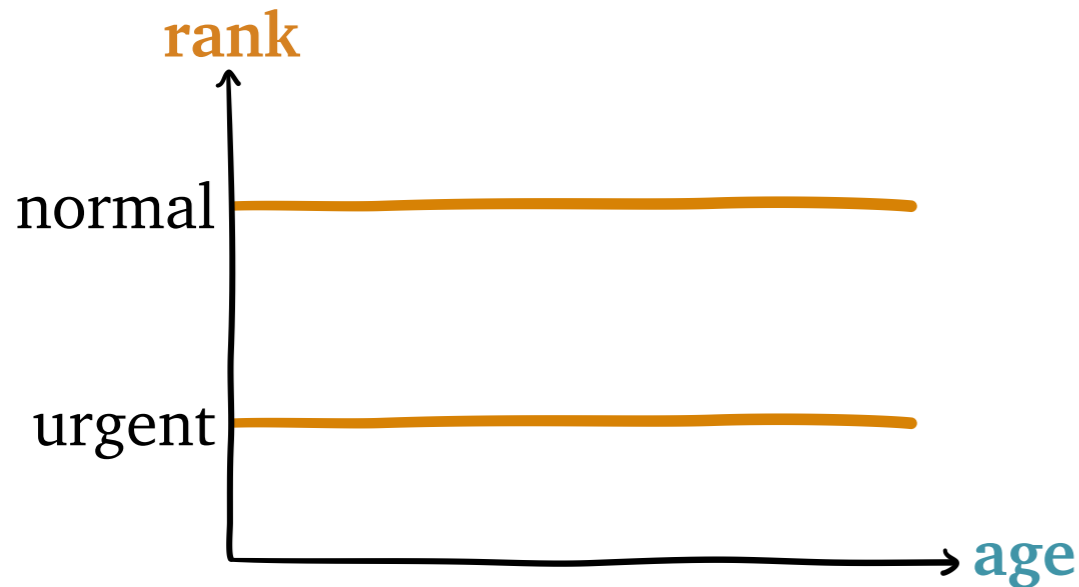
SOAP policy: any scheduling policy where a job's **rank** is a function of its **age**

priority, lower is better

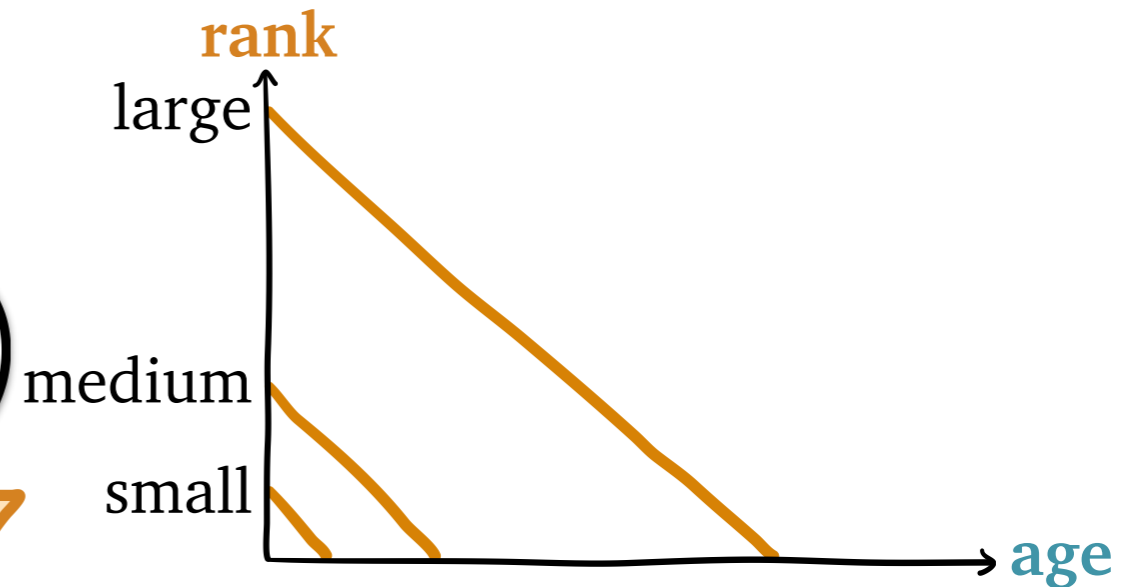
and other "static" info

service so far

Preemptive Priority



SRPT

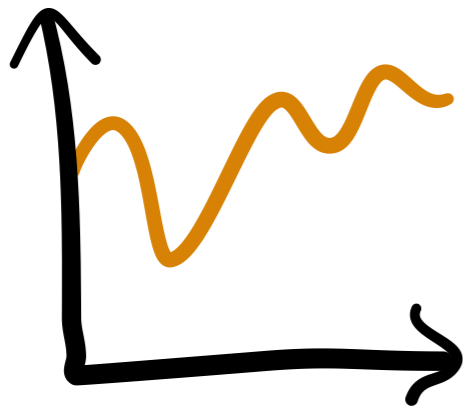


break ties FCFS

lower is better

SOAP Analysis

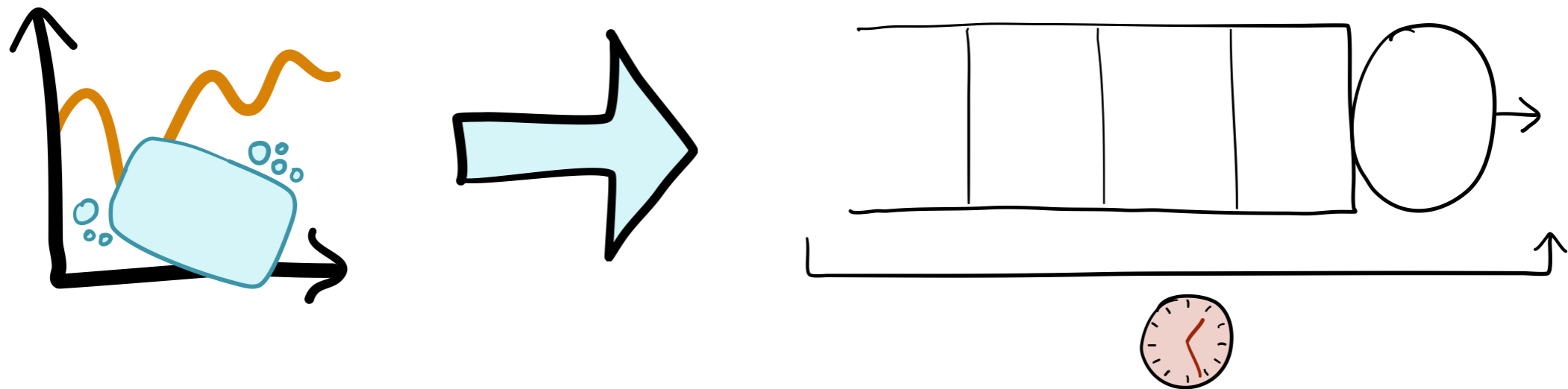
Given *any* **rank** function...



SOAP Analysis

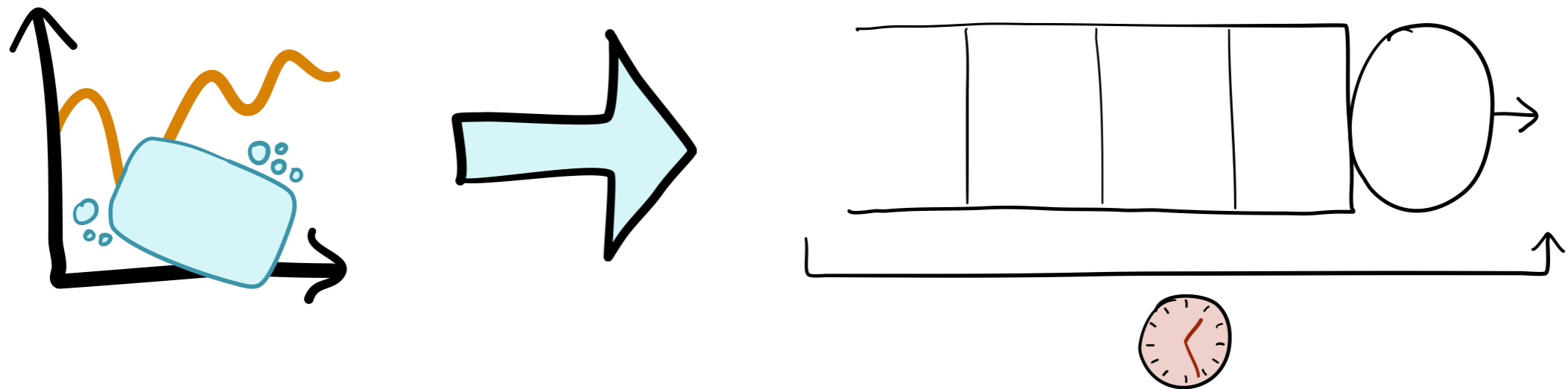
Given *any* **rank** function...

... **SOAP** analyzes its response time



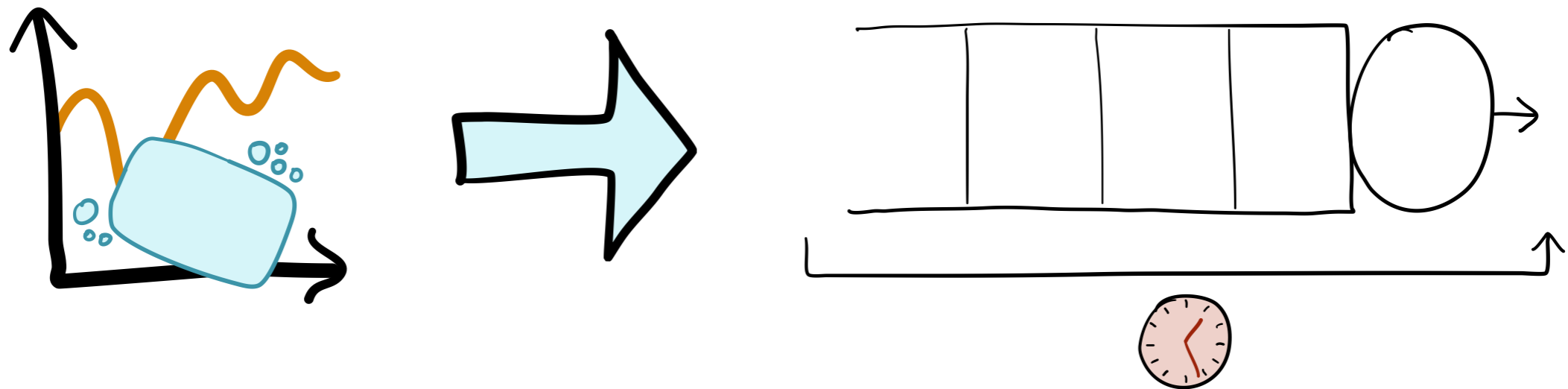
SOAP Analysis

Given *any* **rank** function... exact formula!
... **SOAP** analyzes its response time



SOAP Analysis

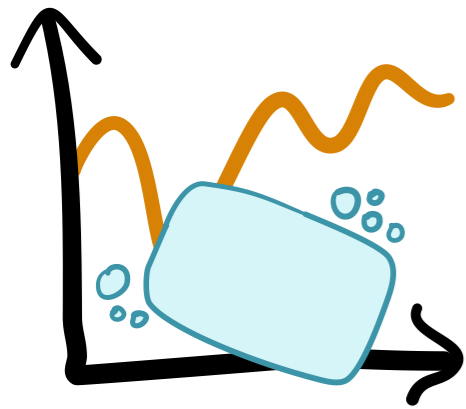
Given *any* **rank** function... exact formula!
... **SOAP** analyzes its response time



[Scully, Harchol-Balter, & Scheller-Wolf, SIGMETRICS 2018]

Overview

New Tools



SOAP

analyzes a huge variety of scheduling heuristics

r-Work

provides a new, deeper understanding of Gittins

Goals

Multiple servers

Simple implementation preferred

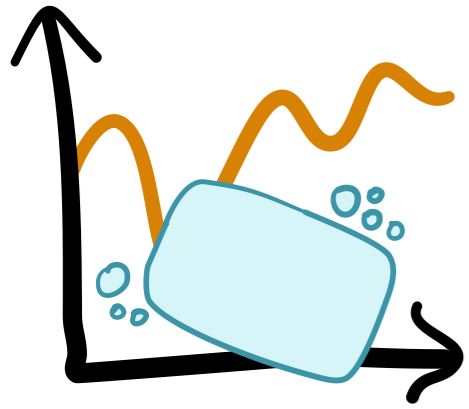
Preemption restricted and/or costly

Limited number of priority levels

Want to optimize other response time metrics

Overview

New Tools



SOAP

analyzes a huge variety of scheduling heuristics

r-Work

provides a new, deeper understanding of Gittins

Goals

Multiple servers

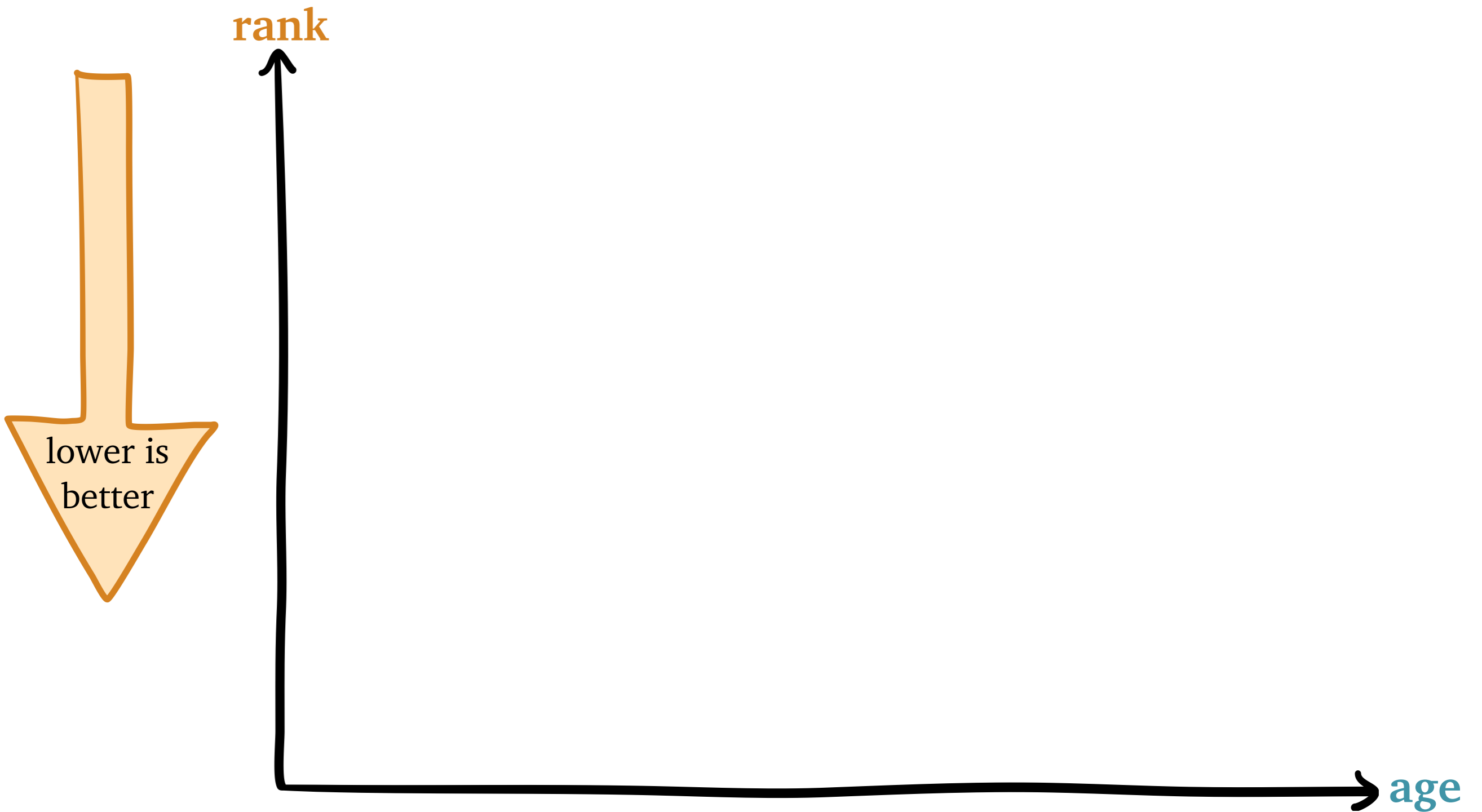
Simple implementation preferred

Preemption restricted and/or costly

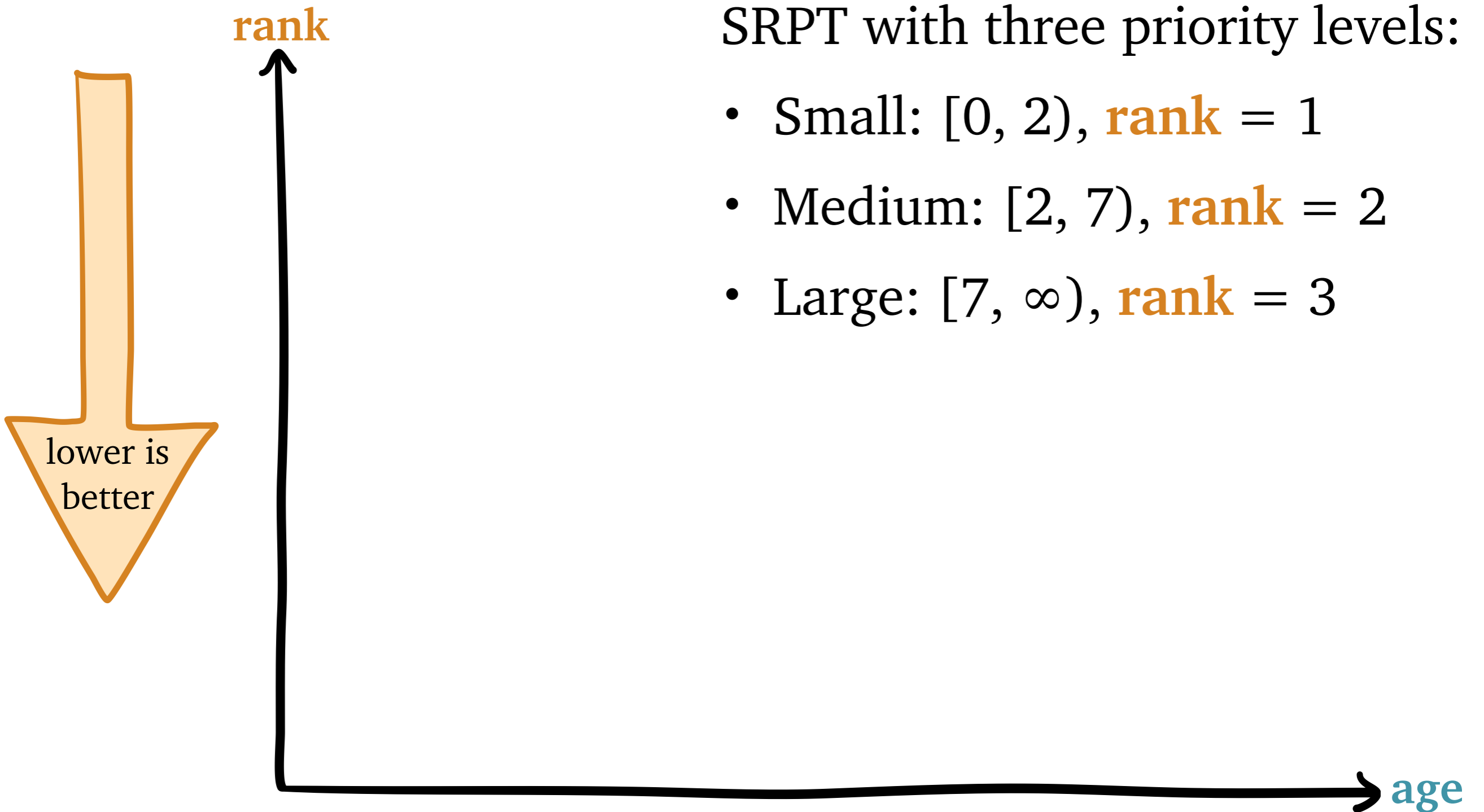
Limited number of priority levels

Want to optimize other response time metrics

Limited Priority Levels



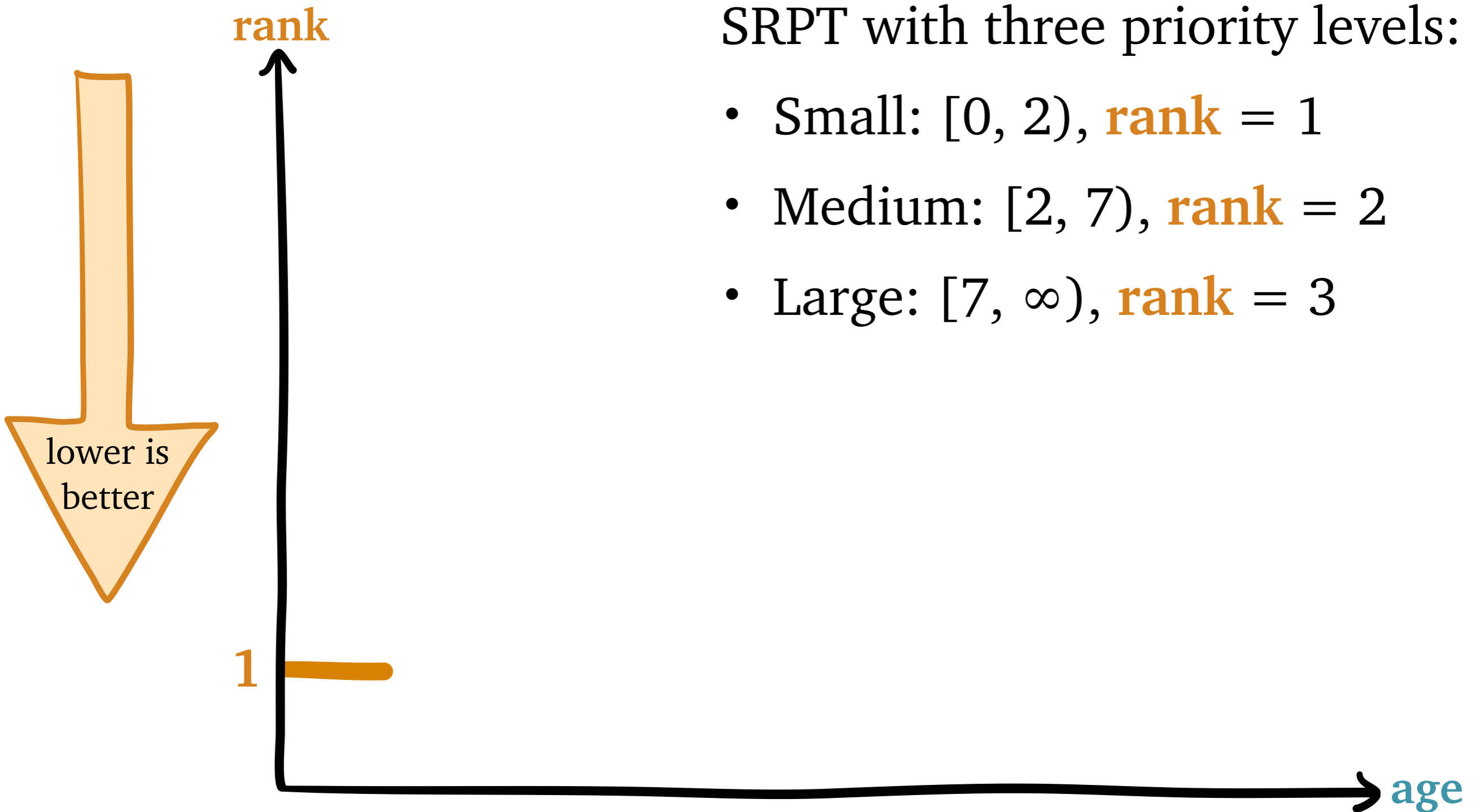
Limited Priority Levels



Limited Priority Levels

SRPT with three priority levels:

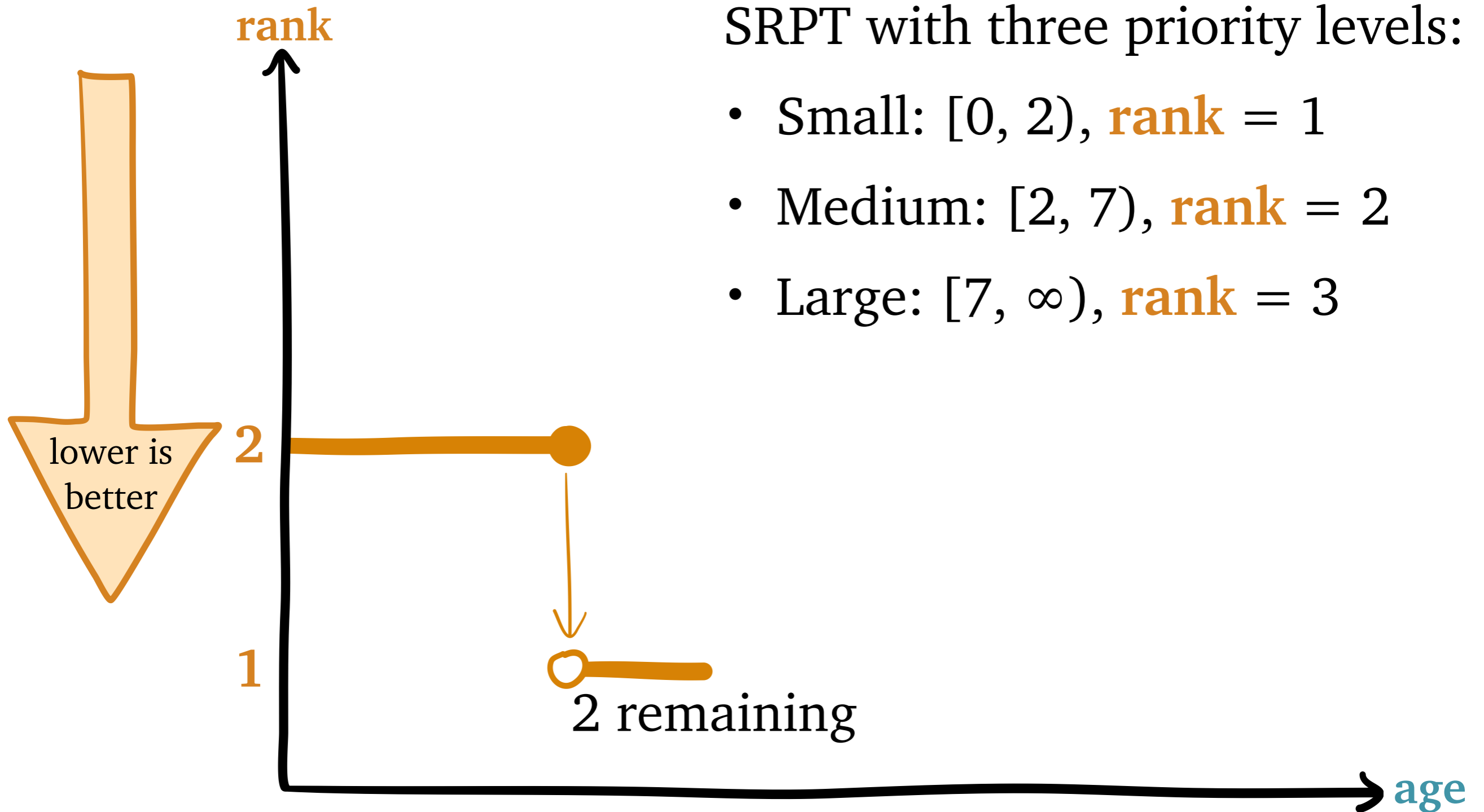
- Small: $[0, 2)$, **rank** = 1
- Medium: $[2, 7)$, **rank** = 2
- Large: $[7, \infty)$, **rank** = 3



Limited Priority Levels

SRPT with three priority levels:

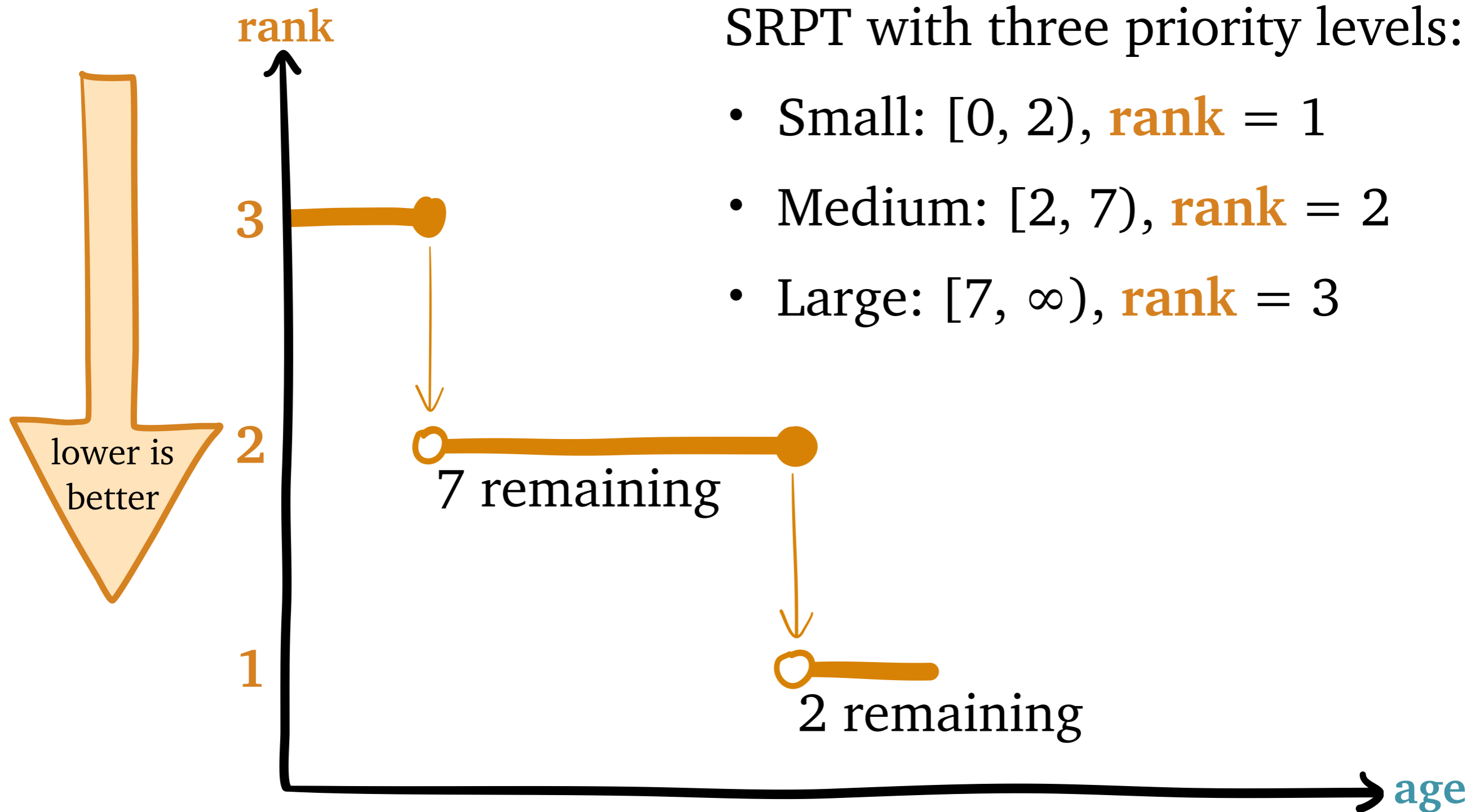
- Small: $[0, 2)$, **rank** = 1
- Medium: $[2, 7)$, **rank** = 2
- Large: $[7, \infty)$, **rank** = 3



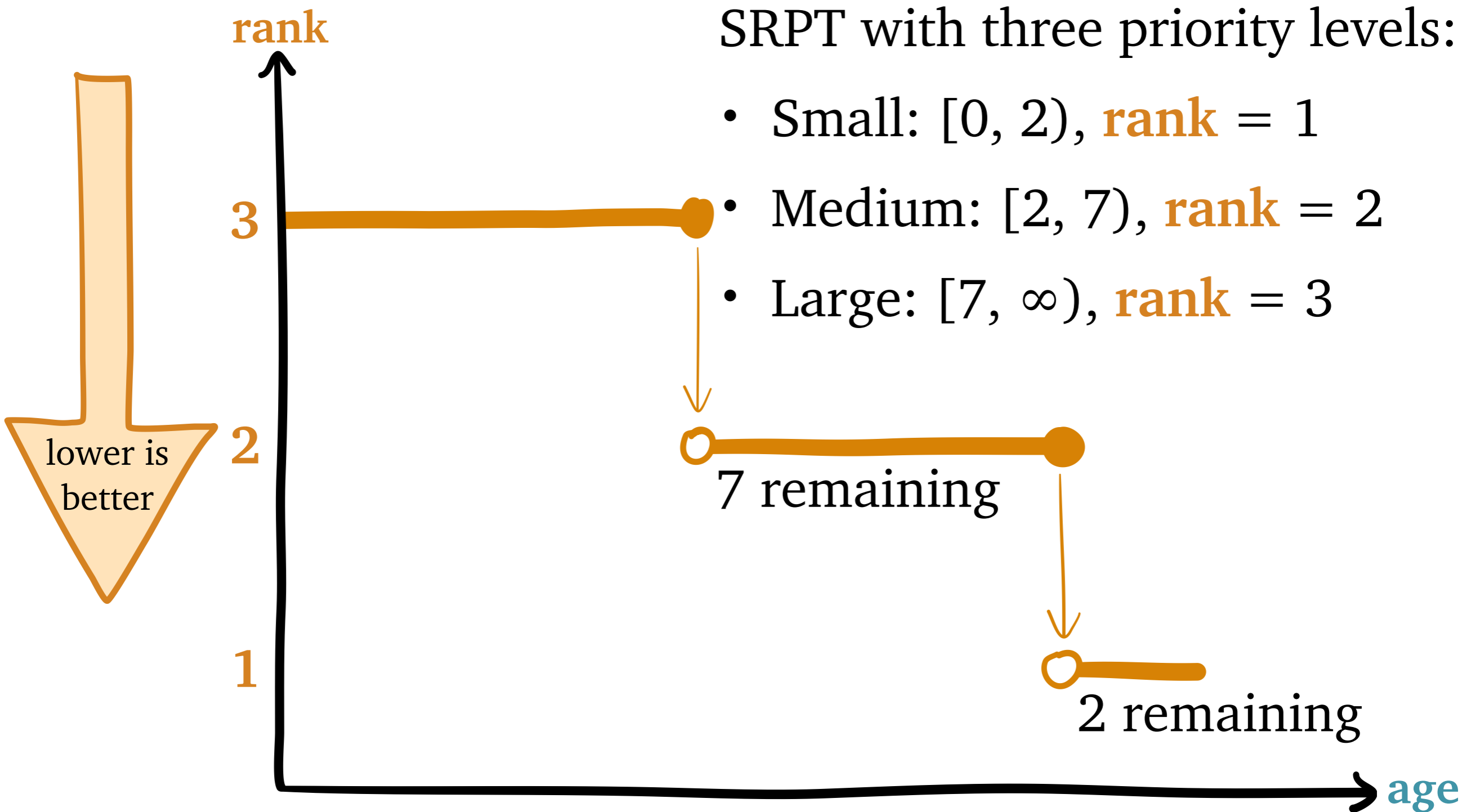
Limited Priority Levels

SRPT with three priority levels:

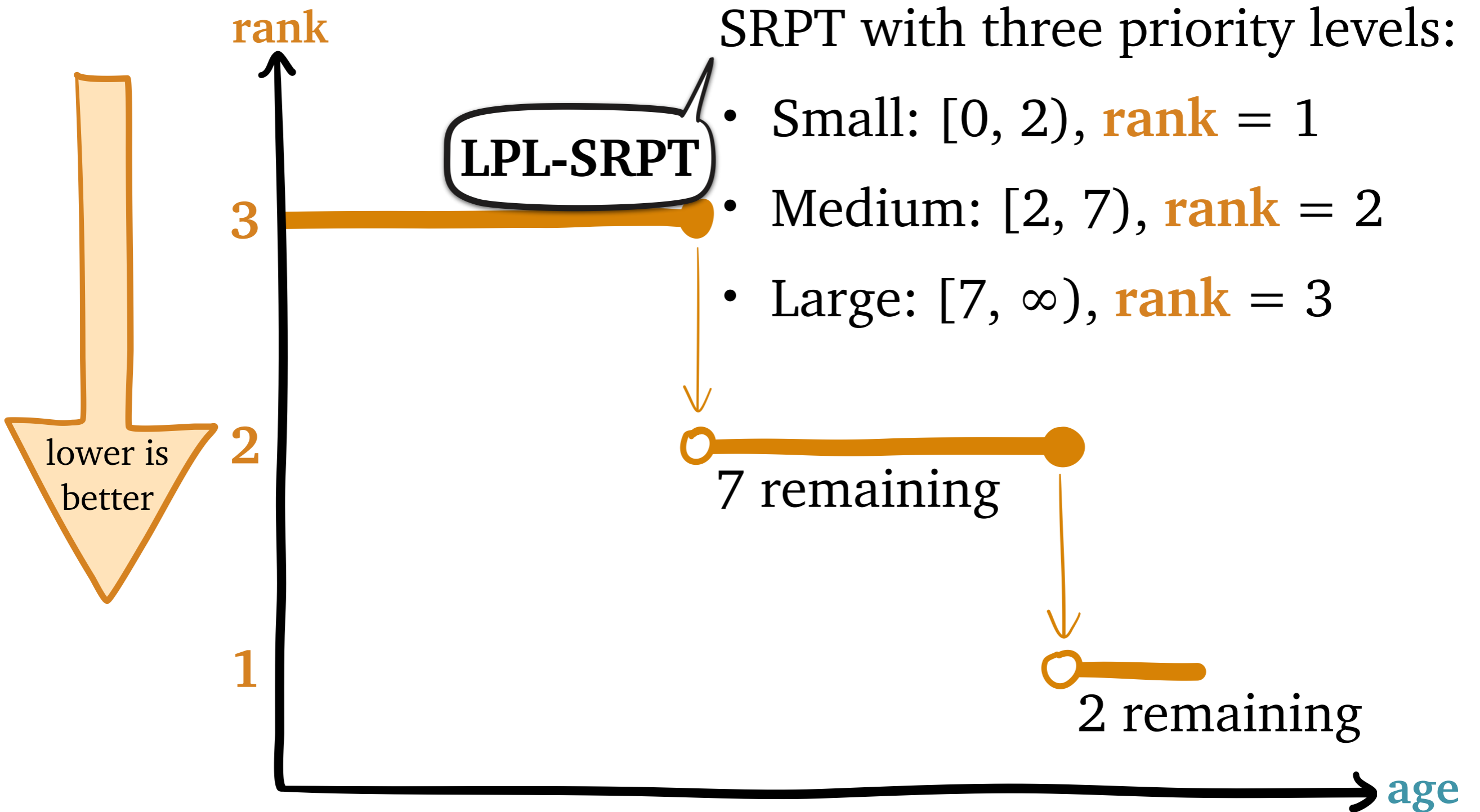
- Small: $[0, 2)$, **rank** = 1
- Medium: $[2, 7)$, **rank** = 2
- Large: $[7, \infty)$, **rank** = 3



Limited Priority Levels



Limited Priority Levels



LPL-SRPT Questions

- How many levels do we need?
- How do we choose size cutoffs?
- Can we do better than LPL-SRPT?

LPL-SRPT Questions

Uniform: 2-ish

*Bounded Pareto,
Weibull: 5-ish*

- How many levels do we need?
- How do we choose size cutoffs?
- Can we do better than LPL-SRPT?

LPL-SRPT Questions

Uniform: 2-ish

*Bounded Pareto,
Weibull: 5-ish*

Load-balancing
heuristic suffices

- How many levels do we need?
- How do we choose size cutoffs?
- Can we do better than LPL-SRPT?

LPL-SRPT Questions

Uniform: 2-ish

*Bounded Pareto,
Weibull: 5-ish*

Load-balancing
heuristic suffices

- How many levels do we need?
- How do we choose size cutoffs?
- Can we do better than LPL-SRPT?

Yes! LPL-PSJF
often better

LPL-SRPT Questions

Uniform: 2-ish

*Bounded Pareto,
Weibull: 5-ish*

Load-balancing
heuristic suffices

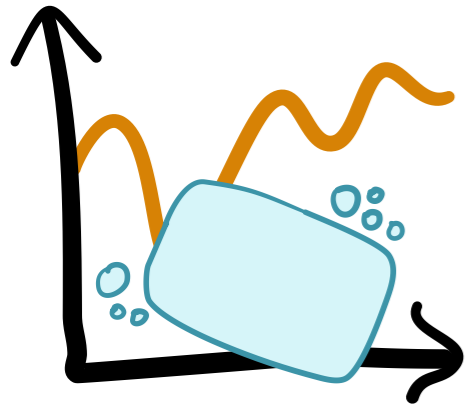
- How many levels do we need?
- How do we choose size cutoffs?
- Can we do better than LPL-SRPT?

Yes! LPL-PSJF
often better

[Scully & Harchol-Balter, in preparation]

Overview

New Tools



SOAP

analyzes a huge variety of scheduling heuristics

r-Work

provides a new, deeper understanding of Gittins

Goals

Multiple servers

Simple implementation preferred

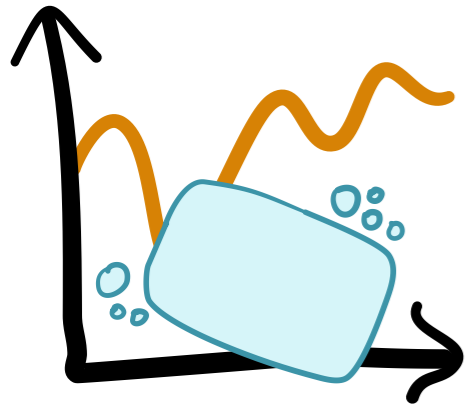
Preemption restricted and/or costly

Limited number of priority levels

Want to optimize other response time metrics

Overview

New Tools



SOAP

analyzes a huge variety of scheduling heuristics

Goals

Multiple servers

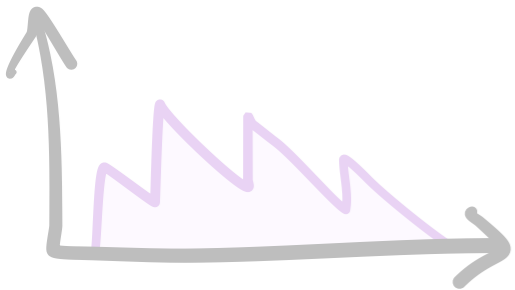
Simple implementation preferred

Preemption restricted and/or costly



Limited number of priority levels

Want to optimize other response time metrics

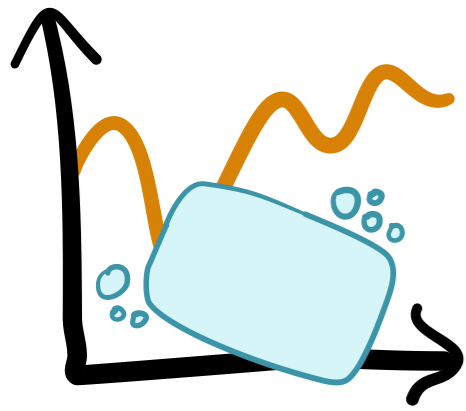


r-Work

provides a new, deeper understanding of Gittins

Overview

New Tools



SOAP

analyzes a huge variety of scheduling heuristics

r-Work

provides a new, deeper understanding of Gittins

Goals

Multiple servers

Simple implementation preferred

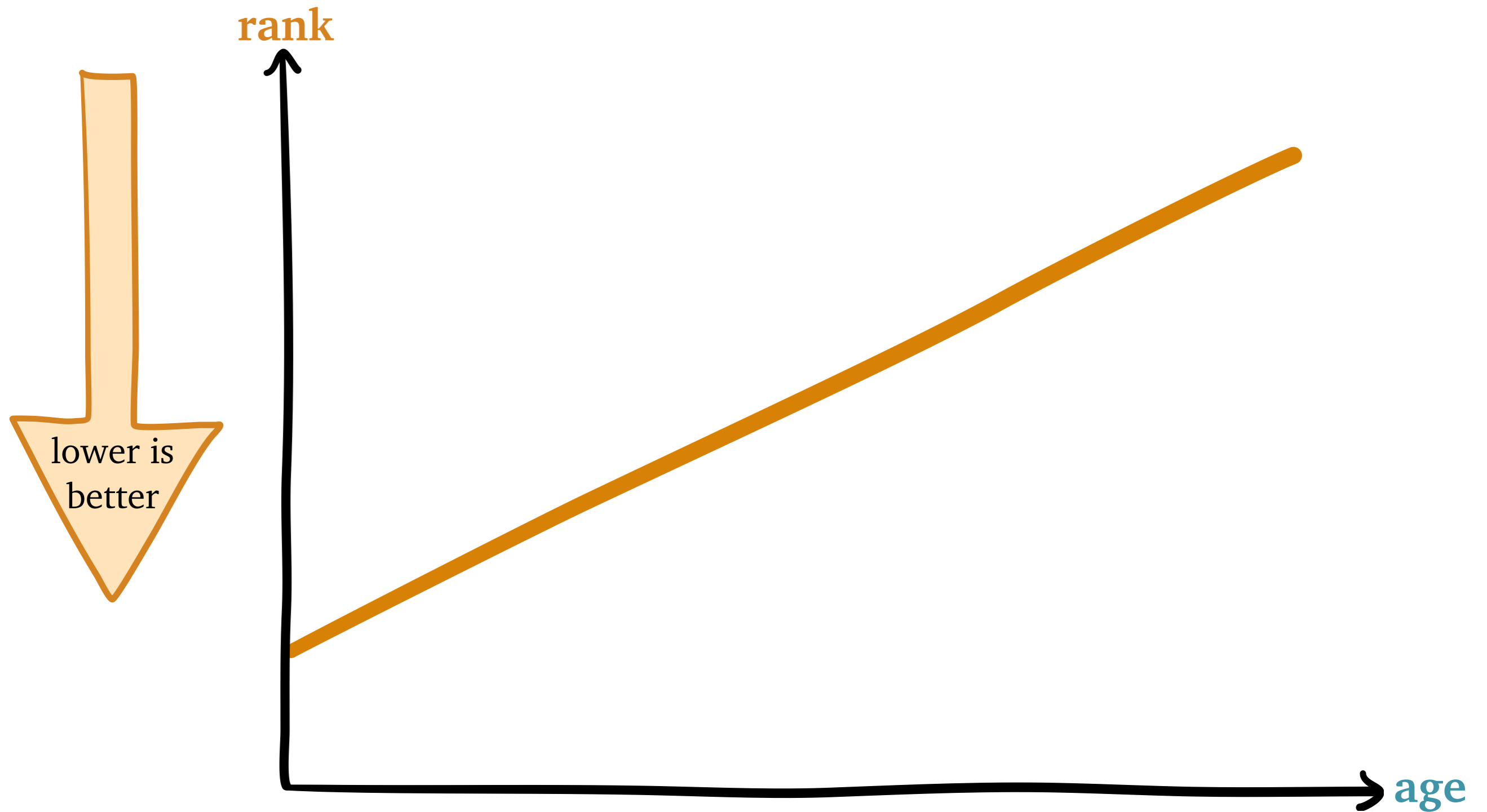
Preemption restricted and/or costly



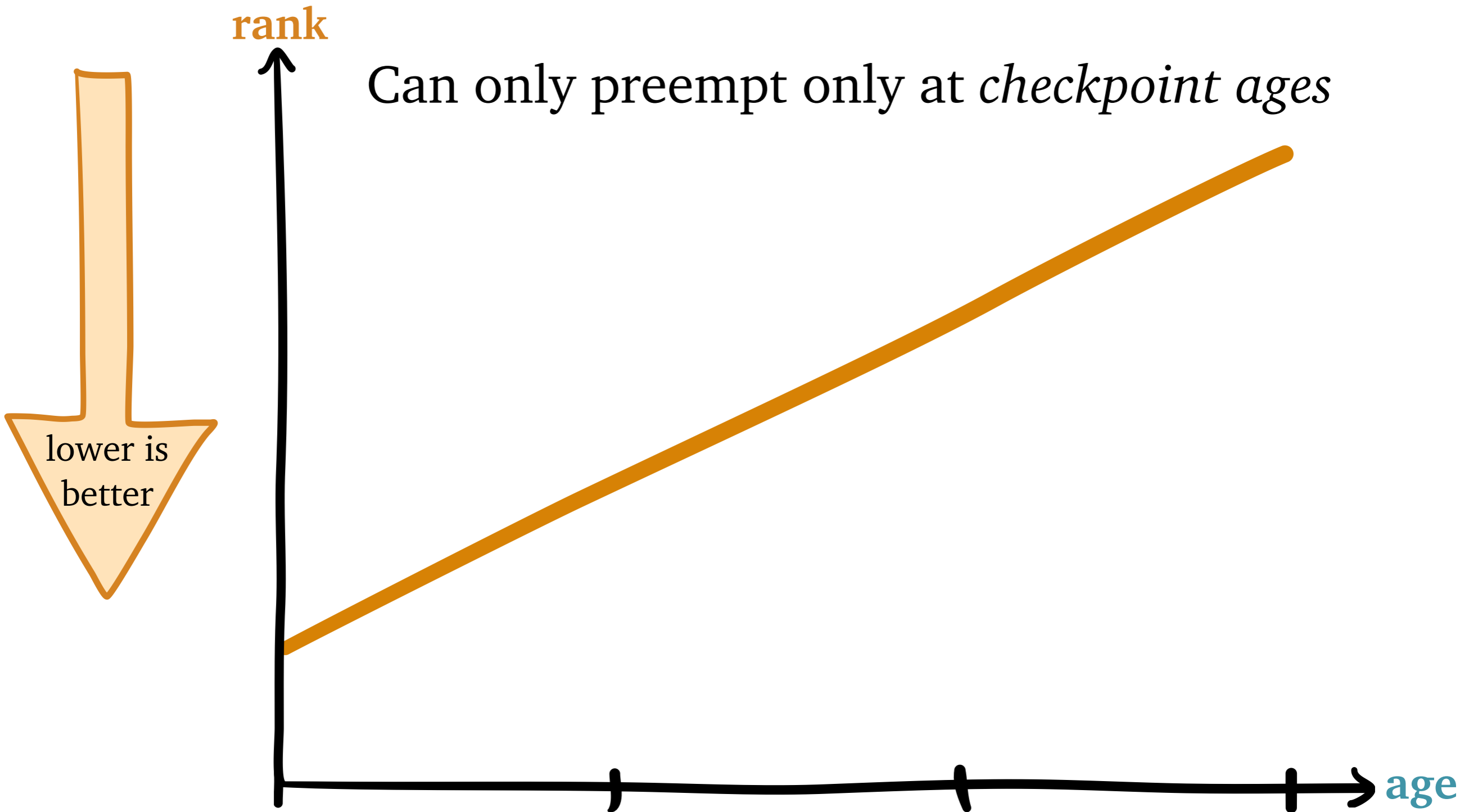
Limited number of priority levels

Want to optimize other response time metrics

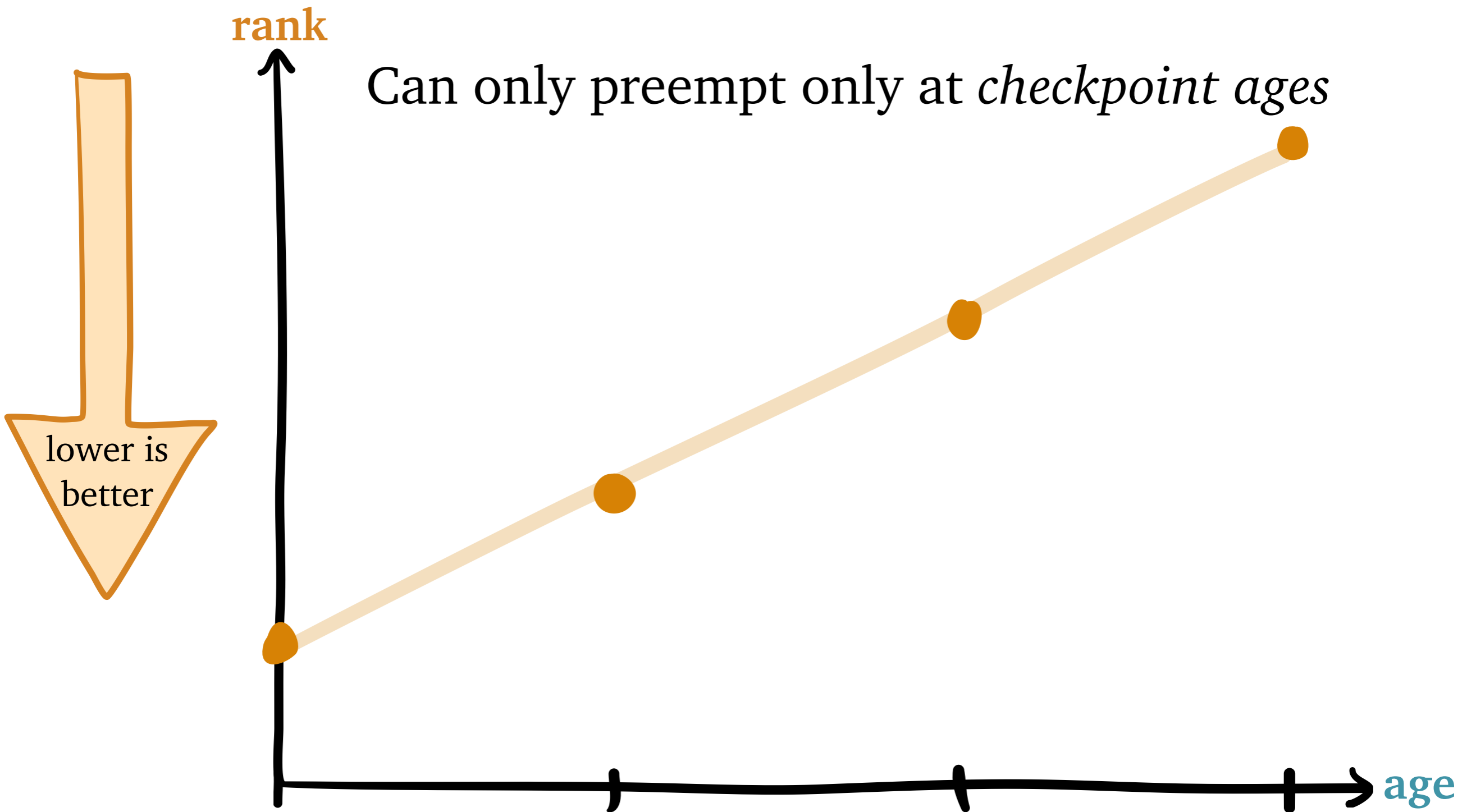
Preemption Checkpoints



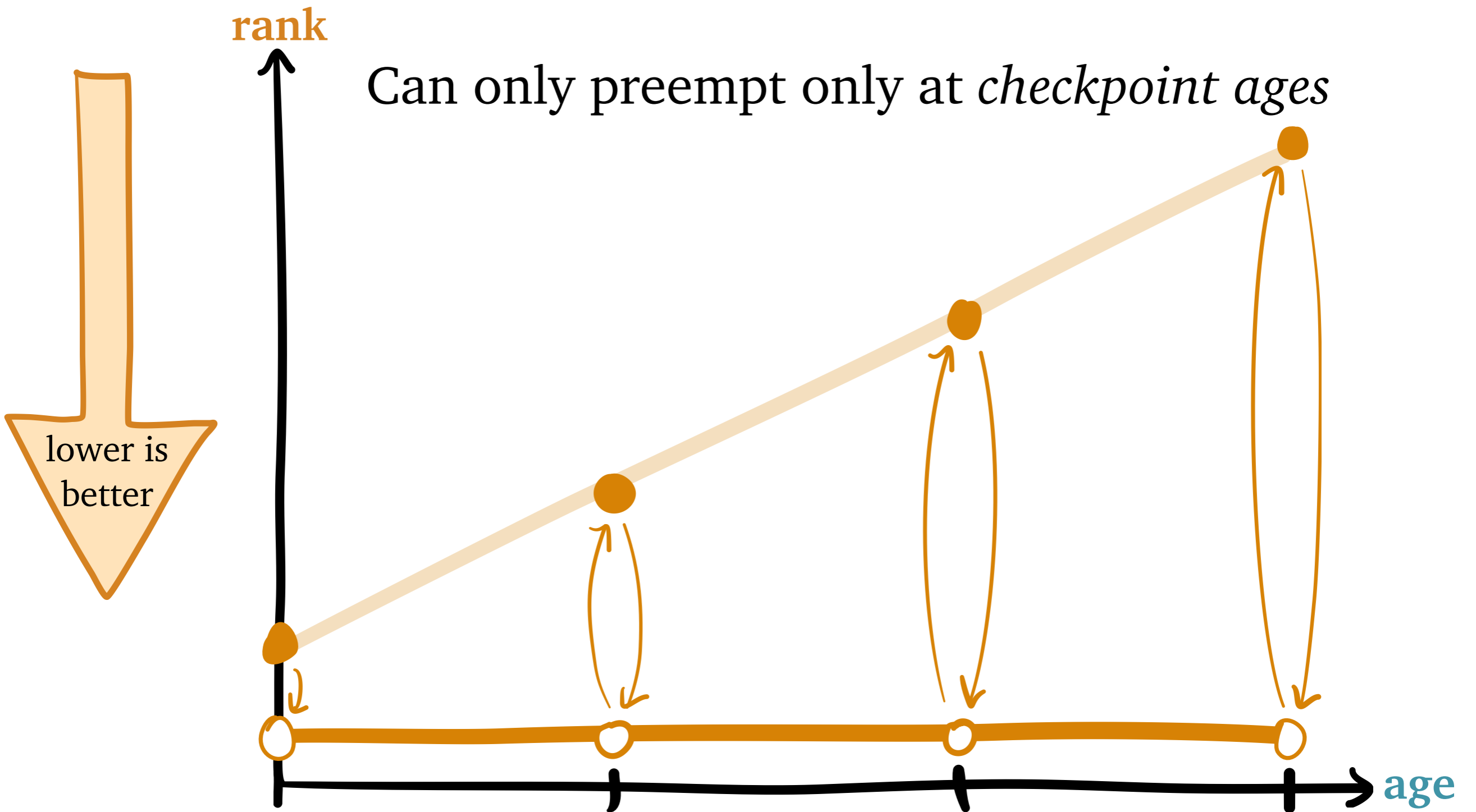
Preemption Checkpoints



Preemption Checkpoints



Preemption Checkpoints



Checkpoint Frequency?

Suppose each checkpoint incurs an overhead

What is the optimal gap Δ between checkpoints?

Checkpoint Frequency?

Suppose each checkpoint incurs an overhead

What is the optimal gap Δ between checkpoints?

- large Δ : less overhead

Checkpoint Frequency?

Suppose each checkpoint incurs an overhead

What is the optimal gap Δ between checkpoints?

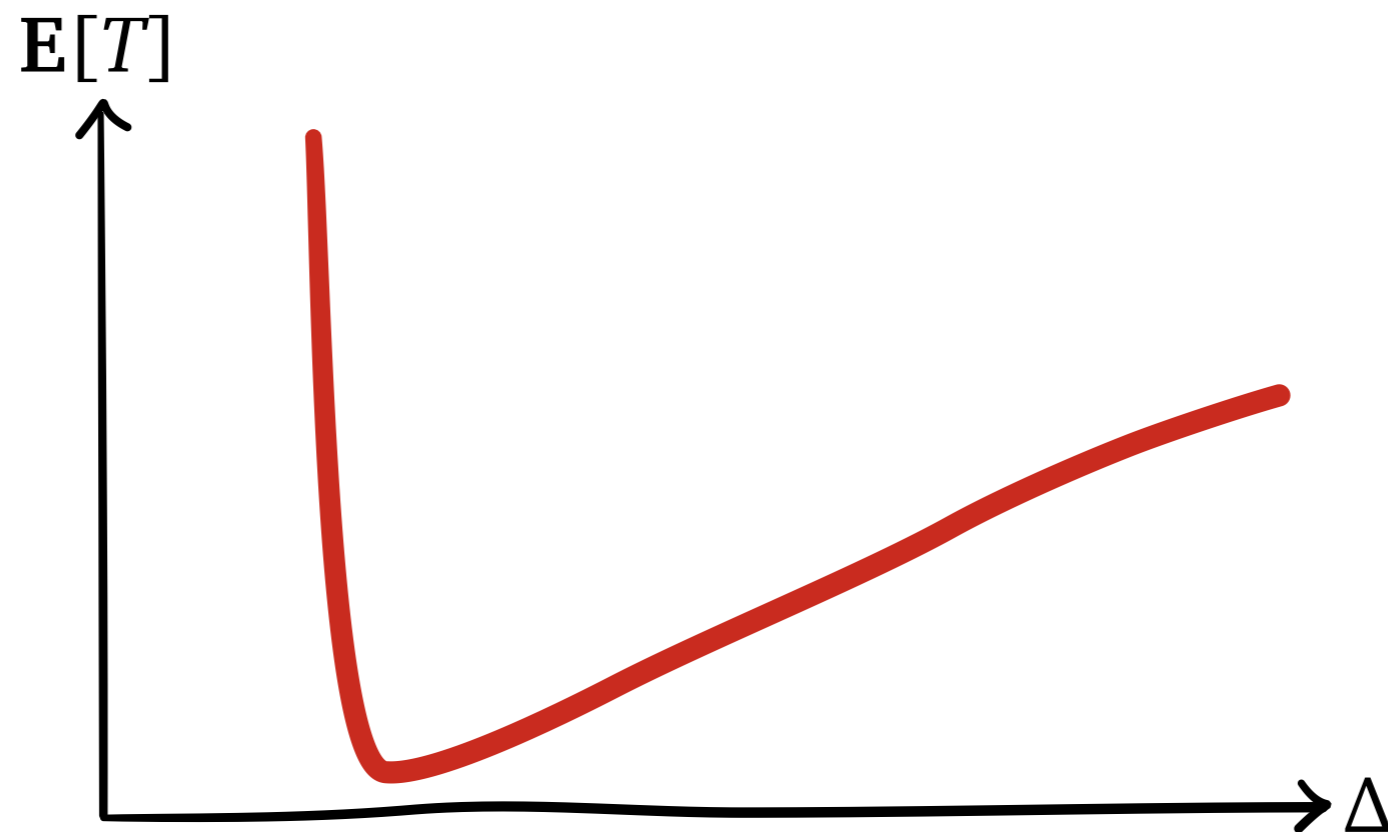
- large Δ : less overhead
- small Δ : better scheduling

Checkpoint Frequency?

Suppose each checkpoint incurs an overhead

What is the optimal gap Δ between checkpoints?

- large Δ : less overhead
- small Δ : better scheduling

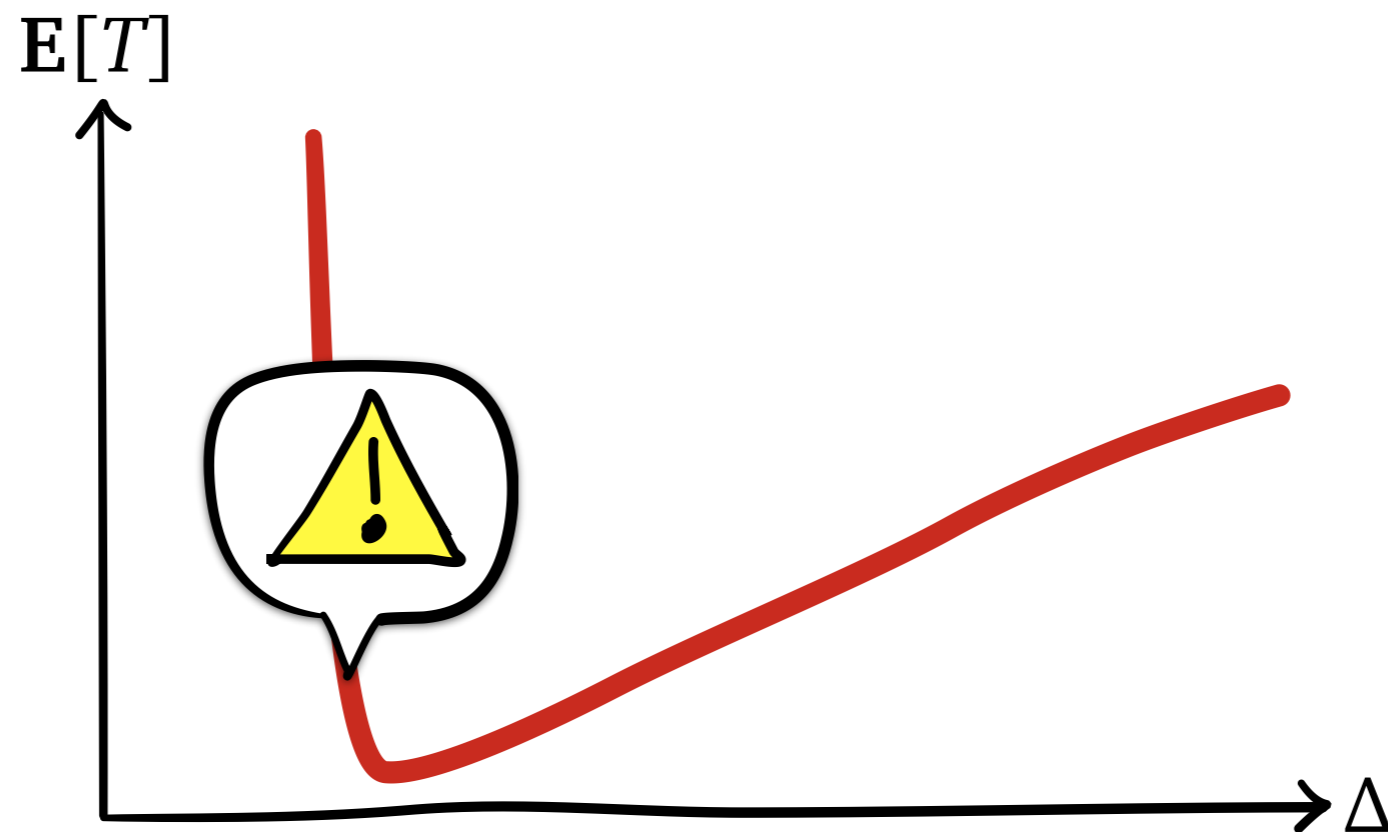


Checkpoint Frequency?

Suppose each checkpoint incurs an overhead

What is the optimal gap Δ between checkpoints?

- large Δ : less overhead
- small Δ : better scheduling

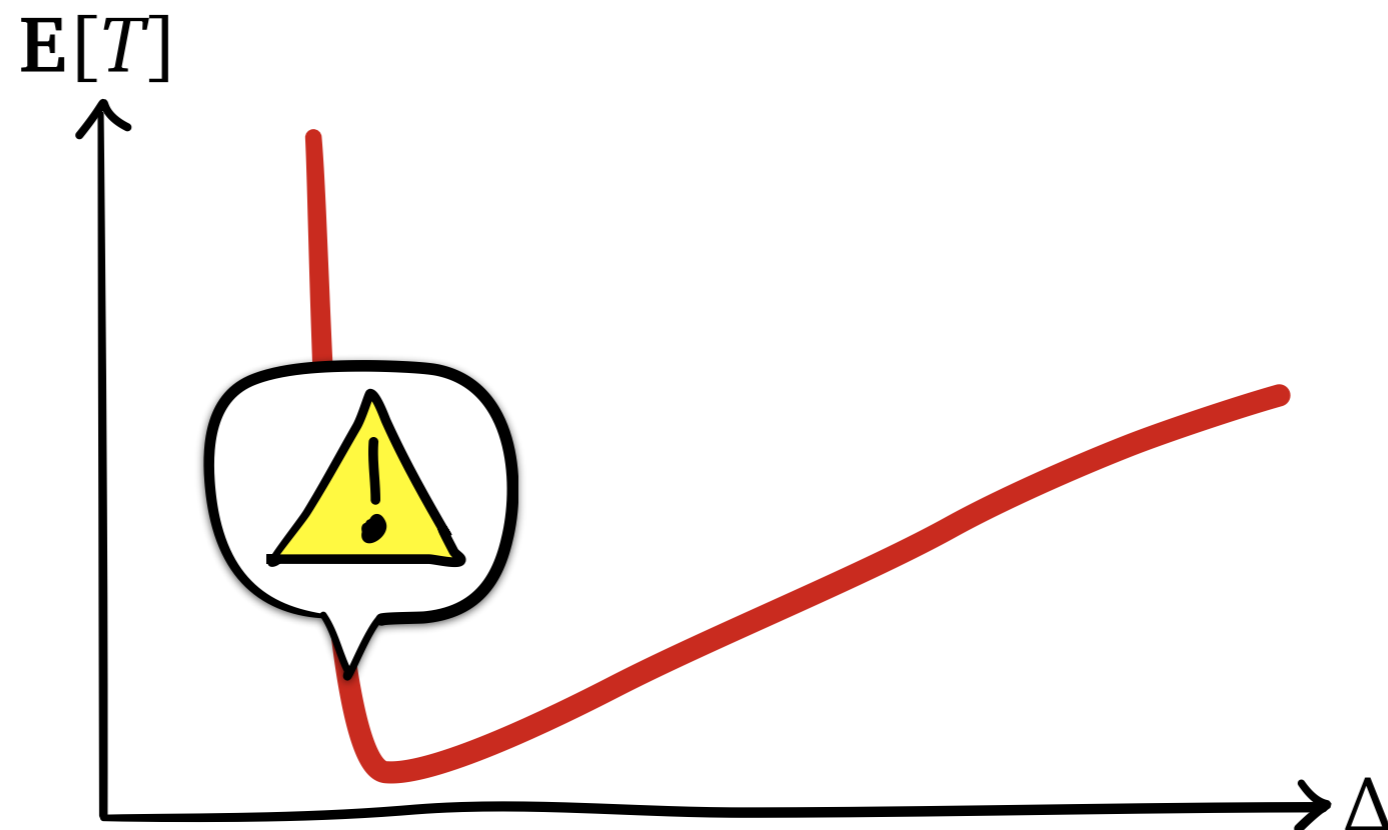


Checkpoint Frequency?

Suppose each checkpoint incurs an overhead

What is the optimal gap Δ between checkpoints?

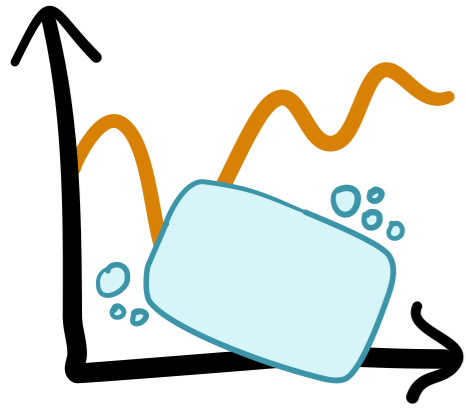
- large Δ : less overhead
- small Δ : better scheduling



[Scully & Harchol-Balter, in preparation]

Overview

New Tools



SOAP

analyzes a huge variety of scheduling heuristics

r-Work

provides a new, deeper understanding of Gittins

Goals

Multiple servers

Simple implementation preferred

Preemption restricted and/or costly

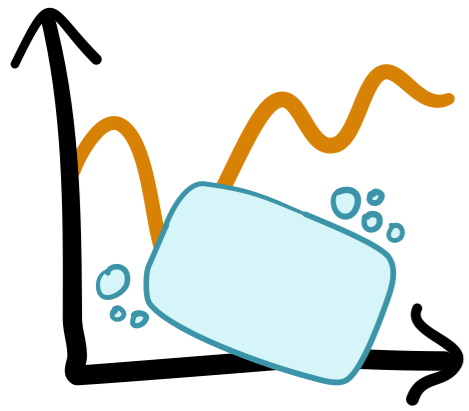


Limited number of priority levels

Want to optimize other response time metrics

Overview

New Tools



SOAP

analyzes a huge variety of scheduling heuristics

r-Work

provides a new, deeper understanding of Gittins

Goals

Multiple servers

Simple implementation preferred



Preemption restricted and/or costly

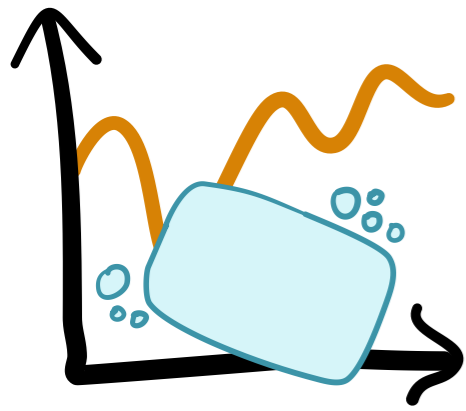


Limited number of priority levels

Want to optimize other response time metrics

Overview

New Tools



SOAP

analyzes a huge variety of scheduling heuristics

r-Work

provides a new, deeper understanding of Gittins

Goals

Multiple servers

Simple implementation preferred



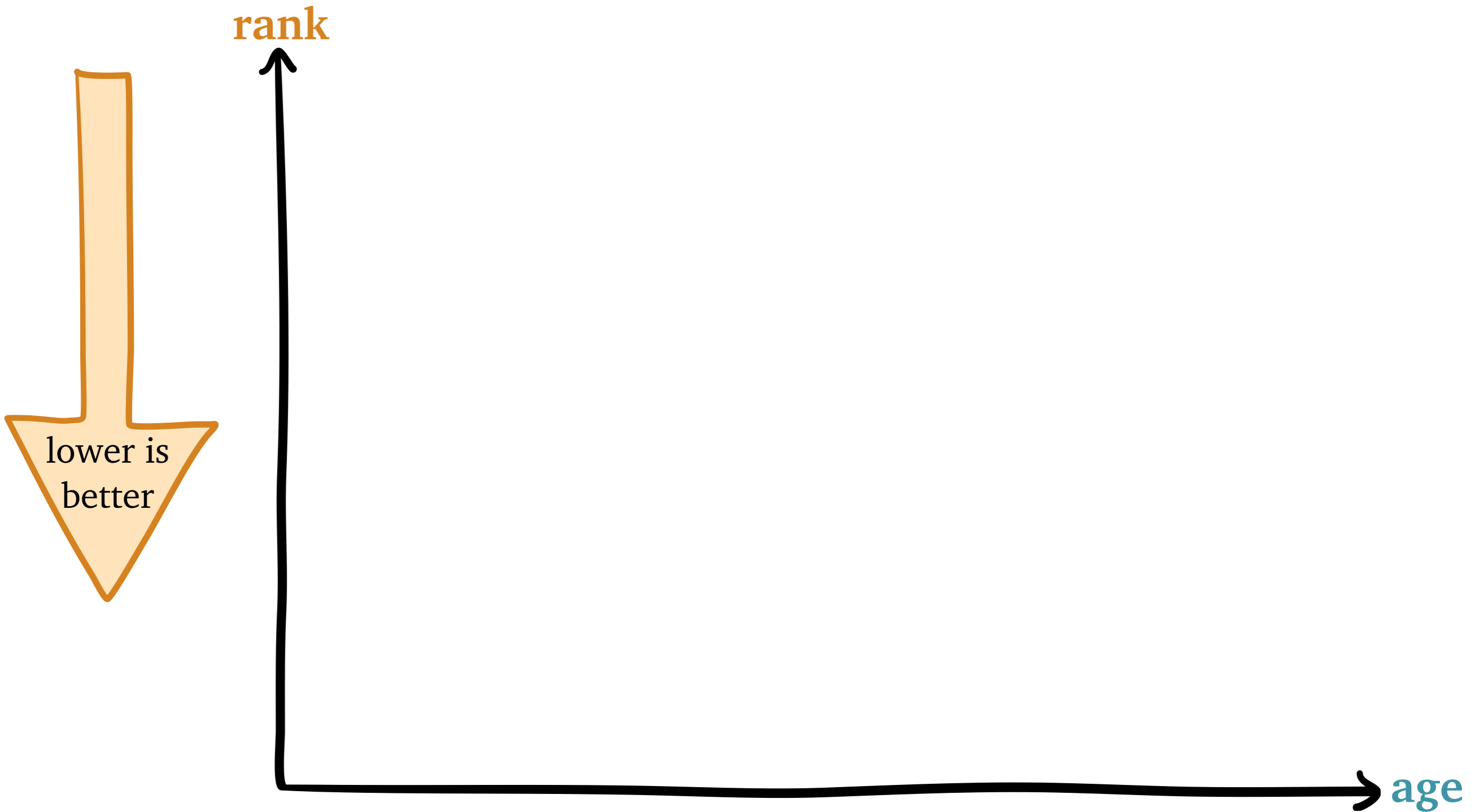
Preemption restricted and/or costly



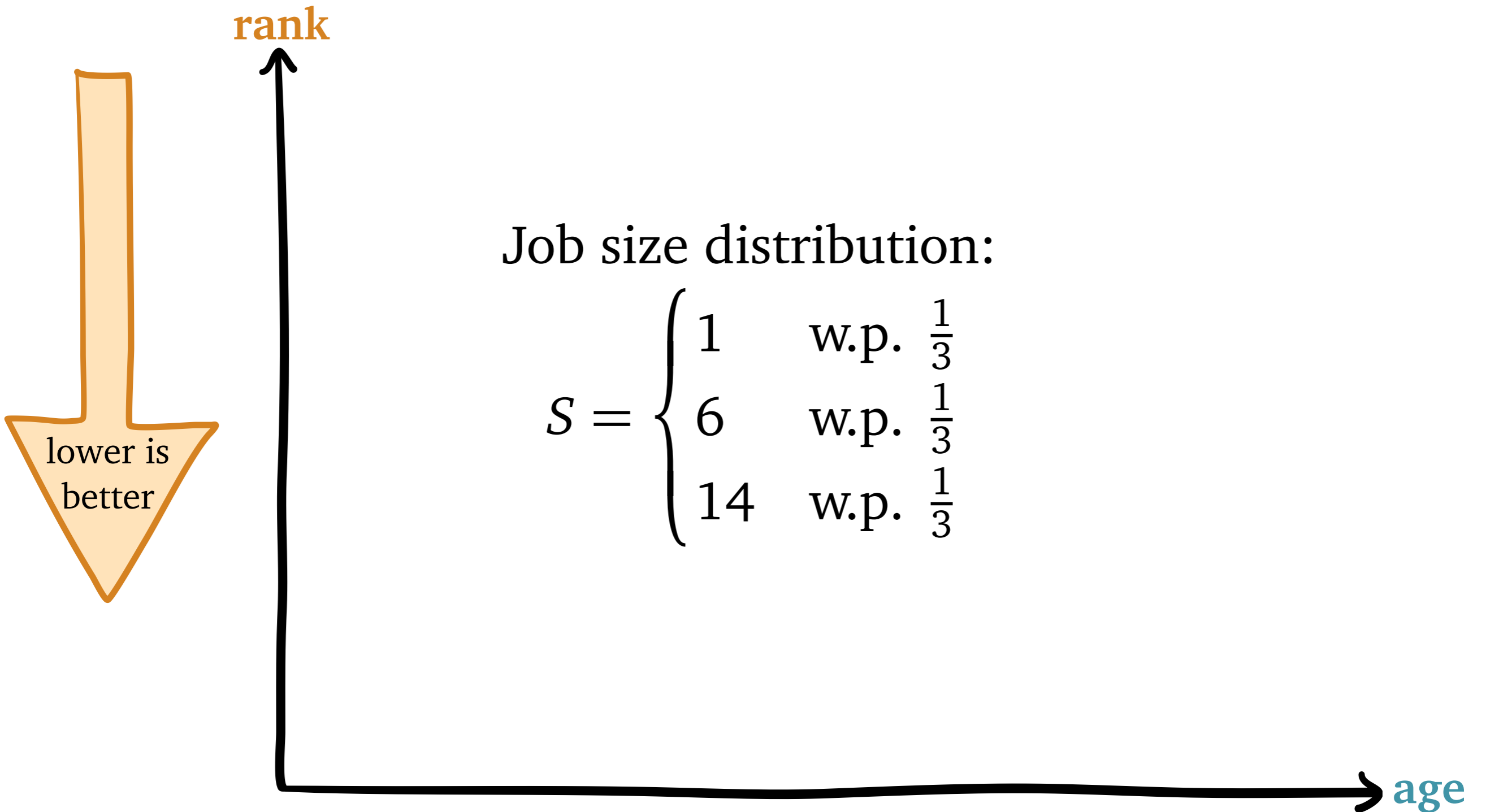
Limited number of priority levels

Want to optimize other response time metrics

Gittins: Optimal but Complex



Gittins: Optimal but Complex



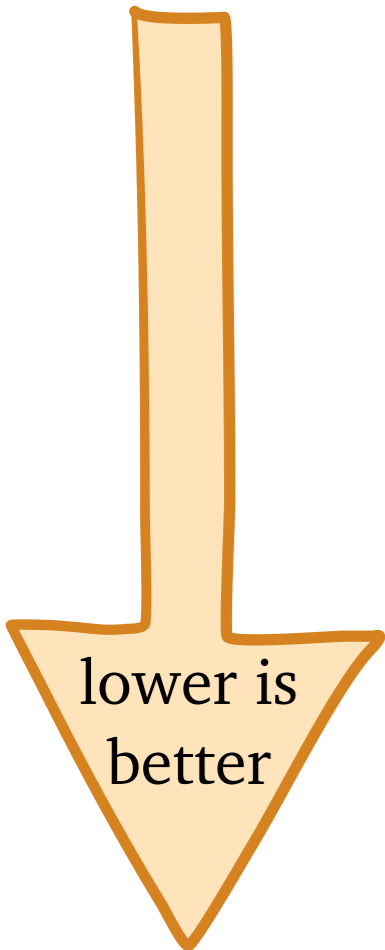
Gittins: Optimal but Complex

$$r_{\text{Gittins}}(a) = \inf_{b>a} \frac{\mathbf{E}[\min\{S - a, b\} \mid S > a]}{\mathbf{P}[S \leq b \mid S > a]}$$

Job size distribution:

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

rank

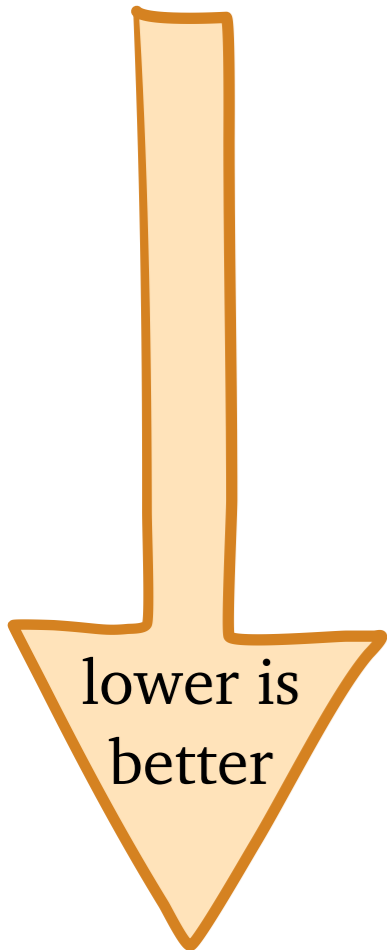


age

Gittins: Optimal but Complex

$$r_{\text{Gittins}}(a) = \inf_{b>a} \frac{\mathbf{E}[\min\{S - a, b\} \mid S > a]}{\mathbf{P}[S \leq b \mid S > a]}$$

rank



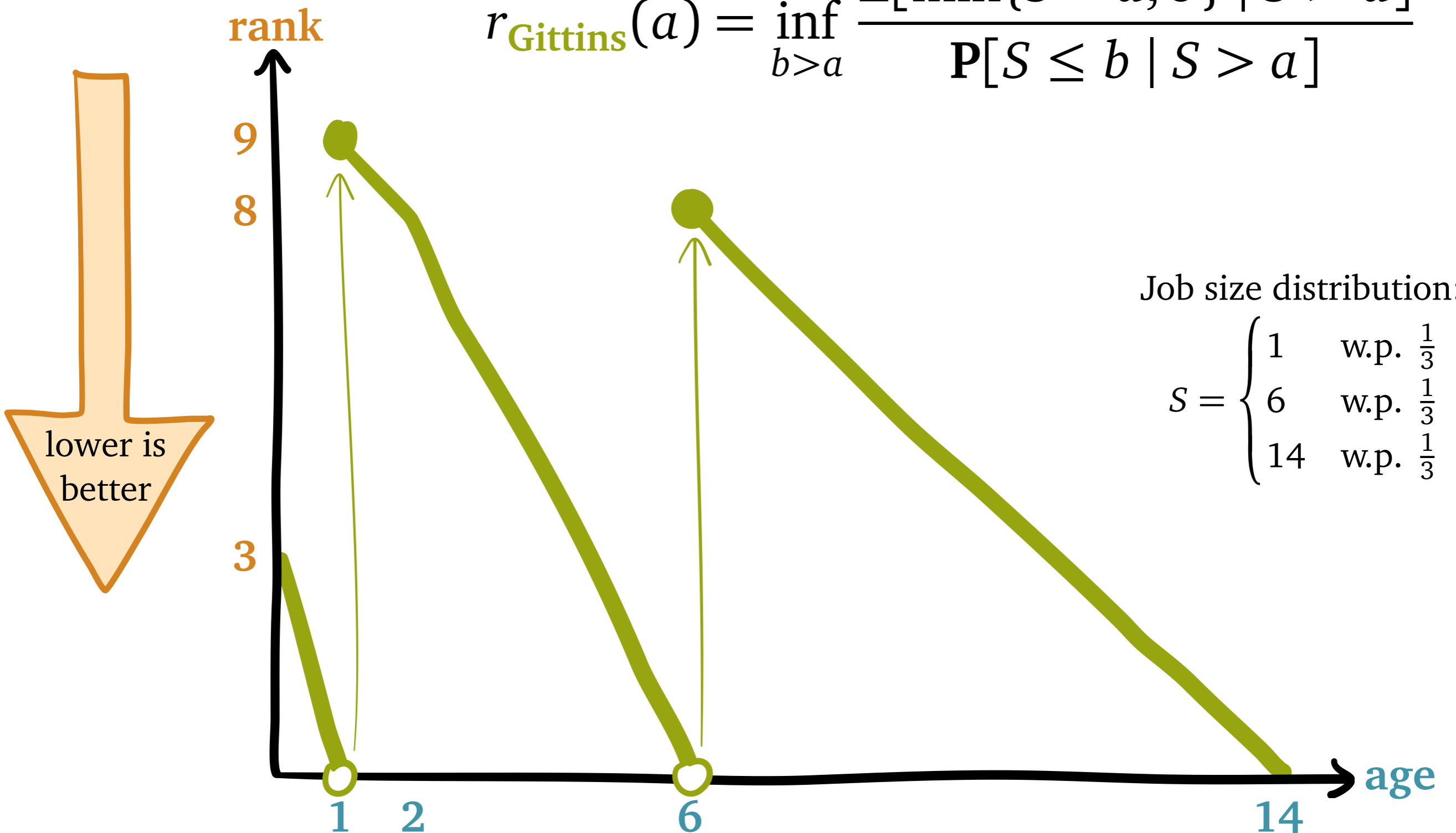
Job size distribution:

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

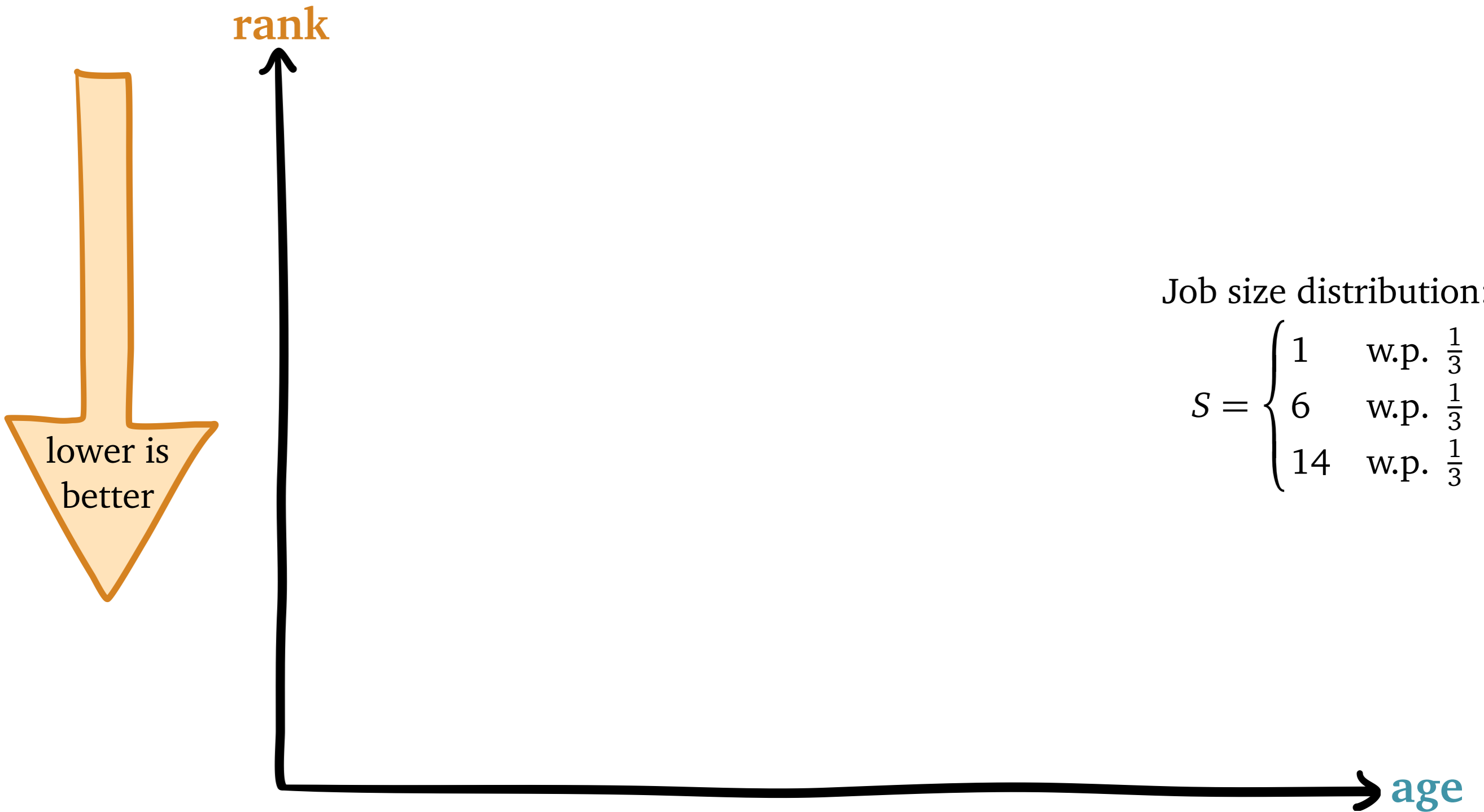
age

Gittins: Optimal but Complex

$$r_{\text{Gittins}}(a) = \inf_{b>a} \frac{\mathbb{E}[\min\{S - a, b\} \mid S > a]}{\mathbb{P}[S \leq b \mid S > a]}$$



SERPT: Simple Heuristic



SERPT: Simple Heuristic

shortest *expected* remaining
processing time

lower is
better

Job size distribution:

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

age

SERPT: Simple Heuristic

shortest *expected* remaining processing time

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

lower is better

Job size distribution:

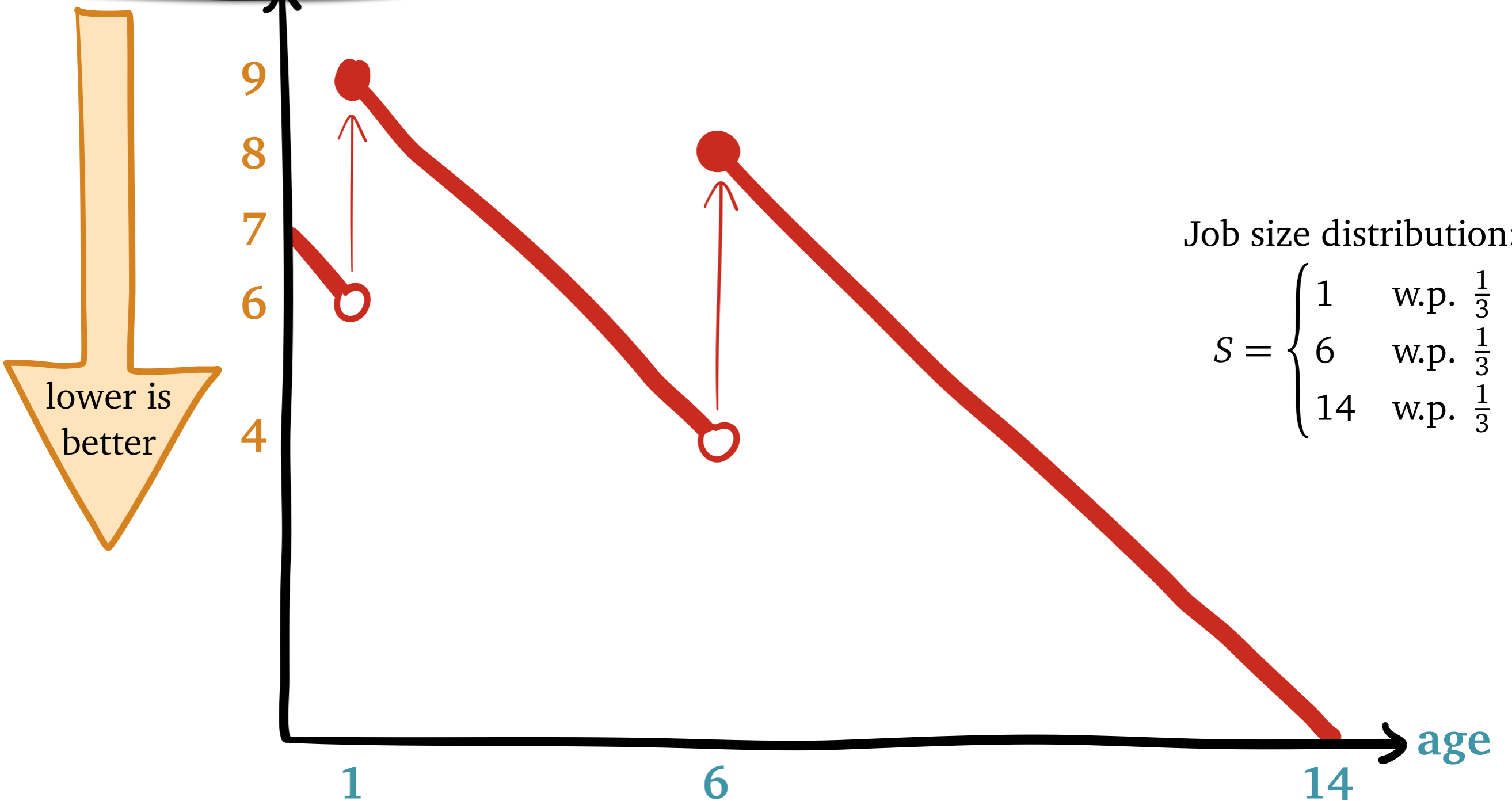
$$S = \begin{cases} 1 & \text{w.p. } \frac{1}{3} \\ 6 & \text{w.p. } \frac{1}{3} \\ 14 & \text{w.p. } \frac{1}{3} \end{cases}$$

age

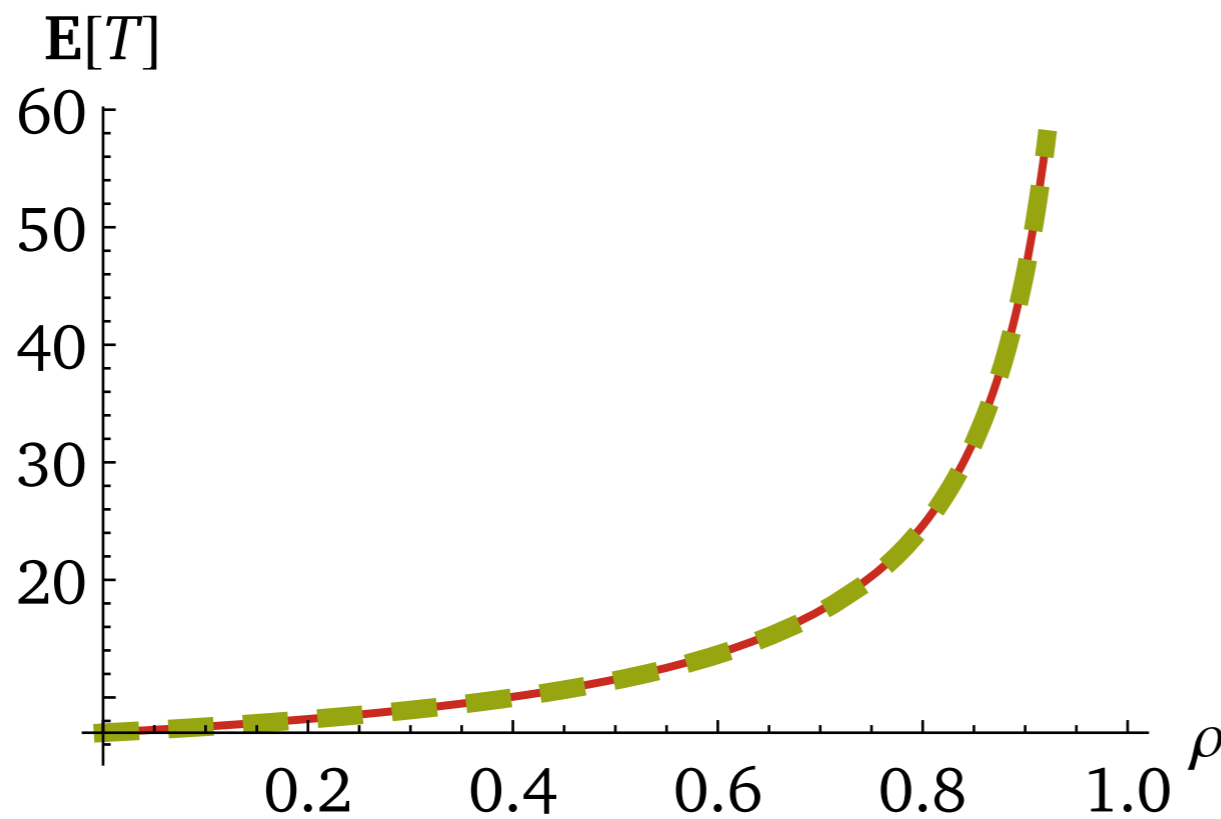
SERPT: Simple Heuristic

shortest *expected* remaining processing time

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



Can **SERPT** Replace **Gittins**?



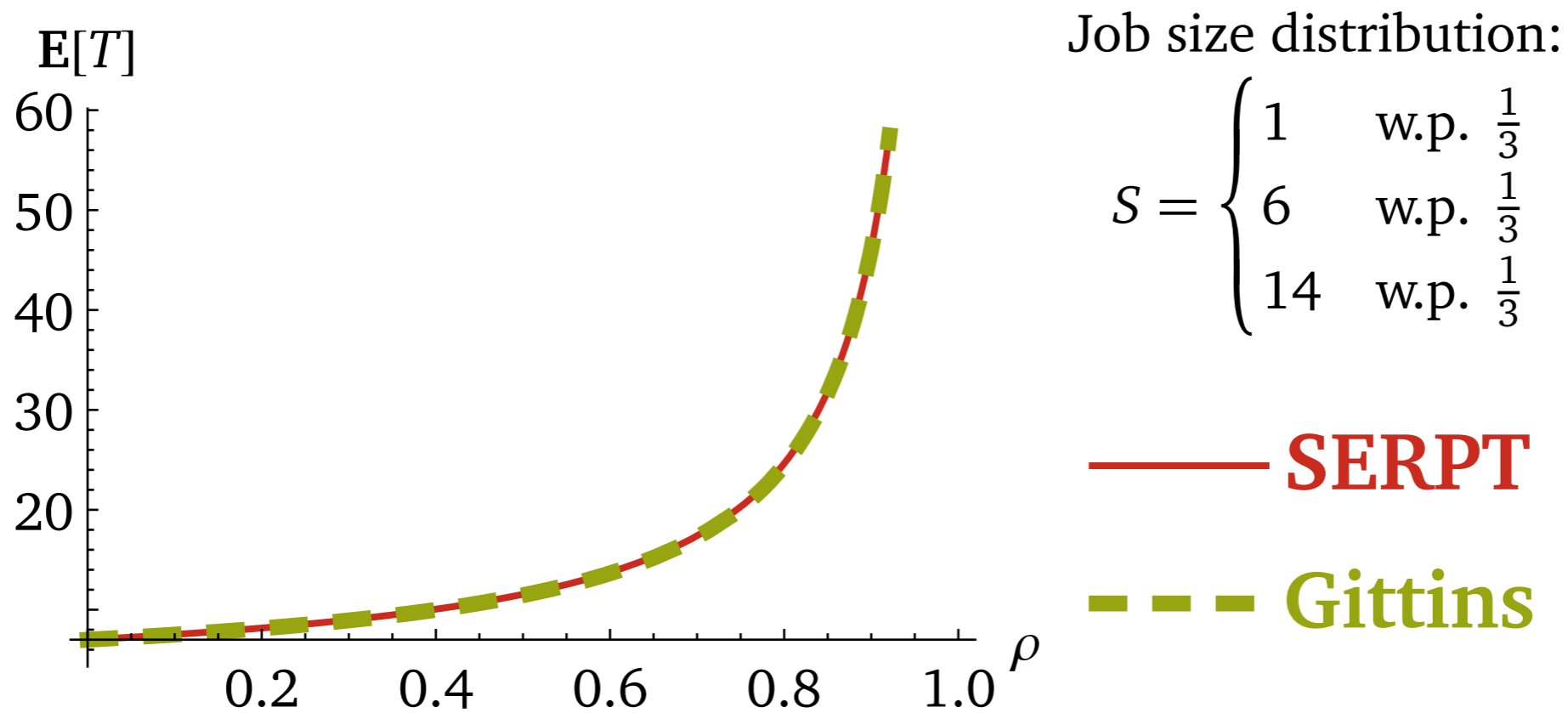
Job size distribution:

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

— **SERPT**

- - - **Gittins**

Can **SERPT** Replace **Gittins**?



- **Gittins** is hard to compute
- **SERPT** has no $E[T]$ guarantee



I wish for a policy with...

- *simple* definition like **SERPT**
- *provable* guarantee on $E[T]$ like **Gittins**



M-SERPT

I wish for a policy with...

- *simple* definition like **SERPT**
- *provable* guarantee on $E[T]$ like **Gittins**

Introducing M-SERPT

rank

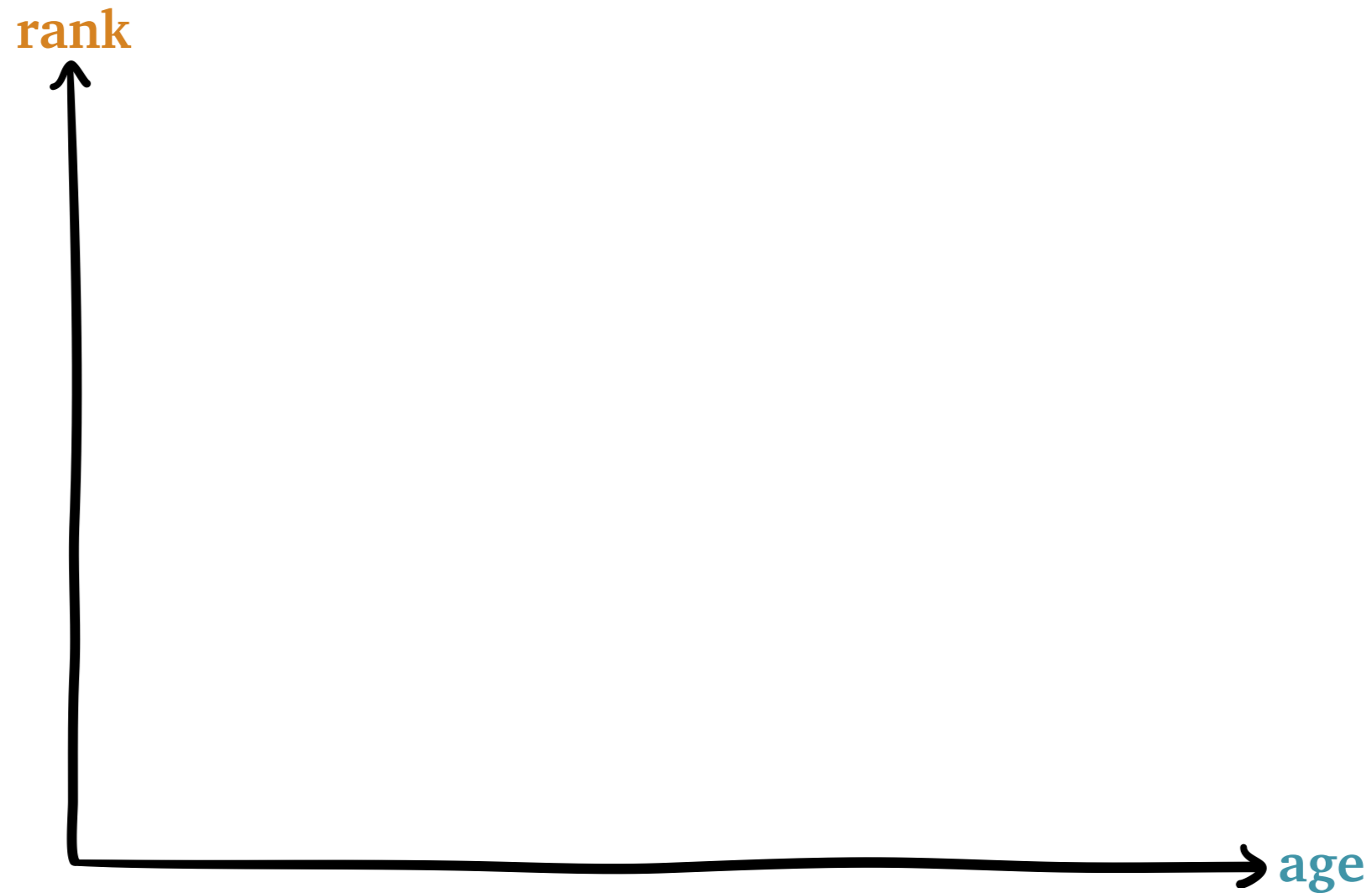


age



Introducing M-SERPT

$$r_{\text{M-SERPT}}(a) = \max_{0 \leq b \leq a} r_{\text{SERPT}}(b)$$



Introducing **M-SERPT**

monotonic

$$r_{\text{M-SERPT}}(a) = \max_{0 \leq b \leq a} r_{\text{SERPT}}(b)$$

rank

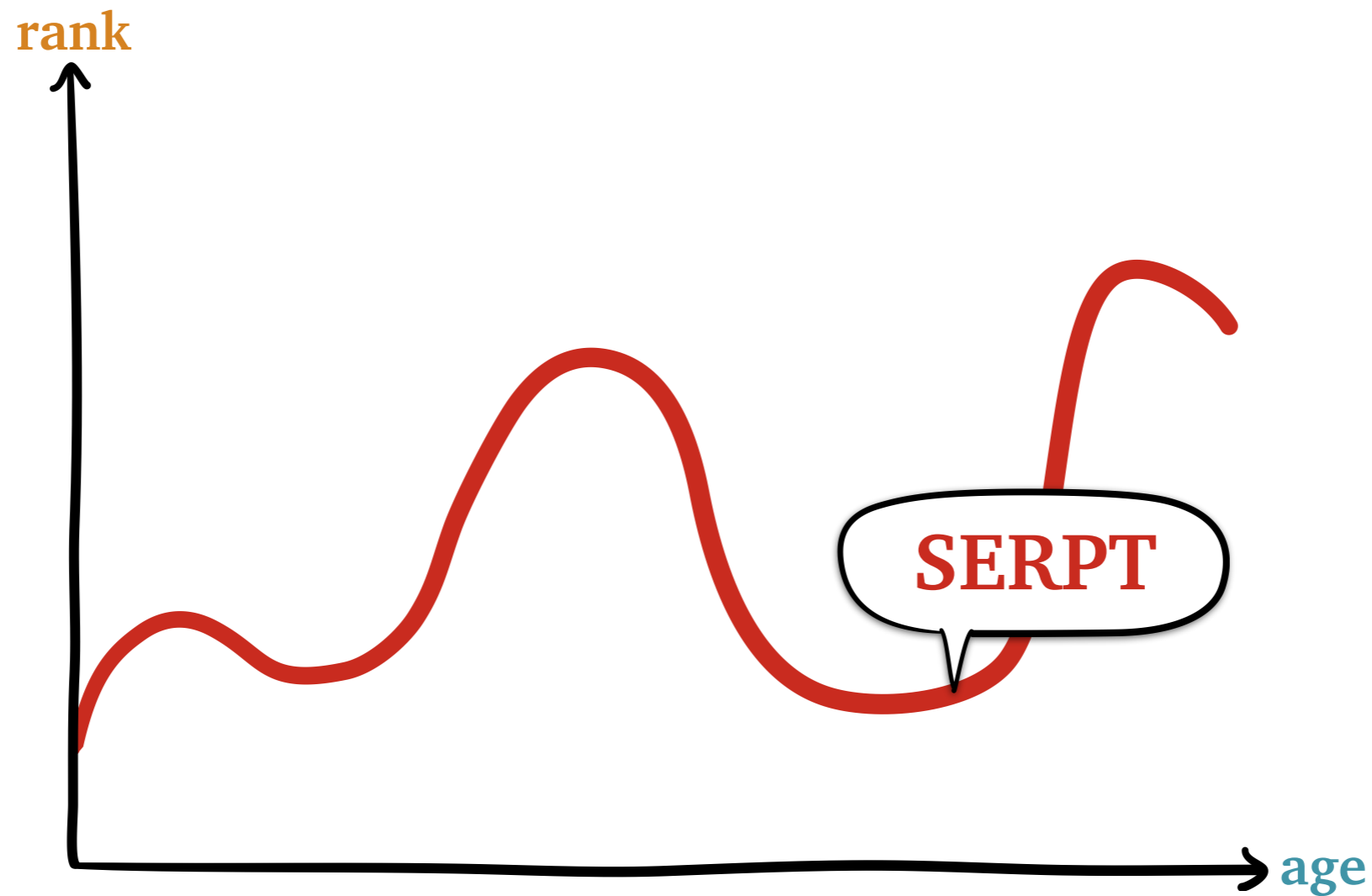


age

Introducing **M-SERPT**

monotonic

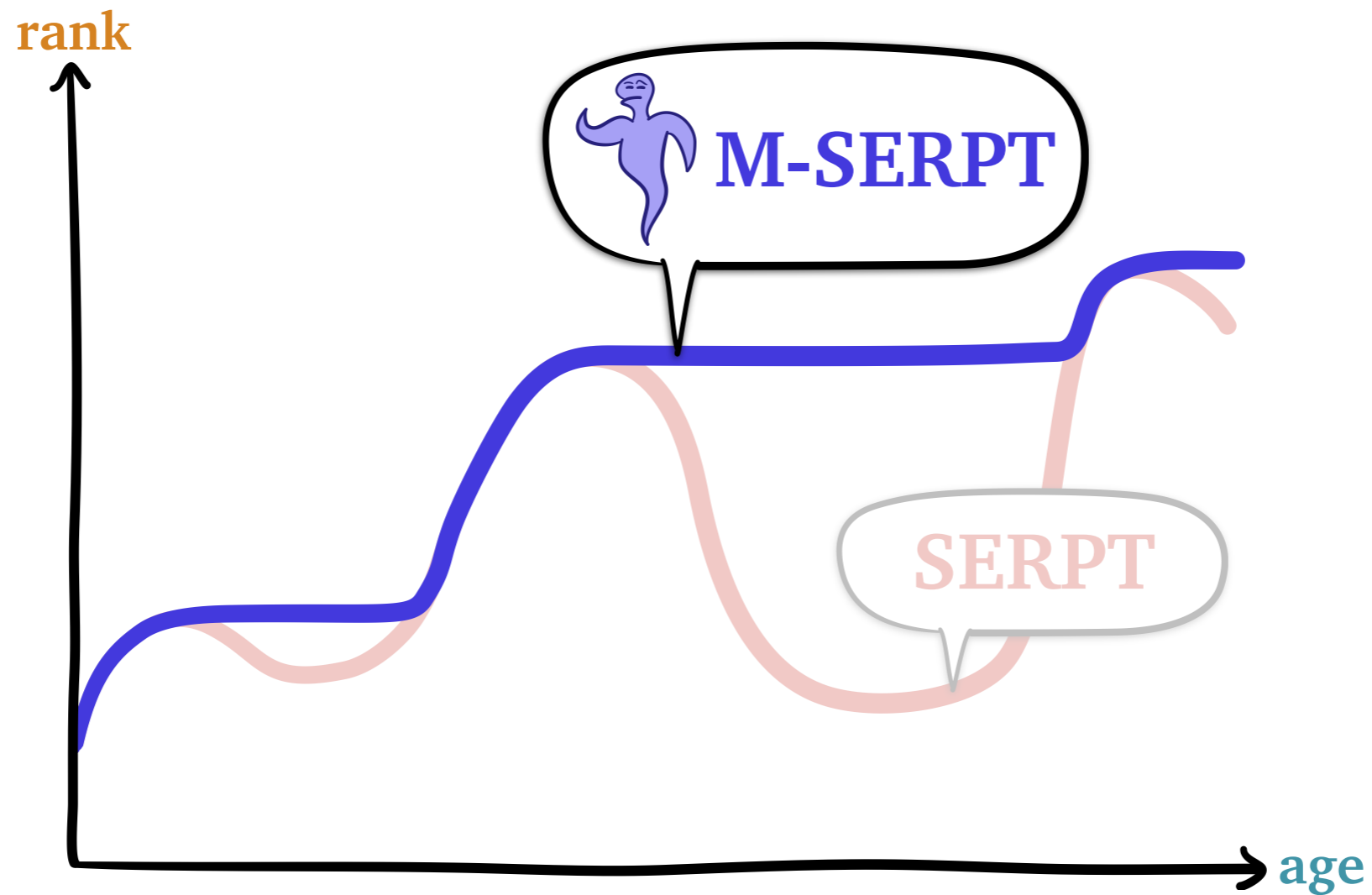
$$r_{\text{M-SERPT}}(a) = \max_{0 \leq b \leq a} r_{\text{SERPT}}(b)$$



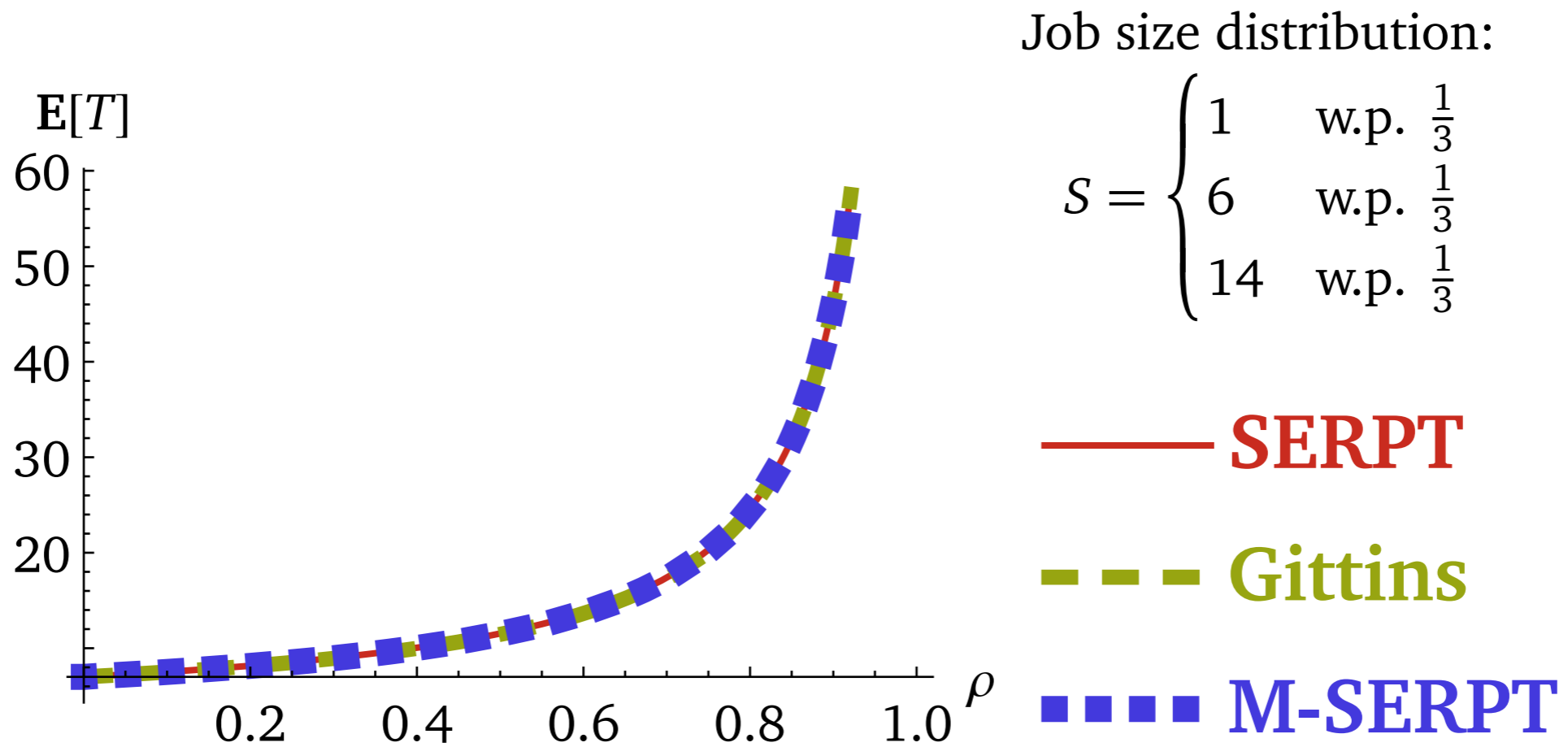
Introducing **M-SERPT**

monotonic

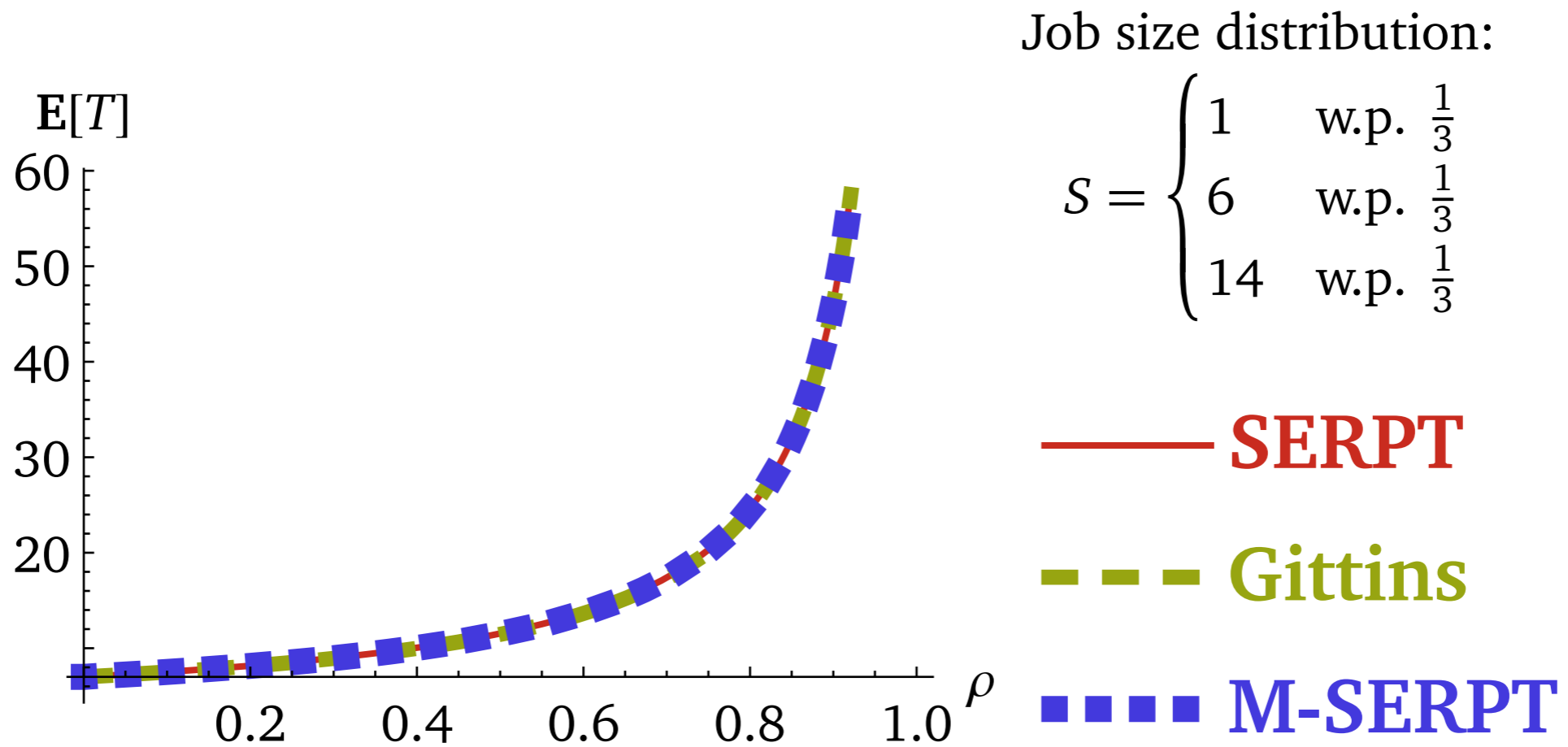
$$r_{\text{M-SERPT}}(a) = \max_{0 \leq b \leq a} r_{\text{SERPT}}(b)$$



M-SERPT Performance



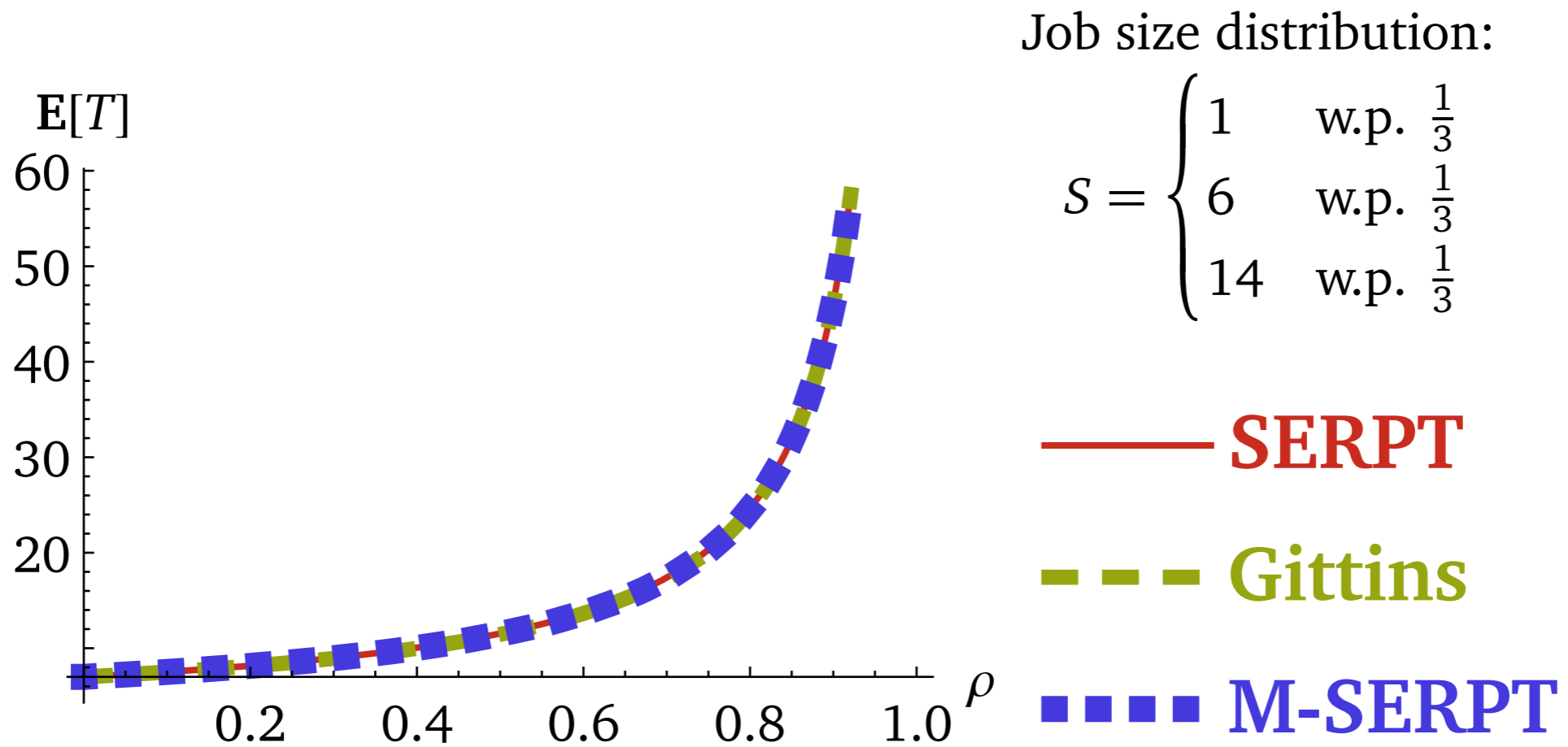
M-SERPT Performance



Theorem:

$$\frac{E[T_{\text{M-SERPT}}]}{E[T_{\text{Gittins}}]} \leq 5$$

M-SERPT Performance

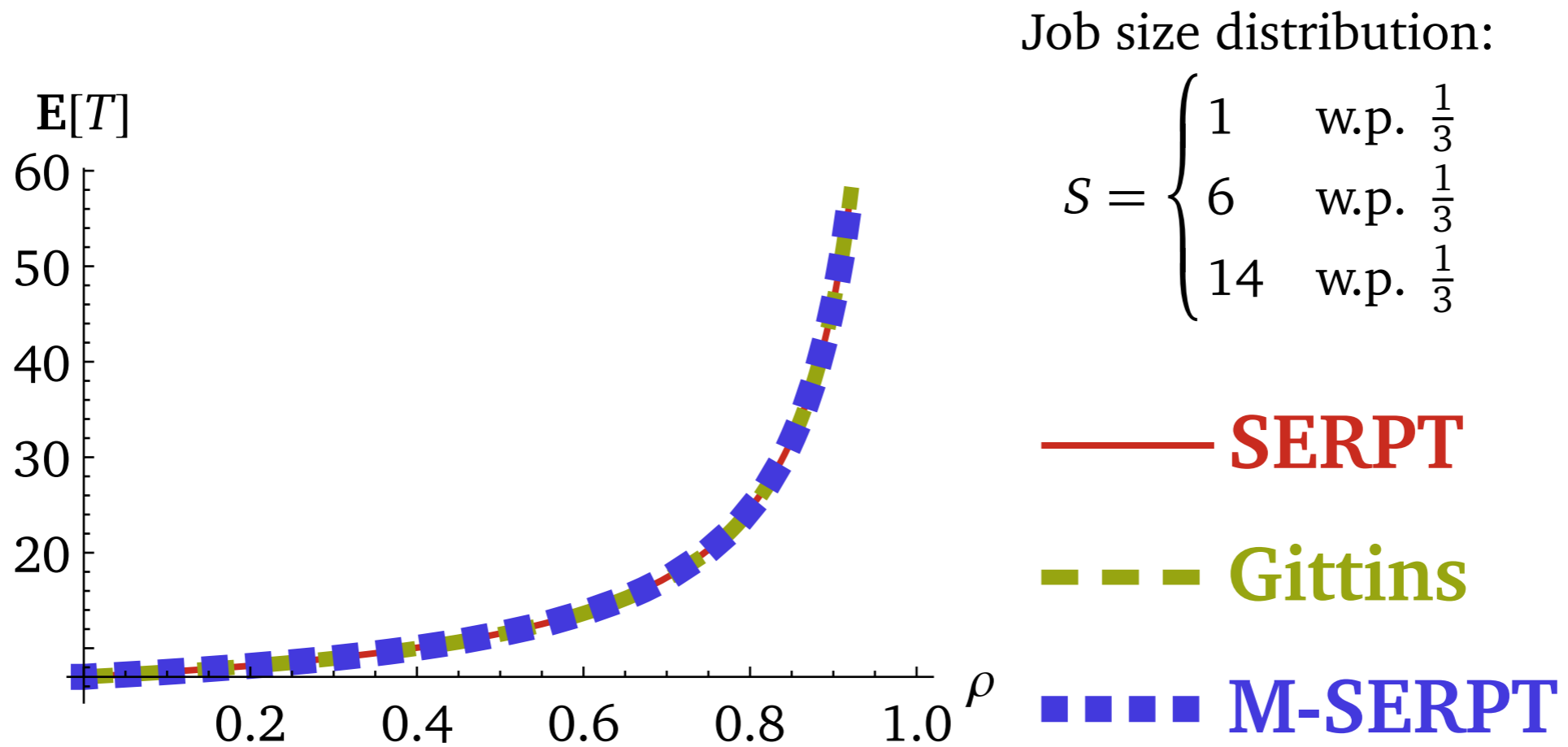


Theorem:

$$\frac{\mathbf{E}[T_{\text{M-SERPT}}]}{\mathbf{E}[T_{\text{Gittins}}]} \leq 5$$

smaller at low load
 first constant ratio

M-SERPT Performance



Theorem:

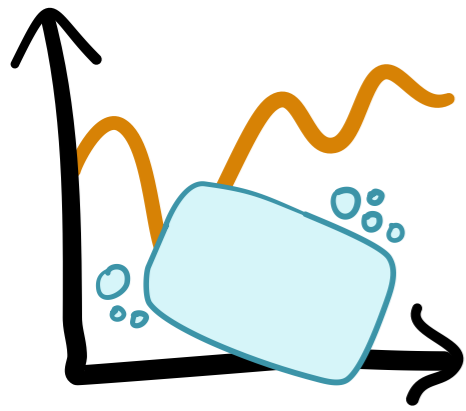
$$\frac{\mathbf{E}[T_{\mathbf{M-SERPT}}]}{\mathbf{E}[T_{\mathbf{Gittins}}]} \leq 5$$

smaller at low load
 first constant ratio

[Scully, Harchol-Balter, & Scheller-Wolf, SIGMETRICS 2020]

Overview

New Tools



SOAP

analyzes a huge variety of scheduling heuristics

r-Work

provides a new, deeper understanding of Gittins

Goals

Multiple servers

Simple implementation preferred



Preemption restricted and/or costly

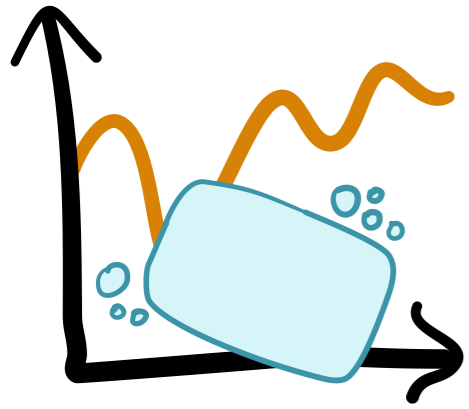


Limited number of priority levels

Want to optimize other response time metrics

Overview

New Tools



SOAP

analyzes a huge variety of scheduling heuristics

r-Work

provides a new, deeper understanding of Gittins

Goals

Multiple servers



Simple implementation preferred



Preemption restricted and/or costly

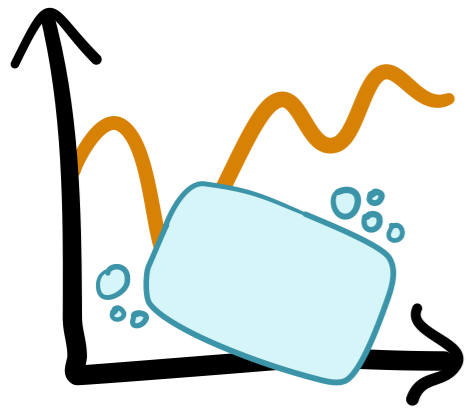


Limited number of priority levels

Want to optimize other response time metrics

Overview

New Tools



SOAP

analyzes a huge variety of scheduling heuristics

r-Work

provides a new, deeper understanding of Gittins

Goals

Multiple servers



Simple implementation preferred



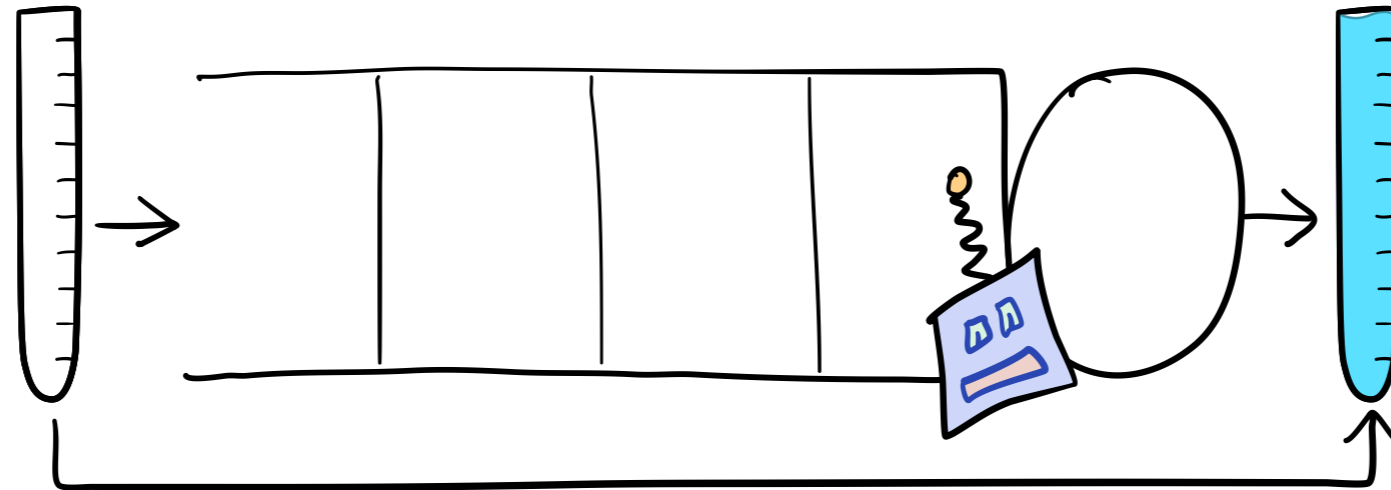
Preemption restricted and/or costly




Limited number of priority levels

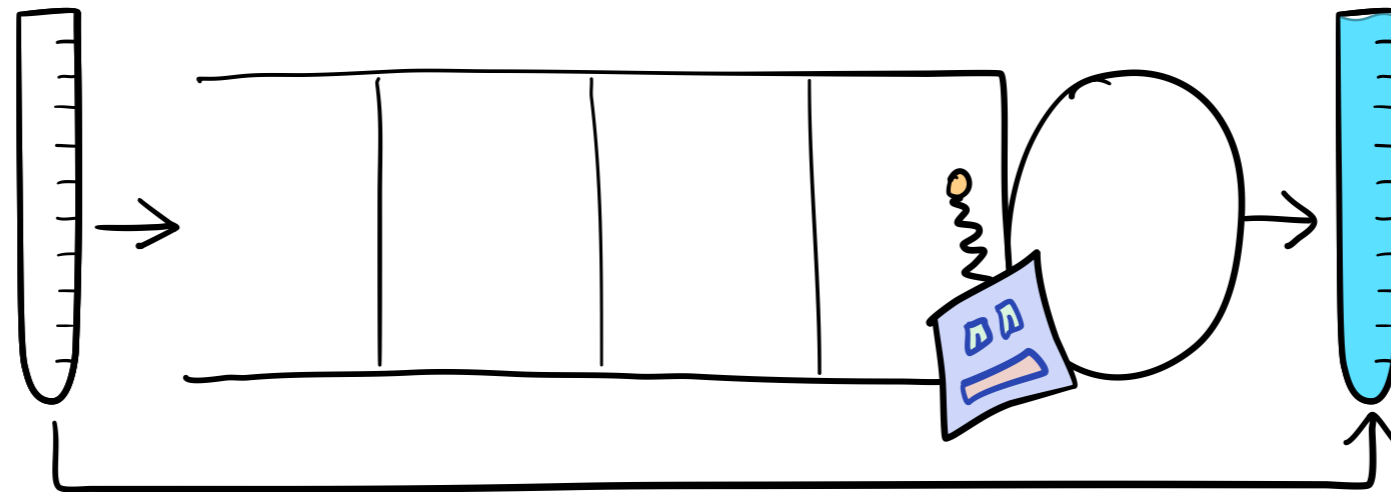
Want to optimize other response time metrics


Response Time Metrics



 = T = *response time*

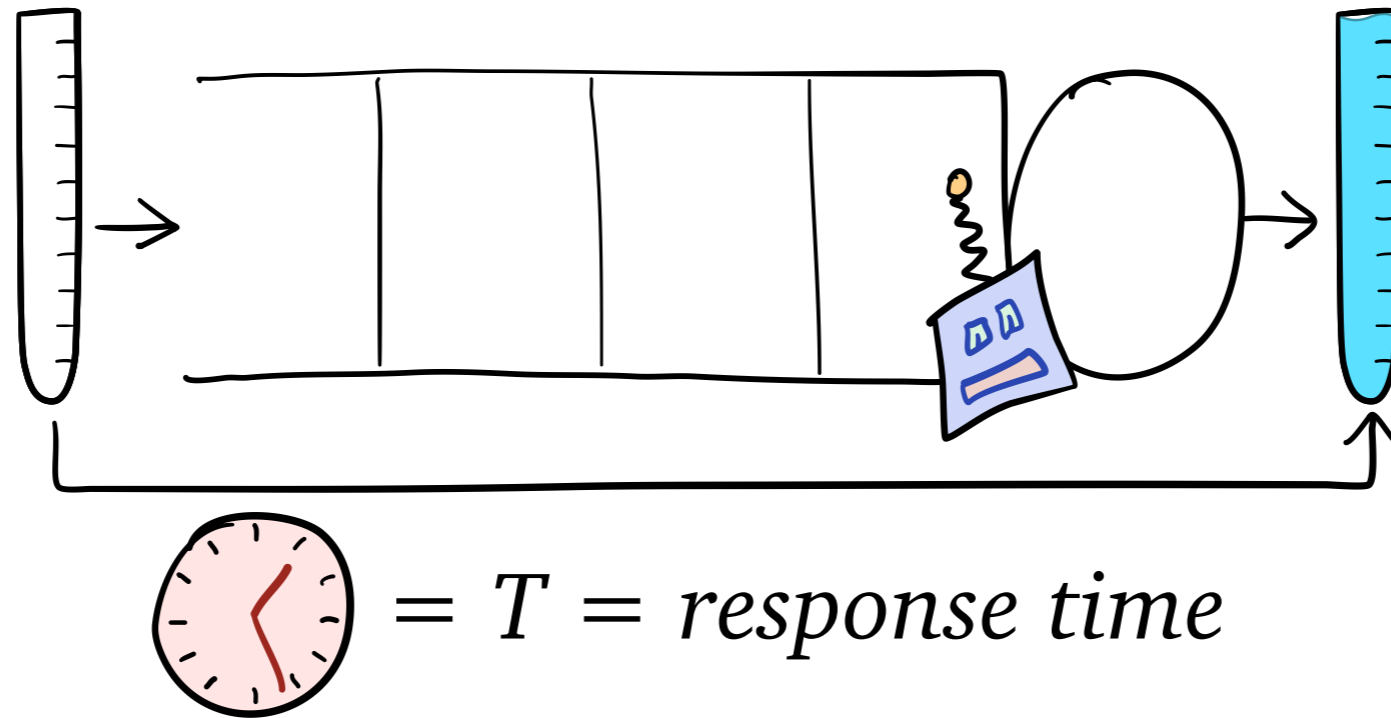
Response Time Metrics



 = T = *response time*

Goal: schedule to minimize two metrics

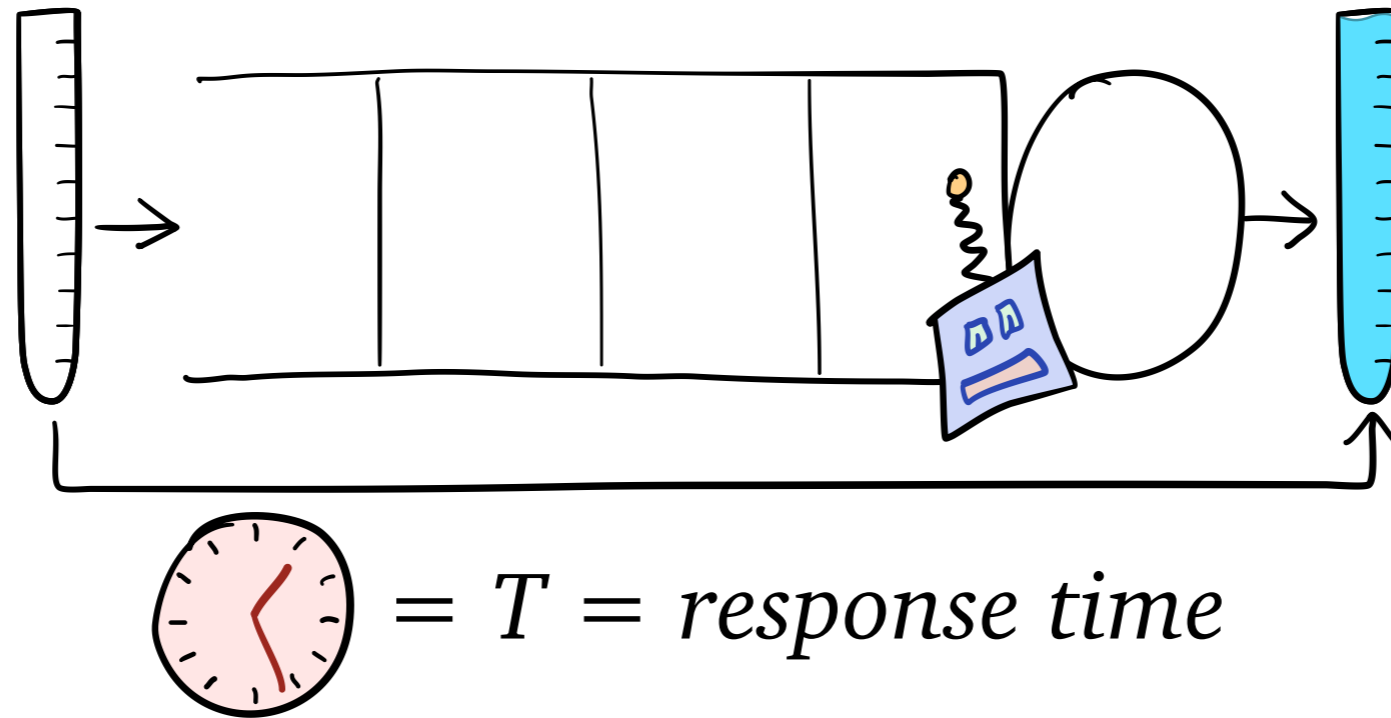
Response Time Metrics



Goal: schedule to minimize two metrics

- *mean* response time $E[T]$

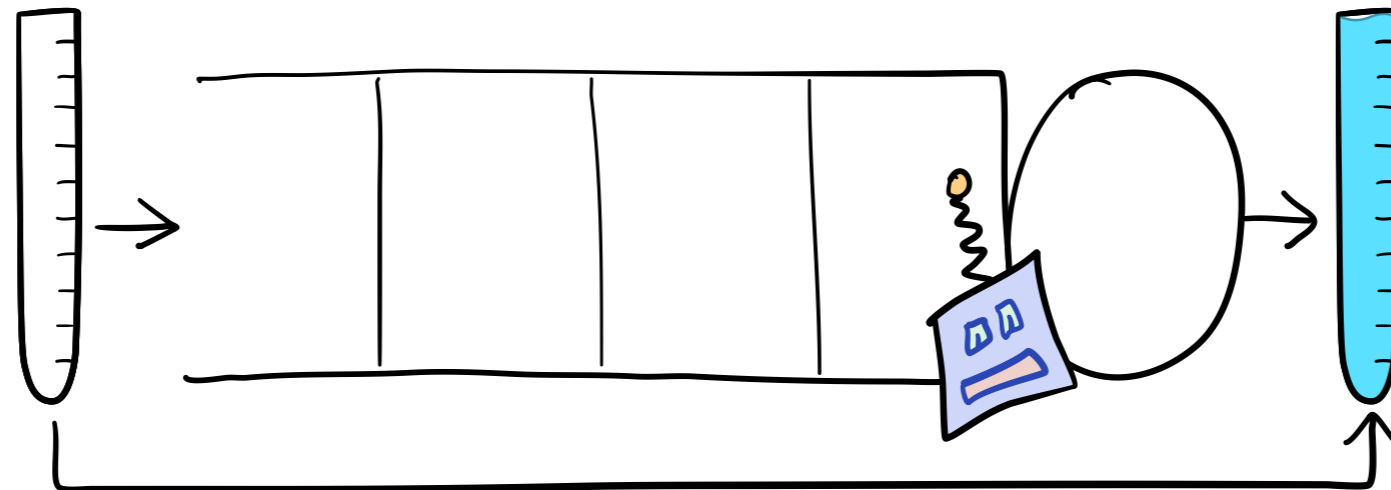
Response Time Metrics




Goal: schedule to minimize two metrics

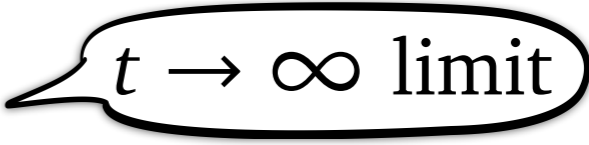
- *mean* response time $\mathbf{E}[T]$
- *tail* of response time $\mathbf{P}[T > t]$

Response Time Metrics

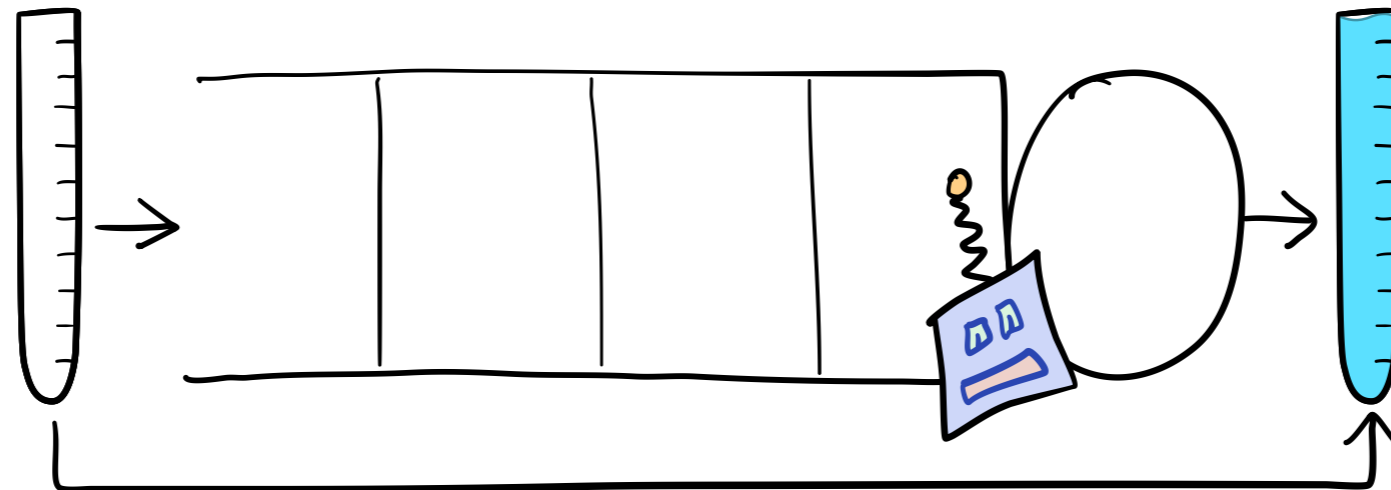



 = T = *response time*

Goal: schedule to minimize two metrics

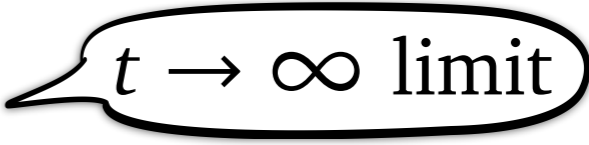
- *mean* response time $E[T]$
- *tail* of response time $P[T > t]$ 

Response Time Metrics



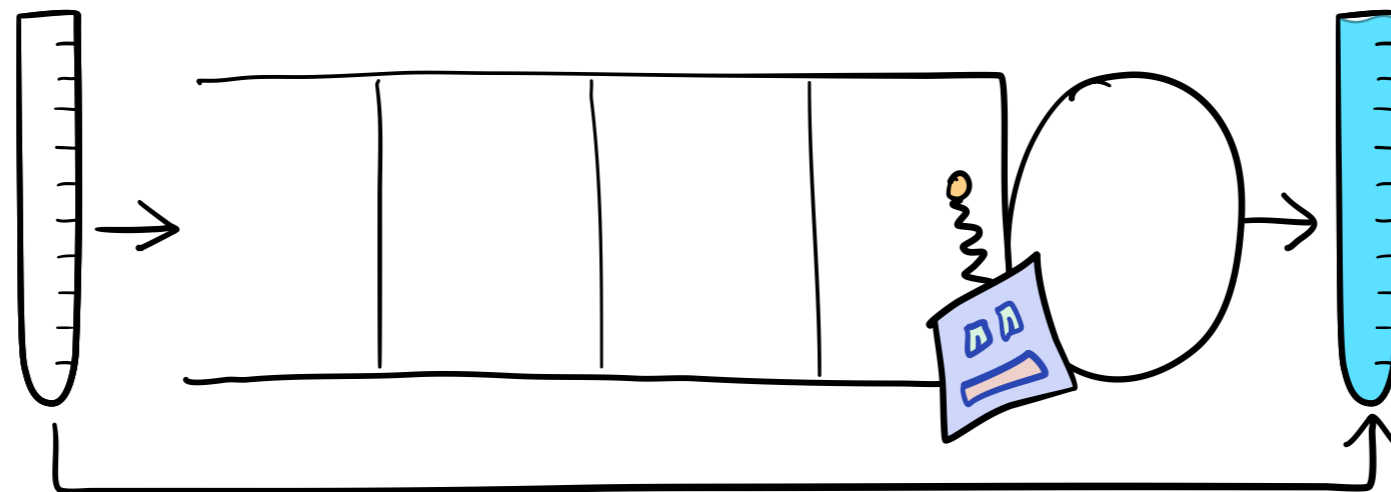
 $= T = \text{response time}$


Goal: schedule to minimize two metrics

- *mean* response time $\mathbf{E}[T]$
- *tail* of response time $\mathbf{P}[T > t]$ 

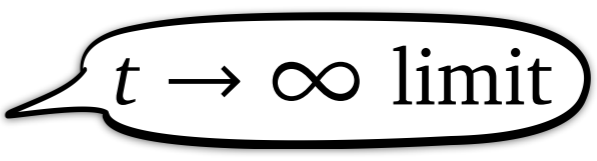
Setting: *heavy-tailed* job size distribution S

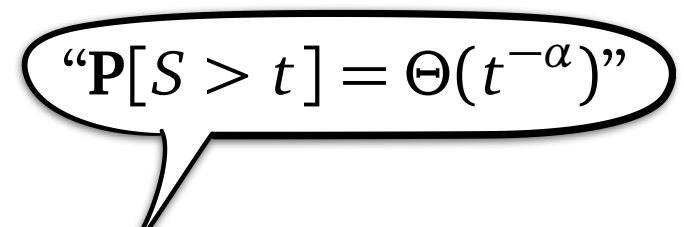
Response Time Metrics



 $= T = \text{response time}$

Goal: schedule to minimize two metrics

- *mean* response time $\mathbf{E}[T]$
- *tail* of response time $\mathbf{P}[T > t]$  $t \rightarrow \infty$ limit

 “ $\mathbf{P}[S > t] = \Theta(t^{-\alpha})$ ”

Setting: *heavy-tailed* job size distribution S

Scheduling with Heavy Tails

Scheduling with Heavy Tails

$t \rightarrow \infty$ limit


Policy 

Mean $E[T]$

Tail $P[T > t]$

Scheduling with Heavy Tails

$t \rightarrow \infty$ limit

<i>Policy</i> 	<i>Mean</i> $E[T]$	<i>Tail</i> $P[T > t]$
FCFS	bad	worst

Scheduling with Heavy Tails

$t \rightarrow \infty$ limit

Policy 

Mean $E[T]$

Tail $P[T > t]$

FCFS


bad

worst

$P[T > t] = \Theta(t) \cdot P[S > t]$


Scheduling with Heavy Tails

$t \rightarrow \infty$ limit

<i>Policy</i> 	<i>Mean</i> $E[T]$	<i>Tail</i> $P[T > t]$
FCFS	bad	worst
SRPT	best (but needs sizes)	best

Scheduling with Heavy Tails


$t \rightarrow \infty$ limit

<i>Policy</i> 	<i>Mean</i> $E[T]$	<i>Tail</i> $P[T > t]$
FCFS	bad	worst
SRPT	best (but needs sizes)	best

$$P[T > t] = \Theta(1) \cdot P[S > t]$$


Scheduling with Heavy Tails

$t \rightarrow \infty$ limit

<i>Policy</i> 	<i>Mean</i> $E[T]$	<i>Tail</i> $P[T > t]$
FCFS	bad	worst
SRPT	best (but needs sizes)	best
FB	good	best


Scheduling with Heavy Tails

$t \rightarrow \infty$ limit

<i>Policy</i> 	<i>Mean</i> $E[T]$	<i>Tail</i> $P[T > t]$
FCFS	bad	worst
SRPT	best (but needs sizes)	best
FB	good	best
Gittins	best	
M-SERPT	5-approx.	


Scheduling with Heavy Tails

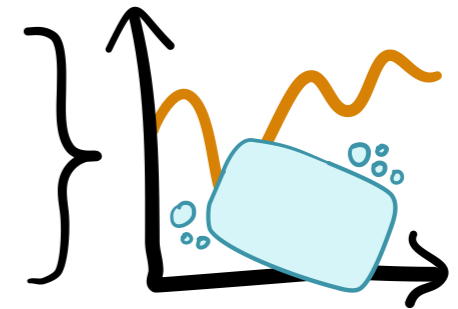
$t \rightarrow \infty$ limit

<i>Policy</i> 	<i>Mean</i> $E[T]$	<i>Tail</i> $P[T > t]$
FCFS	bad	worst
SRPT	best (but needs sizes)	best
FB	good	best
Gittins	best	???
M-SERPT	5-approx.	???

Scheduling with Heavy Tails


$t \rightarrow \infty$ limit

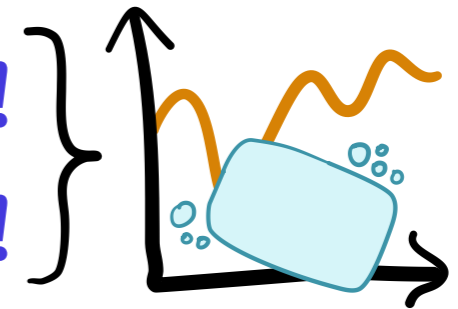
<i>Policy</i> 	<i>Mean</i> $E[T]$	<i>Tail</i> $P[T > t]$
FCFS	bad	worst
SRPT	best (but needs sizes)	best
FB	good	best
Gittins	best	???
M-SERPT	5-approx.	???



Scheduling with Heavy Tails


$t \rightarrow \infty$ limit

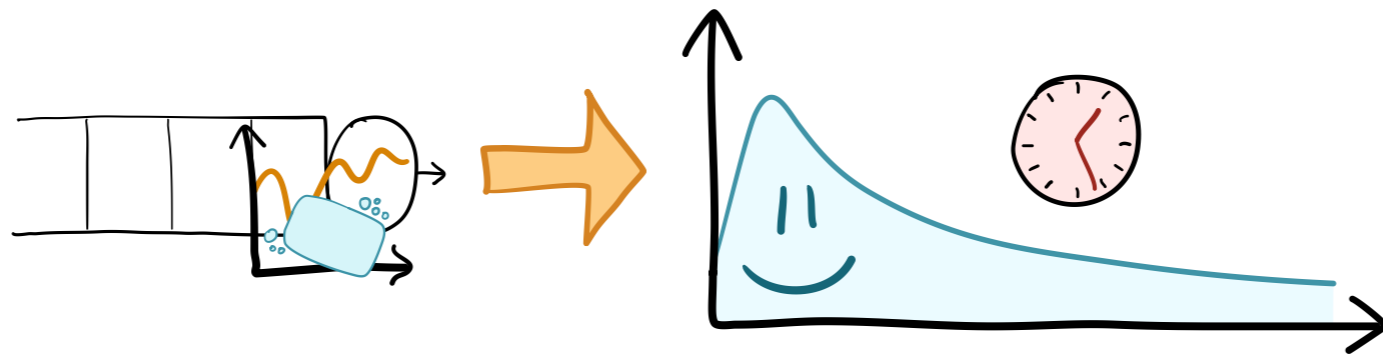
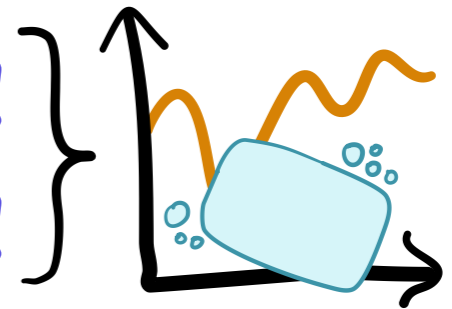
<i>Policy</i> 	<i>Mean</i> $E[T]$	<i>Tail</i> $P[T > t]$
FCFS	bad	worst
SRPT	best (but needs sizes)	best
FB	good	best
Gittins	best	best!
M-SERPT	5-approx.	best!



Scheduling with Heavy Tails

$t \rightarrow \infty$ limit


Policy 	Mean $E[T]$	Tail $P[T > t]$
FCFS	bad	worst
SRPT	best (but needs sizes)	best
FB	good	best
Gittins	best	best!
M-SERPT	5-approx.	best!

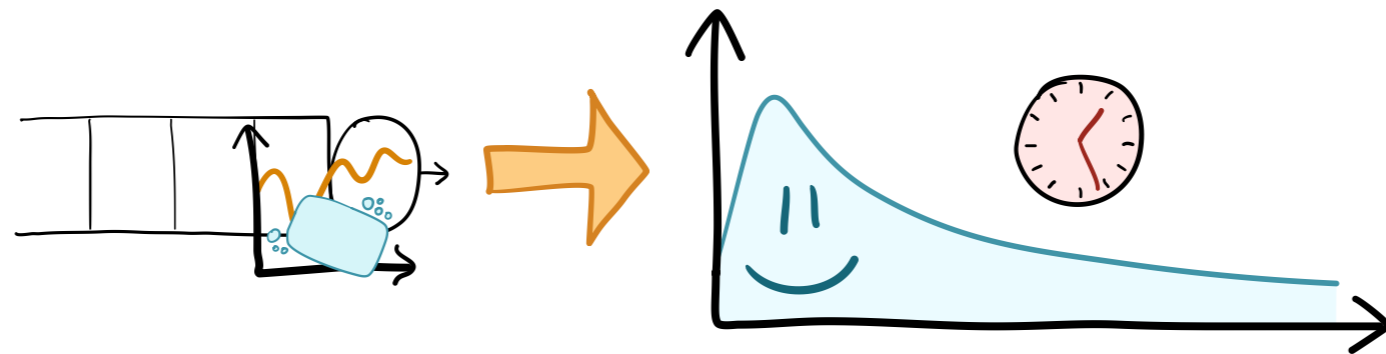
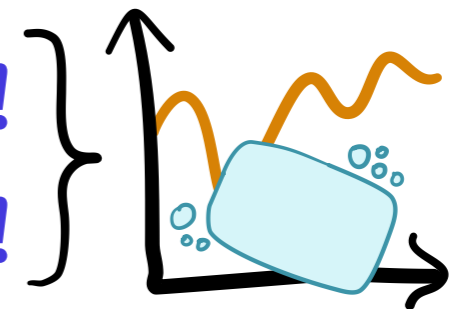


New sufficient condition for **rank** function to be tail-optimal

Scheduling with Heavy Tails

$t \rightarrow \infty$ limit

Policy 	Mean $E[T]$	Tail $P[T > t]$
FCFS	bad	worst
SRPT	best (but needs sizes)	best
FB	good	best
Gittins	best	best!
M-SERPT	5-approx.	best!

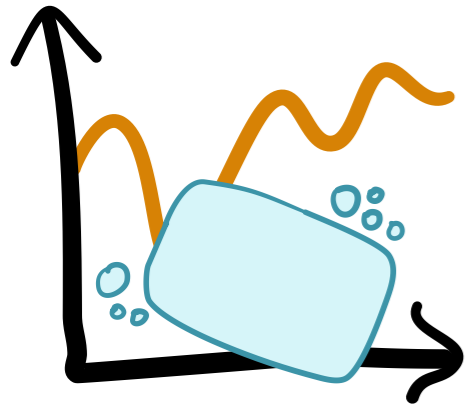


New sufficient condition for **rank** function to be tail-optimal

[Scully, van Kreveld, Boxma, Dorsman, & Wierman, SIGMETRICS 2020]

Overview

New Tools



SOAP

analyzes a huge variety of scheduling heuristics

r-Work

provides a new, deeper understanding of Gittins

Goals

Multiple servers



Simple implementation preferred



Preemption restricted and/or costly

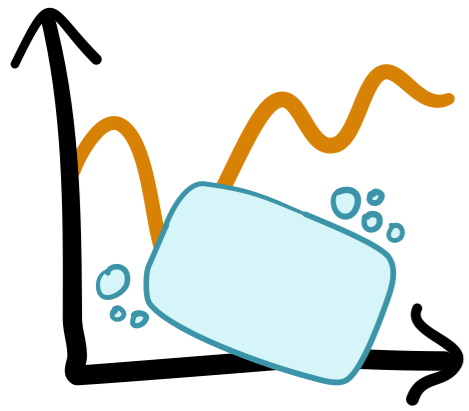


Limited number of priority levels

Want to optimize other response time metrics

Overview

New Tools



SOAP

analyzes a huge variety of scheduling heuristics

r-Work

provides a new, deeper understanding of Gittins

Goals

Multiple servers



Simple implementation preferred



Preemption restricted and/or costly



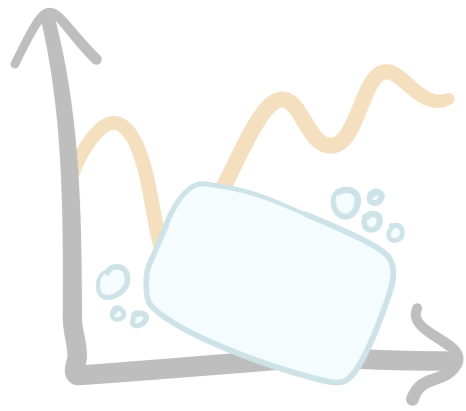
Limited number of priority levels



Want to optimize other response time metrics

Overview

New Tools



SOAP

analyzes a huge variety of scheduling heuristics

r-Work

provides a new, deeper understanding of Gittins

Goals

Multiple servers

✓ *Simple implementation preferred*

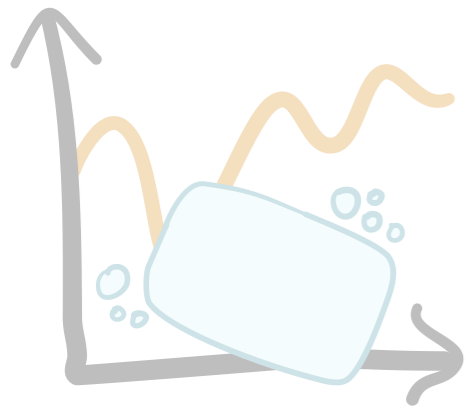
✓ *Preemption restricted and/or costly*

✓ *Limited number of priority levels*

✓ *Want to optimize other response time metrics*

Overview

New Tools



SOAP

analyzes a huge variety of scheduling heuristics

r-Work

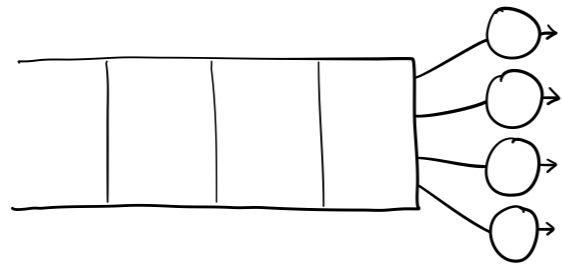
provides a new, deeper understanding of Gittins

Goals

Multiple servers

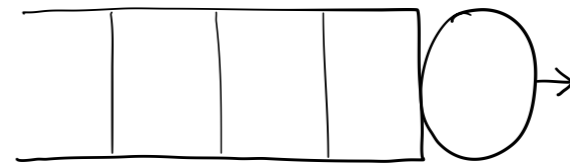
- ✓ *Simple implementation preferred*
- ✓ *Preemption restricted and/or costly*
- ✓ *Limited number of priority levels*
- ✓ *Want to optimize other response time metrics*

Gittins in the $M/G/k$



mean response
time in $M/G/k$

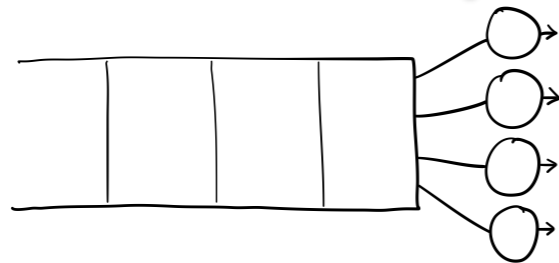
— — — —
bound?



mean response
time in $M/G/1$

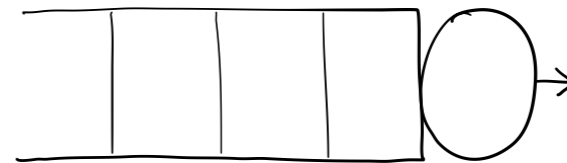
Gittins in the $M/G/k$

k servers, each
speed $1/k$



mean response
time in $M/G/k$

— — — — —
bound?

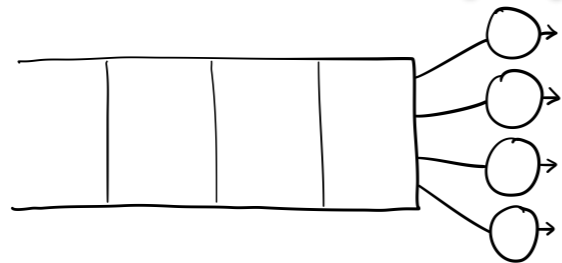


mean response
time in $M/G/1$

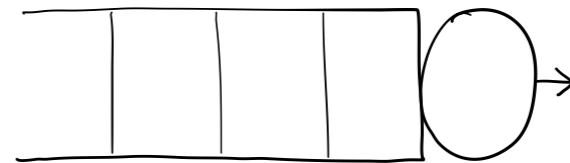
Gittins in the M/G/k

k servers, each speed $1/k$

Gittins serves k jobs of k lowest **rank**s



mean response time in M/G/ k



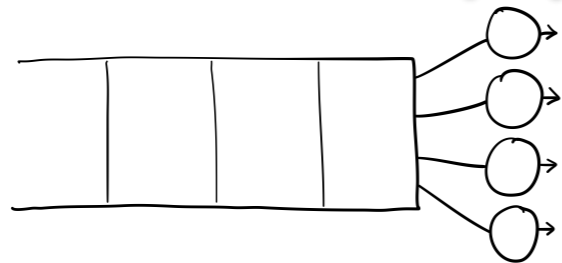
mean response time in M/G/1

bound?

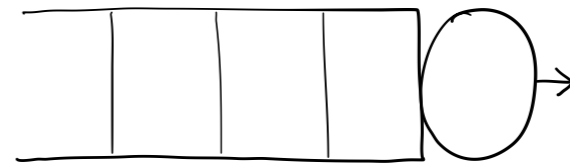
Gittins in the M/G/k

k servers, each speed $1/k$

Gittins serves k jobs of k lowest **rank**s



mean response time in M/G/ k



mean response time in M/G/1

bound?

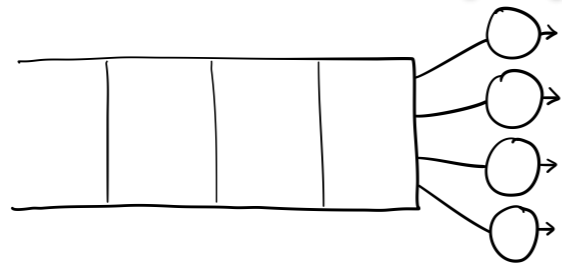
mean **r -work** in M/G/ k

mean **r -work** in M/G/1

Gittins in the M/G/k

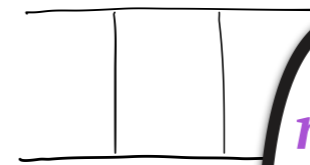
k servers, each speed $1/k$

Gittins serves k jobs of k lowest **rank**s



mean response time in M/G/ k

bound?

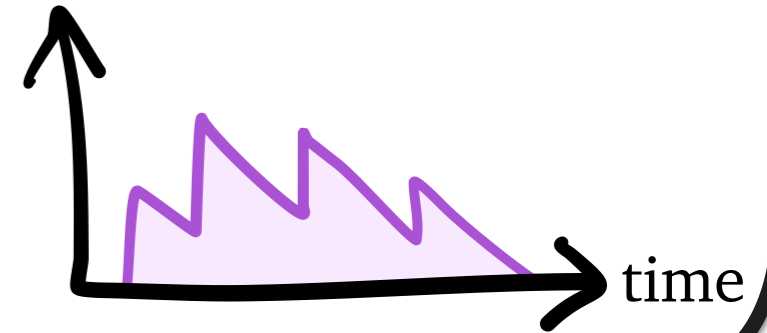


mean response time in M/G/1

mean **r -work** in M/G/ k

mean **r -work** in M/G/1

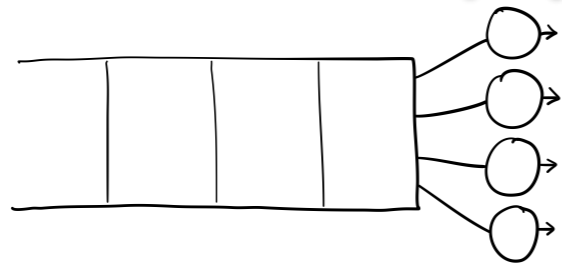
r -work



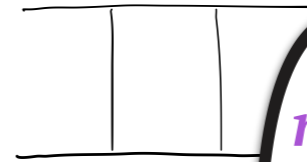
Gittins in the M/G/k

k servers, each speed $1/k$

Gittins serves k jobs of k lowest **rank**s



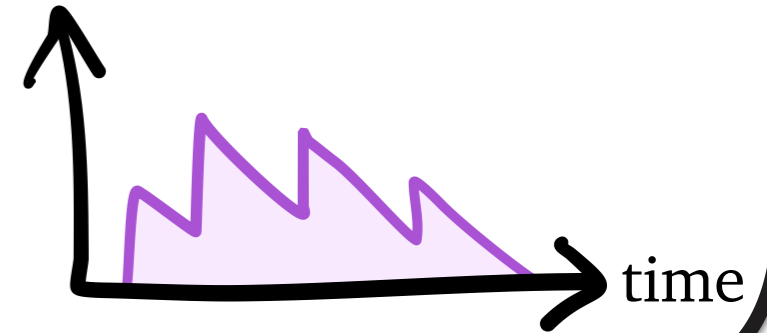
mean response time in M/G/k



mean response time in M/G/1

bound?

r -work



mean r -work in M/G/k

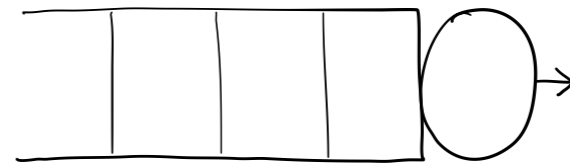
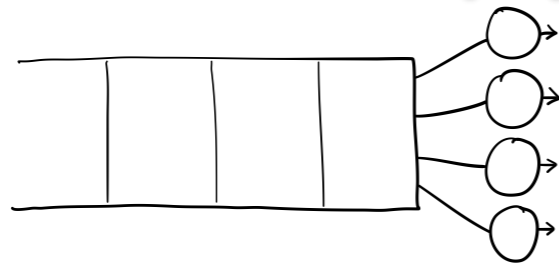
new bound!

mean r -work in M/G/1

Gittins in the M/G/k

k servers, each speed $1/k$

Gittins serves k jobs of k lowest **rank**s



mean response time in M/G/ k

mean response time in M/G/1

bound?

new connection!

new connection!

mean **r -work** in M/G/ k

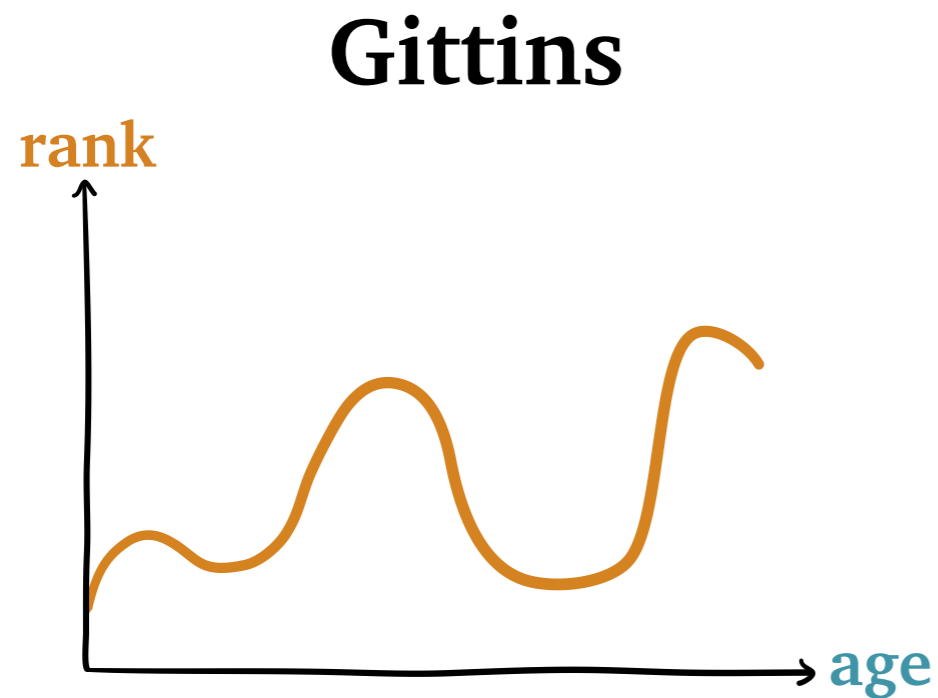
mean **r -work** in M/G/1

new bound!

What is *r*-Work?

r-Work: amount of service a job needs to either

- complete
- or reach Gittins **rank** $\geq r$



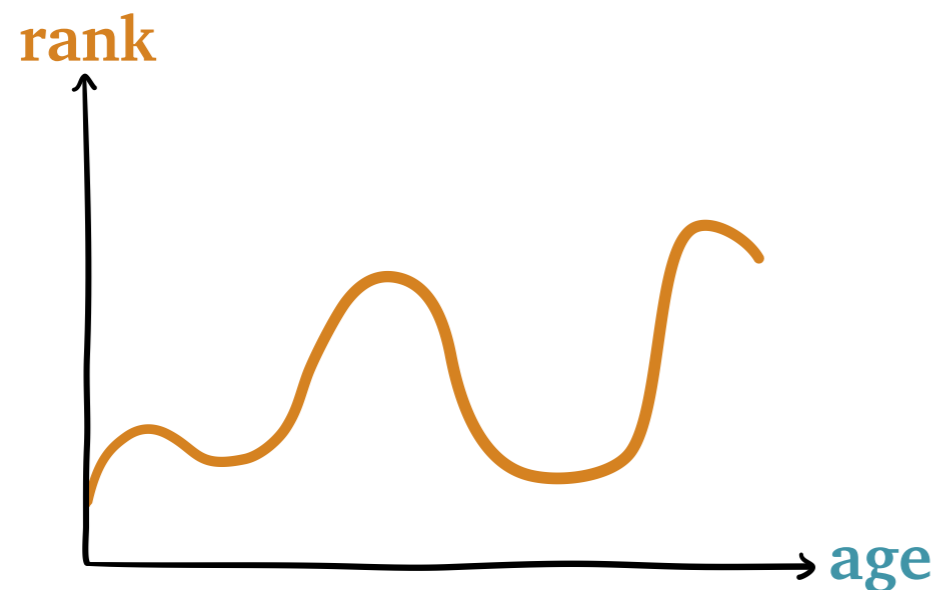
What is *r*-Work?

r-Work: amount of service a job needs to either

- complete
- or reach Gittins **rank** $\geq r$

depends on current age *a* and size *s*

Gittins



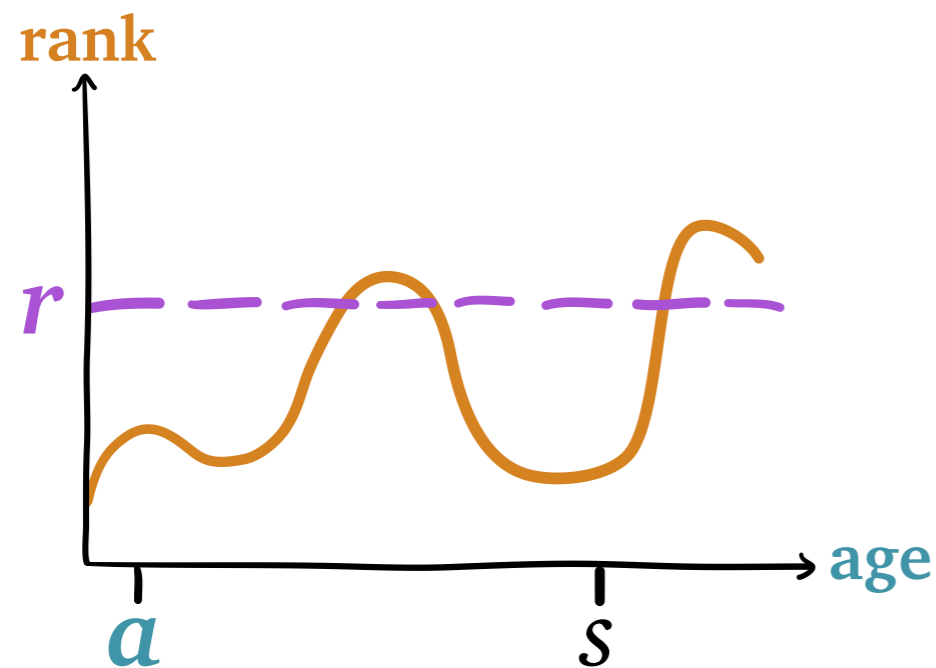
What is *r*-Work?

r-Work: amount of service a job needs to either

- complete
- or reach Gittins **rank** $\geq r$

depends on current age *a* and size *s*

Gittins

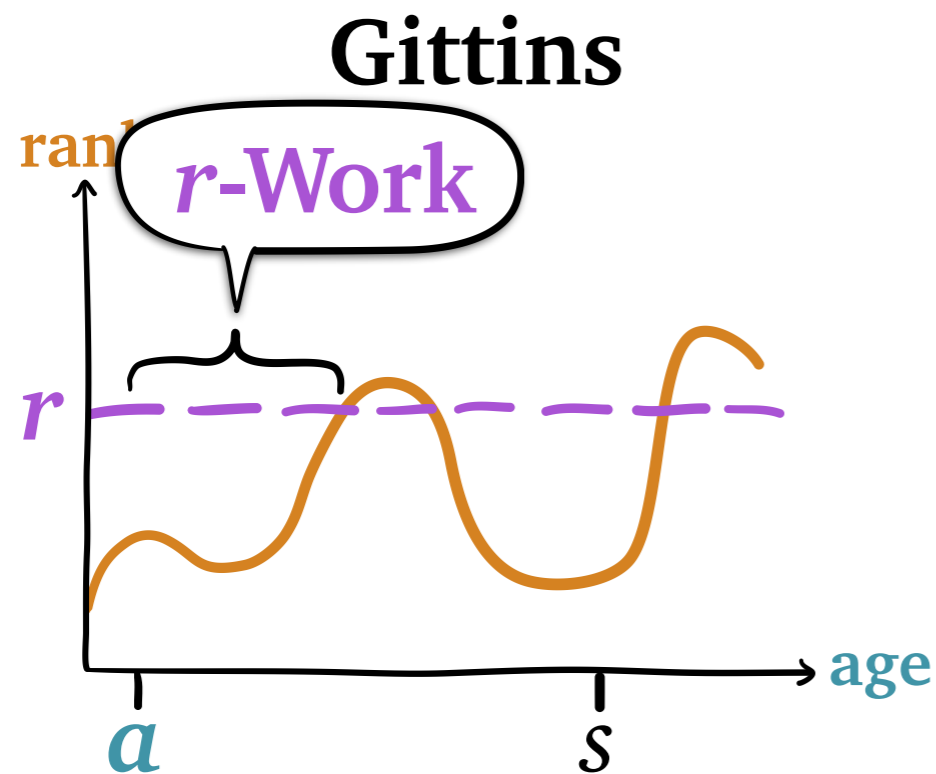


What is *r*-Work?

r-Work: amount of service a job needs to either

- complete
- or reach Gittins **rank** $\geq r$

depends on current age *a* and size *s*

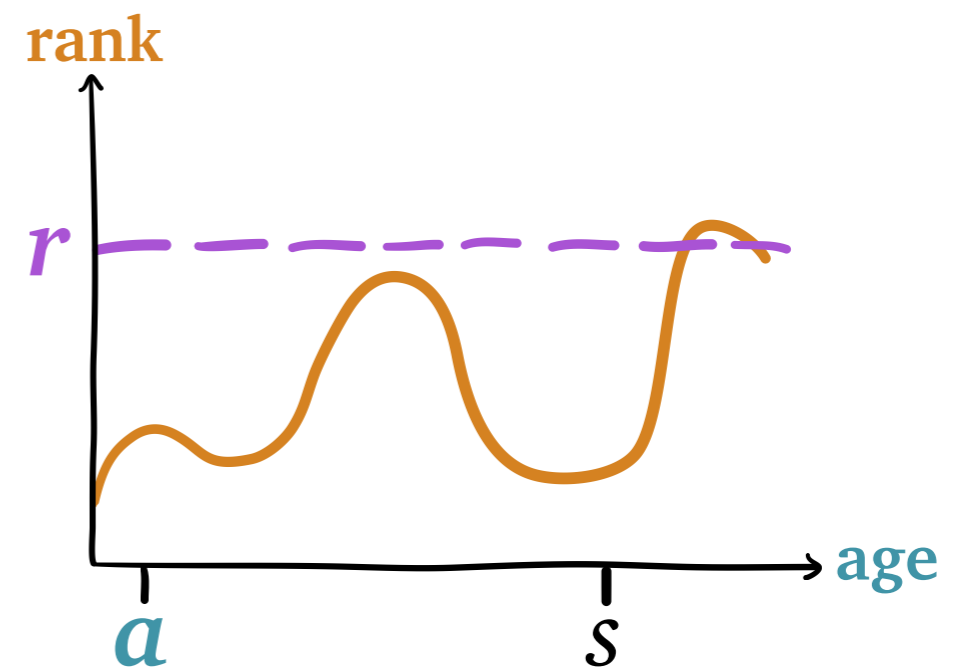
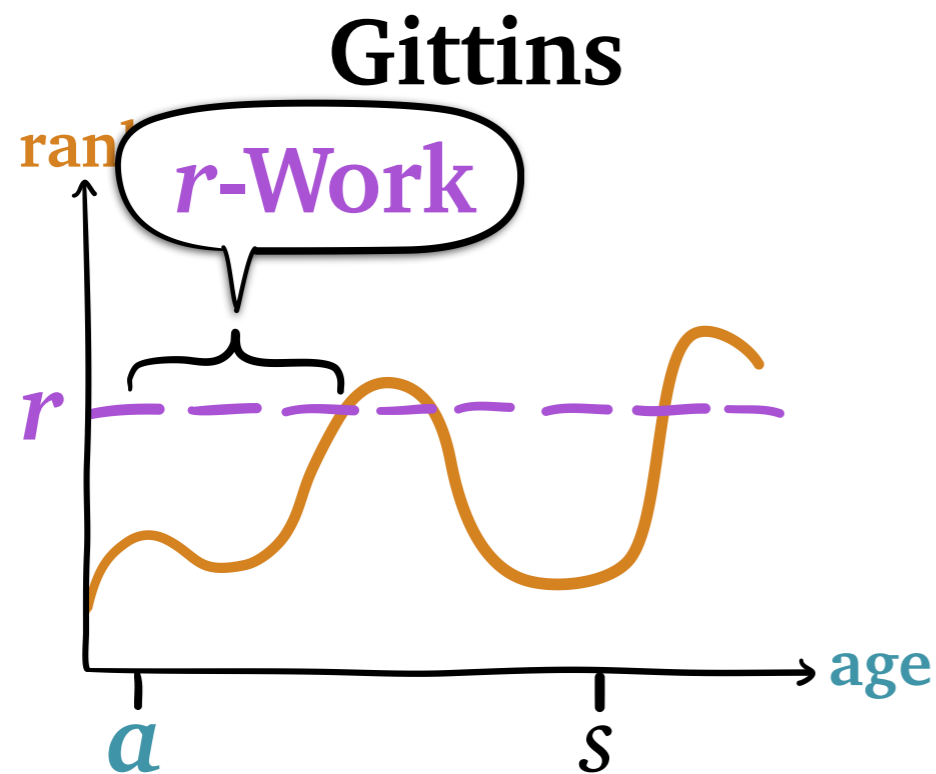


What is *r*-Work?

r-Work: amount of service a job needs to either

- complete
- or reach Gittins **rank** $\geq r$

depends on current age *a* and size *s*

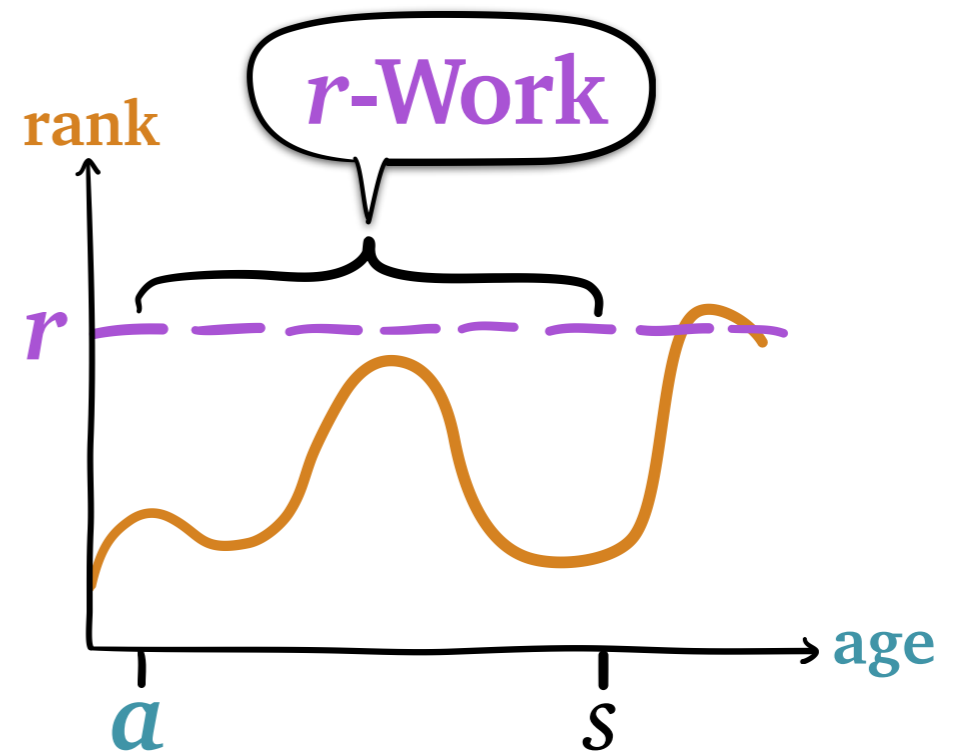
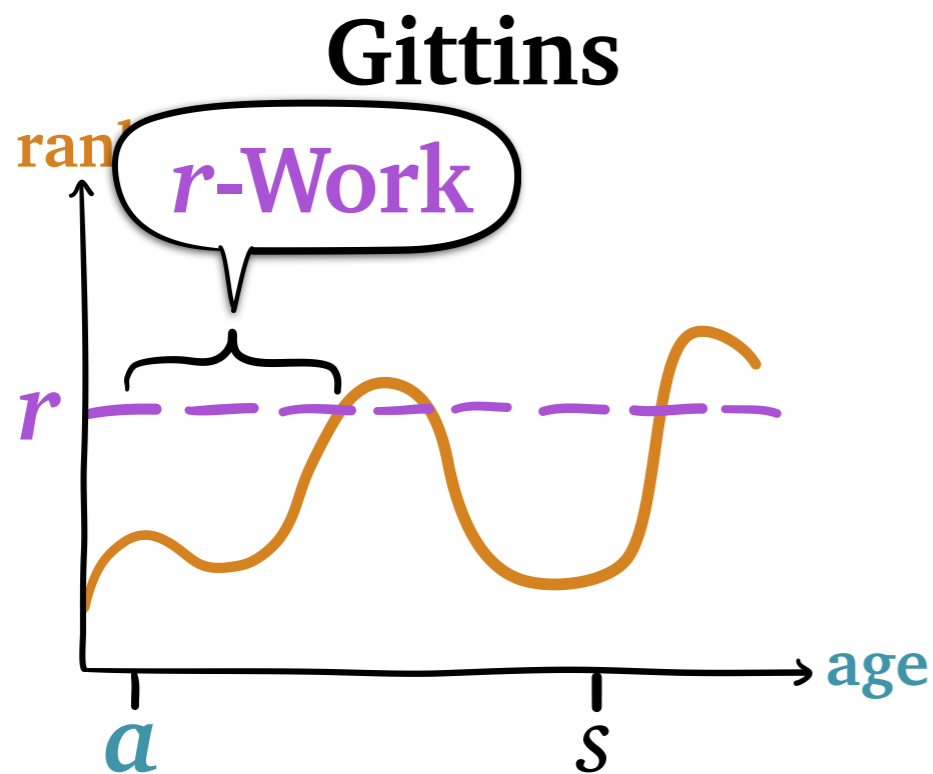


What is *r*-Work?

r-Work: amount of service a job needs to either

- complete
- or reach Gittins **rank** $\geq r$

depends on current age *a* and size *s*

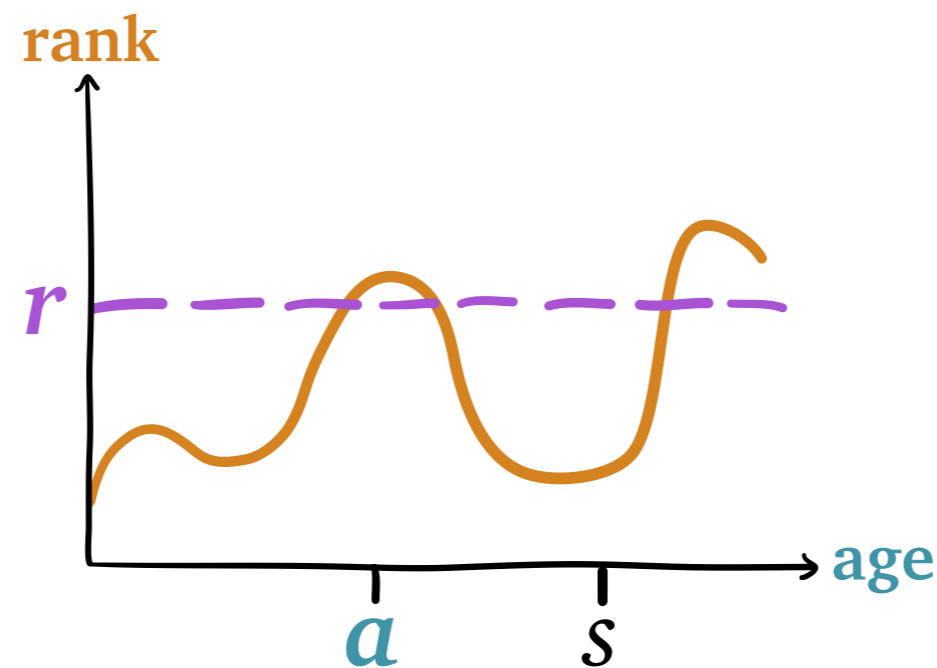


What is *r*-Work?

r-Work: amount of service a job needs to either

- complete
- or reach Gittins **rank** $\geq r$

depends on current age *a* and size *s*

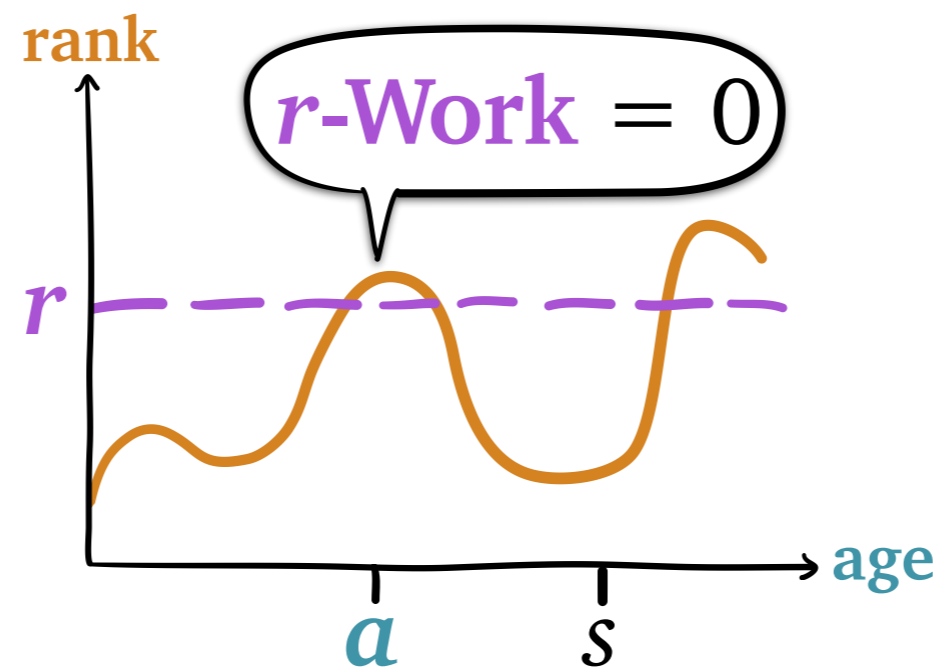


What is *r*-Work?

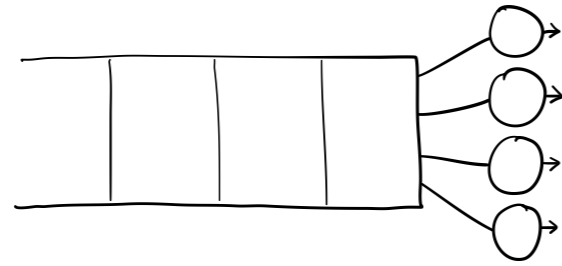
r-Work: amount of service a job needs to either

- complete
- or reach Gittins **rank** $\geq r$

depends on current age *a* and size *s*

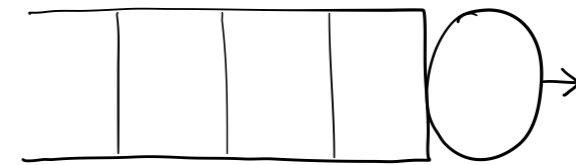


Gittins in the M/G/k



mean response
time in M/G/k

bound?



mean response
time in M/G/1

new connection!

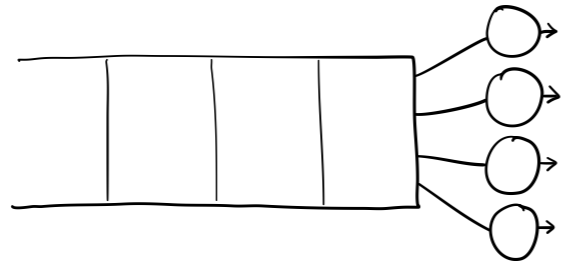
mean *r-work*
in M/G/k

new bound!

new connection!

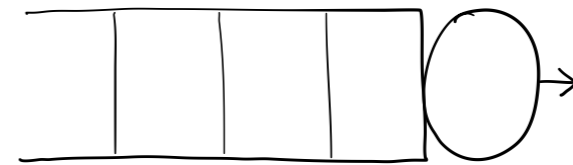
mean *r-work*
in M/G/1

Gittins in the M/G/k



mean response
time in M/G/k

bound?



mean response
time in M/G/1

new connection!

mean *r-work*
in M/G/k

new bound!

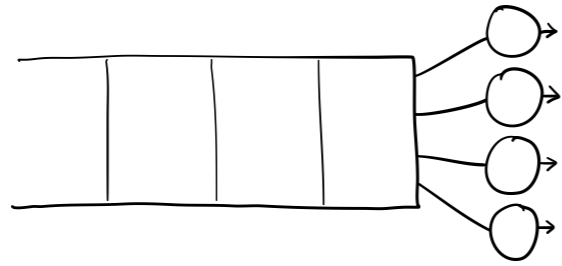
new connection!

mean *r-work*
in M/G/1

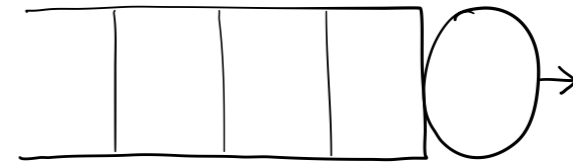
Theorem: under Gittins, if $E[S^{1+\varepsilon}] < \infty$ for some $\varepsilon > 0$,

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

Gittins in the M/G/k



mean response
time in M/G/k



mean response
time in M/G/1

bound?

new connection!

new connection!

mean *r-work*
in M/G/k

new bound!

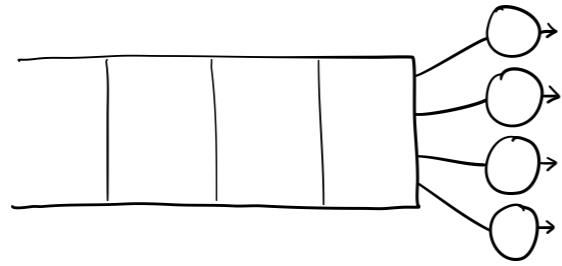
mean *r-work*
in M/G/1

$$\omega\left(\log \frac{1}{1-\rho}\right) \text{ if } \mathbf{E}[S^2(\log S)^+] < \infty$$

Gittins, if $\mathbf{E}[S^{1+\varepsilon}] < \infty$ for some $\varepsilon > 0$,

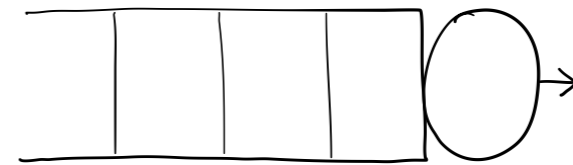
$$\mathbf{E}[T_k] \leq \mathbf{E}[T_1] + (k-1) \cdot O\left(\log \frac{1}{1-\rho}\right)$$

Gittins in the M/G/k



mean response
time in M/G/k

bound?



mean response
time in M/G/1

new connection!

mean *r-work*
in M/G/k

new bound!

mean *r-work*
in M/G/1

new connection!

$$\omega\left(\log \frac{1}{1-\rho}\right) \text{ if } \mathbf{E}[S^2(\log S)^+] < \infty$$

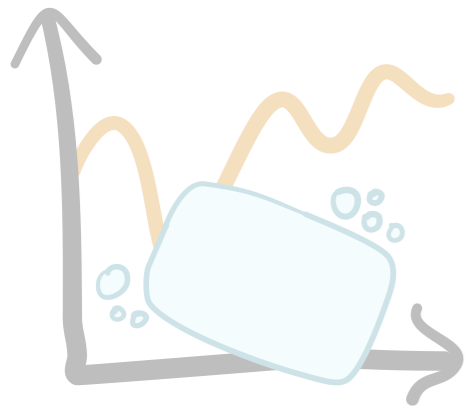
Gittins, if $\mathbf{E}[S^{1+\varepsilon}] < \infty$ for some $\varepsilon > 0$,

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

[Scully, Grosf, & Harchol-Balter, SIGMETRICS 2021]

Overview

New Tools



SOAP

analyzes a huge variety of scheduling heuristics

r-Work

provides a new, deeper understanding of Gittins

Goals

Multiple servers



Simple implementation preferred



Preemption restricted and/or costly



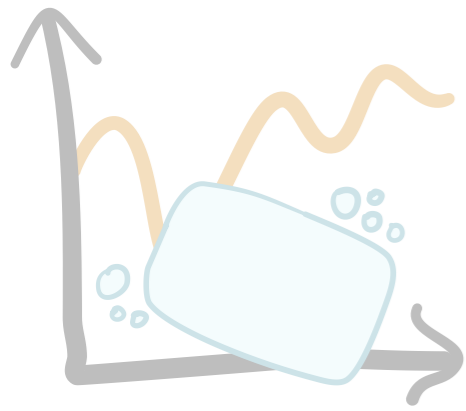
Limited number of priority levels



Want to optimize other response time metrics

Overview

New Tools



SOAP

analyzes a huge variety of scheduling heuristics

r-Work

provides a new, deeper understanding of Gittins

Goals



Multiple servers



Simple implementation preferred



Preemption restricted and/or costly



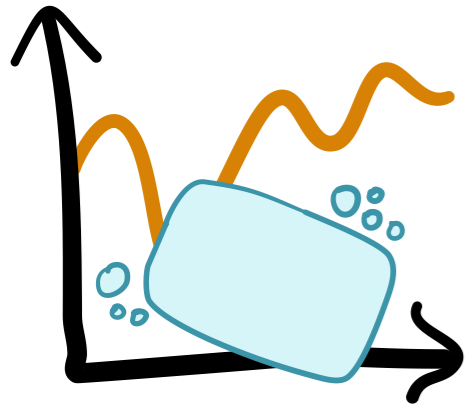
Limited number of priority levels



Want to optimize other response time metrics

Overview

New Tools



SOAP

analyzes a huge variety of scheduling heuristics

r-Work

provides a new, deeper understanding of Gittins

Goals



Multiple servers



Simple implementation preferred



Preemption restricted and/or costly



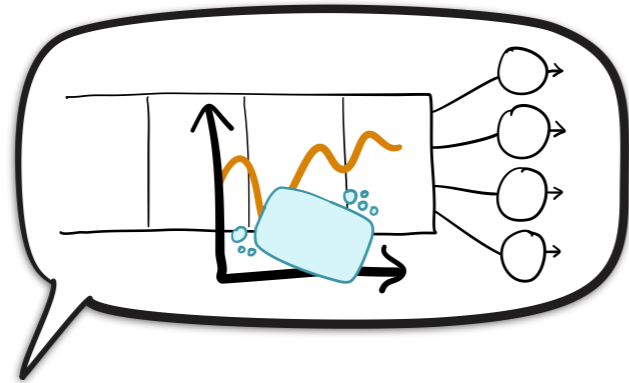
Limited number of priority levels



Want to optimize other response time metrics

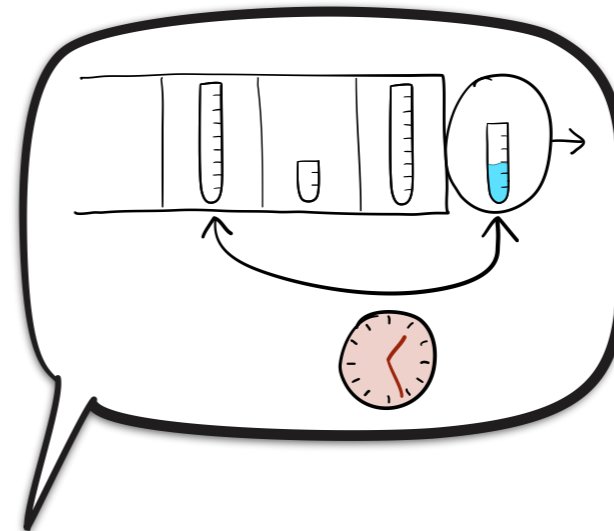
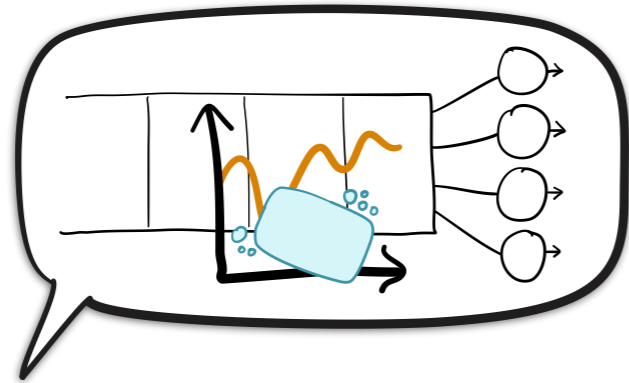
Future Work

Future Work



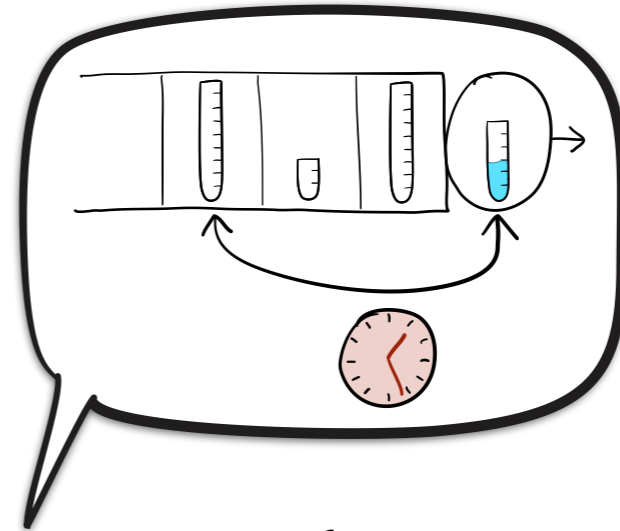
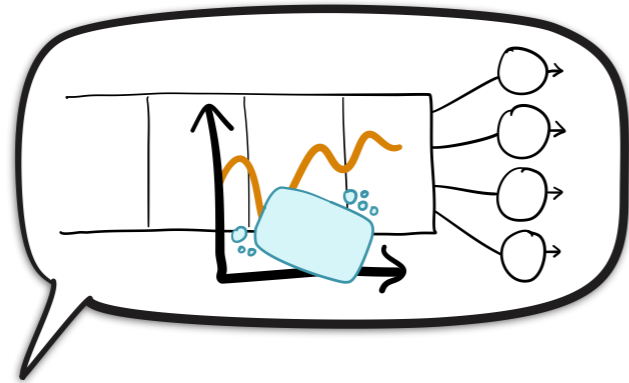
- **SOAP** for $M/G/k$

Future Work

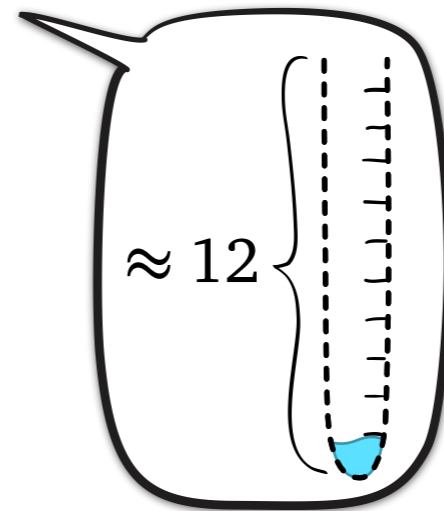


- **SOAP** for $M/G/k$
- Preemption costs with unrestricted preemption timing

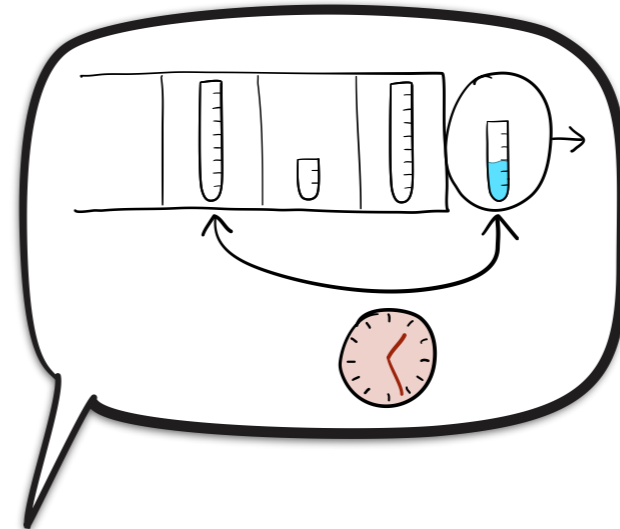
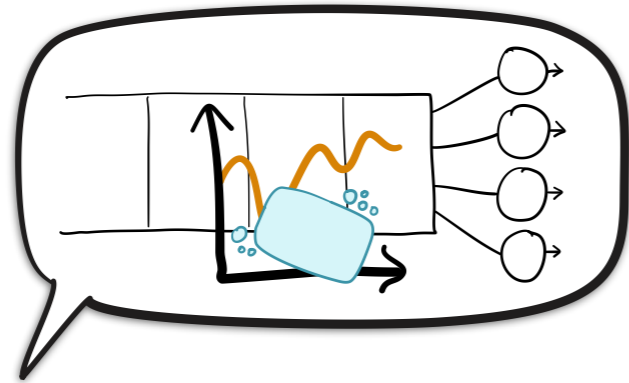
Future Work



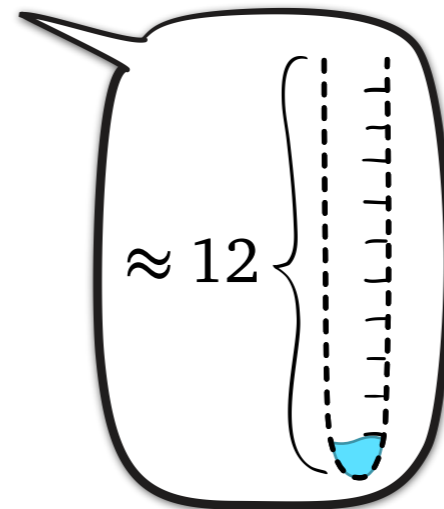
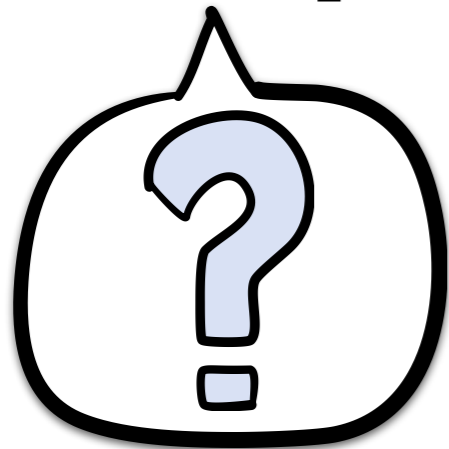
- **SOAP** for $M/G/k$
- Preemption costs with unrestricted preemption timing
- Simplifying Gittins for noisy size estimates



Future Work



- **SOAP** for $M/G/k$
- Preemption costs with unrestricted preemption timing
- Simplifying Gittins for noisy size estimates
- *Your problem here!*



References

Ziv Scully, Mor Harchol-Balter, and Alan Scheller-Wolf.

SOAP: One Clean Analysis of All Age-Based Scheduling Policies.

POMACS, 2018. Presented at SIGMETRICS 2018.

Ziv Scully and Mor Harchol-Balter.

SOAP Bubbles: Robust Scheduling under Adversarial Noise.

Allerton Conference, 2018.

Isaac Grosf, Ziv Scully, and Mor Harchol-Balter.

SRPT for Multiserver Systems.

PEVA, 2018. Presented at PERFORMANCE 2018.

Ziv Scully, Mor Harchol-Balter, and Alan Scheller-Wolf.

Simple Near-Optimal Scheduling for the M/G/1.

POMACS, 2020. Presented at SIGMETRICS 2020.

Ziv Scully, Lucas van Kreveld, Onno J. Boxma, Jan-Pieter Dorsman, and Adam Wierman.

Characterizing Policies with Optimal Response Time Tails under Heavy-Tailed Job Sizes.

POMACS, 2020. Presented at SIGMETRICS 2020.

Ziv Scully, Isaac Grosf, and Mor Harchol-Balter.

Optimal Multiserver Scheduling with Unknown Job Sizes in Heavy Traffic.

PEVA, 2020. Presented at PERFORMANCE 2020.

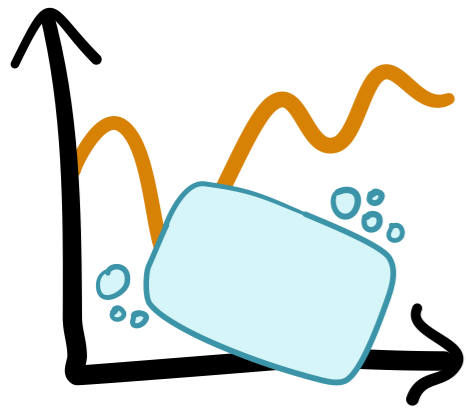
Ziv Scully, Isaac Grosf, and Mor Harchol-Balter.

The Gittins Policy is Nearly Optimal in the M/G/k under Extremely General Conditions.

POMACS, 2020. To be presented at SIGMETRICS 2021.

Overview

New Tools



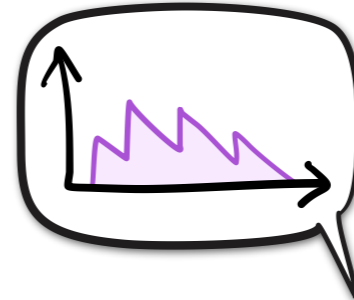
SOAP

analyzes a huge variety of scheduling heuristics

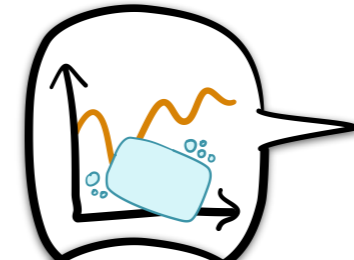
r-Work

provides a new, deeper understanding of Gittins

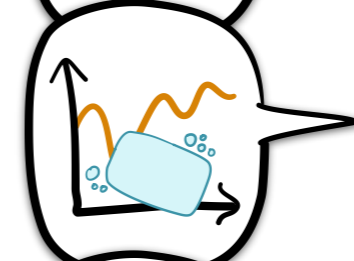
Goals



Multiple servers



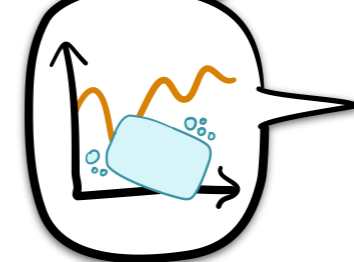
Simple implementation preferred



Preemption restricted and/or costly



Limited number of priority levels



Want to optimize other response time metrics