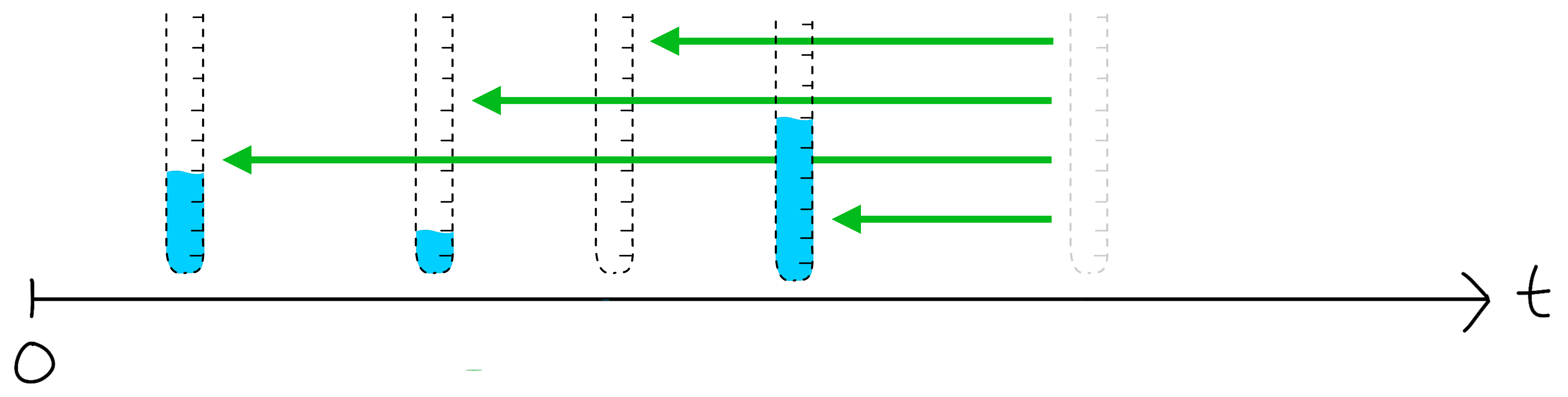
A Gittins Policy for Optimizing Tail Latency

Amit Harlev Cornell CAM

Joint work with

George Yu Cornell ORIE
Ziv Scully Cornell ORIE

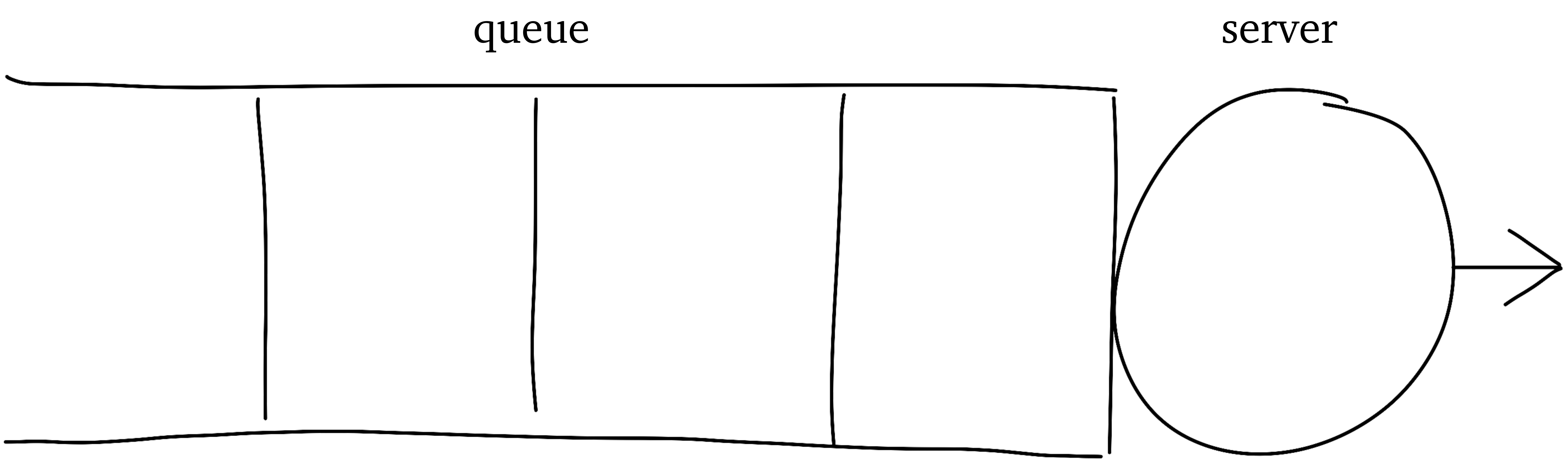


How do we minimize delays when job sizes are unknown?

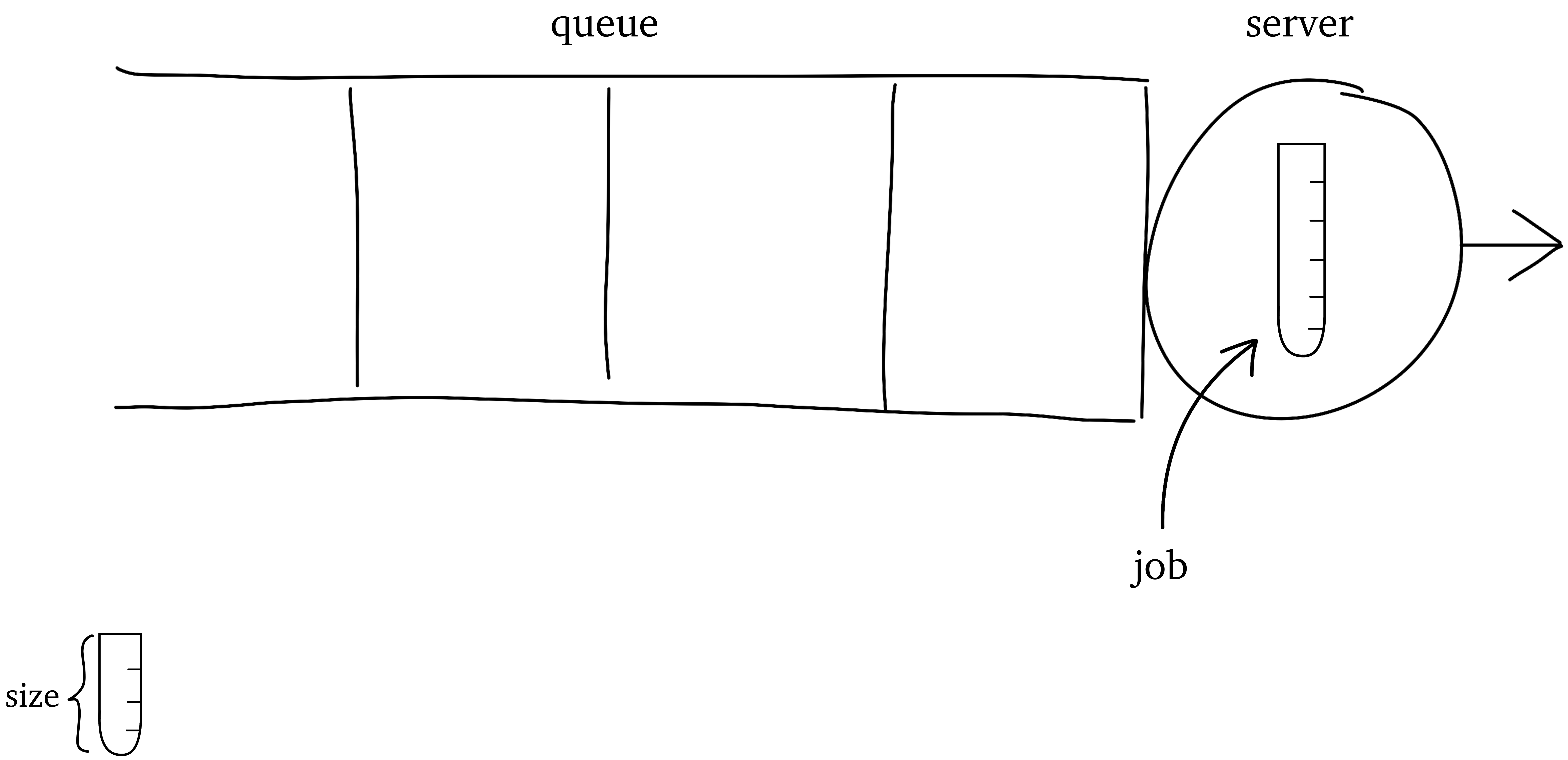
(asymptotic) tail latency in single server queue

How do we minimize delays when job sizes are unknown?

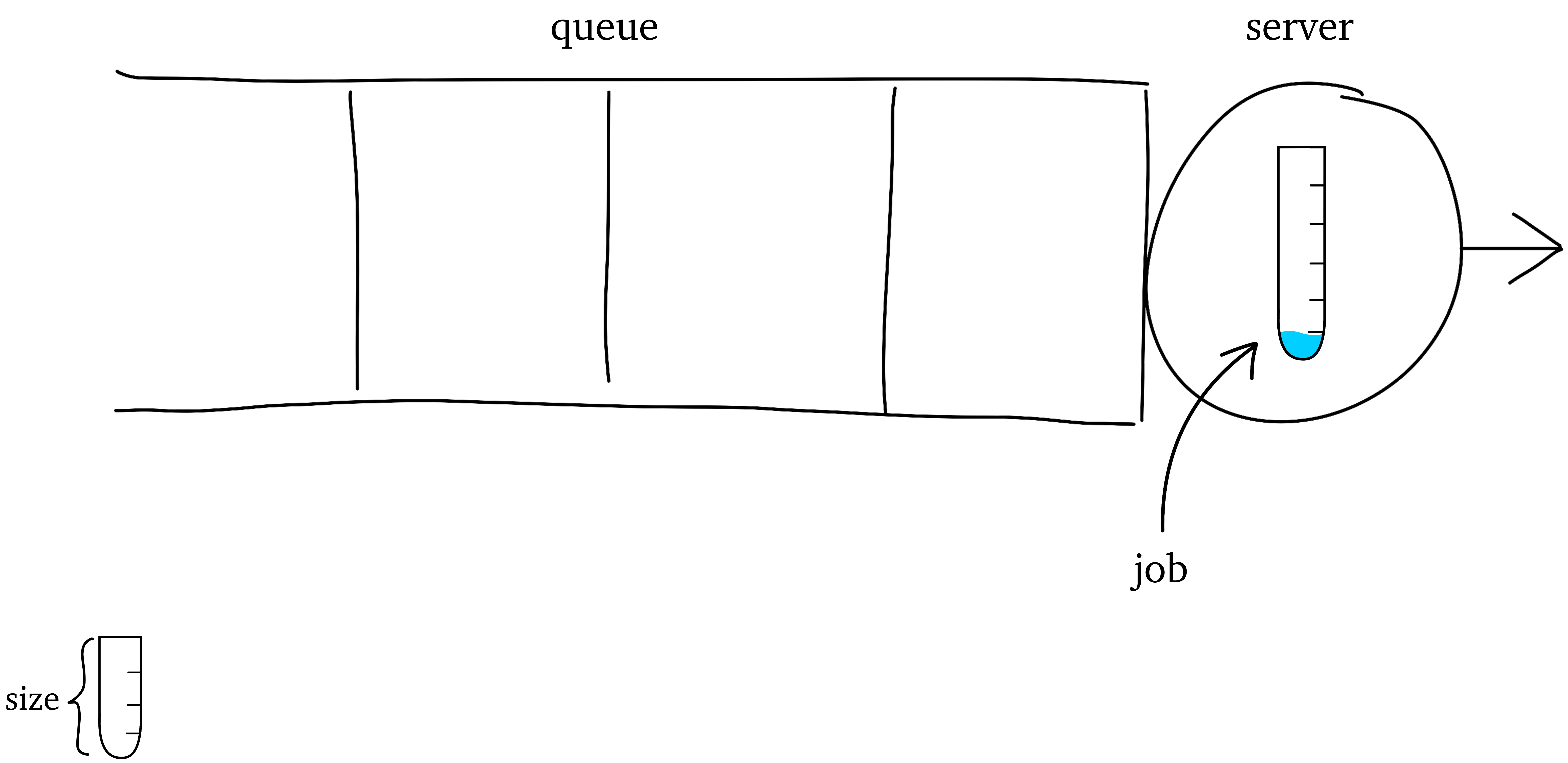
Scheduling in the M/G/1



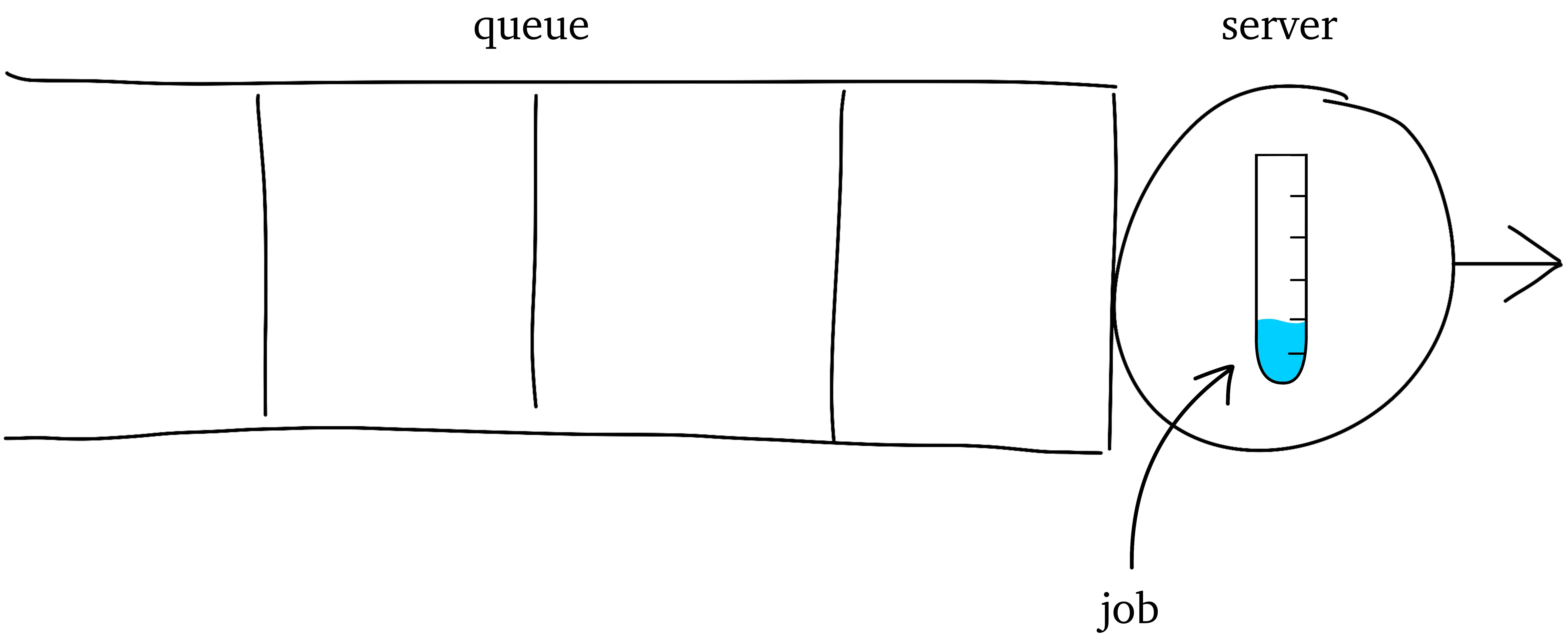
Scheduling in the M/G/1

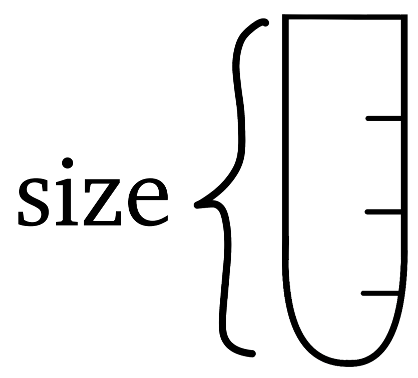


Scheduling in the M/G/1

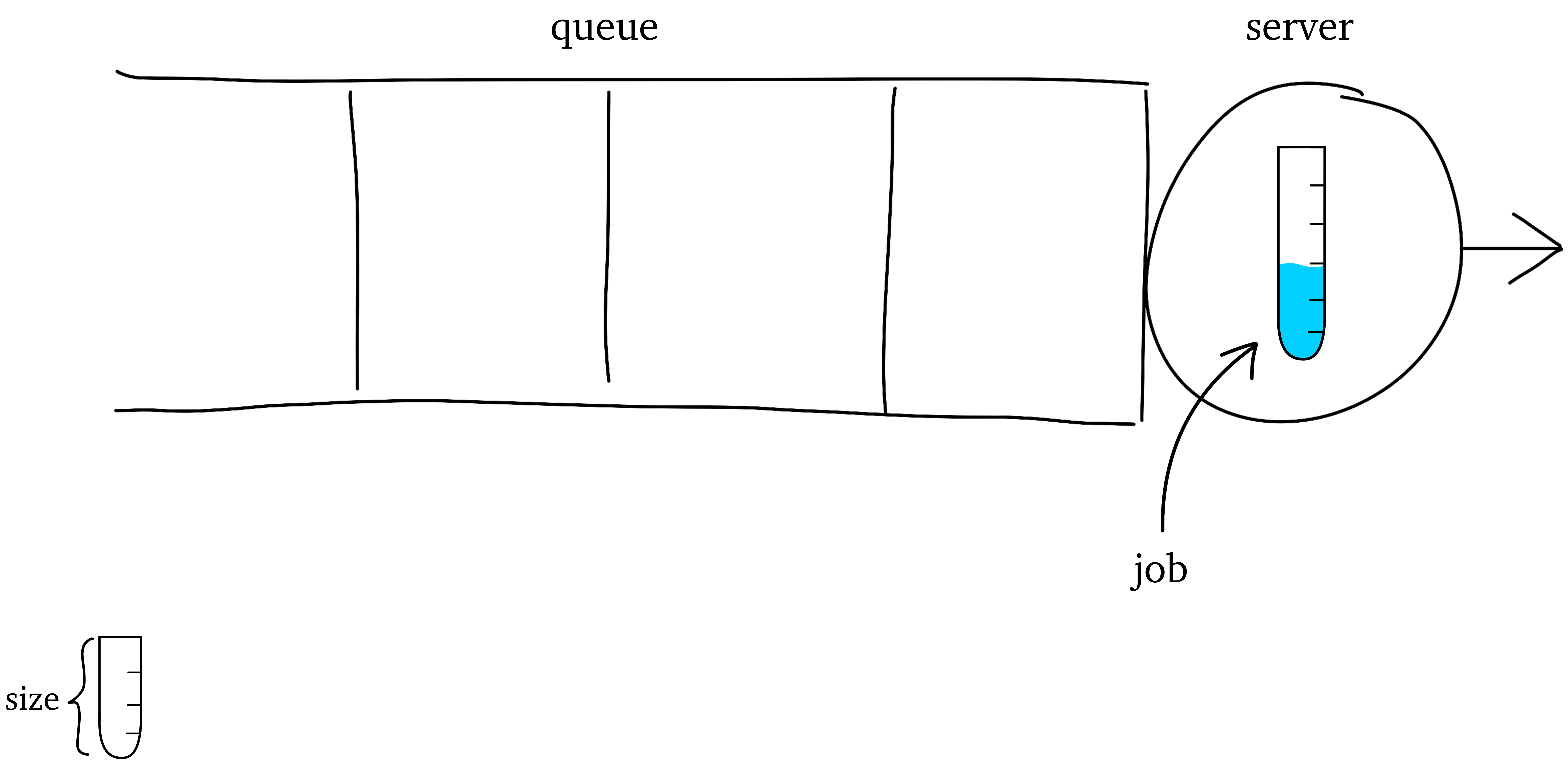


Scheduling in the M/G/1

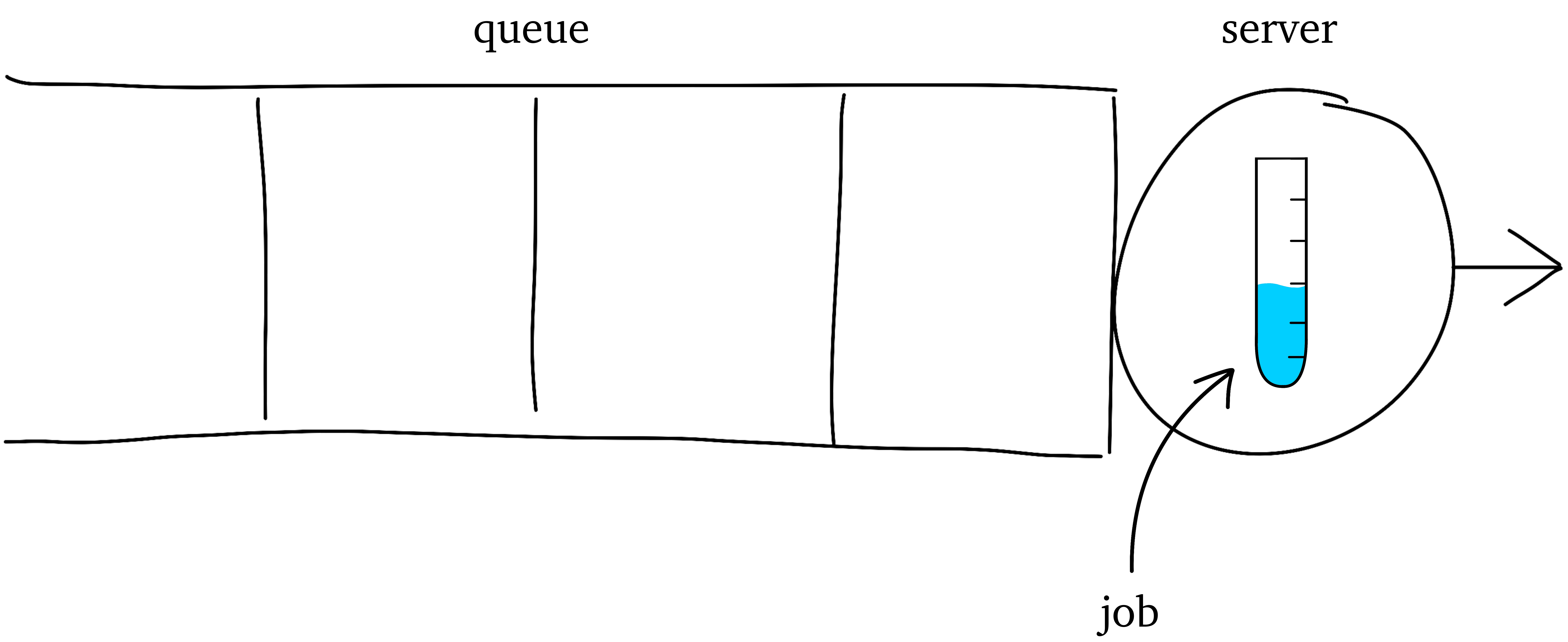


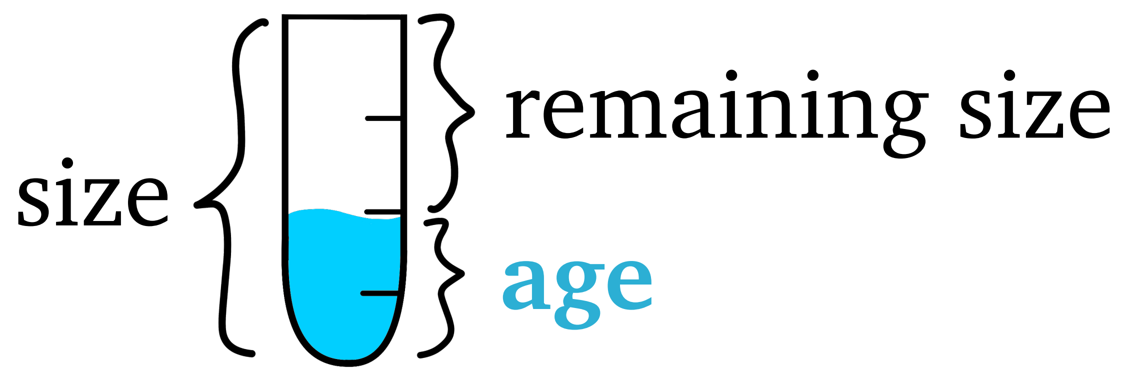


Scheduling in the M/G/1

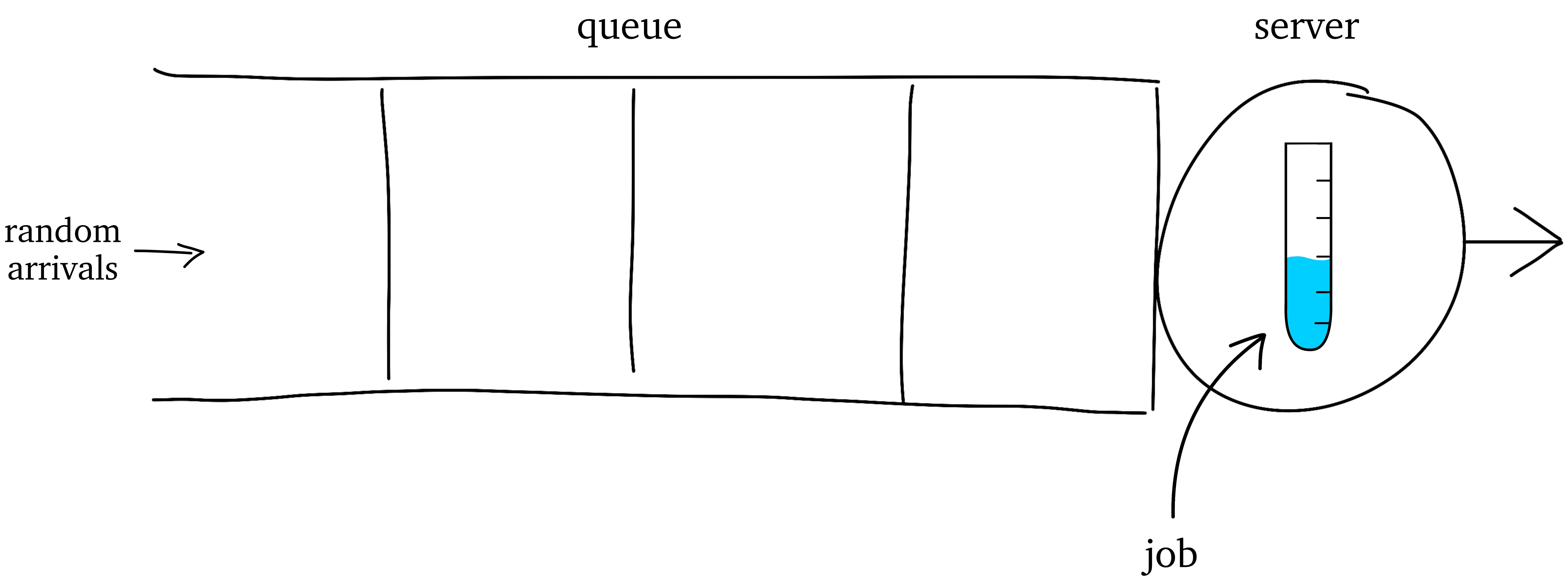


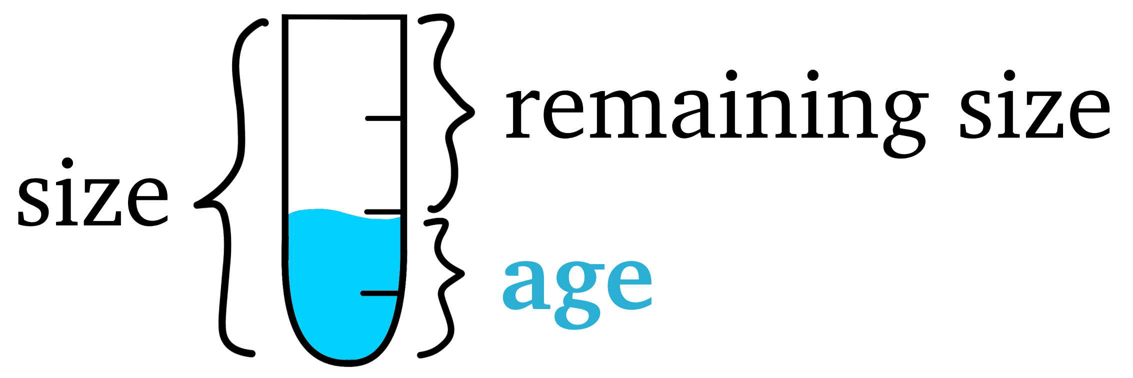
Scheduling in the M/G/1



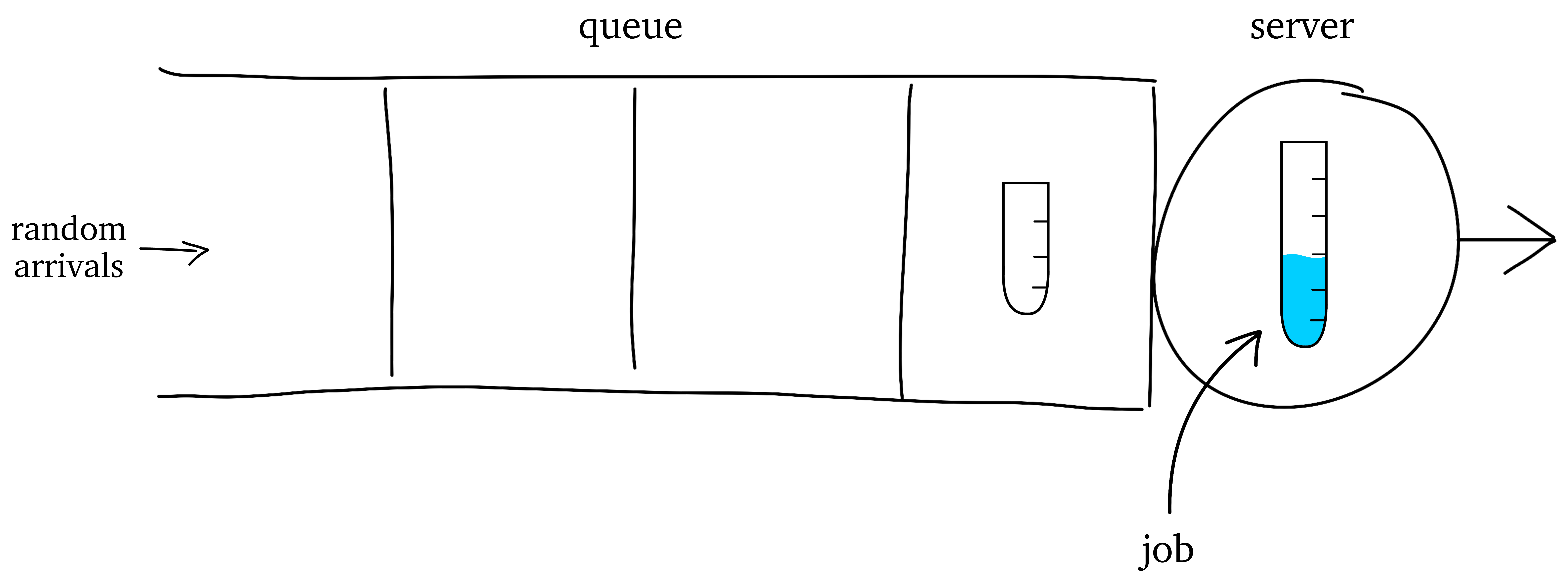


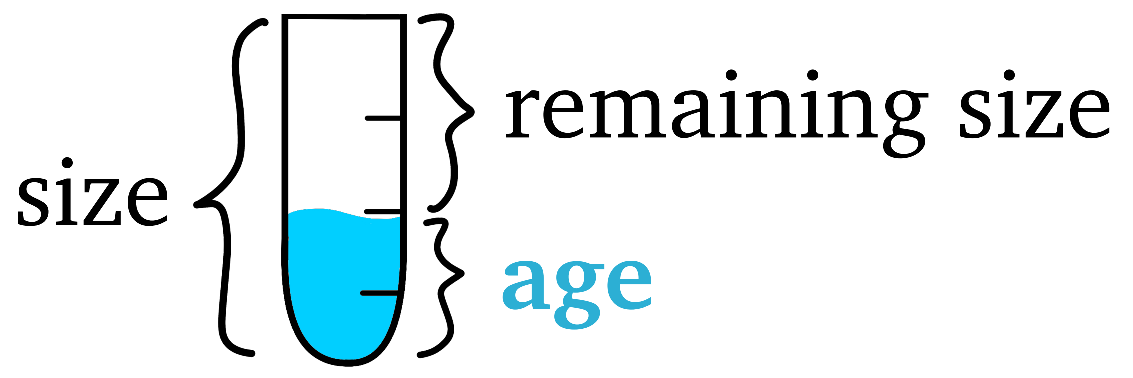
Scheduling in the M/G/1



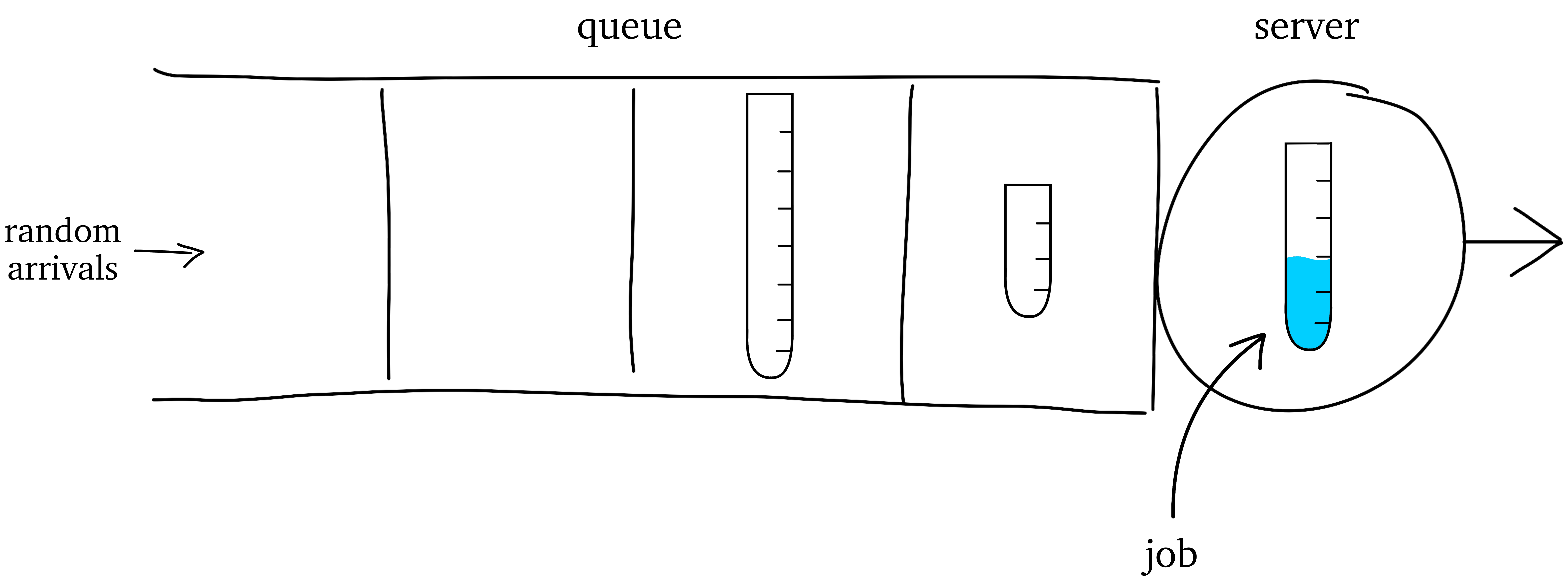


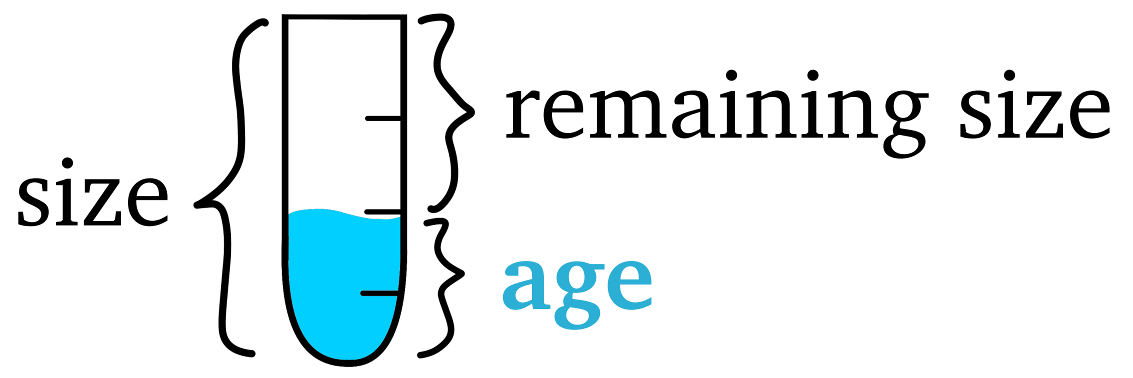
Scheduling in the M/G/1



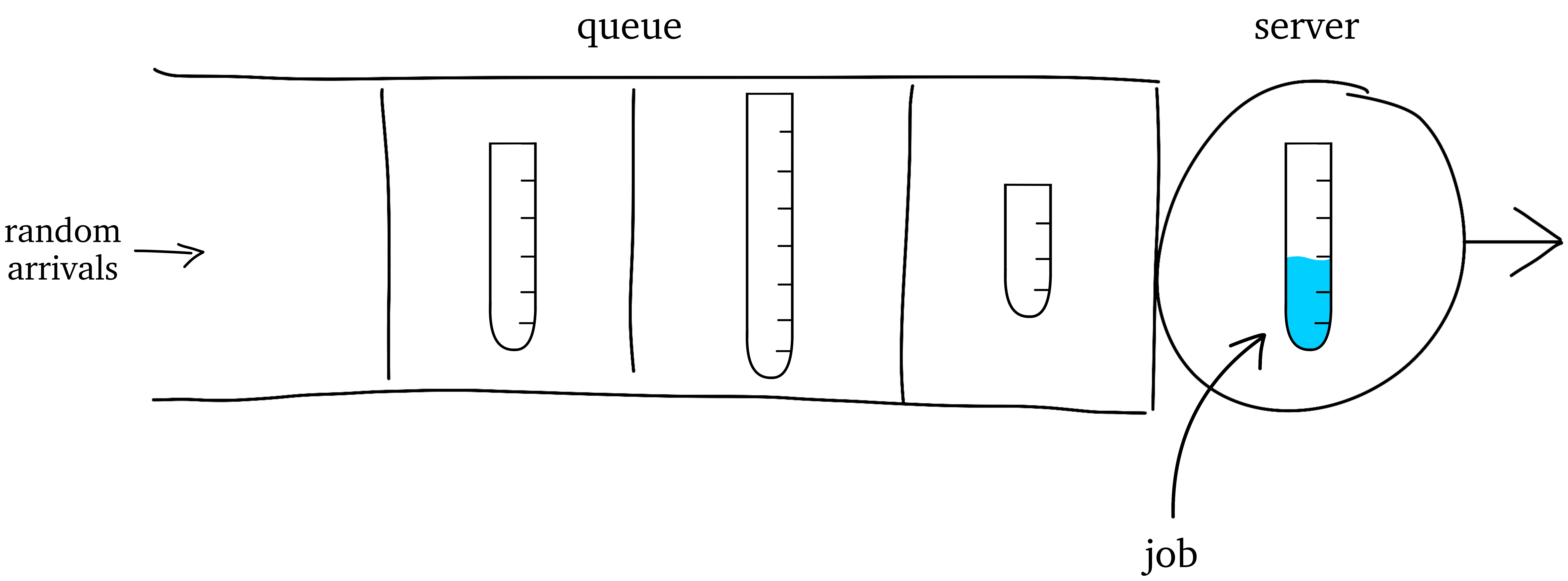


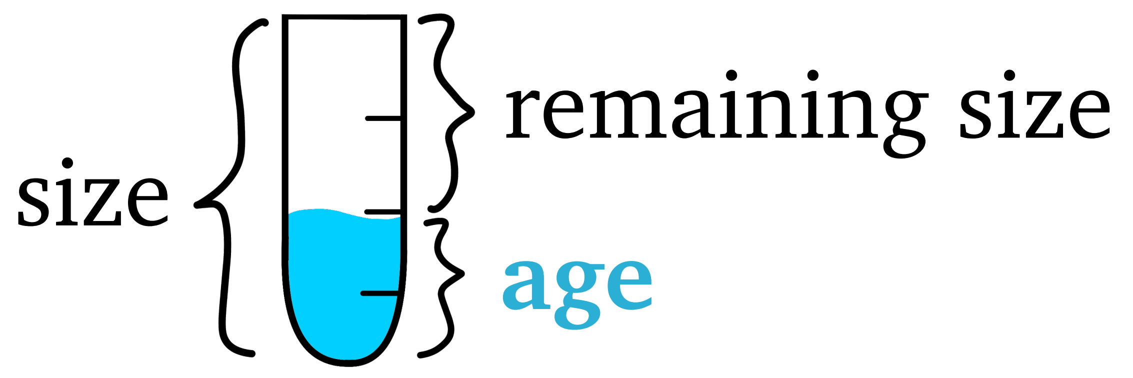
Scheduling in the M/G/1



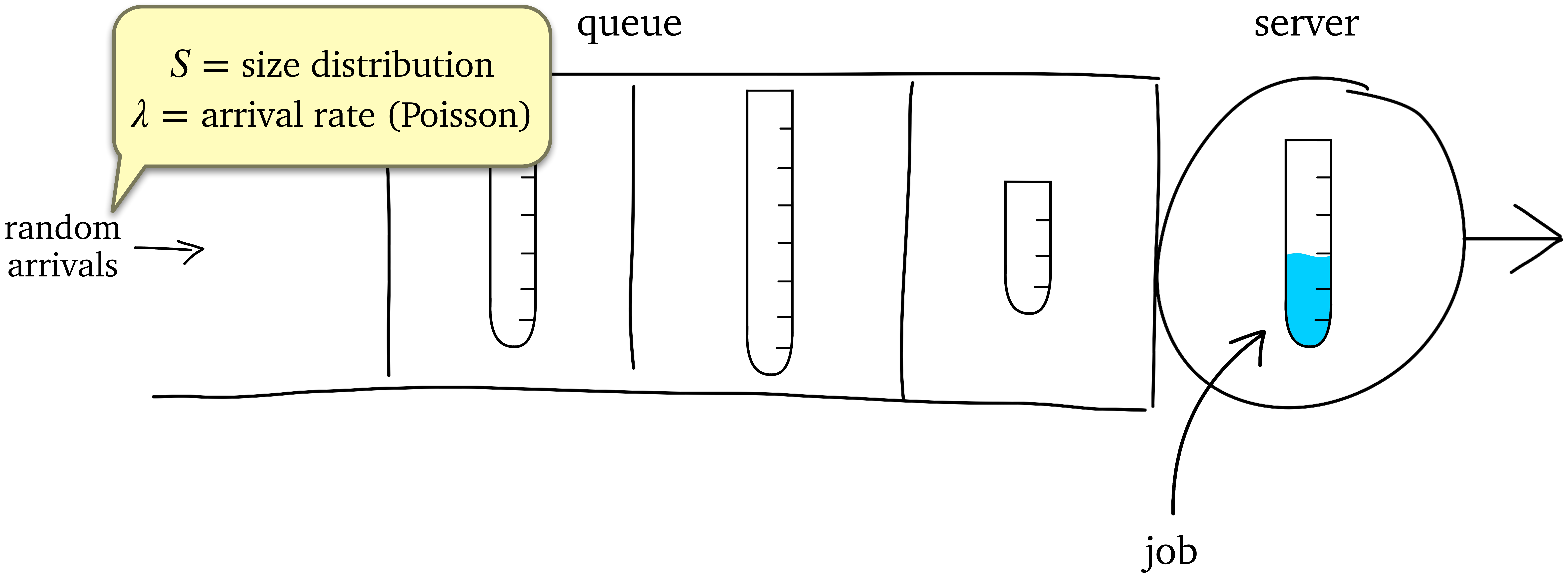


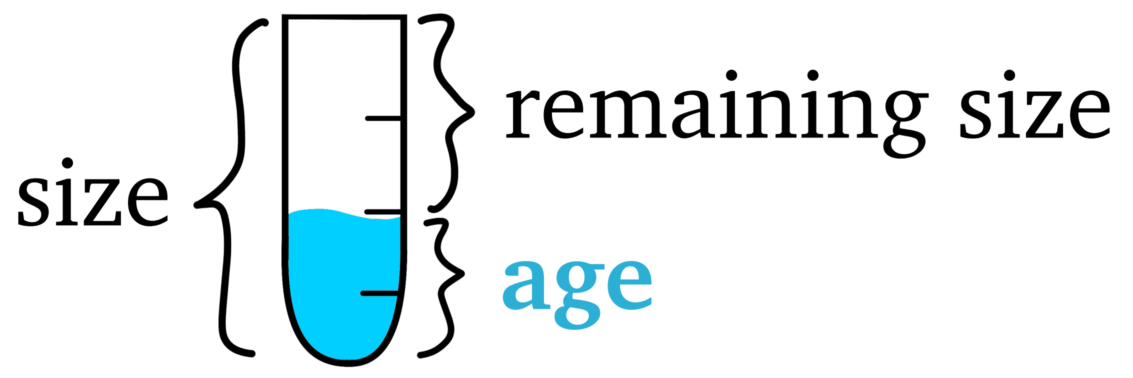
Scheduling in the M/G/1



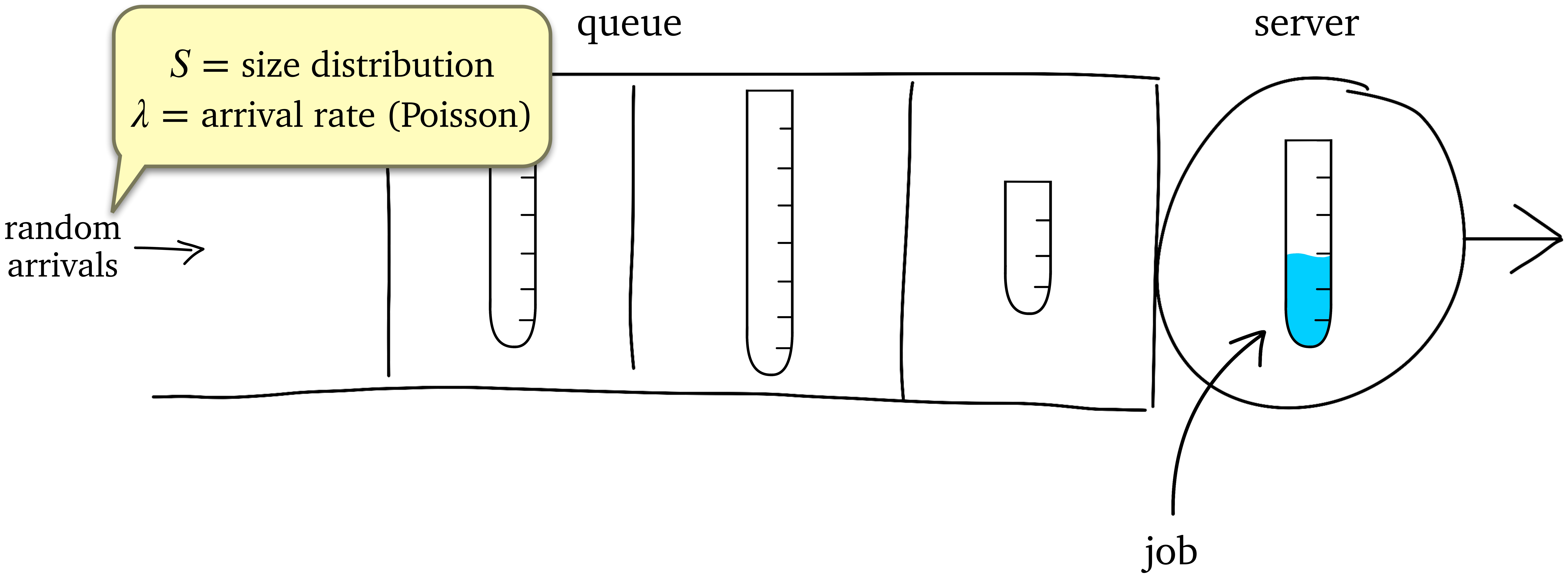


Scheduling in the M/G/1

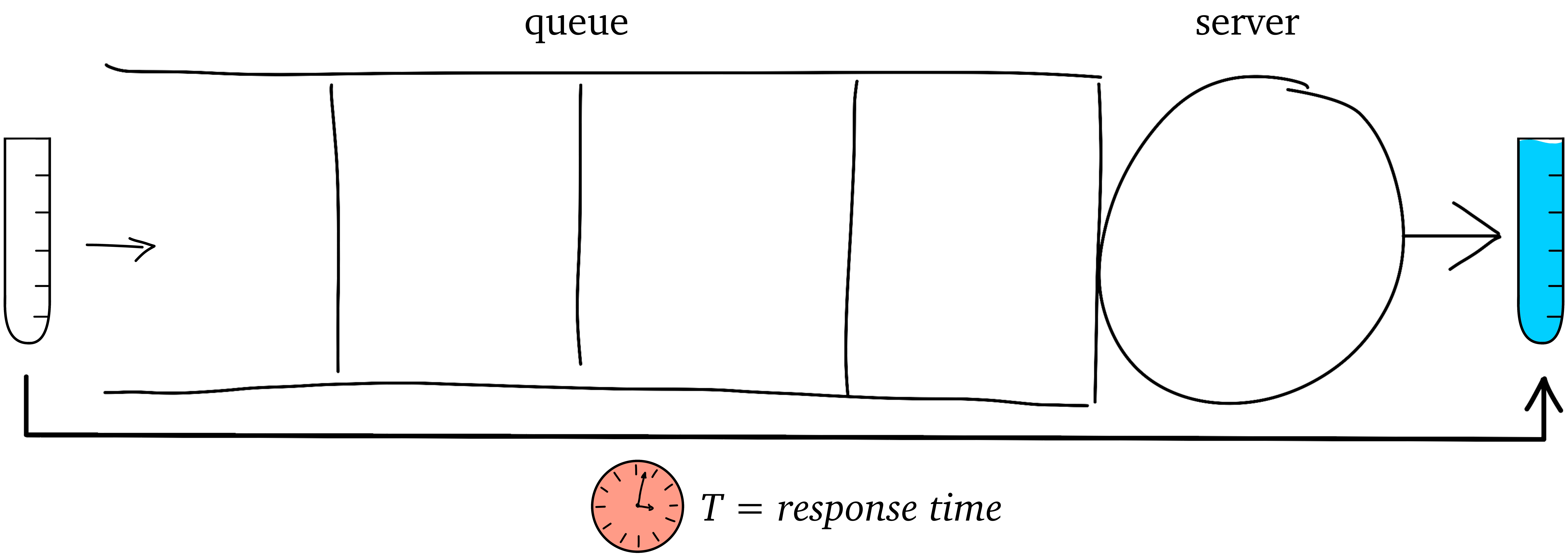




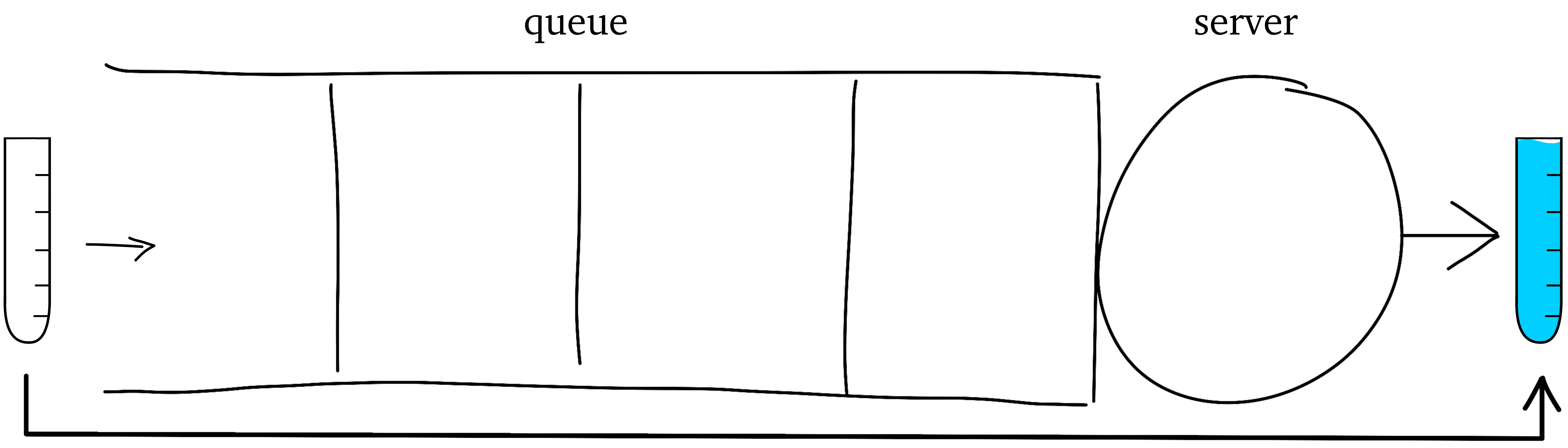
Scheduling in the M/G/1



Scheduling: In which order should we serve jobs to minimize a desired metric?

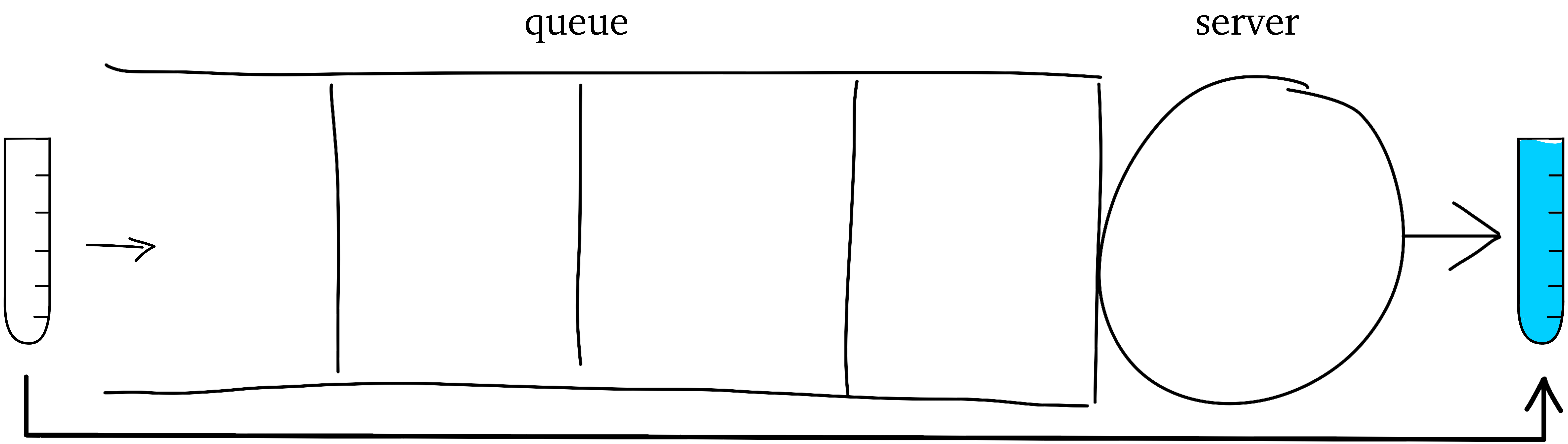


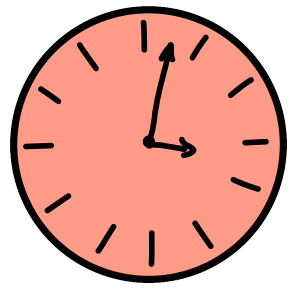
Scheduling: In which order should we serve jobs to minimize a desired metric?



T = *response time*

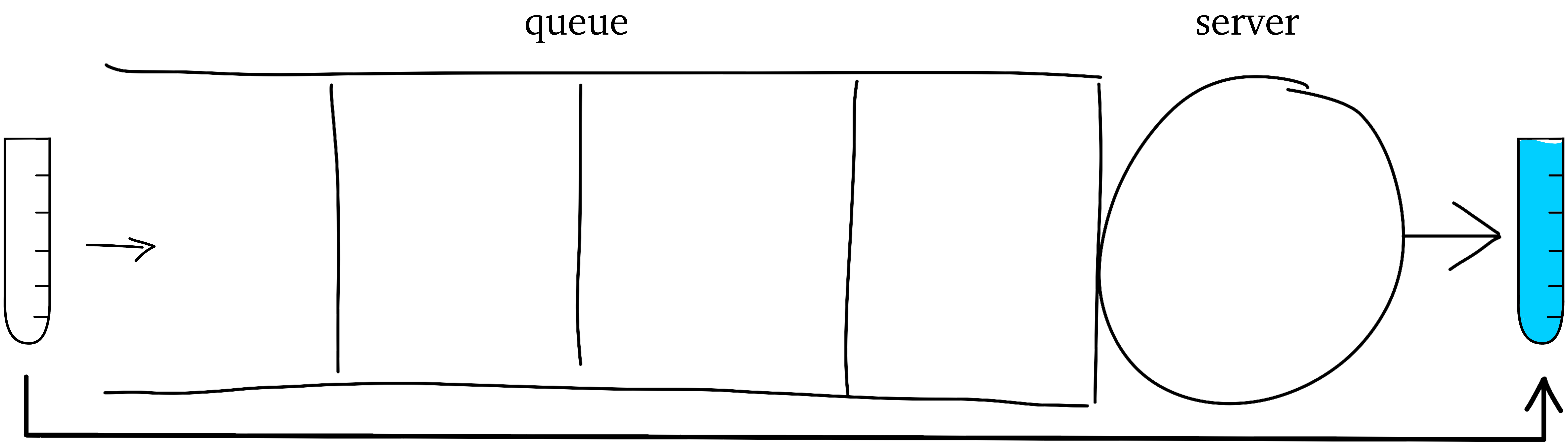
Theory

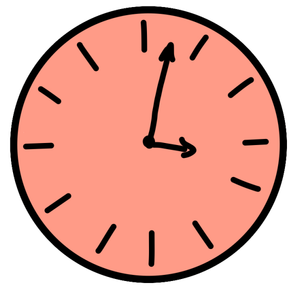


 T = *response time*

 **Theory**

- mean response time, $\mathbf{E}[T]$

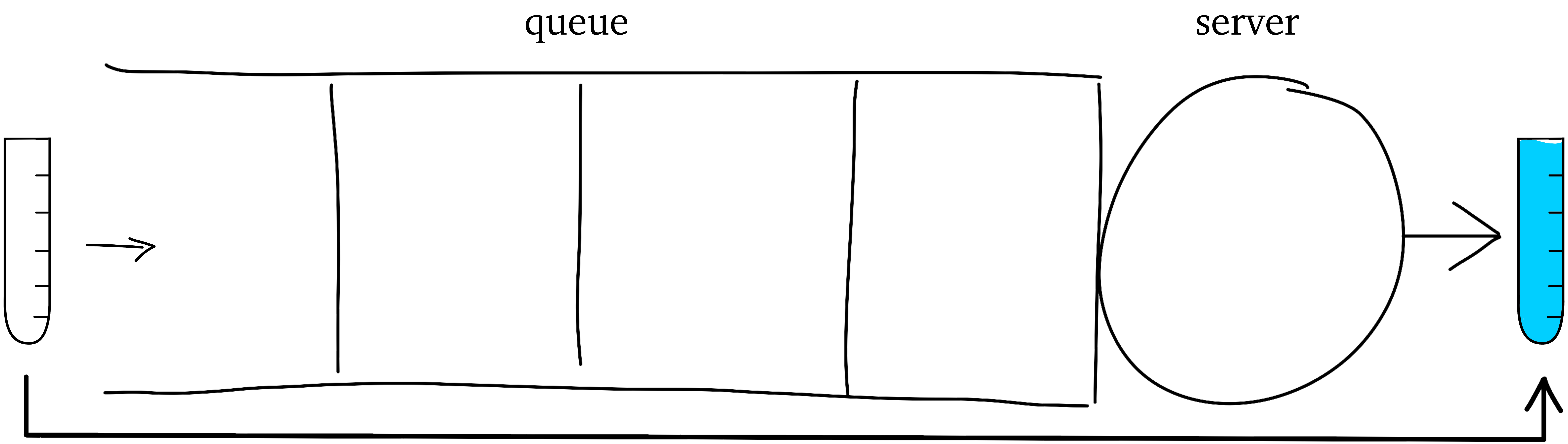


 T = *response time*

SRPT

🎓 **Theory**

- mean response time, $\mathbf{E}[T]$

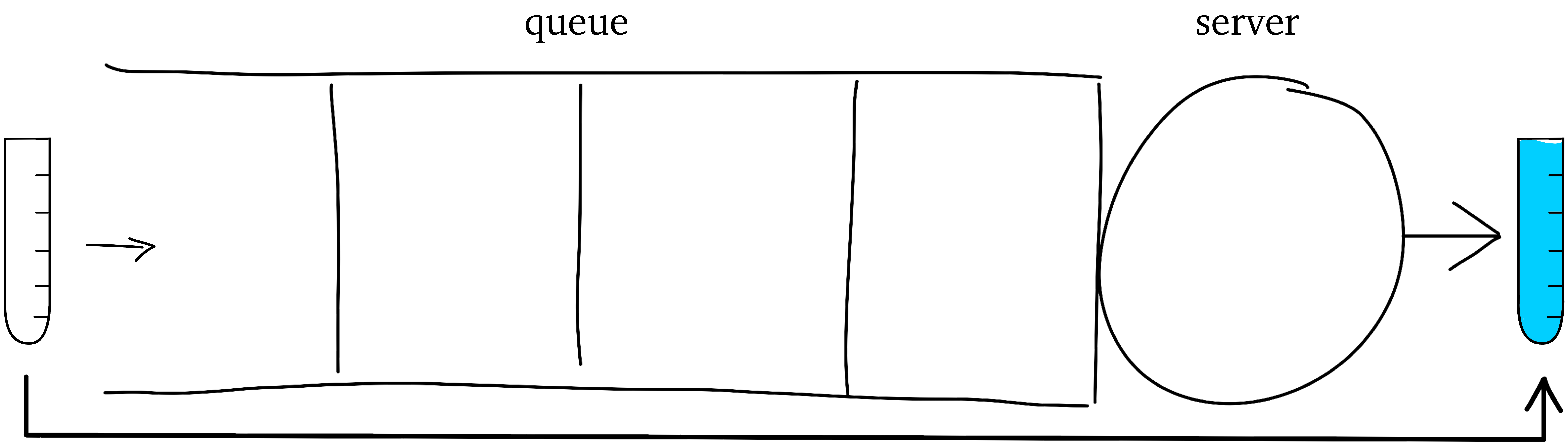


T = *response time*

SRPT

Theory

- mean response time, $\mathbf{E}[T]$
- $\mathbf{E}[T]$ with unknown job sizes



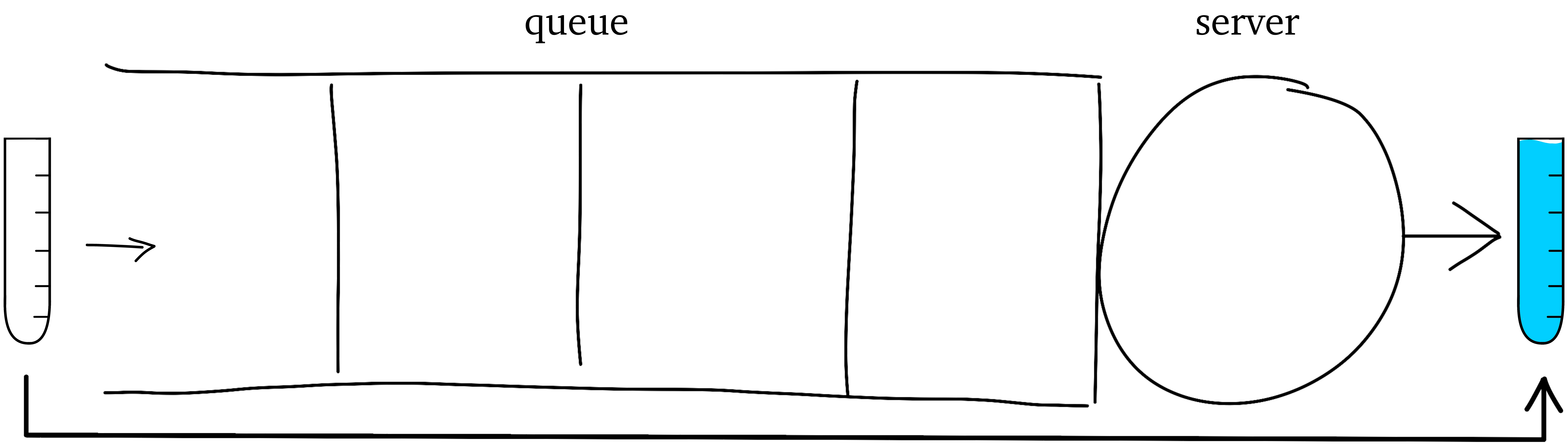
T = *response time*

SRPT

Theory

- mean response time, $\mathbf{E}[T]$
- $\mathbf{E}[T]$ with unknown job sizes

Gittins



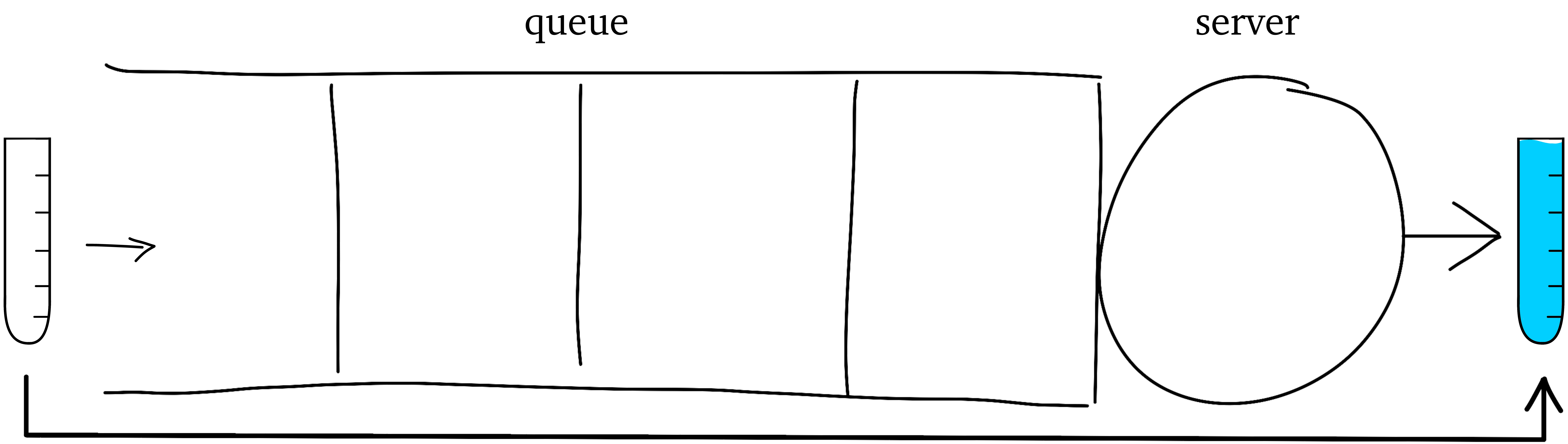
T = *response time*

SRPT

Theory

- mean response time, $\mathbf{E}[T]$
- $\mathbf{E}[T]$ with unknown job sizes
- weighted $\mathbf{E}[T]$

Gittins



T = *response time*

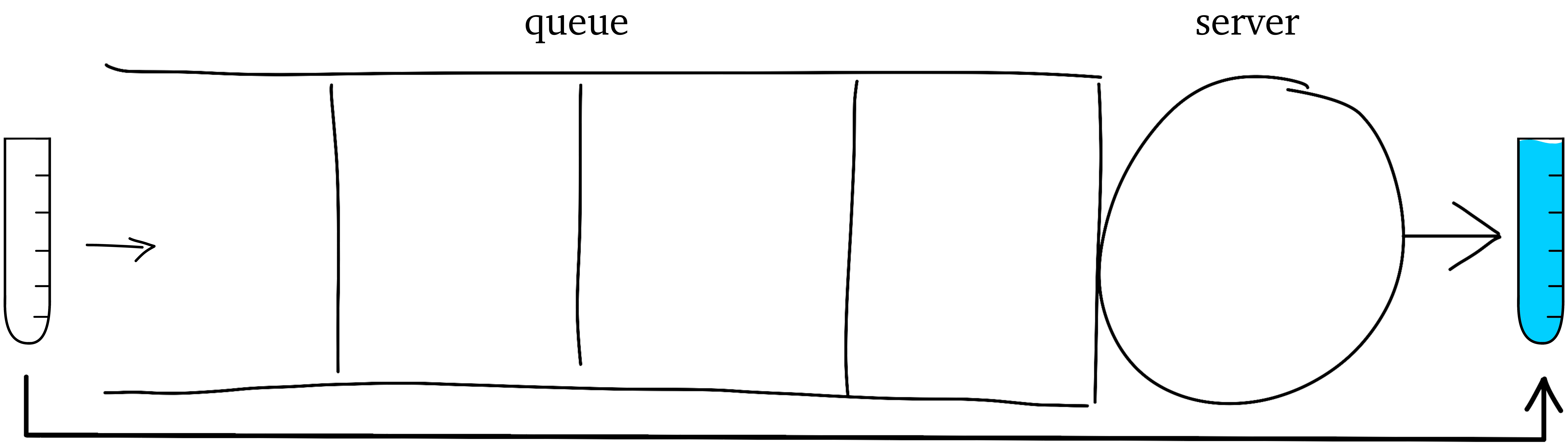
SRPT

Theory

- mean response time, $\mathbf{E}[T]$
- $\mathbf{E}[T]$ with unknown job sizes
- weighted $\mathbf{E}[T]$

Gittins

Practice



T = *response time*

SRPT

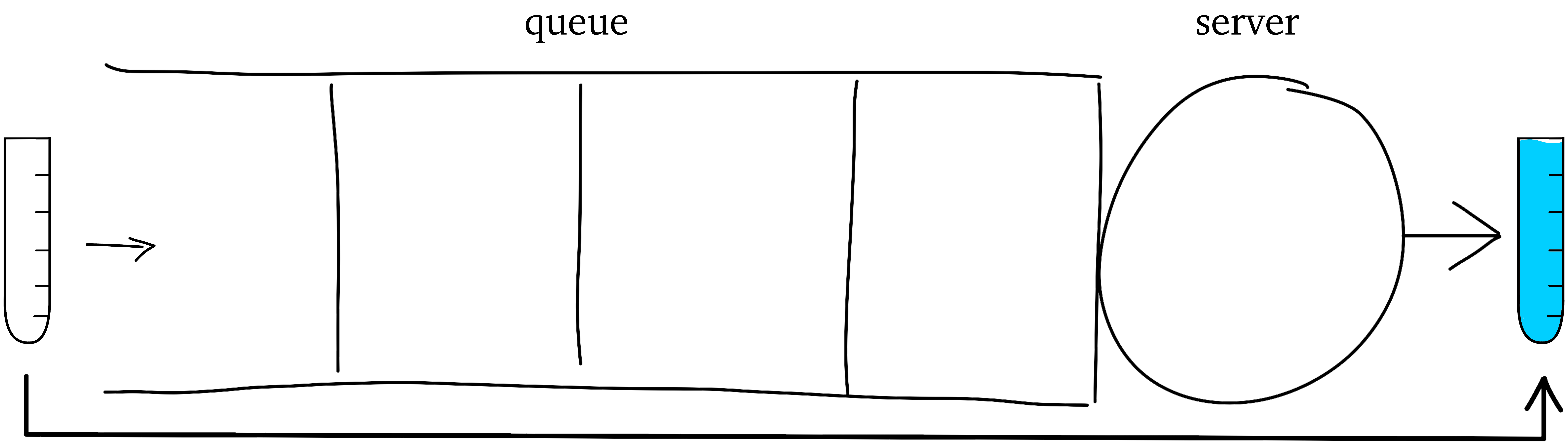
Theory

- mean response time, $\mathbf{E}[T]$
- $\mathbf{E}[T]$ with unknown job sizes
- weighted $\mathbf{E}[T]$

Gittins

Practice

- tail latency, $\mathbf{P}[T > t]$ for large t



T = *response time*

SRPT

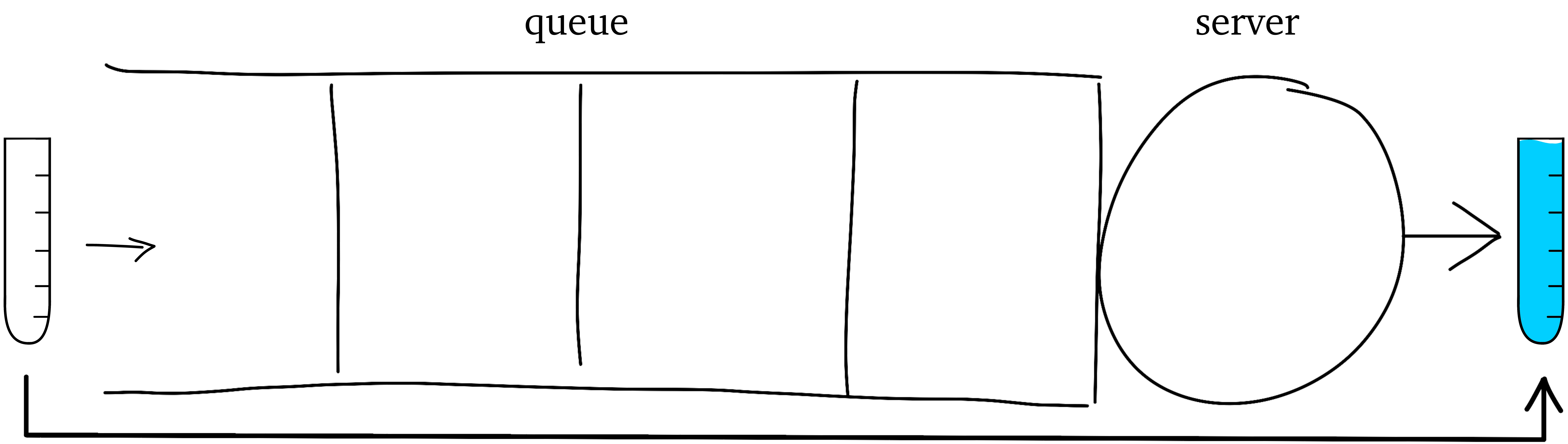
Theory

- mean response time, $\mathbf{E}[T]$
- $\mathbf{E}[T]$ with unknown job sizes
- weighted $\mathbf{E}[T]$

Gittins

Practice

- tail latency, $\mathbf{P}[T > t]$ for large t
- tail latency with unknown job sizes



T = *response time*

SRPT

Theory

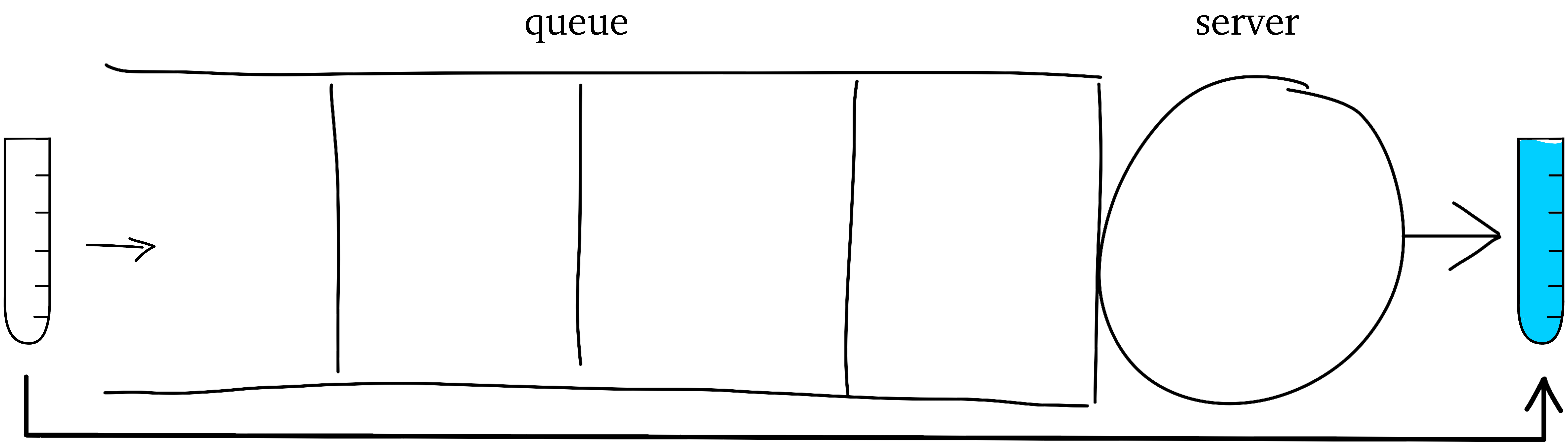
- mean response time, $\mathbf{E}[T]$
- $\mathbf{E}[T]$ with unknown job sizes
- weighted $\mathbf{E}[T]$

Gittins

Practice

hard to analyze

- tail latency, $\mathbf{P}[T > t]$ for large t
- tail latency with unknown job sizes



sponse time

This talk: *asymptotic tail latency*

$\mathbf{P}[T > t]$ as $t \to \infty$

with unknown job sizes!

SRPT

Th

- mean respo
- $\mathbf{E}[T]$ with unknown job sizes
- weighted $\mathbf{E}[T]$

Gittins

Practice

hard to analyze

- tail latency, $\mathbf{P}[T > t]$ for large t
- tail latency with unknown job sizes

What does it mean to minimize asymptotic tail latency?

What does it mean to minimize asymptotic tail latency?

$$K_\pi = \sup_{\pi^*} \lim_{t \to \infty} \frac{\mathbf{P}[T_\pi > t]}{\mathbf{P}[T_{\pi^*} > t]}$$

What does it mean to minimize asymptotic tail latency?

$$K_\pi = \sup_{\pi^*} \lim_{t\to\infty} \frac{\mathbf{P}[T_\pi > t]}{\mathbf{P}[T_{\pi^*} > t]}$$

$K_\pi < \infty$: *weakly optimal*

What does it mean to minimize asymptotic tail latency?

$$K_\pi = \sup_{\pi^*} \lim_{t \to \infty} \frac{\mathbf{P}[T_\pi > t]}{\mathbf{P}[T_{\pi^*} > t]}$$

$K_\pi < \infty$: *weakly optimal*

$K_\pi = 1$: *strongly optimal*

What does it mean to minimize asymptotic tail latency?

$$K_\pi = \sup_{\pi^*} \lim_{t\to\infty} \frac{\mathbf{P}[T_\pi > t]}{\mathbf{P}[T_{\pi^*} > t]}$$

$K_\pi < \infty$: *weakly optimal*

$K_\pi = 1$: *strongly optimal*

Which policies are weakly optimal?

What does it mean to minimize asymptotic tail latency?

$$K_\pi = \sup_{\pi^*} \lim_{t \to \infty} \frac{\mathbf{P}[T_\pi > t]}{\mathbf{P}[T_{\pi^*} > t]}$$

$K_\pi < \infty$: *weakly optimal*

$K_\pi = 1$: *strongly optimal*

Which policies are weakly optimal?

Heavy-Tailed Size Distribution

- PS (Processor Sharing)
- LAS (Least Attained Service)
- SRPT (Shortest Remaining Processing Time)
- PLCFS (Preemptive Last Come First Serve)

What does it mean to minimize asymptotic tail latency?

$$K_\pi = \sup_{\pi^*} \lim_{t\to\infty} \frac{\mathbf{P}[T_\pi > t]}{\mathbf{P}[T_{\pi^*} > t]}$$

$K_\pi < \infty$: *weakly optimal*

$K_\pi = 1$: *strongly optimal*

"$P[S > x] \sim \Omega\left(x^{-\beta}\right)$"

Which policies are weakly optimal?

Heavy-Tailed Size Distribution

- PS (Processor Sharing)
- LAS (Least Attained Service)
- SRPT (Shortest Remaining Processing Time)
- PLCFS (Preemptive Last Come First Serve)

What does it mean to minimize asymptotic tail latency?

$$K_\pi = \sup_{\pi^*} \lim_{t\to\infty} \frac{\mathbf{P}[T_\pi > t]}{\mathbf{P}[T_{\pi^*} > t]}$$

$K_\pi < \infty$: *weakly optimal*

$K_\pi = 1$: *strongly optimal*

“$P[S > x] \sim \Omega\left(x^{-\beta}\right)$”

Which policies are weakly optimal?

“$P[S > x] \sim \mathrm{O}\left(e^{-\beta x}\right)$”

Heavy-Tailed Size Distribution

- PS (Processor Sharing)
- LAS (Least Attained Service)
- SRPT (Shortest Remaining Processing Time)
- PLCFS (Preemptive Last Come First Serve)

Light-Tailed Size Distribution

What does it mean to minimize asymptotic tail latency?

$$K_\pi = \sup_{\pi^*} \lim_{t\to\infty} \frac{\mathbf{P}[T_\pi > t]}{\mathbf{P}[T_{\pi^*} > t]}$$

$K_\pi < \infty$: *weakly optimal*

$K_\pi = 1$: *strongly optimal*

Which policies are weakly optimal?

"$P[S > x] \sim \Omega\left(x^{-\beta}\right)$"

"$P[S > x] \sim \mathrm{O}\left(e^{-\beta x}\right)$"

Heavy-Tailed Size Distribution	Light-Tailed Size Distribution
• PS (Processor Sharing)	• FCFS (First Come First Serve)
• LAS (Least Attained Service)	
• SRPT (Shortest Remaining Processing Time)	
• PLCFS (Preemptive Last Come First Serve)	

What does it mean to minimize asymptotic tail latency?

$$K_\pi = \sup_{\pi^*} \lim_{t\to\infty} \frac{\mathbf{P}[T_\pi > t]}{\mathbf{P}[T_{\pi^*} > t]}$$

$K_\pi < \infty$: *weakly optimal*

$K_\pi = 1$: *strongly optimal*

Which policies are weakly optimal?

"$P[S > x] \sim \Omega\left(x^{-\beta}\right)$"

"$P[S > x] \sim O\left(e^{-\beta x}\right)$"

Heavy-Tailed Size Distribution

- PS (Processor Sharing)
- LAS (Least Attained Service)
- SRPT (Shortest Remaining Processing Time)
- PLCFS (Preemptive Last Come First Serve)

Light-Tailed Size Distribution

- FCFS (First Come First Serve)
- Nudge (Grosof et al. 2021)
- Boost (Yu & Scully 2024)

What does it mean to minimize asymptotic tail latency?

$$K_\pi = \sup_{\pi^*} \lim_{t\to\infty} \frac{\mathbf{P}[T_\pi > t]}{\mathbf{P}[T_{\pi^*} > t]}$$

$K_\pi < \infty$: *weakly optimal*

$K_\pi = 1$: *strongly optimal*

Which policies are weakly optimal?

Heavy-Tailed Size Distribution

strongly optimal:
- PS (Processor Sharing)
- LAS (Least Attained Service)
- SRPT (Shortest Remaining Processing Time)

- PLCFS (Preemptive Last Come First Serve)

Light-Tailed Size Distribution

- FCFS (First Come First Serve)
- Nudge (Grosof et al. 2021)

strongly optimal:
- Boost (Yu & Scully 2024)

What does it mean to minimize asymptotic tail latency?

(without job size information)

$$K_\pi = \sup_{\pi^*} \lim_{t\to\infty} \frac{\mathbf{P}[T_\pi > t]}{\mathbf{P}[T_{\pi^*} > t]}$$

$K_\pi < \infty$: *weakly optimal*

$K_\pi = 1$: *strongly optimal*

Which policies are weakly optimal?

Heavy-Tailed Size Distribution

strongly optimal:

- PS (Processor Sharing)
- LAS (Least Attained Service)
- SRPT (Shortest Remaining Processing Time)

- PLCFS (Preemptive Last Come First Serve)

Light-Tailed Size Distribution

- FCFS (First Come First Serve)
- Nudge (Grosof et al. 2021)

strongly optimal:

- Boost (Yu & Scully 2024)

What does it mean to minimize asymptotic tail latency?

(without job size information)

$$K_\pi = \sup_{\pi^* \in \text{UnknownSize}} \lim_{t \to \infty} \frac{\mathbf{P}[T_\pi > t]}{\mathbf{P}[T_{\pi^*} > t]}$$

$K_\pi < \infty$: *weakly optimal*

$K_\pi = 1$: *strongly optimal*

Which policies are weakly optimal?

Heavy-Tailed Size Distribution

strongly optimal:
- PS (Processor Sharing)
- LAS (Least Attained Service)
- SRPT (Shortest Remaining Processing Time)

- PLCFS (Preemptive Last Come First Serve)

Light-Tailed Size Distribution

- FCFS (First Come First Serve)
- Nudge (Grosof et al. 2021)

strongly optimal:
- Boost (Yu & Scully 2024)

What does it mean to minimize asymptotic tail latency?

(without job size information)

$$K_\pi = \sup_{\pi^* \in \text{UnknownSize}} \lim_{t\to\infty} \frac{\mathbf{P}[T_\pi > t]}{\mathbf{P}[T_{\pi^*} > t]}$$

- $K_\pi < \infty$: *weakly optimal*
- $K_\pi = 1$: *strongly optimal*

Which policies are weakly optimal?

Heavy-Tailed Size Distribution

strongly optimal:
- PS (Processor Sharing)
- LAS (Least Attained Service)
- ~~SRPT (Shortest Remaining Processing Time)~~

- PLCFS (Preemptive Last Come First Serve)

Light-Tailed Size Distribution

- FCFS (First Come First Serve)

~~strongly optimal~~:
- ~~Nudge~~ (Grosof et al. 2021)
- ~~Boost~~ (Yu & Scully 2024)

Our contribution: new policy + proof of strong optimality

GittinsBoost

Our contribution: GittinsBoost + proof of strong optimality

How do we generalize Boost to the unknown size setting

	known sizes	unknown sizes
$\lim_{t\to\infty} \mathbf{P}[T > t]$ (light-tailed)	**Boost** →	**GittinsBoost**

Our contribution: GittinsBoost + proof of strong optimality

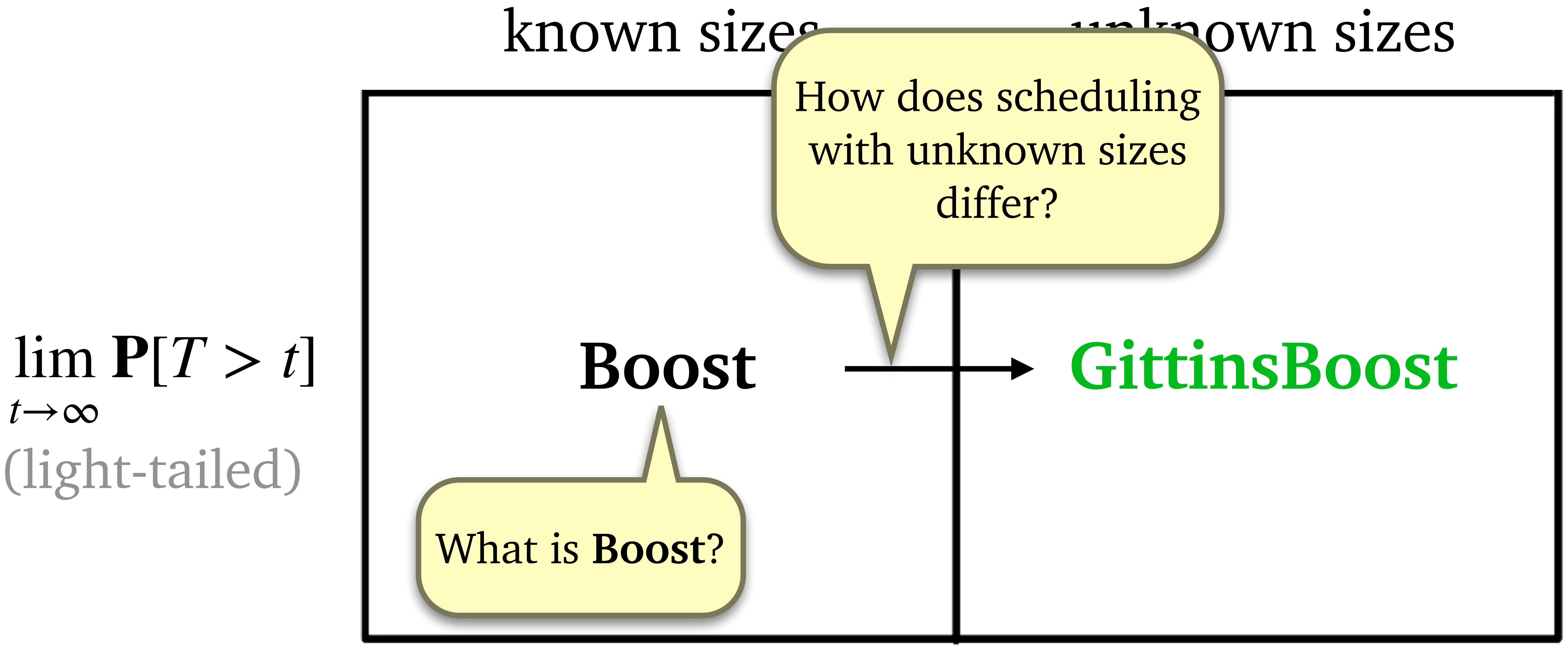
? How do we generalize Boost to the unknown size setting

	known sizes		unknown sizes
$\lim_{t \to \infty} \mathbf{P}[T > t]$ (light-tailed)	**Boost**	→	**GittinsBoost**

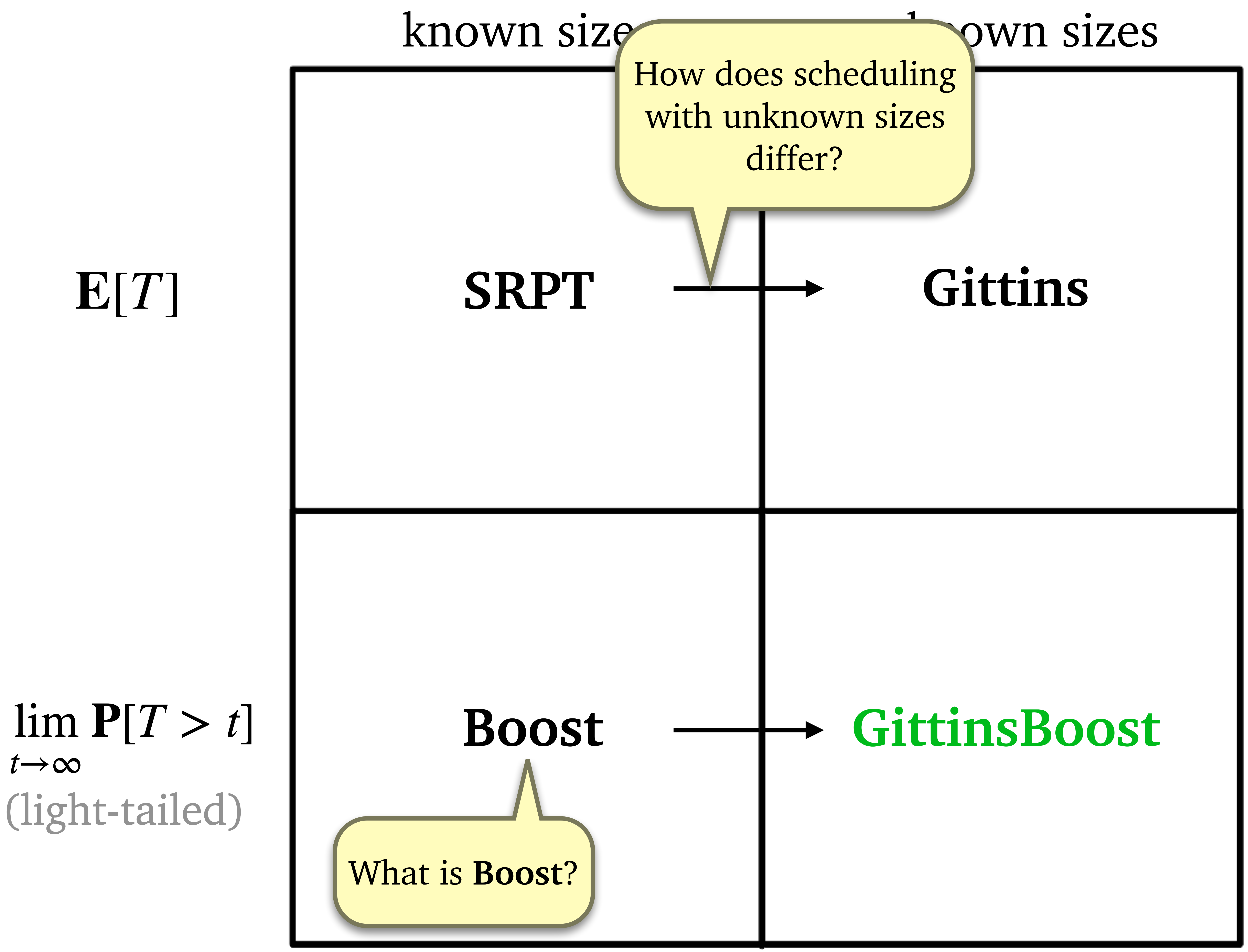
What is **Boost**?

Our contribution: GittinsBoost + proof of strong optimality

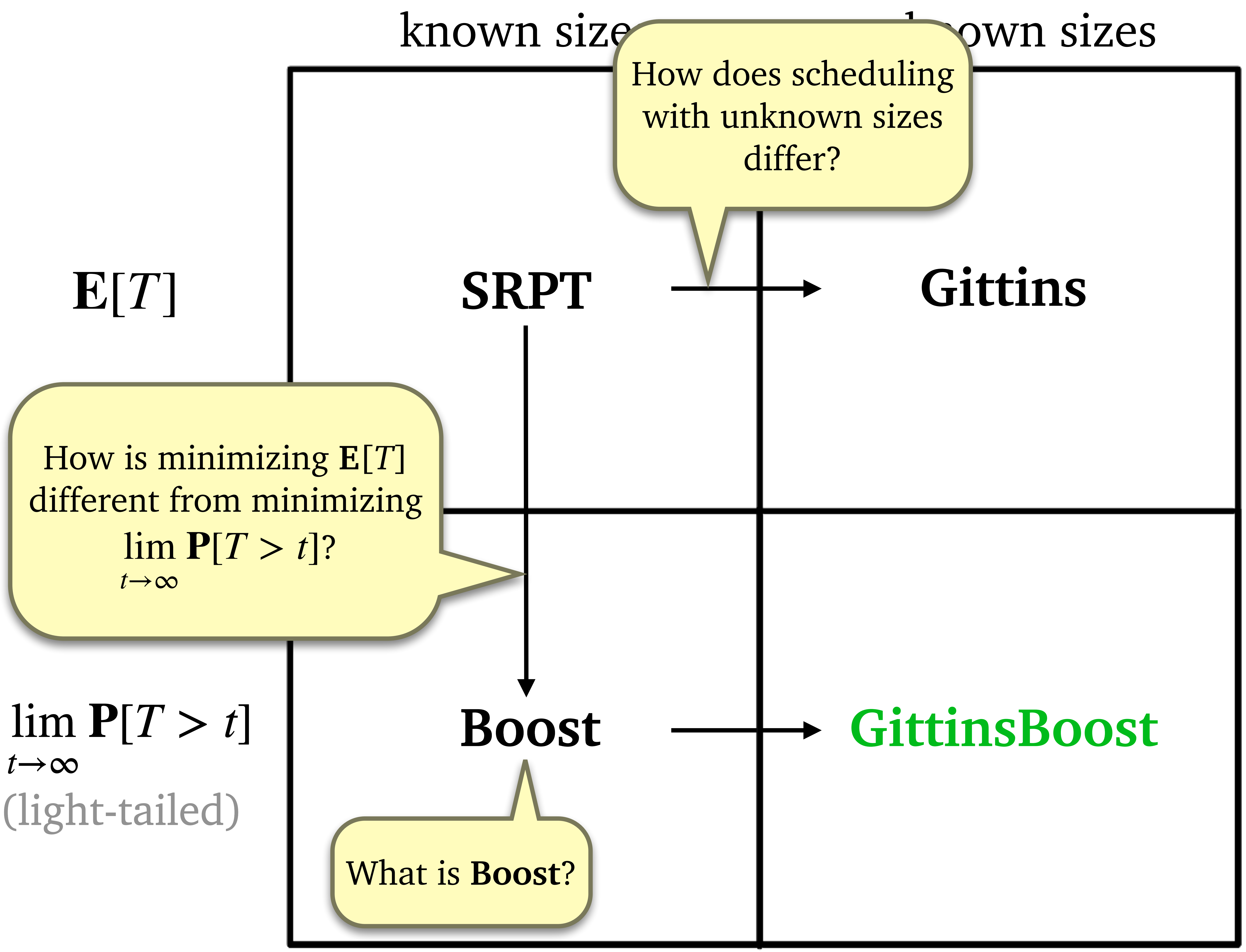
How do we generalize Boost to the unknown size setting



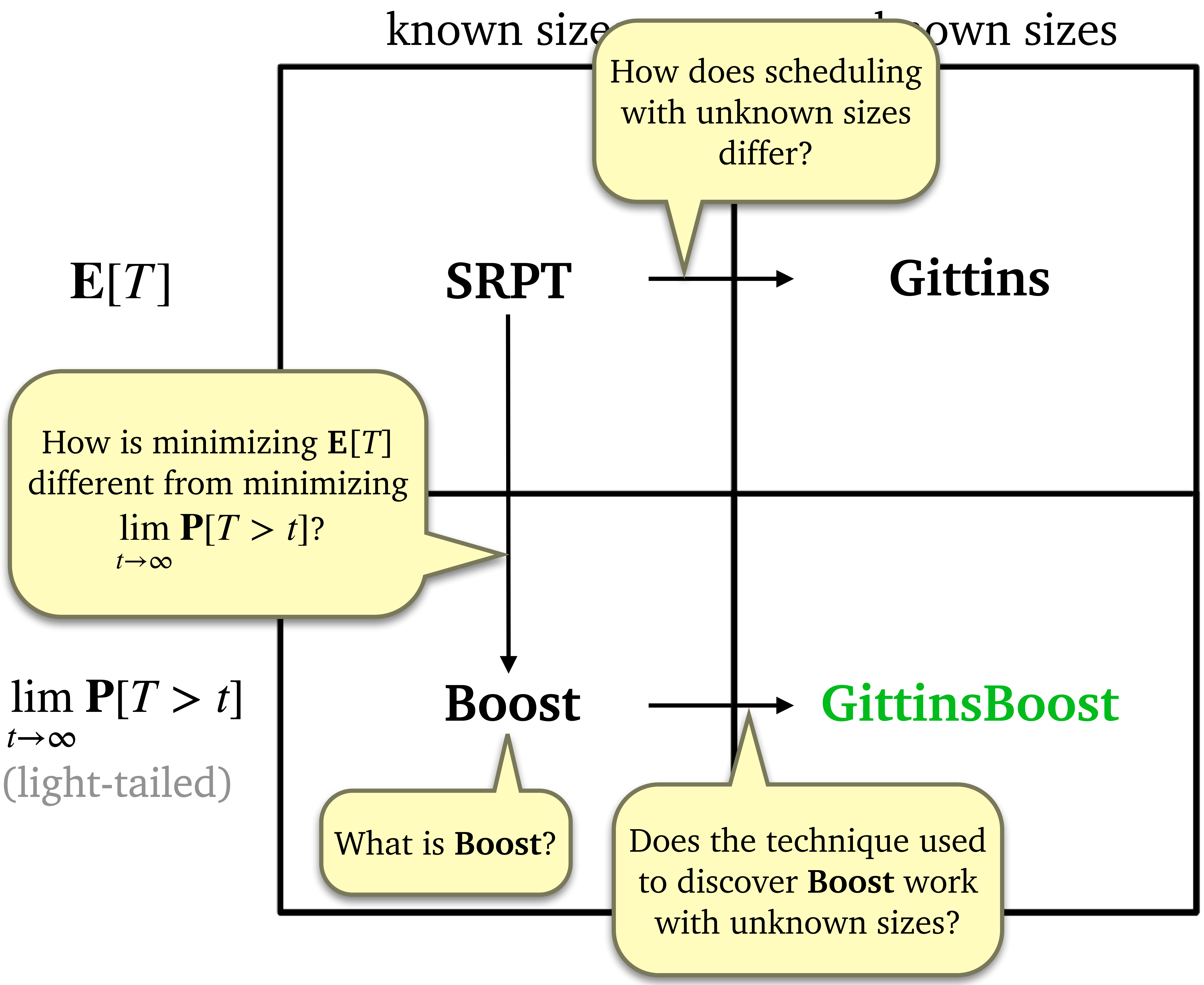
Our contribution: GittinsBoost + proof of strong optimality



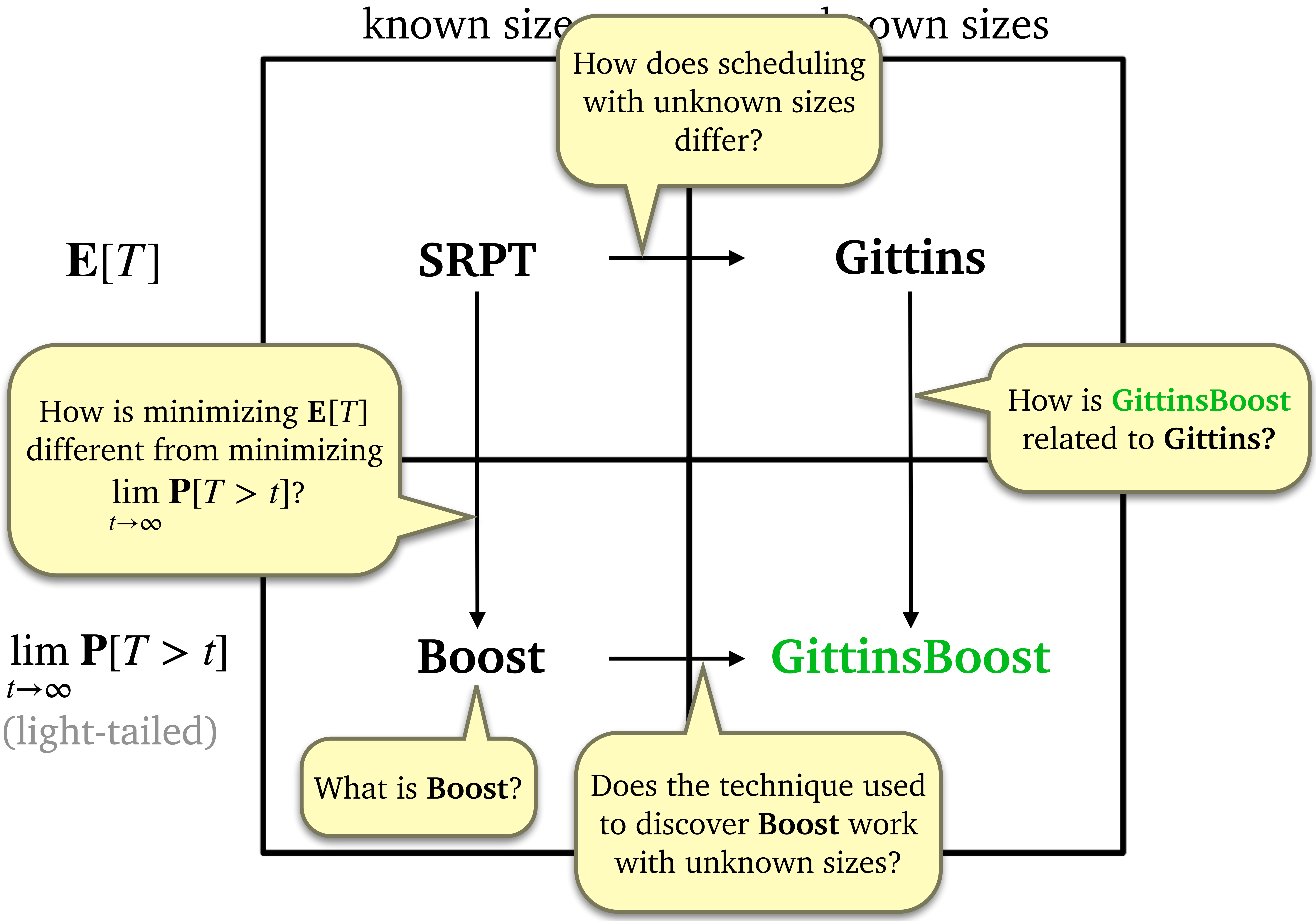
Our contribution: GittinsBoost + proof of strong optimality



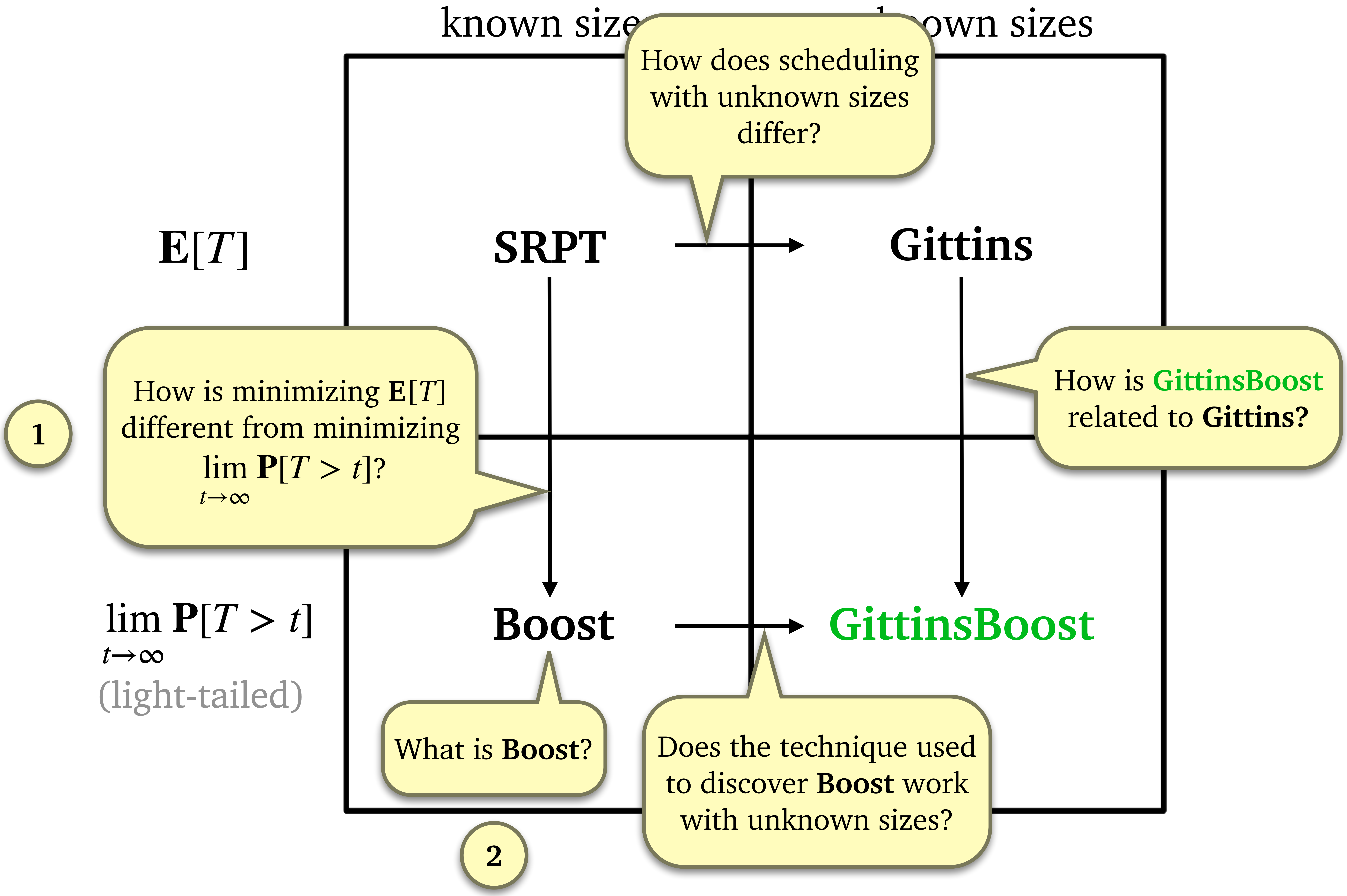
Our contribution: GittinsBoost + proof of strong optimality



Our contribution: GittinsBoost + proof of strong optimality



Our contribution: GittinsBoost + proof of strong optimality



Optimizing Means vs Optimizing Tails

Minimize $\mathbf{E}[T]$

Minimize $\mathbf{P}[T>t]$

Optimizing Means vs Optimizing Tails

Minimize $\mathbf{E}[T]$

- Don't want small jobs stuck behind large job

Minimize $\mathbf{P}[T>t]$

Optimizing Means vs Optimizing Tails

Minimize $\mathbf{E}[T]$

- Don't want small jobs stuck behind large job

Minimize $\mathbf{P}[T>t]$

- Don't want small jobs stuck behind large job

Optimizing Means vs Optimizing Tails

Minimize $\mathbf{E}[T]$

- Don't want small jobs stuck behind large job

Minimize $\mathbf{P}[T>t]$

- Don't want small jobs stuck behind large job
- Don't want large jobs to be overly delayed

Optimizing Means vs Optimizing Tails

Minimize $\mathbf{E}[T]$

- Don't want small jobs stuck behind large job

Minimize $\mathbf{P}[T>t]$

- Don't want small jobs stuck behind large job
- Don't want large jobs to be overly delayed

Heavy Tails: basically doesn't matter

Optimizing Means vs Optimizing Tails

Minimize $\mathbf{E}[T]$

- Don't want small jobs stuck behind large job

Heavy Tails: basically doesn't matter

Minimize $\mathbf{P}[T>t]$

- Don't want small jobs stuck behind large job
- Don't want large jobs to be overly delayed

Light Tails: very important

Optimizing Means vs Optimizing Tails

Minimize $\mathbf{E}[T]$

- Don't want small jobs stuck behind large job

Minimize $\mathbf{P}[T>t]$

- Don't want small jobs stuck behind large job
- Don't want large jobs to be overly delayed

Heavy Tails: basically doesn't matter

Light Tails: very important

Boost: a way to balance this tradeoff!

What is Boost?

Boost serves jobs in order of ascending *boosted arrival time:*

boosted arrival time = arrival time - boost

What is Boost?

Boost serves jobs in order of ascending *boosted arrival time:*

boosted arrival time = arrival time - boost

boost is determined by *boost function* $b(s) \geq 0$ that maps job size to boost.

What is Boost?

Boost serves jobs in order of ascending *boosted arrival time:*

boosted arrival time = arrival time - boost

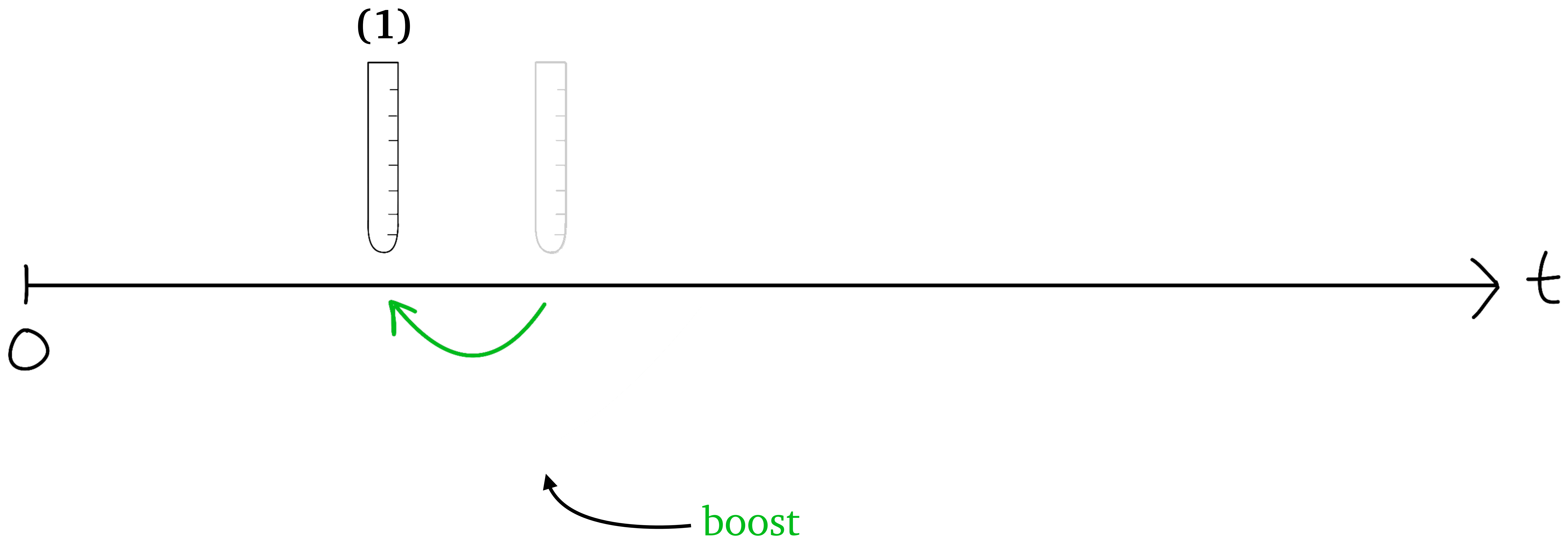
boost is determined by *boost function* $b(s) \geq 0$ that maps job size to boost.

What is Boost?

Boost serves jobs in order of ascending *boosted arrival time:*

boosted arrival time = arrival time - boost

boost is determined by *boost function* $b(s) \geq 0$ that maps job size to boost.

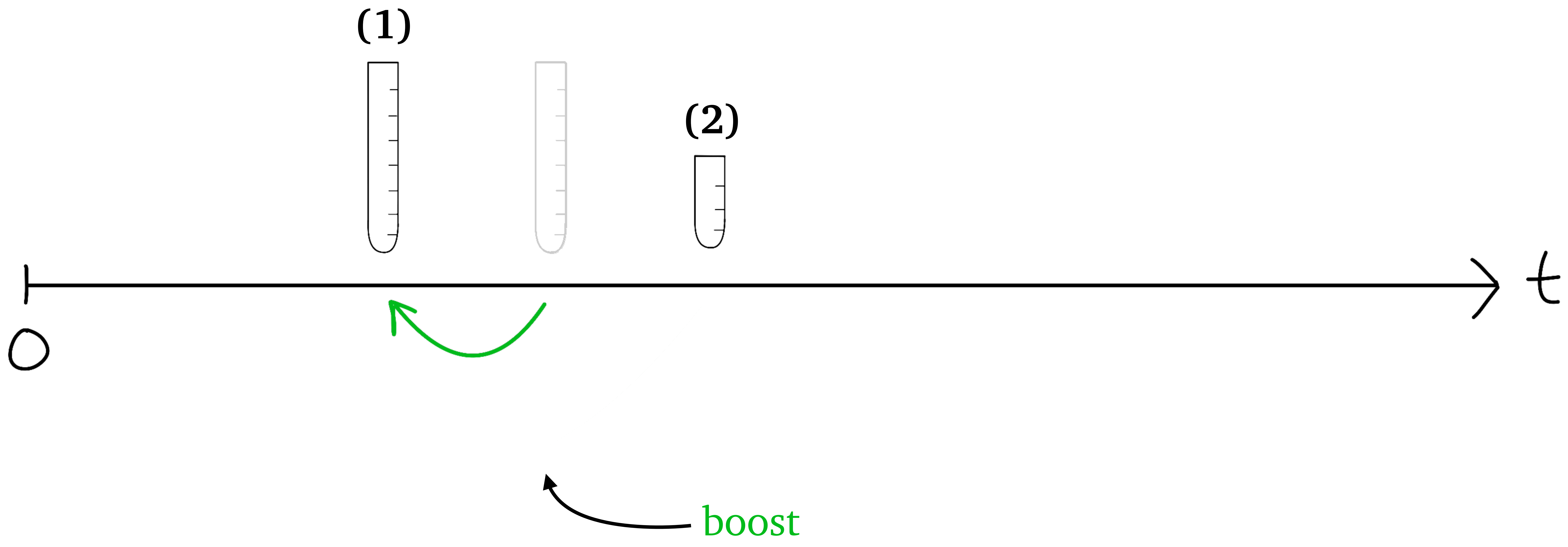


What is Boost?

Boost serves jobs in order of ascending *boosted arrival time:*

boosted arrival time = arrival time - boost

boost is determined by *boost function* $b(s) \geq 0$ that maps job size to boost.

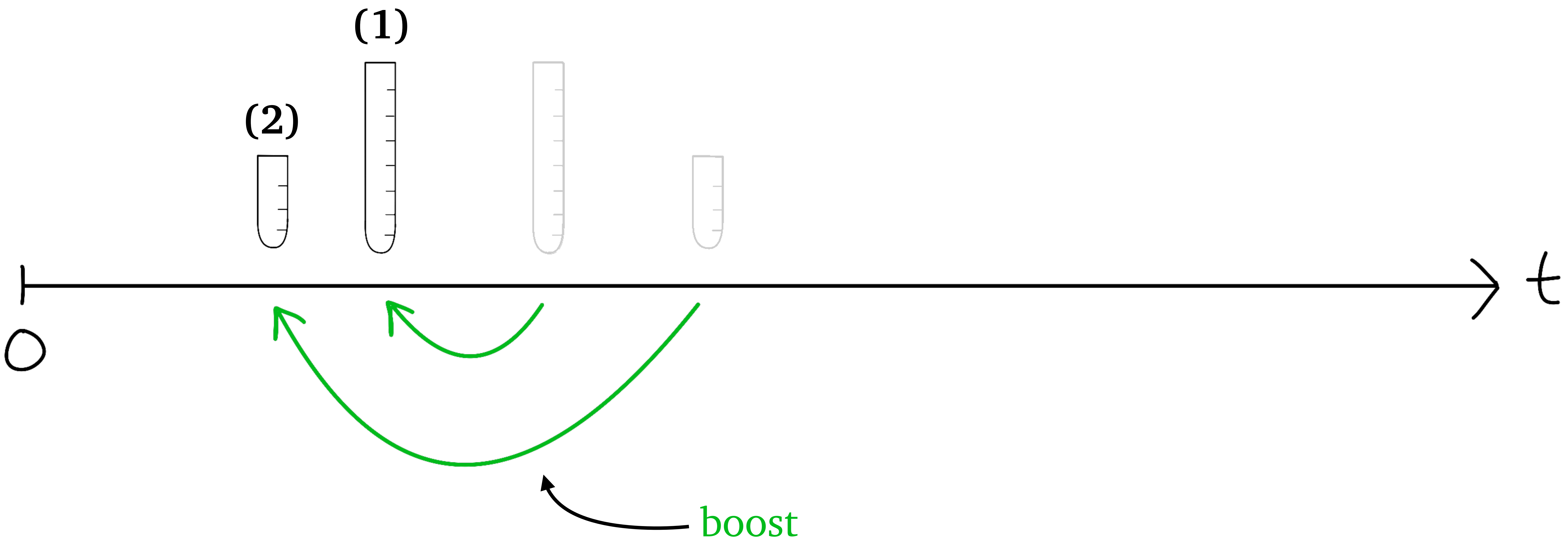


What is Boost?

Boost serves jobs in order of ascending *boosted arrival time:*

boosted arrival time = arrival time - boost

boost is determined by *boost function* $b(s) \geq 0$ that maps job size to boost.

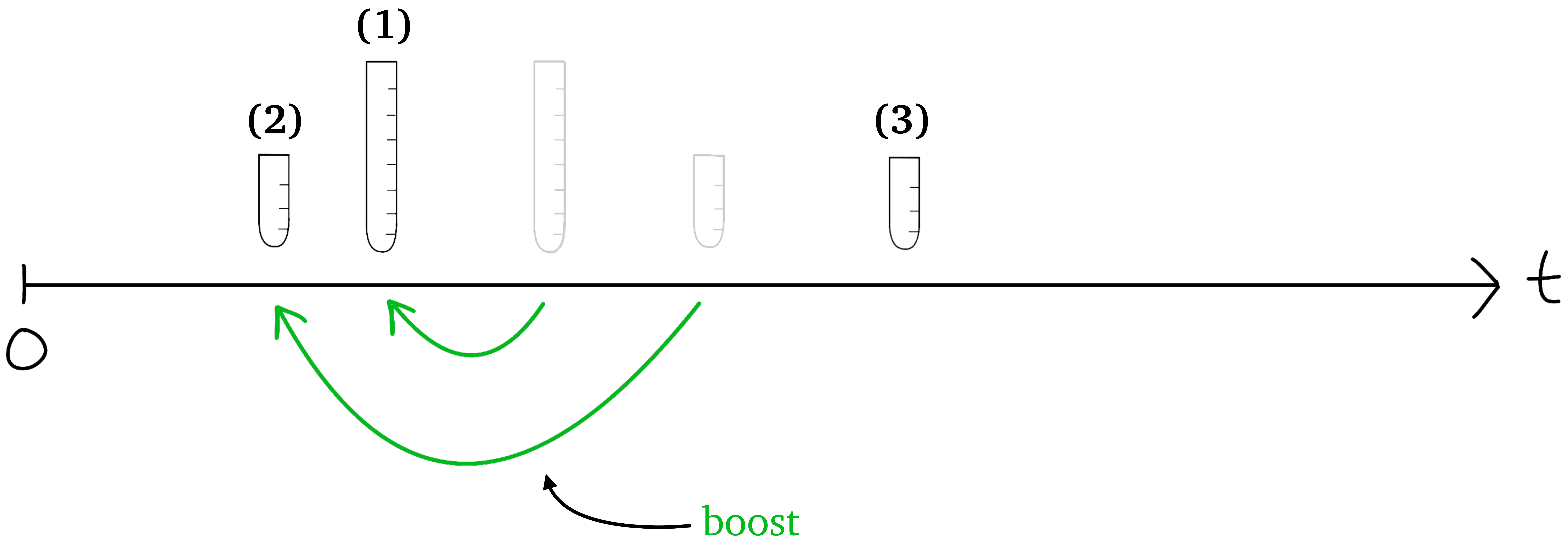


What is Boost?

Boost serves jobs in order of ascending *boosted arrival time:*

boosted arrival time = arrival time - boost

boost is determined by *boost function* $b(s) \geq 0$ that maps job size to boost.

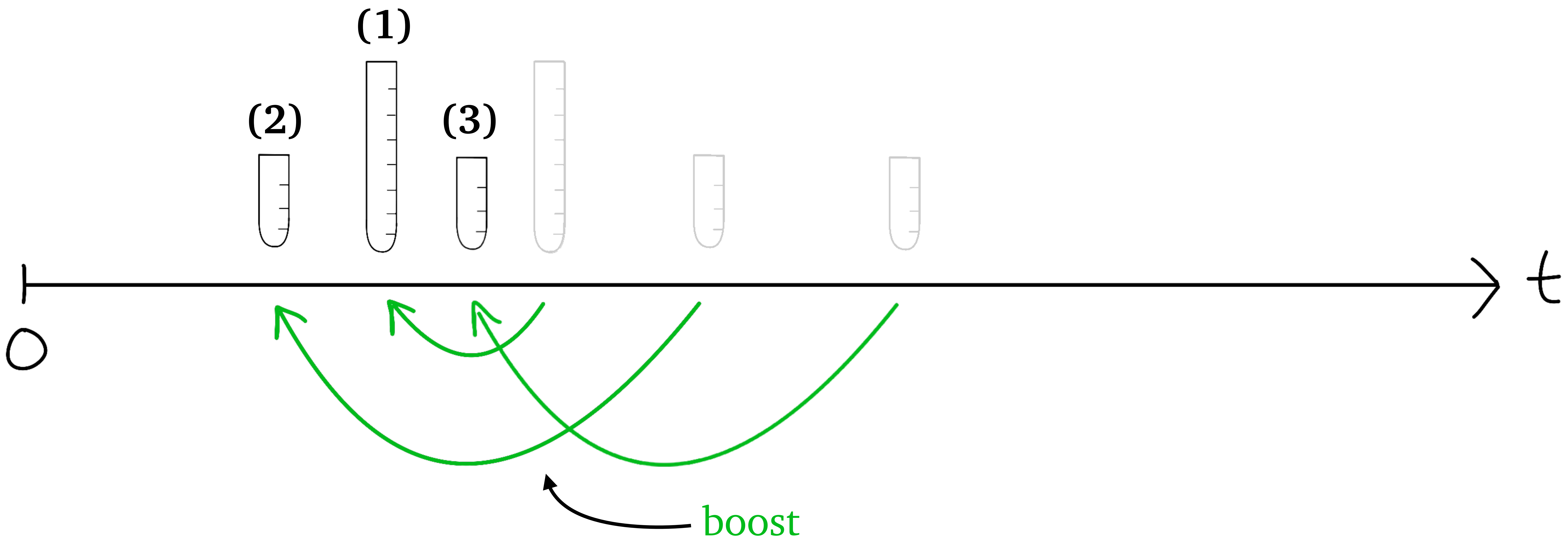


What is Boost?

Boost serves jobs in order of ascending *boosted arrival time:*

boosted arrival time = arrival time - boost

boost is determined by *boost function* $b(s) \geq 0$ that maps job size to boost.



What is Boost?

Boost serves jobs in order of ascending *boosted arrival time:*

boosted arrival time = arrival time - boost

boost is determined by *boost function* $b(s) \geq 0$ that maps job size to boost.

Which boost function minimizes asymptotic tail latency?

What is Boost?

Boost serves jobs in order of ascending *boosted arrival time:*

boosted arrival time = arrival time - boost

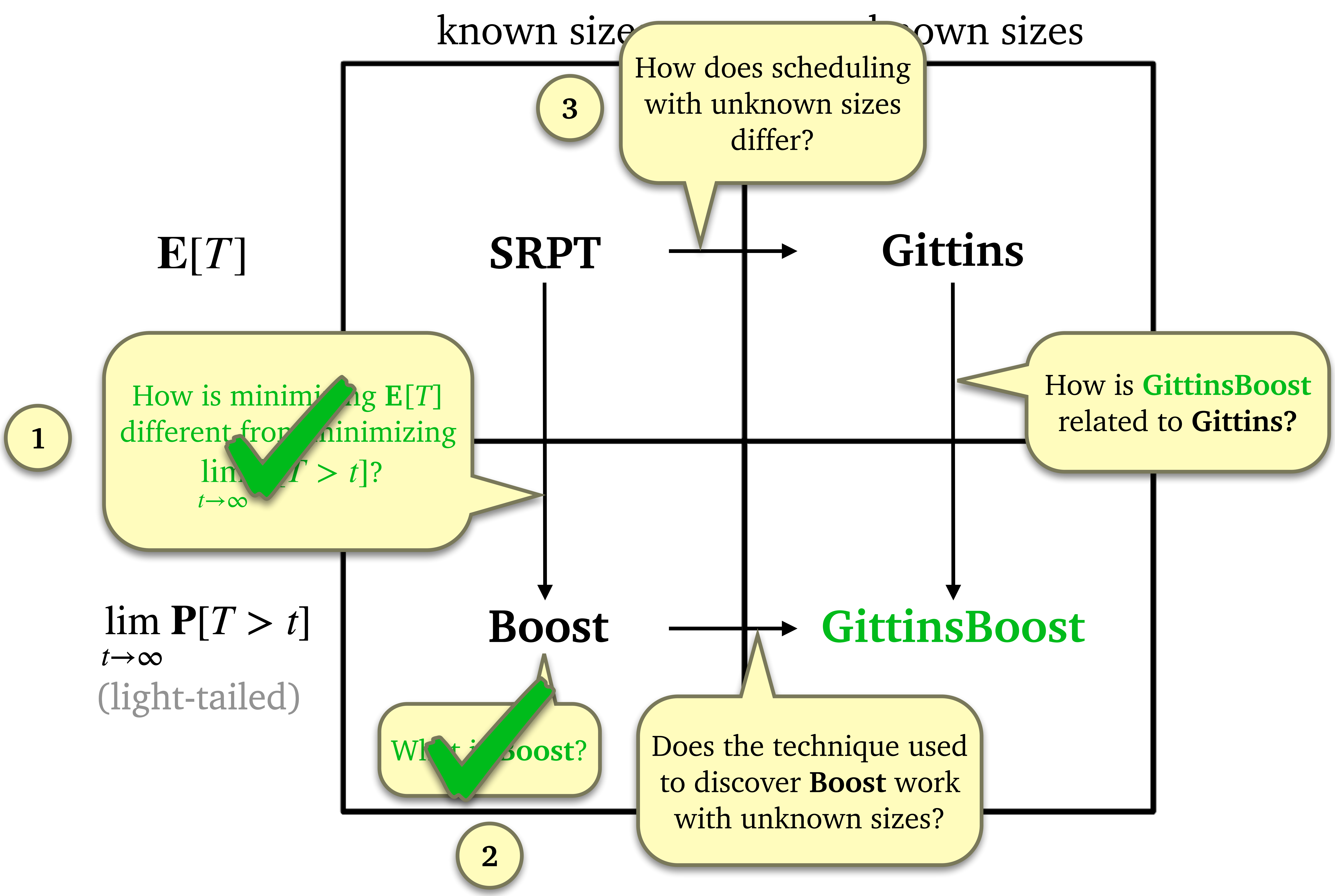
boost is determined by *boost function* $b(s) \geq 0$ that maps job size to boost.

Which boost function minimizes asymptotic tail latency?

choosing:

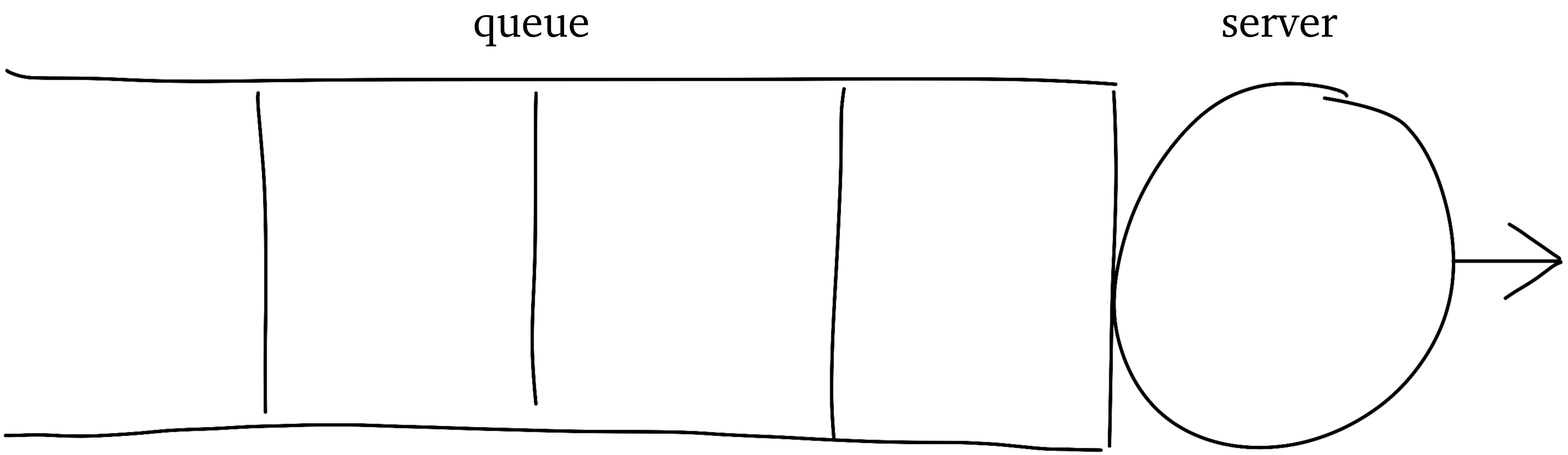
$$b(s) = \frac{1}{\gamma} \log \frac{1}{1 - e^{-\gamma s}}$$

results in strongly optimal policy (Yu & Scully, 2024)

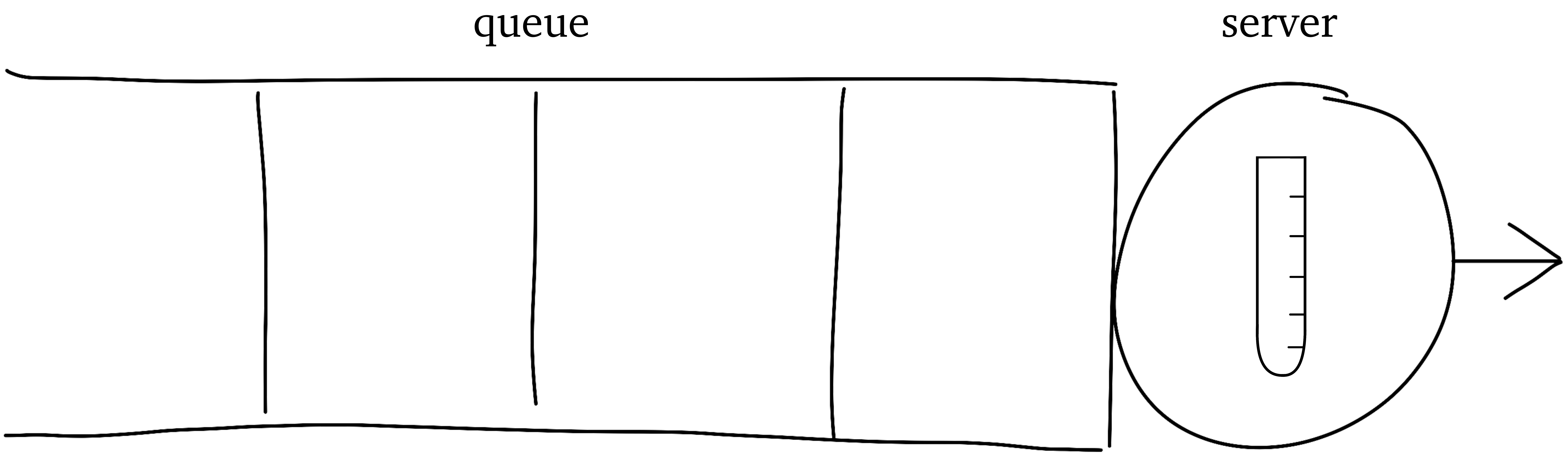


How does scheduling with unknown sizes differ?
3
E[T]
SRPT
Gittins
1
How is GittinsBoost related to Gittins?
lim P[T > t]
t→∞
(light-tailed)
Boost
GittinsBoost
Does the technique used to discover Boost work with unknown sizes?
2

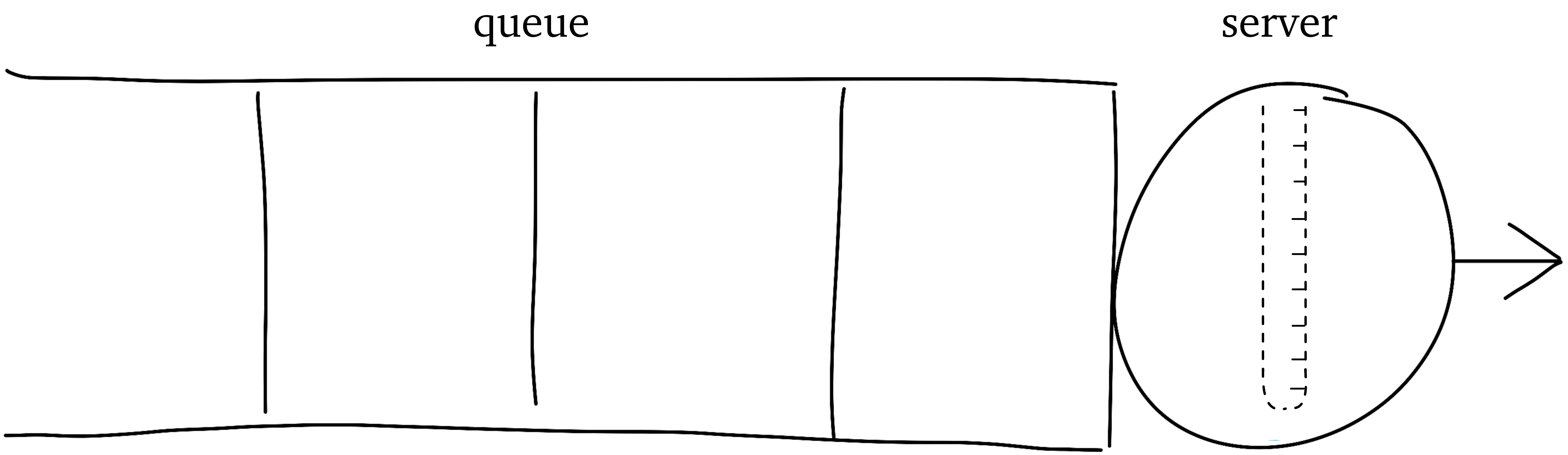
Scheduling with unknown job sizes



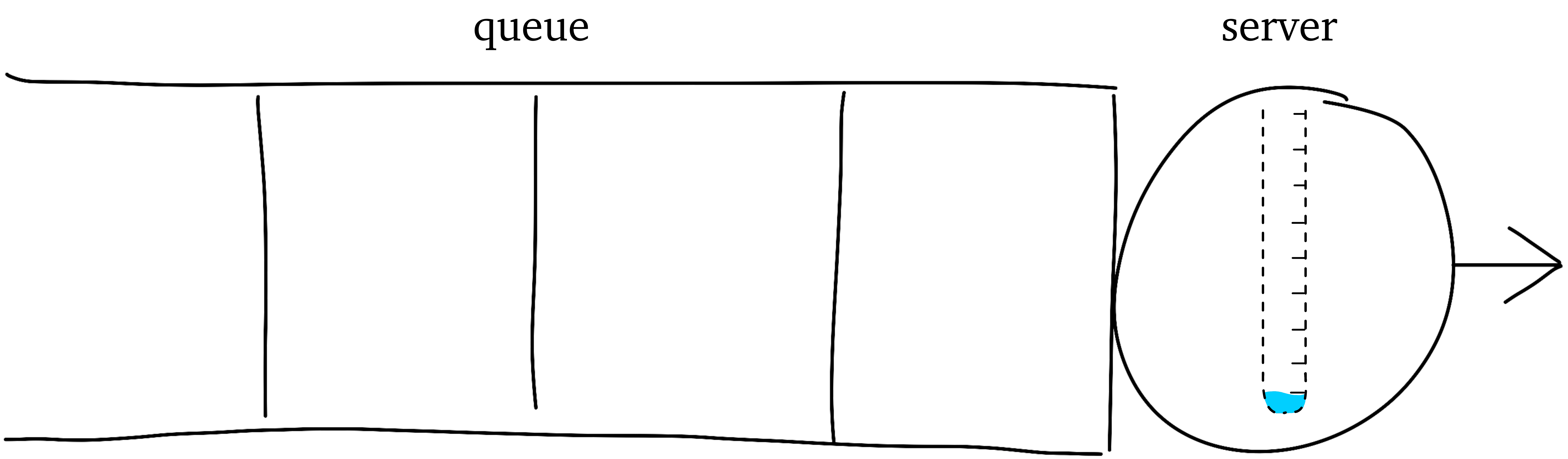
Scheduling with unknown job sizes



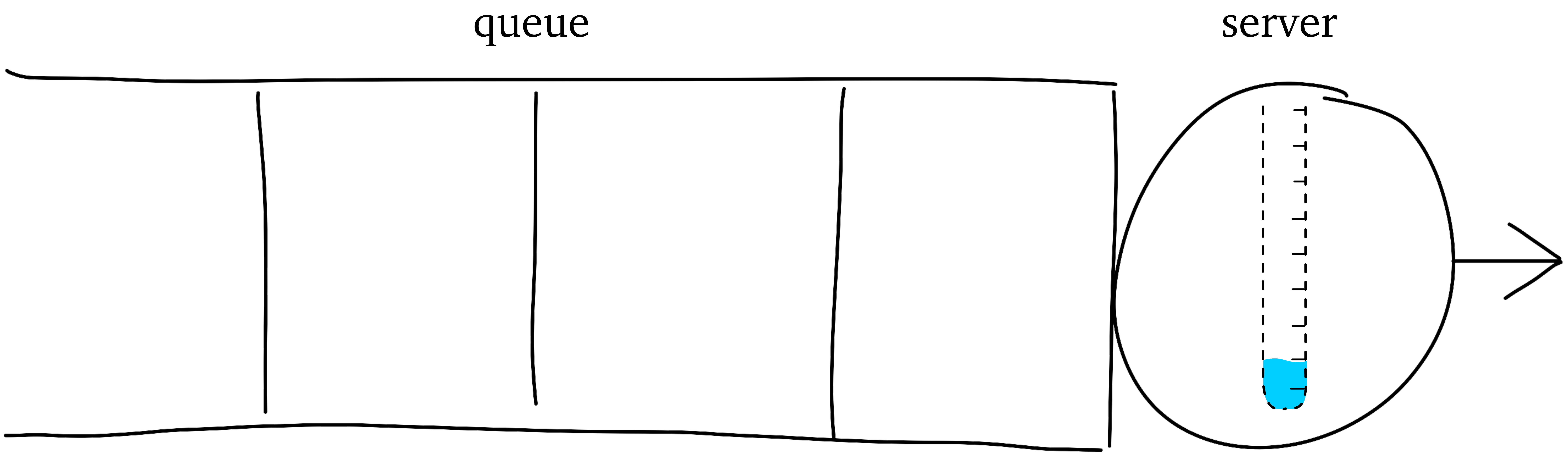
Scheduling with unknown job sizes



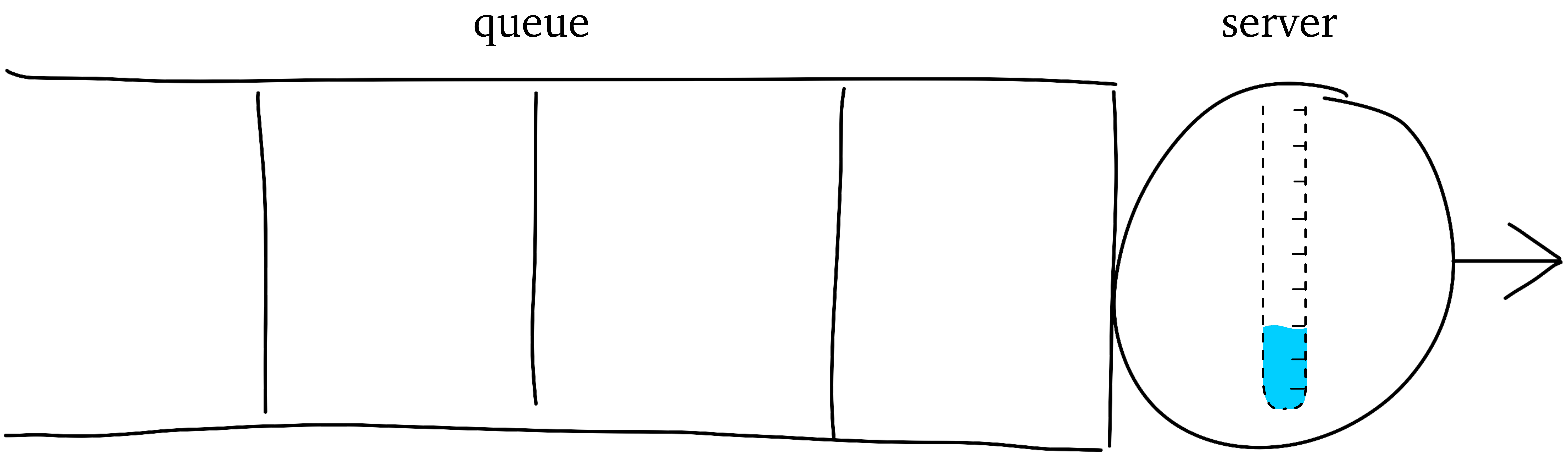
Scheduling with unknown job sizes



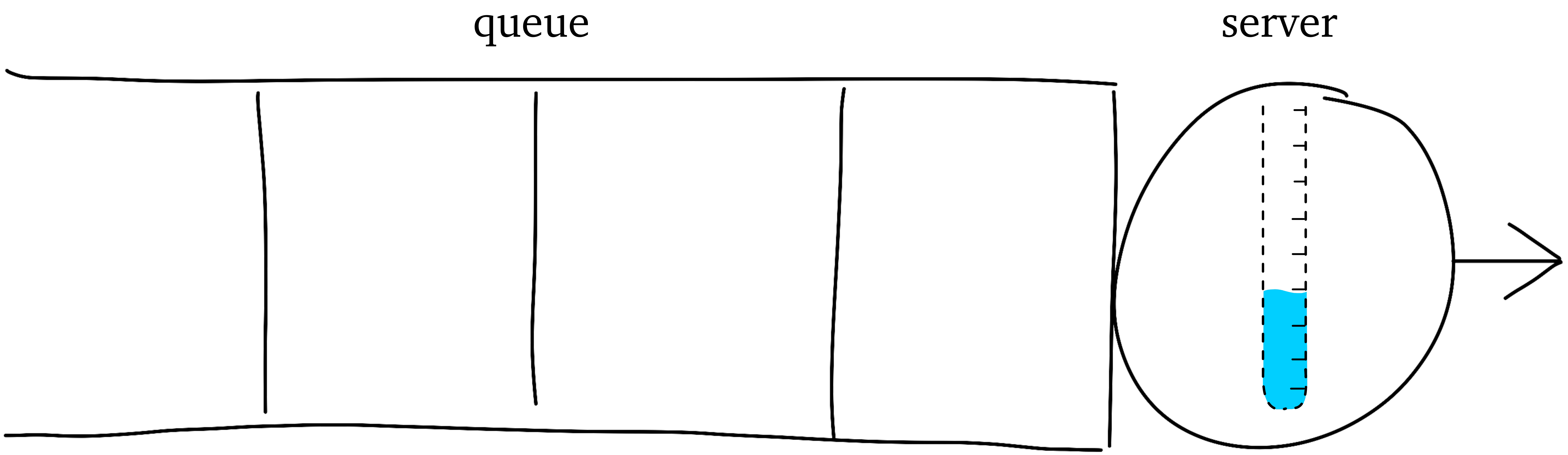
Scheduling with unknown job sizes



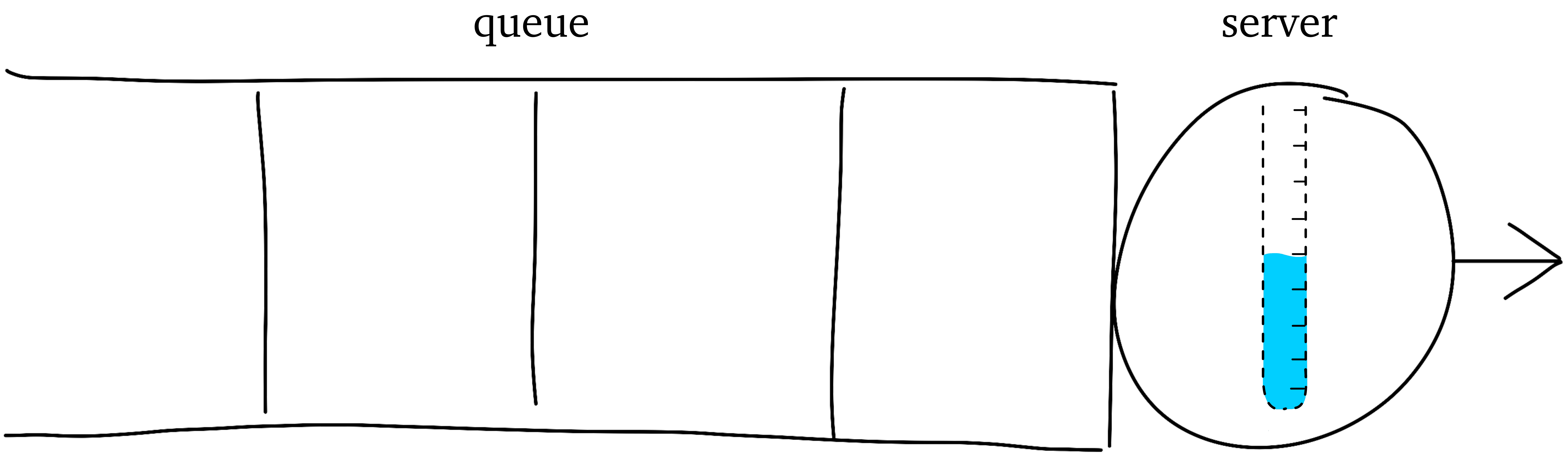
Scheduling with unknown job sizes



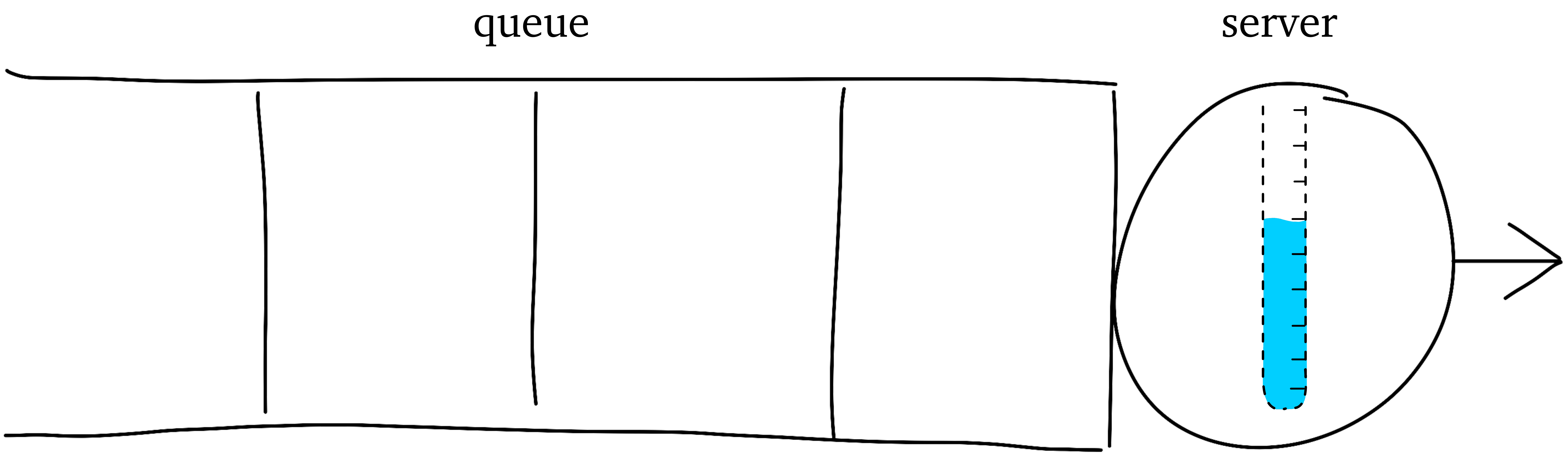
Scheduling with unknown job sizes



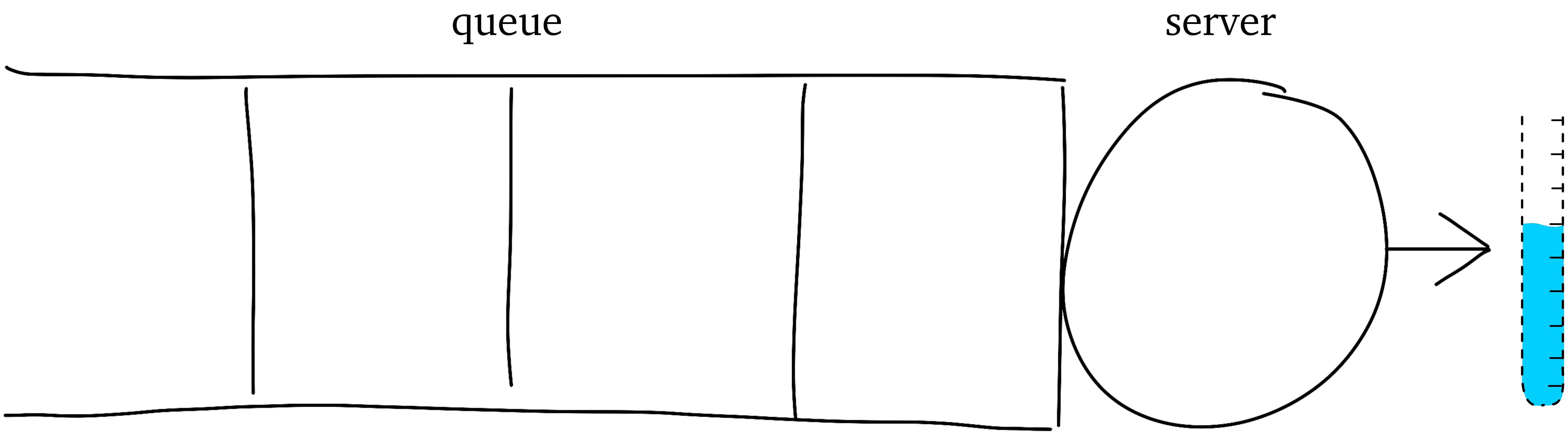
Scheduling with unknown job sizes



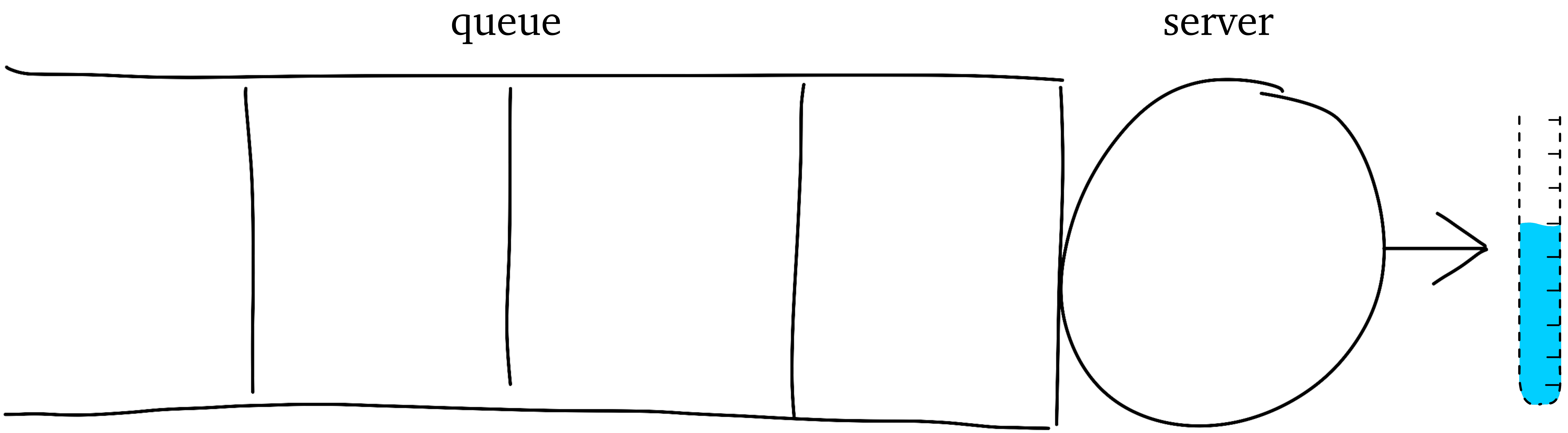
Scheduling with unknown job sizes

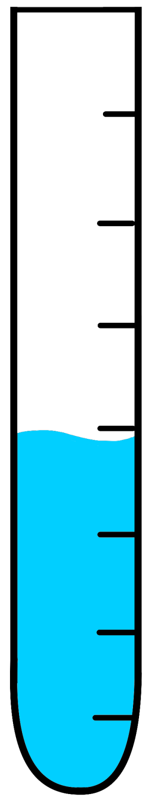


Scheduling with unknown job sizes



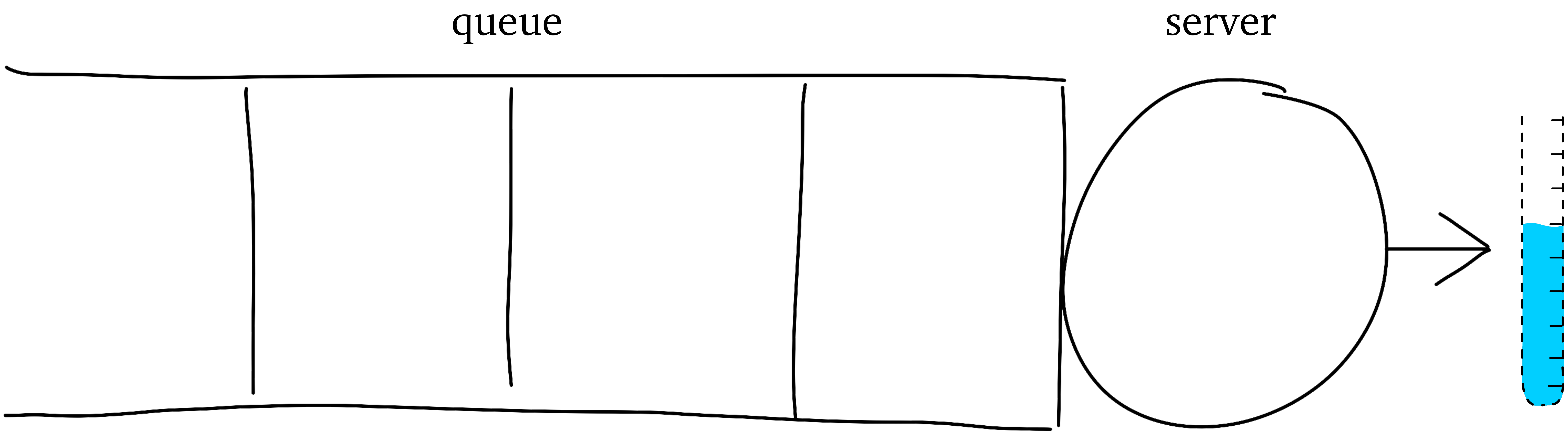
Scheduling with unknown job sizes

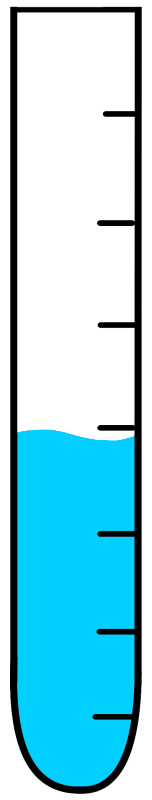




state = (size, age, arrival time)

Scheduling with unknown job sizes

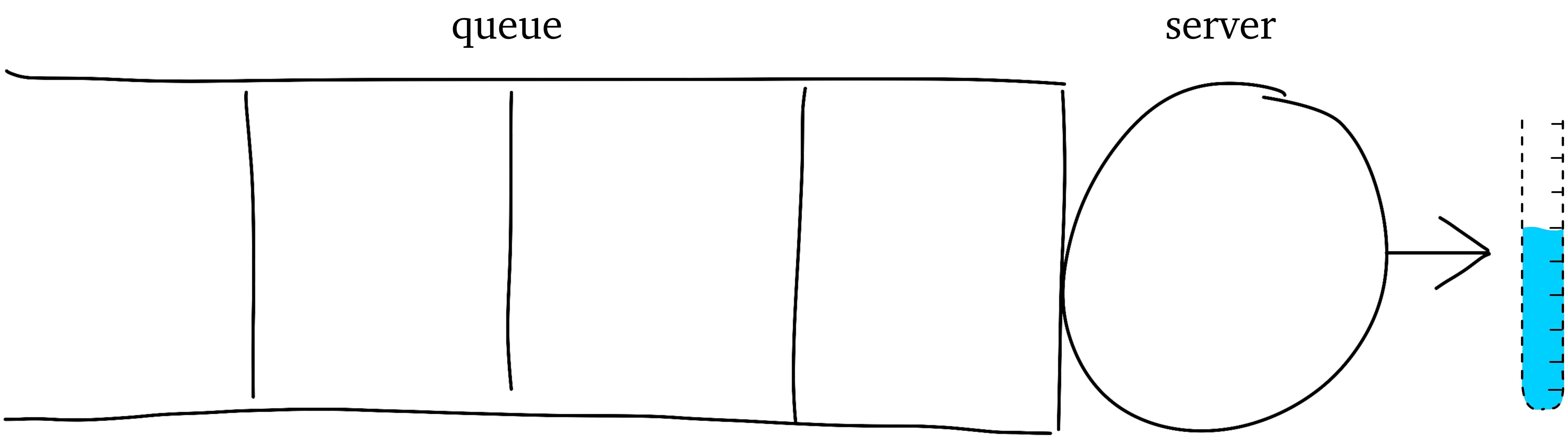


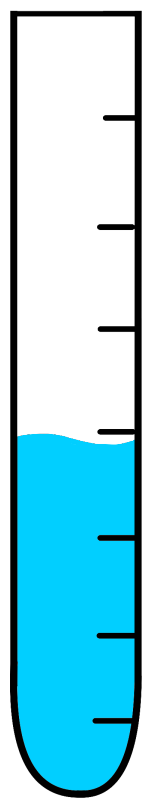


state = (size, age, arrival time)

SRPT: order by [size - age]

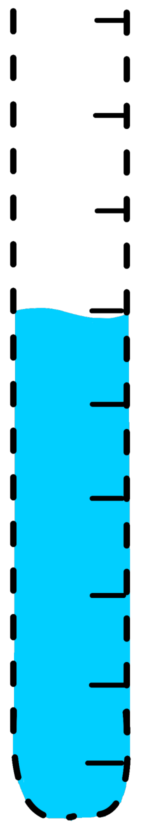
Scheduling with unknown job sizes





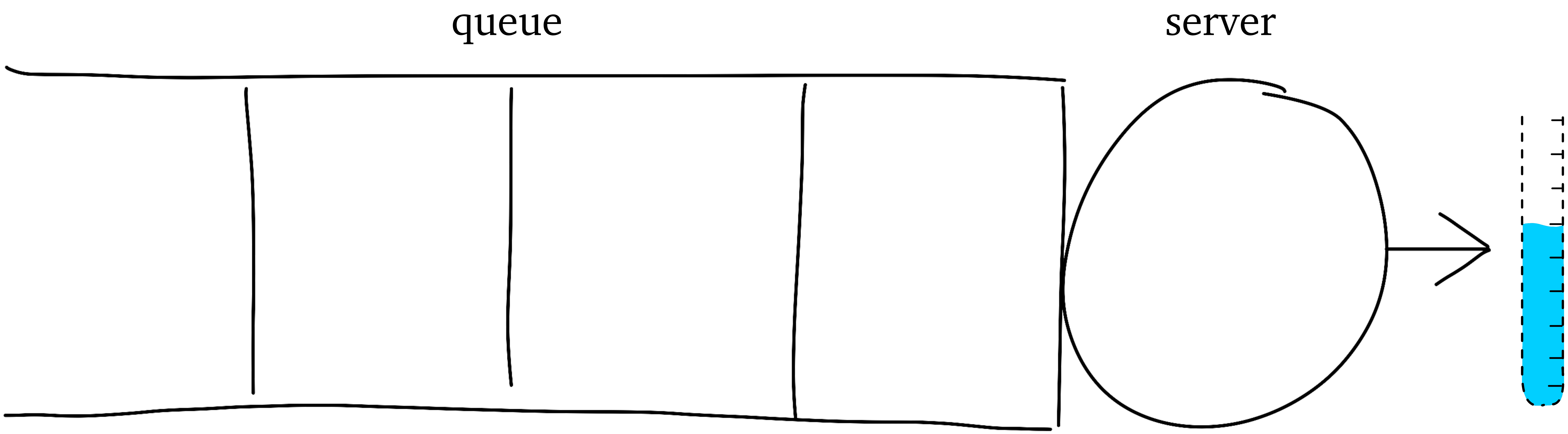
state = (size, age, arrival time)

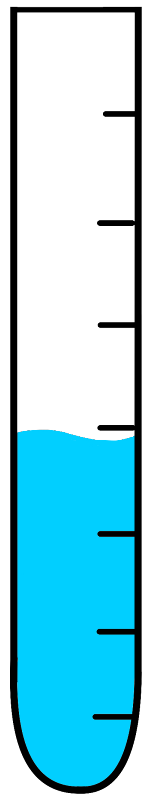
SRPT: order by [size - age]



state = (age, arrival time)

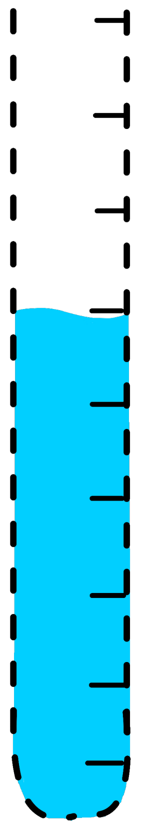
Scheduling with unknown job sizes





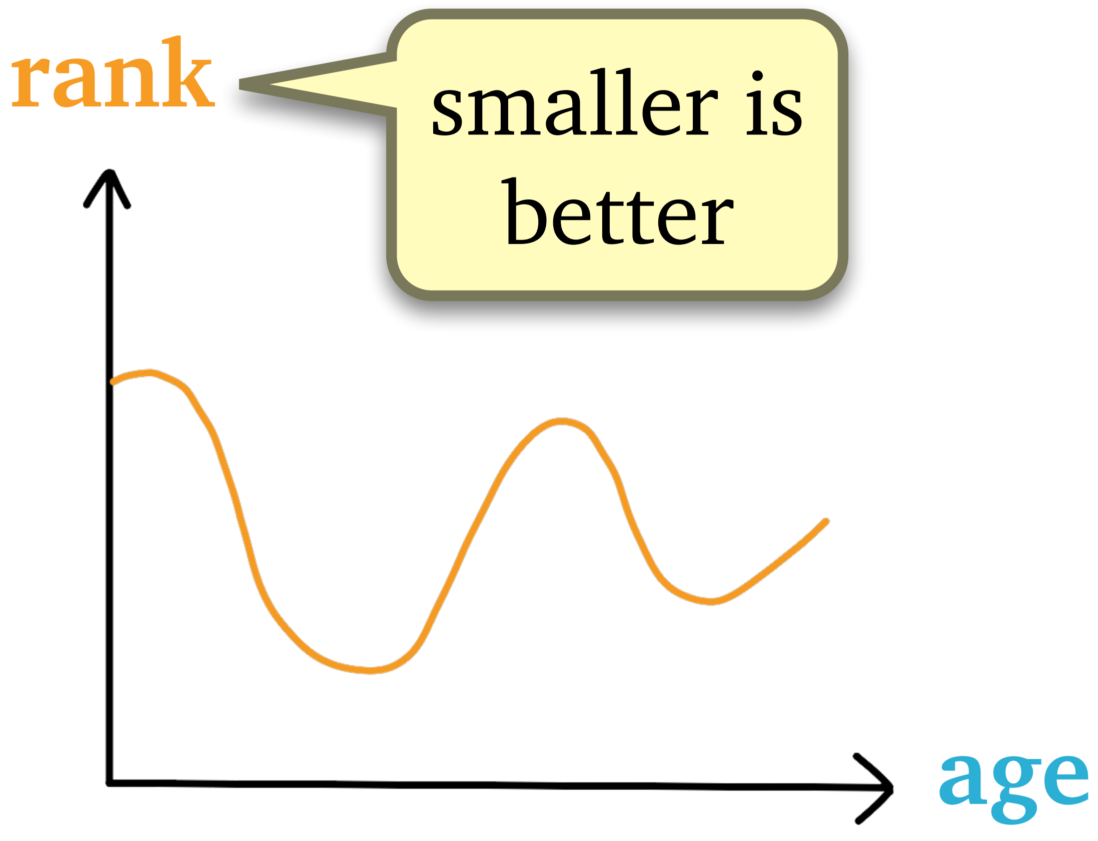
state = (size, age, arrival time)

SRPT: order by [size - age]

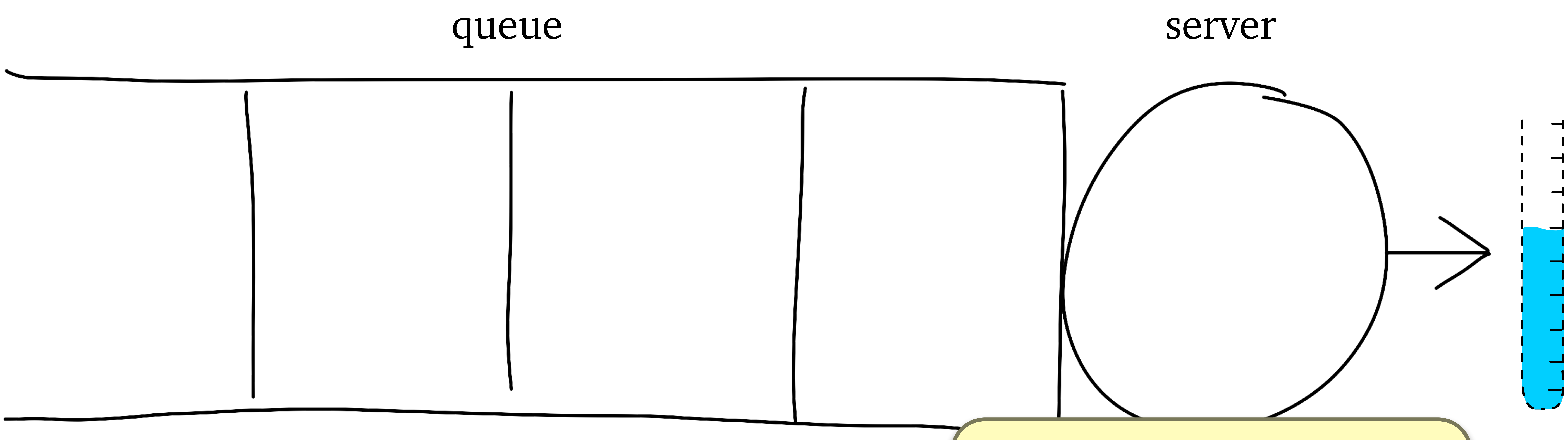


state = (age, arrival time)

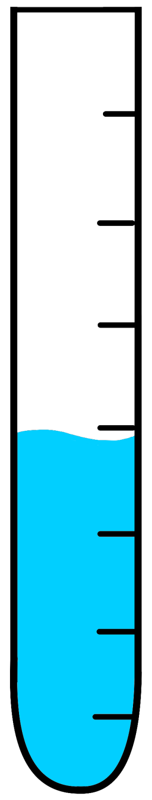
Gittins: order by **rank**(age)



Scheduling with unknown job sizes

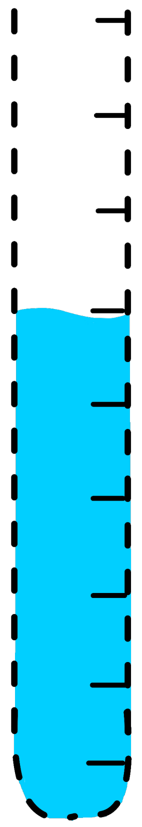


proxy for [size - age], depends on size distribution



state = (size, age, arrival time)

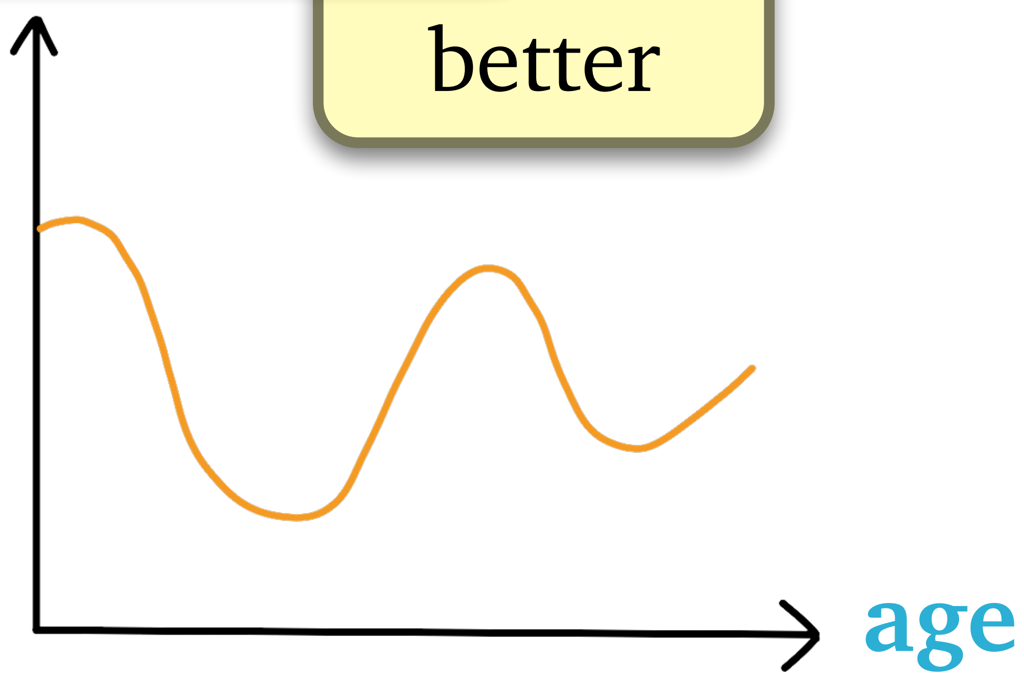
SRPT: order by [size - age]



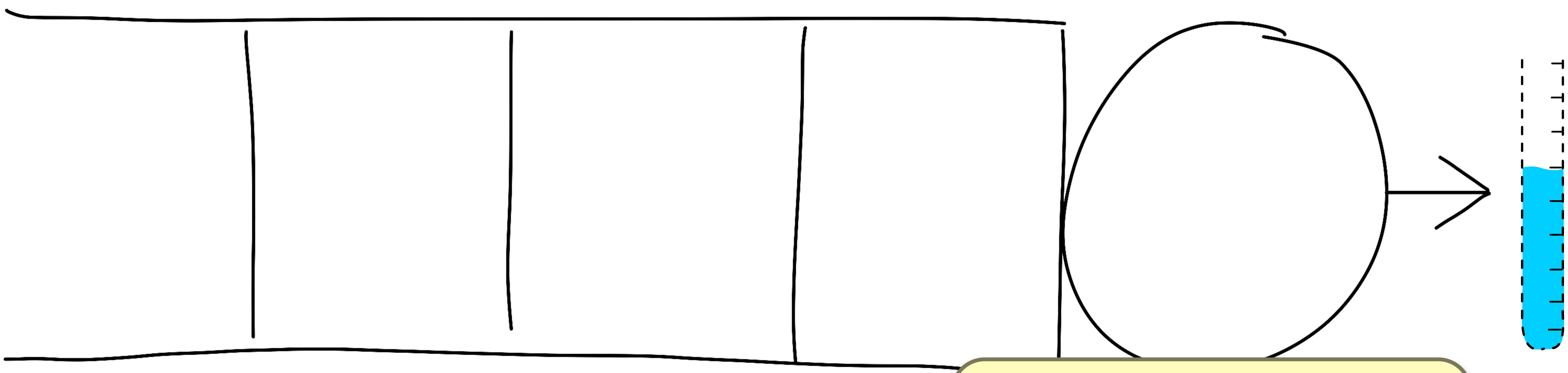
state = (age, arrival time)

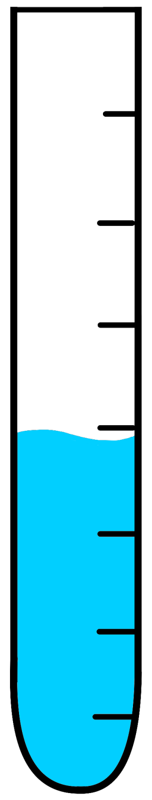
Gittins: order by **rank**(age)

smaller is better



Scheduling with unknown job sizes

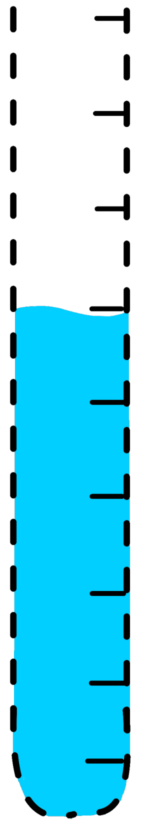




state = (size, age, arrival time)

SRPT: order by [size - age]

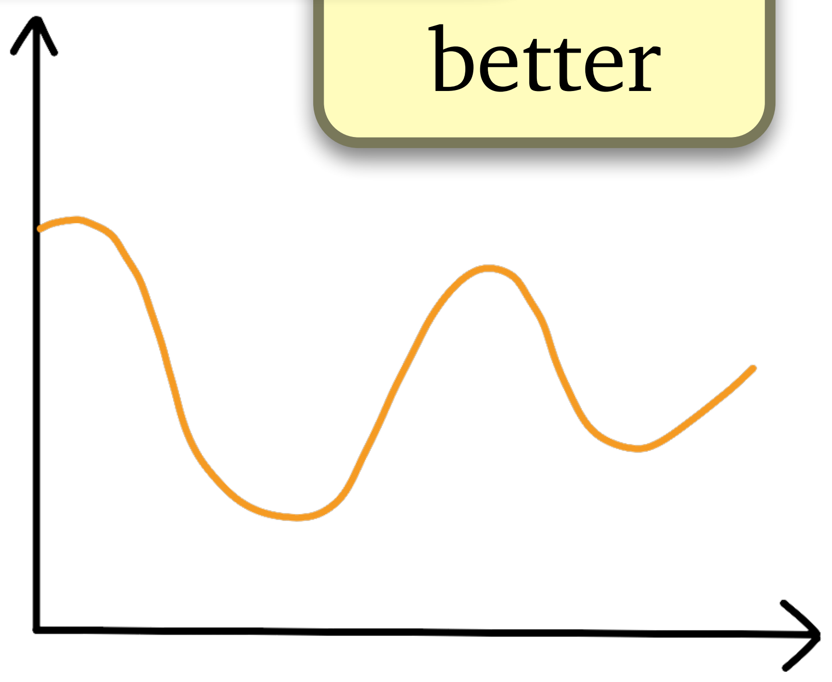
Boost: order by [A - *b(s)]*



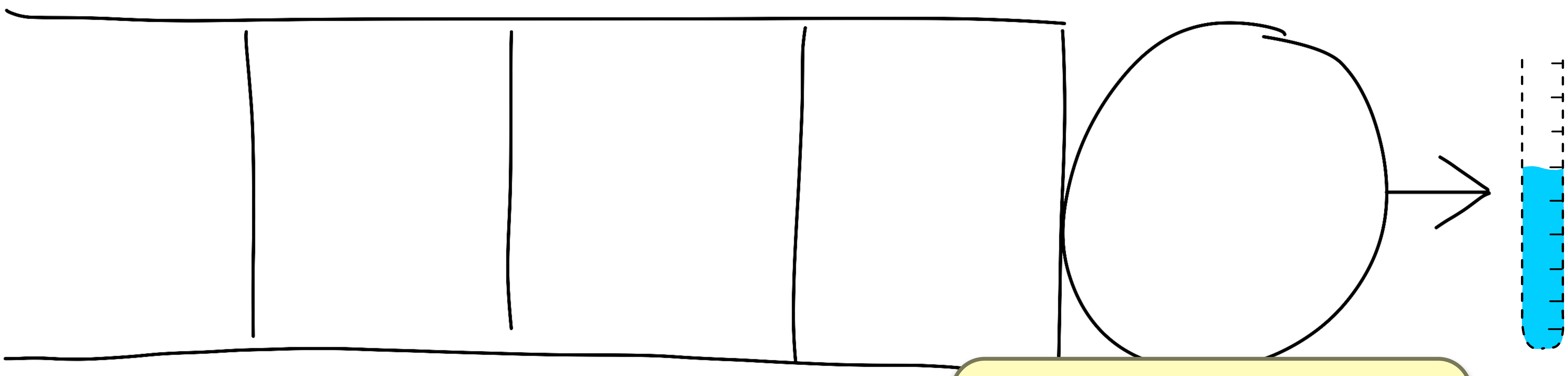
state = (age, arrival time)

proxy for [size - age], depends on size distribution

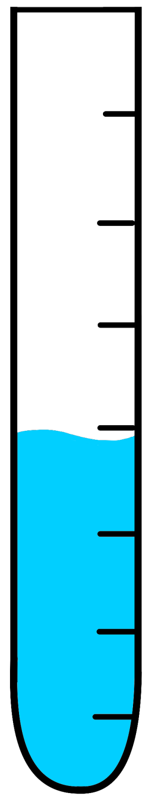
Gittins: order by **rank**(age)



Scheduling with unknown job sizes



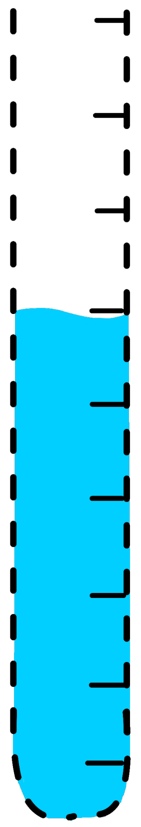
proxy for [size - age],
depends on size distribution



state = (size, age, arrival time)

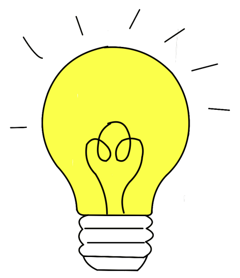
SRPT: order by [size - age]

Boost: order by [A - *b(s)]*



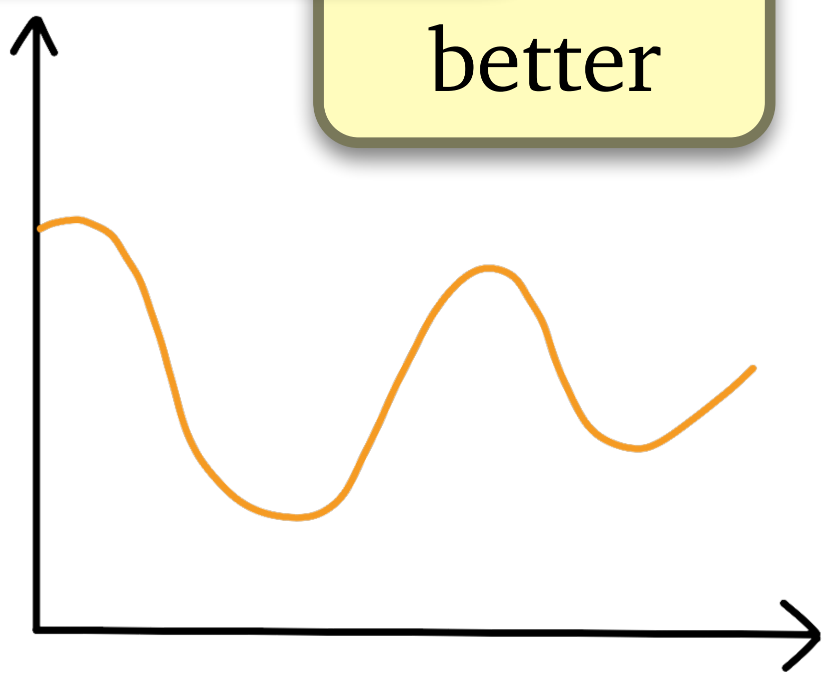
state = (age, arriv time)

Gittins: order by **rank**(age)

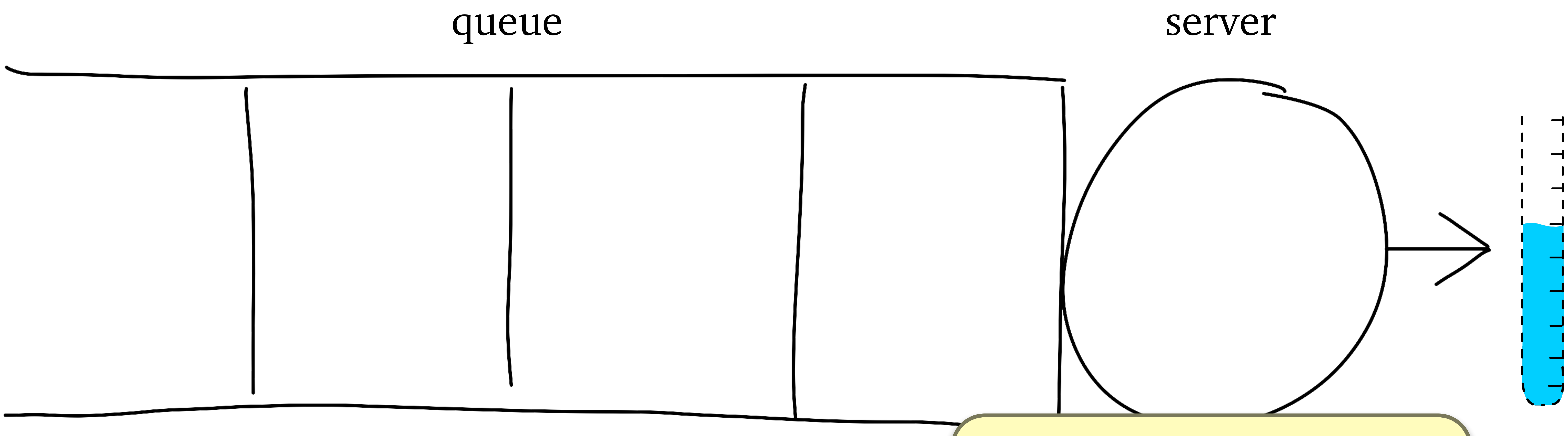


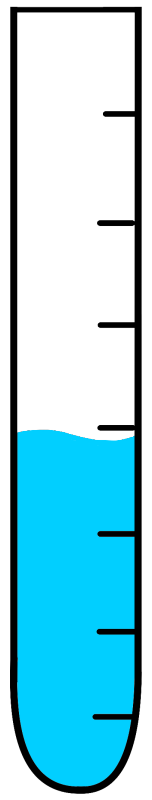
GittinsBoost:

order by [A - *b*(age)]



Scheduling with unknown job sizes

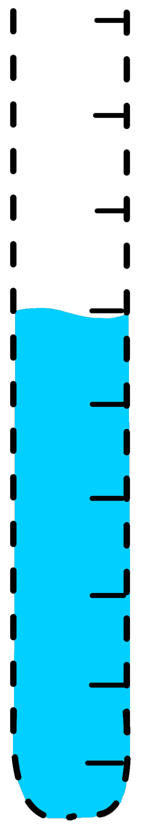




state = (size, age, arrival time)

SRPT: order by [size - age]

Boost: order by [A - *b(s)]*



state = (age, …

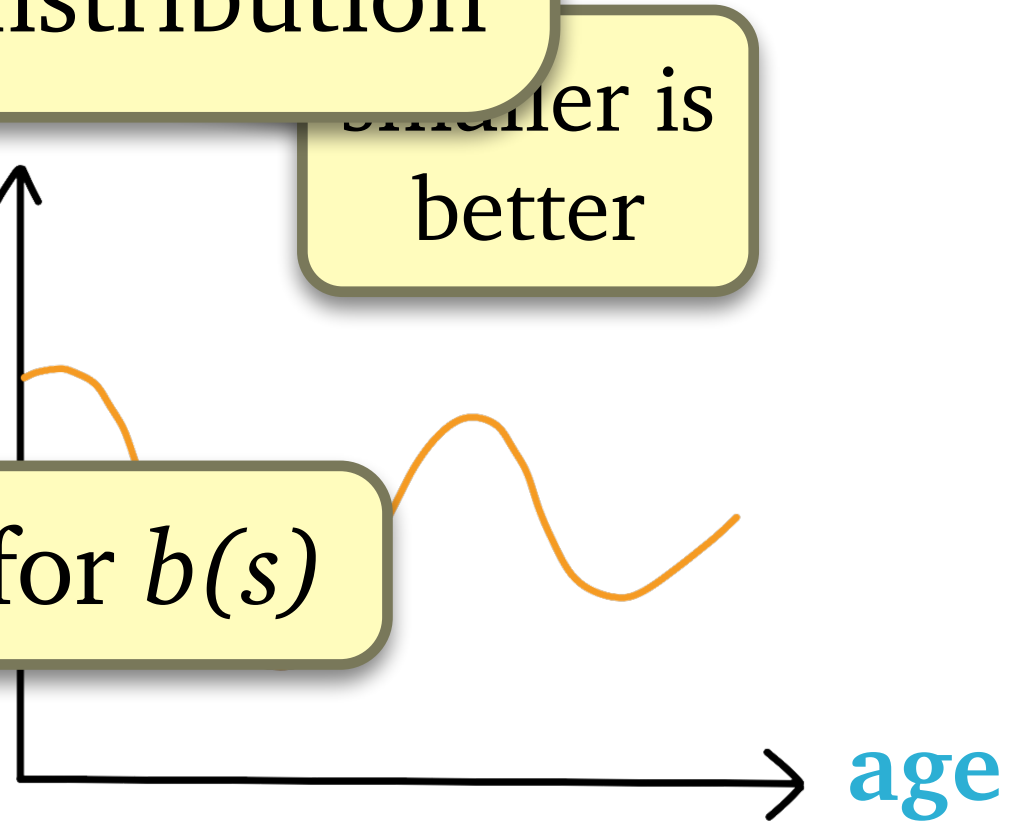
proxy for [size - age], depends on size distribution

Gittins: order by **rank**(age)

GittinsBoost:

order by [A - *b*(age)]

Proxy for *b(s)*

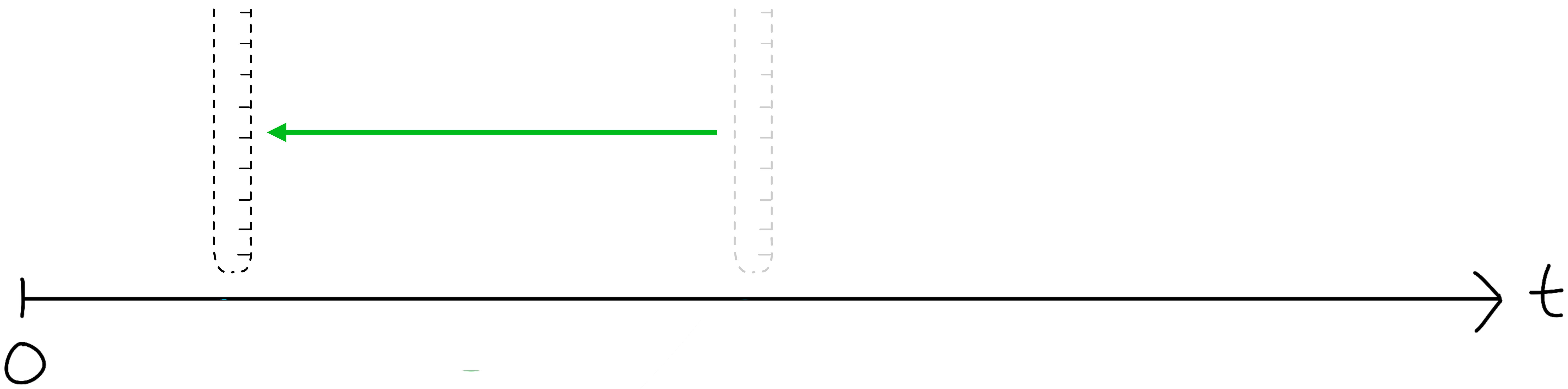


The GittinsBoost policy

defined by a *boost function* $b(x) \geq 0$ that maps **age** to boost

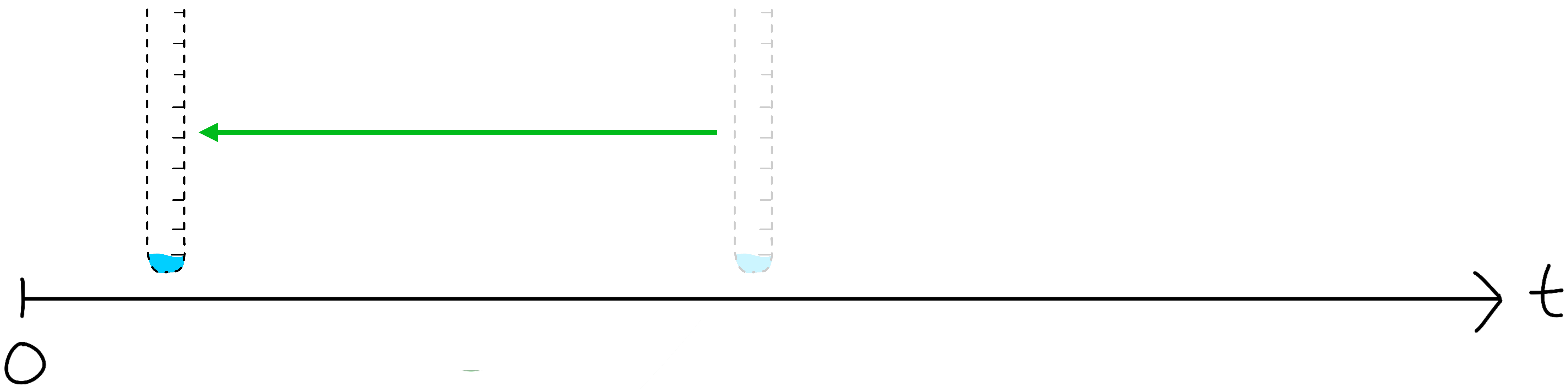
The GittinsBoost policy

defined by a *boost function* $b(x) \geq 0$ that maps **age** to boost



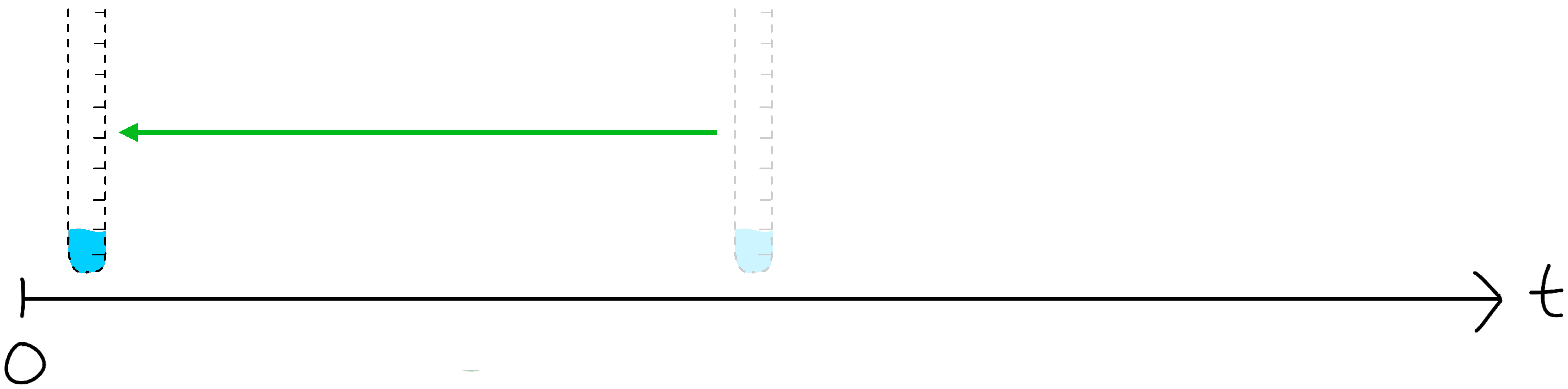
The GittinsBoost policy

defined by a *boost function* $b(x) \geq 0$ that maps **age** to boost



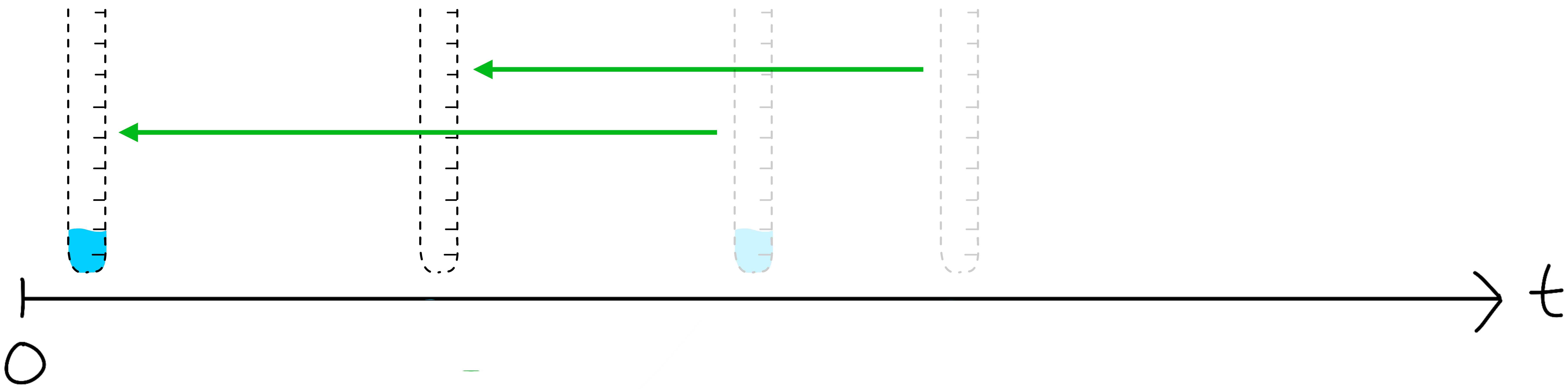
The GittinsBoost policy

defined by a *boost function* $b(x) \geq 0$ that maps **age** to boost



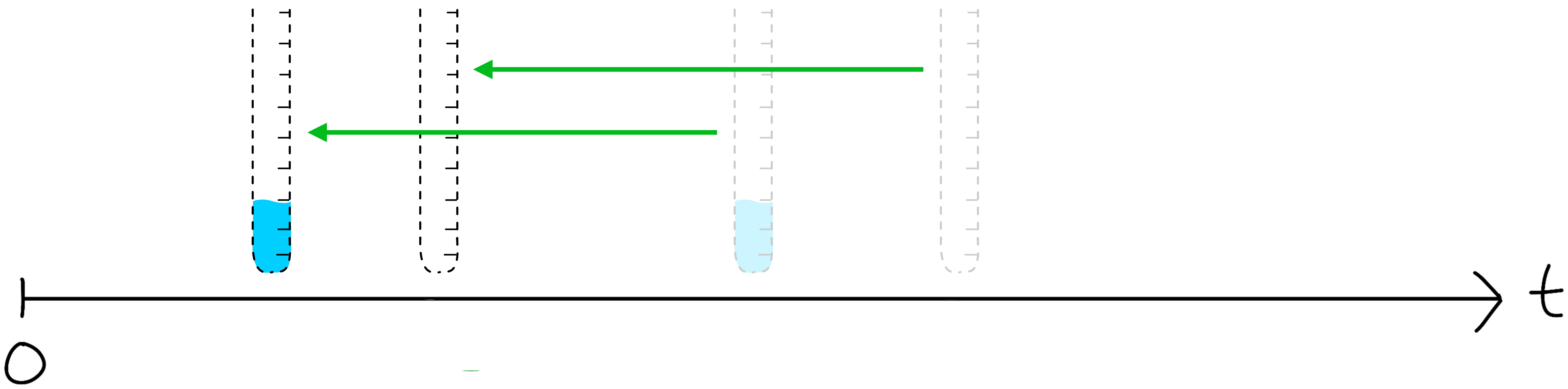
The GittinsBoost policy

defined by a *boost function* $b(x) \geq 0$ that maps **age** to boost



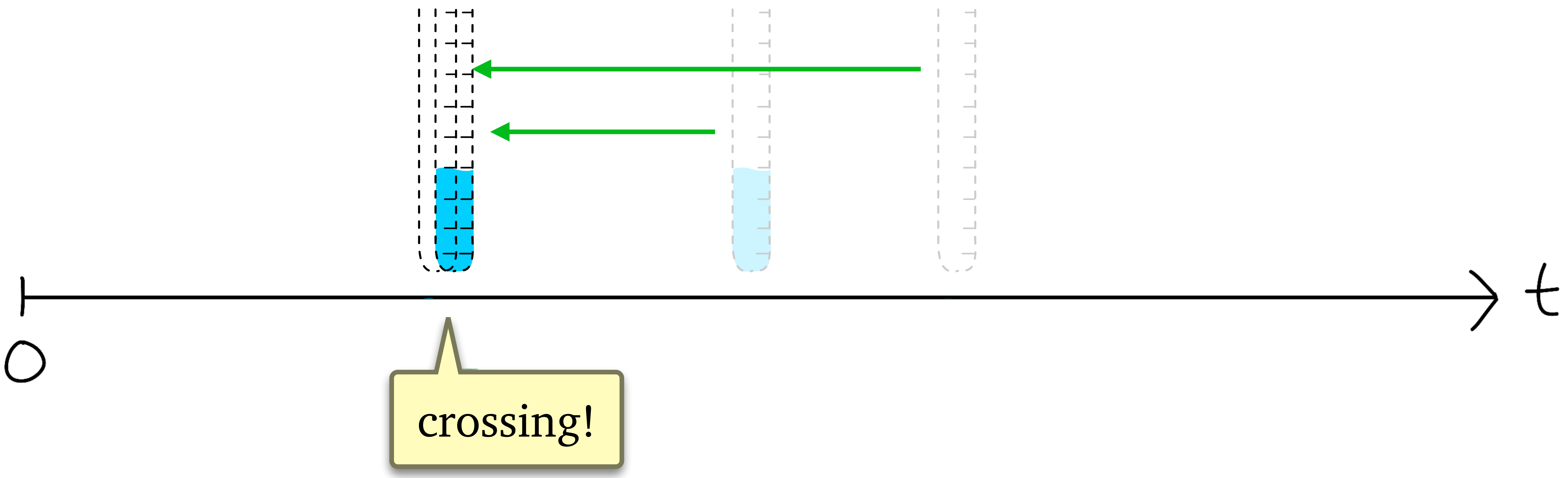
The GittinsBoost policy

defined by a *boost function* $b(x) \geq 0$ that maps **age** to boost



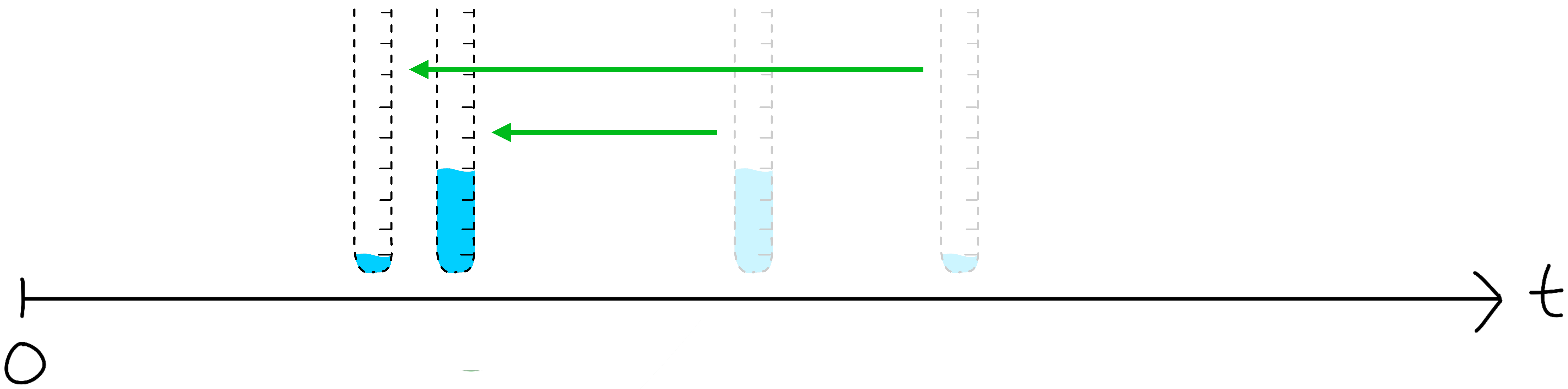
The GittinsBoost policy

defined by a *boost function* $b(x) \geq 0$ that maps **age** to boost



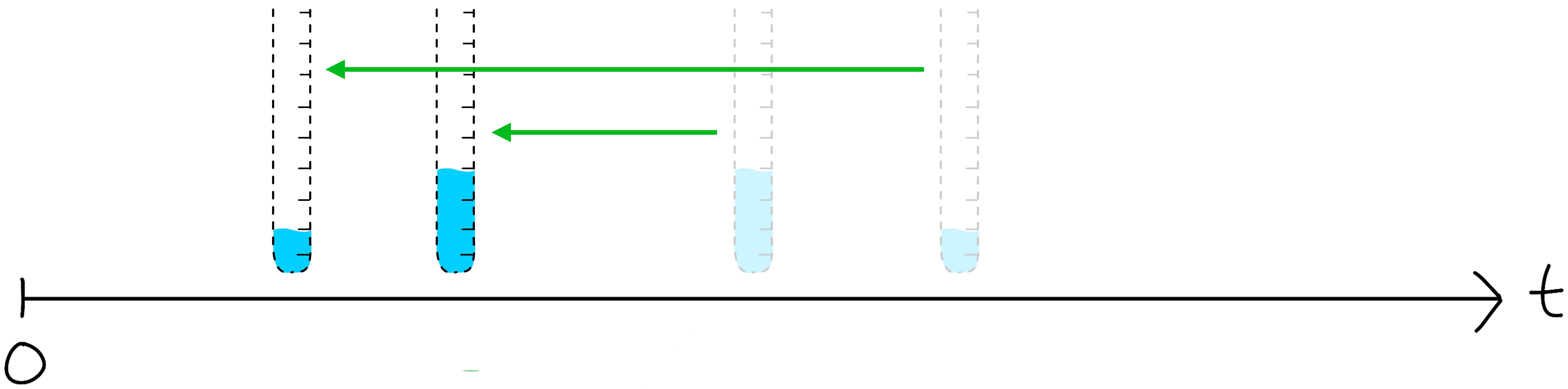
The GittinsBoost policy

defined by a *boost function* $b(x) \geq 0$ that maps **age** to boost



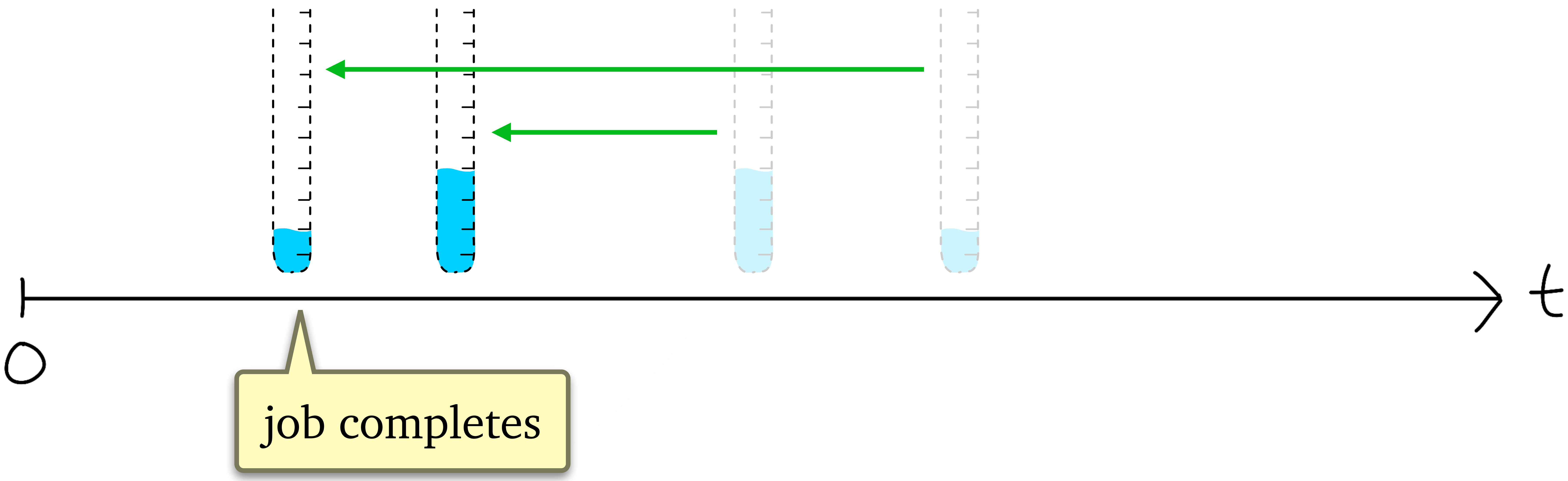
The GittinsBoost policy

defined by a *boost function* $b(x) \geq 0$ that maps **age** to boost



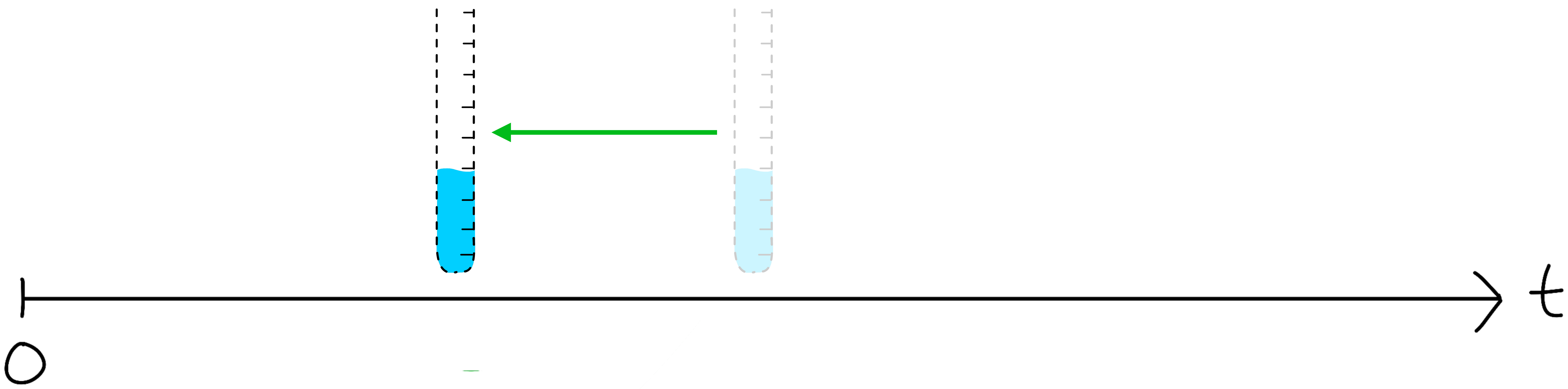
The **GittinsBoost** policy

defined by a *boost function* $b(x) \geq 0$ that maps **age** to boost



The GittinsBoost policy

defined by a *boost function* $b(x) \geq 0$ that maps age to boost



The GittinsBoost policy

defined by a *boost function* $b(x) \geq 0$ that maps **age** to boost

Which boost function is optimal?

The GittinsBoost policy

defined by a *boost function* $b(x) \geq 0$ that maps **age** to boost

Which boost function is optimal?

choosing:

$$b(x) = \frac{1}{\gamma} \log \left(\sup_{y>x} \frac{\mathbf{E}[e^{\gamma S} \mathbf{1}(S \leq y) \mid S > x]}{\mathbf{E}[e^{\gamma((S \wedge y) - x)} - 1 \mid S > x]} \right)$$

gets us a strongly optimal policy in the class of policies that don't use job size information.

The GittinsBoost policy

defined by a *boost function* $b(x) \geq 0$ that maps **age** to boost

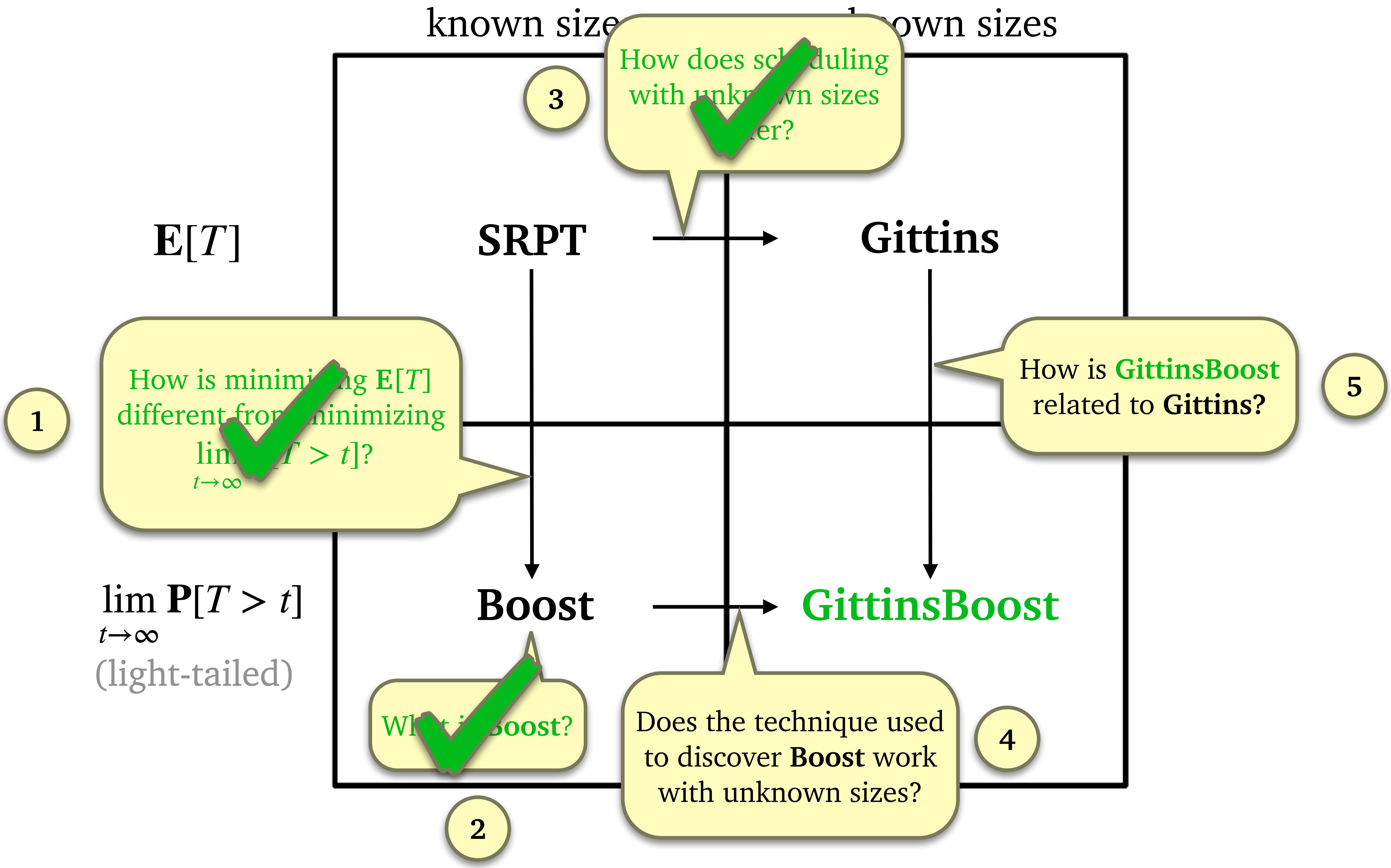
Which boost function is optimal?

choosing:

> looks similar to the Gittins rank function…

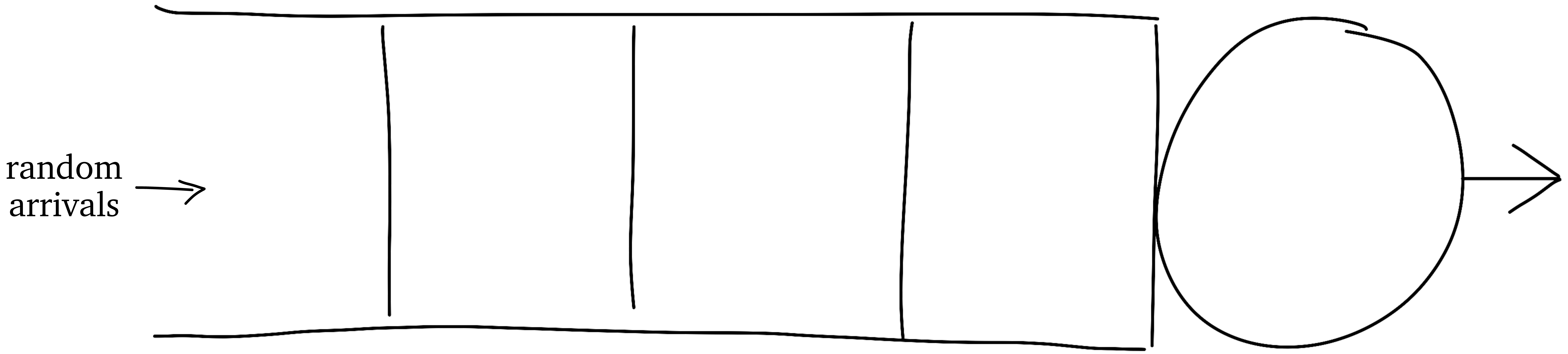
$$b(x) = \frac{1}{\gamma} \log \left(\sup_{y>x} \frac{\mathbf{E}[e^{\gamma S} \mathbf{1}(S \leq y) \mid S > x]}{\mathbf{E}[e^{\gamma((S \wedge y)-x)} - 1 \mid S > x]} \right)$$

gets us a strongly optimal policy in the class of policies that don't use job size information.

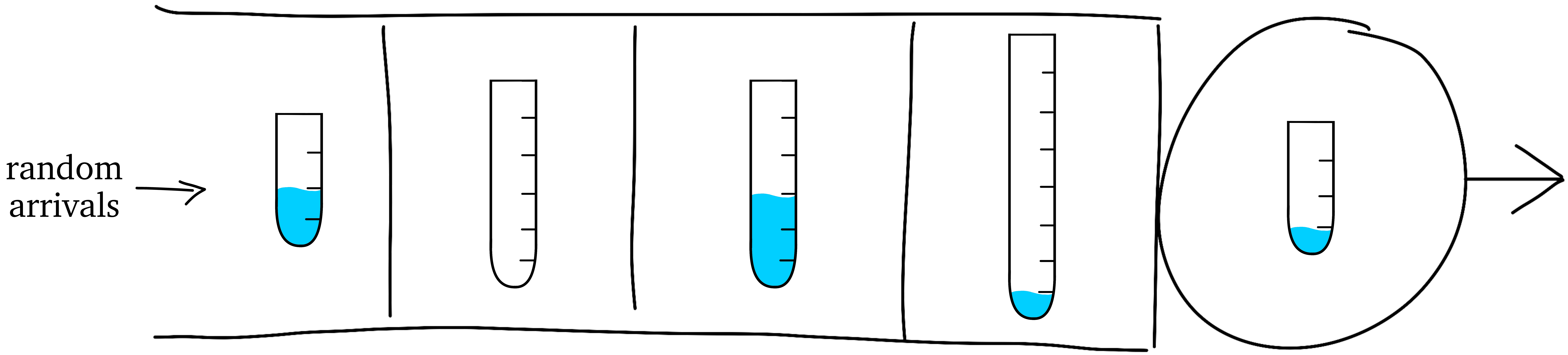


known size
own sizes
How does sc duling
with unk wn sizes
er?
3
E[T]
SRPT
Gittins
1
How is minimi ng E[T]
different fro minimizing
lim t→∞ T > t]?
How is GittinsBoost
related to Gittins?
5
lim t→∞ P[T > t]
(light-tailed)
Boost
GittinsBoost
Wh oost?
2
Does the technique used
to discover Boost work
with unknown sizes?
4

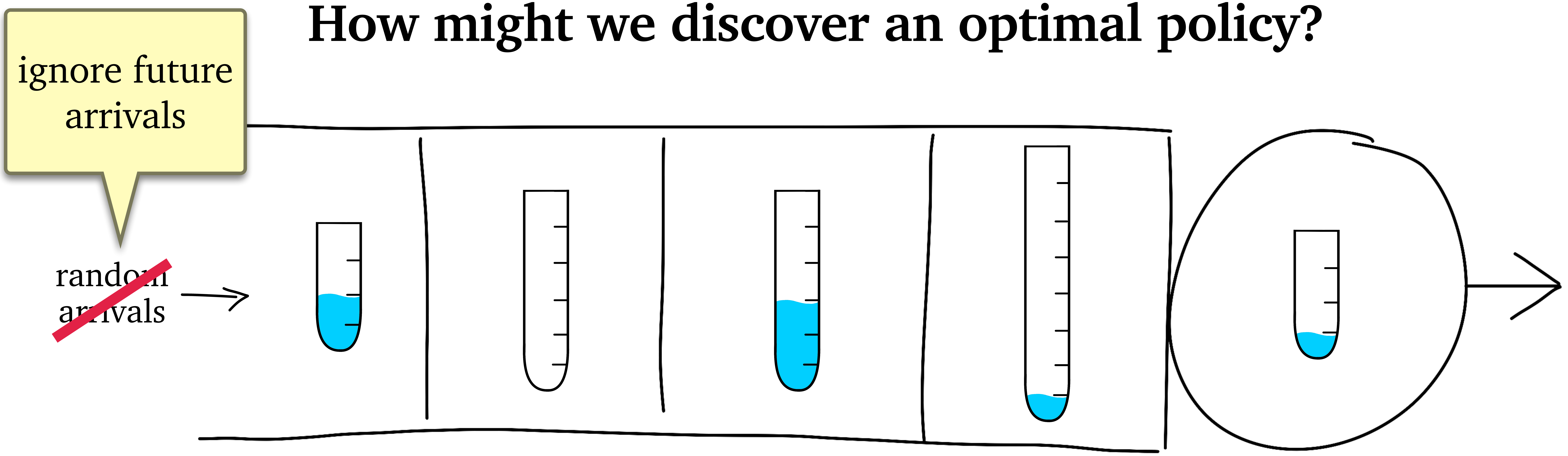
How might we discover an optimal policy?



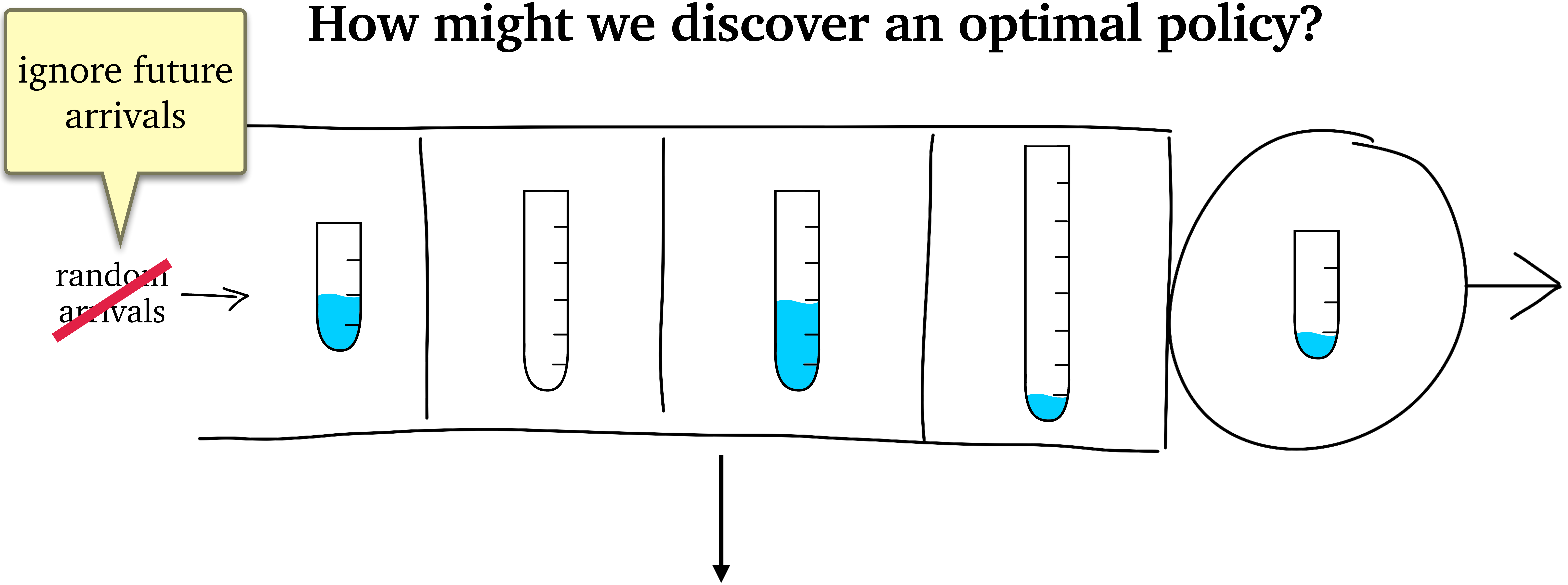
How might we discover an optimal policy?



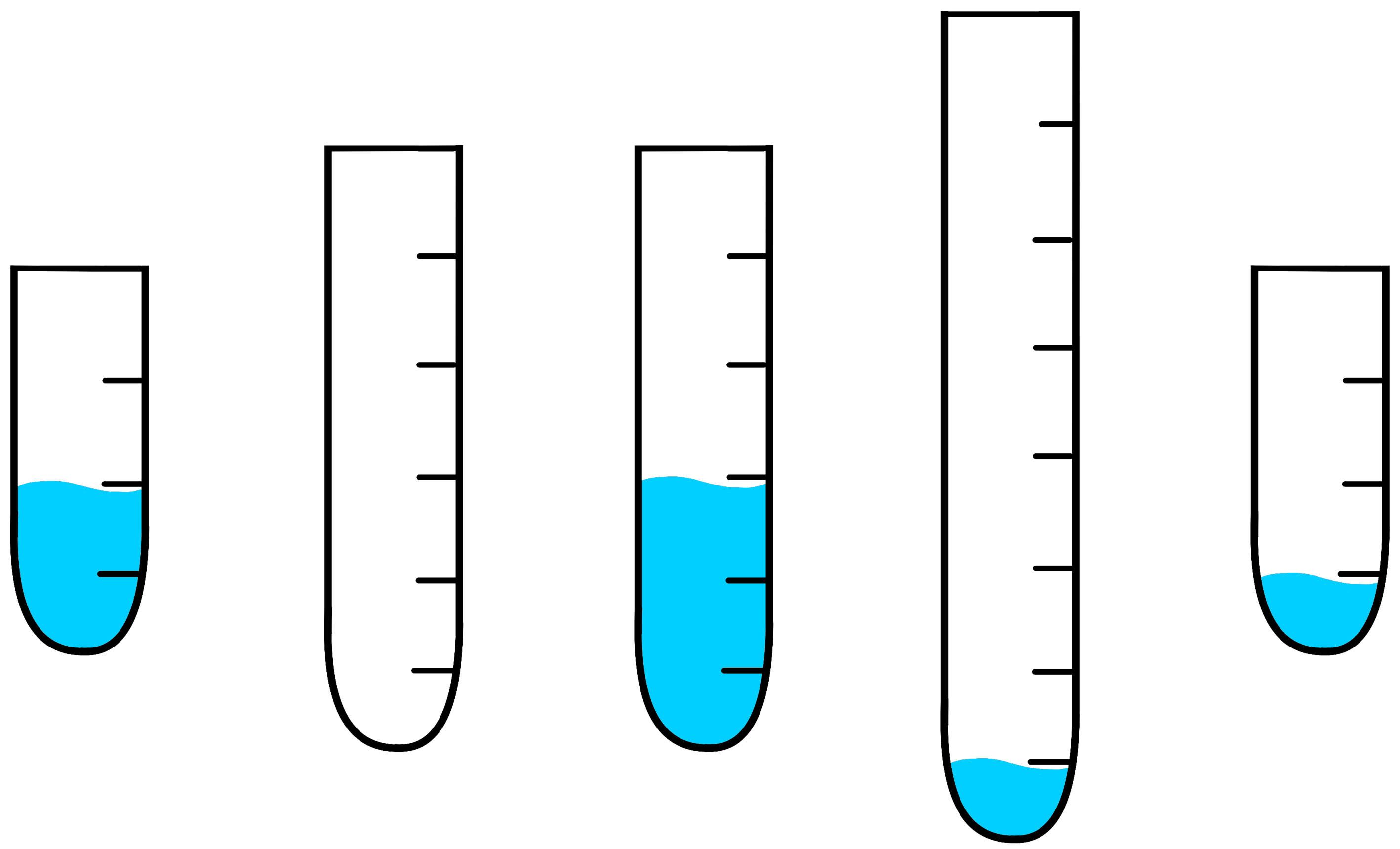
How might we discover an optimal policy?



How might we discover an optimal policy?



Batch Problem:



What is the optimal policy for the batch problem?

Queue Objective				
Batch Objective				
Optimal Policy				

What is the optimal policy for the batch problem?

Queue Objective	$\mathbf{E}_\pi[T]$ w/ known sizes			
Batch Objective				
Optimal Policy				

What is the optimal policy for the batch problem?

Queue Objective	$\mathbf{E}_\pi[T]$ **w/ known sizes**			
Batch Objective	$\frac{1}{N}\sum_{i=1}^{N} T_i$ **w/ known sizes**			
Optimal Policy				

What is the optimal policy for the batch problem?

Queue Objective	$\mathbf{E}_\pi[T]$ w/ known sizes			
Batch Objective	$\sum_{i=1}^{N} T_i$ w/ known sizes			
Optimal Policy				

What is the optimal policy for the batch problem?

Queue Objective	$\mathbf{E}_\pi[T]$ **w/ known sizes**			
Batch Objective	$\sum_{i=1}^{N} (D_i - A_i)$ **w/ known sizes**			
Optimal Policy				

What is the optimal policy for the batch problem?

Queue Objective	$\mathbf{E}_\pi[T]$ w/ known sizes			
Batch Objective	$\sum_{i=1}^{N} D_i - \sum_{i=1}^{N} A_i$ w/ known sizes			
Optimal Policy				

What is the optimal policy for the batch problem?

Queue Objective	$\mathbf{E}_\pi[T]$ w/ known sizes			
Batch Objective	$\sum_{i=1}^{N} D_i$ w/ known sizes			
Optimal Policy				

What is the optimal policy for the batch problem?

Queue Objective	$\mathbf{E}_\pi[T]$ w/ known sizes			
Batch Objective	$\sum_{i=1}^{N} D_i$ w/ known sizes			
Optimal Policy	**SRPT**			

What is the optimal policy for the batch problem?

Queue Objective	$\mathbf{E}_\pi[T]$ w/ known sizes	$\mathbf{E}_\pi[T]$ w/ unknown sizes		
Batch Objective	$\sum_{i=1}^{N} D_i$ w/ known sizes			
Optimal Policy	**SRPT**			

What is the optimal policy for the batch problem?

Queue Objective	$\mathbf{E}_\pi[T]$ w/ known sizes	$\mathbf{E}_\pi[T]$ w/ unknown sizes		
Batch Objective	$\sum_{i=1}^{N} D_i$ w/ known sizes	$\mathbf{E}_\pi\left[\sum_{i=1}^{N} D_i\right]$ w/ unknown sizes		
Optimal Policy	**SRPT**			

What is the optimal policy for the batch problem?

Queue Objective	$\mathbf{E}_\pi[T]$ w/ known sizes	$\mathbf{E}_\pi[T]$ w/ unknown sizes		
Batch Objective	$\sum_{i=1}^{N} D_i$ w/ known sizes	$\mathbf{E}_\pi\left[\sum_{i=1}^{N} D_i\right]$ w/ unknown sizes		
Optimal Policy	**SRPT**	**Gittins**		

What is the optimal policy for the batch problem?

Queue Objective	$\mathbf{E}_\pi[T]$ w/ known sizes	$\mathbf{E}_\pi[T]$ w/ unknown sizes	$\lim_{t\to\infty} \mathbf{P}[T > t]$ w/ known sizes	
Batch Objective	$\sum_{i=1}^{N} D_i$ w/ known sizes	$\mathbf{E}_\pi\left[\sum_{i=1}^{N} D_i\right]$ w/ unknown sizes		
Optimal Policy	**SRPT**	**Gittins**		

What is the optimal policy for the batch problem?

Queue Objective	$\mathbf{E}_\pi[T]$ w/ known sizes	$\mathbf{E}_\pi[T]$ w/ unknown sizes	$\lim_{t\to\infty} \mathbf{P}[T > t]$ w/ known sizes	
Batch Objective	$\sum_{i=1}^{N} D_i$ w/ known sizes	$\mathbf{E}_\pi\left[\sum_{i=1}^{N} D_i\right]$ w/ unknown sizes	$\sum_{i=1}^{N} e^{\gamma D_i} e^{-\gamma A_i}$ w/ known sizes	
Optimal Policy	**SRPT**	**Gittins**		

a key insight in the **Boost** paper

What is the optimal policy for the batch problem?

Queue Objective	$\mathbf{E}_\pi[T]$ w/ known sizes	$\mathbf{E}_\pi[T]$ w/ unknown sizes	$\lim_{t\to\infty} \mathbf{P}[T > t]$ w/ known sizes	
Batch Objective	$\sum_{i=1}^{N} D_i$ w/ known sizes	$\mathbf{E}_\pi\left[\sum_{i=1}^{N} D_i\right]$ w/ unknown sizes	$\sum_{i=1}^{N} e^{\gamma D_i} e^{-\gamma A_i}$ w/ known sizes	
Optimal Policy	**SRPT**	**Gittins**		

a key insight in the **Boost** paper

What is the optimal policy for the batch problem?

Queue Objective	$\mathbf{E}_\pi[T]$ w/ known sizes	$\mathbf{E}_\pi[T]$ w/ unknown sizes	$\lim_{t\to\infty} \mathbf{P}[T > t]$ w/ known sizes	
Batch Objective	$\sum_{i=1}^{N} D_i$ w/ known sizes	$\mathbf{E}_\pi\left[\sum_{i=1}^{N} D_i\right]$ w/ unknown sizes	$\sum_{i=1}^{N} e^{\gamma D_i} e^{-\gamma A_i}$ w/ known sizes	
Optimal Policy	**SRPT**	**Gittins**	**Boost**	

a key insight in the **Boost** paper

What is the optimal policy for the batch problem?

Queue Objective	$\mathbf{E}_\pi[T]$ w/ known sizes	$\mathbf{E}_\pi[T]$ w/ unknown sizes	$\lim_{t\to\infty} \mathbf{P}[T > t]$ w/ known sizes	$\lim_{t\to\infty} \mathbf{P}[T > t]$ w/ unknown sizes
Batch Objective	$\sum_{i=1}^{N} D_i$ w/ known sizes	$\mathbf{E}_\pi\left[\sum_{i=1}^{N} D_i\right]$ w/ unknown sizes	$\sum_{i=1}^{N} e^{\gamma D_i} e^{-\gamma A_i}$ w/ known sizes	
Optimal Policy	**SRPT**	**Gittins**	**Boost**	

What is the optimal policy for the batch problem?

Queue Objective	$\mathbf{E}_\pi[T]$ w/ known sizes	$\mathbf{E}_\pi[T]$ w/ unknown sizes	$\lim_{t\to\infty} \mathbf{P}[T > t]$ w/ known sizes	$\lim_{t\to\infty} \mathbf{P}[T > t]$ w/ unknown sizes
Batch Objective	$\sum_{i=1}^{N} D_i$ w/ known sizes	$\mathbf{E}_\pi\left[\sum_{i=1}^{N} D_i\right]$ w/ unknown sizes	$\sum_{i=1}^{N} e^{\gamma D_i} e^{-\gamma A_i}$ w/ known sizes	$\mathbf{E}_\pi\left[\sum_{i=1}^{N} e^{\gamma D_i} e^{-\gamma A_i}\right]$ w/ unknown sizes
Optimal Policy	**SRPT**	**Gittins**	**Boost**	

What is the optimal policy for the batch problem?

Queue Objective	$\mathbf{E}_\pi[T]$ w/ known sizes	$\mathbf{E}_\pi[T]$ w/ unknown sizes	$\lim_{t\to\infty} \mathbf{P}[T > t]$ w/ known sizes	$\lim_{t\to\infty} \mathbf{P}[T > t]$ w/ unknown sizes
Batch Objective	$\sum_{i=1}^{N} D_i$ w/ known sizes	$\mathbf{E}_\pi\left[\sum_{i=1}^{N} D_i\right]$ w/ unknown sizes	$\sum_{i=1}^{N} e^{\gamma D_i} e^{-\gamma A_i}$ w/ known sizes	$\mathbf{E}_\pi\left[\sum_{i=1}^{N} e^{\gamma D_i} e^{-\gamma A_i}\right]$ w/ unknown sizes
Optimal Policy	**SRPT**	**Gittins**	**Boost**	**GittinsBoost**

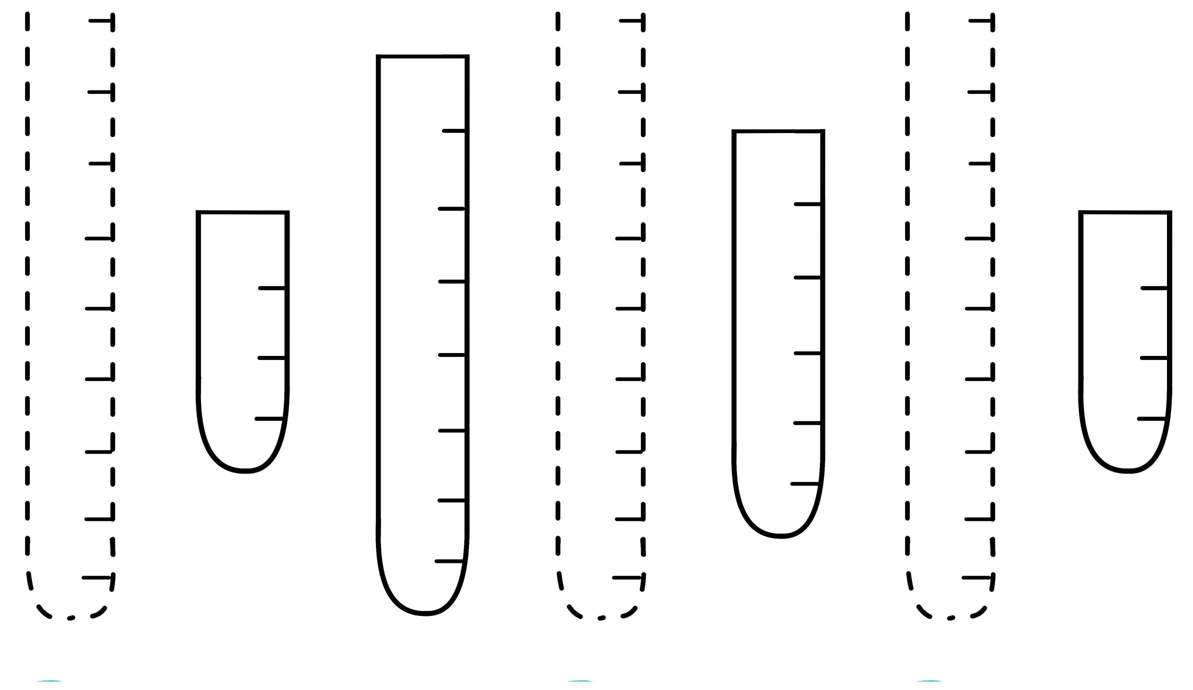
What is the optimal policy for the batch problem?

Queue Objective	$\mathbf{E}_\pi[T]$ w/ known sizes	$\mathbf{E}_\pi[T]$ w/ unknown sizes	$\lim_{t\to\infty} \mathbf{P}[T > t]$ w/ known sizes	$\lim_{t\to\infty} \mathbf{P}[T > t]$ w/ unknown sizes
Batch Objective	$\sum_{i=1}^{N} D_i$ w/ known sizes	$\mathbf{E}_\pi\left[\sum_{i=1}^{N} D_i\right]$ w/ unknown sizes	$\sum_{i=1}^{N} e^{\gamma D_i} e^{-\gamma A_i}$ w/ known sizes	$\mathbf{E}_\pi\left[\sum_{i=1}^{N} e^{\gamma D_i} e^{-\gamma A_i}\right]$ w/ unknown sizes
Optimal Policy	**SRPT**	**Gittins**	**Boost**	**GittinsBoost**

All of these are in the Gittins family of policies!

What is the Gittins family of policies?

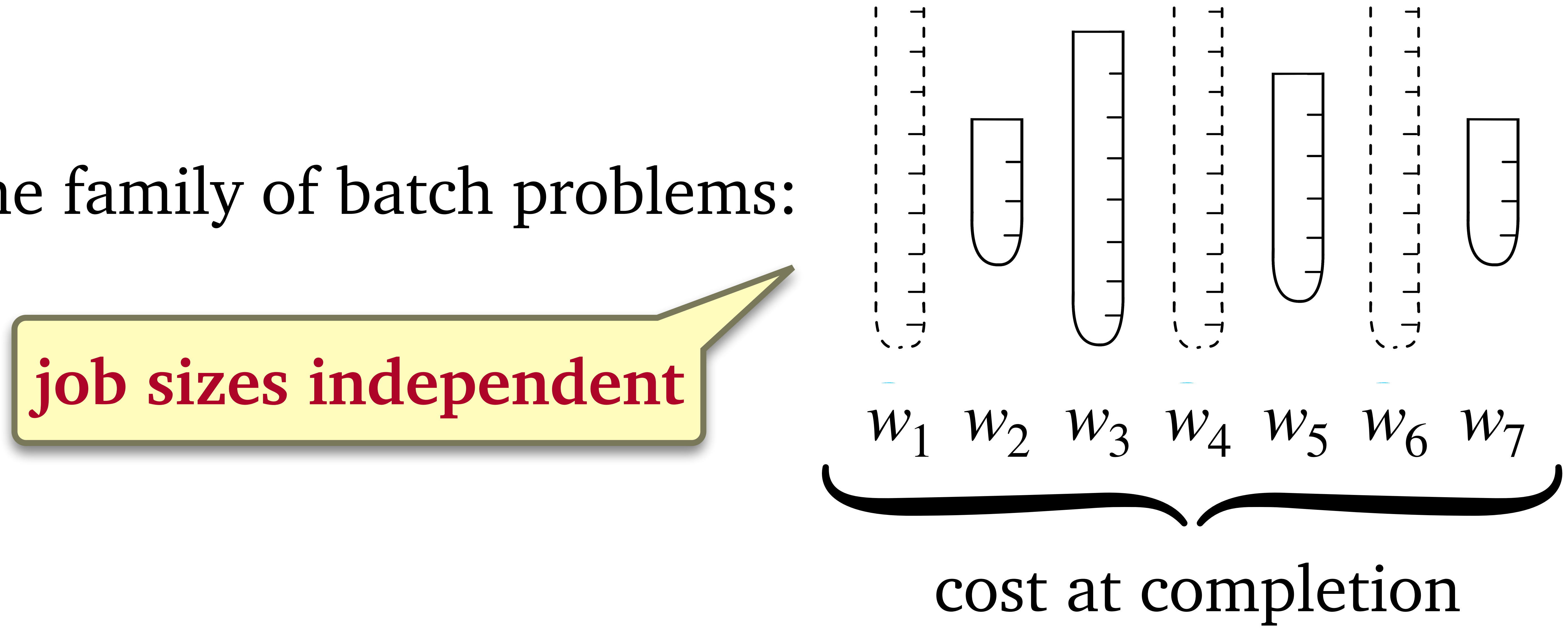
Gittins policies solve the family of batch problems:



job sizes independent

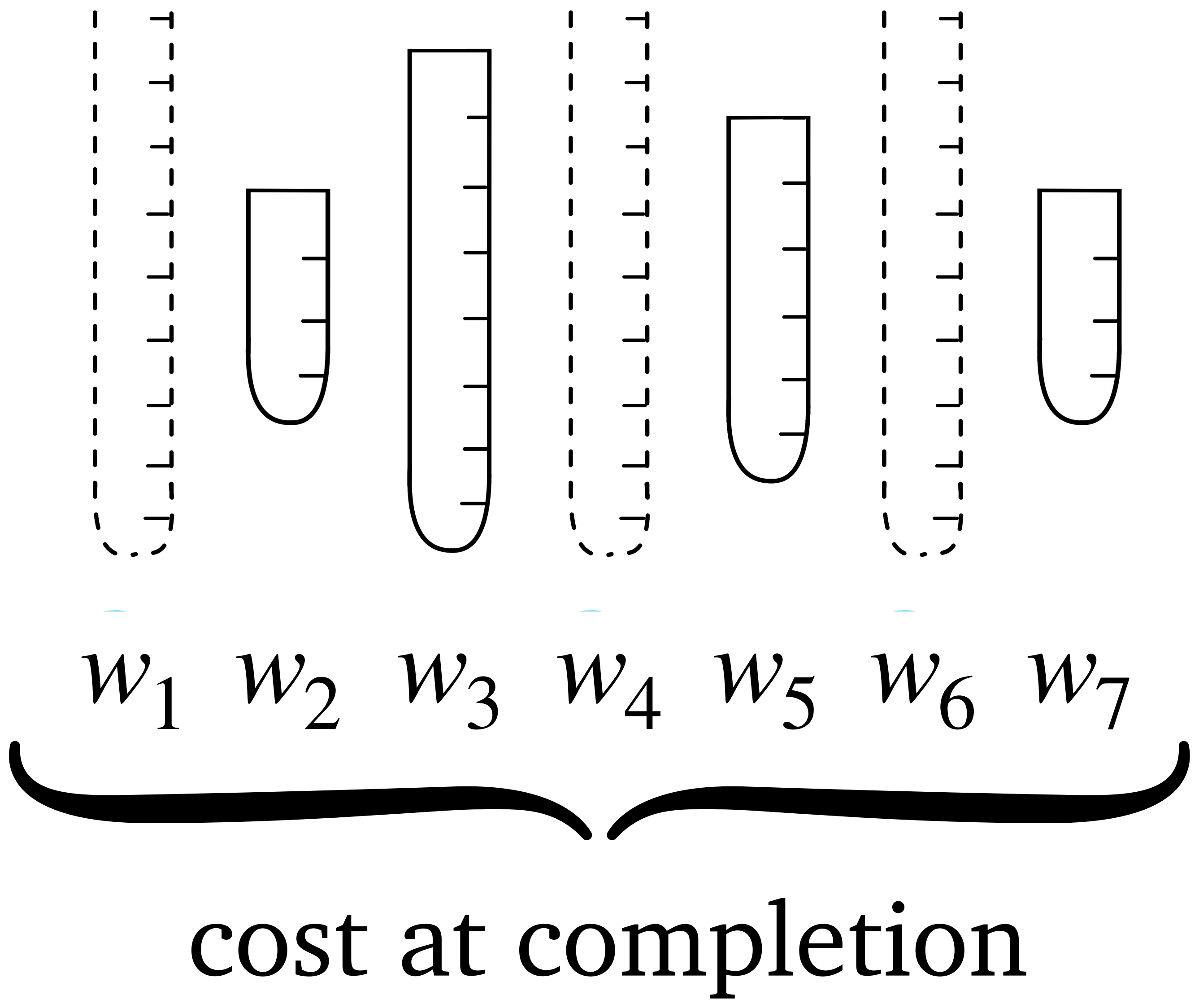
What is the Gittins family of policies?

Gittins policies solve the family of batch problems:



What is the Gittins family of policies?

Gittins policies solve the family of batch problems:



job sizes independent

with objective:

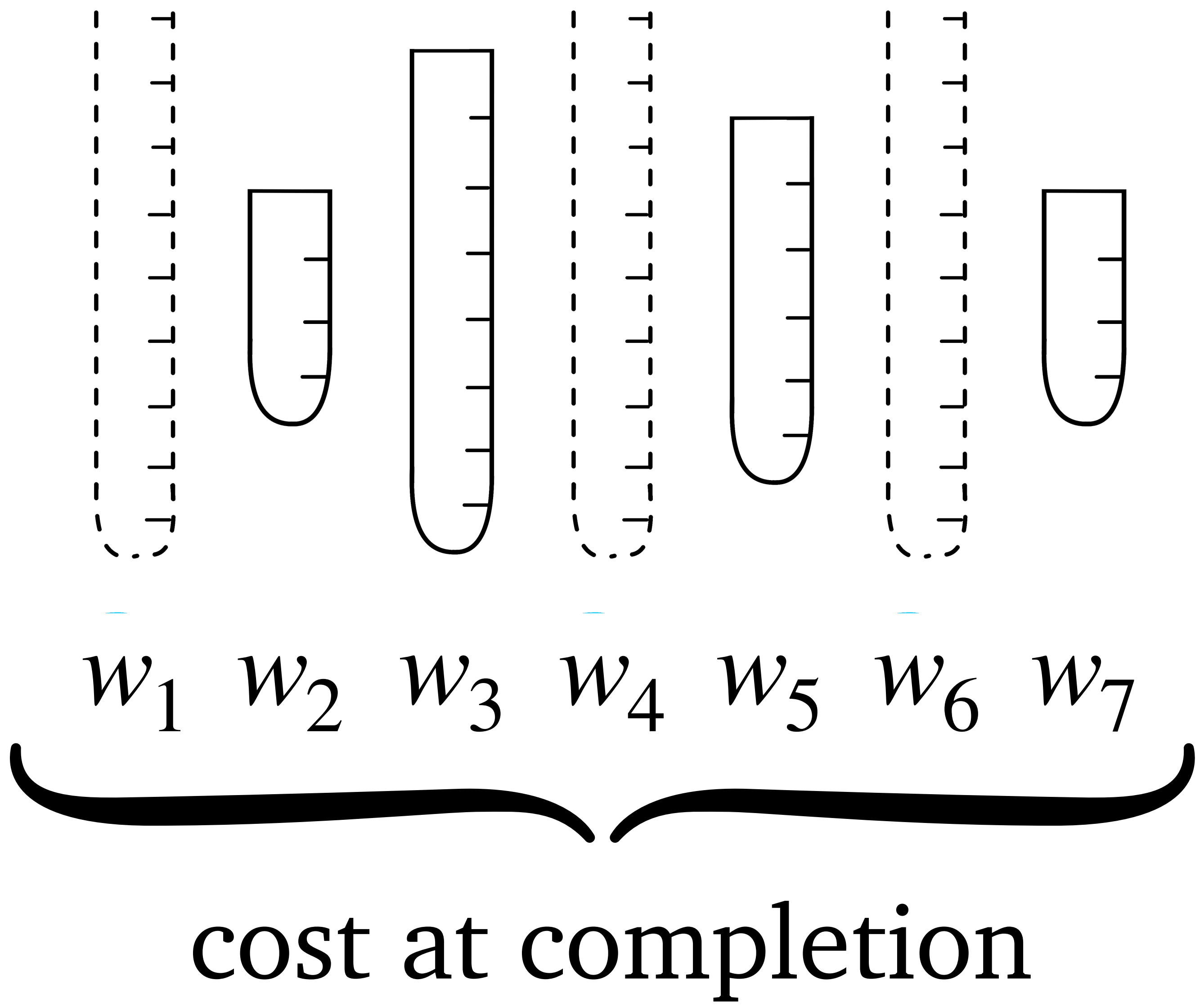
discounting

$$\text{minimize} \quad \mathbf{E}_\pi\Big[\sum_{i=1}^{N} e^{-\beta D_i} w_i\Big]$$

$(\beta > 0)$

What is the Gittins family of policies?

Gittins policies solve the family of batch problems:



job sizes independent

with objective:

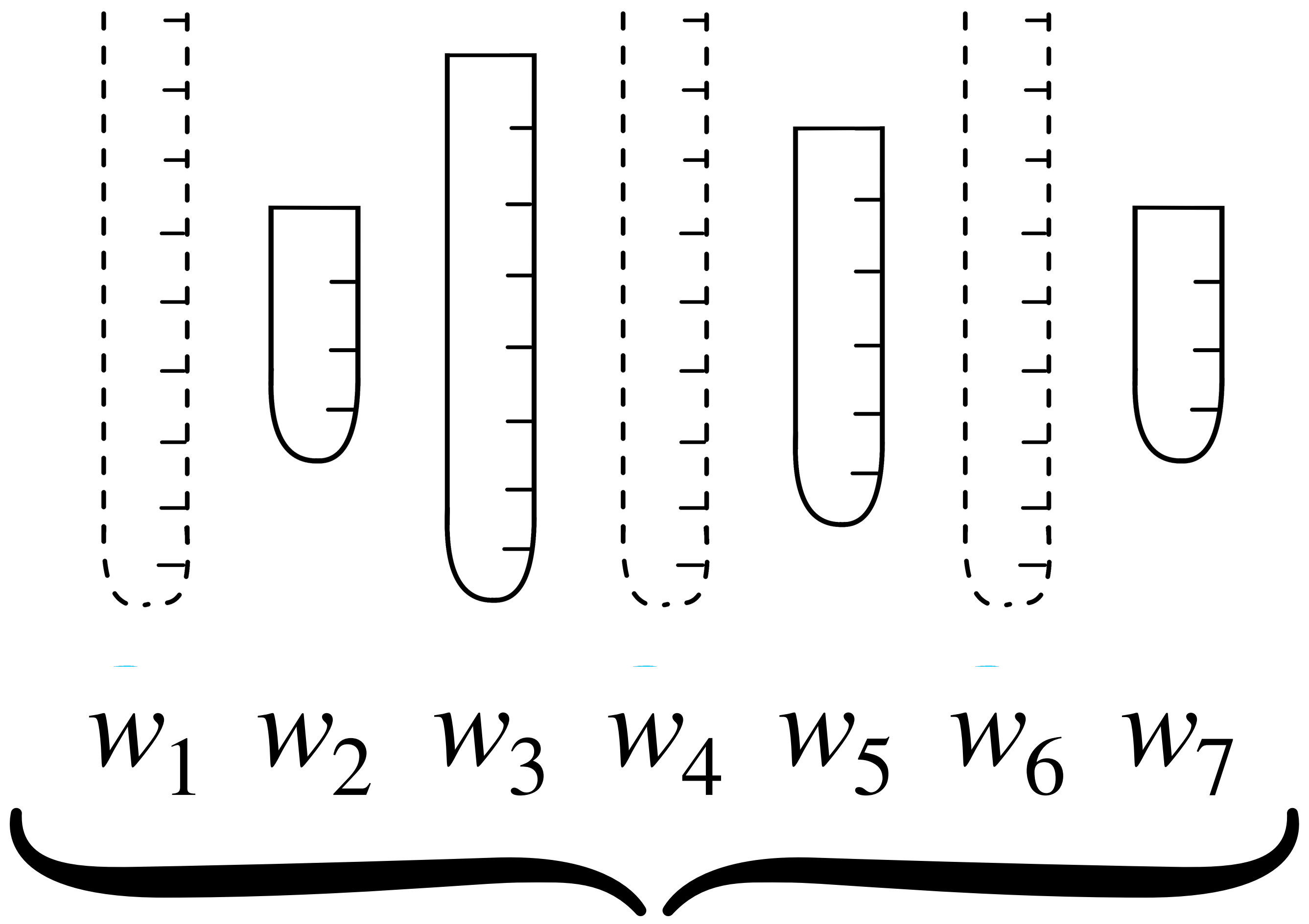
discounting

$$\text{minimize} \;\; \mathbf{E}_\pi\Big[\sum_{i=1}^{N} e^{-\beta D_i} w_i\Big] \quad \text{or} \quad \text{minimize} \;\; \mathbf{E}_\pi\Big[\sum_{i=1}^{N} D_i w_i\Big]$$

$(\beta > 0)$ “$(\beta = 0)$”

What is the Gittins family of policies?

Gittins policies solve the family of batch problems:



job sizes independent

with objective:

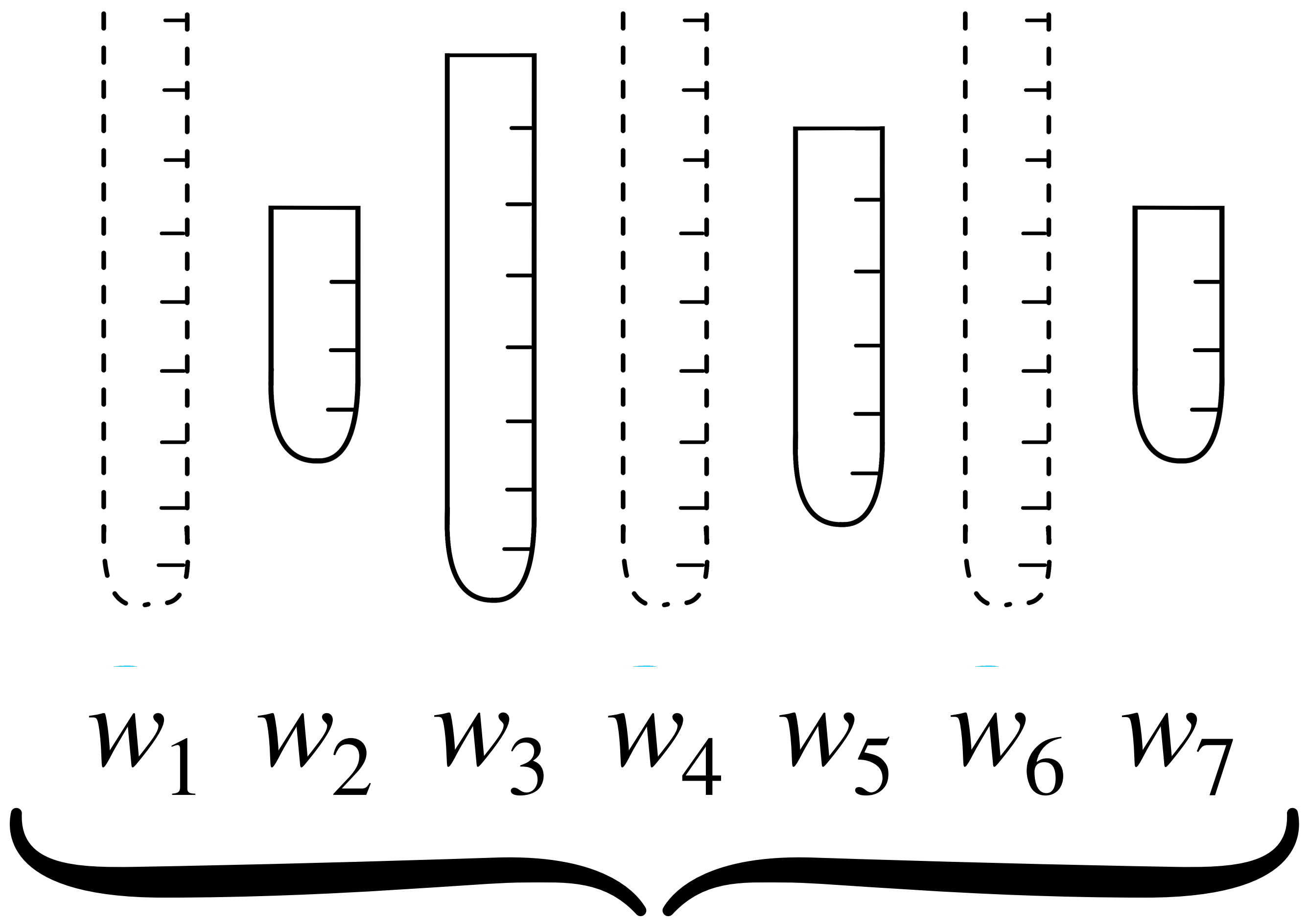
discounting

minimize $\mathbf{E}_\pi\left[\sum_{i=1}^{N} e^{-\beta D_i} w_i\right]$ or minimize $\mathbf{E}_\pi\left[\sum_{i=1}^{N} D_i w_i\right]$ or minimize $\mathbf{E}_\pi\left[\sum_{i=1}^{N} e^{-\beta D_i} w_i\right]$

$(\beta > 0)$ “$(\beta = 0)$” new-ish! $(\beta < 0)$

What is the Gittins family of policies?

Gittins policies solve the family of batch problems:



job sizes independent

with objective:

discounting

negative discounting = inflation

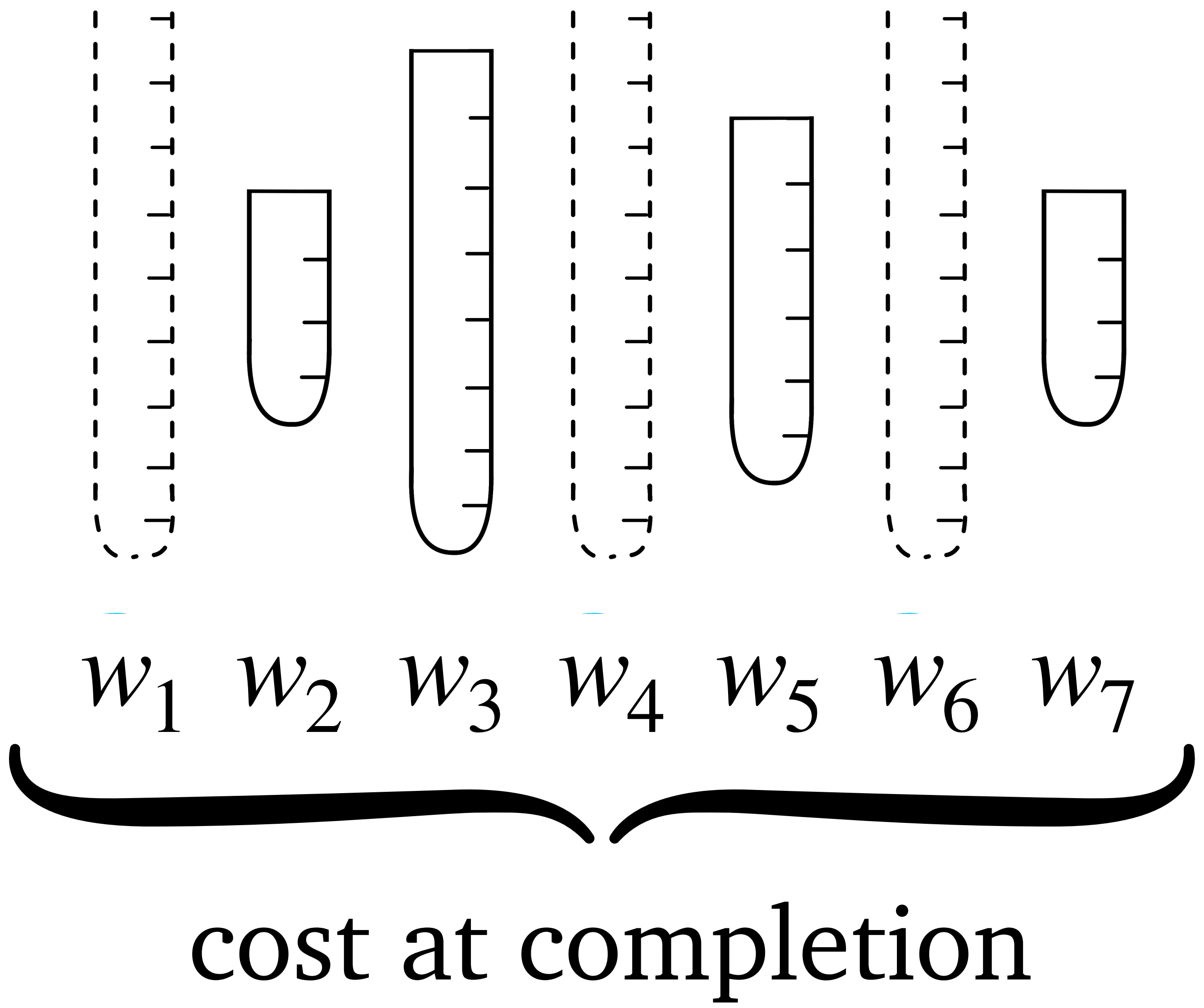
$$\text{minimize } \mathbf{E}_\pi\Big[\sum_{i=1}^{N} e^{-\beta D_i} w_i\Big] \quad \text{or} \quad \text{minimize } \mathbf{E}_\pi\Big[\sum_{i=1}^{N} D_i w_i\Big] \quad \text{or} \quad \text{minimize } \mathbf{E}_\pi\Big[\sum_{i=1}^{N} e^{-\beta D_i} w_i\Big]$$

$(\beta > 0)$ “$(\beta = 0)$” **new-ish!** $(\beta < 0)$

What is the Gittins family of policies?

Gittins policies solve the family of batch problems:

job sizes independent



with objective:

discounting

$e^{\gamma D_i} e^{-\gamma A_i}$

$$\text{minimize } \mathbf{E}_\pi\left[\sum_{i=1}^{N} e^{-\beta D_i} w_i\right] \quad \text{or} \quad \text{minimize } \mathbf{E}_\pi\left[\sum_{i=1}^{N} D_i w_i\right] \quad \text{or} \quad \text{minimize } \mathbf{E}_\pi\left[\sum_{i=1}^{N} e^{-\beta D_i} w_i\right]$$

$(\beta > 0)$ “$(\beta = 0)$” new-ish! $(\beta < 0)$

What is the optimal policy for the batch problem?

Queue Objective	$\mathbf{E}_\pi[T]$ w/ known sizes	$\mathbf{E}_\pi[T]$ w/ unknown sizes	$\lim_{t\to\infty} \mathbf{P}[T > t]$ w/ known sizes	$\lim_{t\to\infty} \mathbf{P}[T > t]$ w/ unknown sizes
Batch Objective	$\sum_{i=1}^{N} D_i$ w/ known sizes	$\mathbf{E}_\pi\left[\sum_{i=1}^{N} D_i\right]$ w/ unknown sizes	$\sum_{i=1}^{N} e^{\gamma D_i} e^{-\gamma A_i}$ w/ known sizes	$\mathbf{E}_\pi\left[\sum_{i=1}^{N} e^{\gamma D_i} e^{-\gamma A_i}\right]$ w/ unknown sizes
Optimal Policy	**SRPT**	**Gittins**	**Boost**	**GittinsBoost**

All of these are in the Gittins family of policies!

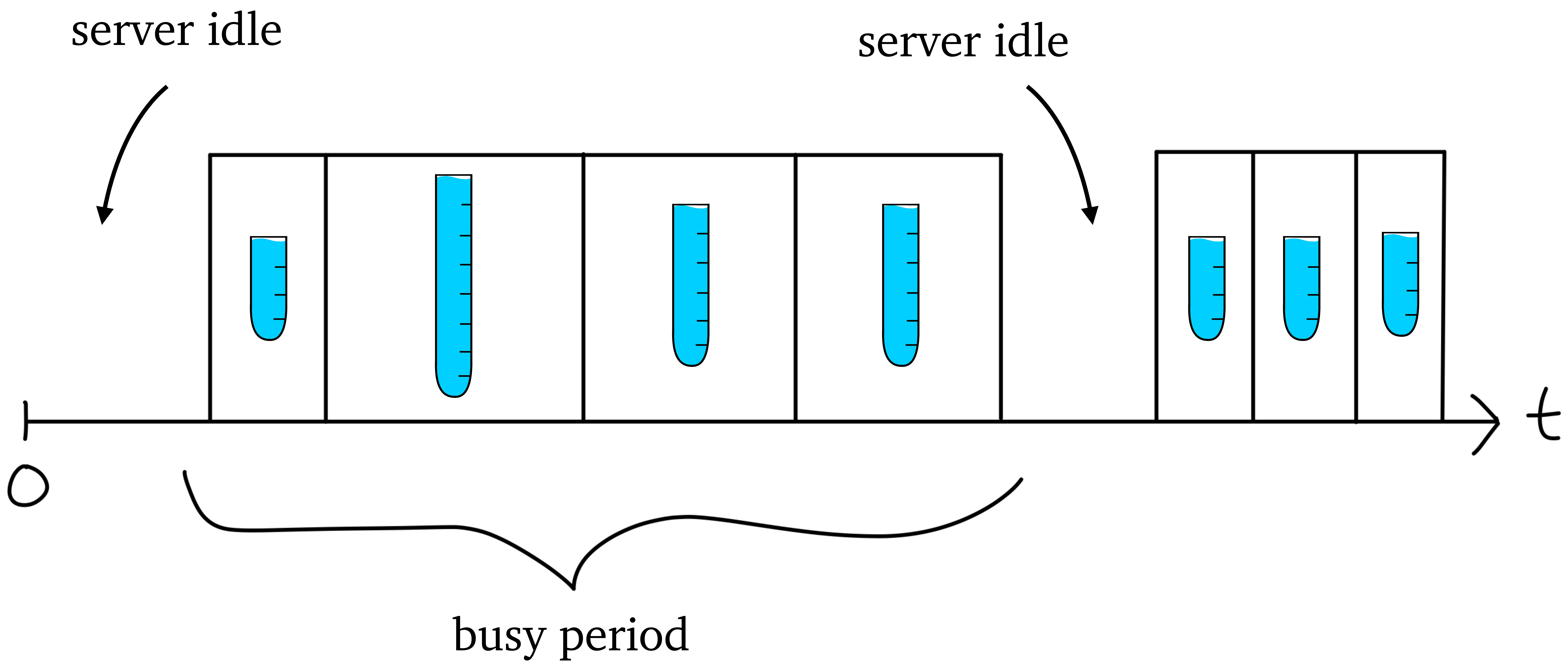
What is the optimal policy for the batch problem?

Queue Objective	$\mathbf{E}_\pi[T]$ w/ known sizes	$\mathbf{E}_\pi[T]$ w/ unknown sizes	$\lim_{t\to\infty} \mathbf{P}[T > t]$ w/ known sizes	$\lim_{t\to\infty} \mathbf{P}[T > t]$ w/ unknown sizes
Batch Objective	$\sum_{i=1}^{N} D_i$ w/ known sizes	$\mathbf{E}_\pi\left[\sum_{i=1}^{N} D_i\right]$ w/ unknown sizes	$\sum_{i=1}^{N} e^{\gamma D_i} e^{-\gamma A_i}$ w/ known sizes	$\mathbf{E}_\pi\left[\sum_{i=1}^{N} e^{\gamma D_i} e^{-\gamma A_i}\right]$ w/ unknown sizes
Optimal Policy	**SRPT**	**Gittins**	**Boost**	**GittinsBoost**

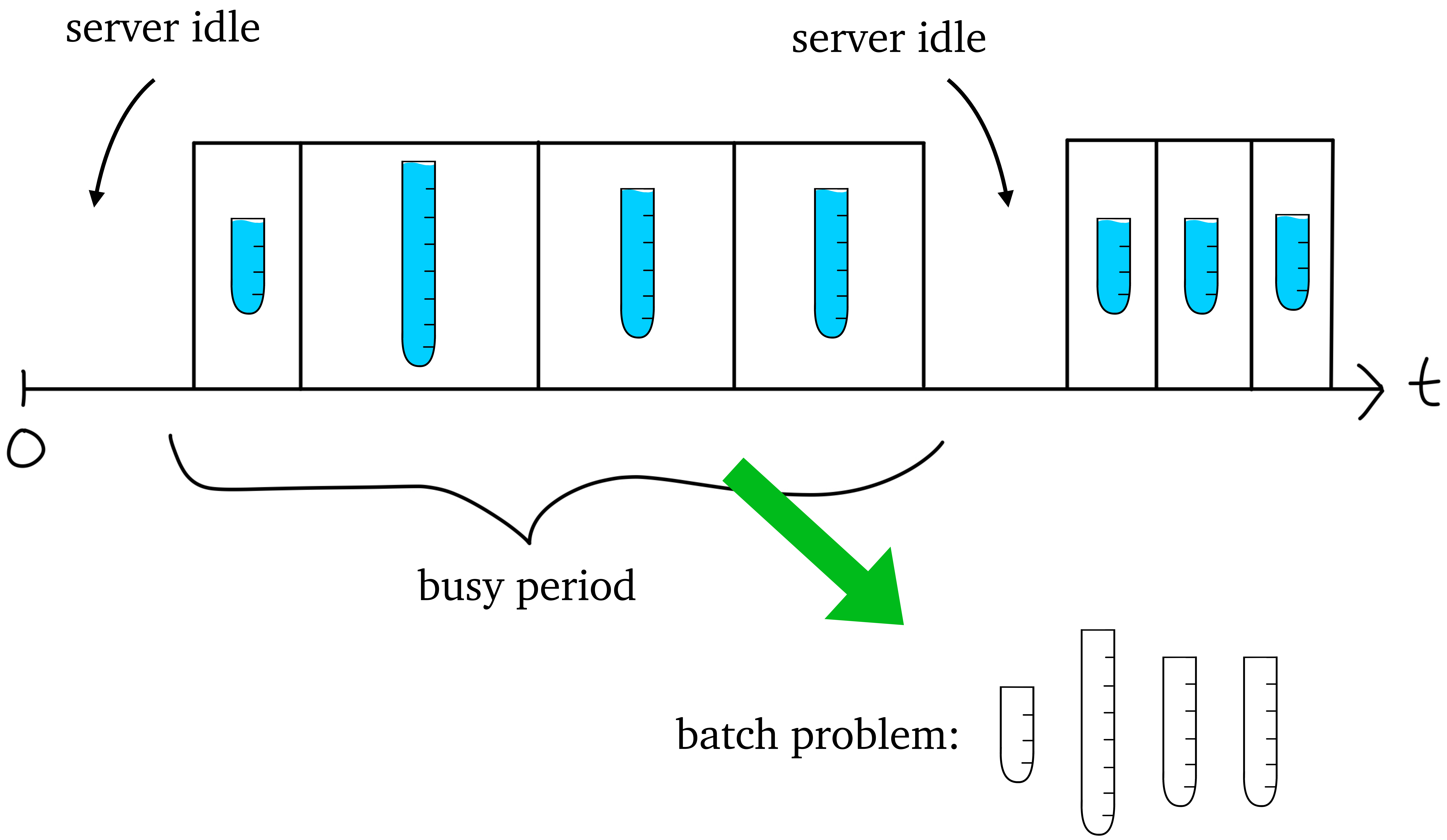
All of these are in the Gittins family of policies!

How do we show optimality in the queue setting?

Boost optimality in the queue setting

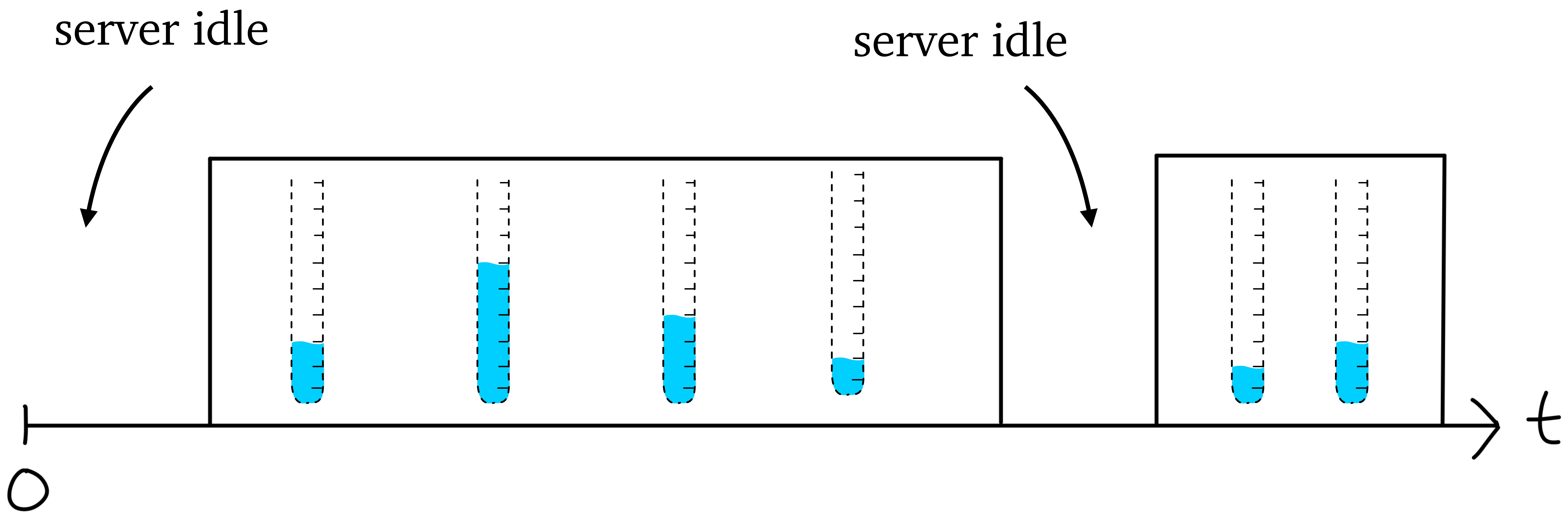


Boost optimality in the queue setting

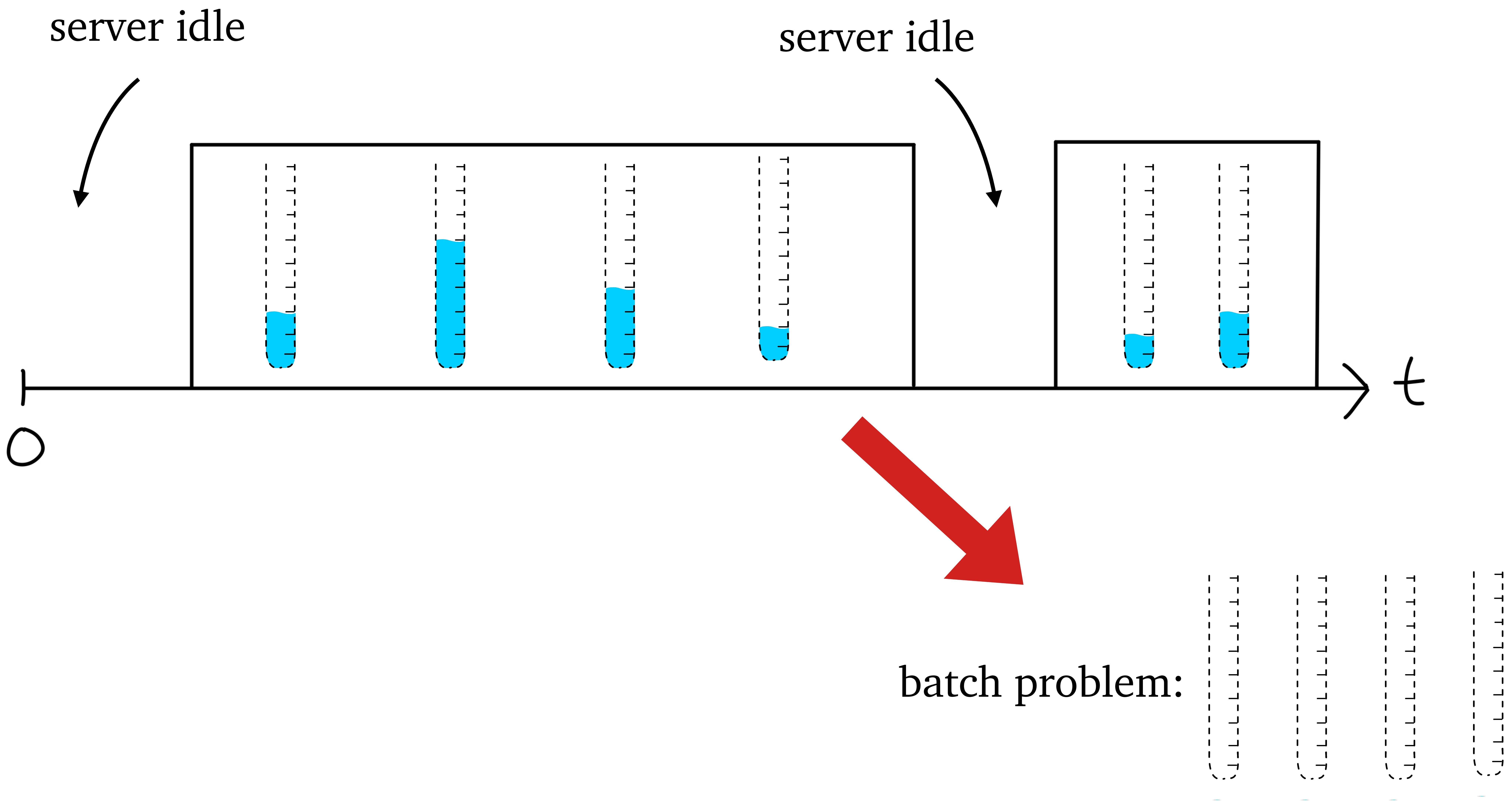


GittinsBoost optimality in the queue setting

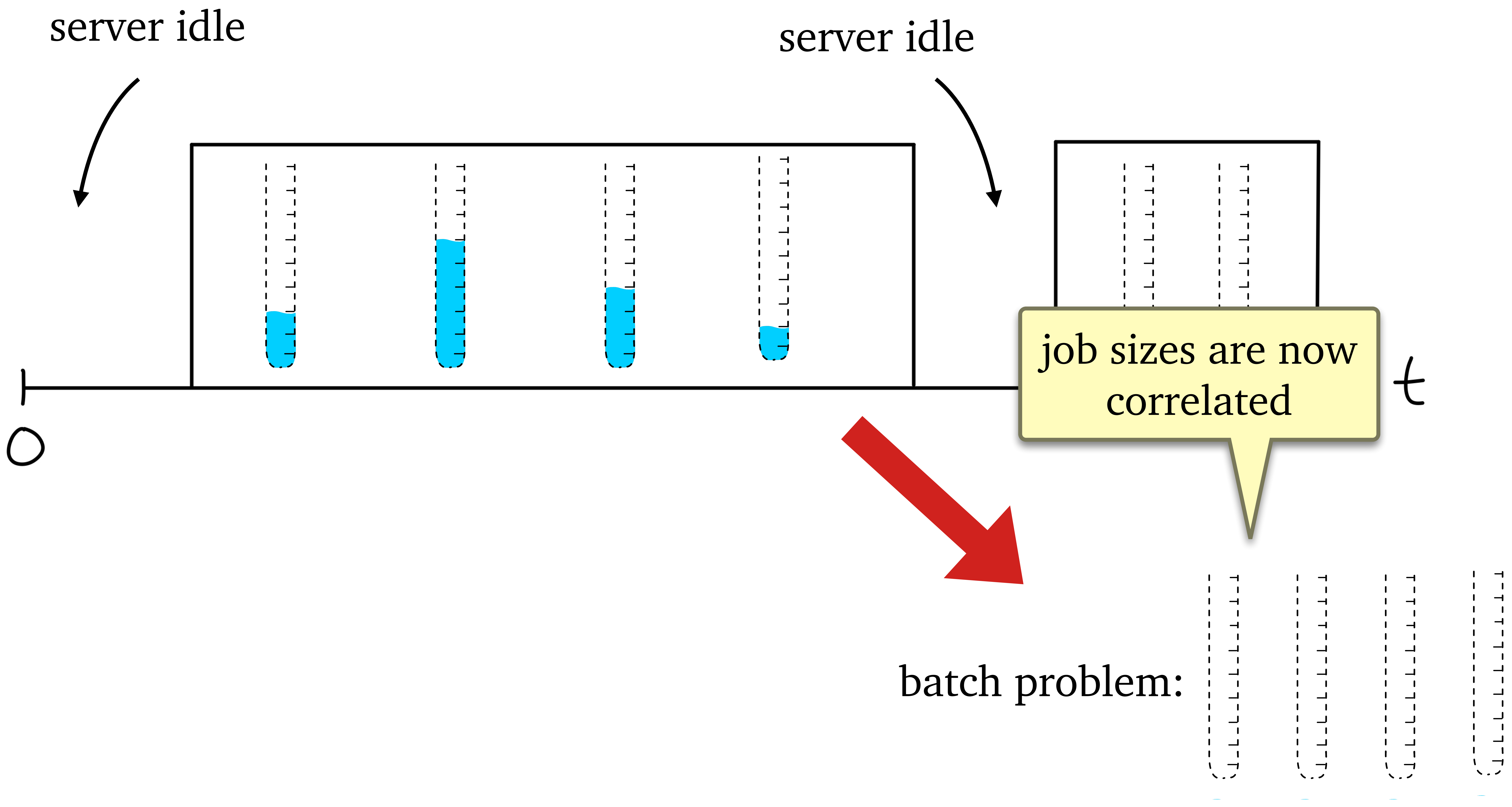
GittinsBoost optimality in the queue setting



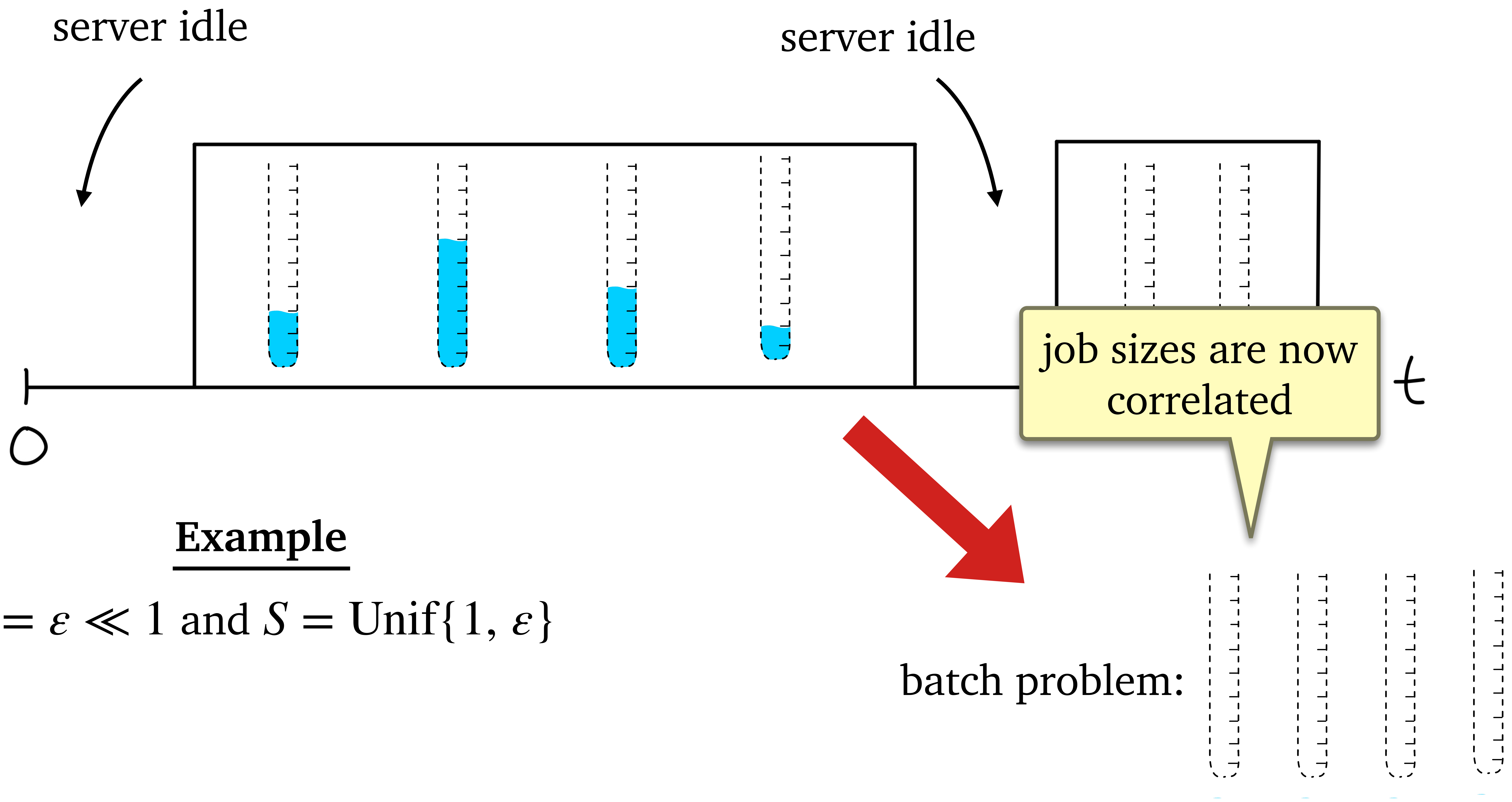
GittinsBoost optimality in the queue setting



GittinsBoost optimality in the queue setting



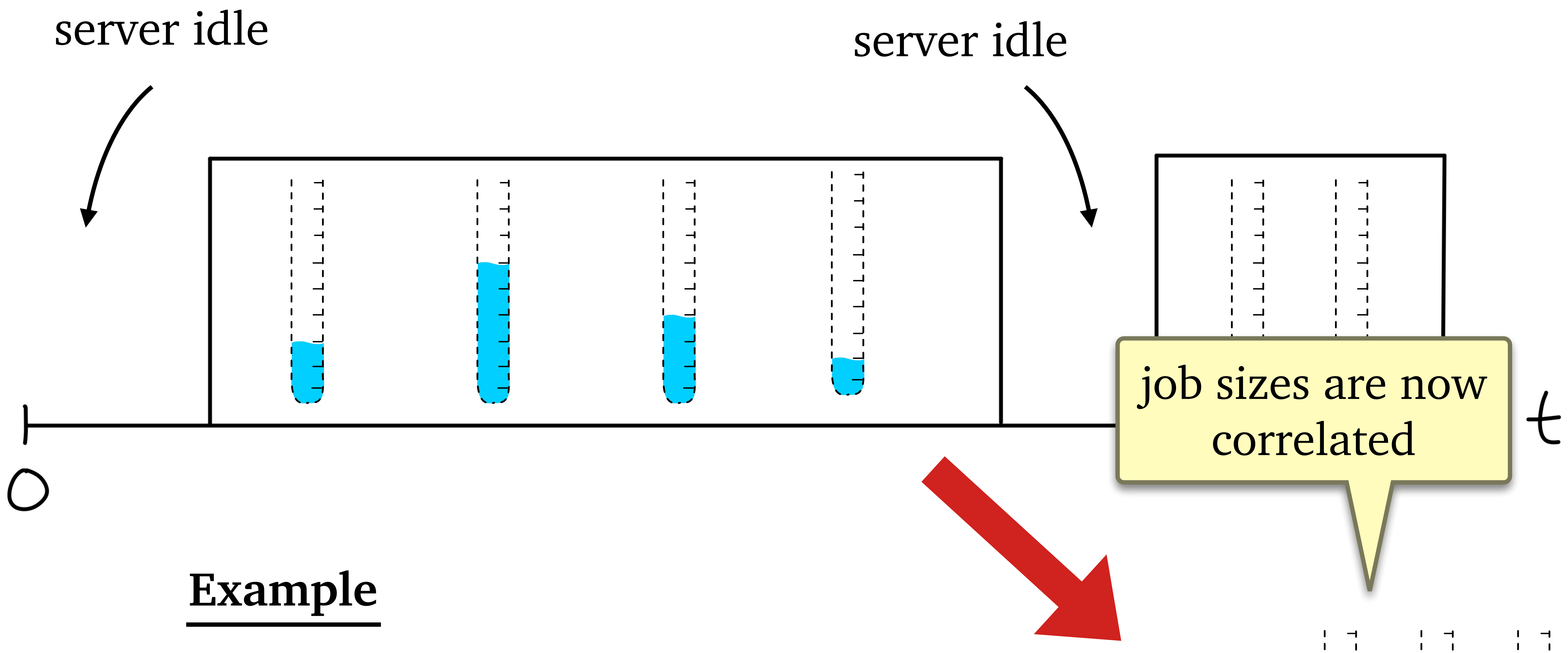
GittinsBoost optimality in the queue setting



Example

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

GittinsBoost optimality in the queue setting

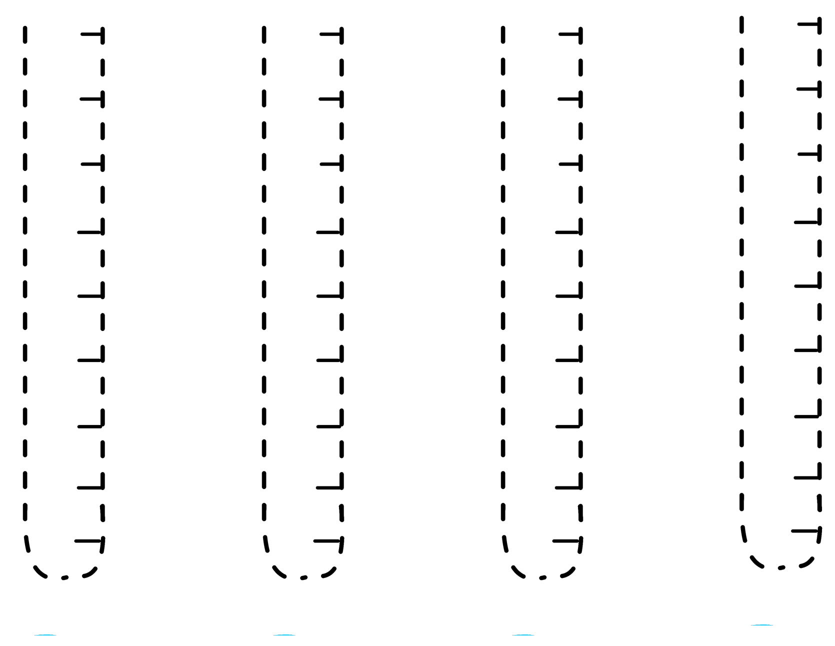


Example

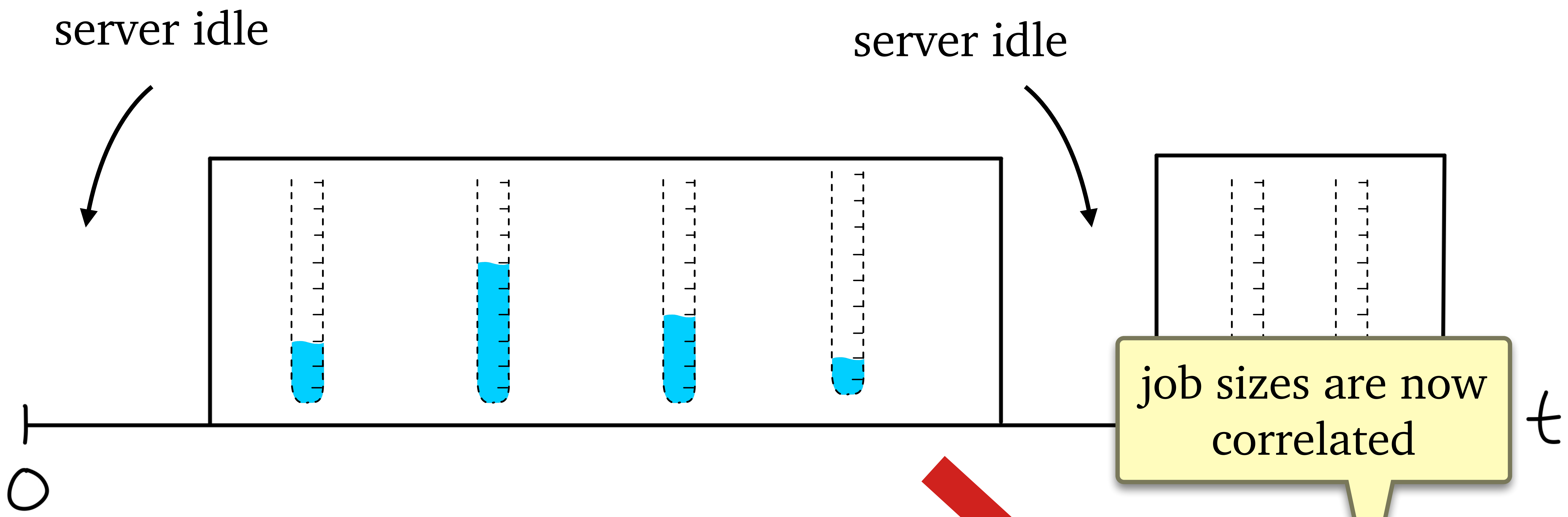
$\lambda = \varepsilon \ll 1$ and $S = \text{Unif}\{1, \varepsilon\}$

3 jobs: $A_1 = 0$, $A_2 = \varepsilon^2$, $A_3 = 1$

batch problem:



GittinsBoost optimality in the queue setting

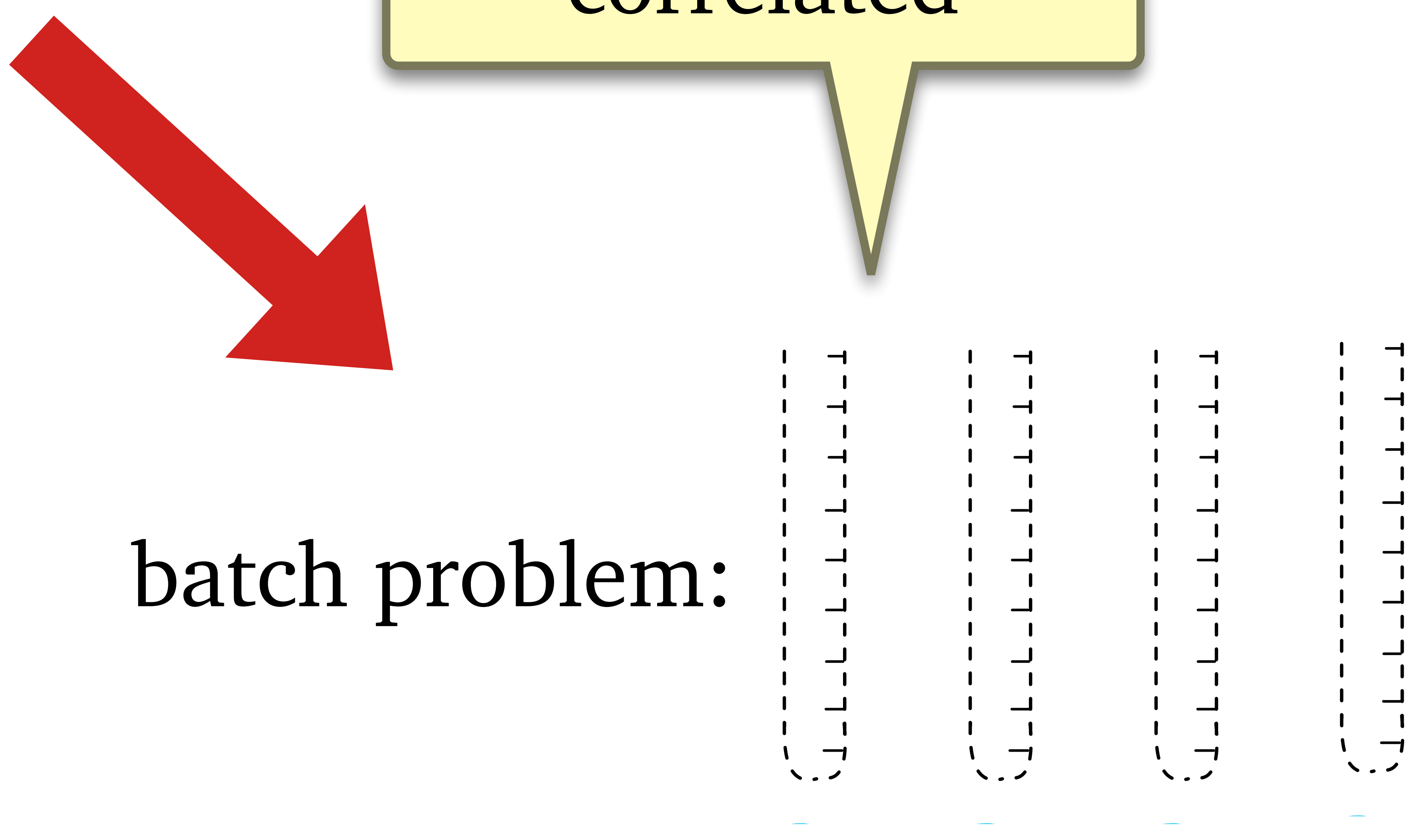


Example

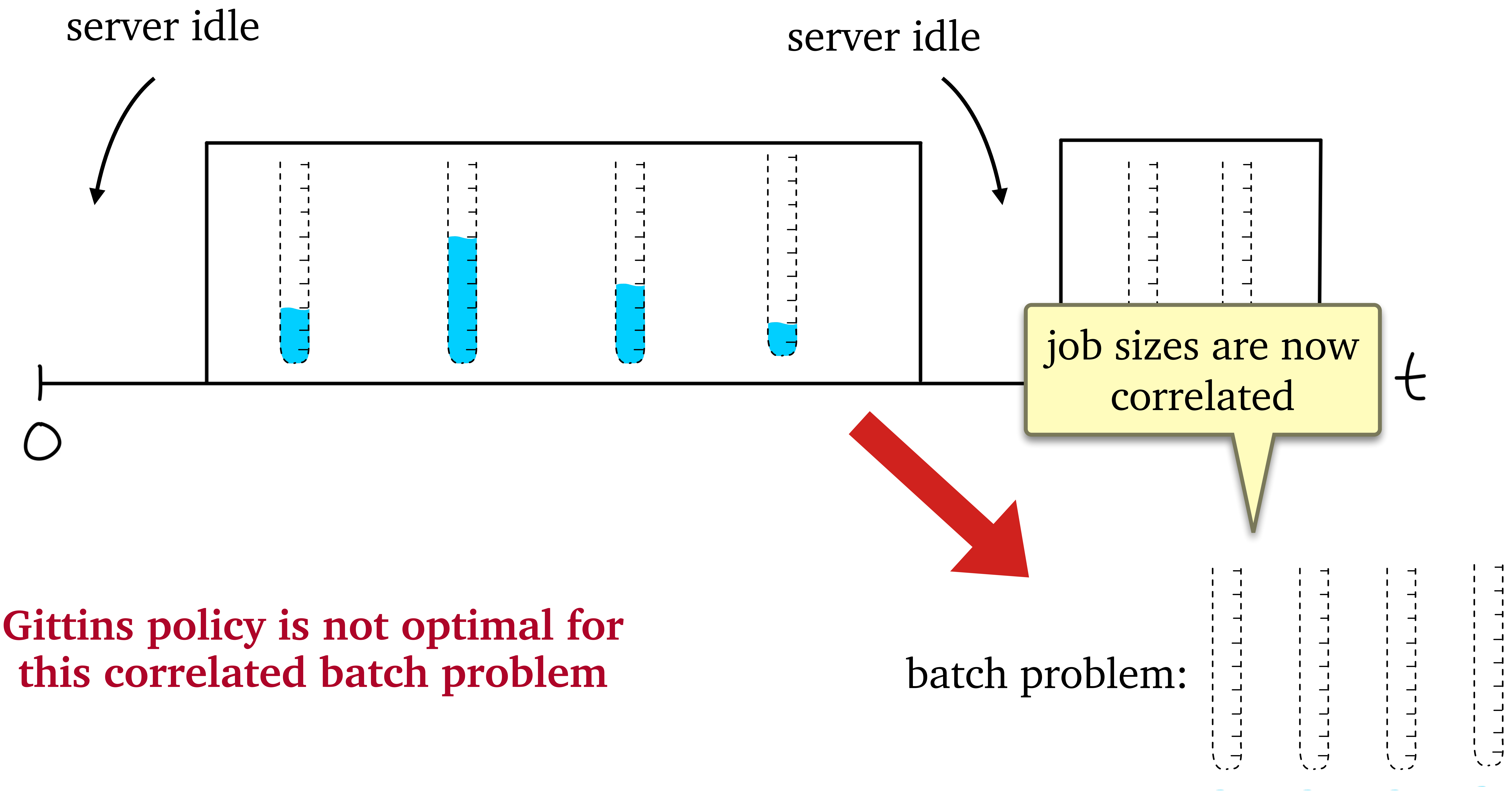
$\lambda = \varepsilon \ll 1$ and $S = \text{Unif}\{1, \varepsilon\}$

3 jobs: $A_1 = 0$, $A_2 = \varepsilon^2$, $A_3 = 1$

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

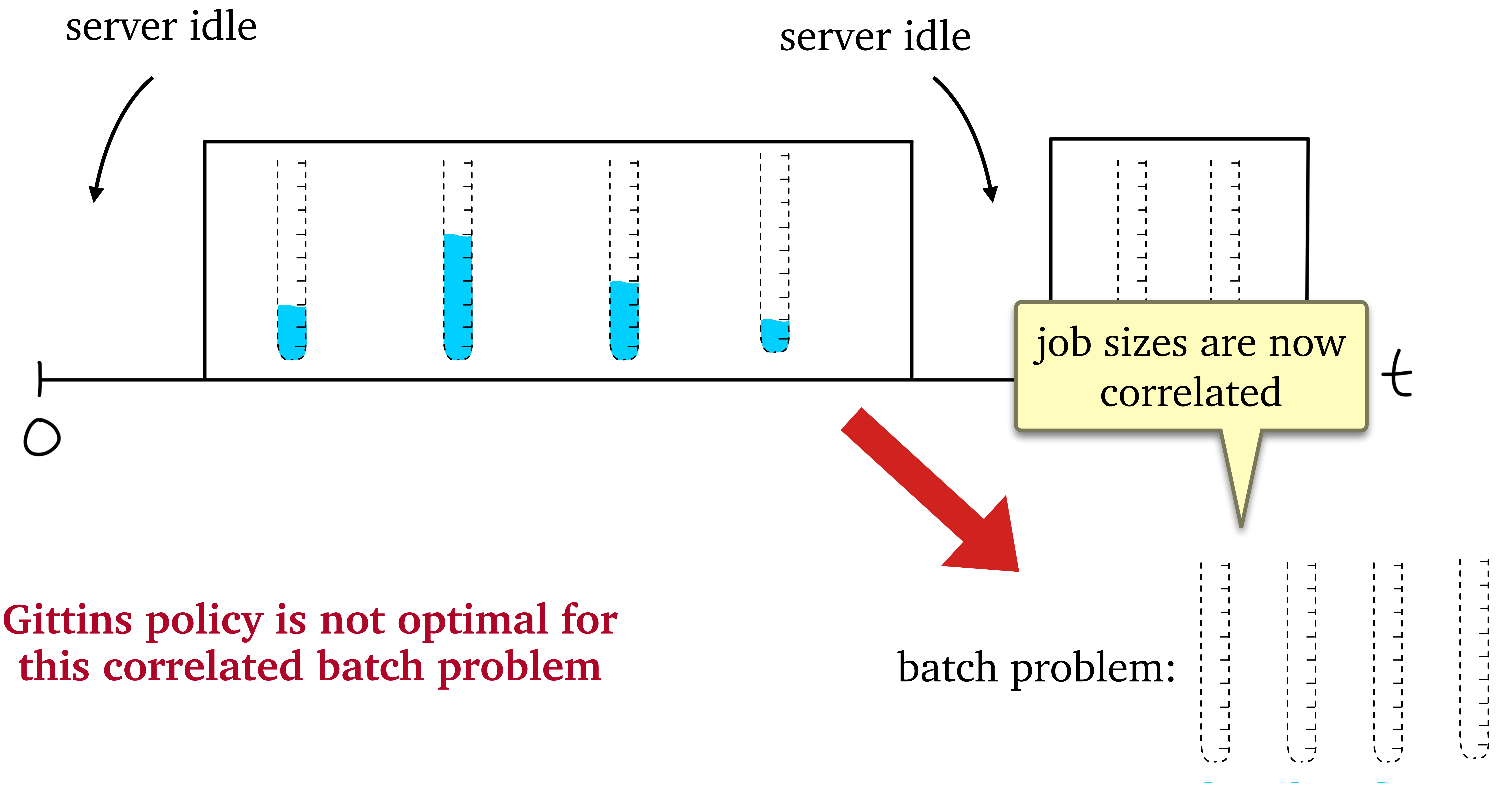


GittinsBoost optimality in the queue setting



Gittins policy is not optimal for this correlated batch problem

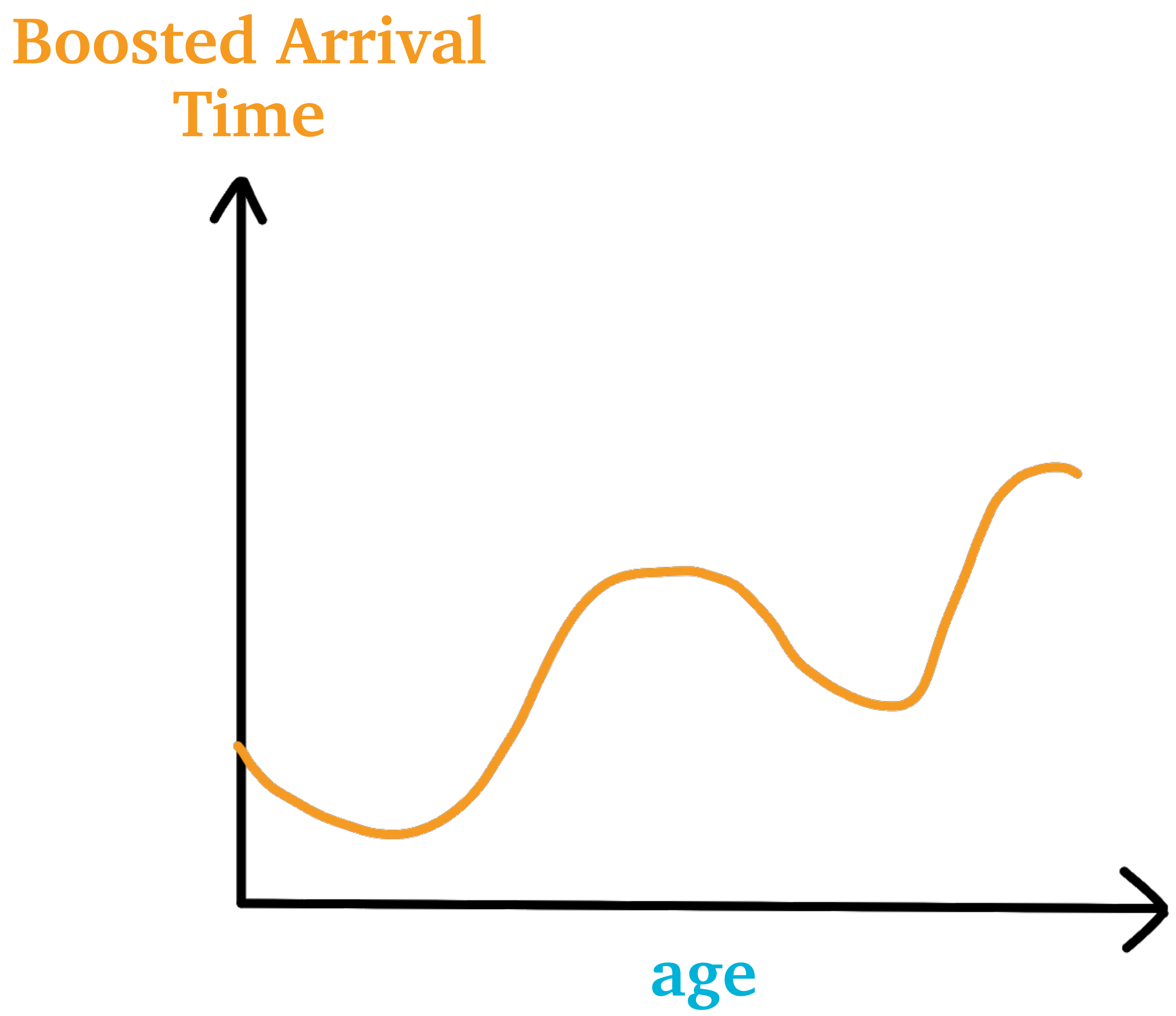
GittinsBoost optimality in the queue setting



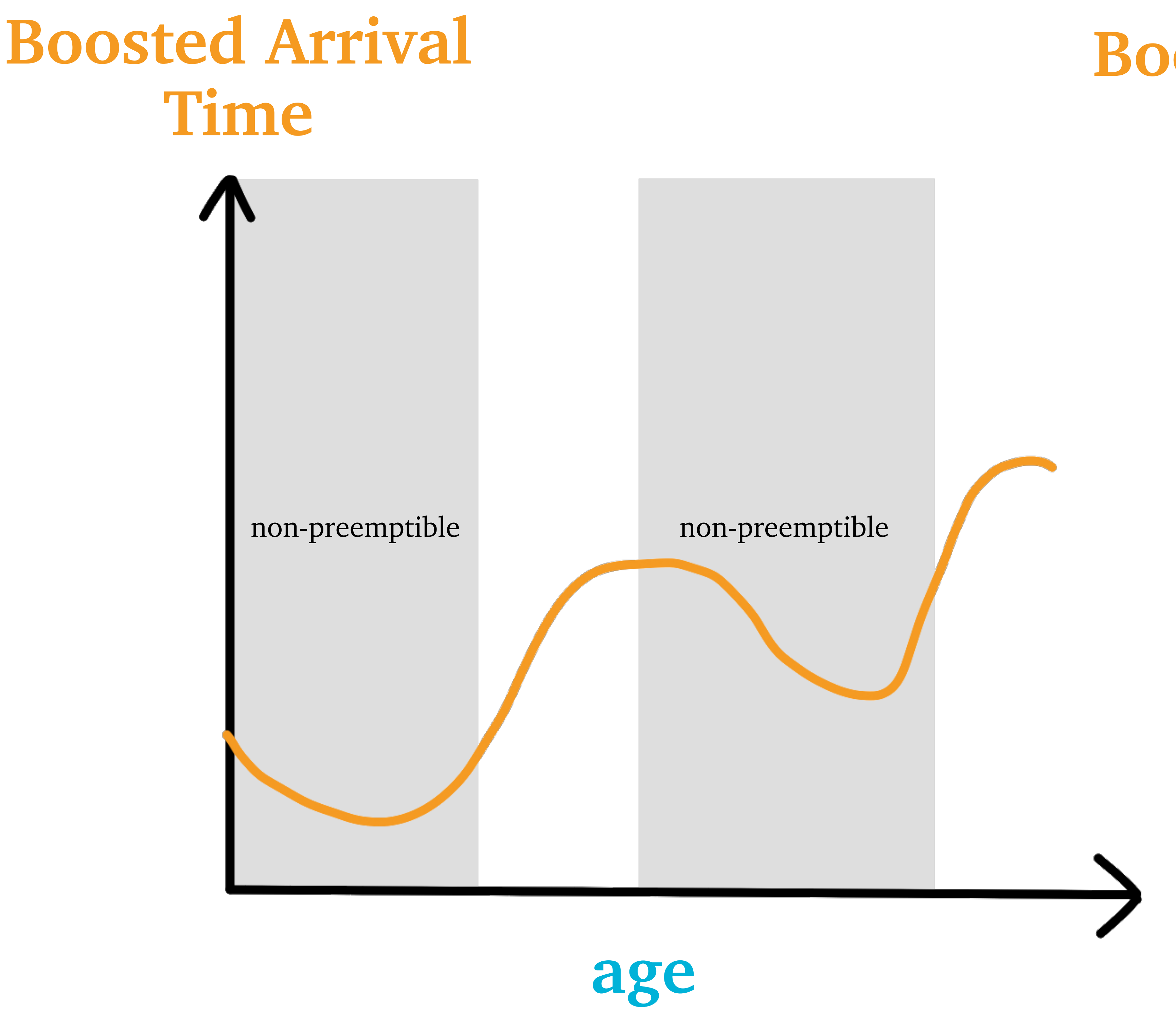
Gittins policy is not optimal for this correlated batch problem

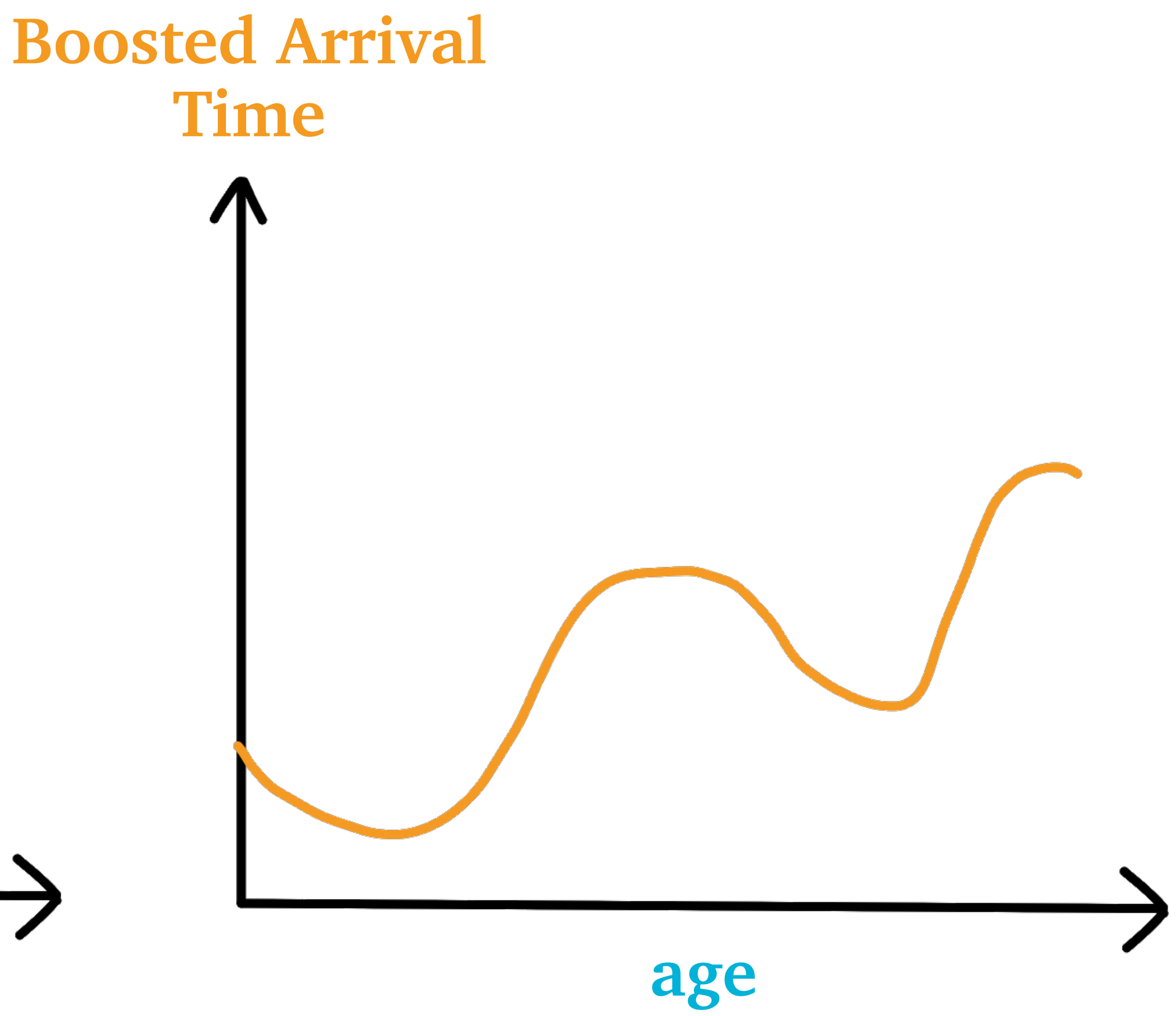
***main technical challenge*: showing optimality in queue setting**

What was our approach?

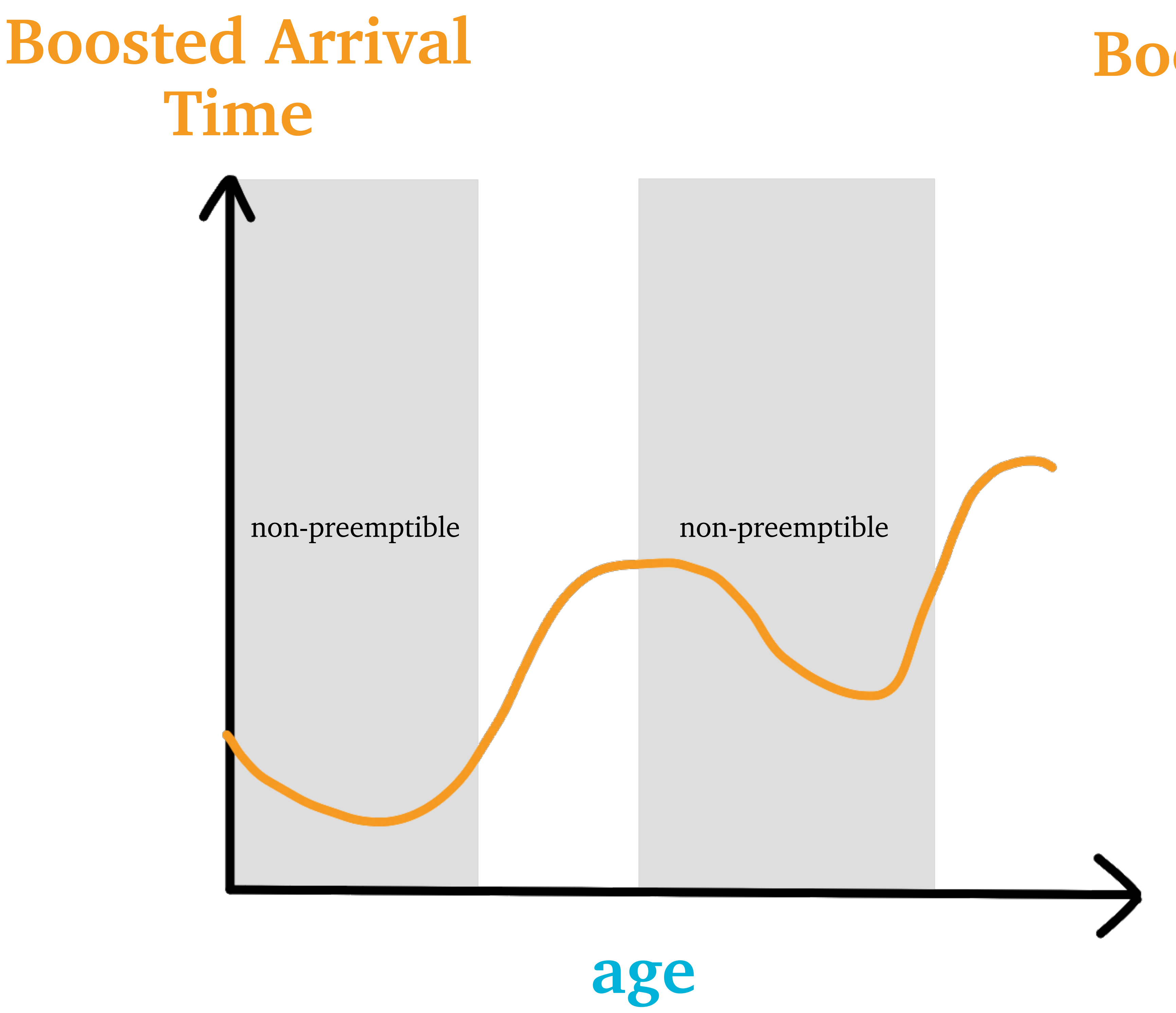


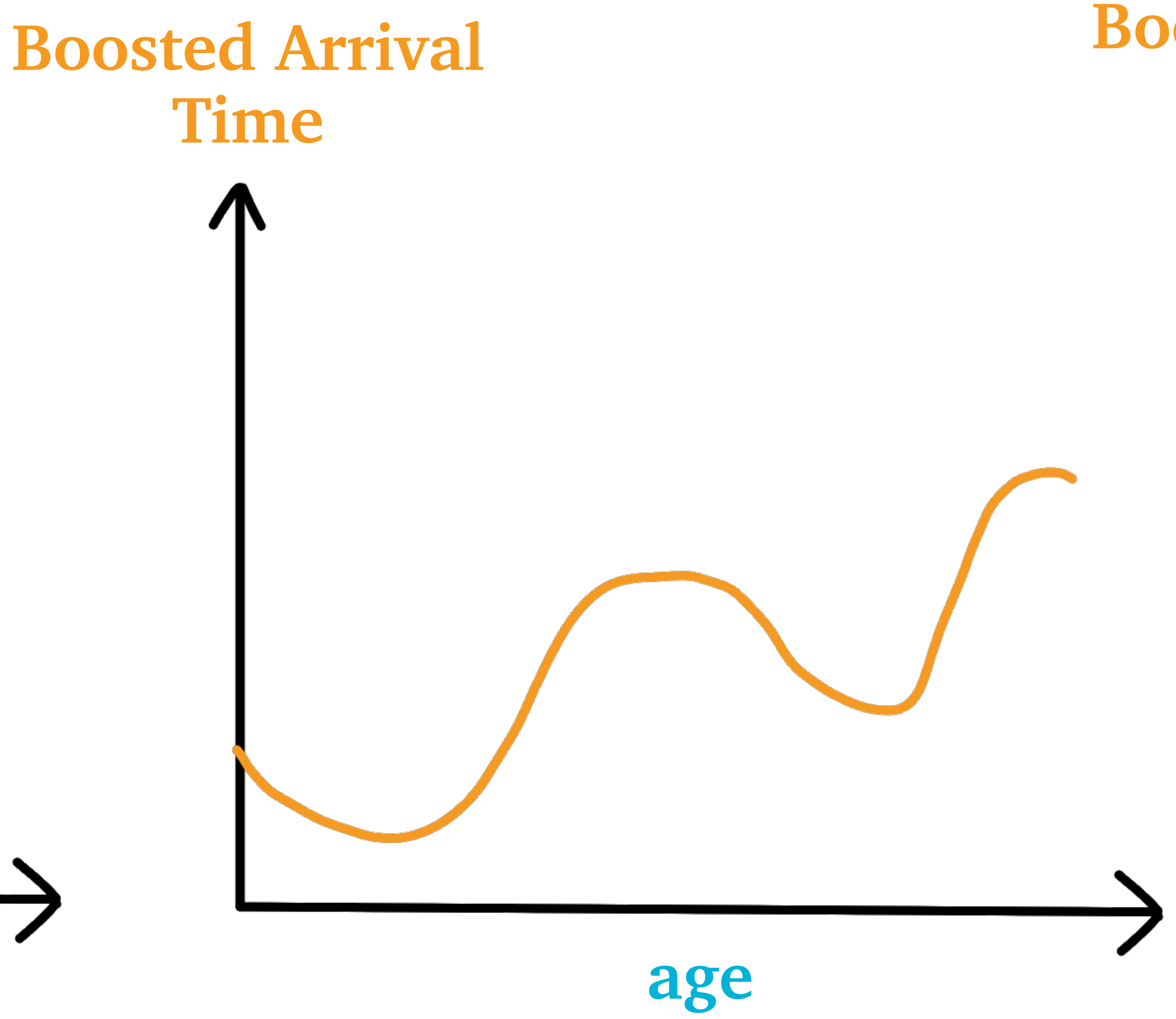
What was our approach?

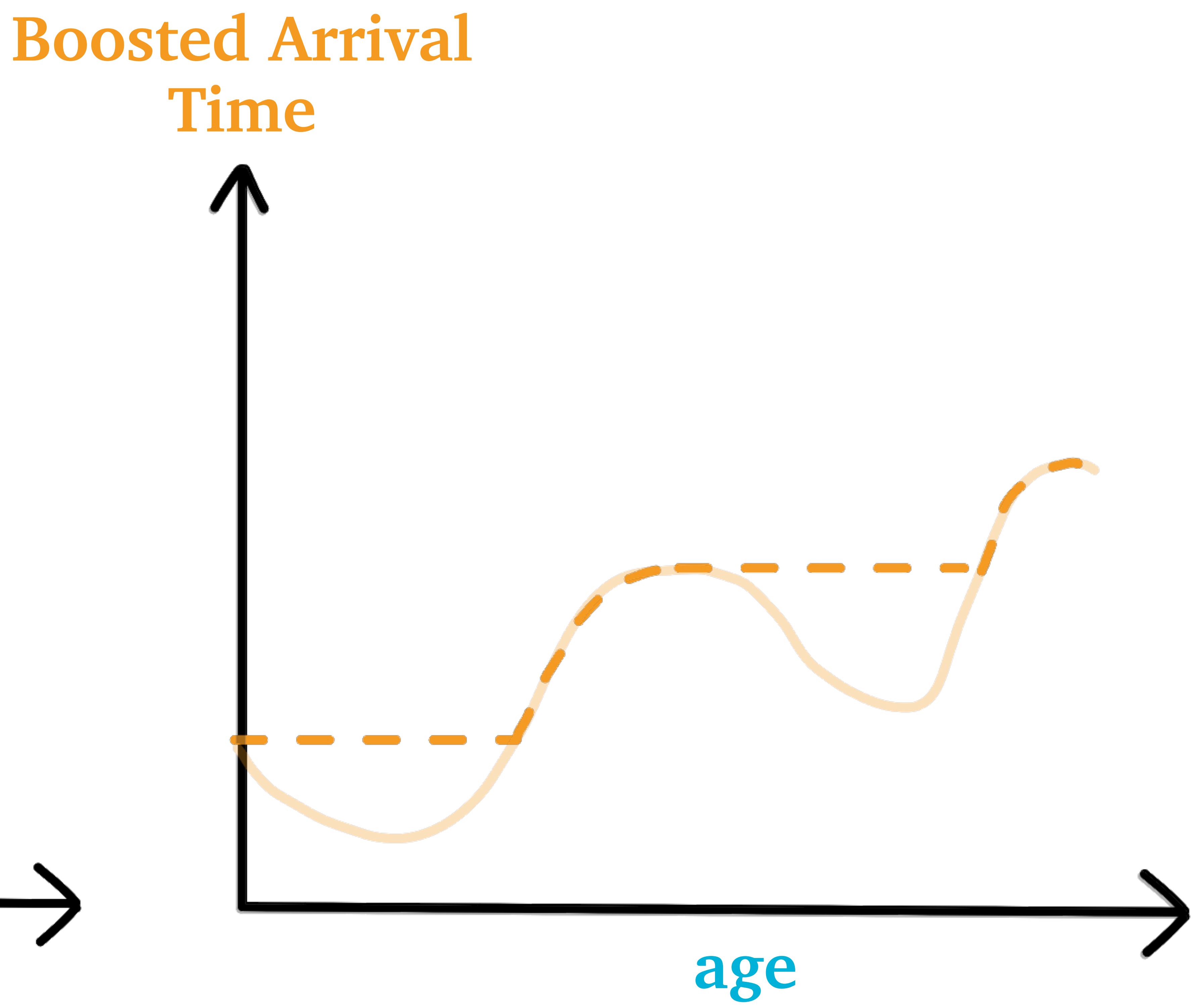




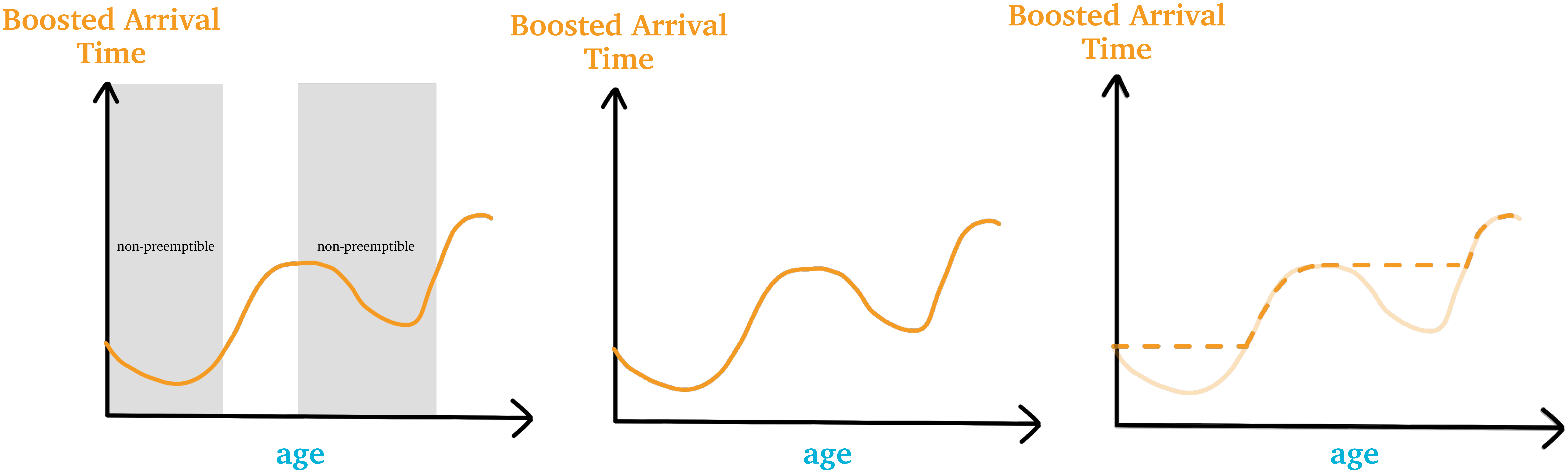
What was our approach?





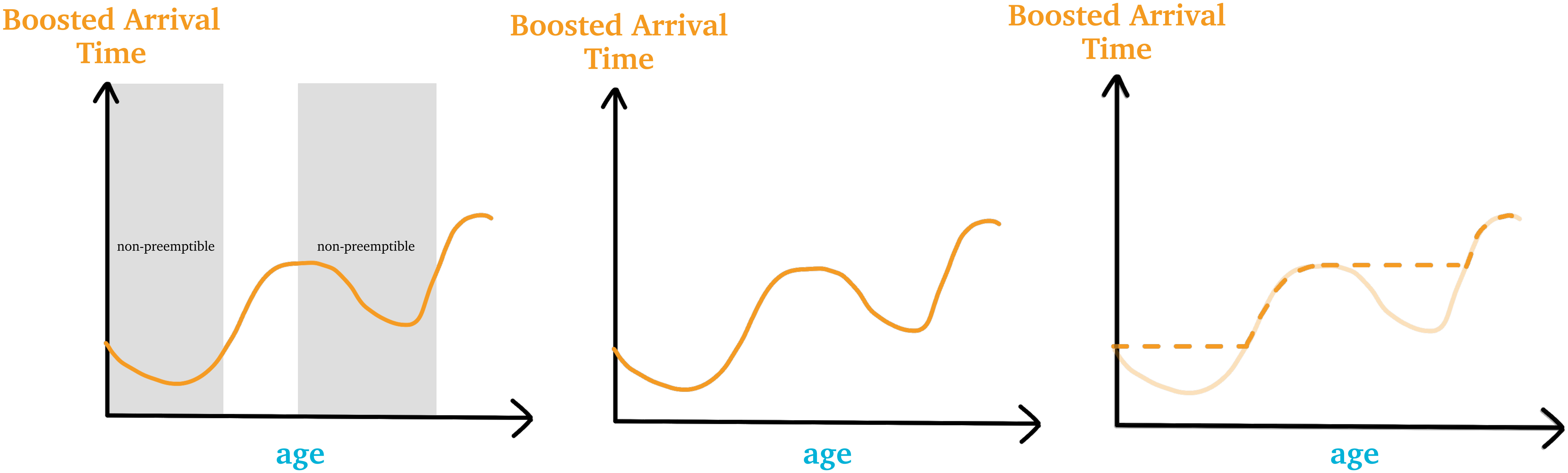


What was our approach?



Batch Setting Optimality: all three policies are the same

What was our approach?



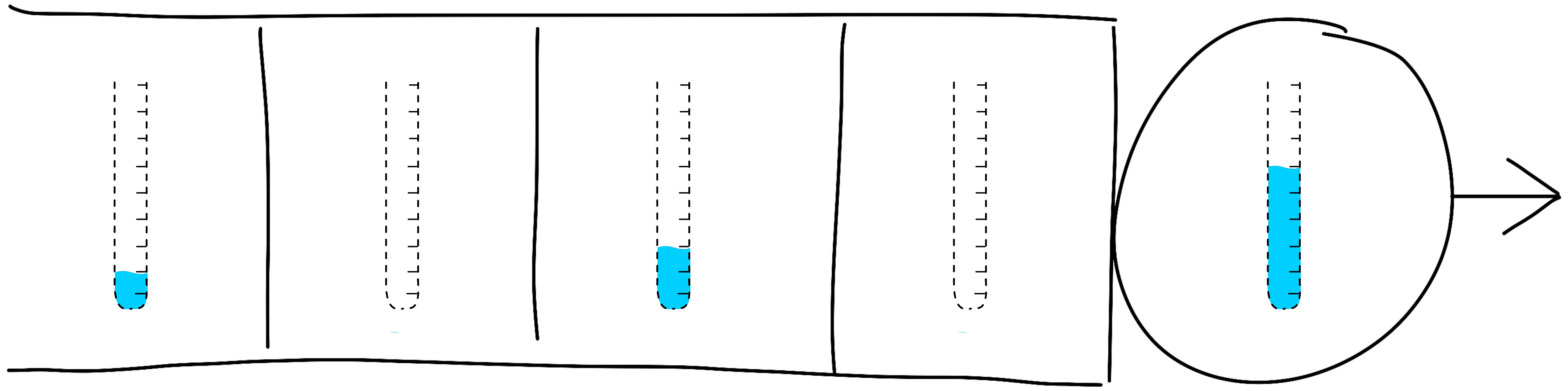
Batch Setting Optimality: all three policies are the same

Queue Setting Optimality: all three policies have the same asymptotic tail behavior

Summary

Summary

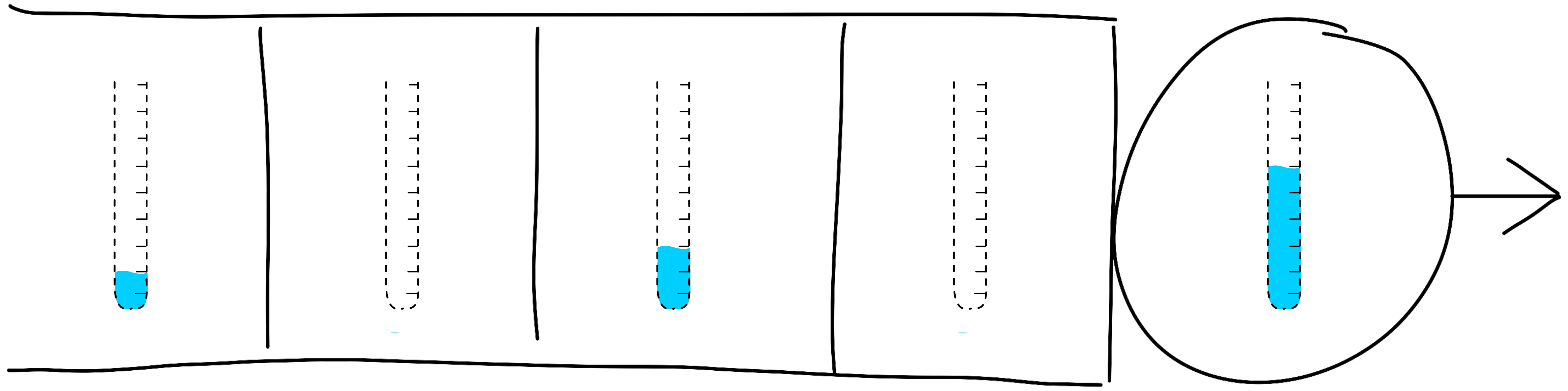
Problem



Schedule for $\mathbf{P}[T > t]$ as $t \to \infty$

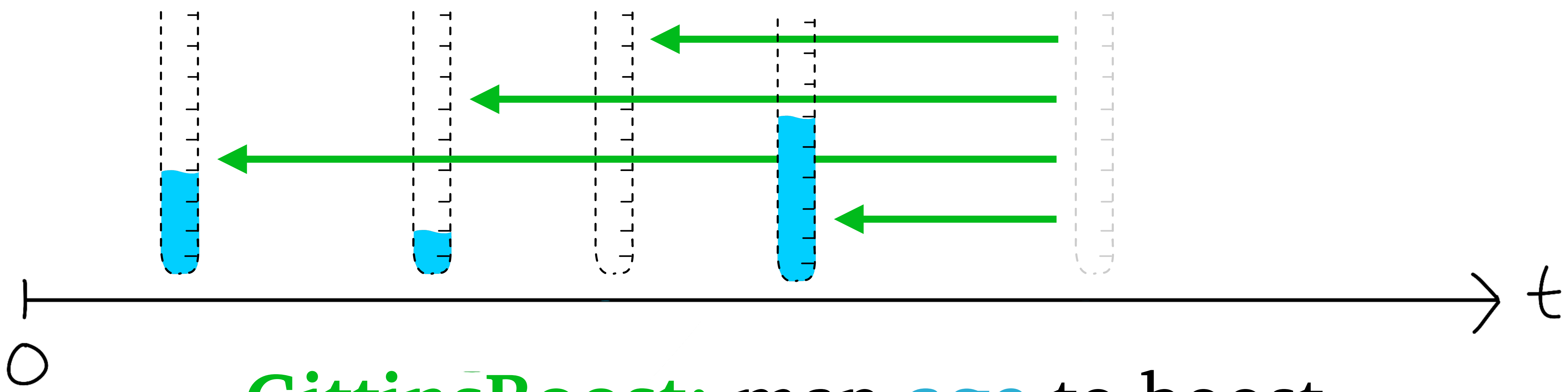
Summary

Problem



Schedule for $\mathbf{P}[T > t]$ as $t \to \infty$

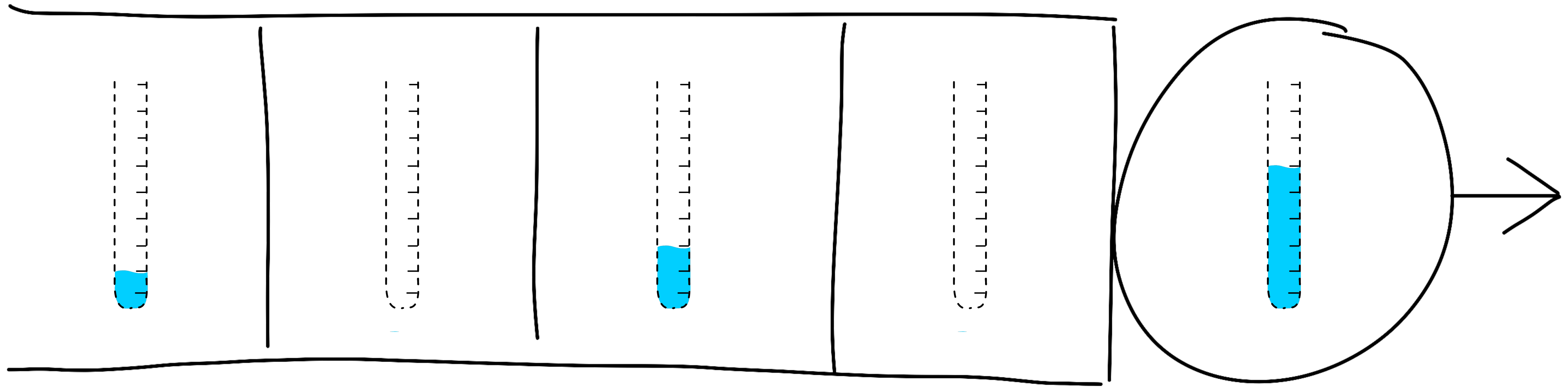
Contribution



GittinsBoost: map **age** to boost

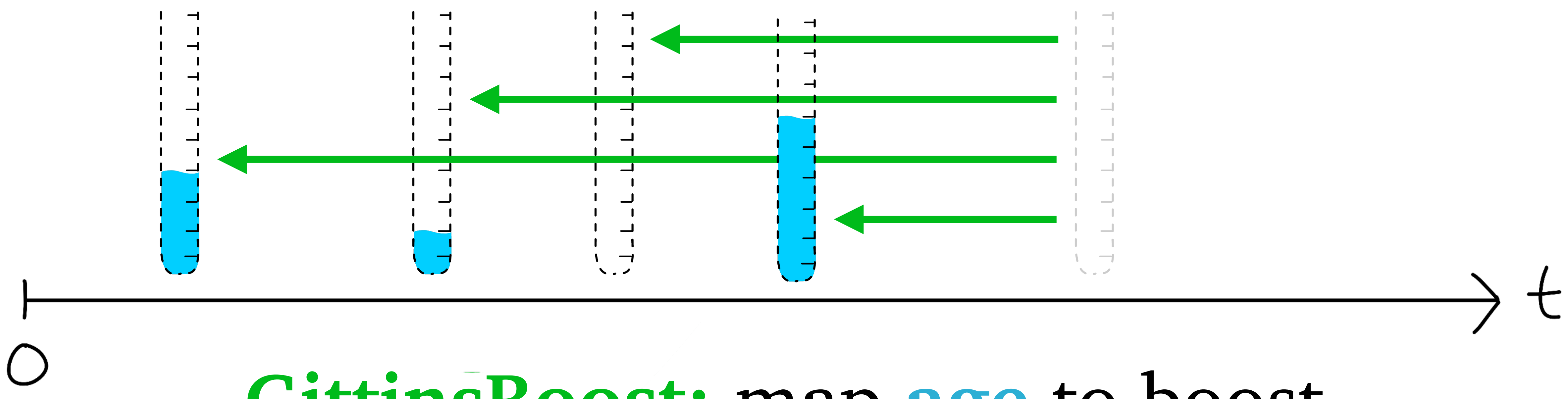
Summary

Problem



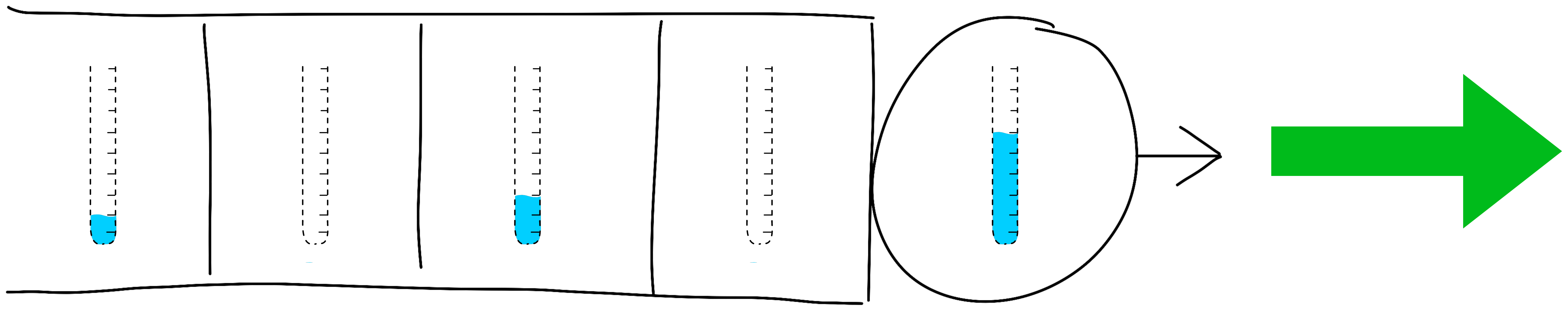
Schedule for $\mathbf{P}[T > t]$ as $t \to \infty$

Contribution

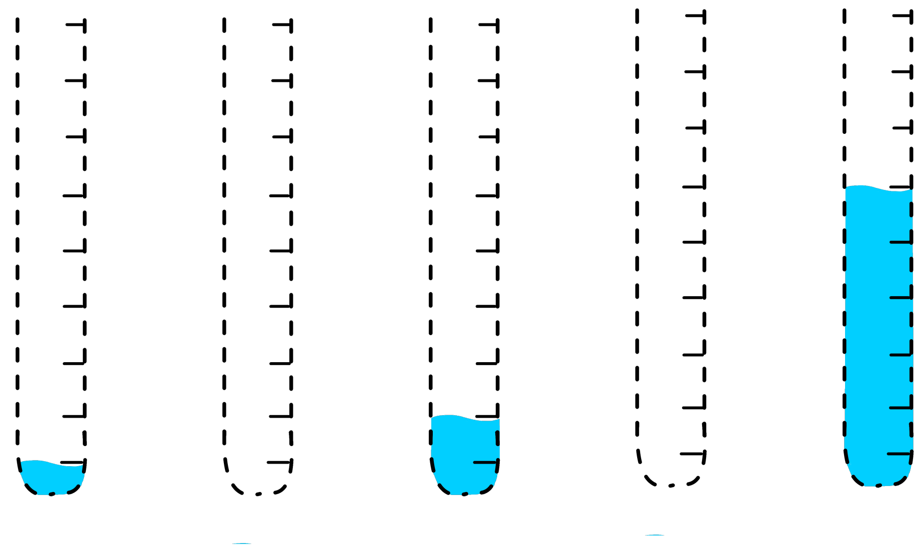


GittinsBoost: map **age** to boost

Main Ideas

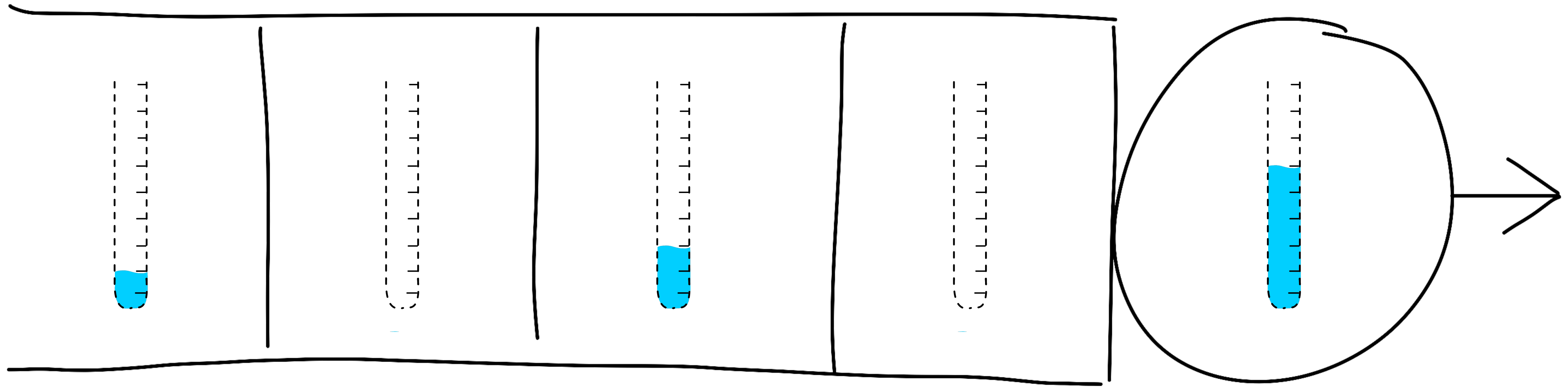


batch problem:



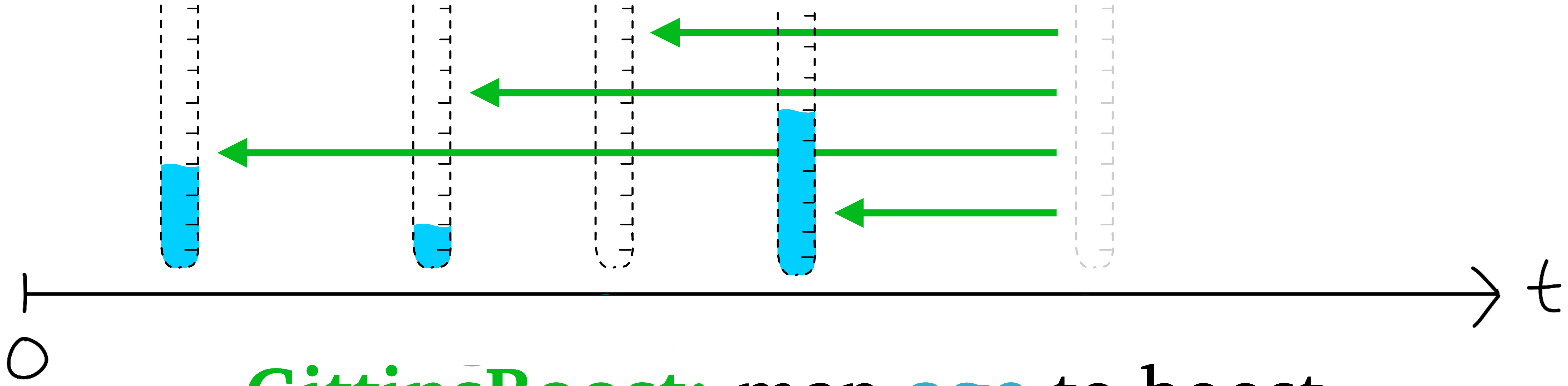
Summary

Problem



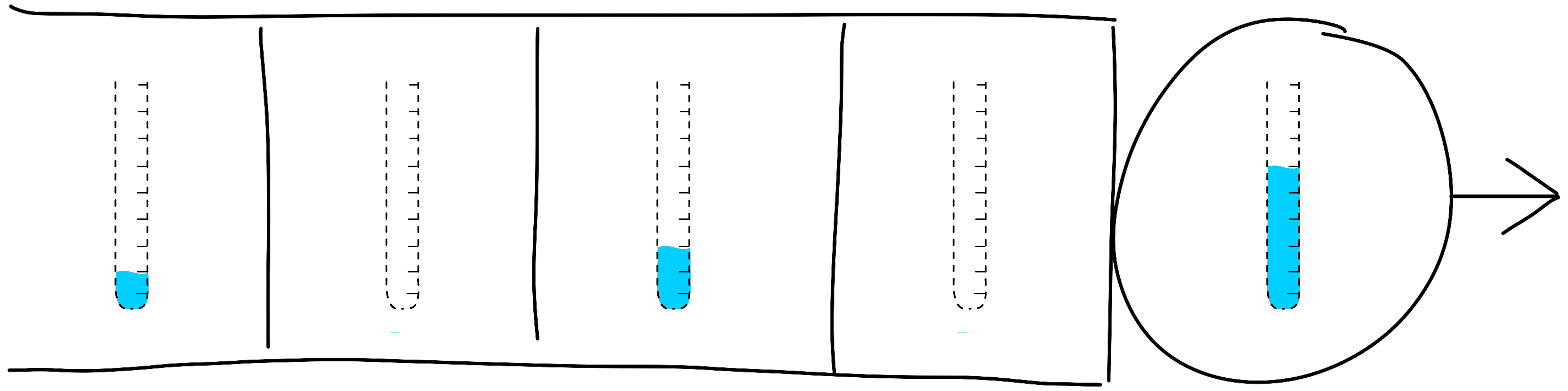
Schedule for $\mathbf{P}[T > t]$ as $t \to \infty$

Contribution

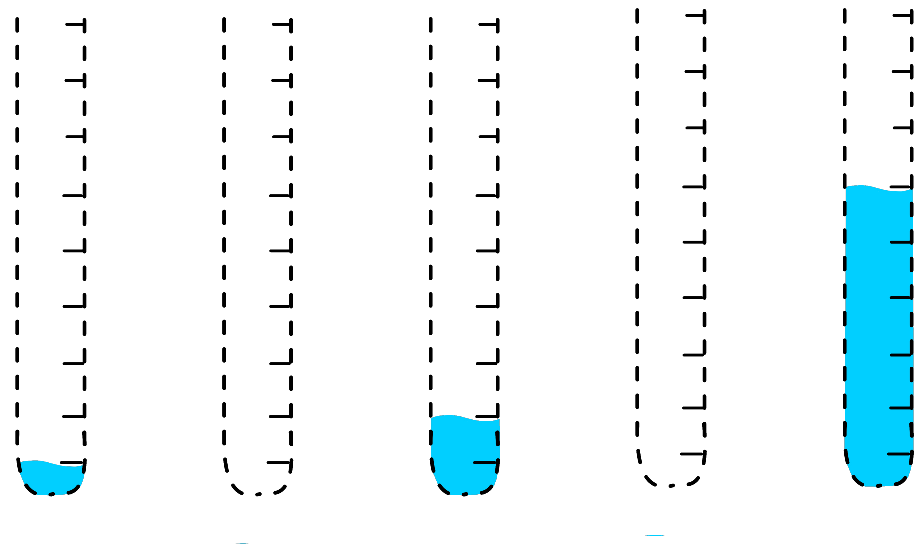


GittinsBoost: map **age** to boost

Main Ideas



batch problem:



queue optimality ← batch optimality

main technical challenge