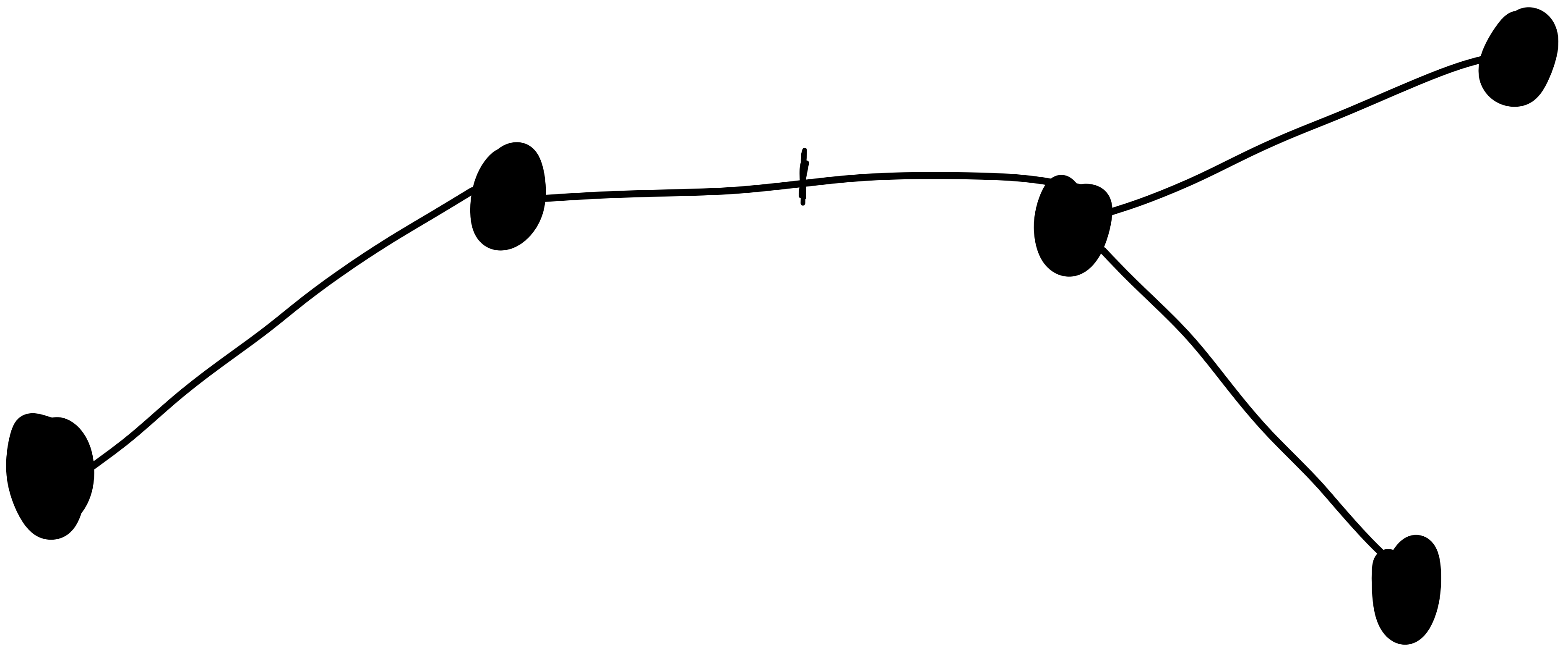
Crash-tolerant Exploration of Trees by Energy Sharing Mobile Agents

Quentin Bramas, Toshimitsu Masuzawa and Sebastien Tixeuil

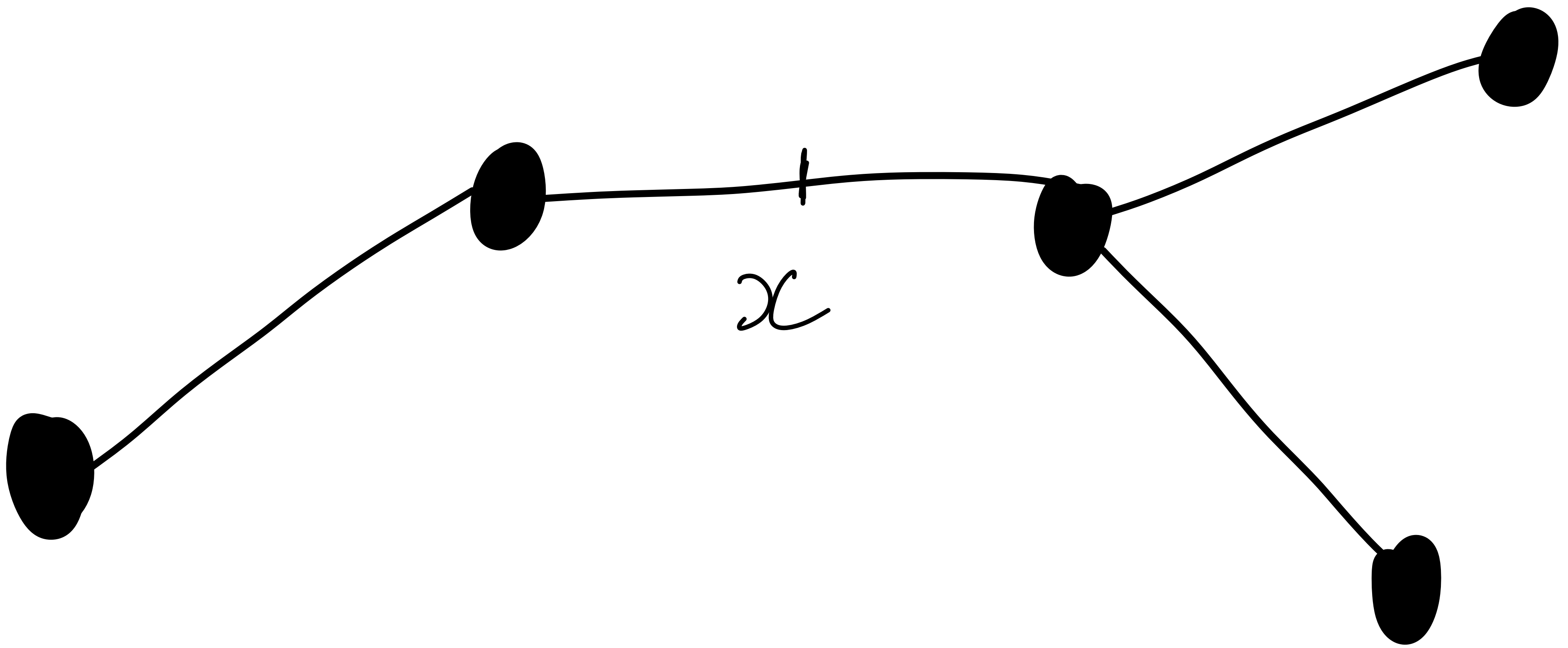
University of Strasbourg, Osaka University, Sorbonne University

SAND 2024

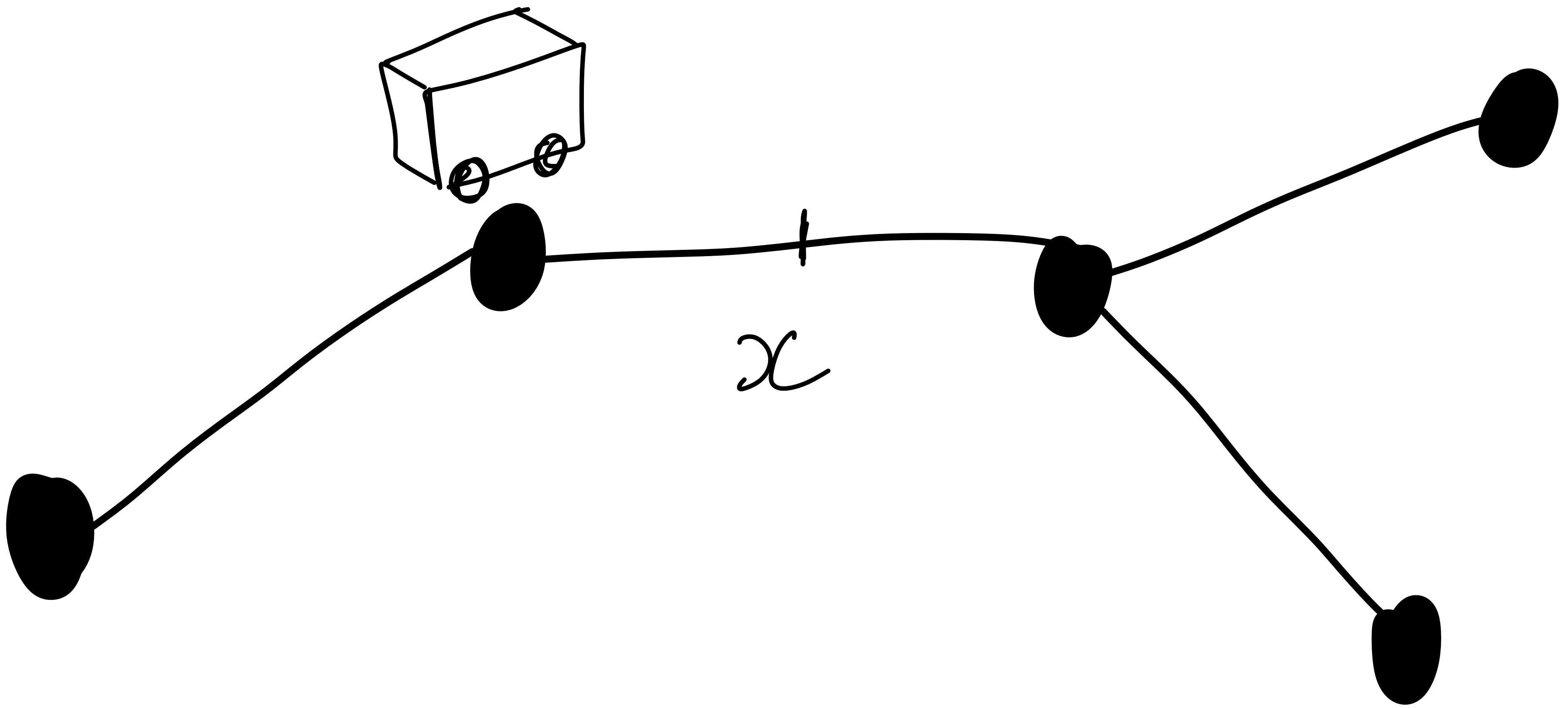
Mobile agents with energy



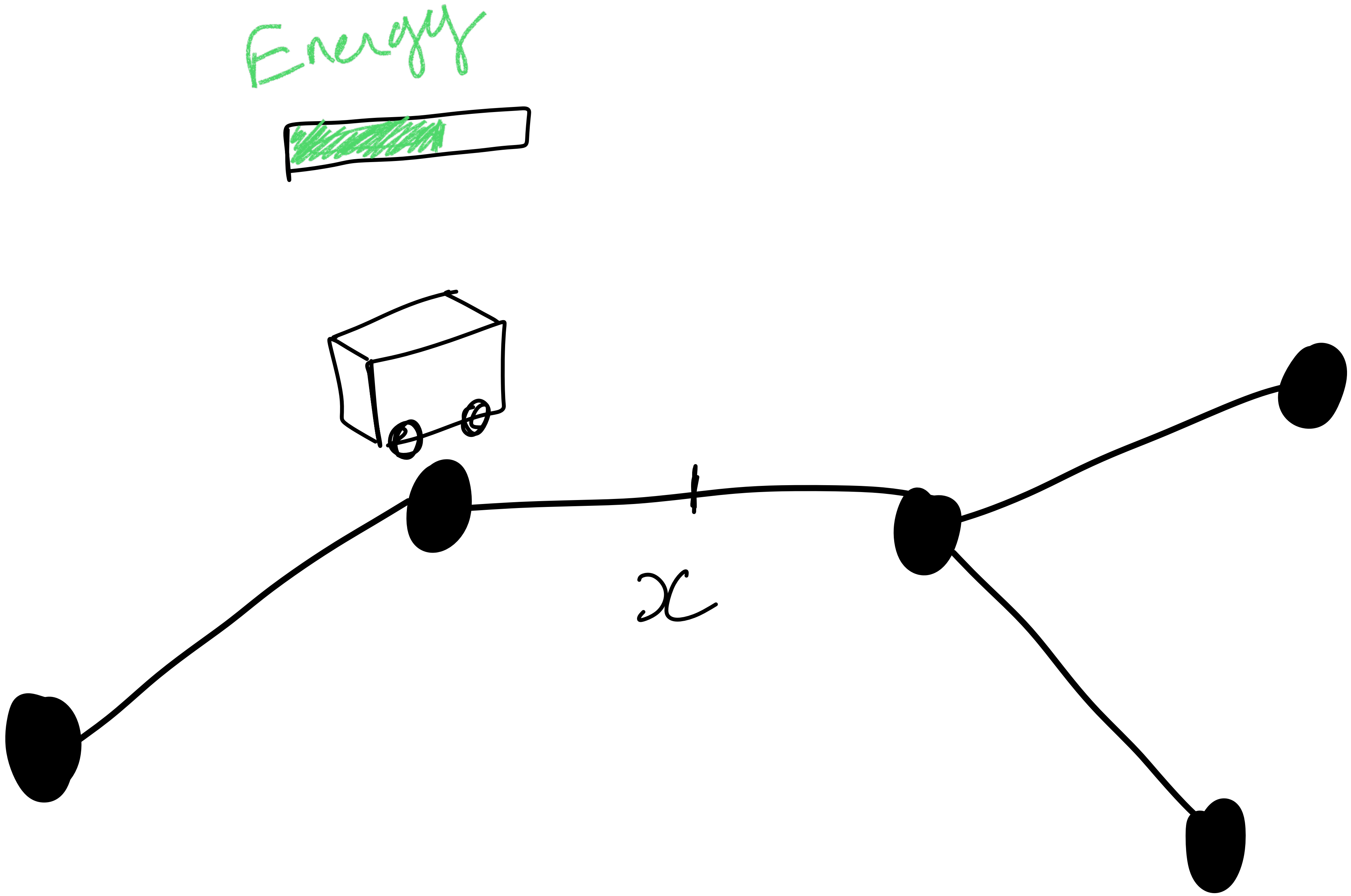
Mobile agents with energy



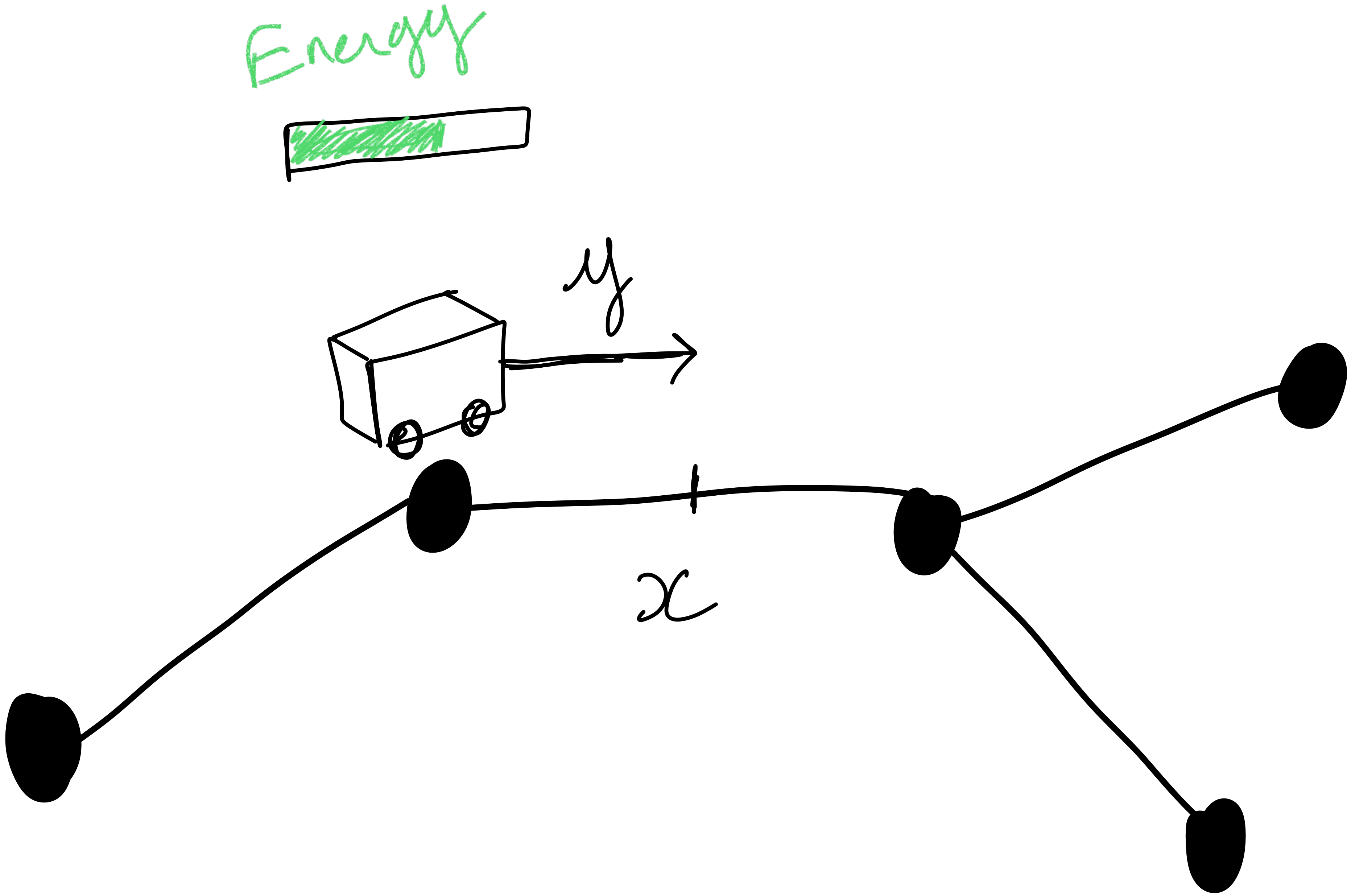
Mobile agents with energy



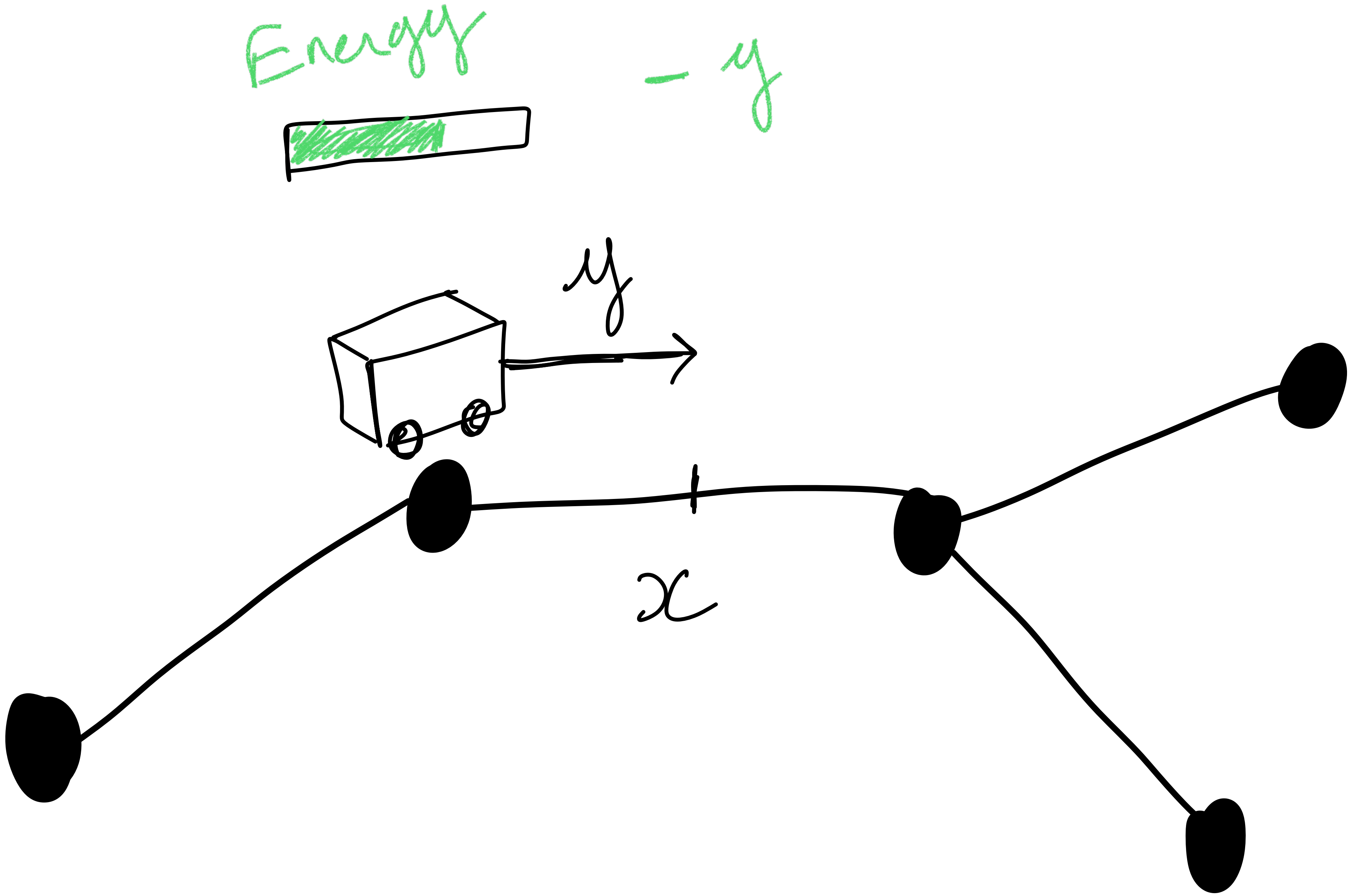
Mobile agents with energy



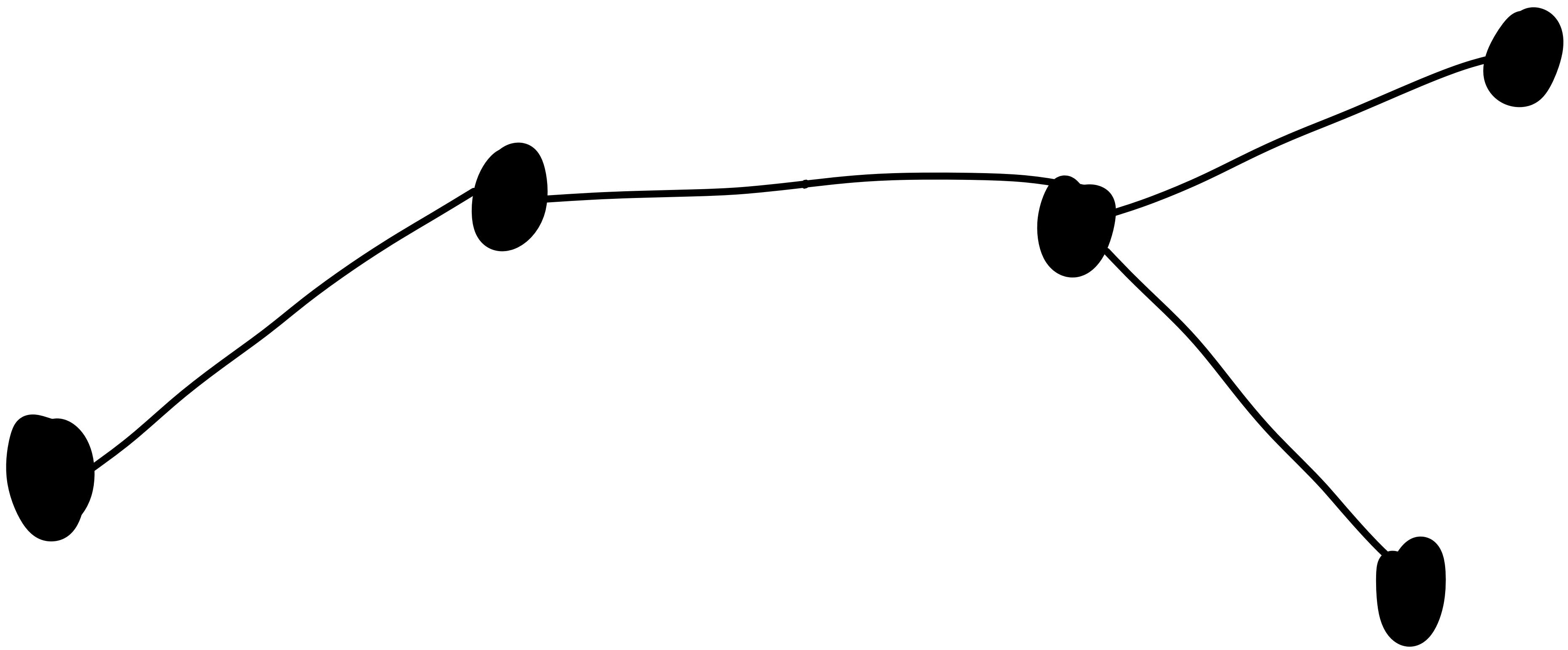
Mobile agents with energy



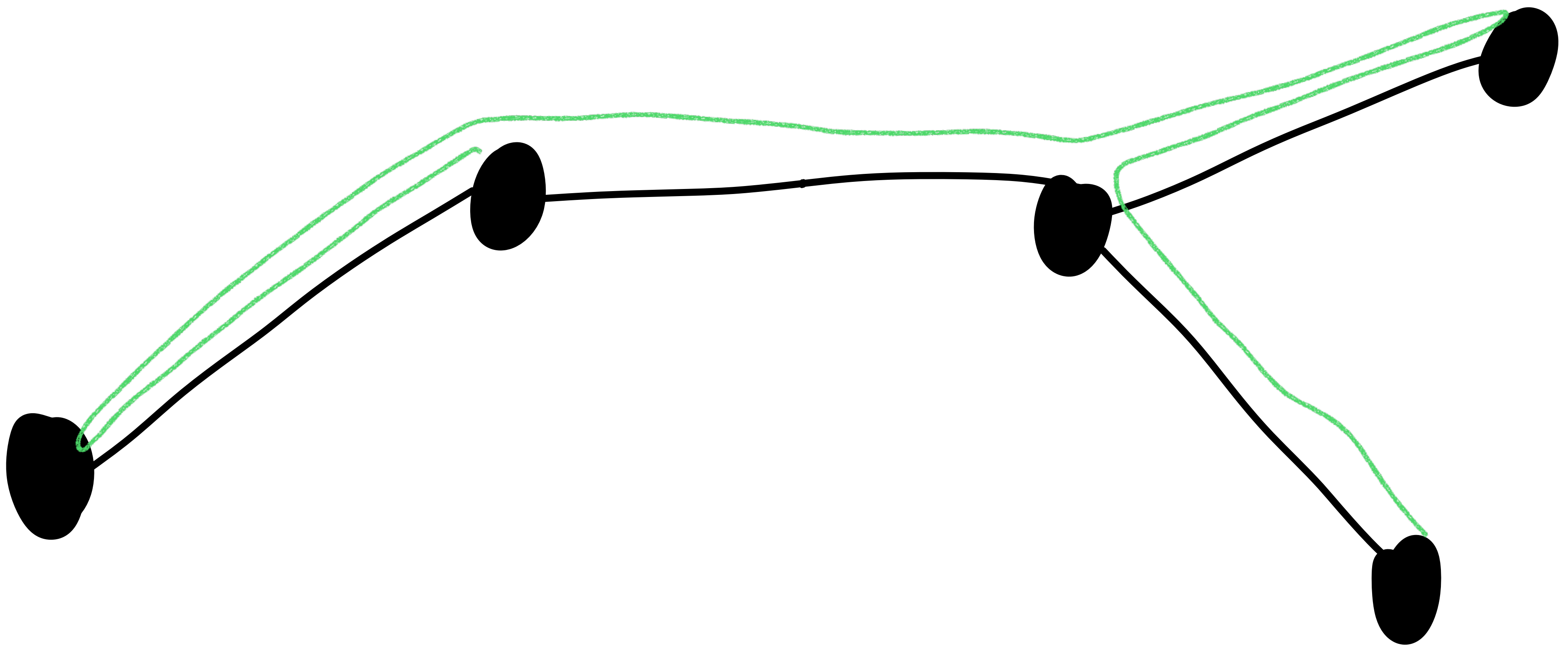
Mobile agents with energy



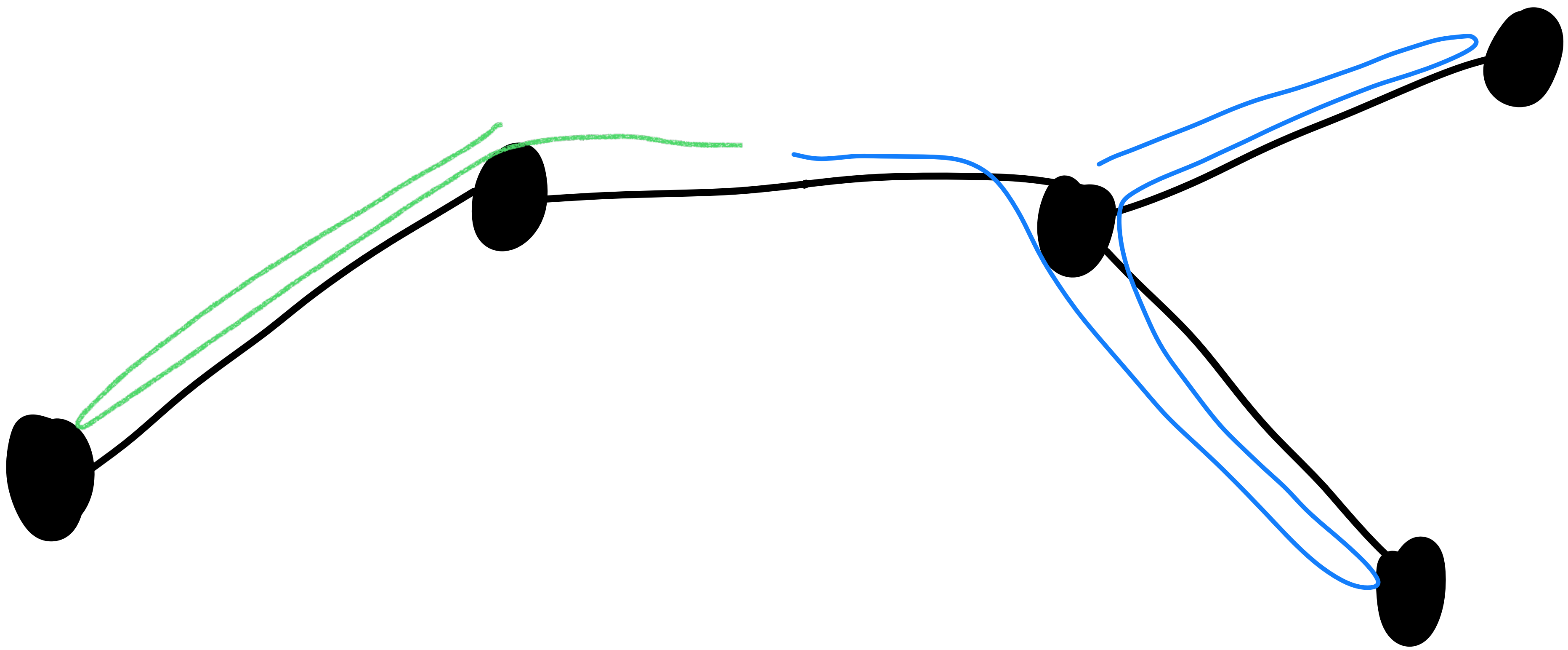
Graph exploration



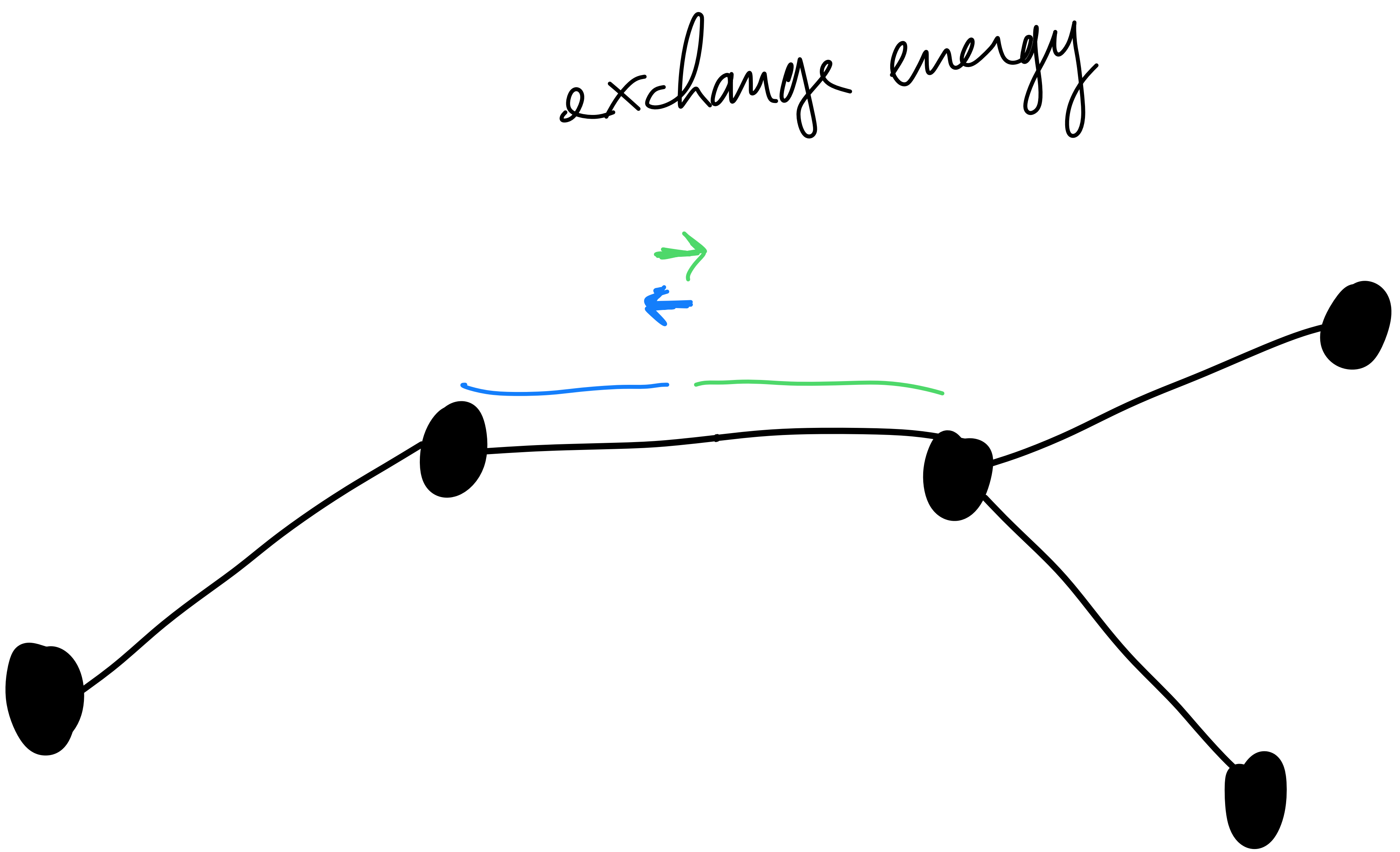
Graph exploration



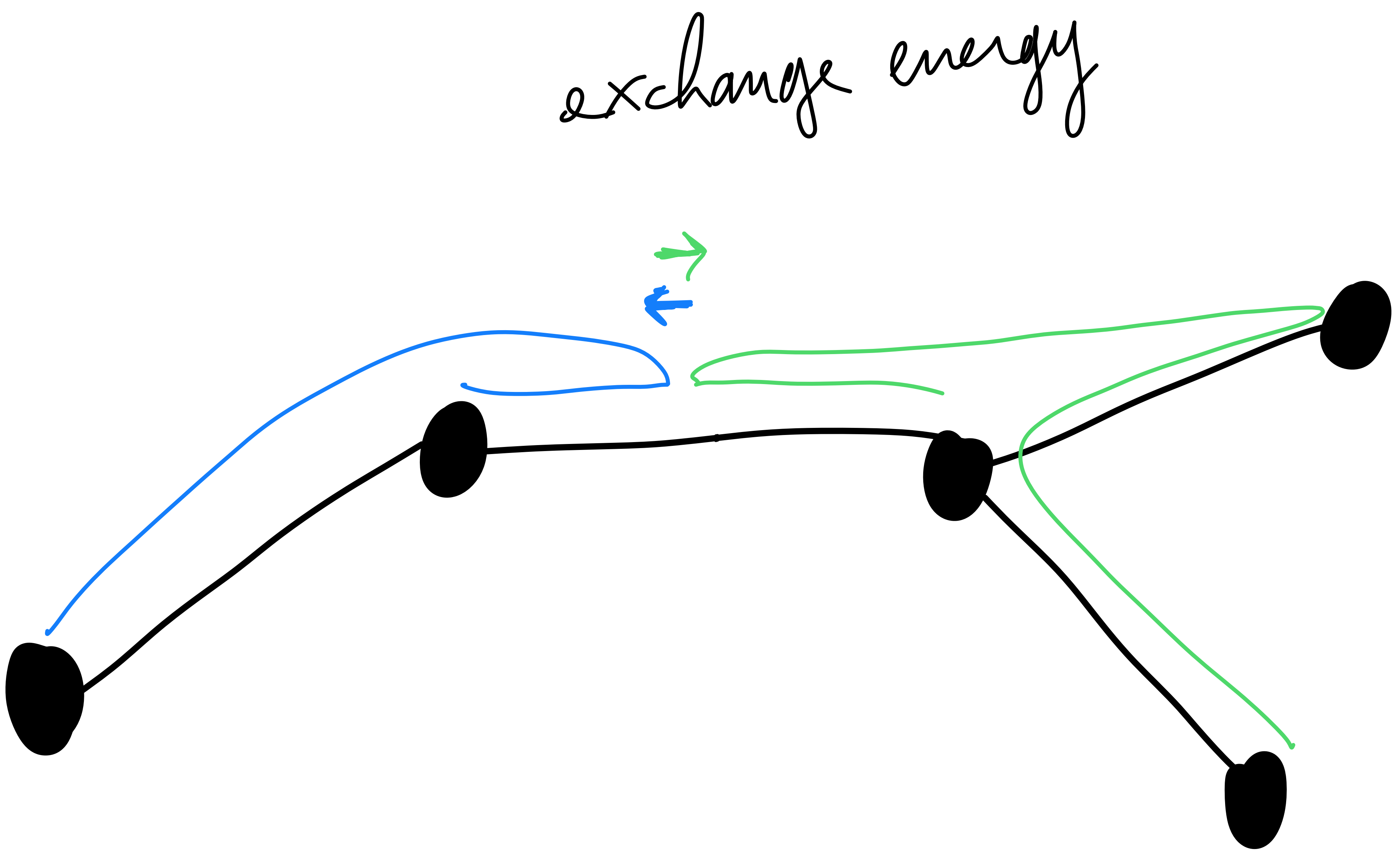
Graph exploration



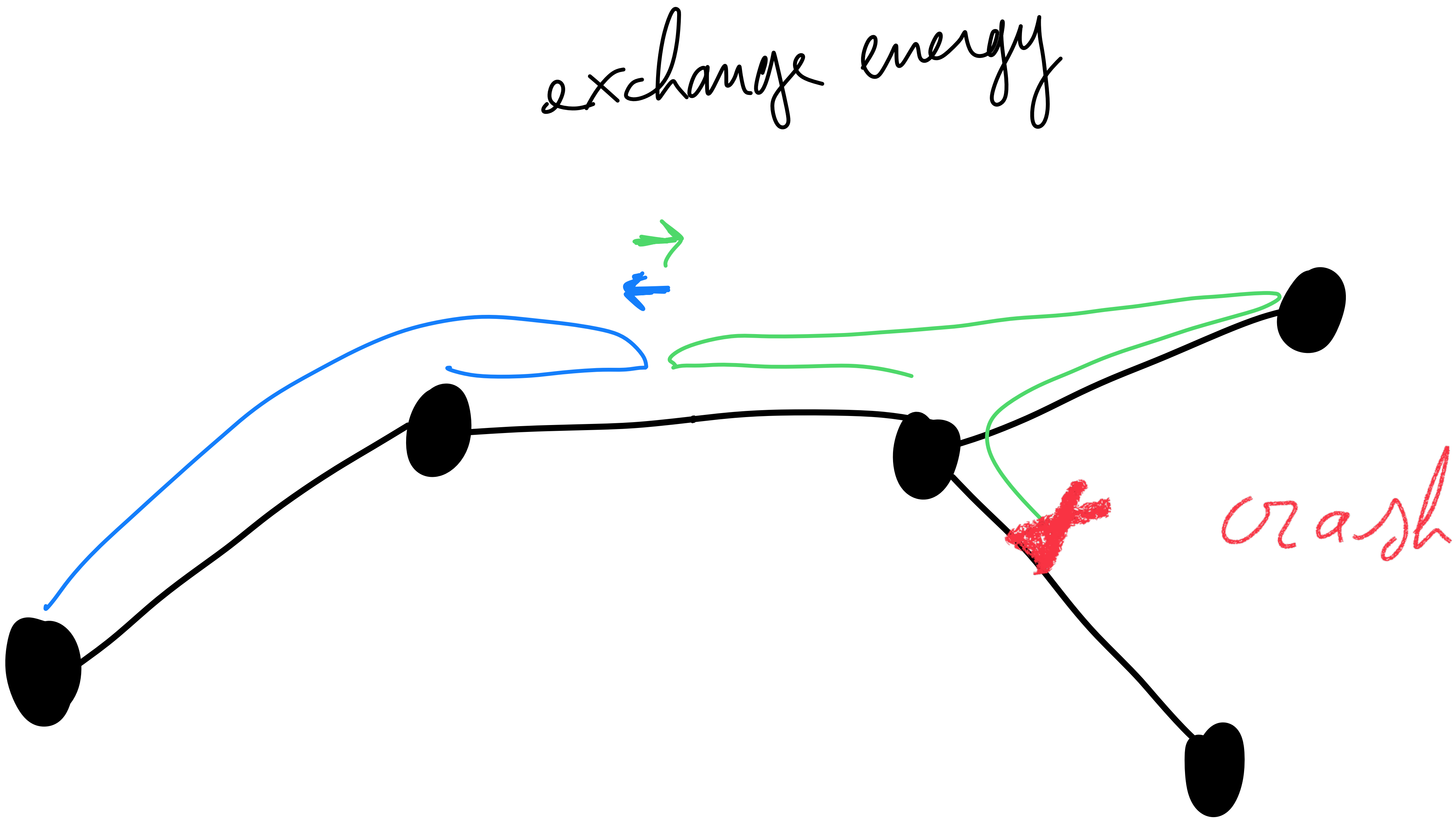
Graph exploration



Graph exploration



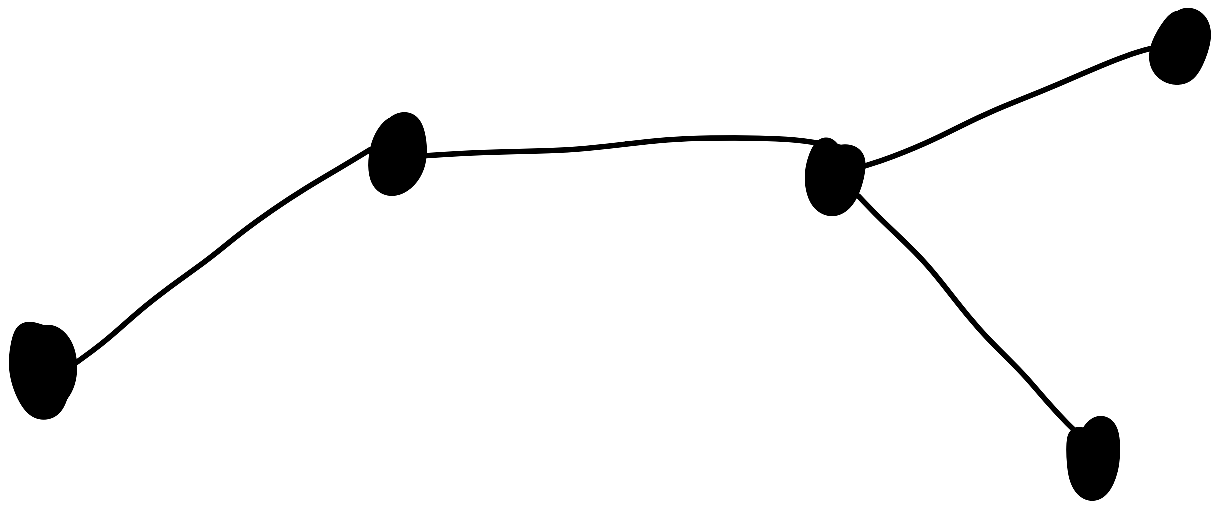
Graph exploration



Localized algorithms

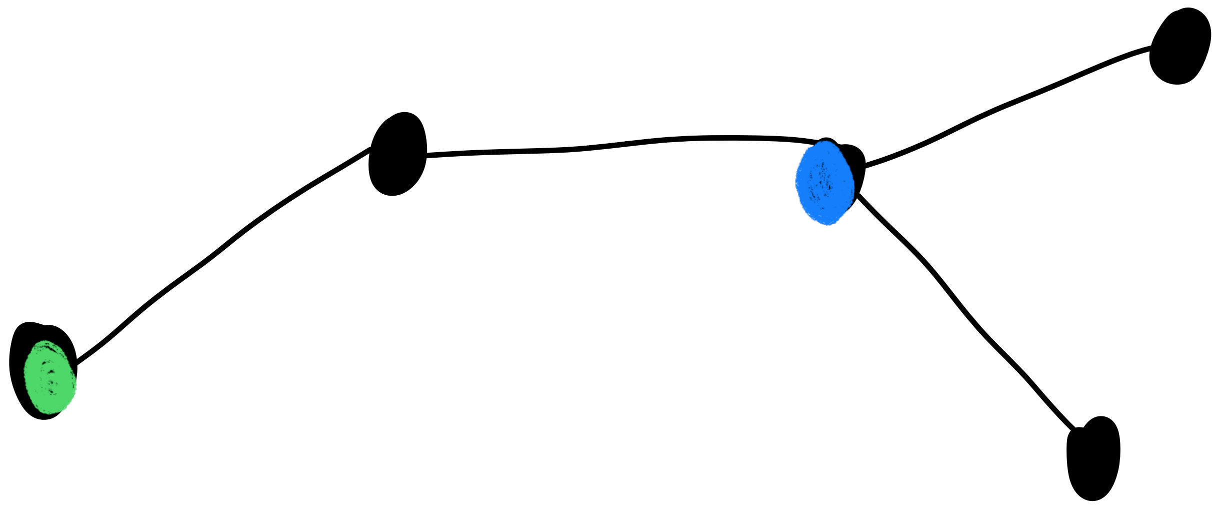
Localized algorithms

Given:



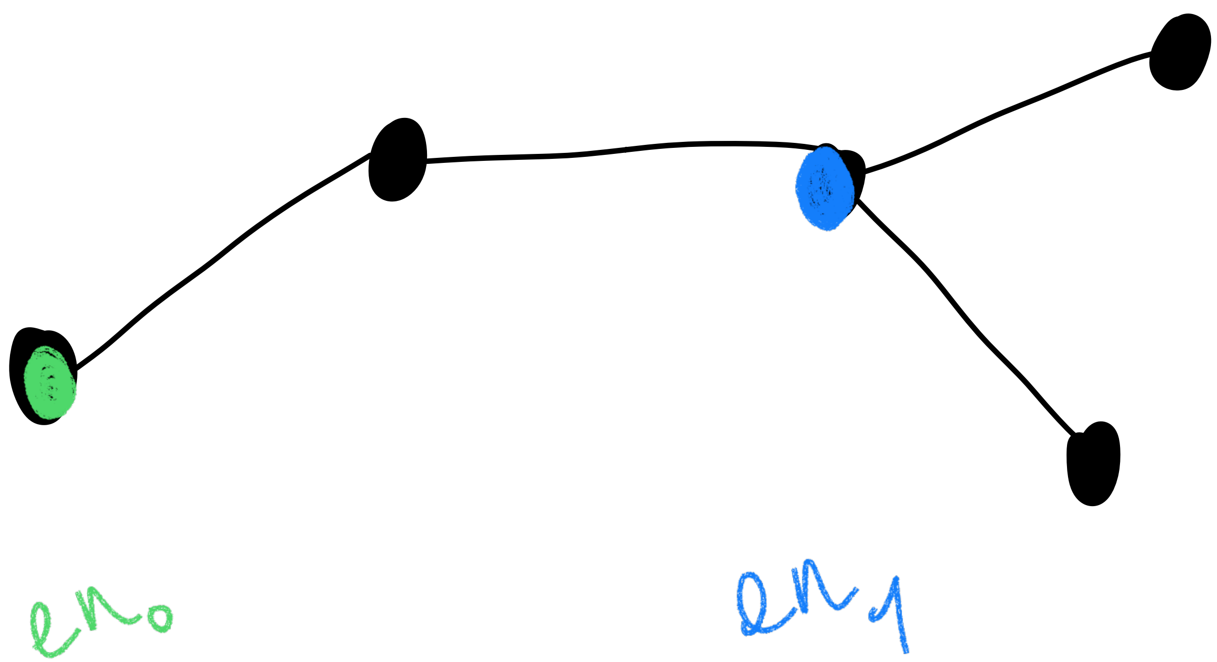
Localized algorithms

Given:



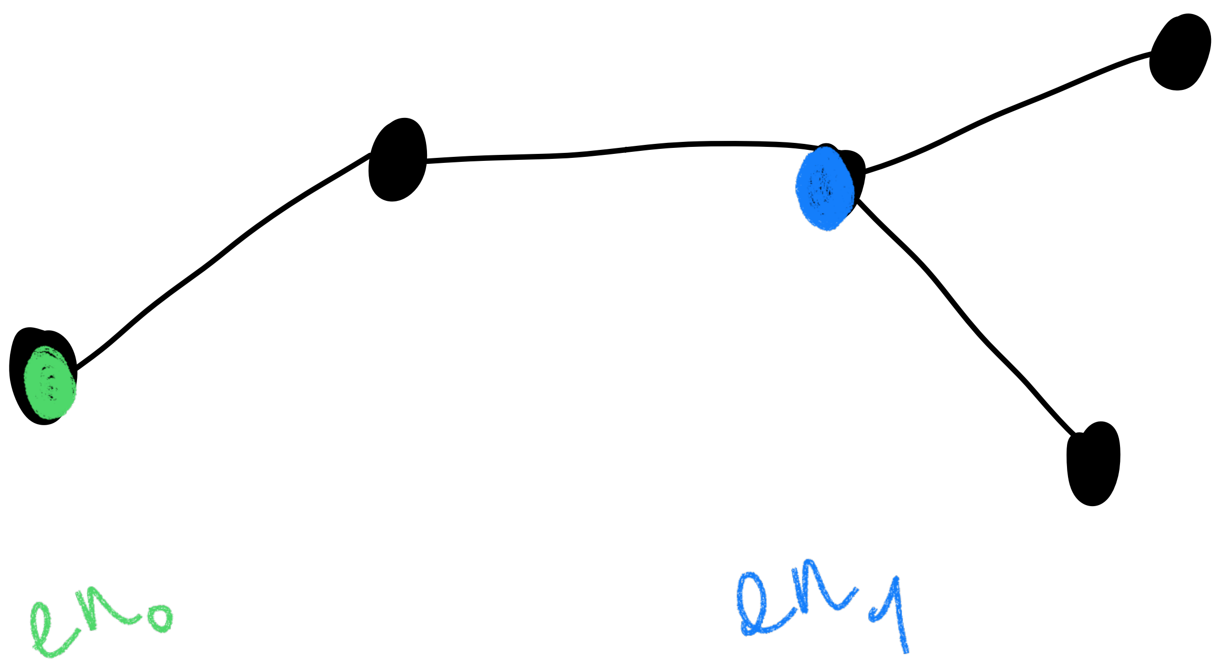
Localized algorithms

Given:



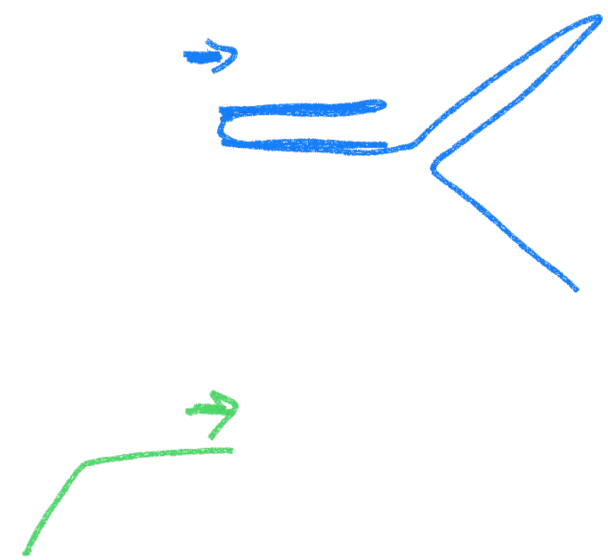
Localized algorithms

Given:



Find:

2 Algorithms



Localized algorithms

Find:

2 Algorithms

$\Rightarrow$

Once deployed,

- autonomous
- communicate when meet

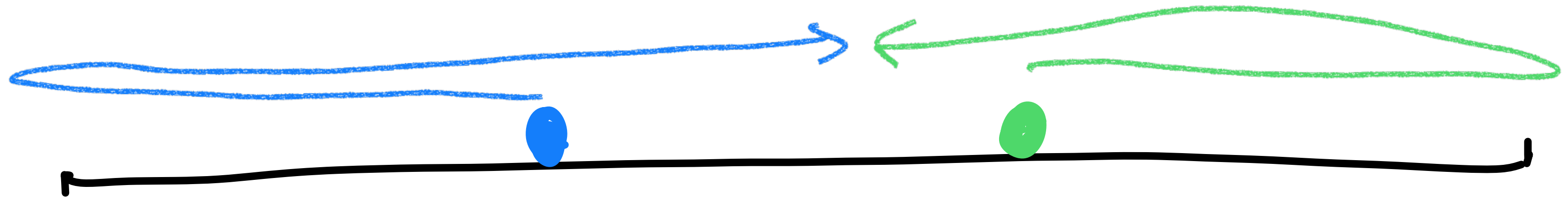
Synchronous

or

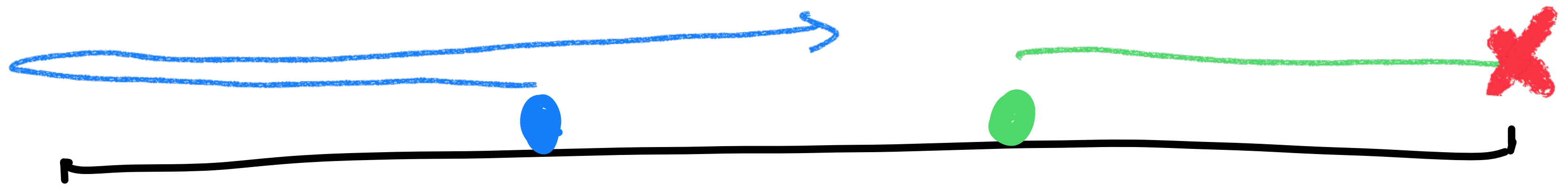
Asynchronous

Exploring the line with 2 agents

Exploring the line with 2 agents

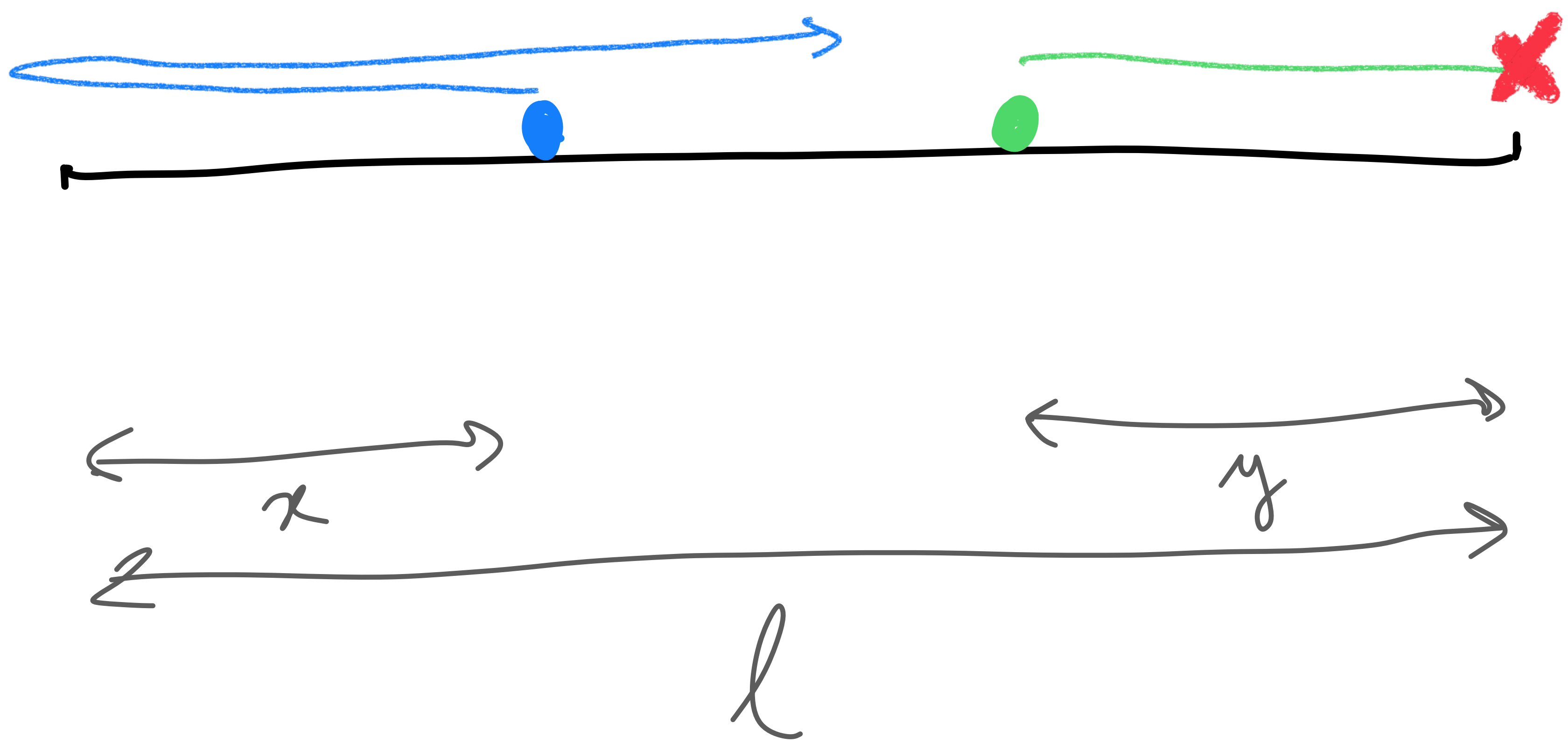


Exploring the line with 2 agents



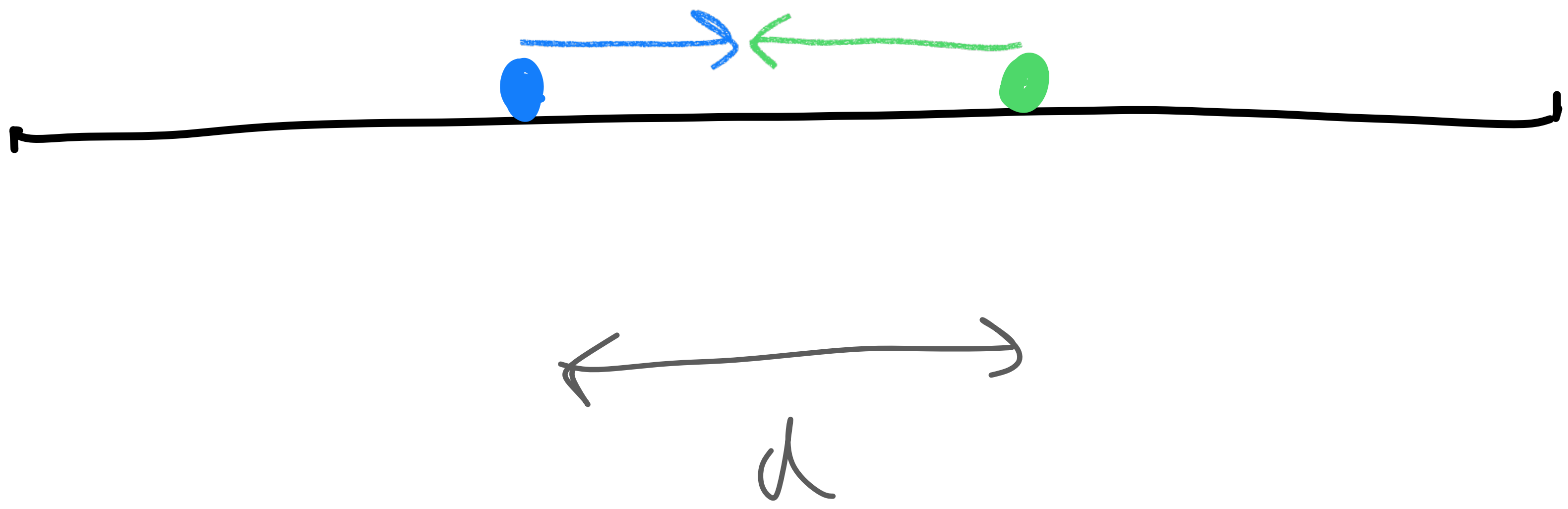
One easy algorithm

$$en_0 \geq \ell + x \ \wedge \ en_1 \geq \ell + y$$



Another easy algorithm

$$en_0 \geq d \ \wedge \ en_1 \geq d \ \wedge \ en_0 + en_1 \geq 3\ell + d$$



How many easy algorithms are there?

How many easy algorithms are there?

We found the necessary conditions and the corresponding algorithms

$$\boldsymbol{c_1} : (en_0 \geq x + y) \wedge (en_1 \geq y) \wedge (en_0 + en_1 \geq 2\ell + x + y)$$
$$\boldsymbol{c_2} : (en_0 \geq \ell - x) \wedge (en_1 \geq 2\ell - (x + y)) \wedge (en_0 + en_1 \geq 4\ell - (x + y))$$
$$\boldsymbol{c_3} : (en_0 \geq \ell + x) \wedge (en_1 \geq 2\ell - y)$$
$$\boldsymbol{c_4} : (en_0 \geq y - x) \wedge (en_1 \geq y - x) \wedge (en_0 + en_1 \geq min(3\ell + y - x, 2\ell - x + 3y))$$

How many easy algorithms are there?

We found the necessary conditions and the corresponding algorithms

$$\boldsymbol{c_1} \;:\; (en_0 \geq x + y) \wedge (en_1 \geq y) \wedge (en_0 + en_1 \geq 2\ell + x + y)$$
$$\boldsymbol{c_2} \;:\; (en_0 \geq \ell - x) \wedge (en_1 \geq 2\ell - (x + y)) \wedge (en_0 + en_1 \geq 4\ell - (x + y))$$
$$\boldsymbol{c_3} \;:\; (en_0 \geq \ell + x) \wedge (en_1 \geq 2\ell - y)$$
$$\boldsymbol{c_4} \;:\; (en_0 \geq y - x) \wedge (en_1 \geq y - x) \wedge (en_0 + en_1 \geq min(3\ell + y - x, 2\ell - x + 3y))$$

Same for the synchronous case

How many easy algorithms are there?

We found the necessary conditions and the corresponding algorithms

$$\boldsymbol{c_1} : (en_0 \geq x + y) \wedge (en_1 \geq y) \wedge (en_0 + en_1 \geq 2\ell + x + y)$$
$$\boldsymbol{c_2} : (en_0 \geq \ell - x) \wedge (en_1 \geq 2\ell - (x + y)) \wedge (en_0 + en_1 \geq 4\ell - (x + y))$$
$$\boldsymbol{c_3} : (en_0 \geq \ell + x) \wedge (en_1 \geq 2\ell - y)$$
$$\boldsymbol{c_4} : (en_0 \geq y - x) \wedge (en_1 \geq y - x) \wedge (en_0 + en_1 \geq min(3\ell + y - x, 2\ell - x + 3y))$$

Same for the synchronous case

And for trees?

How many easy algorithms are there?

We found the necessary conditions and the corresponding algorithms

$$\boldsymbol{c_1} \;:\; (en_0 \geq x + y) \wedge (en_1 \geq y) \wedge (en_0 + en_1 \geq 2\ell + x + y)$$
$$\boldsymbol{c_2} \;:\; (en_0 \geq \ell - x) \wedge (en_1 \geq 2\ell - (x + y)) \wedge (en_0 + en_1 \geq 4\ell - (x + y))$$
$$\boldsymbol{c_3} \;:\; (en_0 \geq \ell + x) \wedge (en_1 \geq 2\ell - y)$$
$$\boldsymbol{c_4} \;:\; (en_0 \geq y - x) \wedge (en_1 \geq y - x) \wedge (en_0 + en_1 \geq min(3\ell + y - x, 2\ell - x + 3y))$$

Same for the synchronous case

And for trees?

Almost tight algorithms

How many easy algorithms are there?

We found the necessary conditions and the corresponding algorithms

$$\boldsymbol{c_1} \,:\, (en_0 \geq x + y) \wedge (en_1 \geq y) \wedge (en_0 + en_1 \geq 2\ell + x + y)$$
$$\boldsymbol{c_2} \,:\, (en_0 \geq \ell - x) \wedge (en_1 \geq 2\ell - (x + y)) \wedge (en_0 + en_1 \geq 4\ell - (x + y))$$
$$\boldsymbol{c_3} \,:\, (en_0 \geq \ell + x) \wedge (en_1 \geq 2\ell - y)$$
$$\boldsymbol{c_4} \,:\, (en_0 \geq y - x) \wedge (en_1 \geq y - x) \wedge (en_0 + en_1 \geq min(3\ell + y - x, 2\ell - x + 3y))$$

Same for the synchronous case

And for trees?

Almost tight algorithms

Thank you

How many easy algorithms are there?

We found the necessary conditions and the corresponding algorithms

$$\boldsymbol{c_1} : (en_0 \geq x + y) \wedge (en_1 \geq y) \wedge (en_0 + en_1 \geq 2\ell + x + y)$$
$$\boldsymbol{c_2} : (en_0 \geq \ell - x) \wedge (en_1 \geq 2\ell - (x + y)) \wedge (en_0 + en_1 \geq 4\ell - (x + y))$$
$$\boldsymbol{c_3} : (en_0 \geq \ell + x) \wedge (en_1 \geq 2\ell - y)$$
$$\boldsymbol{c_4} : (en_0 \geq y - x) \wedge (en_1 \geq y - x) \wedge (en_0 + en_1 \geq min(3\ell + y - x, 2\ell - x + 3y))$$

Same for the synchronous case

And for trees?

Almost tight algorithms

Thank you

Synchronous lines

Action a_1

r_0 : $(v_0) \leftarrow; \rightarrow (v_{n-1})$

r_1 :

wait $x + y$ *time units for* r_0 **then**

$\rightarrow (v_{n-1})$

timeout

$(r_0) \leftarrow$;

if $\overleftarrow{r_0}$ **then**

$(v_0) \leftarrow$; $\rightarrow (v_{n-1})$

else $\rightarrow (v_{n-1})$;

end

Action a_2

r_0 :

wait $2\ell - (x + y)$ *time units for* r_1

then

$(v_0) \leftarrow$

timeout

$\rightarrow (r_1)$;

if $\overrightarrow{r_1}$ **then**

$\rightarrow (v_{n-1})$; $(v_0) \leftarrow$

else $(v_0) \leftarrow$;

end

r_1 : $\rightarrow (v_{n-1}); (v_0) \leftarrow$

Action a_3

r_0 : $(v_0) \leftarrow; \rightarrow (v_{n-1})$;

r_1 : $\rightarrow (v_{n-1}); (v_0) \leftarrow$

Action a_4

r_0 :

wait $y - x$ *time units for* r_1 **then**

wait $2x$ *time units for* r_1 **then**

$\rightarrow (v_{n-1})$

timeout

$(r_1) \leftarrow$;

if $\overrightarrow{r_1}$ **then** $\rightarrow (v_{n-1})$

else $(v_0) \leftarrow$; $\rightarrow (v_{n-1})$

end

timeout

$\rightarrow (r_1)$;

if $v_0 \prec v_{n-1}$ **then**

$(v_0) \leftarrow$; $\rightarrow (v_{n-1})$;

else $\rightarrow (v_{n-1})$; $(v_0) \leftarrow$;

end

r_1 : $(v_0) \leftarrow; \rightarrow (v_{n-1})$

Trees

$ct_1 \; : (en_0 \geq x) \wedge (en_1 \geq x) \wedge (en_0 + en_1 \geq 2W + 2d\lceil \log_{3/2} W \rceil + x + 2)$

The following shows the actions of the agents. It is assumed that subtrees $T_1, T_2, \ldots T_k$ ($k \leq \lceil \log_{3/2} W \rceil$) are a priori determined by the recursive centroid-based partitions.

Step 0: The agents meet on the shortest path between their initial locations. (This is executed only once at the beginning of the execution.) Set $i = 1$.

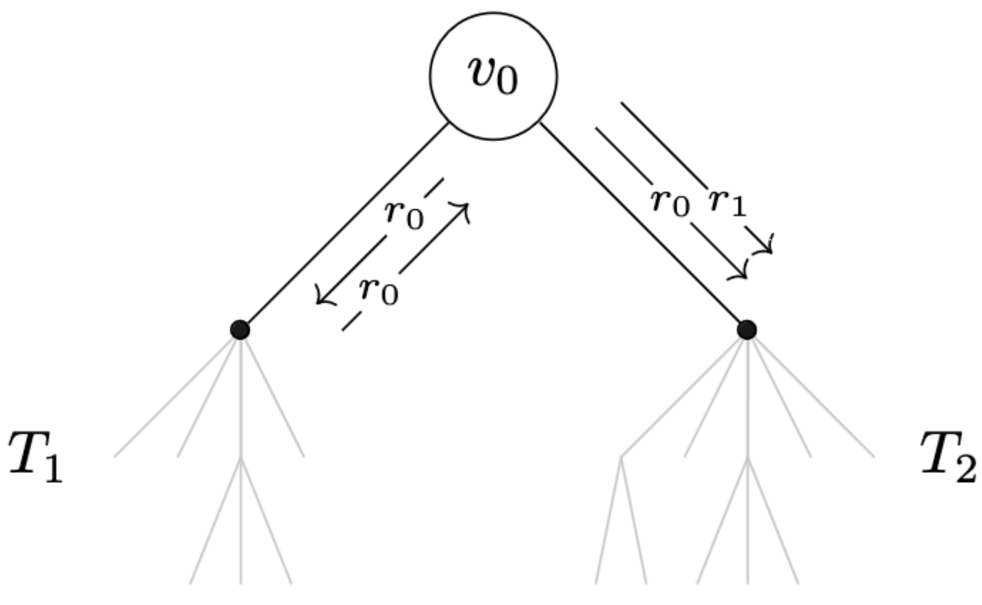
Step 1: When they meet at a point, say p (possibly on an edge), they evenly share the remaining energy.

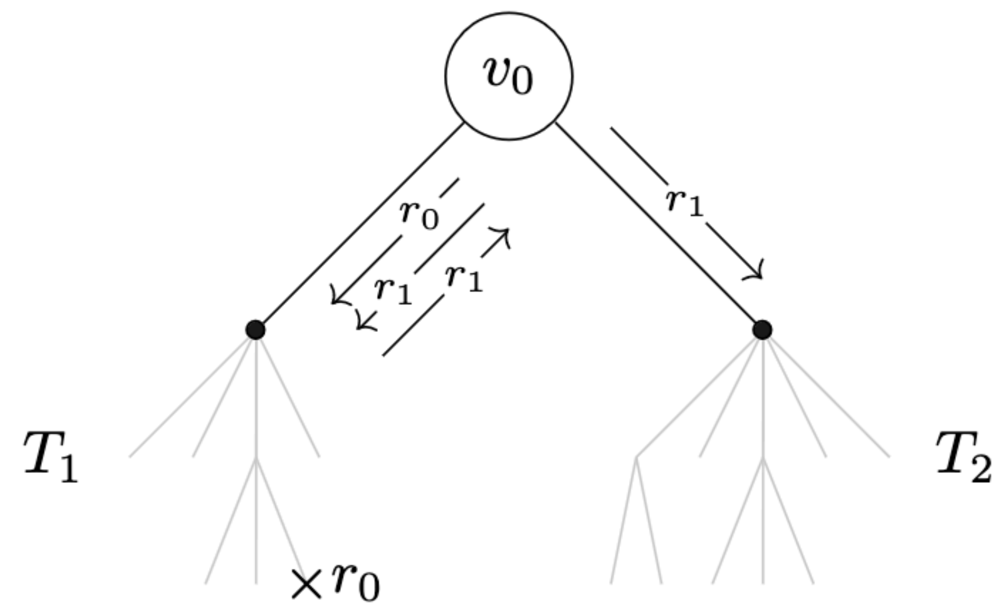
Step 2 Agent r_0 (resp. r_1) performs the following sequence of moves: move to the nearest node v_i of T_i; traverse T_i along a Eulerian tour of T_i in the clockwise direction (resp. the counter-clockwise direction) until the agent *(i)* meets the other, or *(ii)* completes the Eulerian tour traversal without meeting the other; move toward p until the agent meets the other in case of *ii)* if $i < k$; If $i < k$, set p be the meeting point, set $i = i + 1$, and continue from **Step 1**.

▶ **Theorem 12.** *If the agents start at the center of an unweighted star of size $\Delta + 1$, then the total energy consumption of any algorithm cannot be in $2\Delta + 2\log(o(\Delta))$.*

Trees

$$ct_2 \ : \ (en_0 \geq x) \wedge (en_1 \geq x) \wedge (en_0 + en_1 \geq 2W + d + x)$$





► **Theorem 13.** *There exists an infinite family of trees such that the required total energy is at least* $2W + \frac{d}{2} - 3$