Online Algorithms

Admin

Exam Review Session led by Daphne:
Wednesday April 24th, 7-9pm, BBB 1670.

HW11 is due Tuesday 4/23 at 8pm.

Reminder: Filling out the course evaluations is 1% of your grade, which is otherwise covered by the final exam. Remember to submit receipt on Gradescope.

It’s the end of 376 as we know it

If you enjoyed this course:

Fall ‘24:

475: Introduction to Cryptography
477: Introduction to Algorithms
498: Advanced Data Structures
498/598: Algorithms for Machine Learning and Data Science
572: Randomness and Computation
575: Advanced Cryptography
598: Machine Learning Theory

Winter '25:

475: Introduction to Cryptography
477: Introduction to Algorithms
574: Computational Complexity Theory
598: Machine Learning Theory
598: Graph Algorithms (taught by me)

Also, consider applying to become an IA for 376

*recently some courses were renamed “CSE” instead of “EECS” so try both names
**When registering for grad-level courses, uncheck the box for “open courses only”

It’s the end of 376 as we know it

If you really enjoyed this course:

Consider trying research in theoretical computer science (TCS)!

What is research?

Pushing the boundaries of human knowledge.

What is TCS research?

Well, it’s pretty similar to the experience of working on 376 HW …except you’re working on the same problem for months/years and nobody knows the solution.

How to get involved in research:

- Research programs at UM: cse.engin.umich.edu/academics/undergraduate/undergraduate-resources-and-opportunities/undergraduate-research/
- Apply to **REU** programs
- Look up the research of professors at UM and reach out to them

A puzzle game made by UM alums!

Online Algorithms

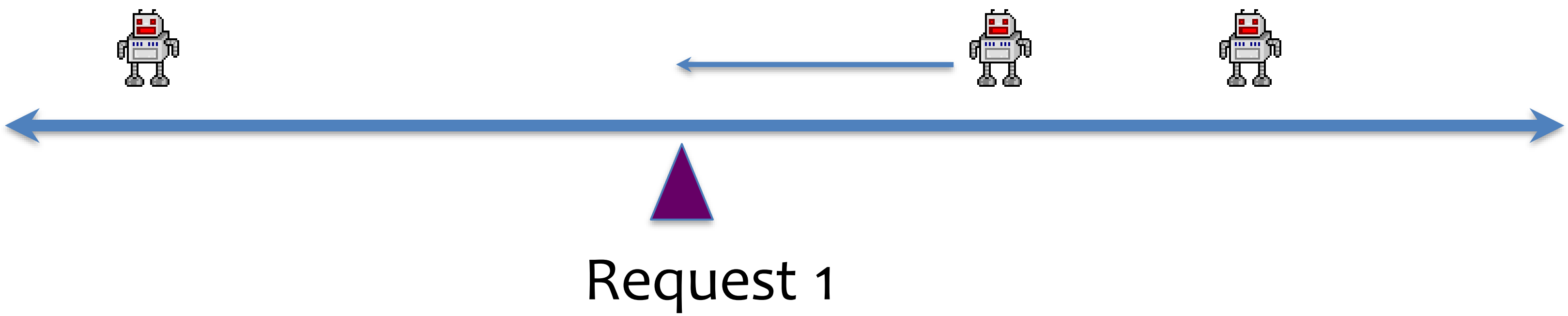
The input is revealed over time.

The algorithm needs to make decisions with partial information!

E.g. ridesharing, stock market, caching, …

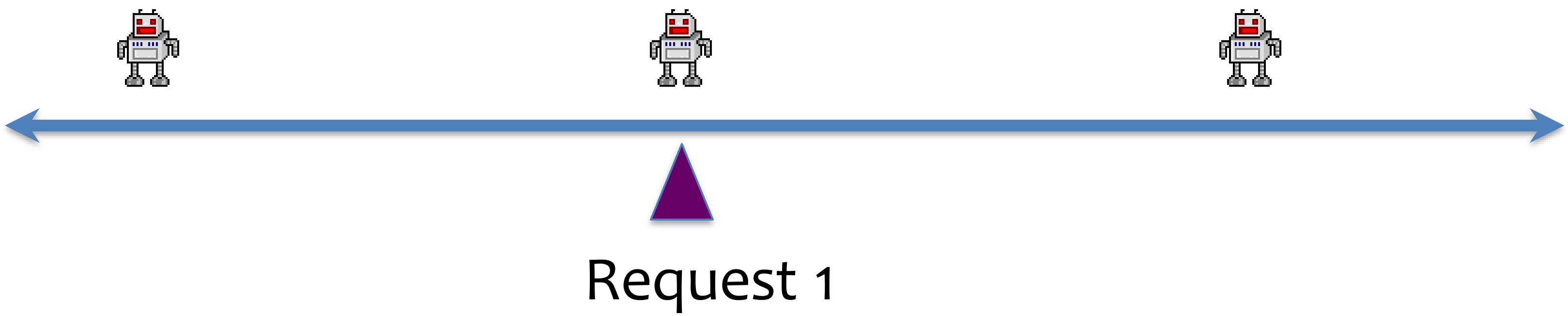
A classic problem in online algorithms:

The k-Server Problem



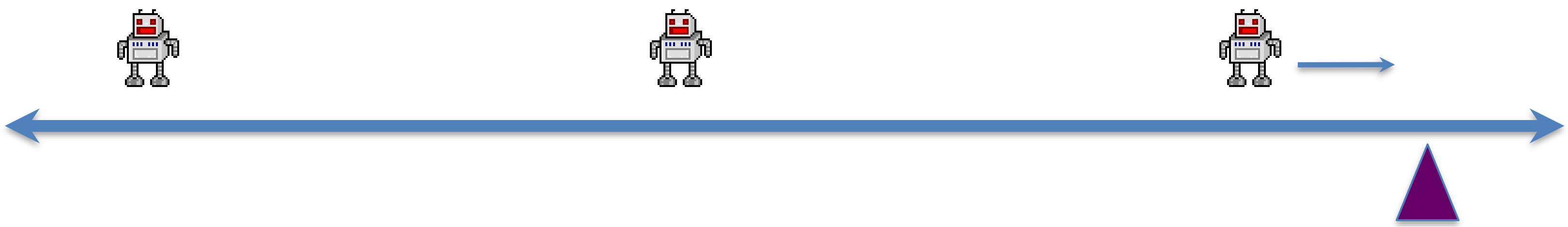
A classic problem in online algorithms:

The k-Server Problem



A classic problem in online algorithms:

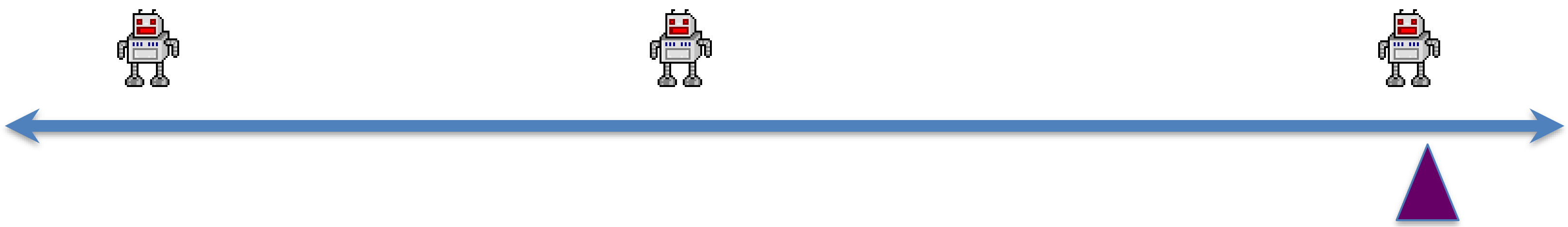
The k-Server Problem



Request 2

A classic problem in online algorithms:

The k-Server Problem



Request 2

Assumptions:

- *k* identical robots. Any robot can service any request.
- Requests arrive one-at-a-time. A request must be serviced before the next request arrives.

Goal: Minimize total travel distance of all robots

Competitive Ratio

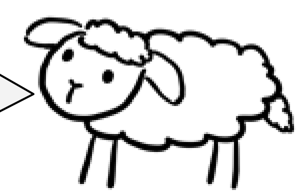
We measure the quality of an online algorithm by its ***competitive ratio***.

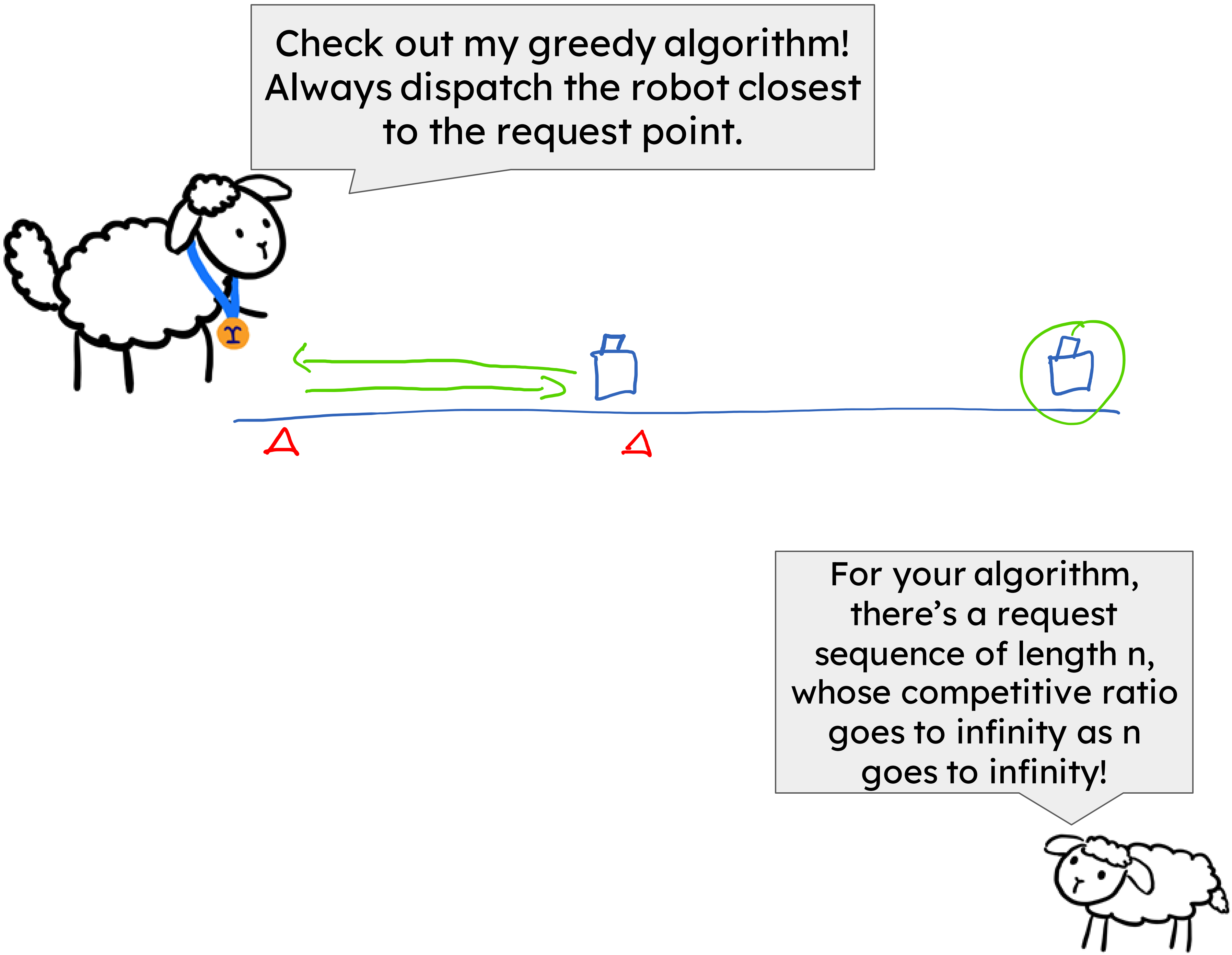
An online algorithm for the k-Server problem is **c*-competitive*** if its cost is *<u>at most</u>* **c** times the cost of an optimal ***offline*** algorithm i.e. an algorithm that *knows the entire request sequence in advance.*

$$\underbrace{\mathbf{ALG}}_{\text{Cost of our algorithm}} \leq \mathbf{c} \cdot \underbrace{\mathbf{OPT}}_{\text{Cost of optimal offline algorithm (can see future)}}, \quad \mathbf{c} > 1$$

c is called the ***competitive ratio*** (smaller is better here).

This is similar to approximation algorithms, but here OPT can see the future and ALG can't



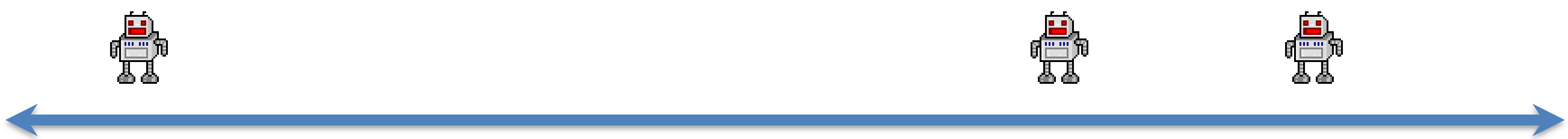


Check out my greedy algorithm! Always dispatch the robot closest to the request point.
For your algorithm, there's a request sequence of length n, whose competitive ratio goes to infinity as n goes to infinity!

The Double Coverage (DC) Algorithm

The DC Algorithm (Chrobak, Karloff, Payne, Vishwanathan, 1990)**:**

- The closest robots on both sides of the request (if they exist) move towards the request at equal speed.
- Both stop moving once one of them reaches the request.
- If two or more robots are co-located, only one will move.



We will show: If there are k robots, DC is k-competitive!

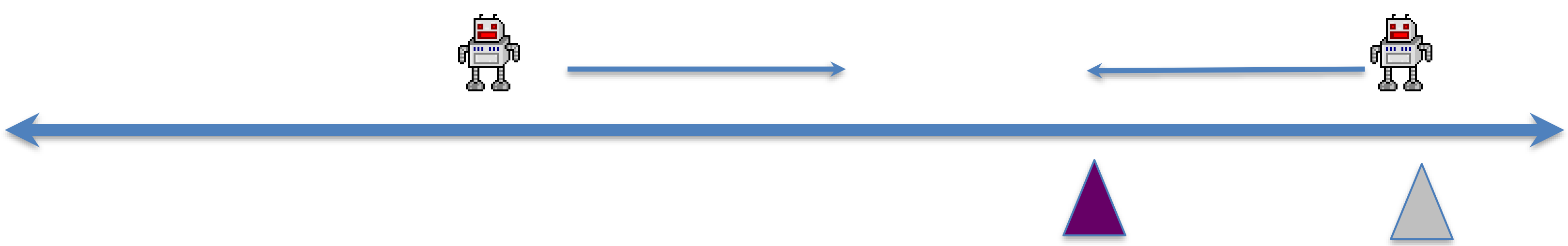
It turns out k is the *best possible* competitive ratio for a deterministic algorithm.

But you can do better with randomization (we won't show).

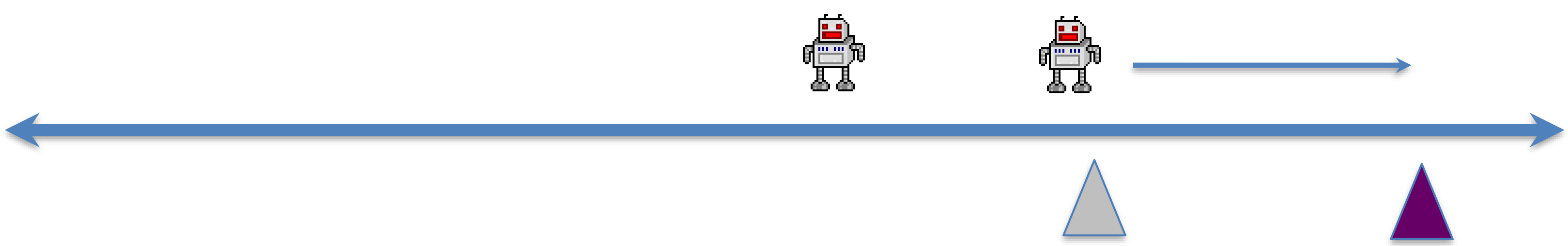
The Double Coverage (DC) Algorithm

The Double Coverage (DC) Algorithm

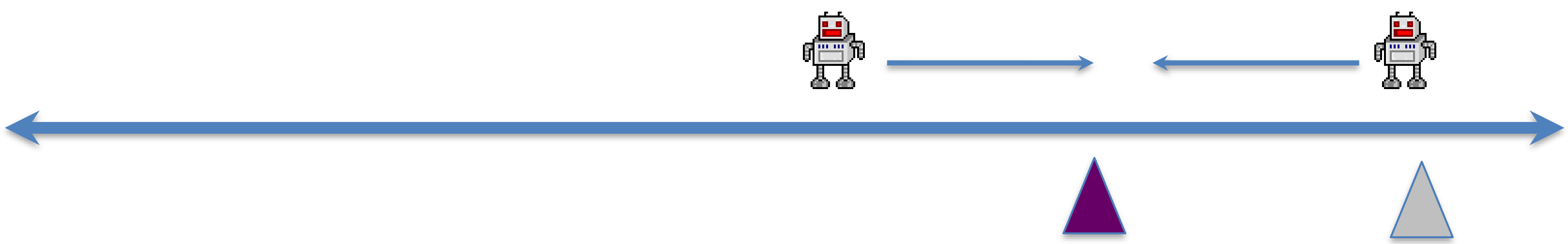
The Double Coverage (DC) Algorithm



The Double Coverage (DC) Algorithm



The Double Coverage (DC) Algorithm

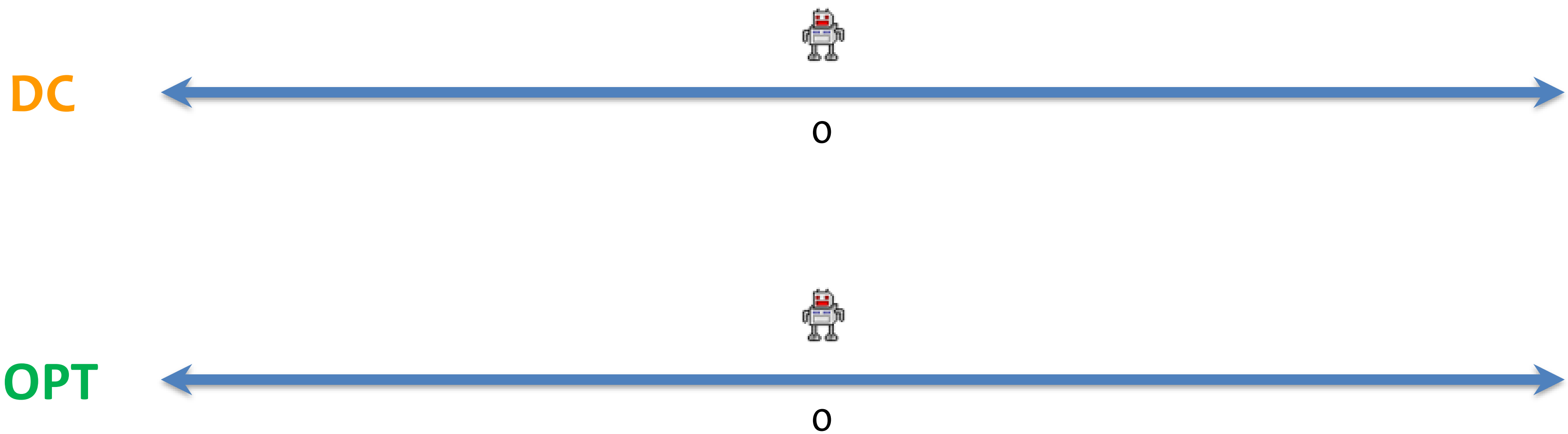


The Double Coverage (DC) Algorithm

The Double Coverage (DC) Algorithm

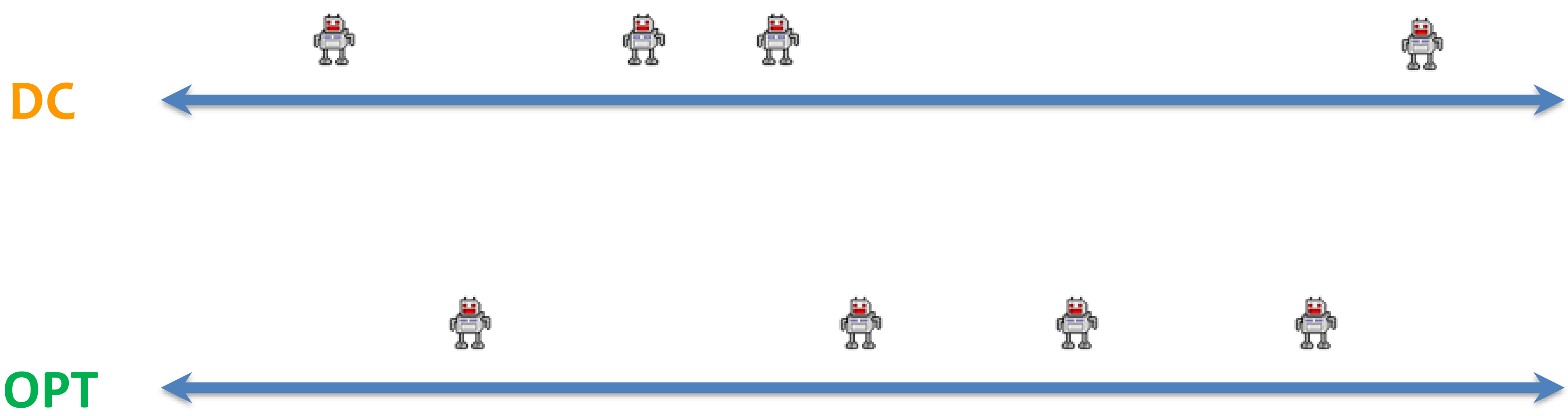
DC certainly looks better than the greedy algorithm

Goal: Show DC is k-competitive



Assumption: Initially, DC and OPT have all of their robots at the origin…

Goal: Show DC is k-competitive



… but over time, as they service requests, the robots spread out and the configurations of DC and OPT may start looking different

Claim: Without loss of generality, we can assume OPT only moves 1 robot per request. **Why?** Consider alg OPT' that's the same as OPT but only moves the 1 robot that services the request.

A Potential Function Argument

To prove that DC is k-competitive, we will use a **potential function!**

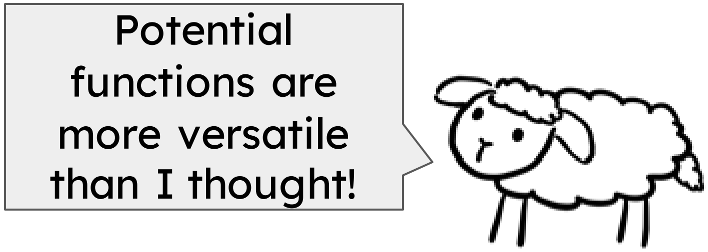
Previously: We used a potential function S_i to measure the ***running time*** of an algorithm.

⇒ Smaller S_i value means the algorithm is closer to termination.

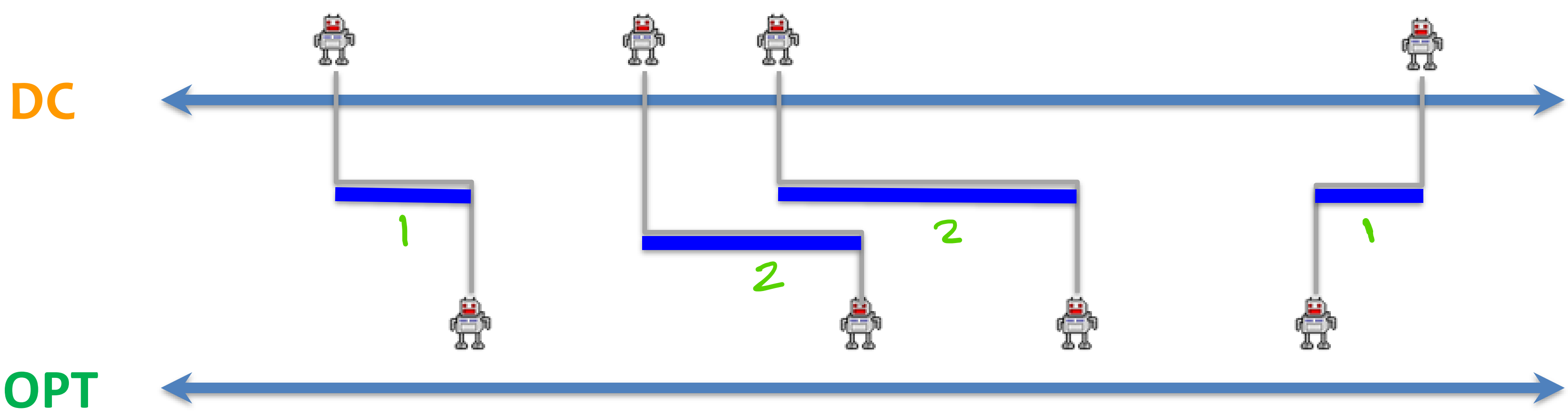
Now: We will use a potential function S_i to measure the ***competitive ratio*** of the DC algorithm.

⇒ Smaller S_i value means DC is in a "better configuration" ≈ closer to OPT.

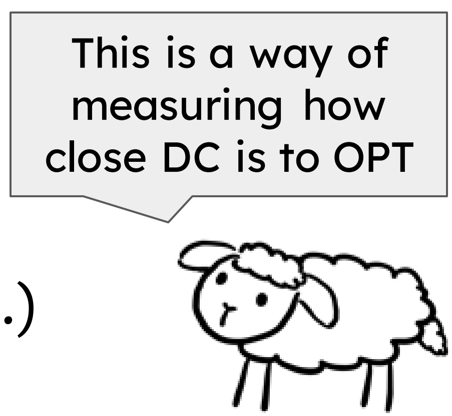
(Here, S_i can go up and down over time.)



Potential Function: Attempt #1



M = "match" the robots and take the total length of the **"horizontal legs"** 6



Potential function (attempt #1): S_i = **M**

(i.e. value of **M** right after i^{th} request has been serviced.)

How will we use our potential function?

Observations:

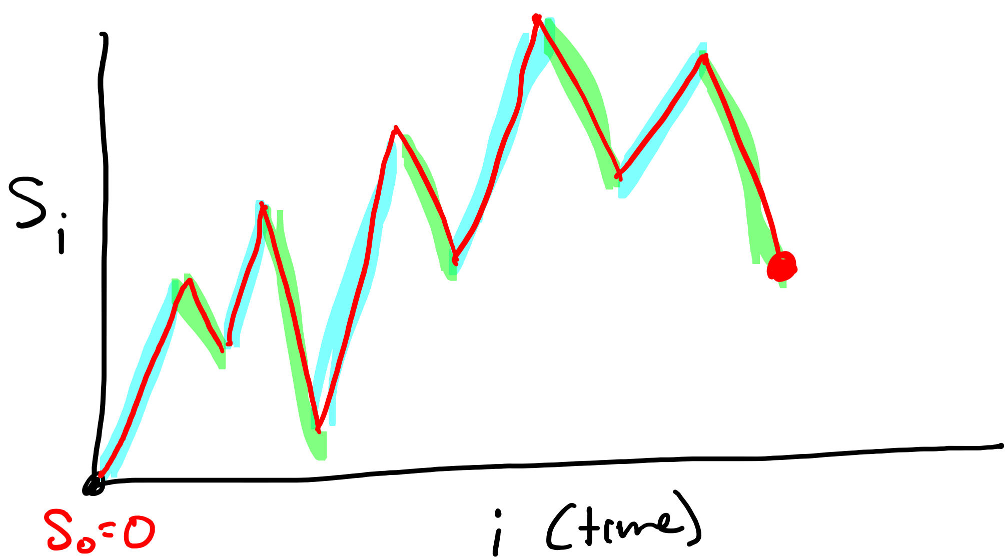
- $S_0 = 0$.
- $S_i \geq 0$ for all i.

therefore $\Rightarrow$

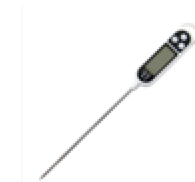
total amount S_i *ever decreases*

$\leq$

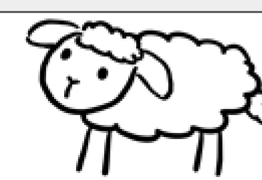
total amount S_i *ever increases*



S_0 OPT moves $S_{0.5}$ DC moves S_1 OPT moves.



For each request, we measure the change in potential right after OPT moves and then again after DC moves!



How will we use our potential function?

Observations:

- $S_0 = 0$.
- $S_i \geq 0$ for all i.

therefore $\Rightarrow$

total amount S_i *ever decreases*

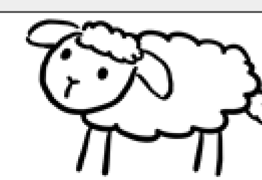
$\leq$

total amount S_i *ever increases*

Key properties we want to show:

- For any given request, if **OPT** moves its server a distance of **d** then **S_i increases by $\leq$ k·d** (or decreases).
- For any given request, if **DC** moves its server(s) a total distance **d** then **S_i decreases by $\geq$ d**.

For each request, we measure the change in potential right after OPT moves and then again after DC moves!

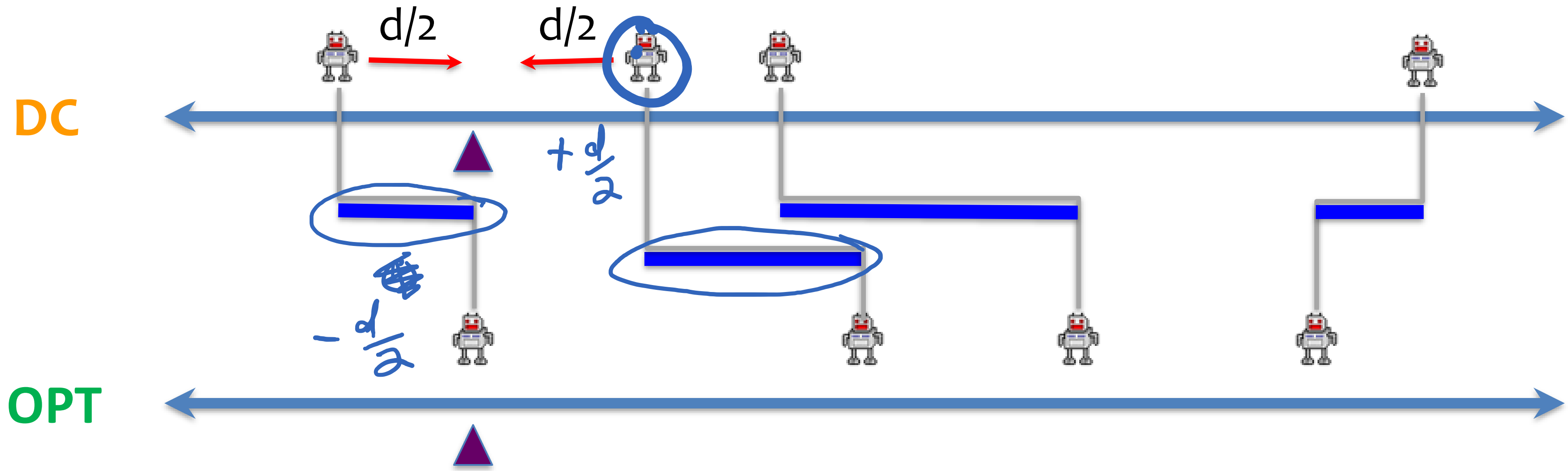


Consequence of key properties:

- total amount S_i *ever increases* $\leq$ **k·OPT**
- total amount S_i *ever decreases* $\geq$ **DC**

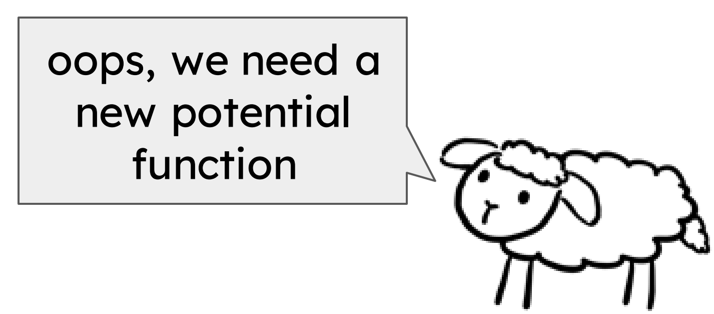
$\Rightarrow$ **DC** $\leq$ **k** · **OPT** (as desired)

Potential Function: Attempt #1

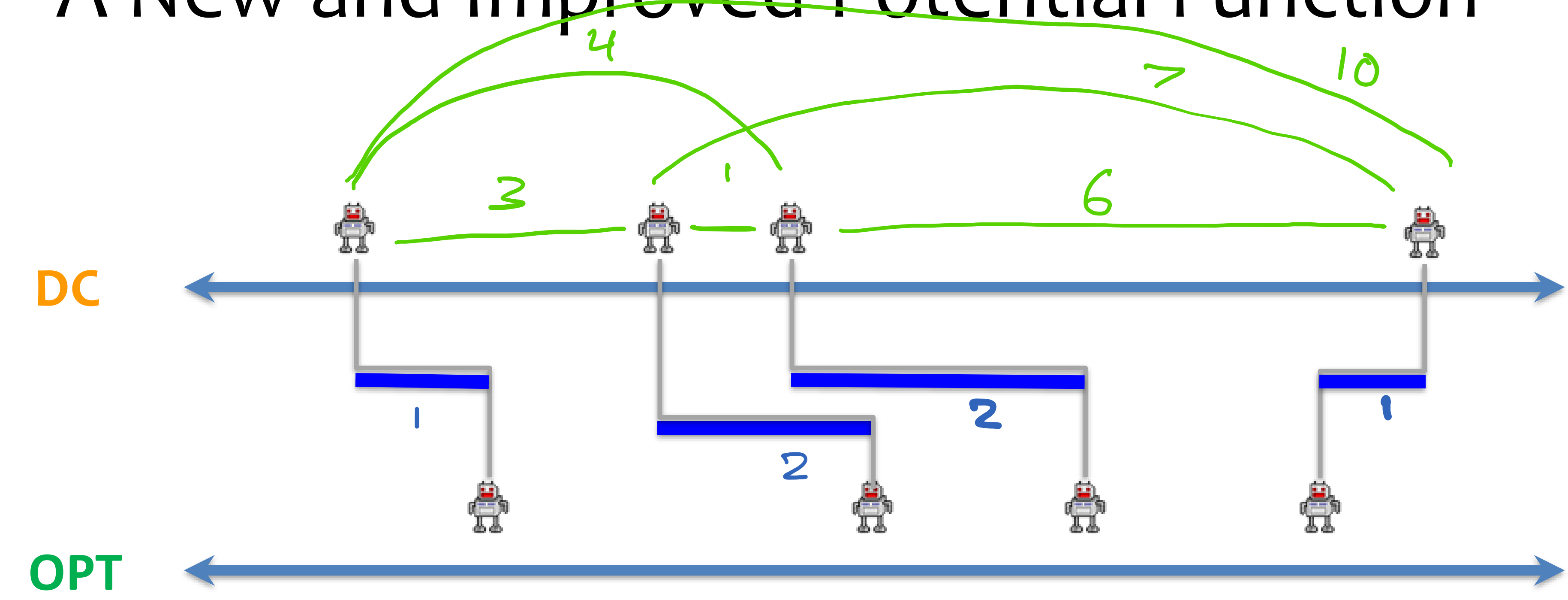


S_i (attempt #1) = **M** (= total length of "**horizontal legs**")

Want: if DC moves its server(s) a total distance **d** then S_i **decreases by ≥ d.**



A New and Improved Potential Function



M = total length of the **"horizontal legs"** 6

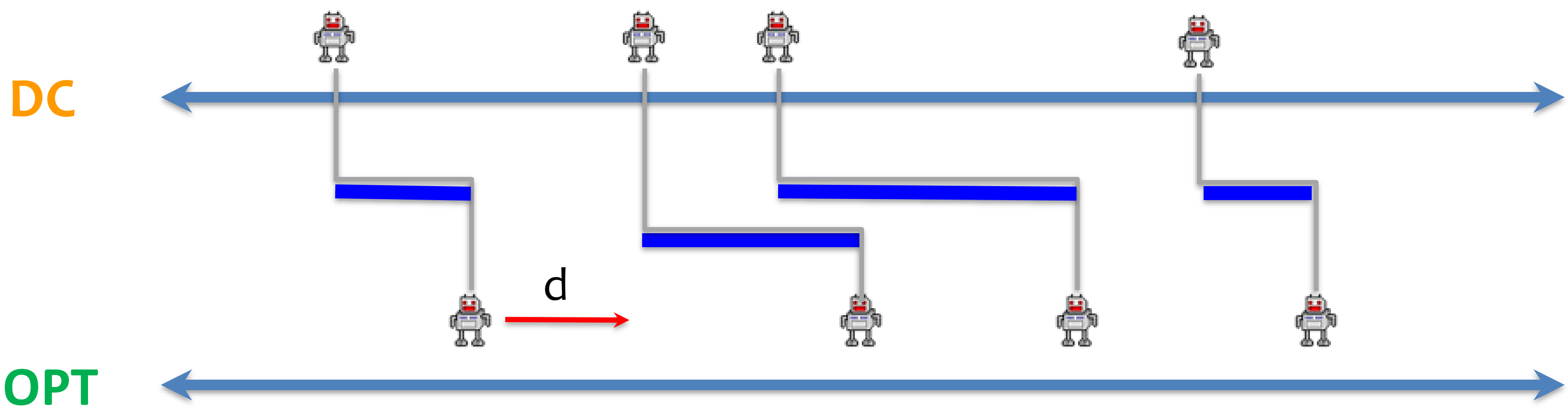
> What planet did this potential function come from?

Σ_{DC} = Sum of all pairwise distances between **DC**'s servers 31

New potential function: $S_i = \underset{4 \cdot 6}{k \cdot M} + \underset{31}{\Sigma_{DC}} = 55$

Proving the Key Properties

Want to show: If **OPT** moves its server a distance of **d** then S_i **increases by $\le k \cdot d$** (or decreases).



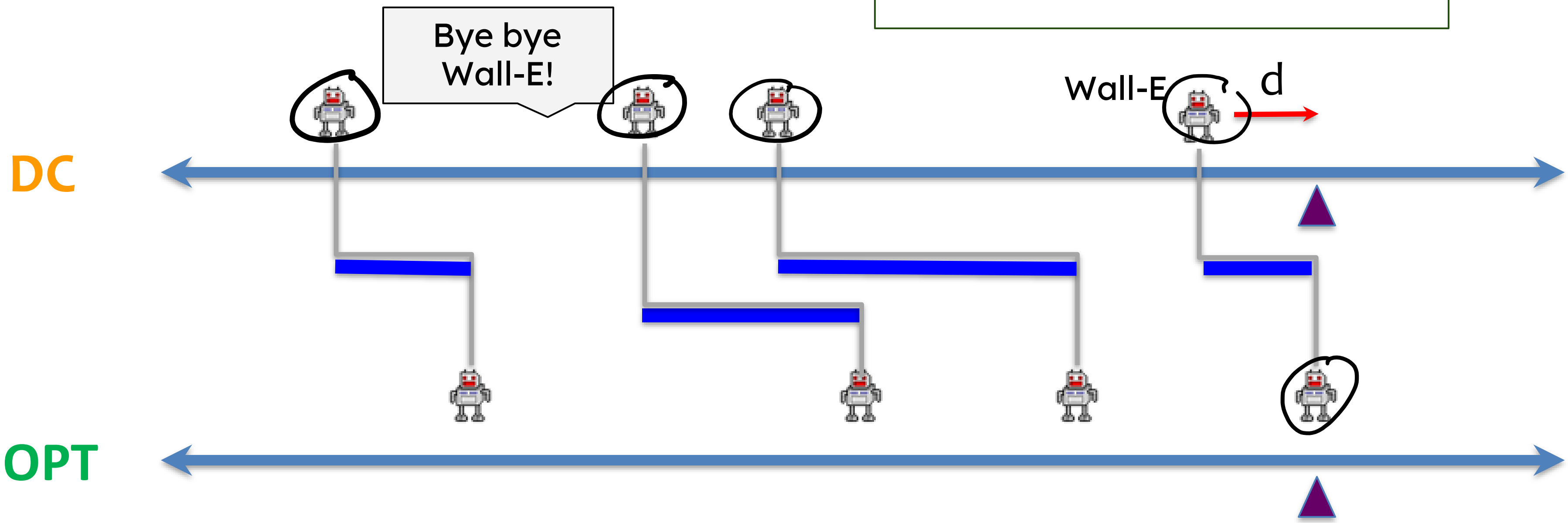
Potential function: $S_i = k \cdot \underbrace{M}_{+d} + \underbrace{\Sigma DC}_{+0}$

$\le kd$

Proving the Key Properties

Want to show: If DC moves its server(s) a total distance **d** then S_i **decreases by ≥ d.**

Case 1: DC moves 1 robot



Potential function: $S_i = k \cdot \underline{M} + \underline{\Sigma DC}$

$-d \qquad +(k-1)d$

$-kd + (k-1)d$

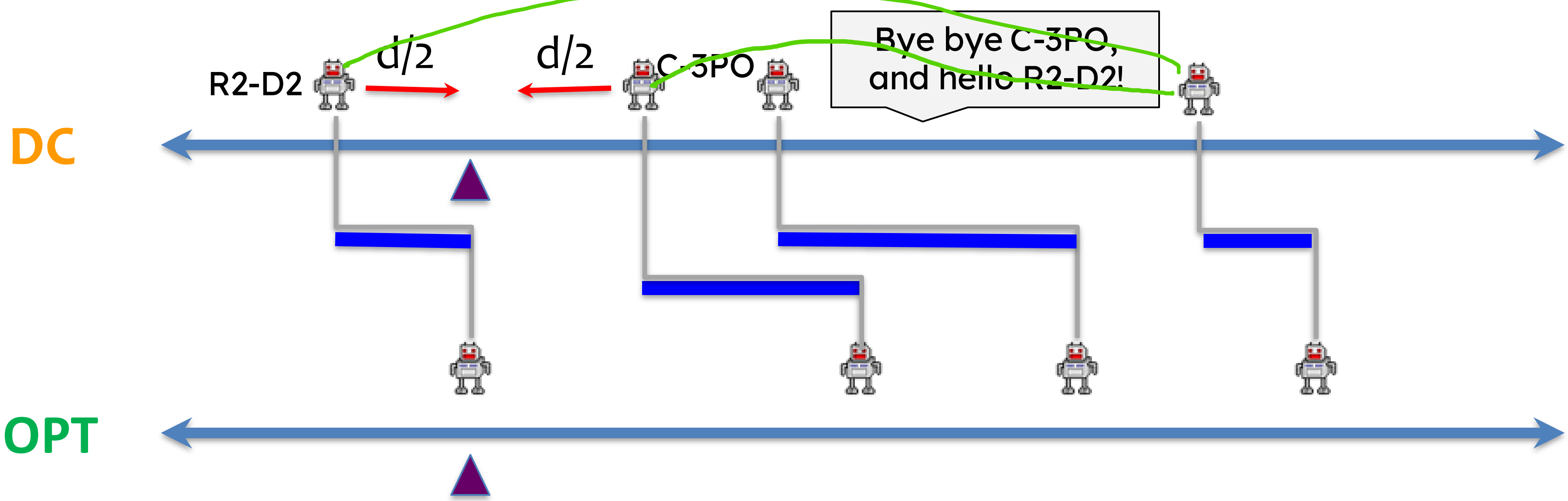
$-d$

Note that OPT already has a robot at the request

Proving the Key Properties

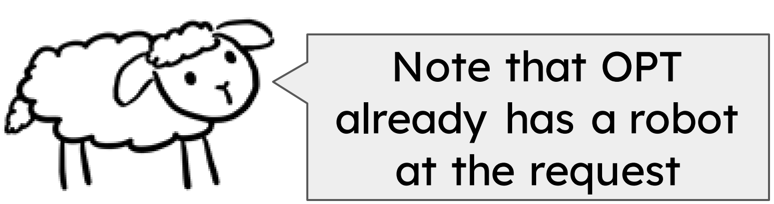
Want to show: If DC moves its server(s) a total distance **d** then S_i **decreases by ≥ d.**

Case 2: DC moves 2 robots



First, how does ΣDC change?

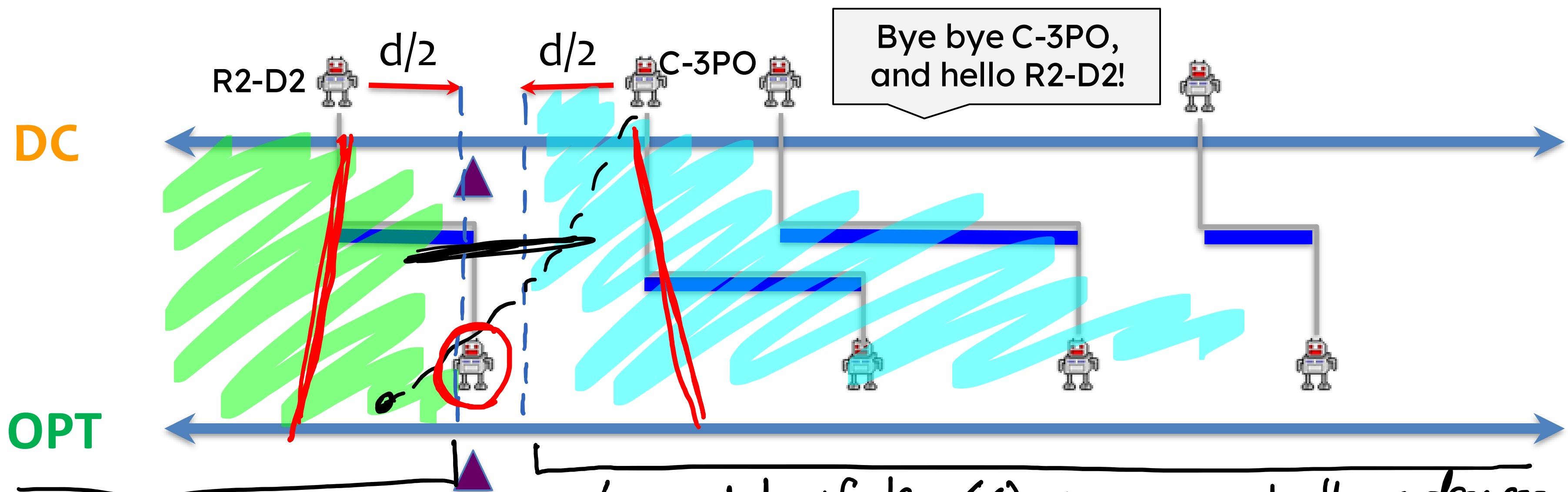
− d



Proving the Key Properties

Want to show: If DC moves its server(s) a total distance **d** then S_i **decreases by ≥ d.**

Case 2: DC moves 2 robots

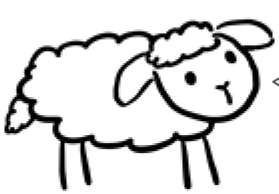


R's match if leg(R) increases at all or decreases by $< \frac{d}{2}$

C's match if leg(C) increases at all or decreases by $< \frac{d}{2}$

Both can't happen!

At least one of leg(C) or leg(R) decrease by $\geq \frac{d}{2}$



Note that OPT already has a robot at the request

Second, how does **M** change?

Claim: net same or dec

How we used our potential function

Observations:

- $S_0 = 0$.
- $S_i \geq 0$ for all i.

therefore $\Rightarrow$

total amount S_i *ever decreases*

$\leq$

total amount S_i *ever increases*

Key properties we want to show:

- For any given request, if OPT moves its server a distance of d then S_i **increases by** $\leq$ $k \cdot d$ (or decreases).
- For any given request, if DC moves its server(s) a total distance d then S_i **decreases by** $\geq$ d.

Consequence of key properties:

- total amount S_i *ever increases* $\leq k \cdot OPT$
- total amount S_i *ever decreases* $\geq DC$

$\Rightarrow DC \leq k \cdot OPT$ (as desired)