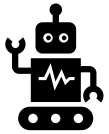
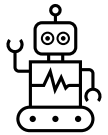
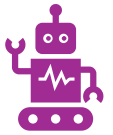
Optimal Deterministic Rendezvous in Labeled Lines

Yann Bourreau, Ananth Narayanan, Alexandre Nolin

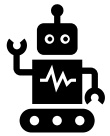
The Rendez-Vous Problem

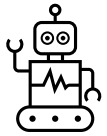


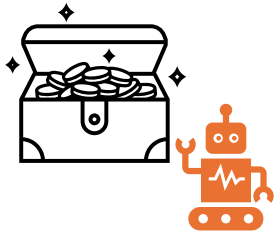


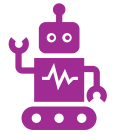


The Rendez-Vous Problem

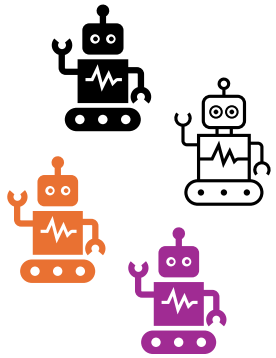




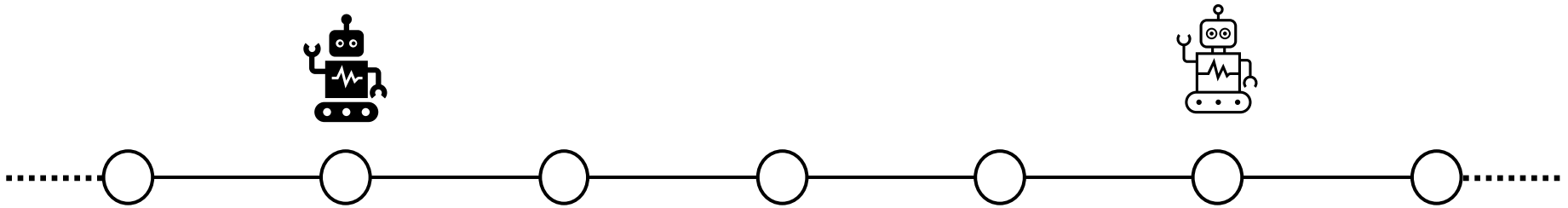




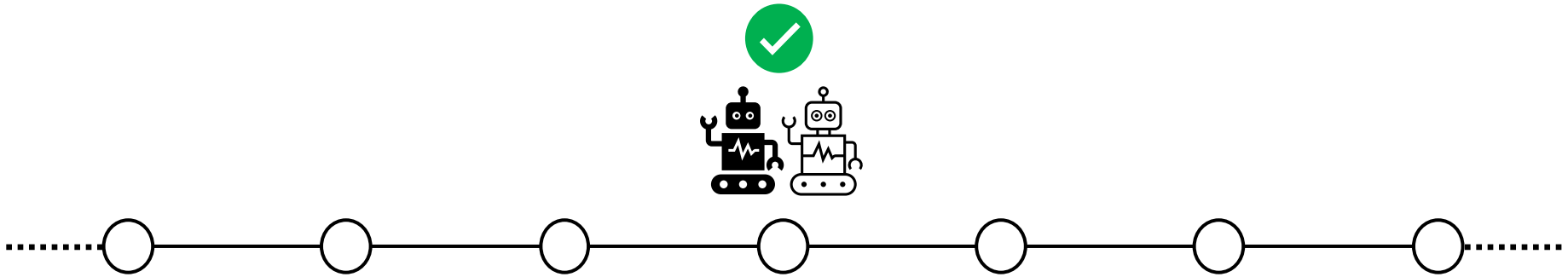
The Rendez-Vous Problem



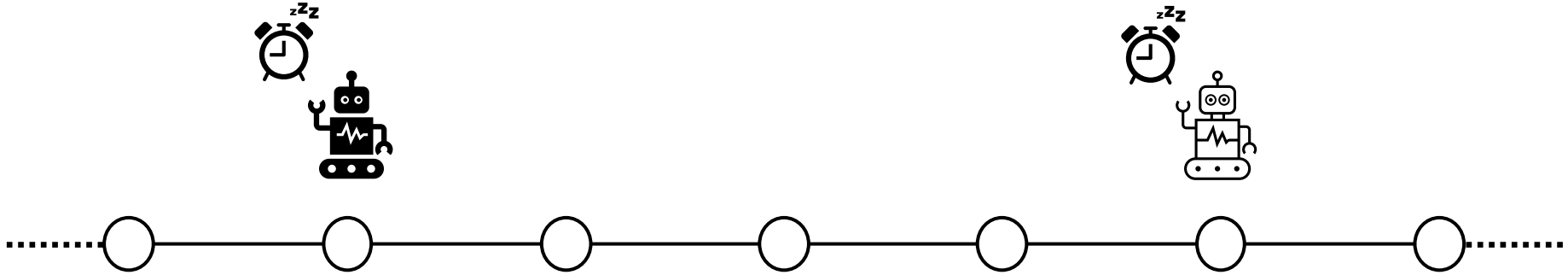
The Rendez-Vous Problem



The Rendez-Vous Problem

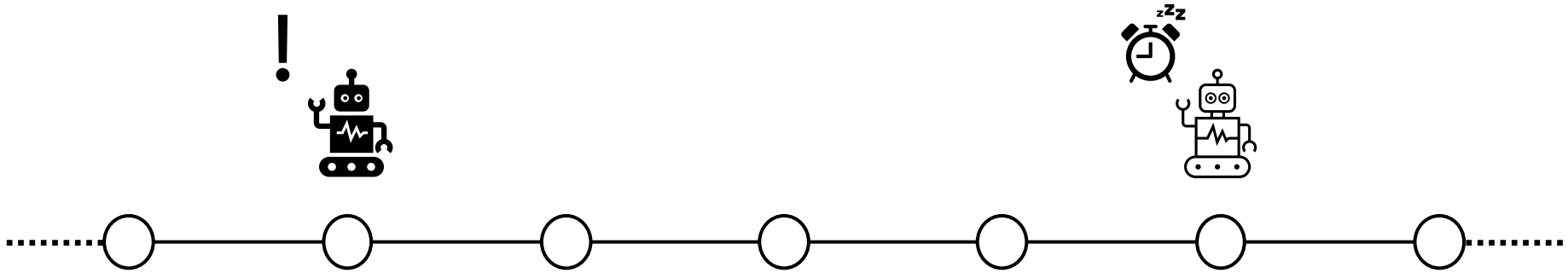


Model settings



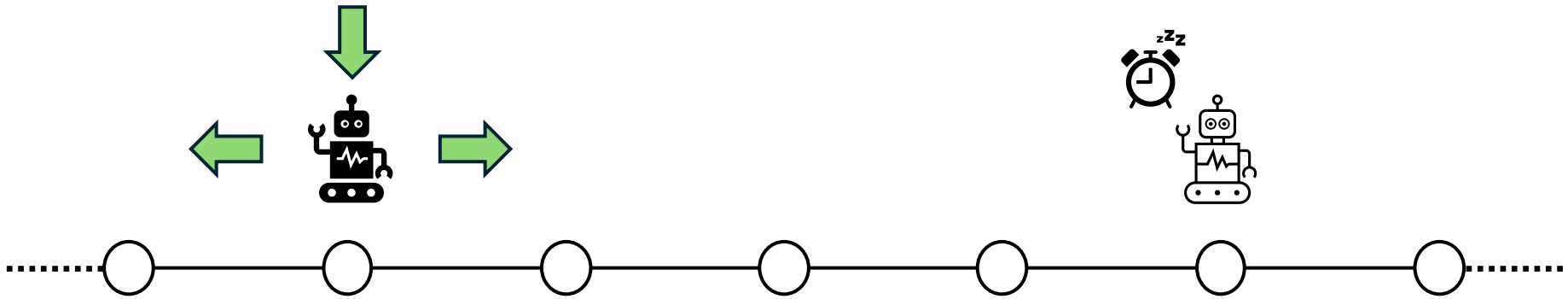
- Adversarily chosen wake up time
- Synchronous rounds

Model settings



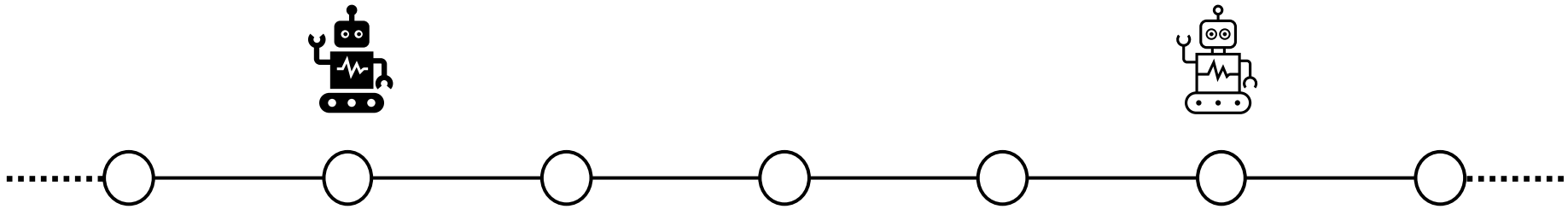
- Adversarily chosen wake up time
- Synchronous rounds

Model settings



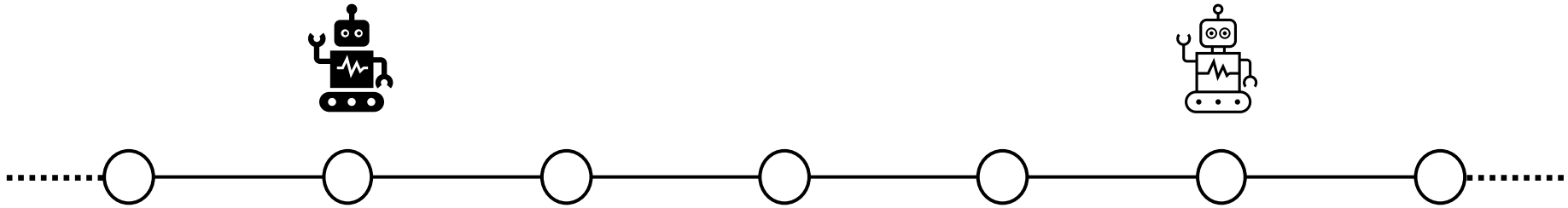
- Adversarily chosen wake up time
- Synchronous rounds

Model settings



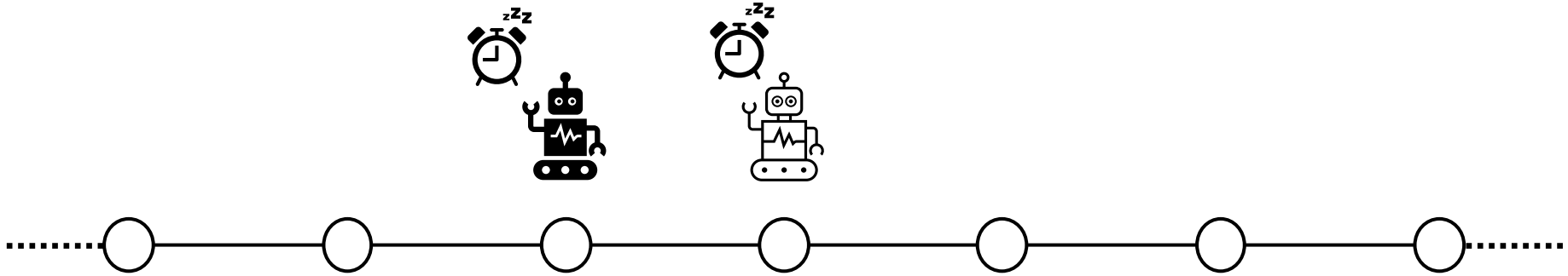
- Adversarily chosen wake up time
- Synchronous rounds
- Rendez-vous: Same node during same round

Model settings



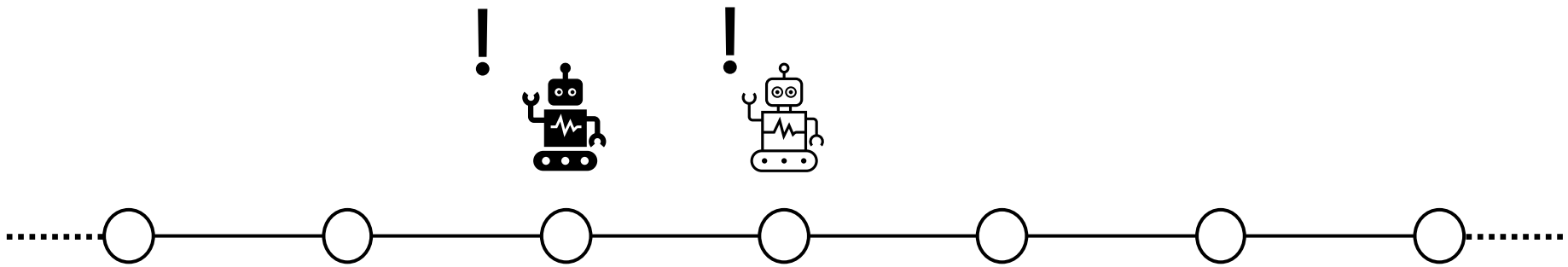
- Adversarily chosen wake up time
- Synchronous rounds
- Rendez-vous: Same node during same round
- Same **deterministic** algorithm
- Robots see each other only on the same node

Why do we need IDs?



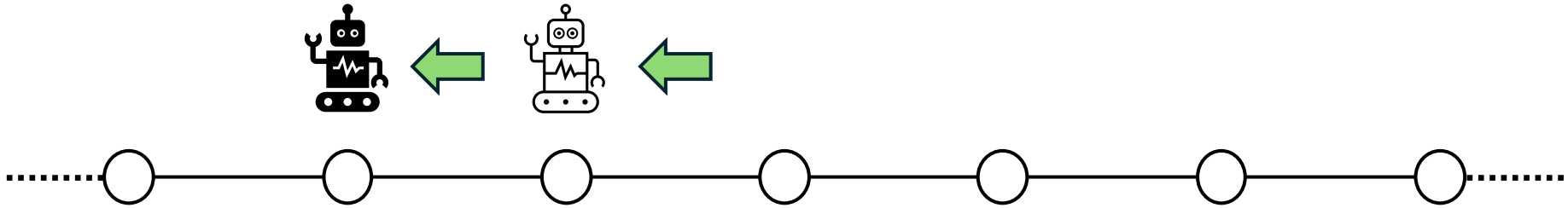
- Adversarily chosen wake up time
- Synchronous rounds
- Rendez-vous: Same node during same round
- Same **deterministic** algorithm
- Robots see what is on their node

Why do we need IDs?



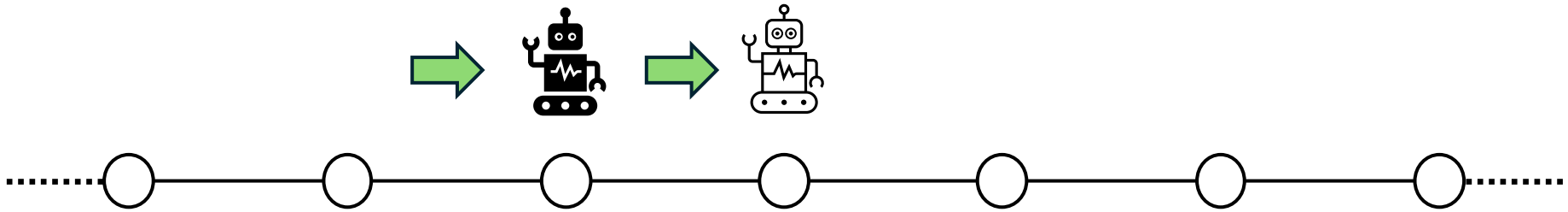
- Adversarily chosen wake up time
- Synchronous rounds
- Rendez-vous: Same node during same round
- Same **deterministic** algorithm
- Robots see what is on their node

Why do we need IDs?



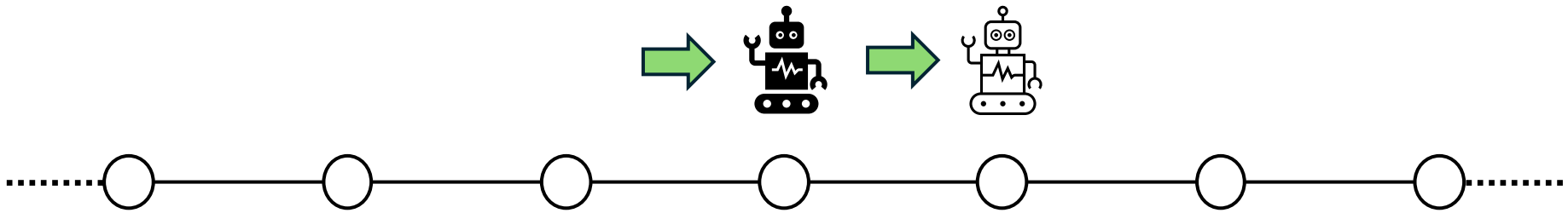
- Adversarily chosen wake up time
- Synchronous rounds
- Rendez-vous: Same node during same round
- Same **deterministic** algorithm
- Robots see what is on their node

Why do we need IDs?



- Adversarily chosen wake up time
- Synchronous rounds
- Rendez-vous: Same node during same round
- Same **deterministic** algorithm
- Robots see what is on their node

Why do we need IDs?

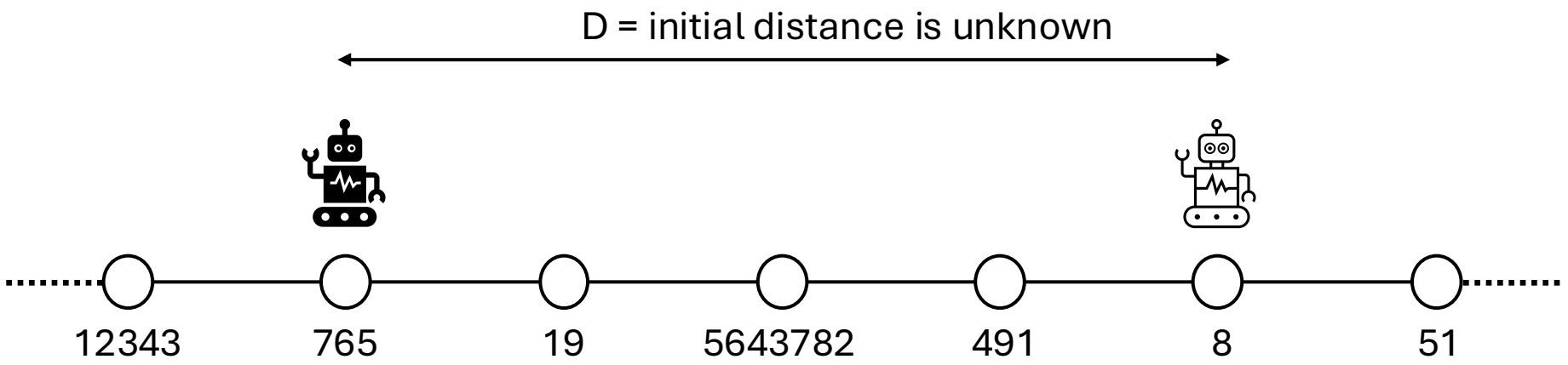


- Adversarily chosen wake up time
- Synchronous rounds
- Rendez-vous: Same node during same round
- Same **deterministic** algorithm
- Robots see what is on their node

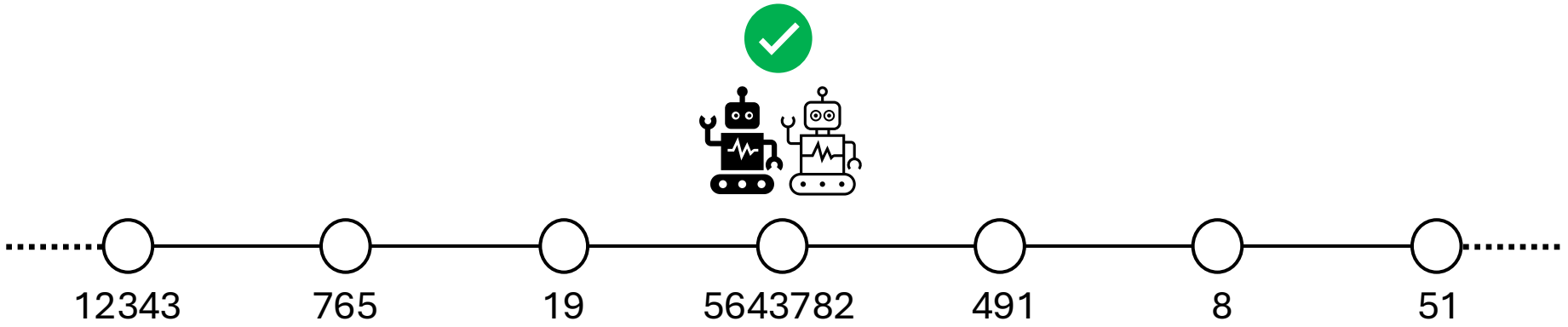
not enough to achieve Rendez-vous

Problem setting

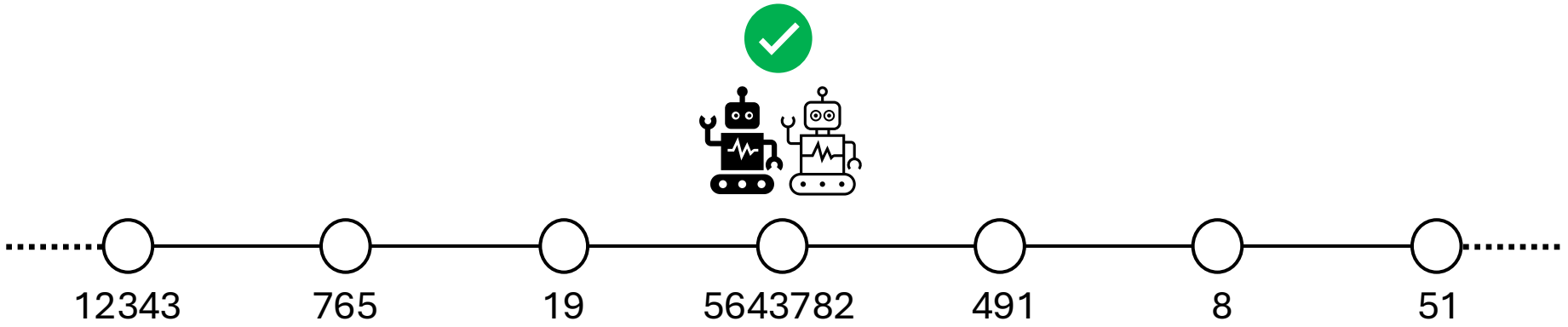
Synchronous rounds and infinite memory



Problem setting



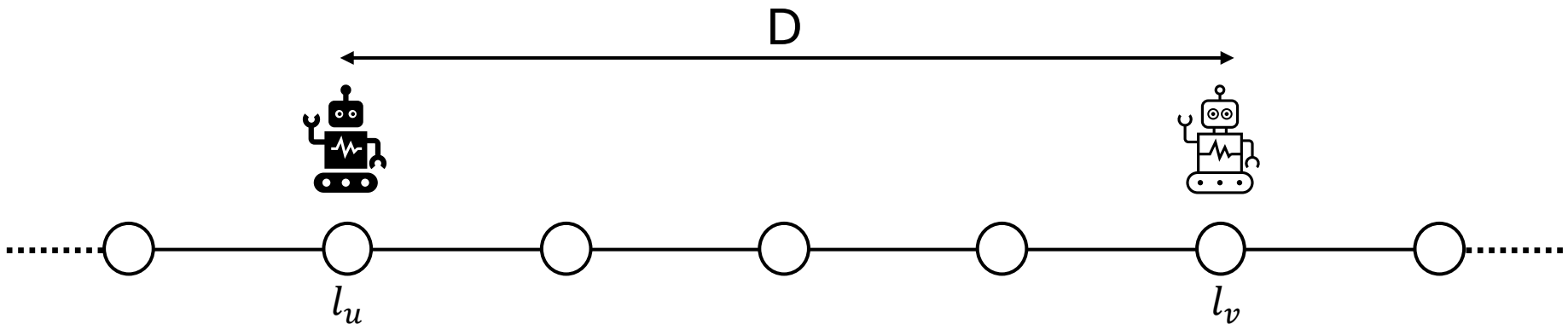
Problem setting



How long does it take?

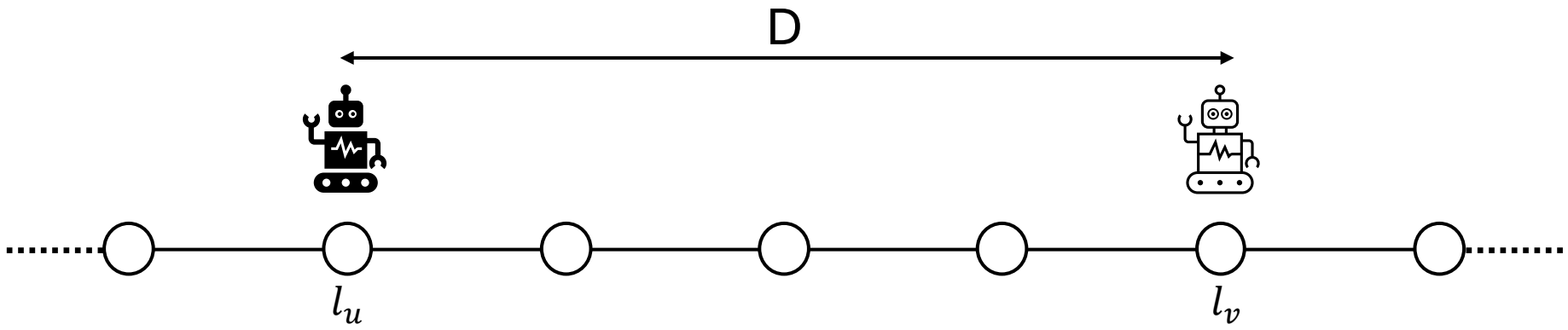
Previous work and our results

$$l_{max} = max(l_u, l_v)$$



Previous work and our results

$$l_{max} = max(l_u, l_v)$$

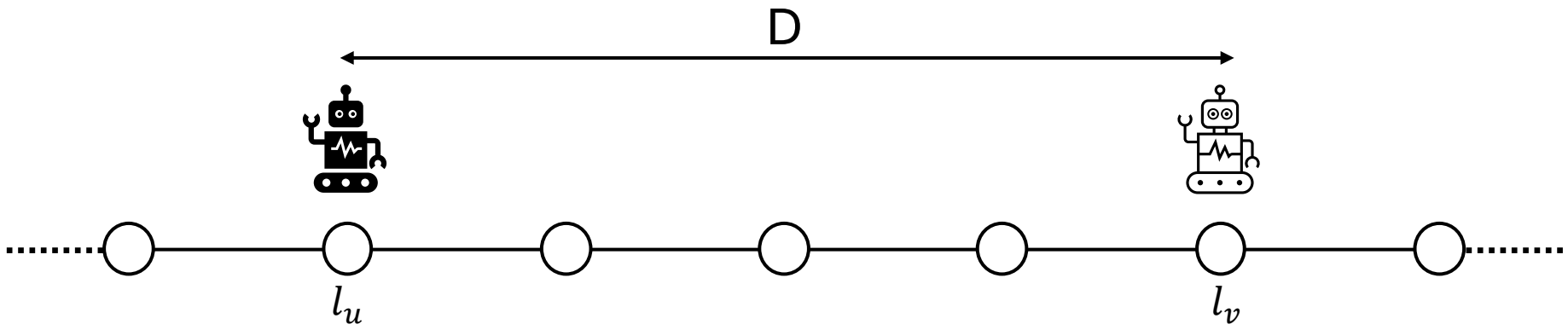


- $\Omega(D \cdot \log^*(l_{max}))$ - Miller and Pelc 2023
- $O(D^2 \cdot (\log^*(l_{max}))^3)$ - Miller and Pelc 2023

Measured when first agent starts

Previous work and our results

$$l_{max} = max(l_u, l_v)$$

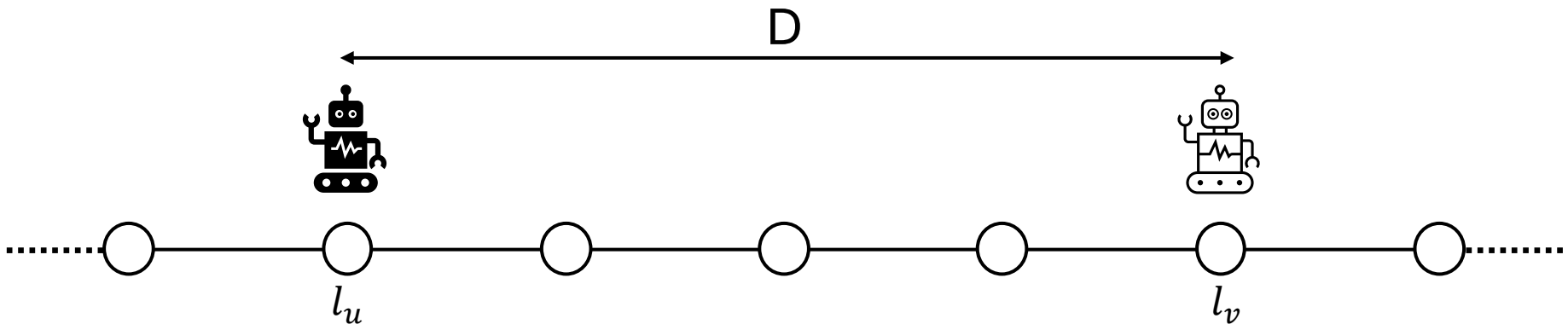


- $\Omega(D \cdot \log^*(l_{max}))$ - Miller and Pelc 2023
- $O(D^2 \cdot (\log^*(l_{max}))^3)$ - Miller and Pelc 2023

$$\log^* \mathrm{k} = \begin{cases} 0 \text{ if } \mathrm{k} \leq 1 \\ 1 + \log * (\log \mathrm{k}) \text{ if } \mathrm{k} > 1 \end{cases}$$

Previous work and our results

$$l_{max} = max(l_u, l_v)$$

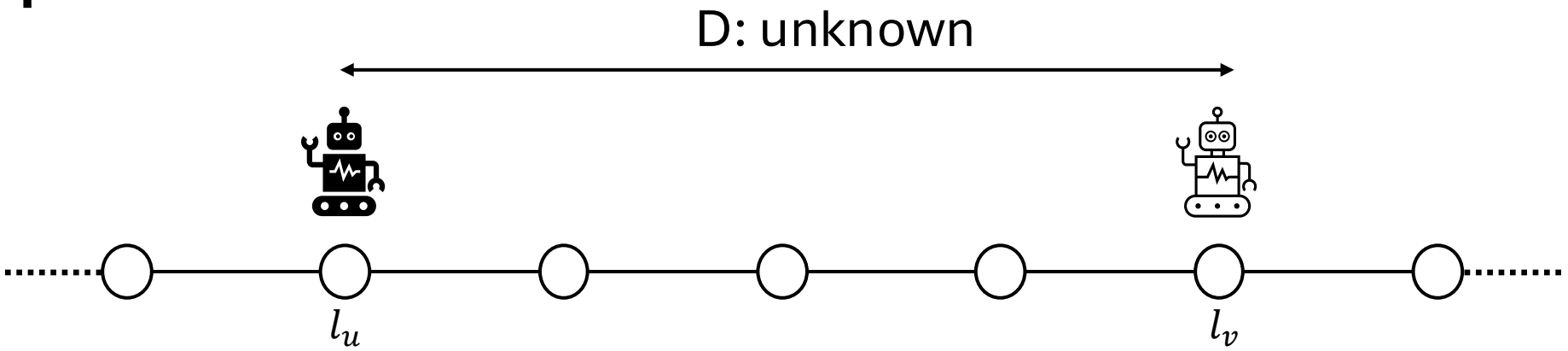


- $\Omega(D \cdot \log^*(l_{max}))$ - Miller and Pelc 2023
- $O(D^2 \cdot (\log^*(l_{max}))^3)$ - Miller and Pelc 2023

$$\log^* \mathrm{k} = \begin{cases} 0 \text{ if } \mathrm{k} \leq 1 \\ 1 + \log * (\log \mathrm{k}) \text{ if } \mathrm{k} > 1 \end{cases}$$

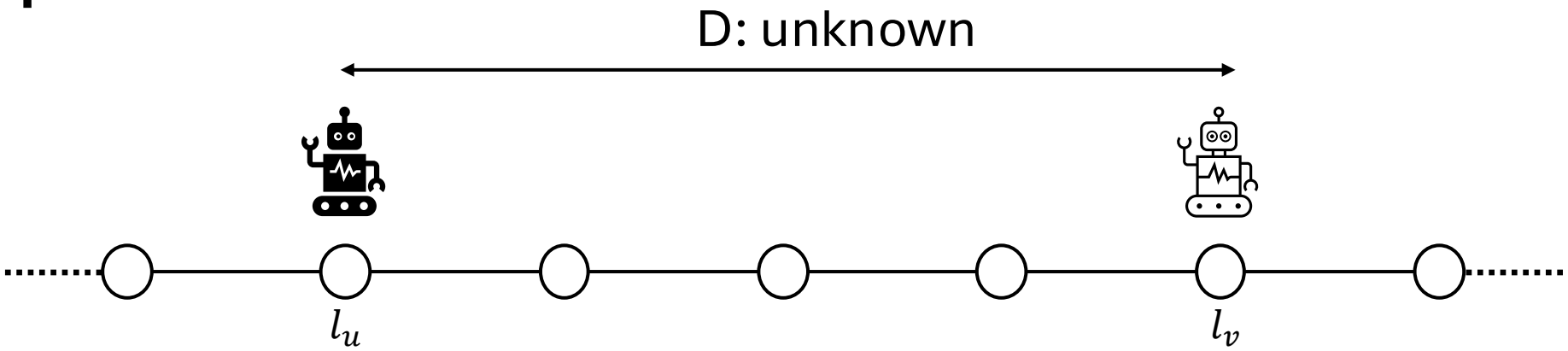
Our result: $O(\mathrm{D} \cdot \log^*(l_{max}))$

Simple case



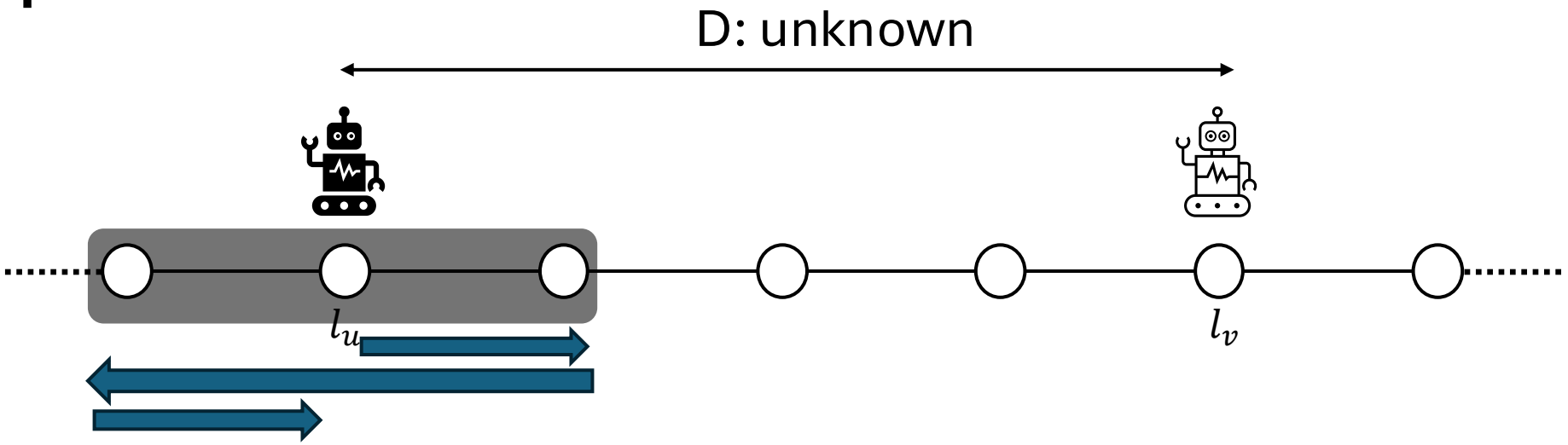
- 2 different algorithms and same wake up time

Simple case



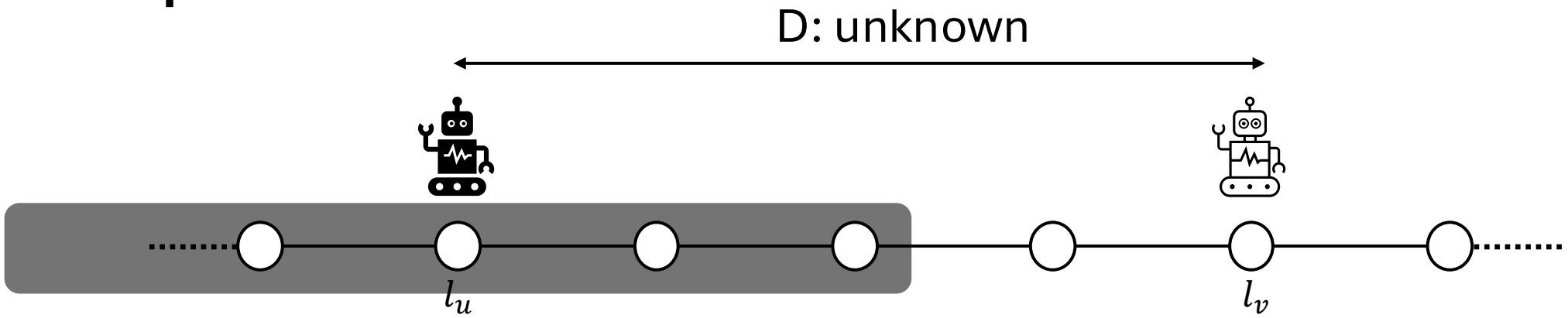
- 2 different algorithms and same wake up time
- Have one robot search and the other one stay!

Simple case



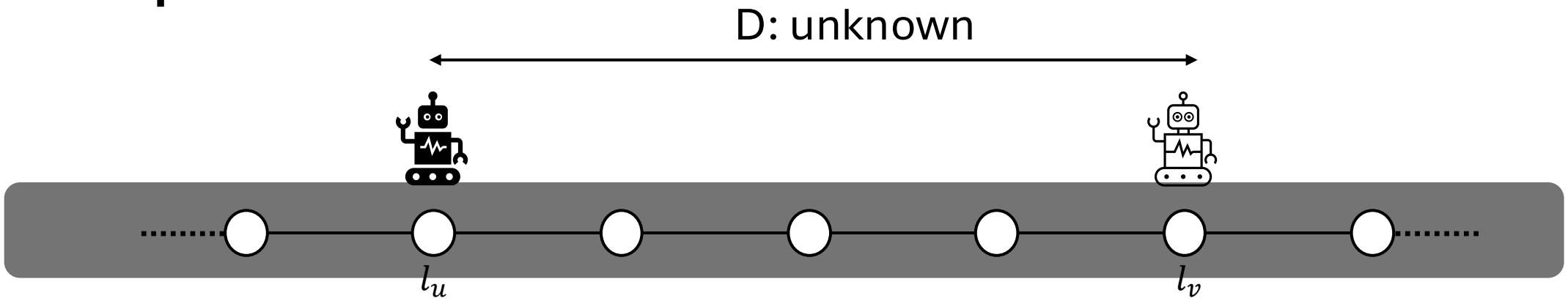
- 2 different algorithms and same wake up time
- Have one robot search and the other one stay!

Simple case



- 2 different algorithms and same wake up time
- Have one robot search and the other one stay!

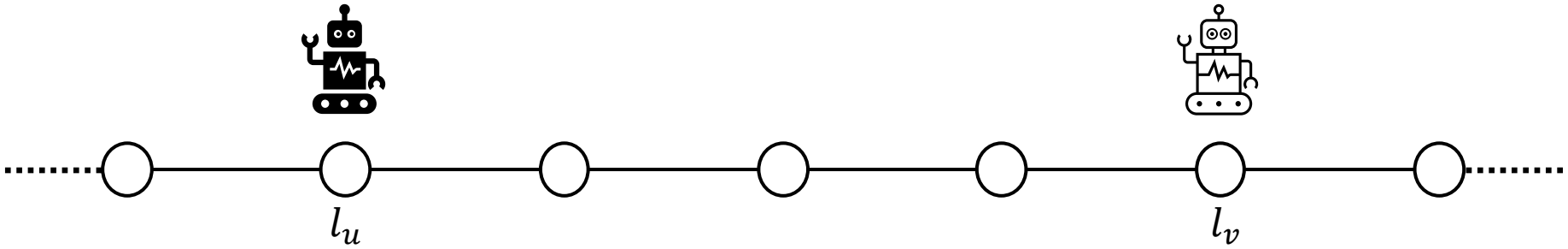
Simple case



- 2 different algorithms and same wake up time
- Have one robot search and the other one stay!

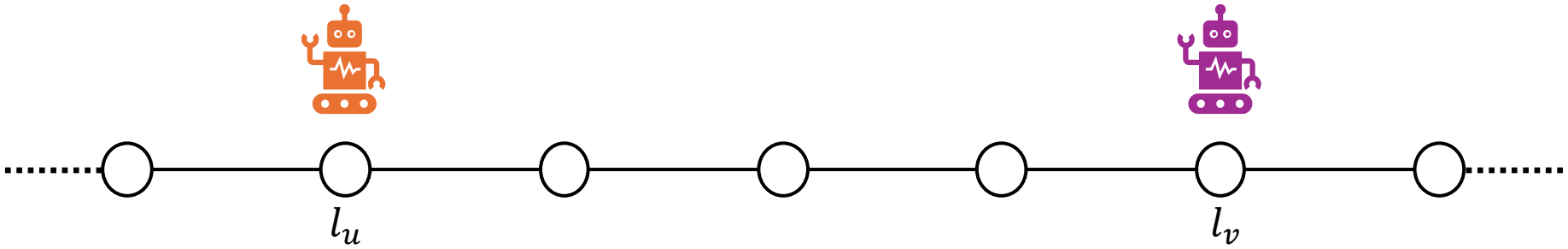
Rendez-Vous: O(D)

Algorithm sketch



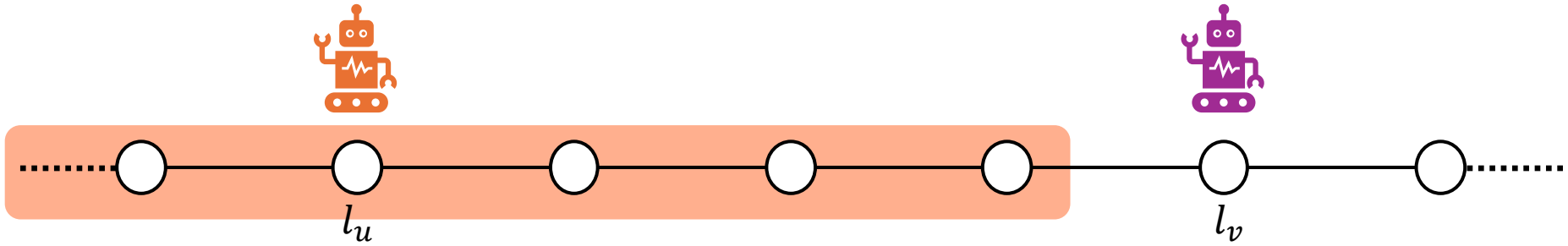
1. Color each agent

Algorithm sketch



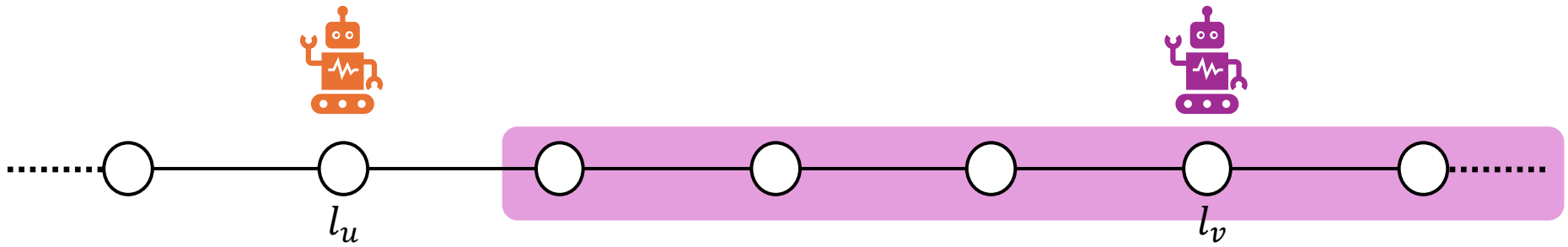
1. Color each agent

Algorithm sketch



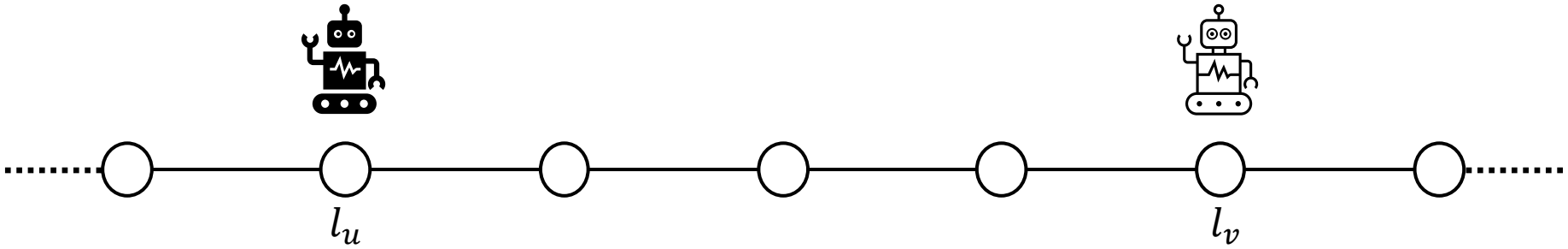
1. Color each agent
2. Iterate through color class -> search k-hop radius

Algorithm sketch



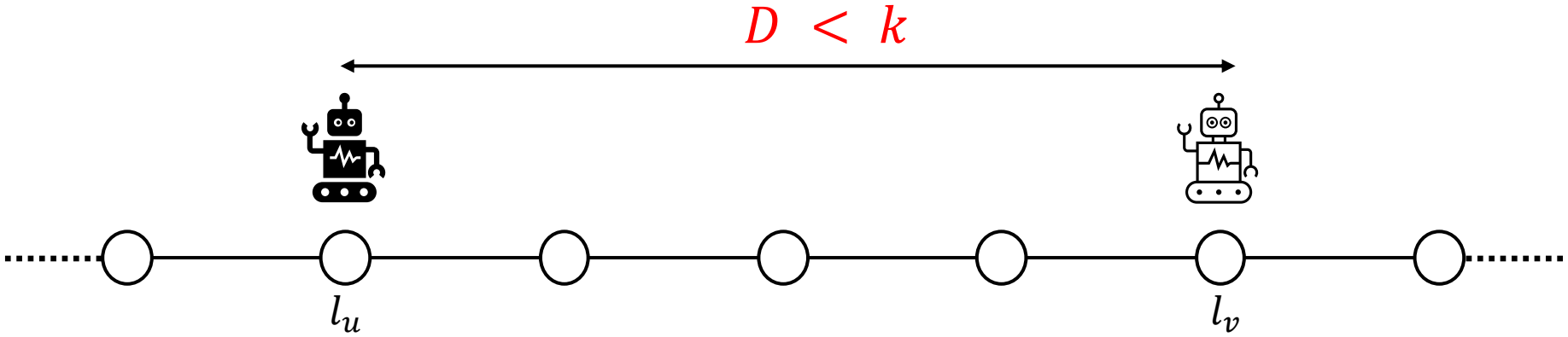
1. Color each agent
2. Iterate through color class -> search k-hop radius

Algorithm sketch



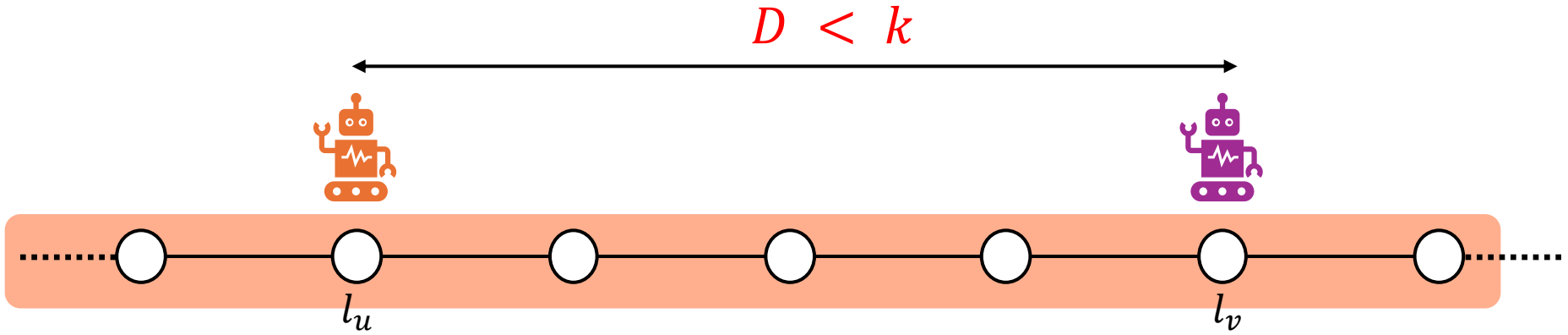
1. Color each agent
2. Iterate through color class -> search k-hop radius
3. Double k and move to step 1

Algorithm sketch



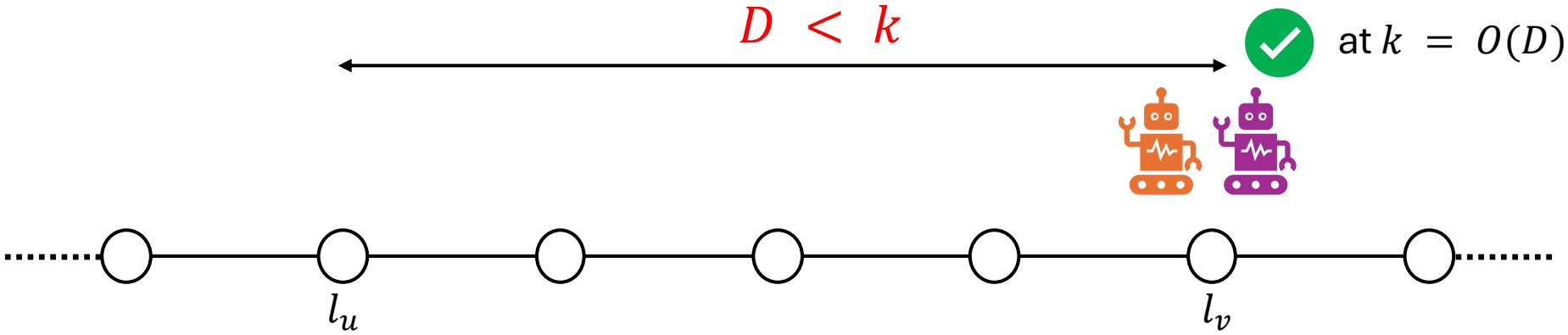
1. Color each agent
2. Iterate through color class -> search k-hop radius
3. Double k and move to step 1

Algorithm sketch



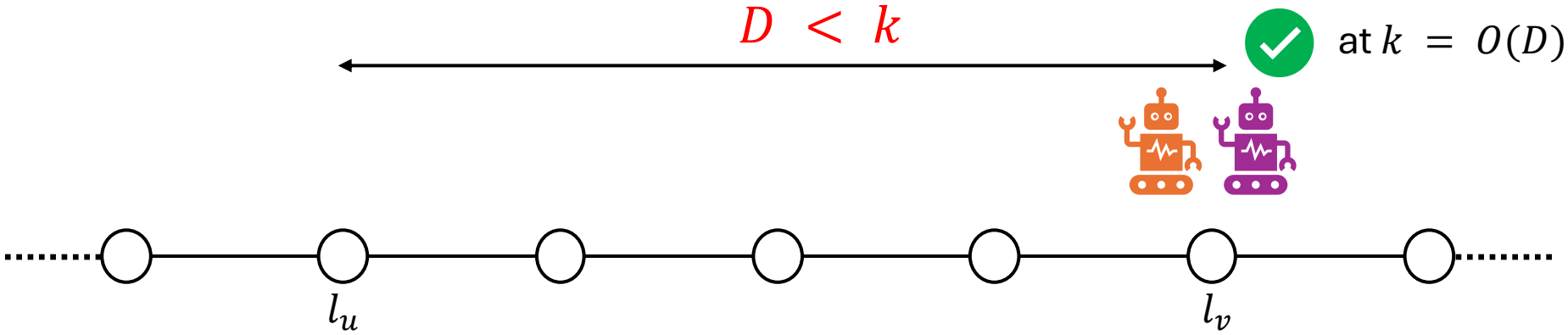
1. Color each agent
2. Iterate through color class -> search k-hop radius
3. Double k and move to step 1

Algorithm sketch



1. Color each agent
2. Iterate through color class -> search k-hop radius
3. Double k and move to step 1

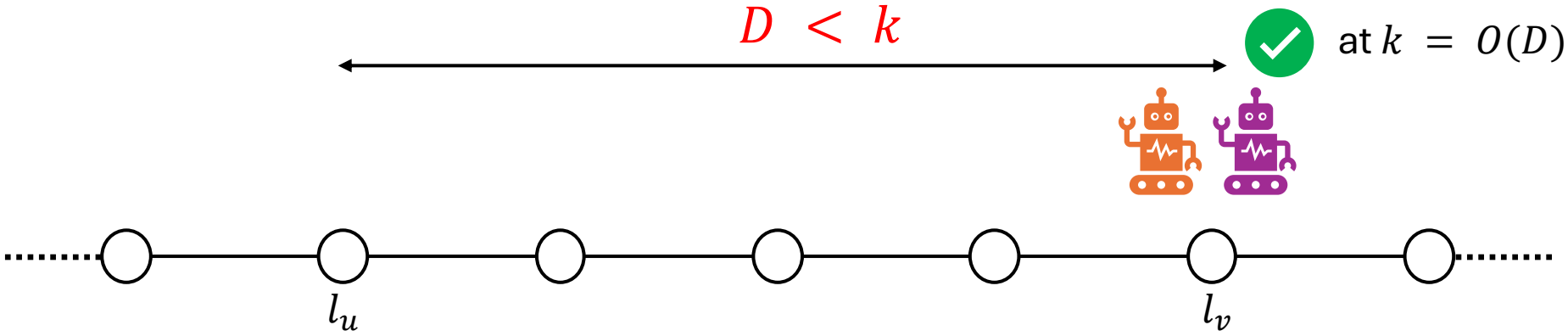
Algorithm sketch



1. Color each agent
2. Iterate through color class -> search k-hop radius
3. Double k and move to step 1

Runtime: $\underbrace{\textbf{Searching Phase}}_{O(D)}$ **+** **Coloring Phase**

Algorithm sketch



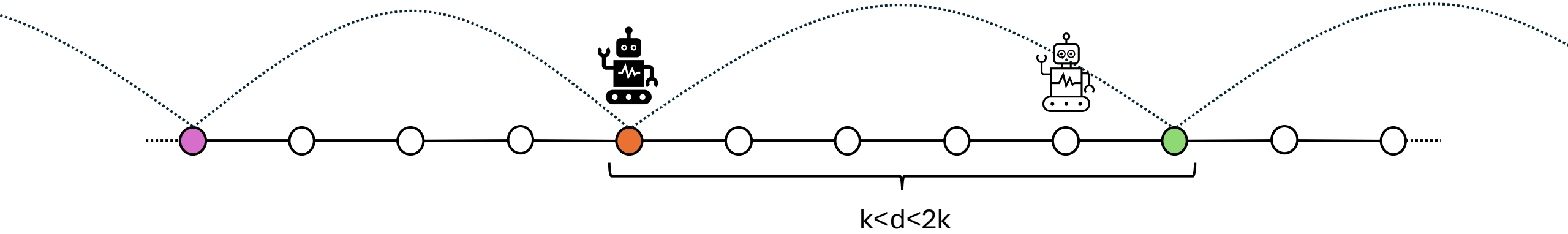
1. Color each agent
2. Iterate through color class -> search k-hop radius
3. Double k and move to step 1

Runtime: **Searching Phase** **+** **Coloring Phase**

$O(D)$ Why? Color palette = constant

Idea for coloring agents

Colored ruling sets!!

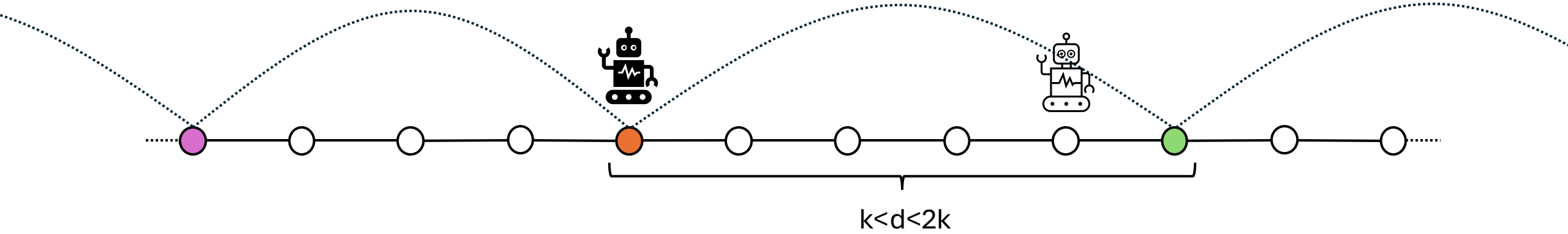


$S_k \subset V$:

- Roughly equivalently distant

Idea for coloring agents

Colored ruling sets!!

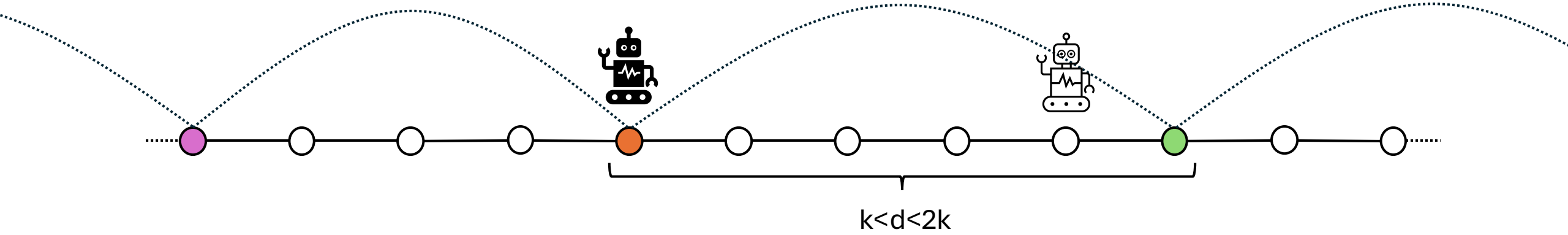


$S_k \subset V$:

- Roughly equivalently distant
- Colored with constant size palette

Idea for coloring agents

Colored ruling sets!!

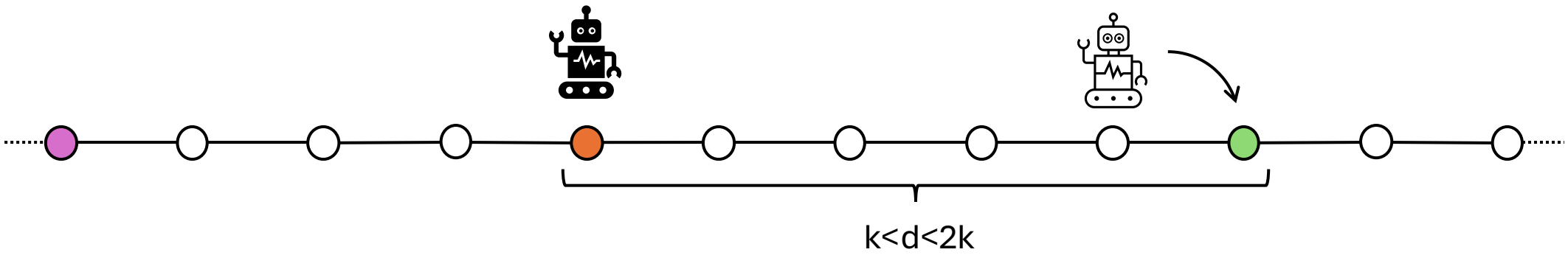


$S_k \subset V$:

- Roughly equivalently distant
- Colored with constant size palette
- Colored differently as nodes at distance 10k

Idea for coloring agents

Colored ruling sets!!

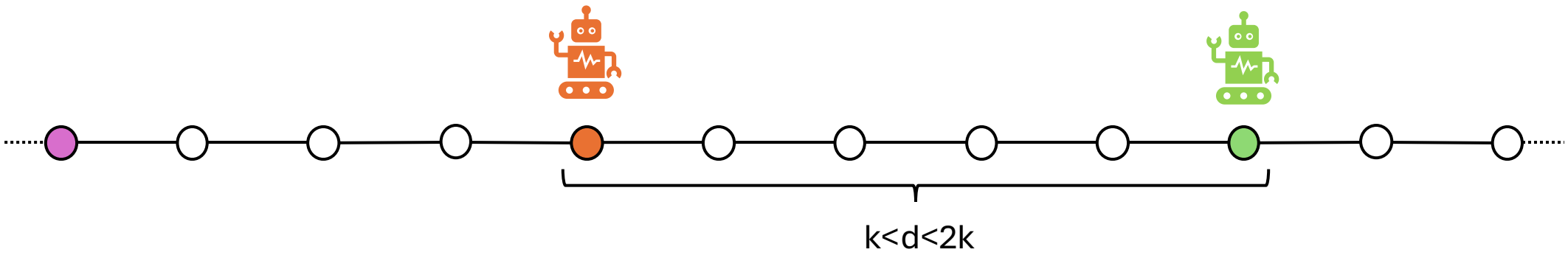


$S_k \subset V$:

- Roughly equivalently distant
- Colored with constant size palette
- Colored differently as nodes at distance 10k

Idea for coloring agents

Colored ruling sets!!

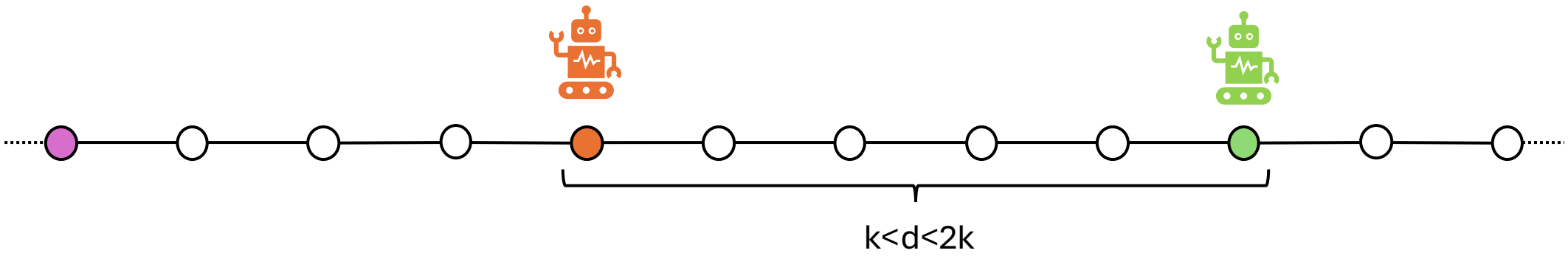


$S_k \subset V$:

- Roughly equivalently distant
- Colored with constant size palette
- Colored differently as nodes at distance 10k

Idea for coloring agents

Colored ruling sets!!



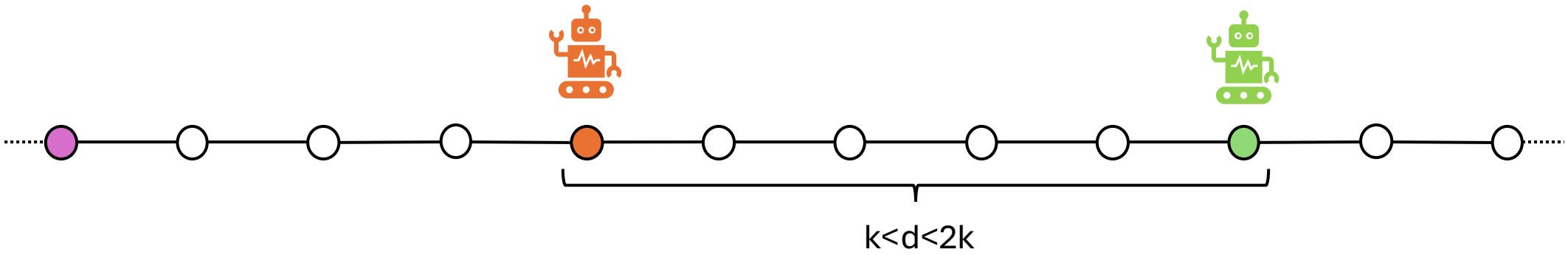
$S_k \subset V$:

- Roughly equivalently distant
- Colored with constant size palette
- Colored differently as nodes at distance 10k

Agents can find S_k in $O(k \log^* l_{max})$ rounds

Idea for coloring agents

Colored ruling sets!!



$S_k \subset V$:

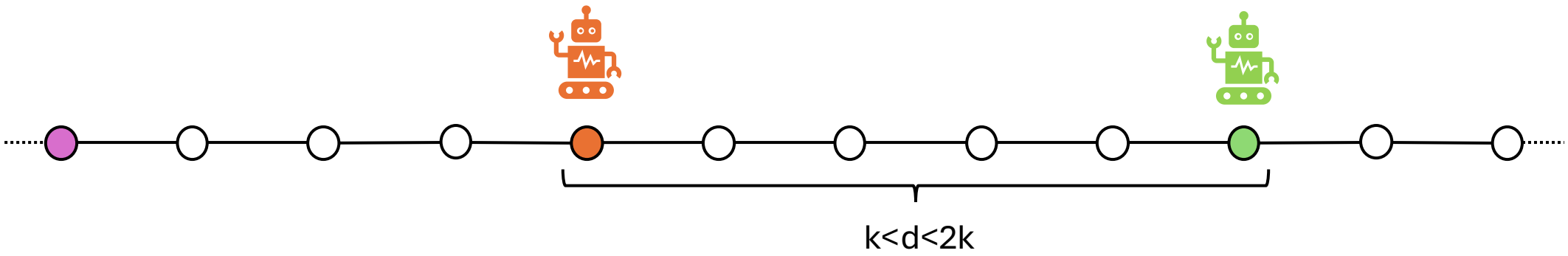
- Roughly equivalently distant
- Colored with constant size palette
- Colored differently as nodes at distance 10k

Agents can find S_k in $O(k \log^* l_{max})$ rounds

For k in {2,4,8,16,….,O(D)}

Idea for coloring agents

Colored ruling sets!!



$S_k \subset V$:

- Roughly equivalently distant
- Colored with constant size palette
- Colored differently as nodes at distance 10k

Agents can find S_k in $O(k \log^* l_{max})$ rounds

For k in {2,4,8,16,….,O(D)}

Coloring phase: $O(D \log^* l_{max})$

Running time

Runtime : $\underbrace{\textbf{Searching Phase}}_{O(D)}$ **+** $\underbrace{\textbf{Coloring Phase}}_{O(D \log^* l_{max})}$

Rendezvous in $O(D \log^* l_{max})$

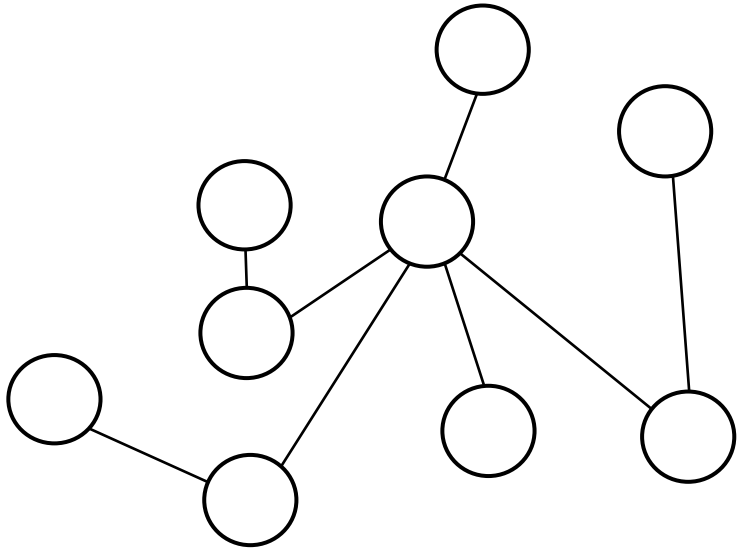
Obtaining the ruling set

Agents can find S_k in $O(k \log^* l_{max})$ rounds

Proved using another model: The LOCAL model of distributed computing

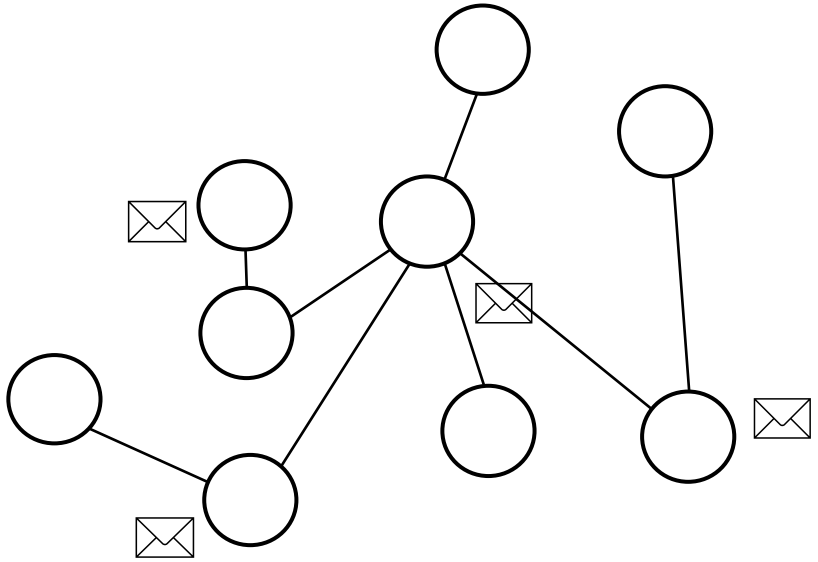
LOCAL model

- Synchronous model:
  - messages of unlimited size
  - unbounded computation
- Undirected graph G
  - unique ID
- Complexity: number of rounds until last node terminates



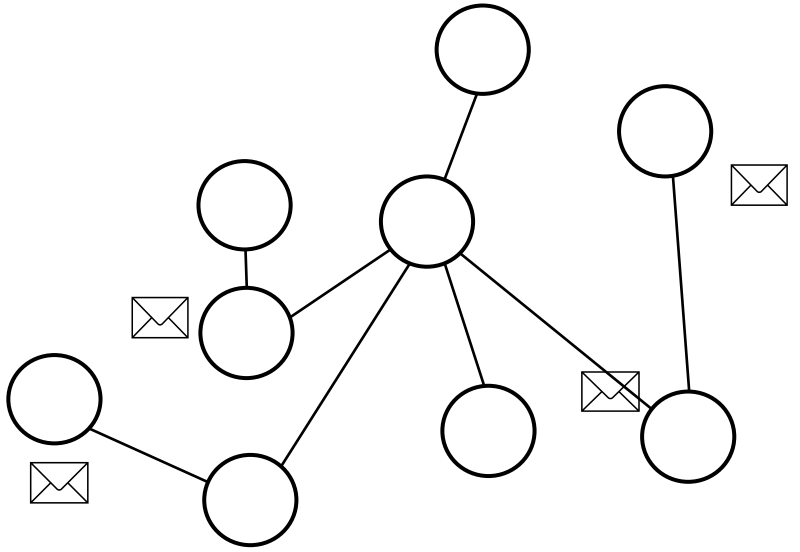
LOCAL model

- Synchronous model:
  - messages of unlimited size
  - unbounded computation
- Undirected graph G
  - n nodes
  - maximum degree Δ
  - unique ID
- Complexity: number of rounds until last node terminates



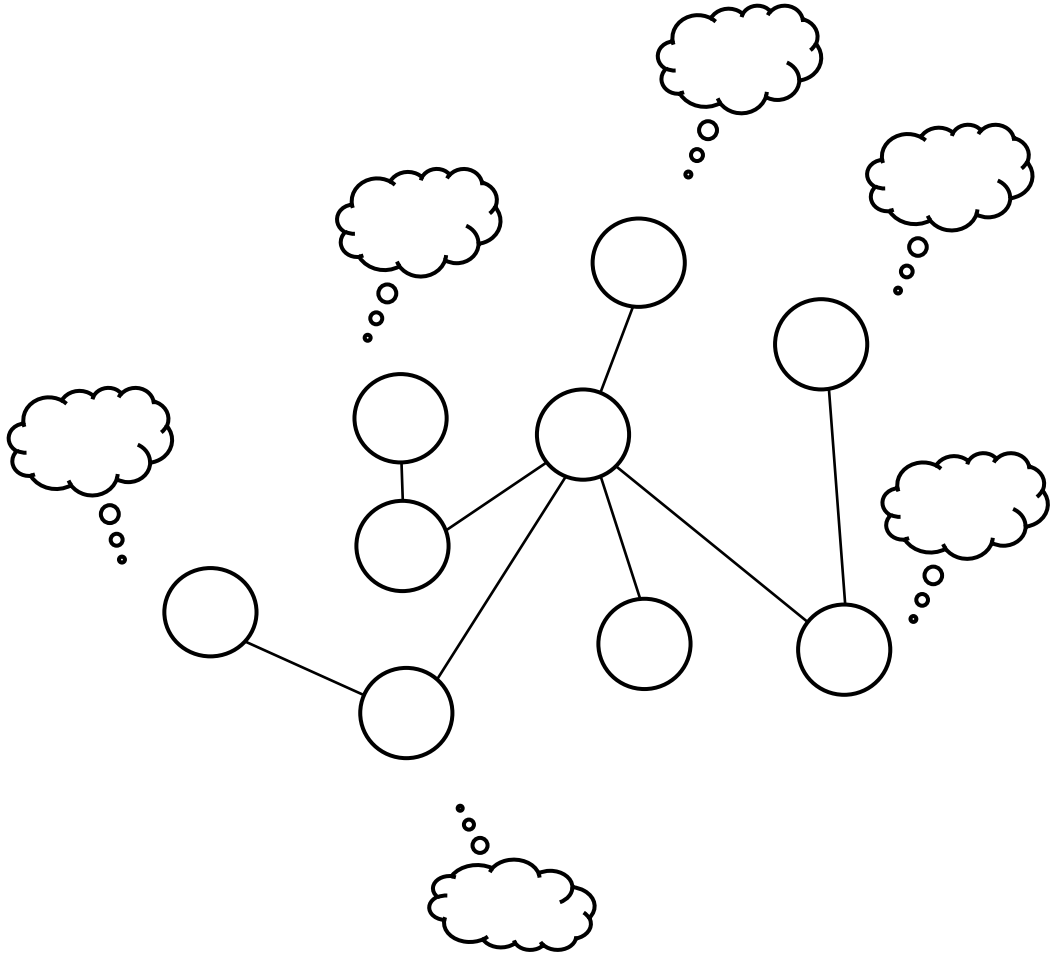
LOCAL model

- Synchronous model:
  - messages of unlimited size
  - unbounded computation
- Undirected graph G
  - n nodes
  - maximum degree Δ
  - unique ID
- Complexity: number of rounds until last node terminates

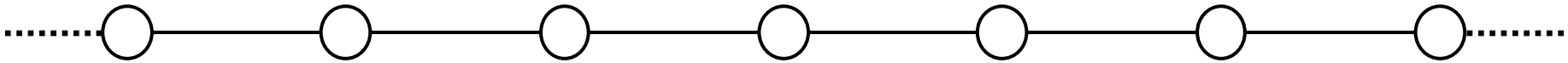


LOCAL model

- Synchronous model:
  - messages of unlimited size
  - unbounded computation
- Undirected graph G
  - n nodes
  - maximum degree Δ
  - unique ID
- Complexity: number of rounds until last node terminates



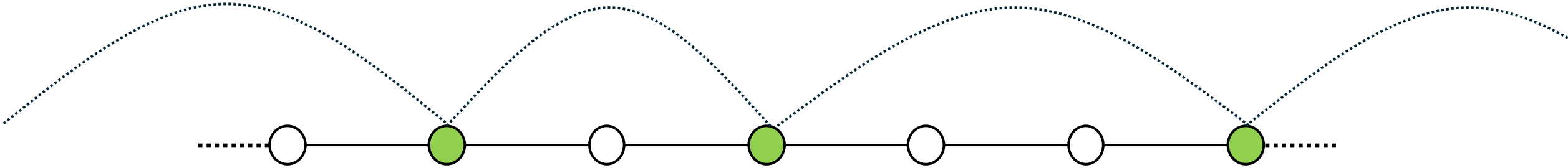
Colored ruling set in LOCAL



Colored ruling set in LOCAL

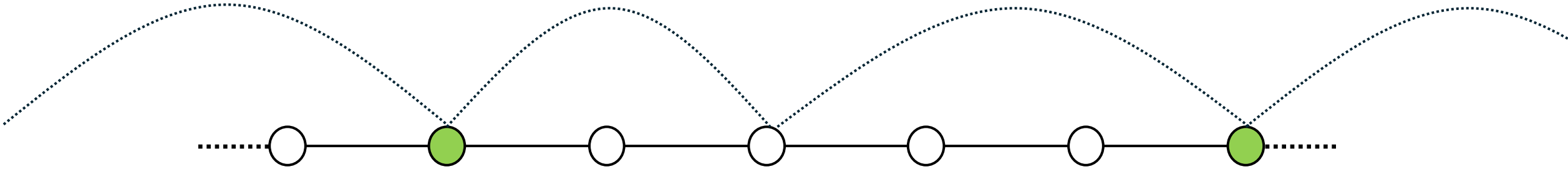
- Maximal independent set in $O(\log^* ID(v))$ rounds for every v

Colored ruling set in LOCAL



- Maximal independent set in $O(\log^* ID(v))$ rounds for every v
- Iterate on the subgraph induced by the node of the set

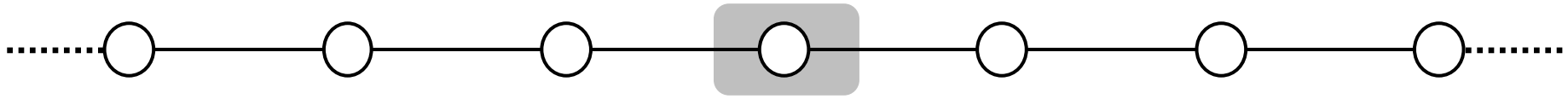
Colored ruling set in LOCAL



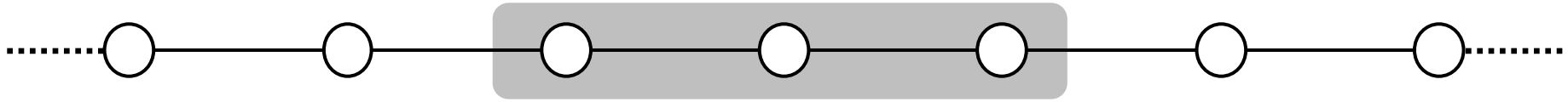
- Maximal independent set in $O(\log^* ID(v))$ rounds for every v
- Iterate on the subgraph induced by the node of the set

$$O(\mathrm{D} \log^* ID(v))$$

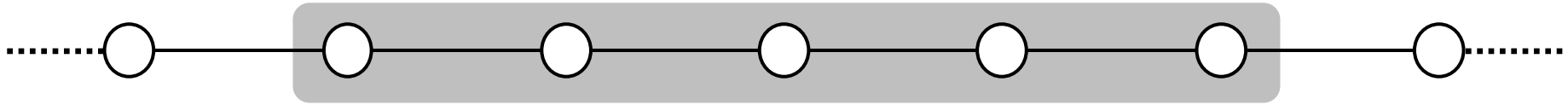
Equivalence LOCAL and agent setting



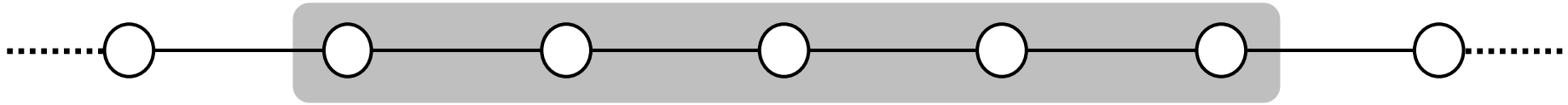
Equivalence LOCAL and agent setting



Equivalence LOCAL and agent setting



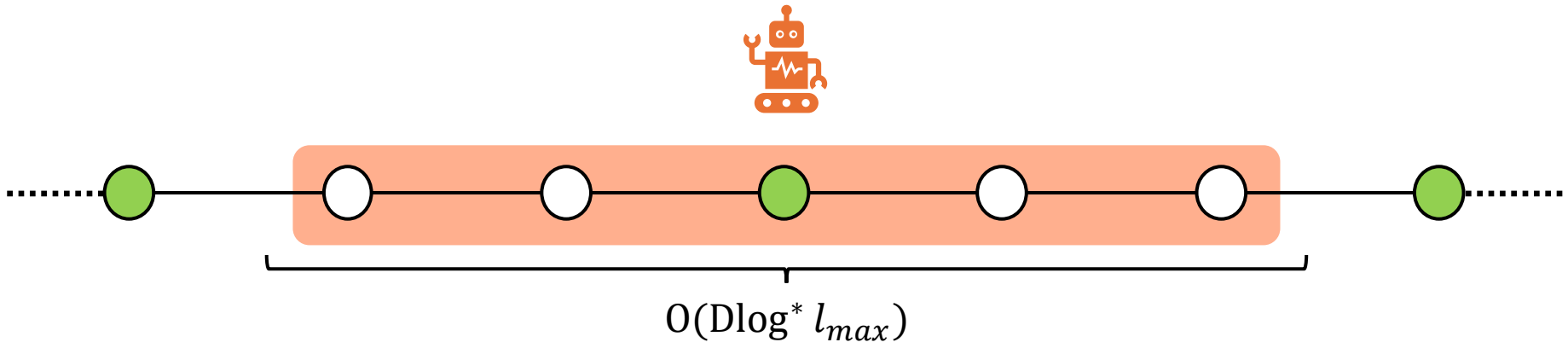
Equivalence LOCAL and agent setting



After $O(\log^* l_{max})$ rounds – sees the $O(\log^* l_{max})$-hop

Looking $O(D\log^* l_{max})$ neighborhood = messages during $O(D\log^* l_{max})$

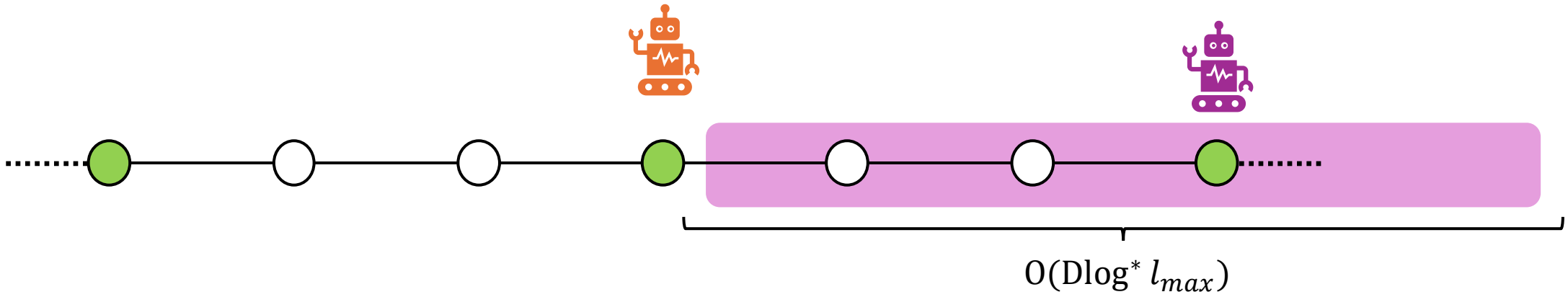
Equivalence LOCAL and agent setting



After $O(D\log^* l_{max})$ rounds – sees the $O(D\log^* l_{max})$-hop

Looking $O(D\log^* l_{max})$ neighborhood = messages during $O(D\log^* l_{max})$

Equivalence LOCAL and agent setting



After $O(D\log^* l_{max})$ rounds – sees the $O(D\log^* l_{max})$-hop

Looking $O(D\log^* l_{max})$ neighborhood = messages during $O(D\log^* l_{max})$

Exactly the same ruling set nodes

Conclusion

$\theta(\mathrm{D}\log^* l_{max})$ —rounds algorithm for labeled lines.

Open questions:

- Labeled trees?
- Labeled grids?