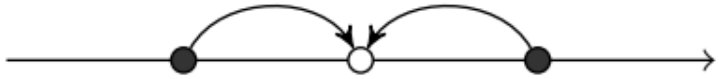


Stand Up Indulgent Rendezvous

Quentin Bramas, Anissa Lamani, Sébastien Tixeuil

Slides available at https://bramas.fr

Model

Model

- Two Robots

Model

- Two Robots
- No Communication

Model

- Two Robots
- No Communication
- No Memory

Model

- Two Robots
- No Communication
- No Memory
- Disoriented

Model

- Two Robots
- No Communication
- No Memory
- Disoriented
- No Common Unit-Distance

Model

Model

- What they do?

  Look (where is the other robot?)
  Compute (where am I going next?)
  Move (who am I?)

Model

- What they do?
  Look (where is the other robot?)
  Compute (where am I going next?)
  Move (who am I?)
- When?

Model

- What they do?

  Look (where is the other robot?)
  Compute (where am I going next?)
  Move (who am I?)

- When?
  - Fully-Synchronous:

    Robot 1: LCM LCM LCM LCM LCM LCM …
    Robot 2: LCM LCM LCM LCM LCM LCM …

Model

- What they do?

  Look (where is the other robot?)
  Compute (where am I going next?)
  Move (who am I?)

- When?
  - Fully-Synchronous:

    Robot 1: LCM LCM LCM LCM LCM LCM …
    Robot 2: LCM LCM LCM LCM LCM LCM …
  - Semi-Synchronous:

    Robot 1: LCM LCM LCM LCM LCM …
    Robot 2: LCM LCM LCM LCM …

The Rendezvous Problem

The Rendezvous Problem

- What we want:

The Rendezvous Problem

- What we want:
  - Starting from an arbitrary configuration

The Rendezvous Problem

- What we want:
  - Starting from an arbitrary configuration
  - After a finite number of rounds

The Rendezvous Problem

- What we want:
  - Starting from an arbitrary configuration
  - After a finite number of rounds
  - End up (and remain) at the same position

Related Work

Related Work

- Suzuki-Yamashita (99)

Related Work

- Suzuki-Yamashita (99)
  - Solvable in FSYNC

Related Work

- Suzuki-Yamashita (99)
  - Solvable in FSYNC
  - Solvable in SSYNC with common coordinate system.

Related Work

- Suzuki-Yamashita (99)
  - Solvable in FSYNC
  - Solvable in SSYNC with common coordinate system.
  - **Not** solvable in SSYNC **without** common coordinate system.

What if robots may crash

What if robots may crash

- Two Possibilities:

What if robots may crash

- Two Possibilities:
  - Ignore crashed robots (weak gathering)

What if robots may crash

- Two Possibilities:
  - Ignore crashed robots (weak gathering)
  - Gather at the position of the crashed robot (strong gathering)

Related Work

Related Work

Related Work

- Strong gathering:

Related Work

- Strong gathering:
  - Not possible when $f > 1$

Related Work

- Strong gathering:
  - Not possible when $f > 1$
  - Possible if robots move one at a time (round-robin SSYNC)

Related Work

- Strong gathering:
  - Not possible when $f > 1$
  - Possible if robots move one at a time (round-robin SSYNC)
- Weak gathering (n-robots) tolerating crash faults:

Related Work

- Strong gathering:
  - Not possible when $f > 1$
  - Possible if robots move one at a time (round-robin SSYNC)
- Weak gathering (n-robots) tolerating crash faults:
  - possible for any f in SSYNC with multiplicity detection

The Stand Up Indulgent Rendezvous Problem

- What we want:
  - Starting from an arbitrary configuration
  - After a finite number of rounds
  - End up (and remain) at the same position
  - Even if one robot crashes (strong gathering)

Summary

	Rendezvous	SUIR
SSYNC oblivious, disoriented	Impossible [SY99]	
SSYNC oblivious, common x-axis		
SSYNC oblivious, common x-y-axis	Possible [SY99]	
SSYNC luminous, common x-y-axis	Possible	
FSYNC oblivious, disoriented	Possible [SY99]	

Summary

	Rendezvous	SUIR
SSYNC oblivious, disoriented	Impossible [SY99]	Impossible (our paper)
SSYNC oblivious, common x-axis		Impossible (our paper)
SSYNC oblivious, common x-y-axis	Possible [SY99]	Impossible (our paper)
SSYNC luminous, common x-y-axis	Possible	Impossible (our paper)
FSYNC oblivious, disoriented	Possible [SY99]	

Summary

	Rendezvous	SUIR
SSYNC oblivious, disoriented	Impossible [SY99]	Impossible (our paper)
SSYNC oblivious, common x-axis		Impossible (our paper)
SSYNC oblivious, common x-y-axis	Possible [SY99]	Impossible (our paper)
SSYNC luminous, common x-y-axis	Possible	Impossible (our paper)
FSYNC oblivious, disoriented	Possible [SY99]	Possible (our paper)

Summary

	Rendezvous	SUIR
SSYNC oblivious, disoriented	Impossible [SY99]	Impossible (our paper)
SSYNC oblivious, common x-axis	Possible (our paper)	Impossible (our paper)
SSYNC oblivious, common x-y-axis	Possible [SY99]	Impossible (our paper)
SSYNC luminous, common x-y-axis	Possible	Impossible (our paper)
FSYNC oblivious, disoriented	Possible [SY99]	Possible (our paper)

Impossibility Result

Impossibility Result

Let A be an algorithm that solves the SUIR problem with lights
(infinite memory and communication capabilities)

Impossibility Result

Let A be an algorithm that solves the SUIR problem with lights
(infinite memory and communication capabilities)

Lemma: in an execution, if only one robot r is activated, then there is a round when r is dictated to move to the other robot.

Impossibility Result

Let A be an algorithm that solves the SUIR problem with lights
(infinite memory and communication capabilities)

Lemma: in an execution, if only one robot r is activated, then there is a round when r is dictated to move to the other robot.

Proof: Indeed, if the other robot is crashed, we know that in finite number of rounds r moves to the other robot.

Impossibility Result

Theorem: SUIR is not Solvable in SSYNC (even with lights and common coordinate system)

Impossibility Result

Theorem: SUIR is not Solvable in SSYNC (even with lights and common coordinate system)

Let $\mathcal{A}$ be an algorithm that solves the SUIR problem with lights
(infinite memory and communication capabilities)

Impossibility Result

Theorem: SUIR is not Solvable in SSYNC (even with lights and common coordinate system)

Let A be an algorithm that solves the SUIR problem with lights
(infinite memory and communication capabilities)

Let r and r' be the two robots.

Impossibility Result

Theorem: SUIR is not Solvable in SSYNC (even with lights and common coordinate system)

Let A be an algorithm that solves the SUIR problem with lights
(infinite memory and communication capabilities)

Let r and r' be the two robots.

Consider the following scheduler:

Impossibility Result

Theorem: SUIR is not Solvable in SSYNC (even with lights and common coordinate system)

Let A be an algorithm that solves the SUIR problem with lights (infinite memory and communication capabilities)

Let r and r' be the two robots.

Consider the following scheduler:

- If r is dictated to move to $p \neq r'$, then activate only r

Impossibility Result

Theorem: SUIR is not Solvable in SSYNC (even with lights and common coordinate system)

Let A be an algorithm that solves the SUIR problem with lights (infinite memory and communication capabilities)

Let r and r' be the two robots.

Consider the following scheduler:

- If r is dictated to move to $p \neq r'$, then activate only r
- Otherwise, If r' is dictated to move to $p \neq r$, then activate only r'

Impossibility Result

Theorem: SUIR is not Solvable in SSYNC (even with lights and common coordinate system)

Let A be an algorithm that solves the SUIR problem with lights (infinite memory and communication capabilities)

Let r and r' be the two robots.

Consider the following scheduler:

- If r is dictated to move to $p \neq r'$, then activate only r
- Otherwise, If r' is dictated to move to $p \neq r$, then activate only r'
- Otherwise, activate both robots.

Impossibility Result

Theorem: SUIR is not Solvable in SSYNC (even with lights and common coordinate system)

Let A be an algorithm that solves the SUIR problem with lights (infinite memory and communication capabilities)

Let r and r' be the two robots.

Consider the following scheduler:

- If r is dictated to move to $p \neq r'$, then activate only r
- Otherwise, If r' is dictated to move to $p \neq r$, then activate only r'
- Otherwise, activate both robots.

The scheduler is fair otherwise there exists an execution where only r (or r') is activated, without moving to the other robot, contradicting the previous Lemma.

Impossibility Result

Theorem: SUIR is not Solvable in SSYNC (even with lights and common coordinate system)

Let A be an algorithm that solves the SUIR problem with lights (infinite memory and communication capabilities)

Let r and r' be the two robots.

Consider the following scheduler:

- If r is dictated to move to $p \neq r'$, then activate only r
- Otherwise, If r' is dictated to move to $p \neq r$, then activate only r'
- Otherwise, activate both robots.

The scheduler is fair otherwise there exists an execution where only r (or r') is activated, without moving to the other robot, contradicting the previous Lemma.

After each round the robots are not gathered.

Solving The SUIR in FSYNC

Solving The SUIR in FSYNC

How to define an algorithm:

Solving The SUIR in FSYNC

How to define an algorithm:

A rule defines the target for a set of views

Solving The SUIR in FSYNC

How to define an algorithm:

A rule defines the target for a set of views

Examples:

Solving The SUIR in FSYNC

How to define an algorithm:

A rule defines the target for a set of views

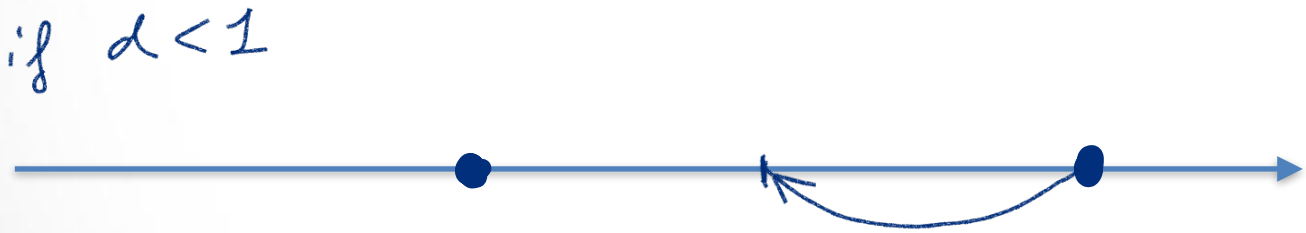
Examples:

Solving The SUIR in FSYNC

How to define an algorithm:

A rule defines the target for a set of views

Examples:

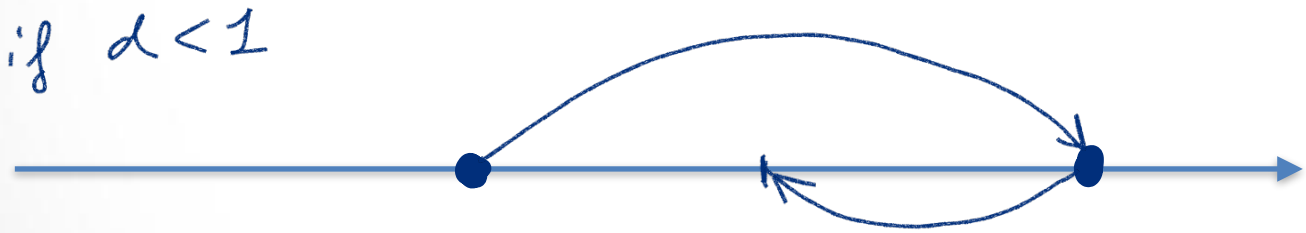


Solving The SUIR in FSYNC

How to define an algorithm:

A rule defines the target for a set of views

Examples:

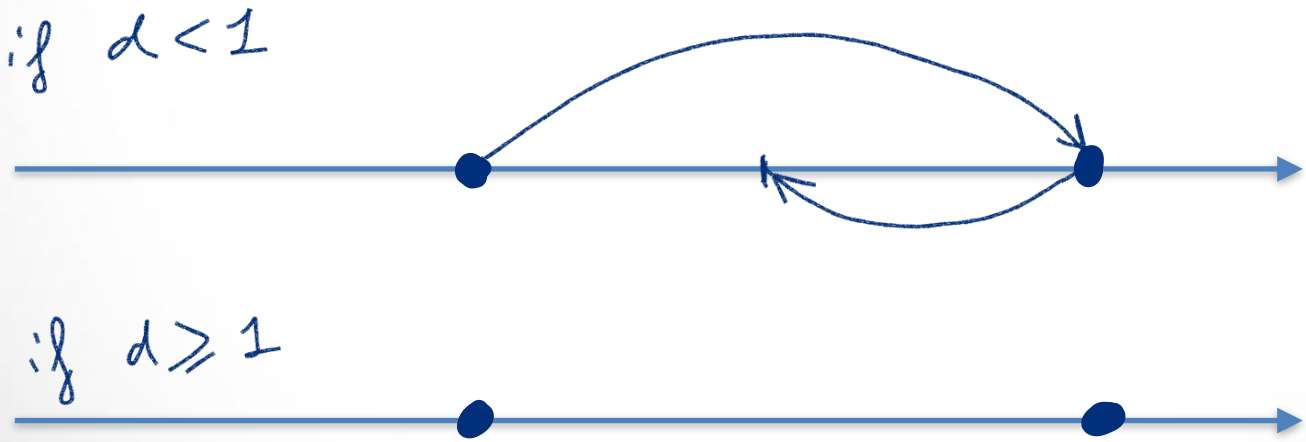


Solving The SUIR in FSYNC

How to define an algorithm:

A rule defines the target for a set of views

Examples:

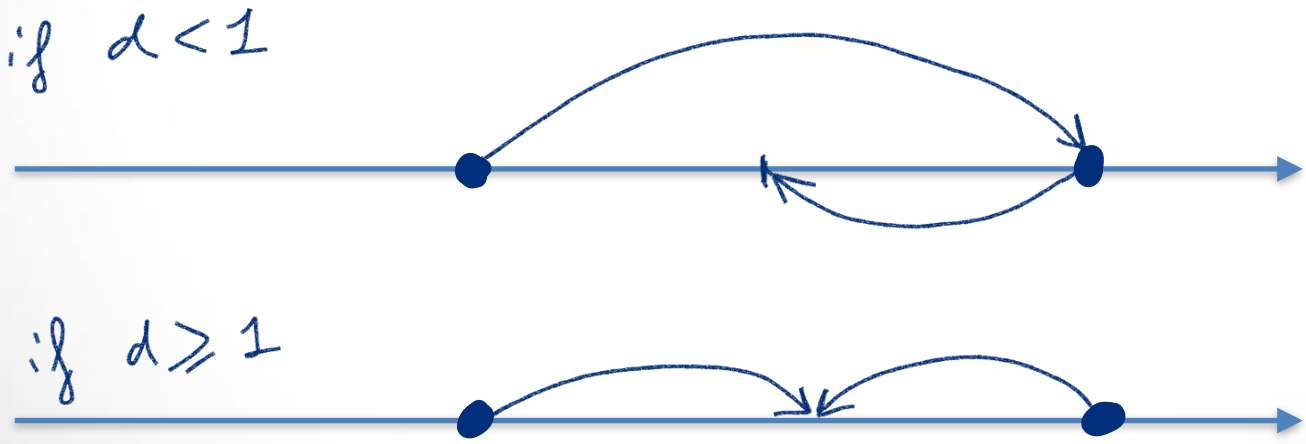


Solving The SUIR in FSYNC

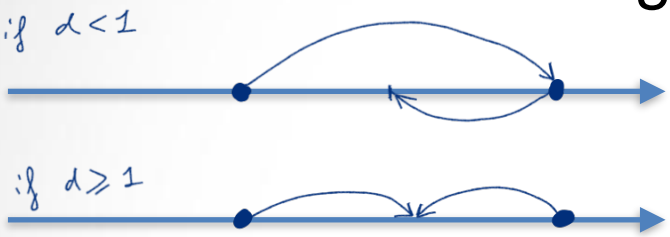
How to define an algorithm:

A rule defines the target for a set of views

Examples:

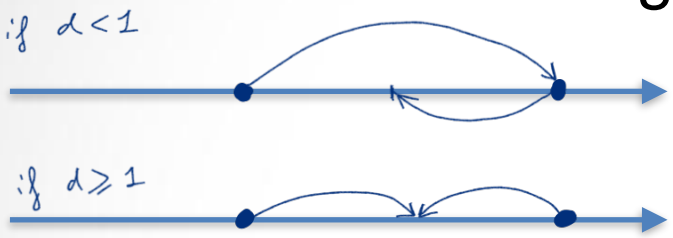


Solving The SUIR in FSYNC



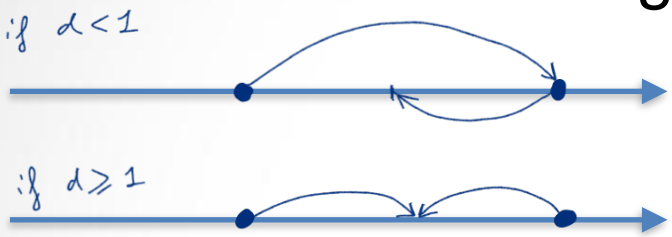
Example execution:

Solving The SUIR in FSYNC

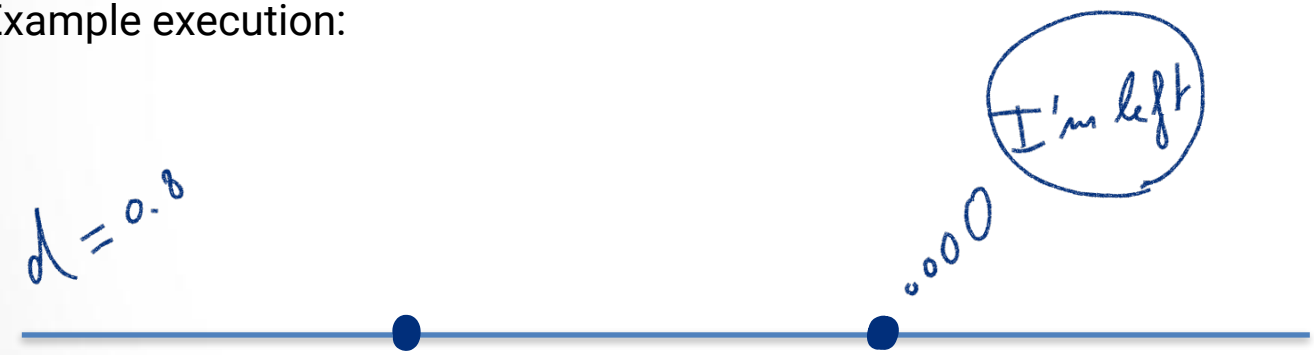


Example execution:

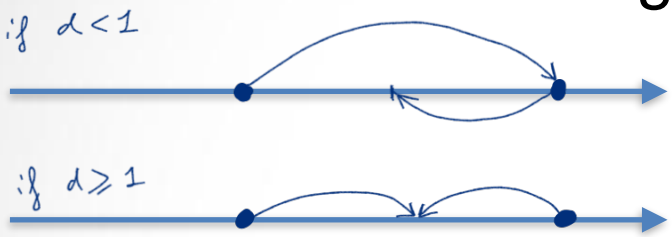
Solving The SUIR in FSYNC



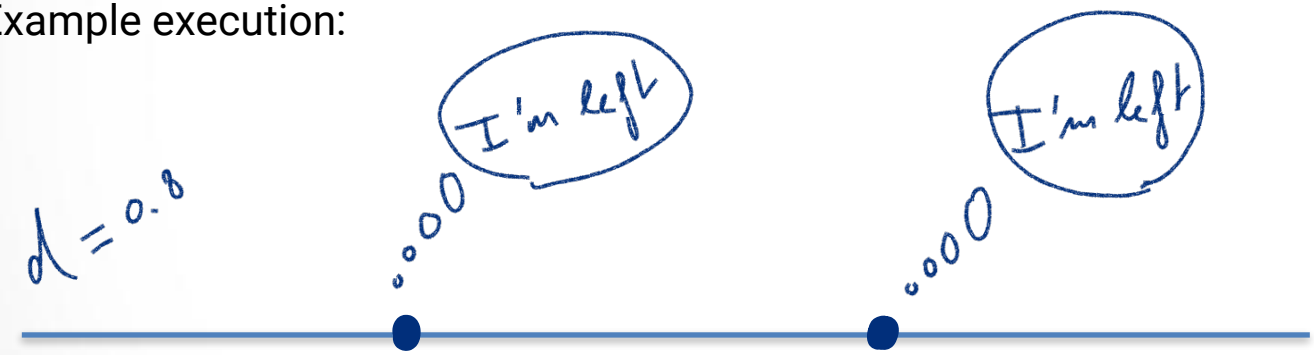
Example execution:



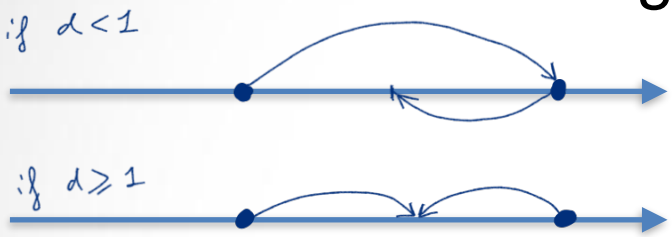
Solving The SUIR in FSYNC



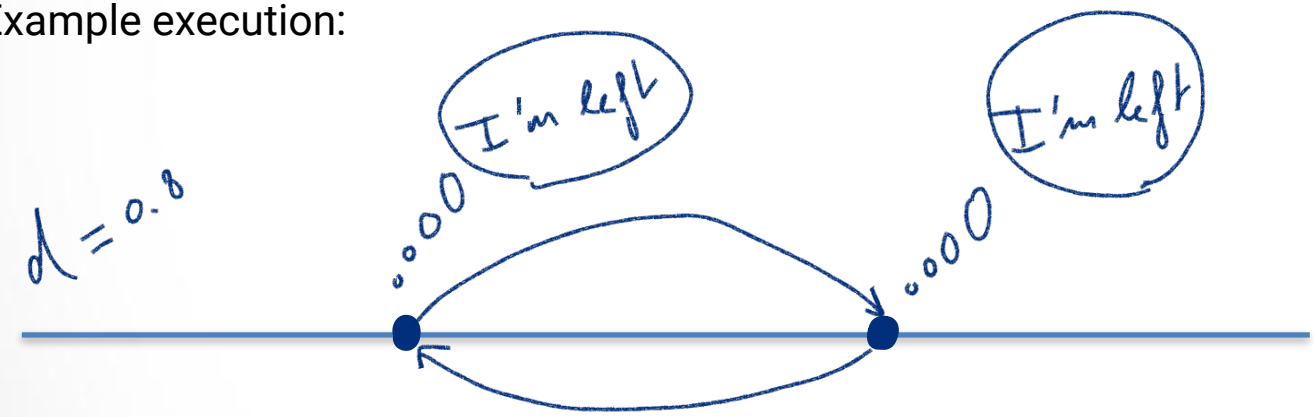
Example execution:



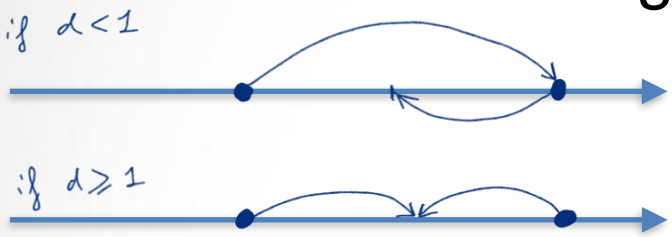
Solving The SUIR in FSYNC



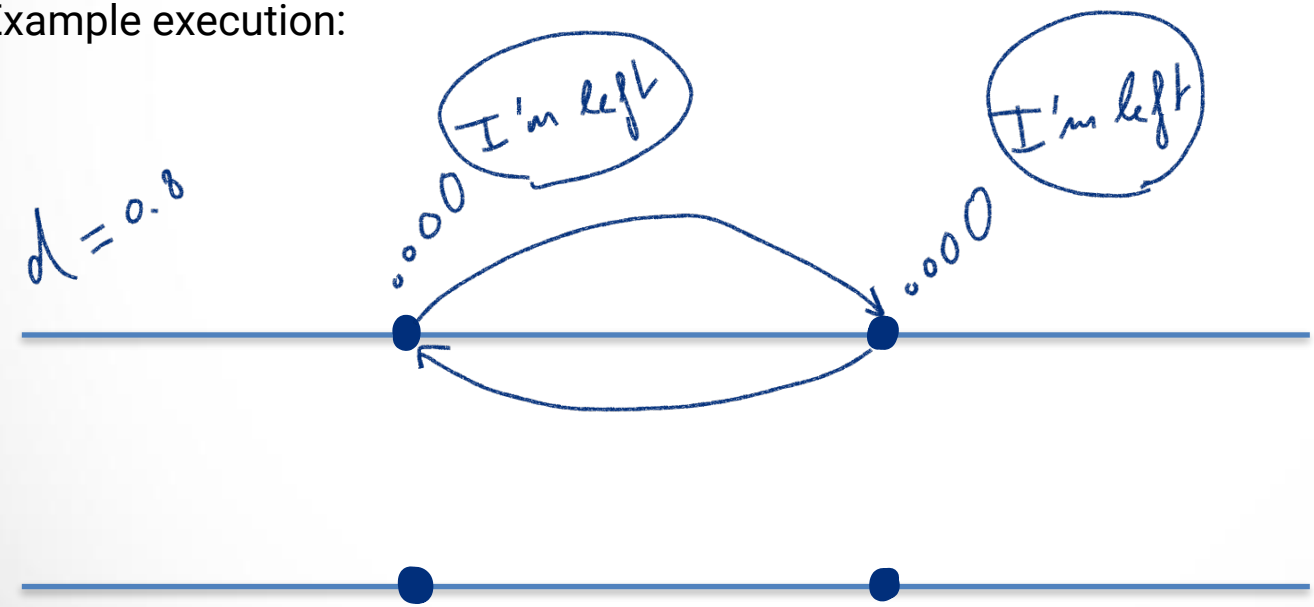
Example execution:



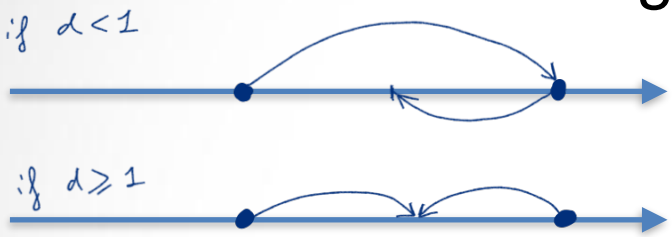
Solving The SUIR in FSYNC



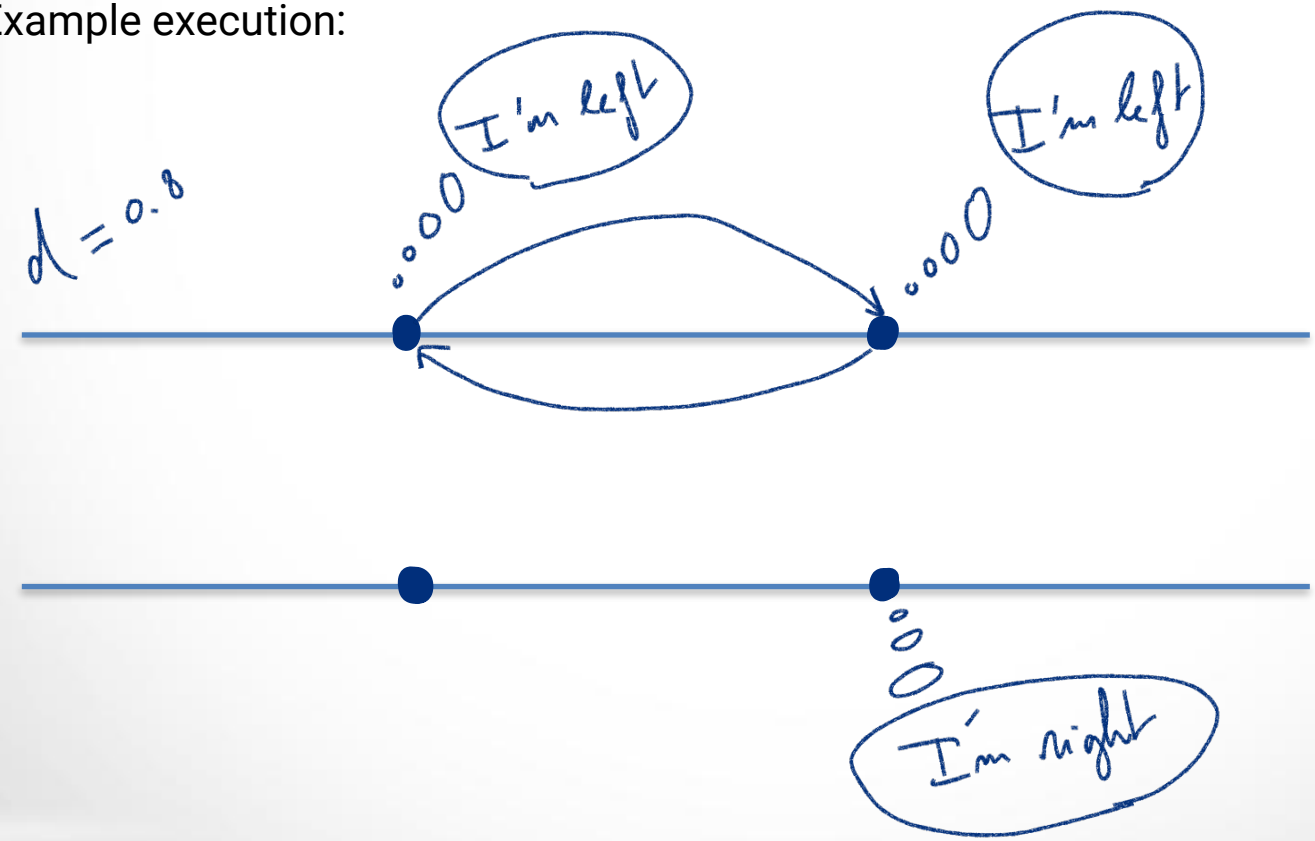
Example execution:



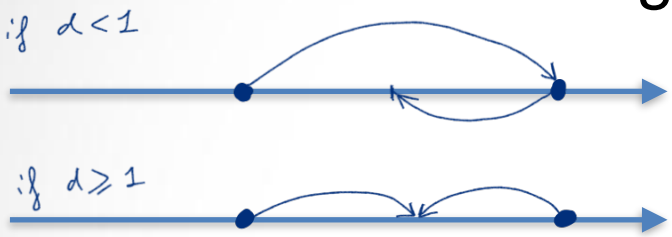
Solving The SUIR in FSYNC



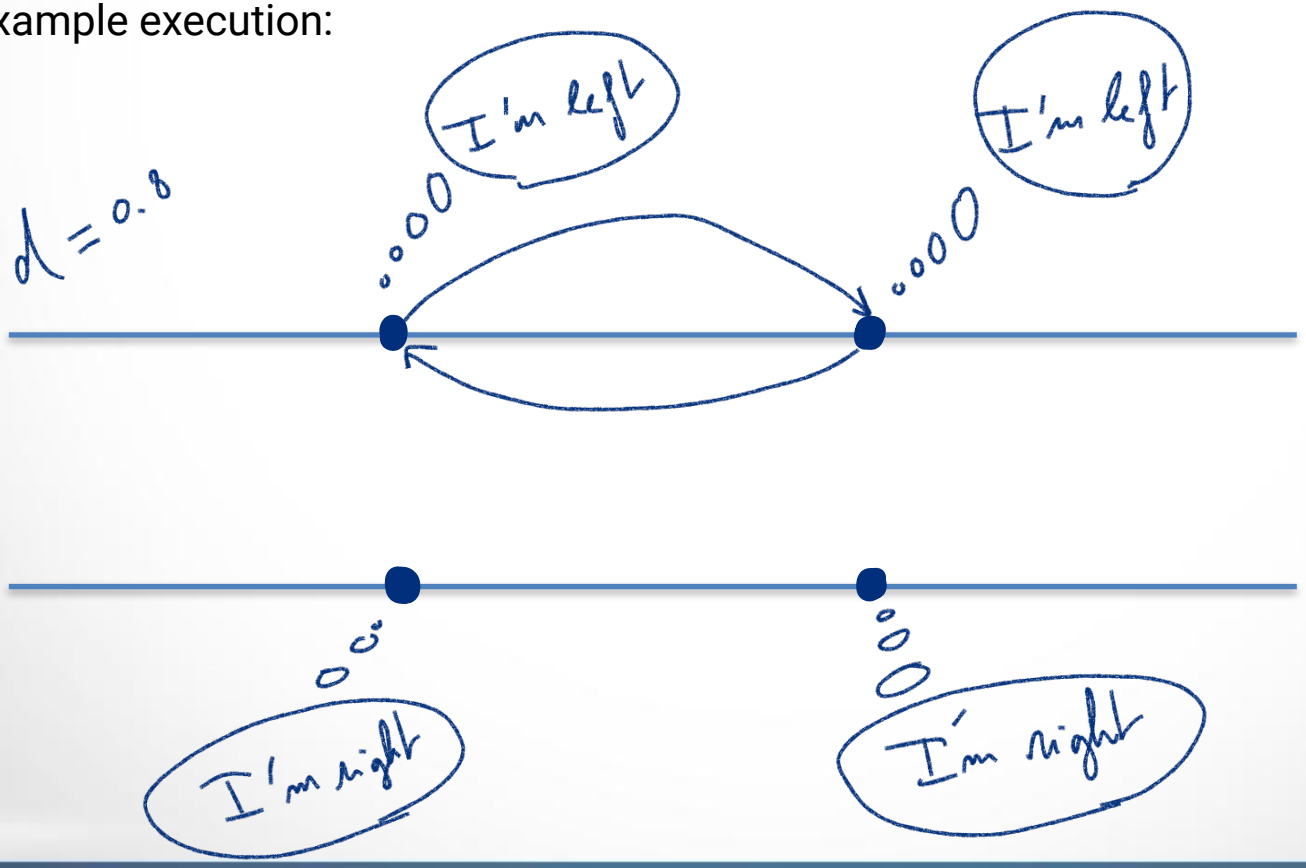
Example execution:



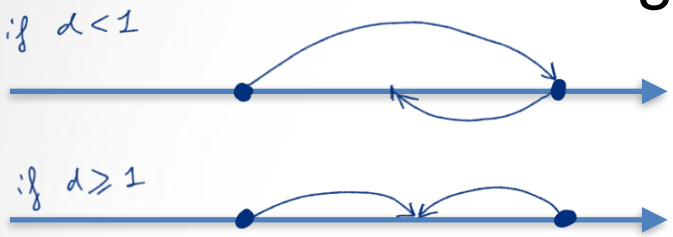
Solving The SUIR in FSYNC



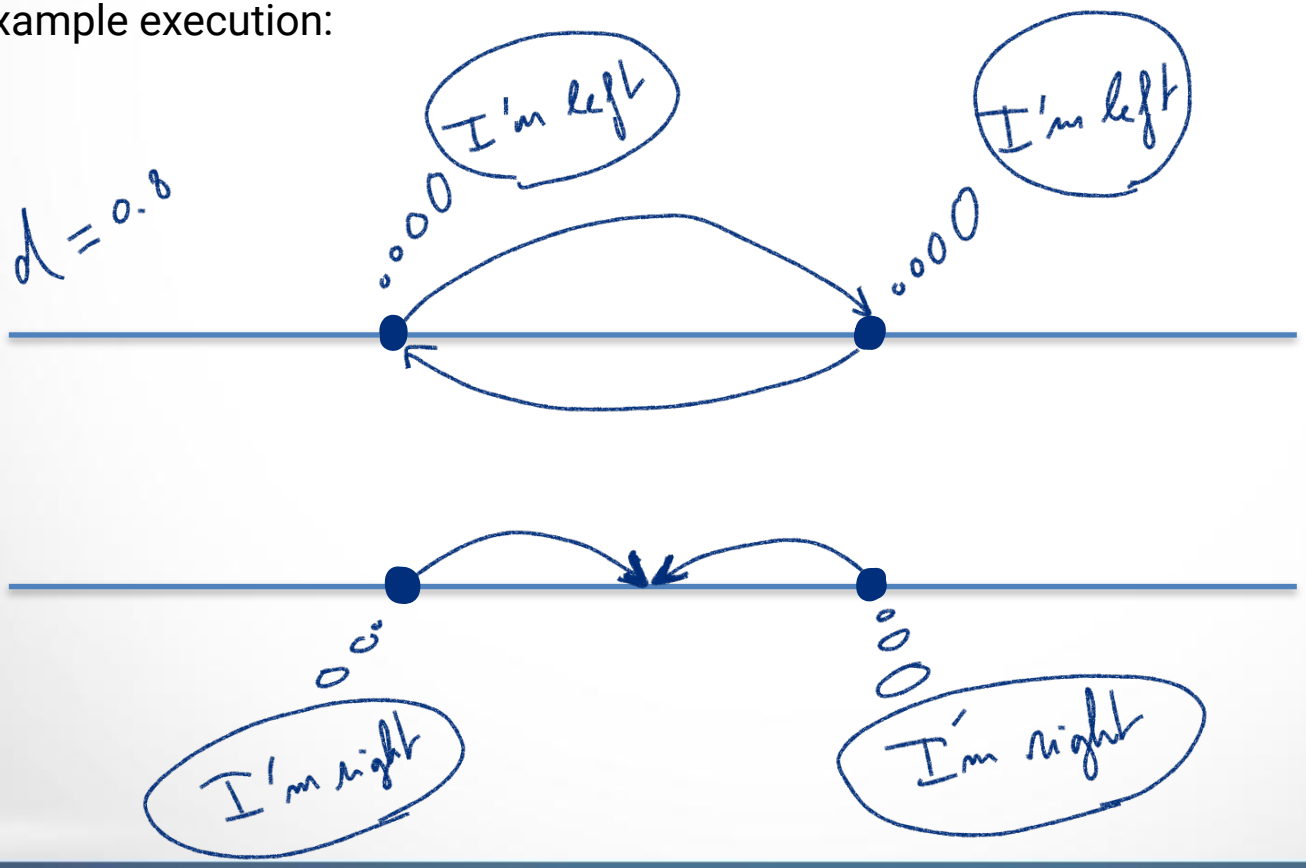
Example execution:



Solving The SUIR in FSYNC



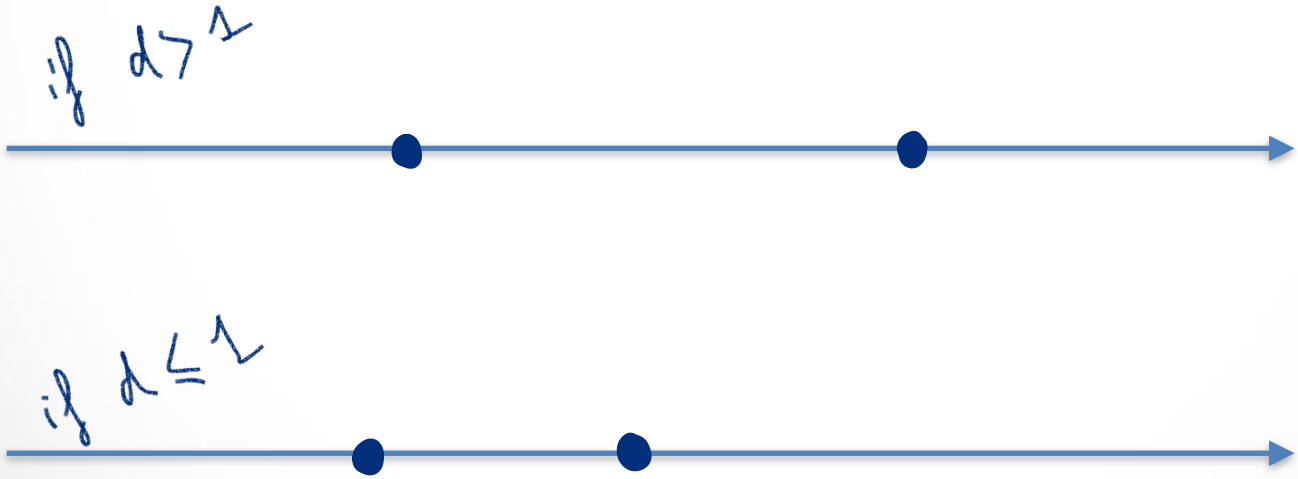
Example execution:



Solving The SUIR in FSYNC

Assume for now that robots :

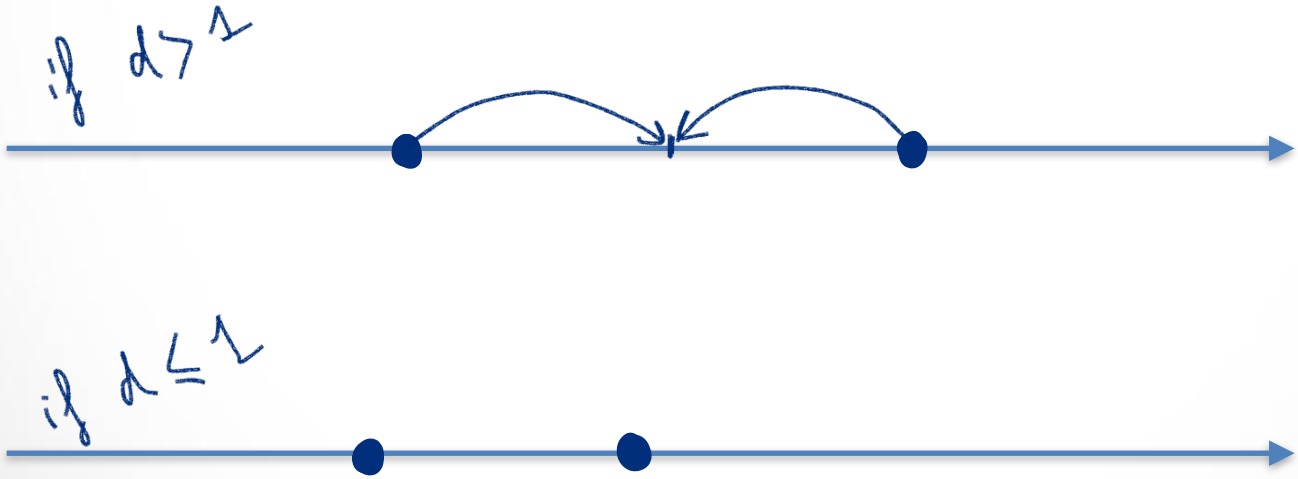
- Are on an oriented line
- Have a common unit distance
- Have rigid movements



Solving The SUIR in FSYNC

Assume for now that robots :

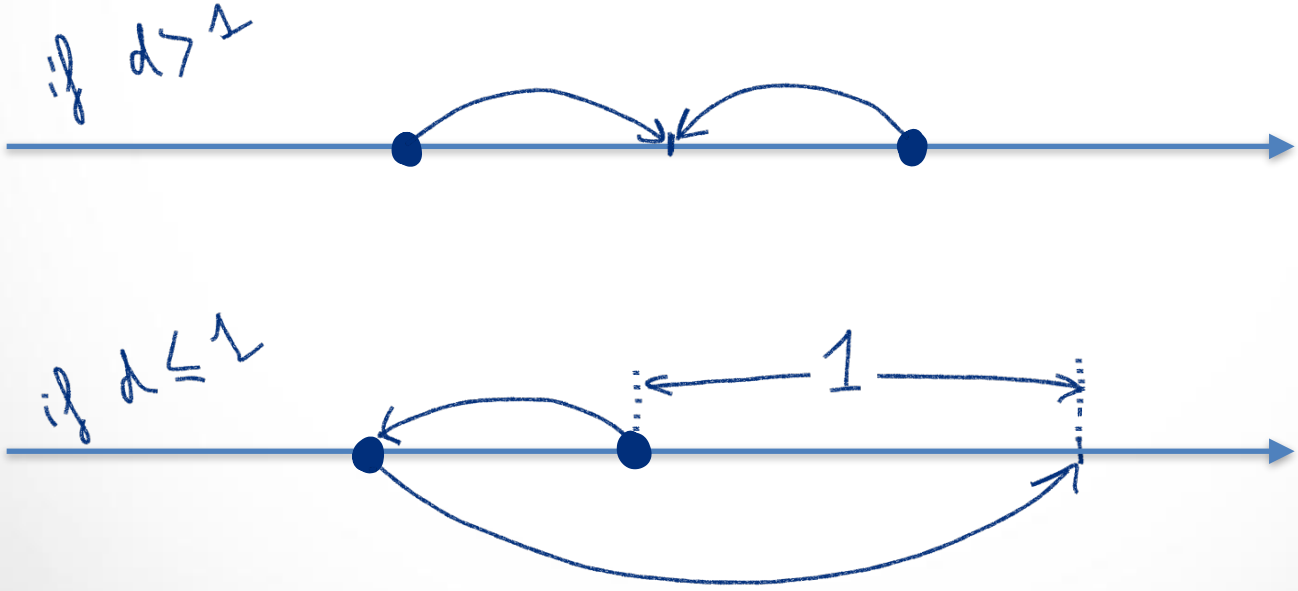
- Are on an oriented line
- Have a common unit distance
- Have rigid movements



Solving The SUIR in FSYNC

Assume for now that robots :

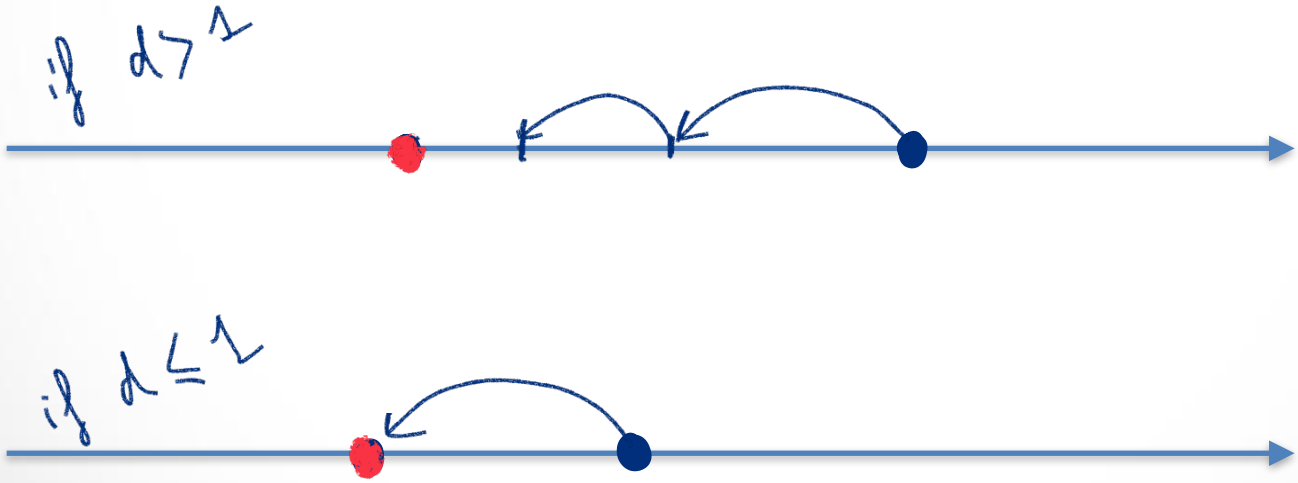
- Are on an oriented line
- Have a common unit distance
- Have rigid movements



Solving The SUIR in FSYNC

Assume for now that robots :

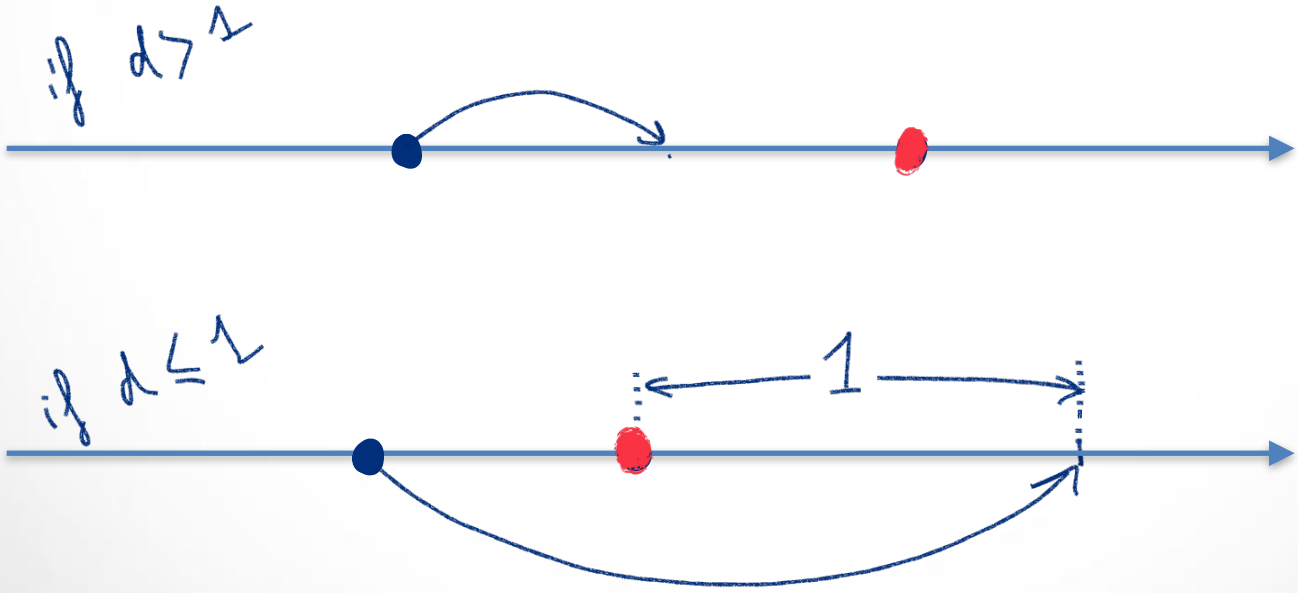
- Are on an oriented line
- Have a common unit distance
- Have rigid movements



Solving The SUIR in FSYNC

Assume for now that robots :

- Are on an oriented line
- Have a common unit distance
- Have rigid movements



Solving The SUIR in FSYNC

Assume for now that robots :

- Are on an oriented line
- Have a common unit distance
- ~~Have rigid movements~~

Solving The SUIR in FSYNC

Assume for now that robots :

- Are on an oriented line
- Have a common unit distance
- ~~Have rigid movements~~

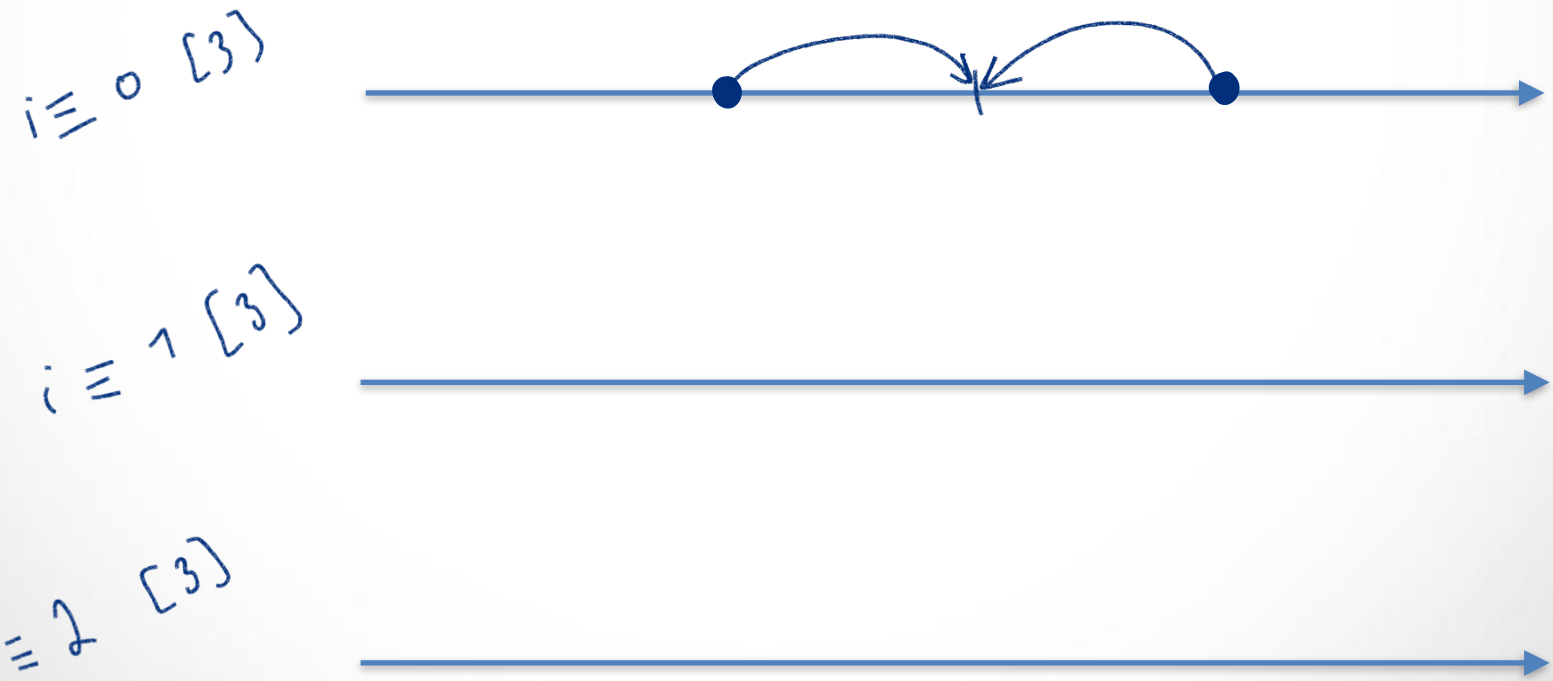
$d \in [2^{-i}, 2^{1-i})$

Solving The SUIR in FSYNC

Assume for now that robots :

- Are on an oriented line
- Have a common unit distance
- ~~Have rigid movements~~

$d \in [2^{-i}, 2^{1-i})$

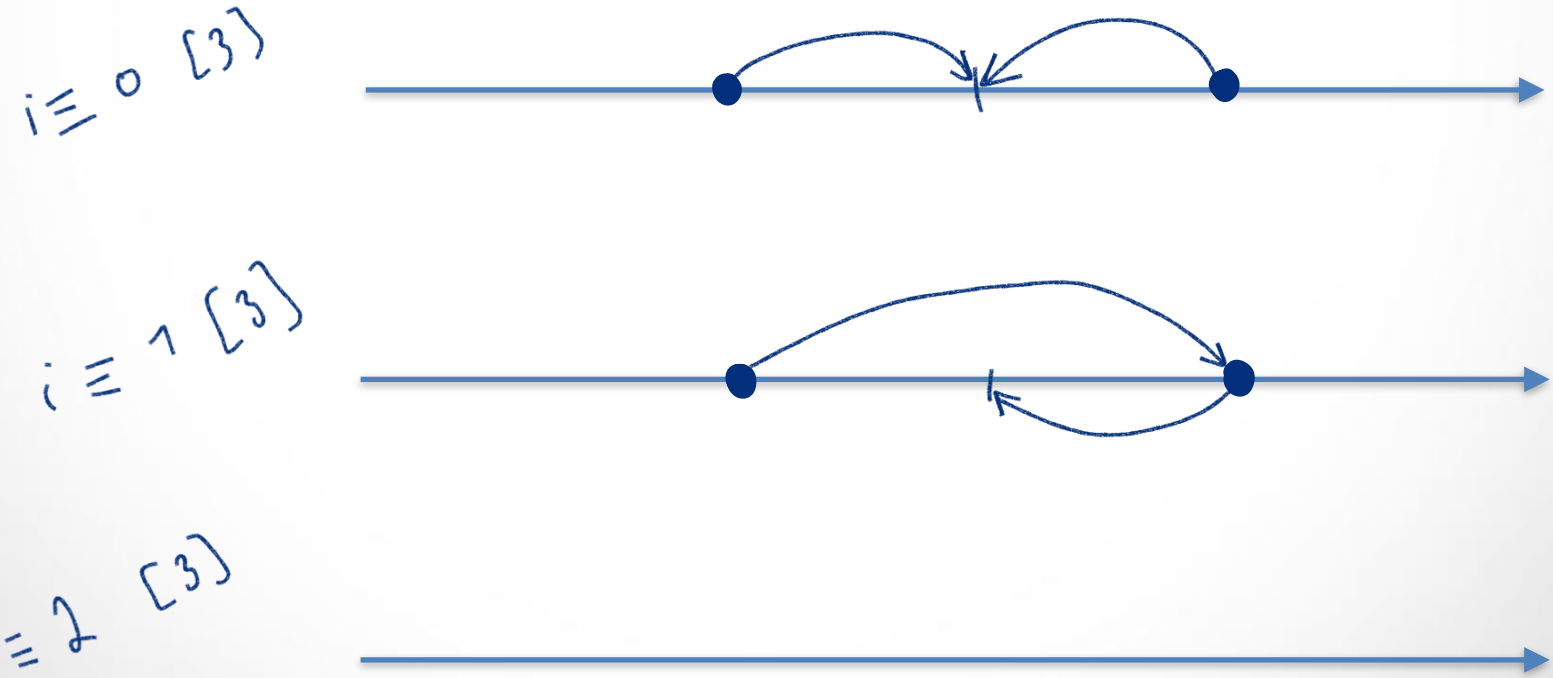


Solving The SUIR in FSYNC

Assume for now that robots :

- Are on an oriented line
- Have a common unit distance
- ~~Have rigid movements~~

$d \in [2^{-i}, 2^{1-i})$



Solving The SUIR in FSYNC

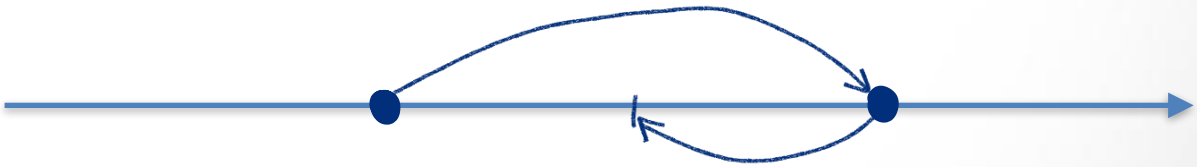
Assume for now that robots :

- Are on an oriented line
- Have a common unit distance
- ~~Have rigid movements~~

$d \in [2^{-i}, 2^{1-i})$

$i \equiv 0 \ [3]$

$i \equiv 1 \ [3]$

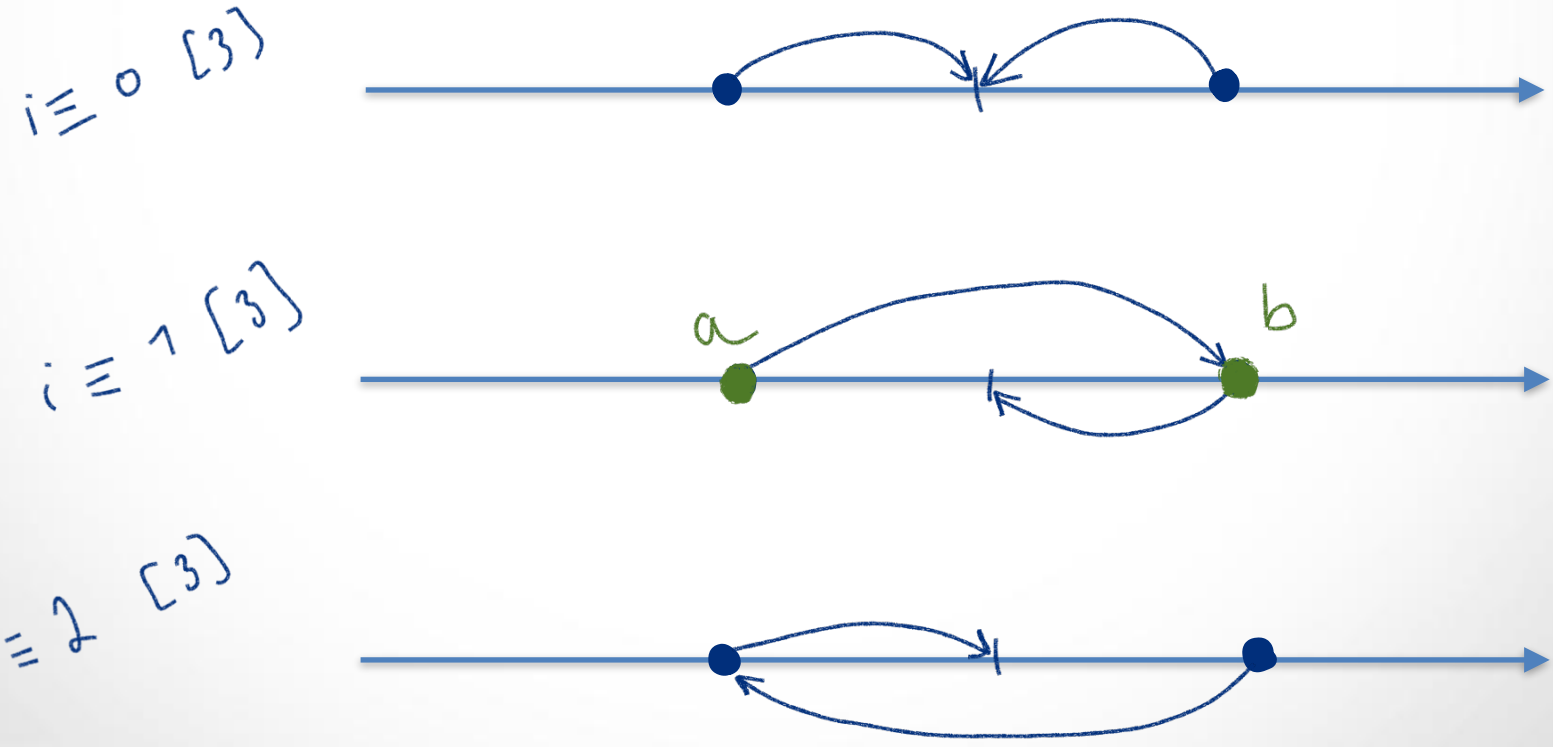


$i \equiv 2 \ [3]$

Solving The SUIR in FSYNC

Assume for now that robots :

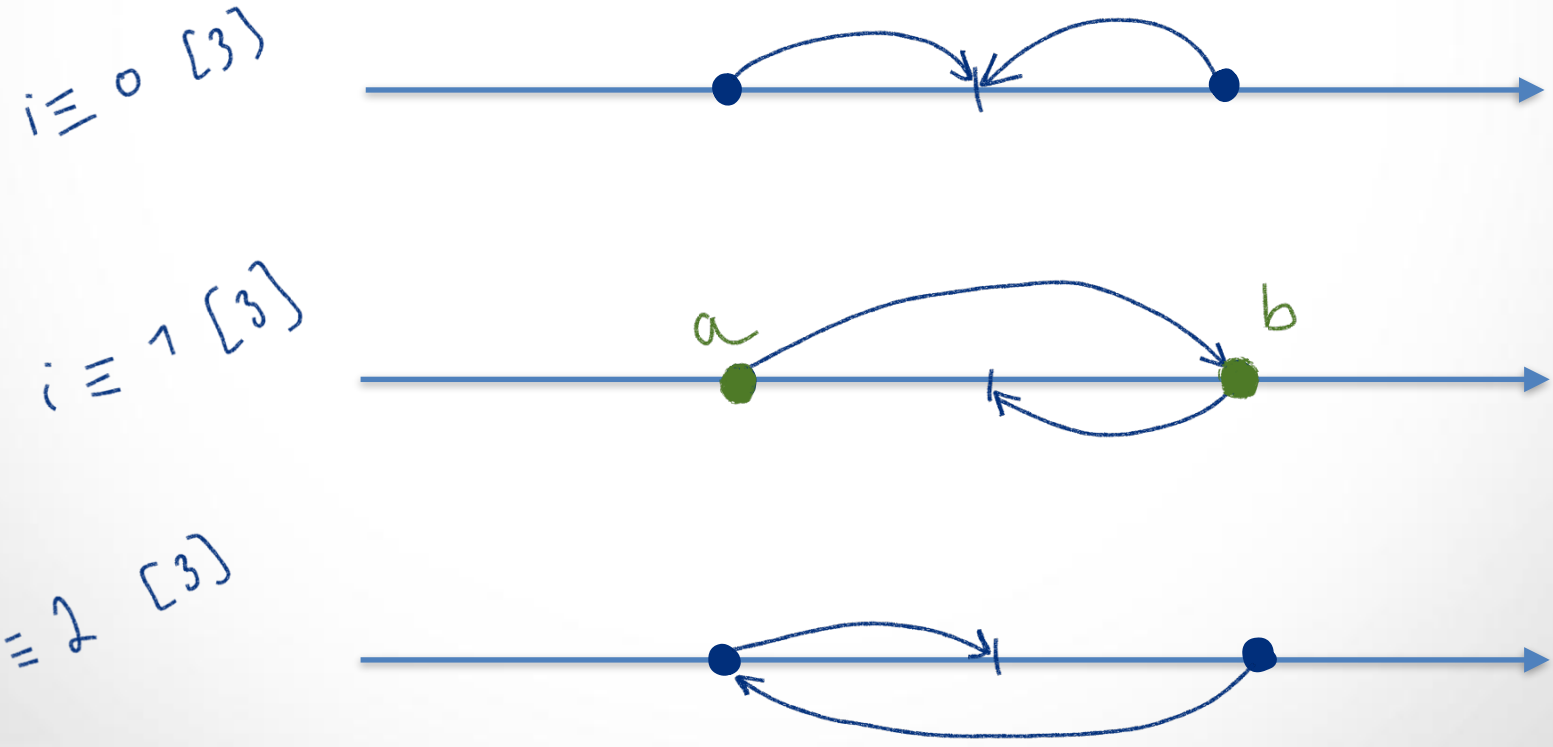
- Are on an oriented line
- Have a common unit distance
- ~~Have rigid movements~~



Solving The SUIR in FSYNC

Assume for now that robots :

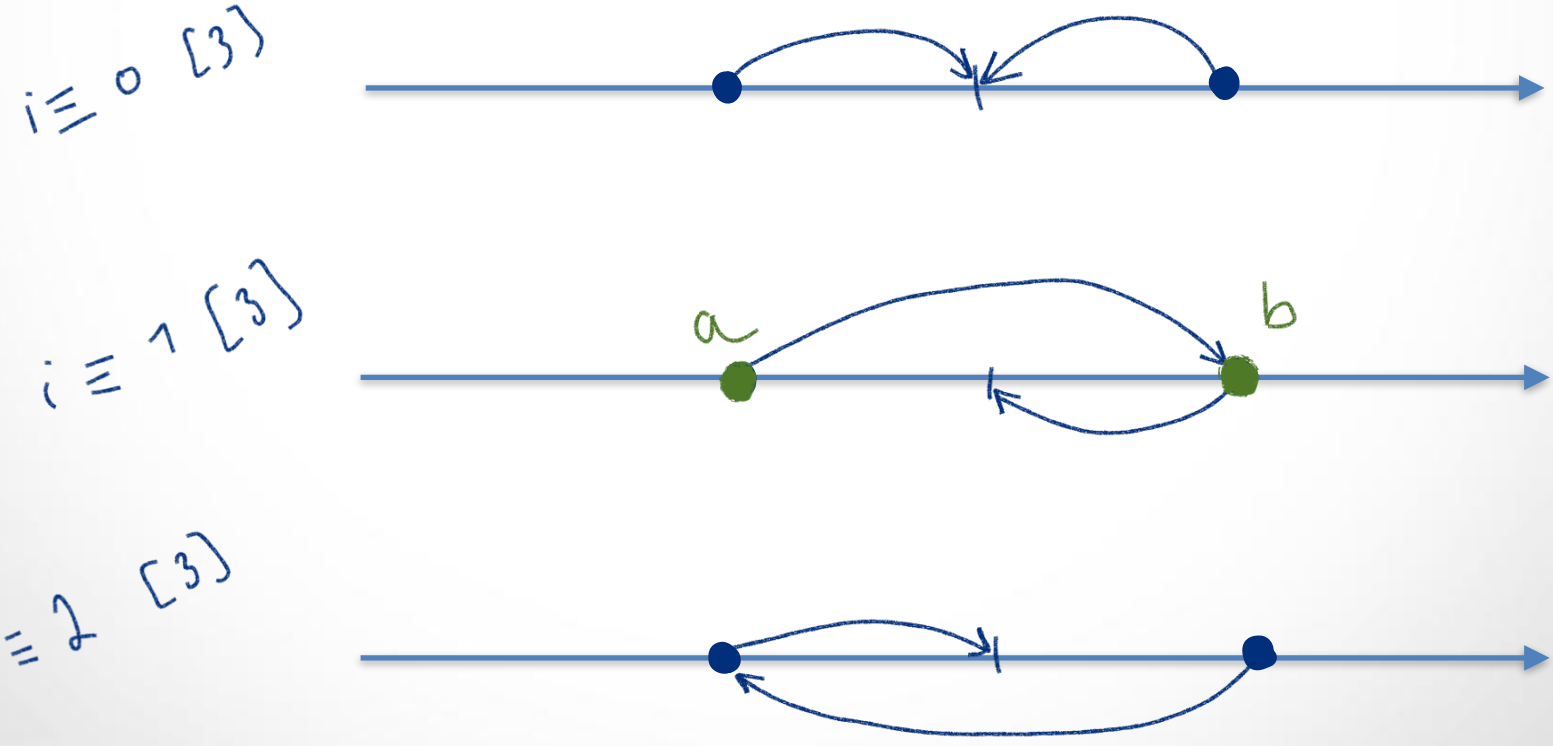
- Are on an oriented line
- Have a common unit distance
- ~~Have rigid movements~~



Solving The SUIR in FSYNC

Assume for now that robots :

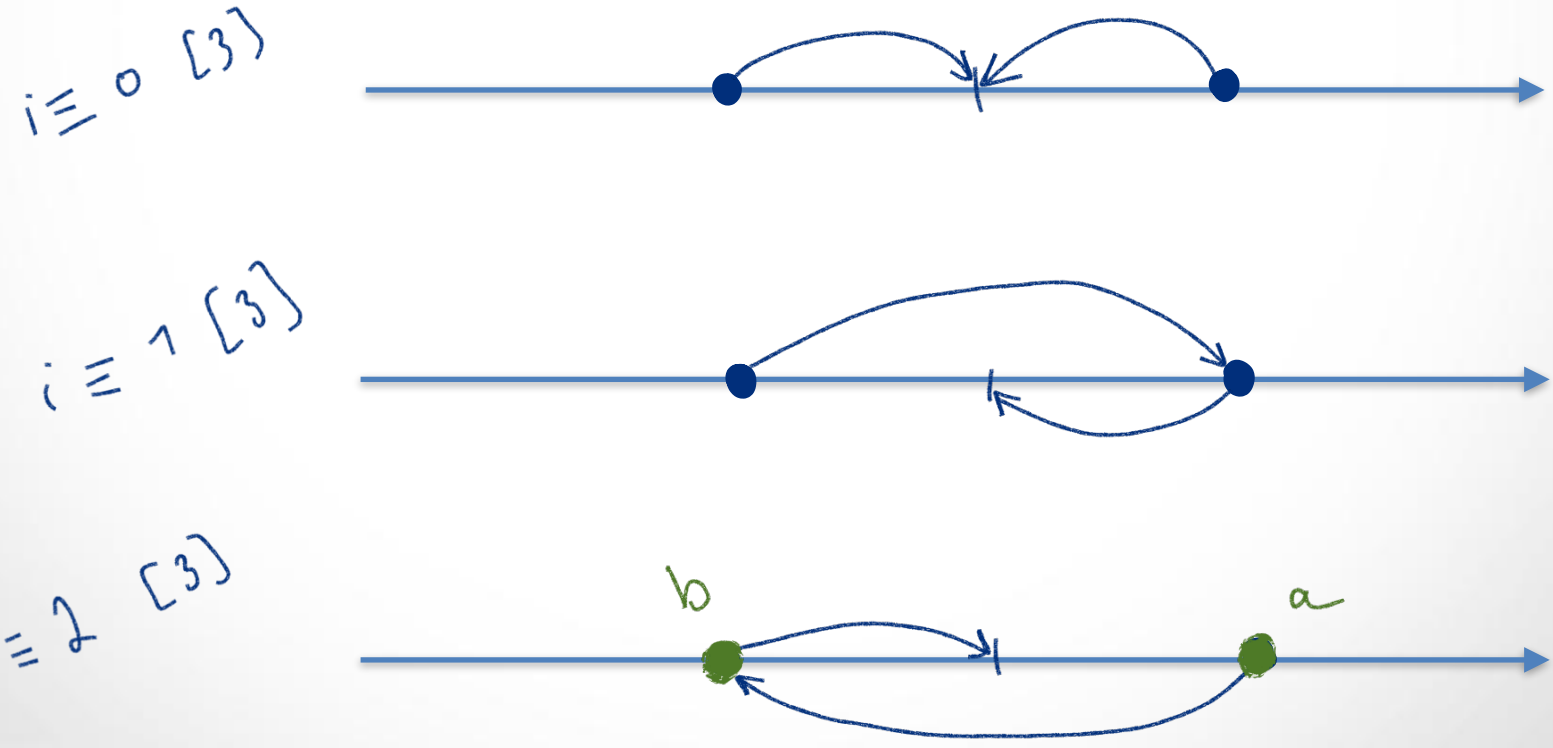
- Are on an oriented line
- Have a common unit distance
- ~~Have rigid movements~~



Solving The SUIR in FSYNC

Assume for now that robots :

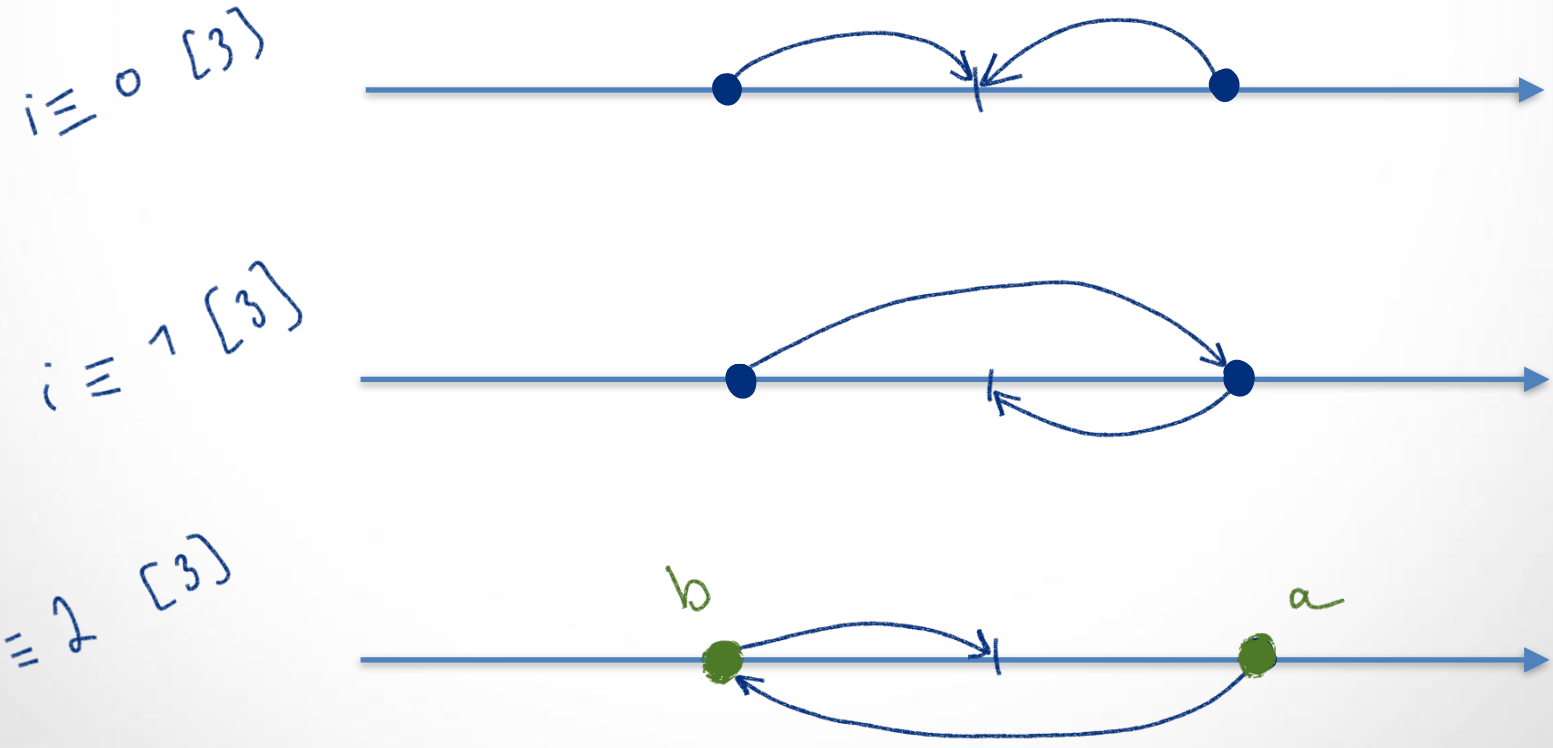
- Are on an oriented line
- Have a common unit distance
- ~~Have rigid movements~~



Solving The SUIR in FSYNC

Assume for now that robots :

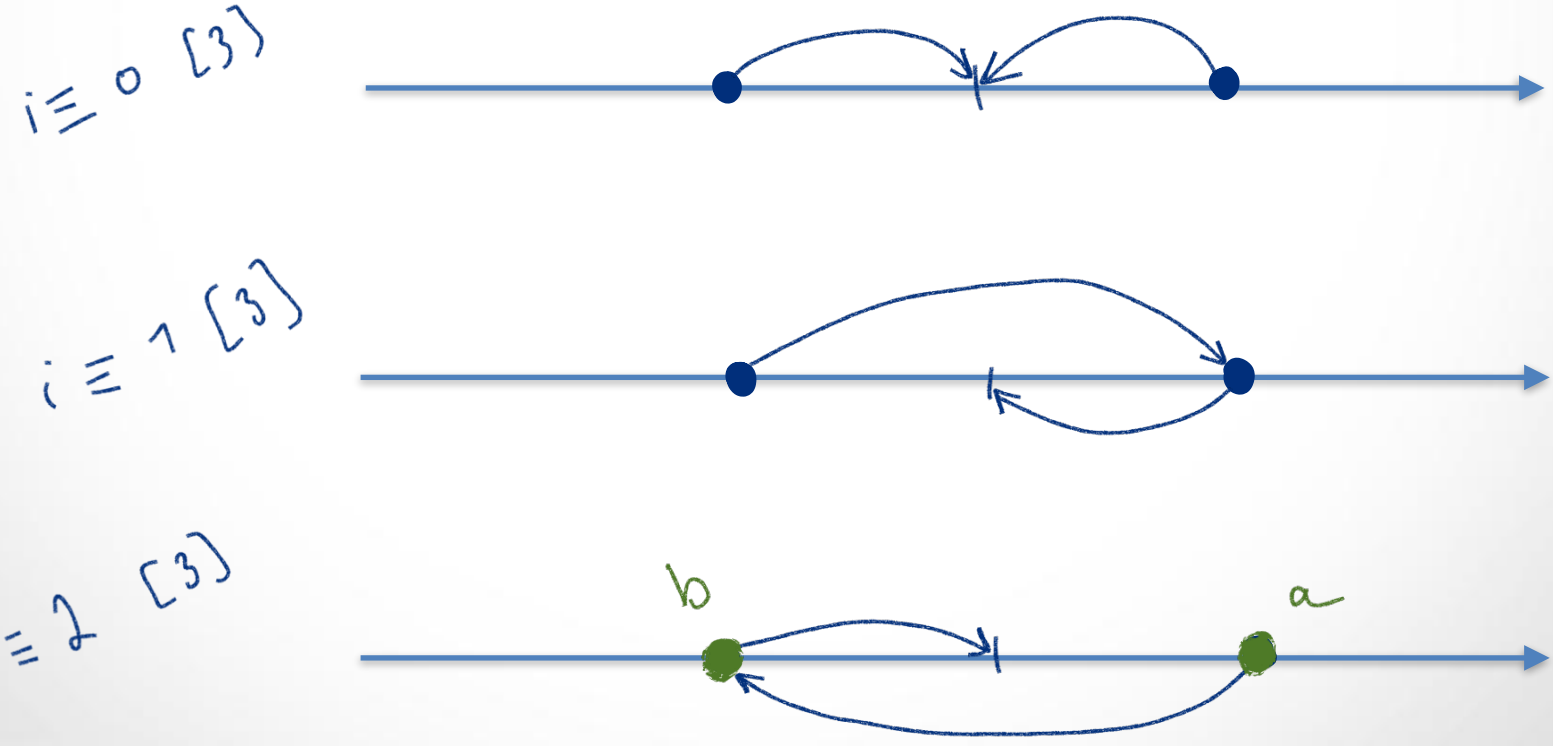
- Are on an oriented line
- Have a common unit distance
- ~~Have rigid movements~~



Solving The SUIR in FSYNC

Assume for now that robots :

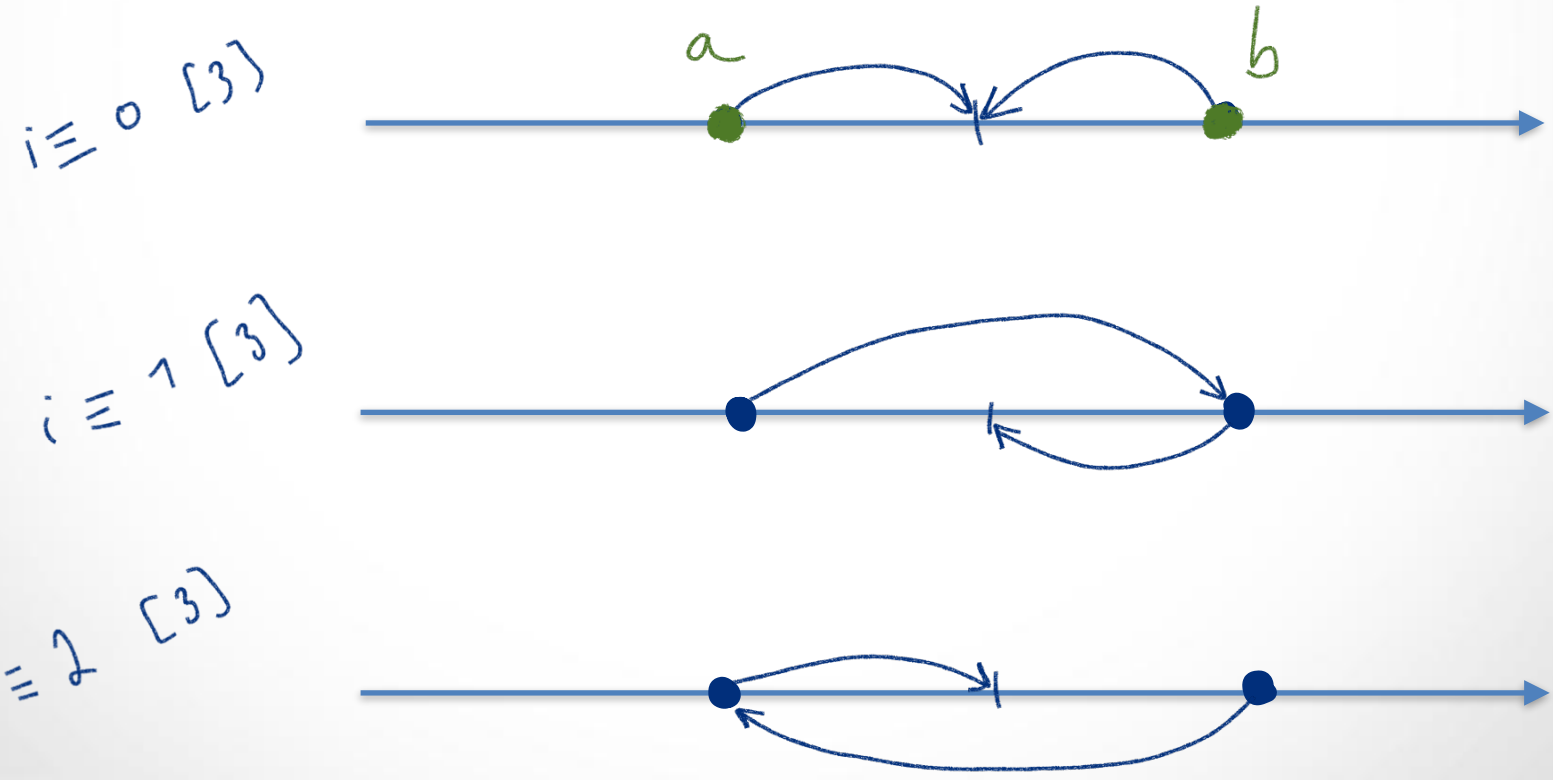
- Are on an oriented line
- Have a common unit distance
- ~~Have rigid movements~~



Solving The SUIR in FSYNC

Assume for now that robots :

- Are on an oriented line
- Have a common unit distance
- ~~Have rigid movements~~



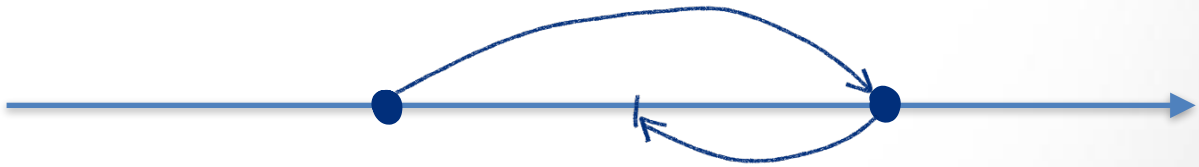
Solving The SUIR in FSYNC

Assume for now that robots :

- Are on an oriented line
- Have a common unit distance
- ~~Have rigid movements~~

$i \equiv 0 \ [3]$

$i \equiv 1 \ [3]$



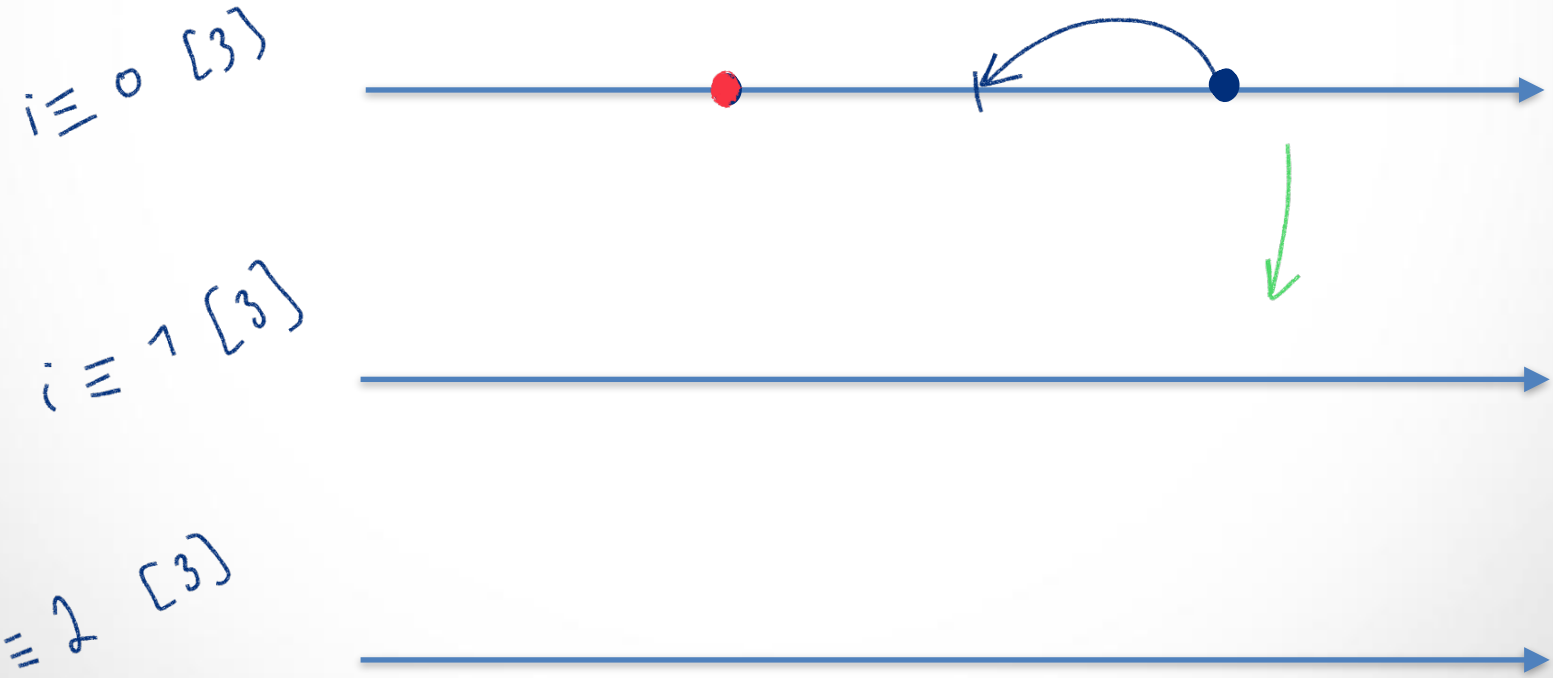
$i \equiv 2 \ [3]$

Solving The SUIR in FSYNC

Assume for now that robots :

- Are on an oriented line
- Have a common unit distance
- ~~Have rigid movements~~

$d \in [2^{-i}, 2^{1-i})$

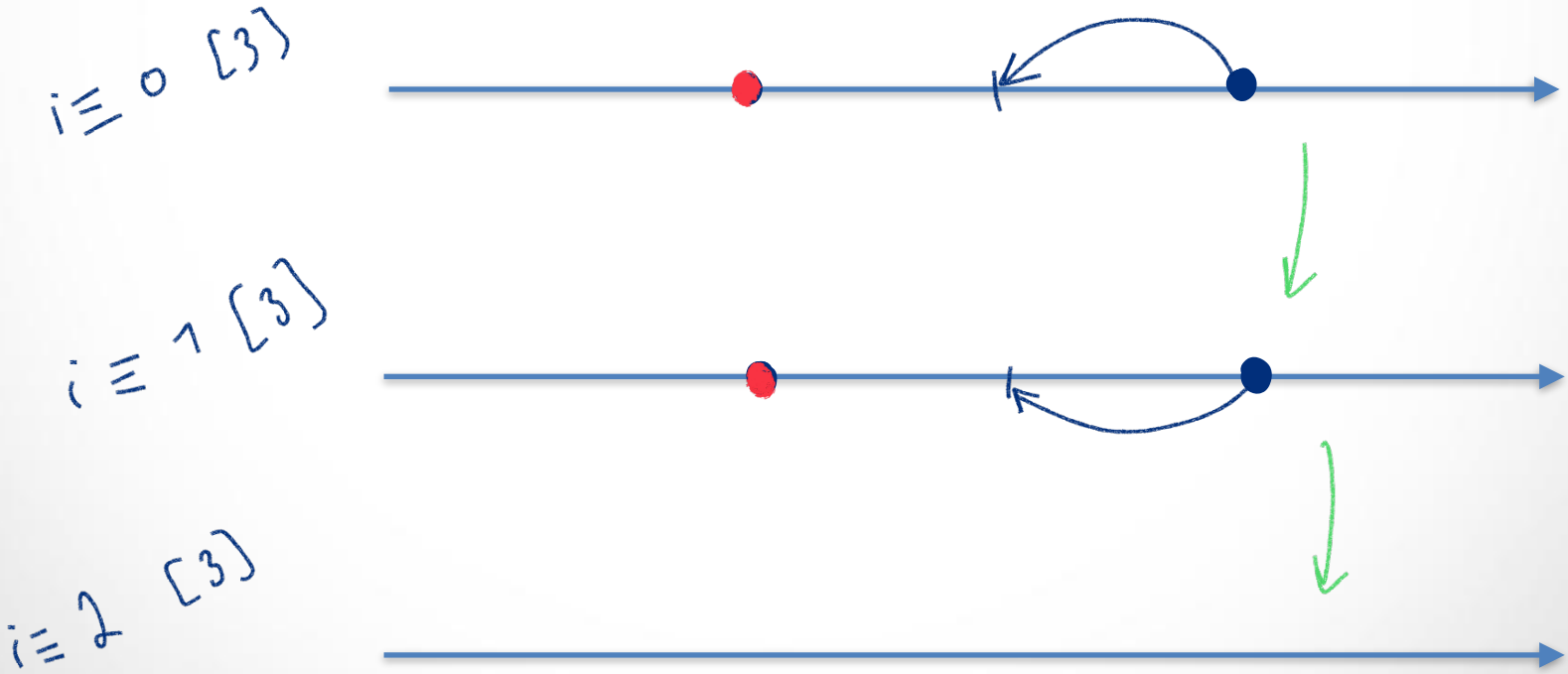


Solving The SUIR in FSYNC

Assume for now that robots :

- Are on an oriented line
- Have a common unit distance
- ~~Have rigid movements~~

$d \in [2^{-i}, 2^{1-i})$

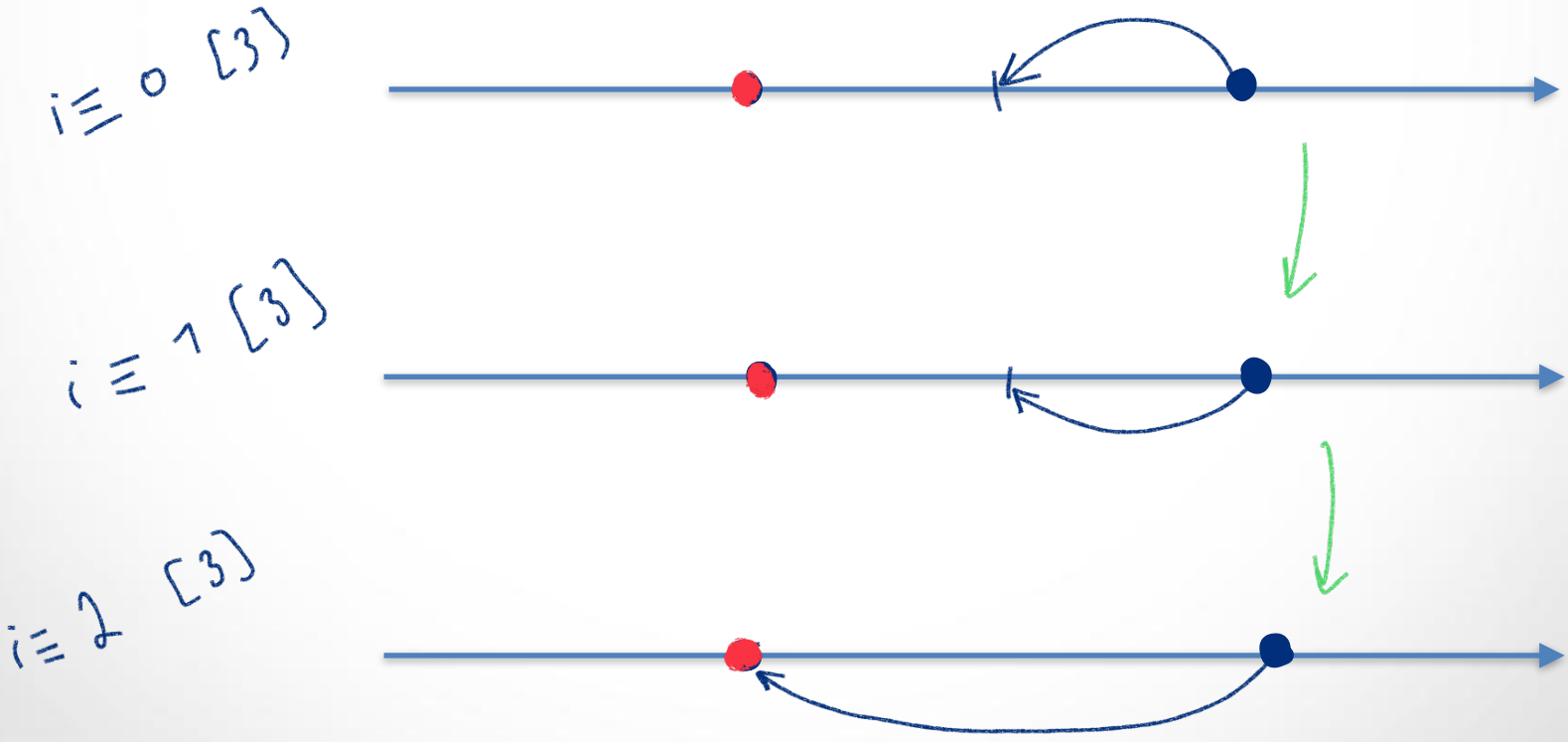


Solving The SUIR in FSYNC

Assume for now that robots :

- Are on an oriented line
- Have a common unit distance
- ~~Have rigid movements~~

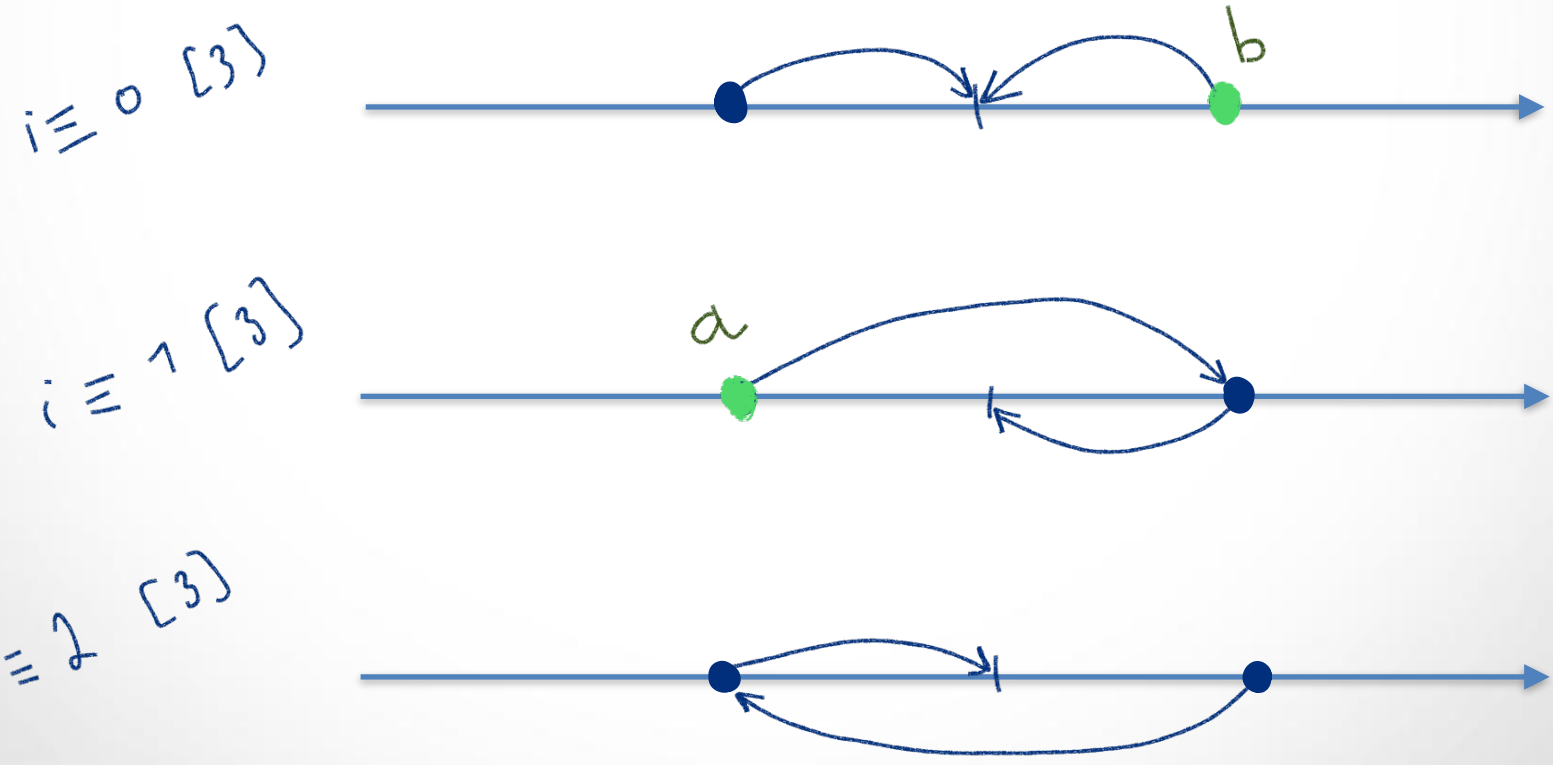
$d \in [2^{-i}, 2^{1-i})$



Solving The SUIR in FSYNC

Assume for now that robots :

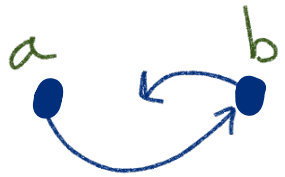
- Are on an oriented line
- ~~Have a common unit distance~~
- ~~Have rigid movements~~



Solving The SUIR in FSYNC

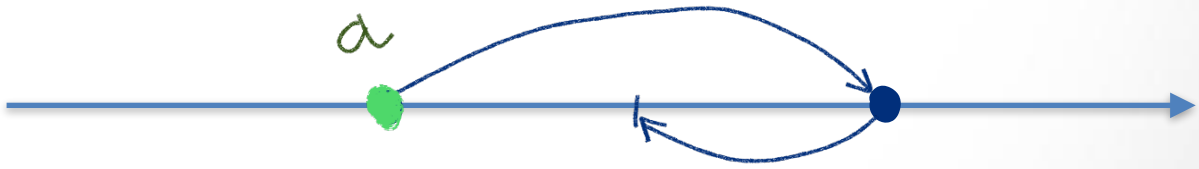
Assume for now that robots :

- Are on an oriented line
- ~~Have a common unit distance~~
- ~~Have rigid movements~~



$i \equiv 0 \ [3]$

$i \equiv 1 \ [3]$

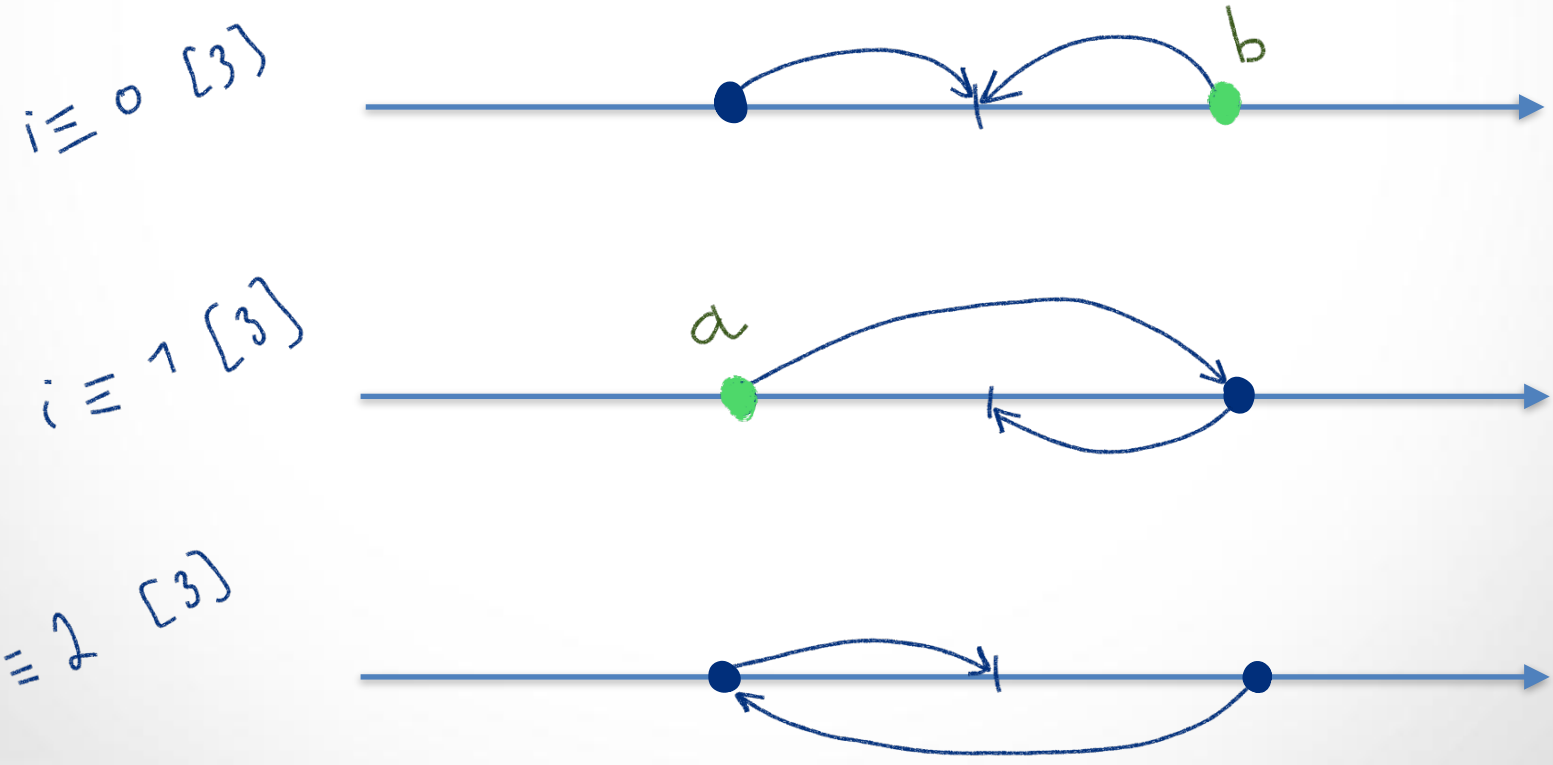


$i \equiv 2 \ [3]$

Solving The SUIR in FSYNC

Assume for now that robots :

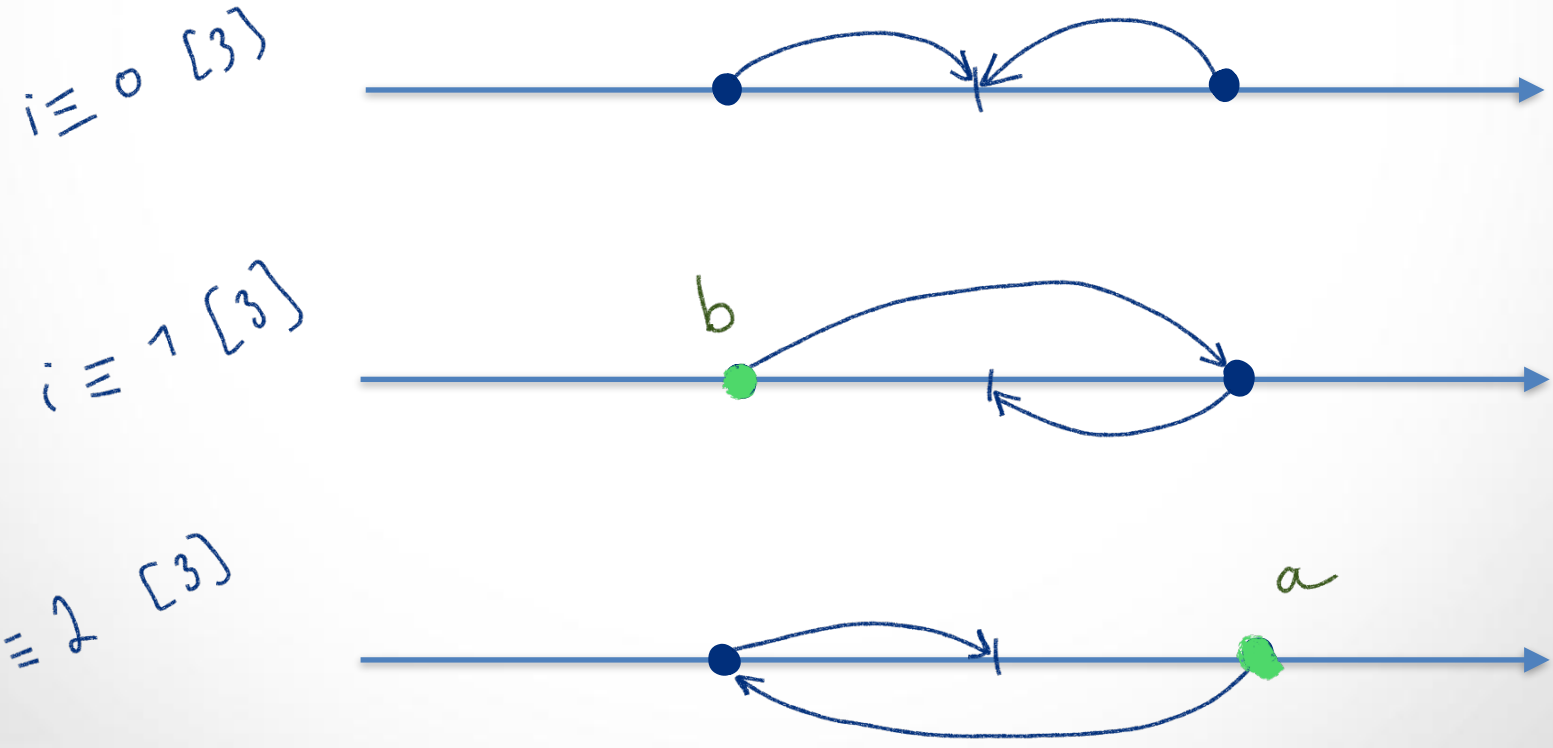
- Are on an oriented line
- ~~Have a common unit distance~~
- ~~Have rigid movements~~



Solving The SUIR in FSYNC

Assume for now that robots :

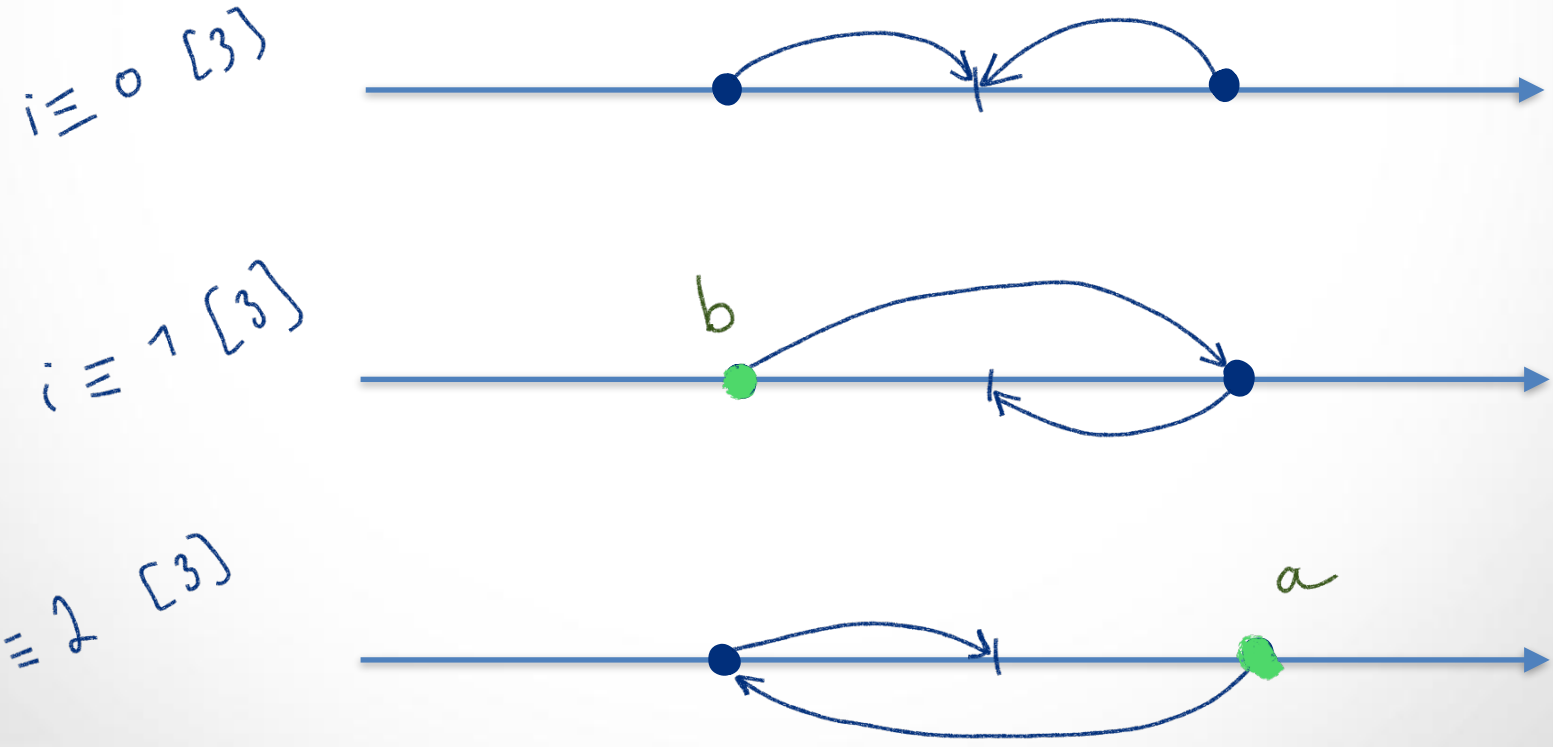
- Are on an oriented line
- ~~Have a common unit distance~~
- ~~Have rigid movements~~



Solving The SUIR in FSYNC

Assume for now that robots :

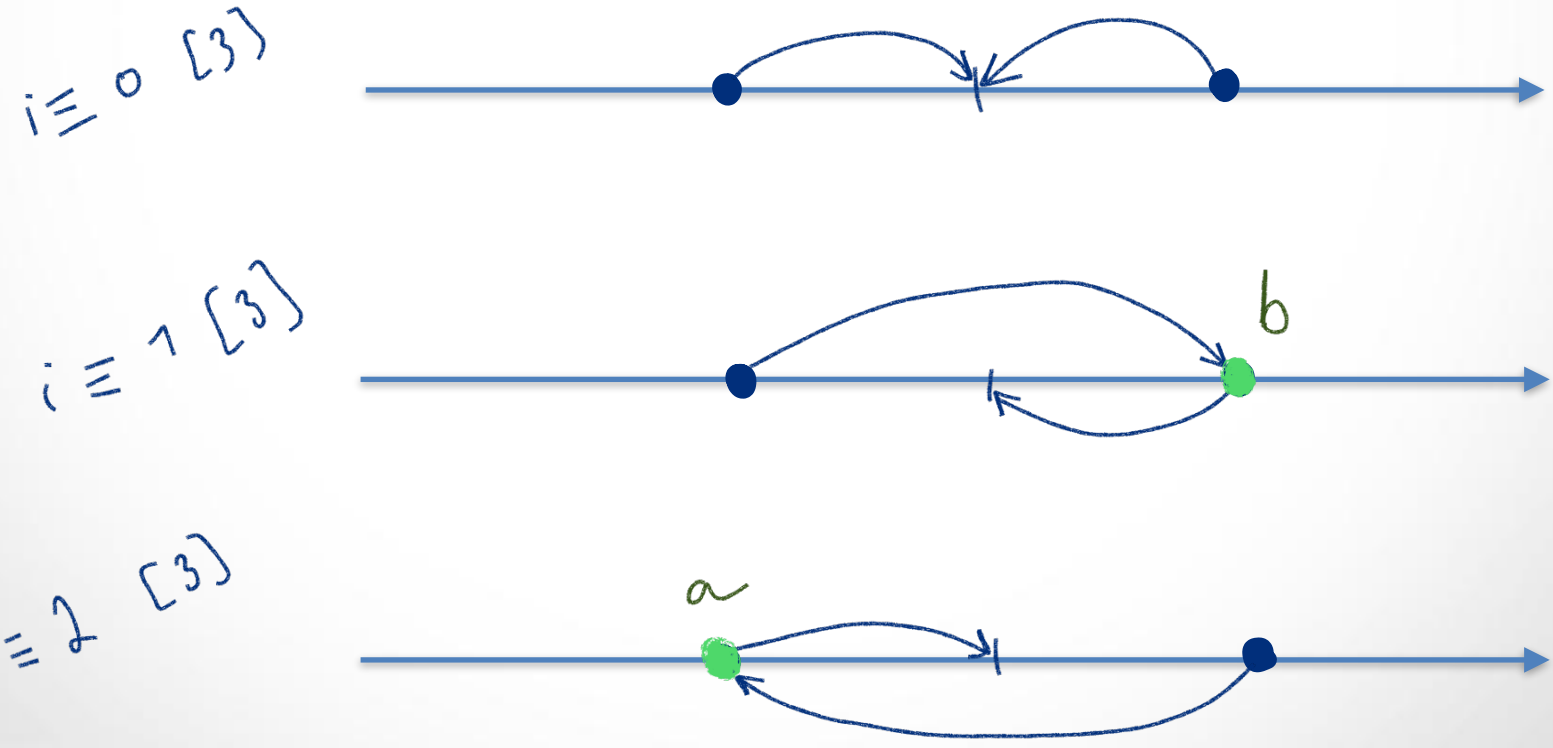
- Are on an oriented line
- ~~Have a common unit distance~~
- ~~Have rigid movements~~



Solving The SUIR in FSYNC

Assume for now that robots :

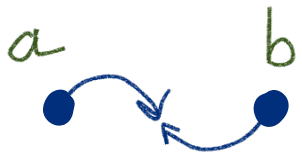
- Are on an oriented line
- ~~Have a common unit distance~~
- ~~Have rigid movements~~

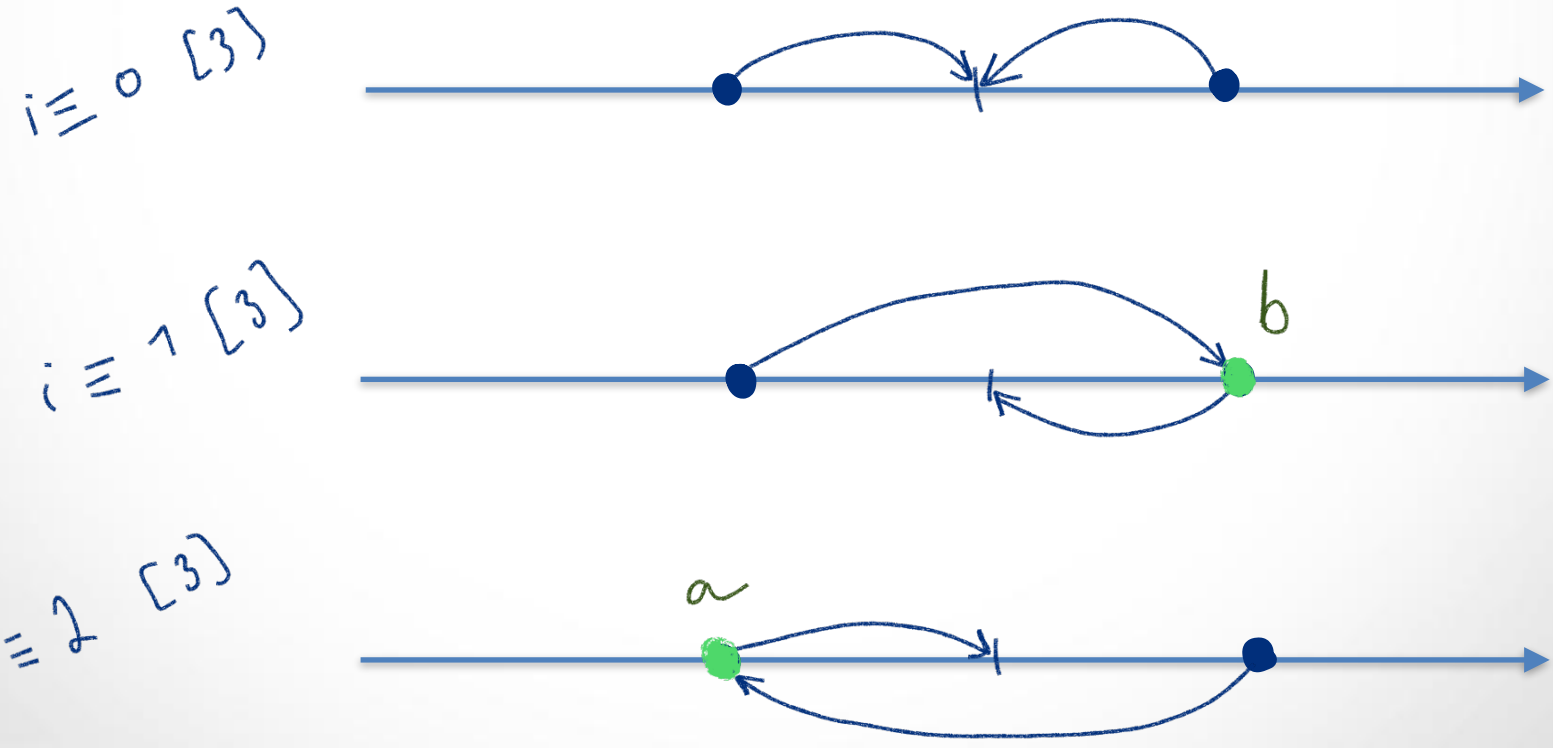


Solving The SUIR in FSYNC

Assume for now that robots :

- Are on an oriented line
- ~~Have a common unit distance~~
- ~~Have rigid movements~~





Solving The SUIR in FSYNC

Assume for now that robots :

- ~~Are on an oriented line~~
- ~~Have a common unit distance~~
- ~~Have rigid movements~~

Solving The SUIR in FSYNC

Assume for now that robots :

- ~~Are on an oriented line~~
- ~~Have a common unit distance~~
- ~~Have rigid movements~~

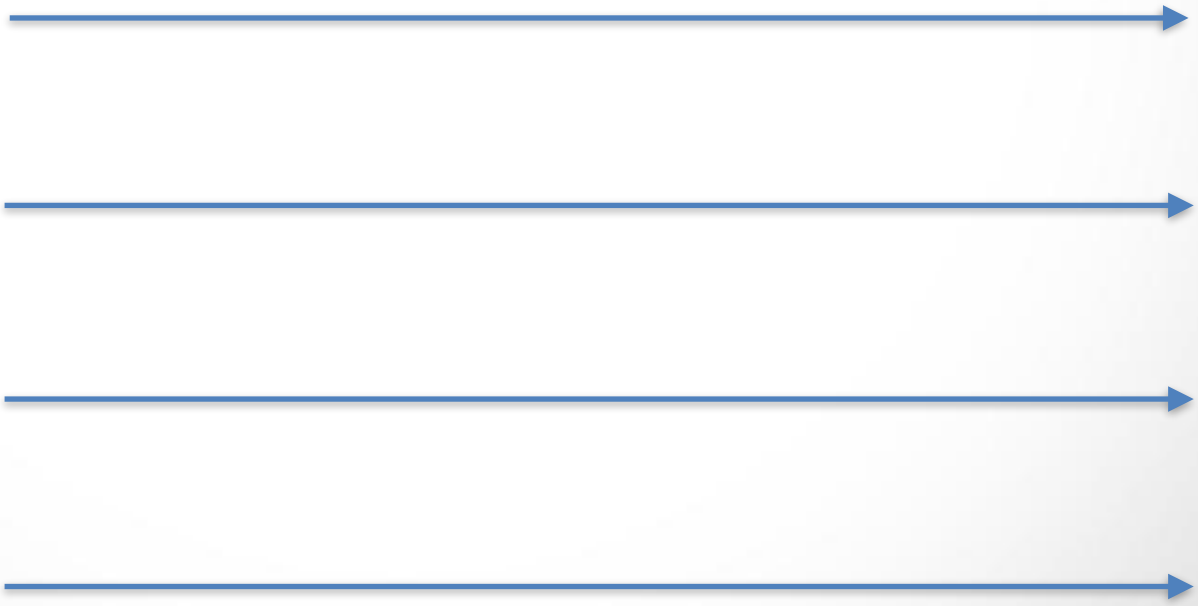
$d \in [2^{-i}, 2^{1-i})$

$i \equiv 0 \ [4]$

$i \equiv 1 \ [4]$

$i \equiv 2 \ [4]$

$i \equiv 3 \ [4]$



Solving The SUIR in FSYNC

Assume for now that robots :

- ~~Are on an oriented line~~
- ~~Have a common unit distance~~
- ~~Have rigid movements~~

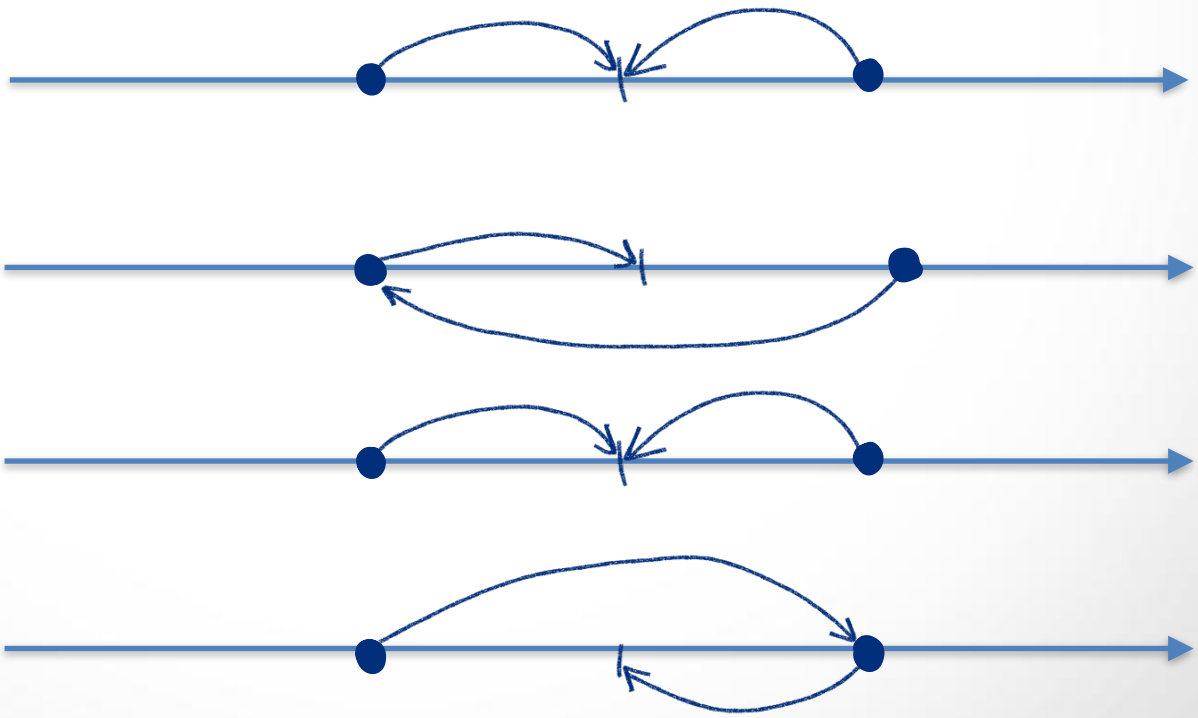
$d \in [2^{-i}, 2^{1-i})$

$i \equiv 0 \ [4]$

$i \equiv 1 \ [4]$

$i \equiv 2 \ [4]$

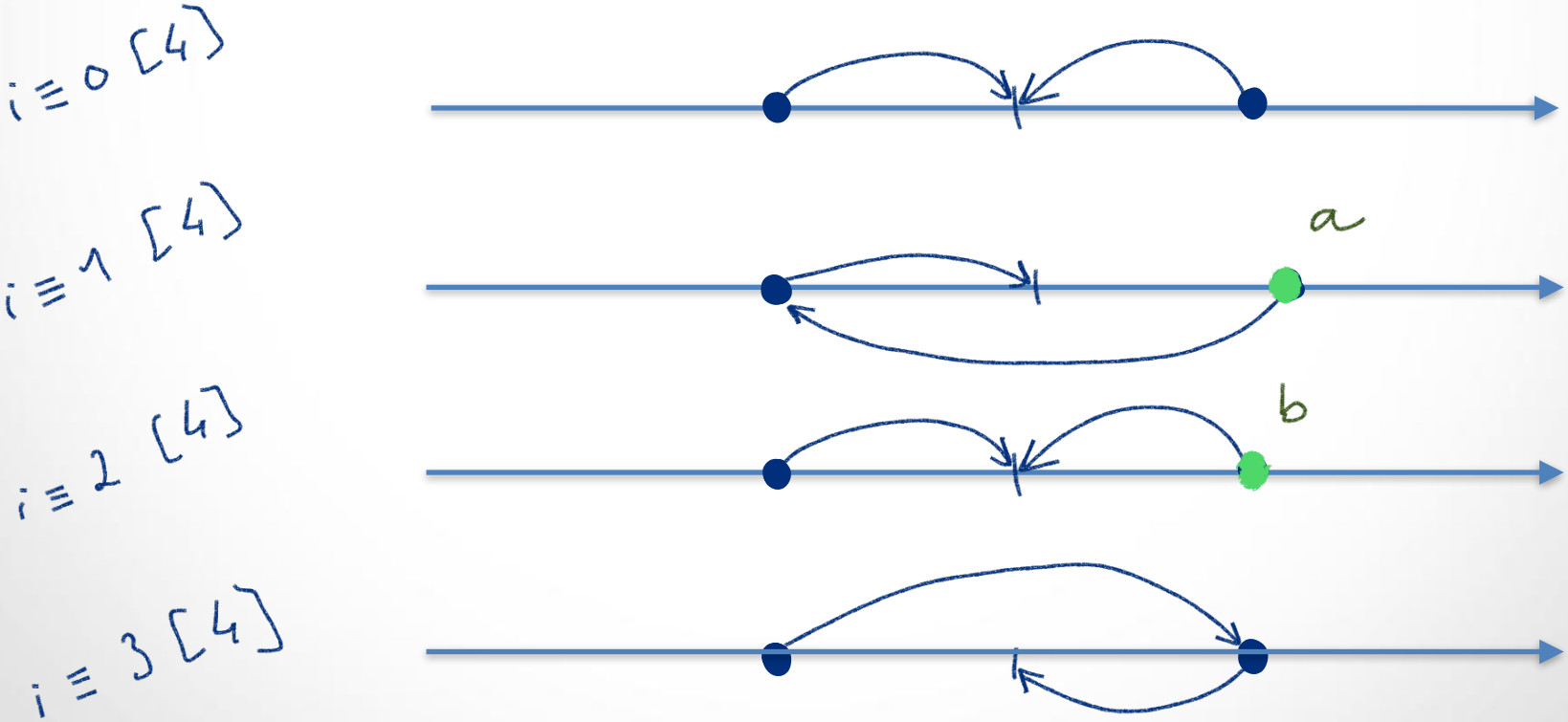
$i \equiv 3 \ [4]$



Solving The SUIR in FSYNC

Assume for now that robots :

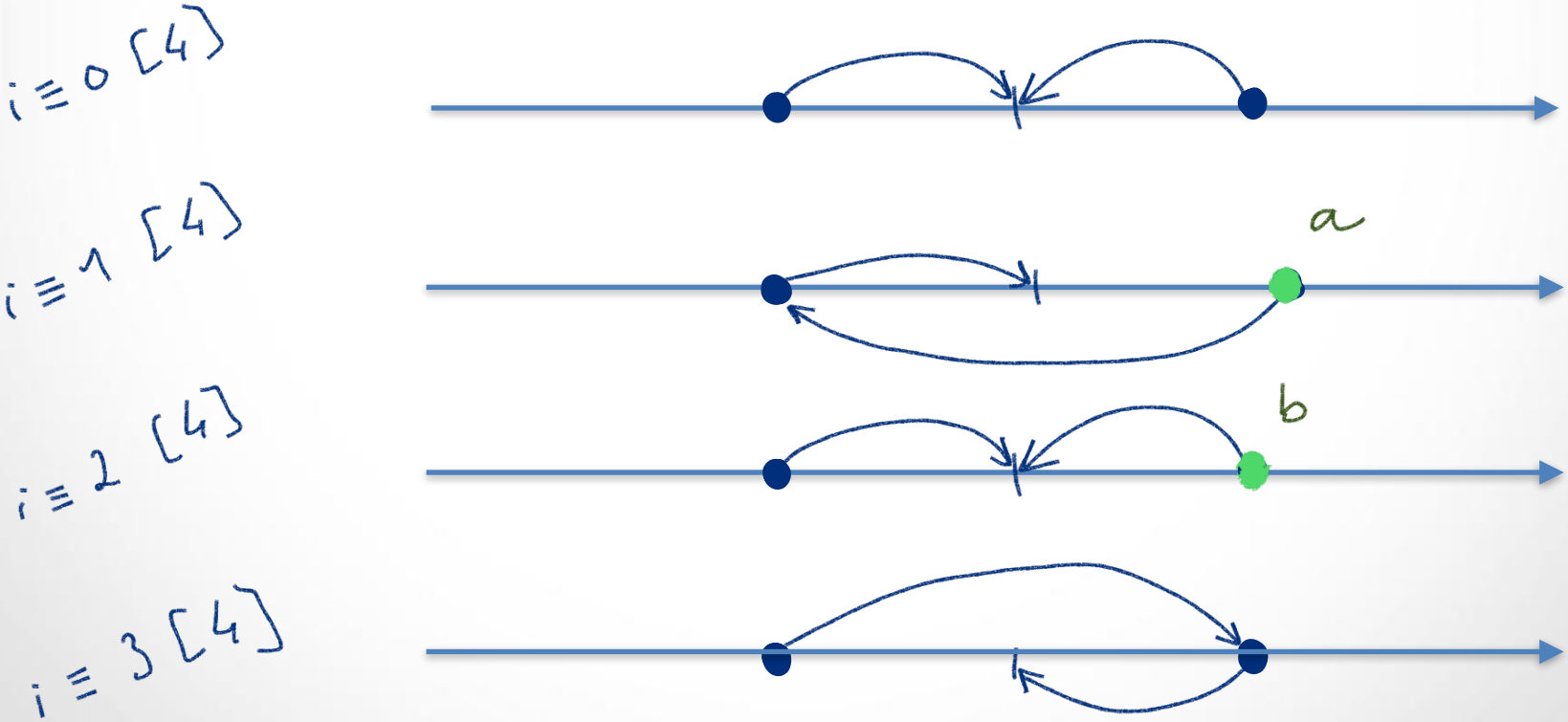
- ~~Are on an oriented line~~
- ~~Have a common unit distance~~
- ~~Have rigid movements~~



Solving The SUIR in FSYNC

Assume for now that robots :

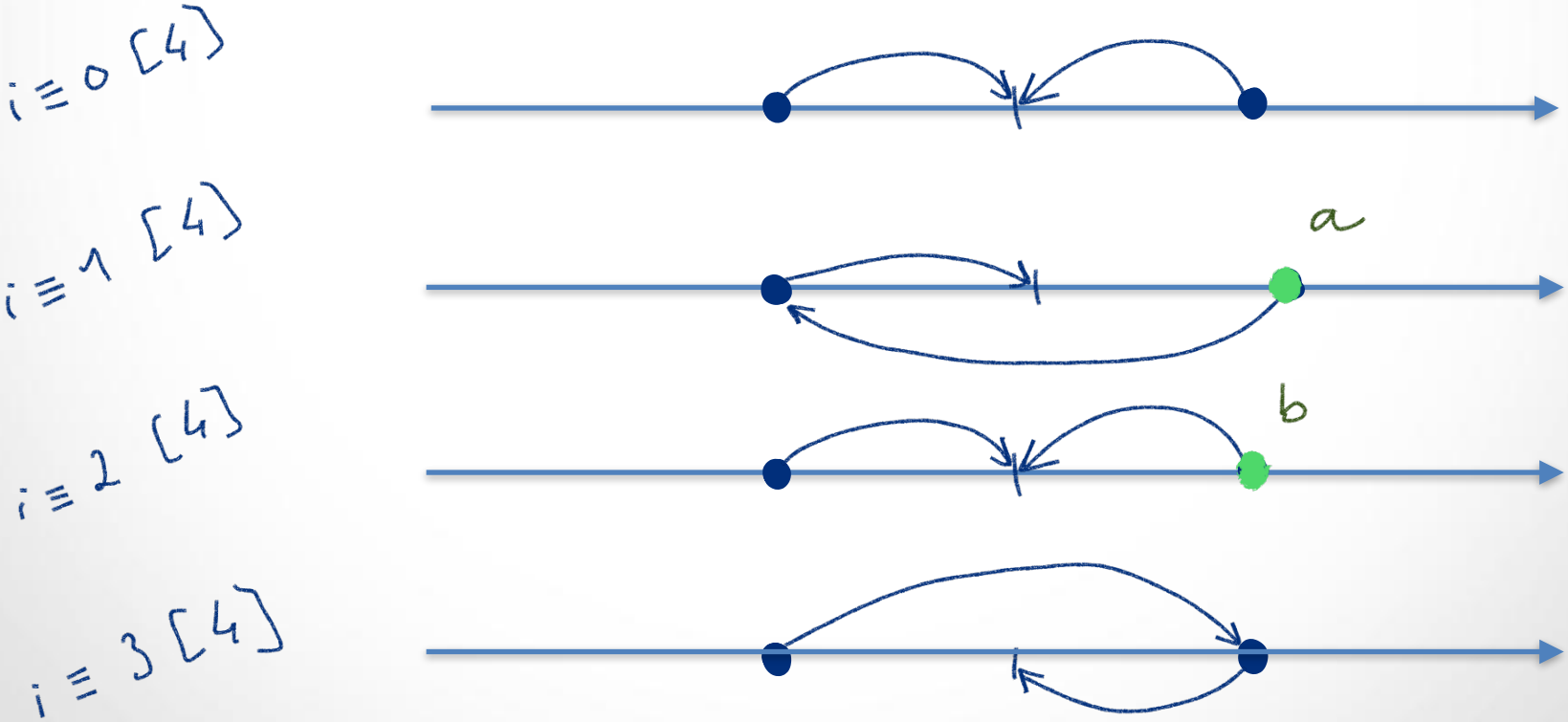
- ~~Are on an oriented line~~
- ~~Have a common unit distance~~
- ~~Have rigid movements~~



Solving The SUIR in FSYNC

Assume for now that robots :

- ~~Are on an oriented line~~
- ~~Have a common unit distance~~
- ~~Have rigid movements~~



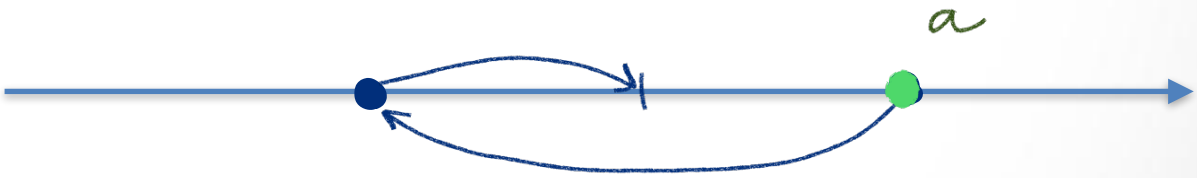
Solving The SUIR in FSYNC

Assume for now that robots :

- ~~Are on an oriented line~~
- ~~Have a common unit distance~~
- ~~Have rigid movements~~

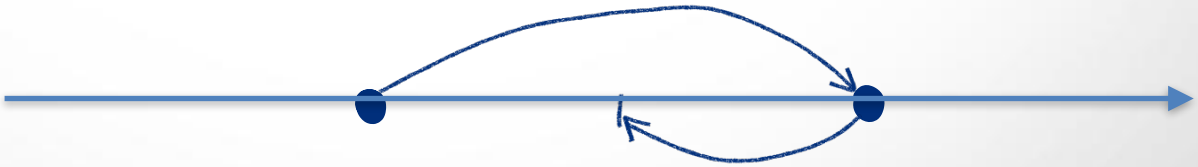
$i \equiv 0 \ [4]$

$i \equiv 1 \ [4]$



$i \equiv 2 \ [4]$

$i \equiv 3 \ [4]$



Solving The SUIR in FSYNC

Assume for now that robots :

- ~~Are on an oriented line~~
- ~~Have a common unit distance~~
- ~~Have rigid movements~~

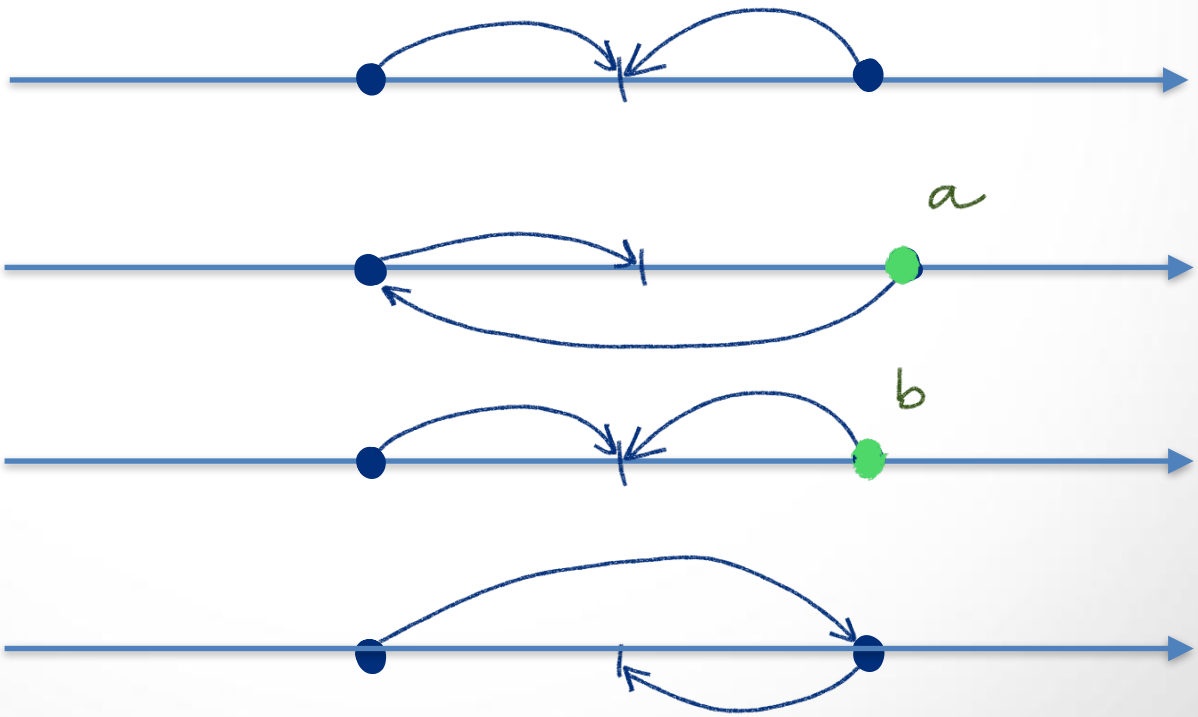
i++

$i \equiv 0 \ [4]$

$i \equiv 1 \ [4]$

$i \equiv 2 \ [4]$

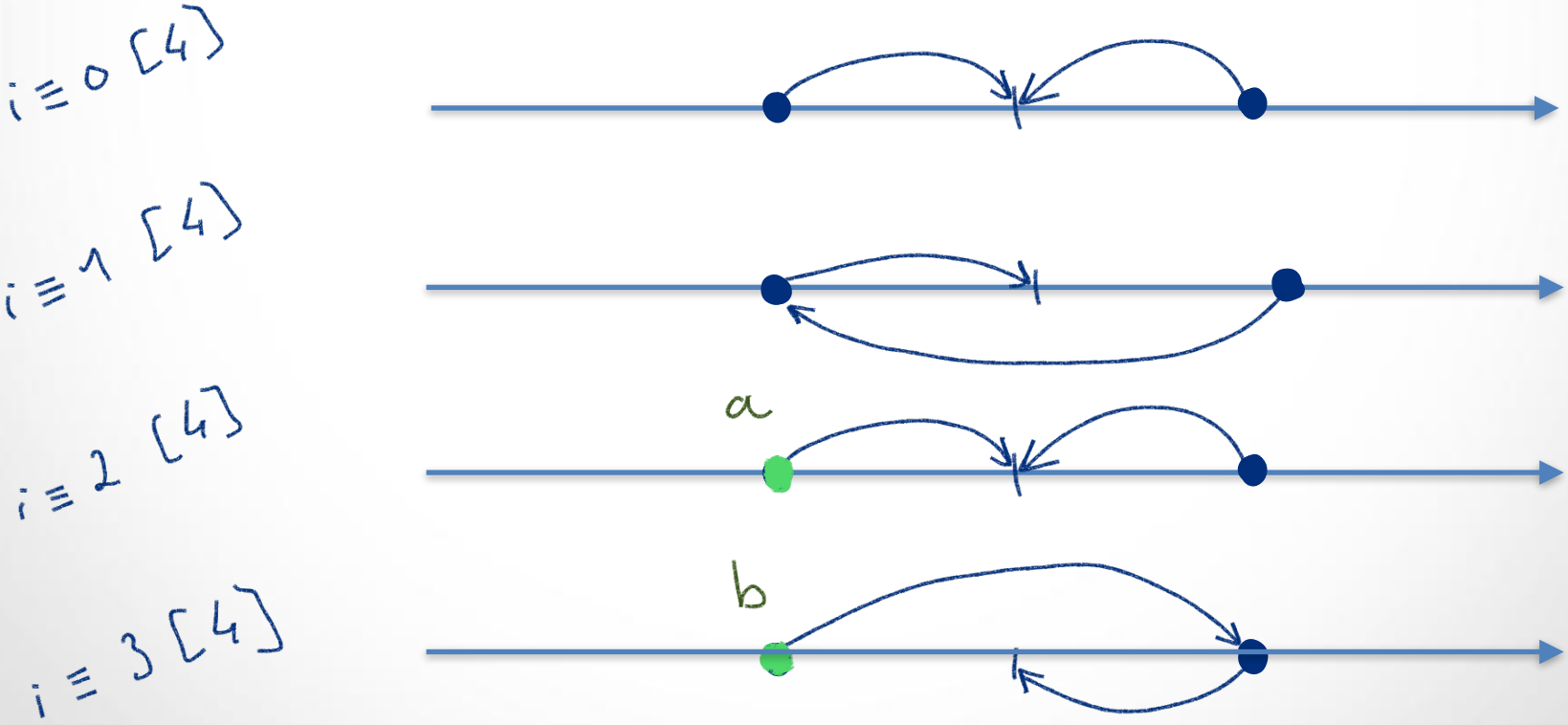
$i \equiv 3 \ [4]$



Solving The SUIR in FSYNC

Assume for now that robots :

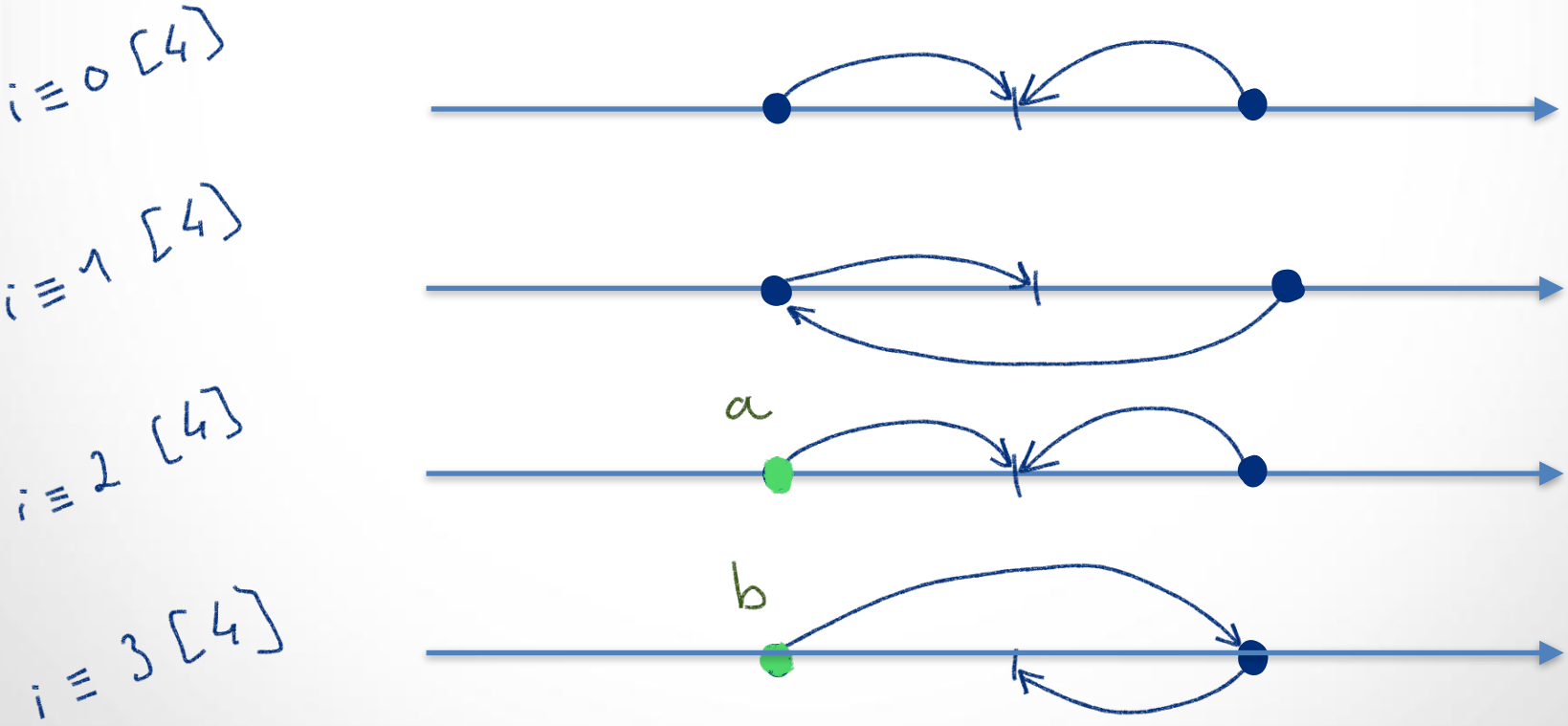
- ~~Are on an oriented line~~
- ~~Have a common unit distance~~
- ~~Have rigid movements~~



Solving The SUIR in FSYNC

Assume for now that robots :

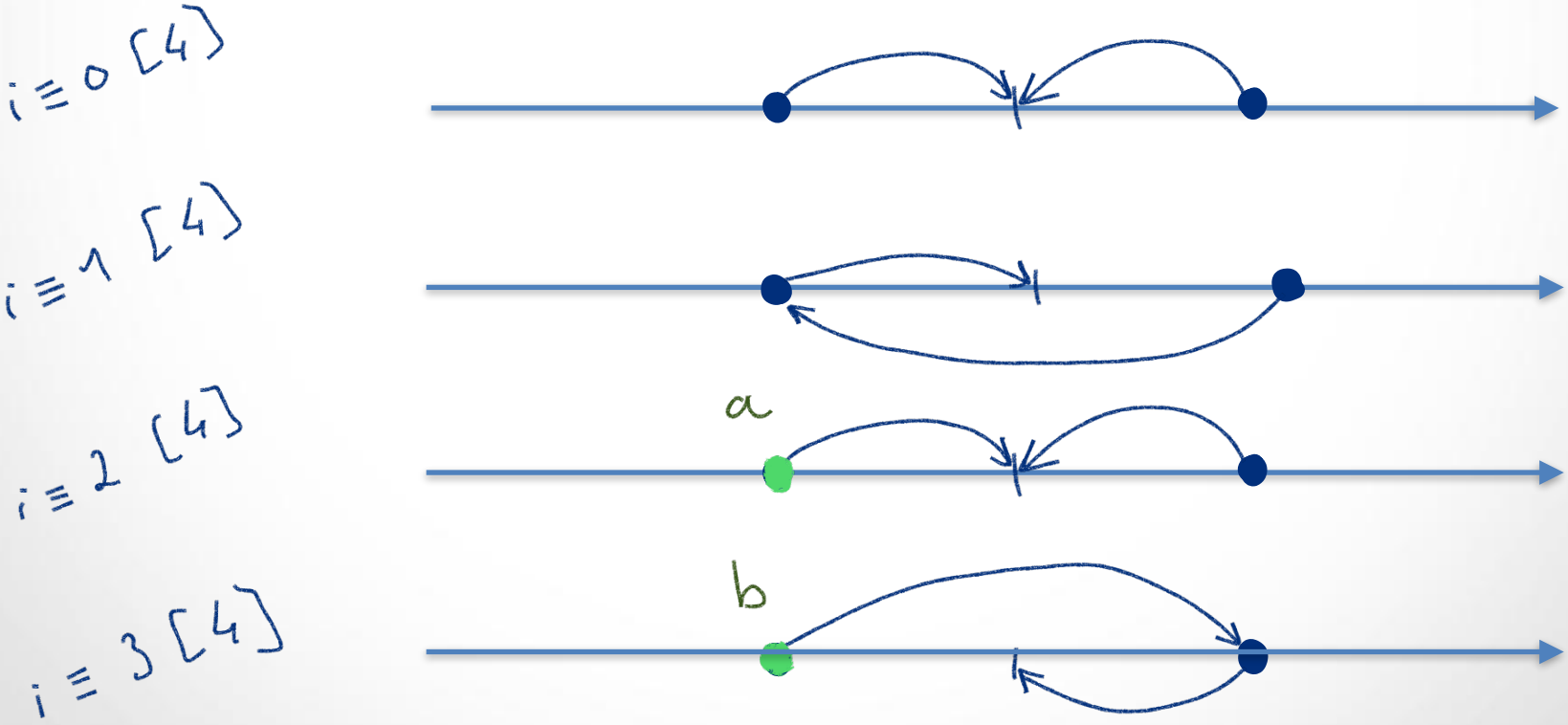
- ~~Are on an oriented line~~
- ~~Have a common unit distance~~
- ~~Have rigid movements~~



Solving The SUIR in FSYNC

Assume for now that robots :

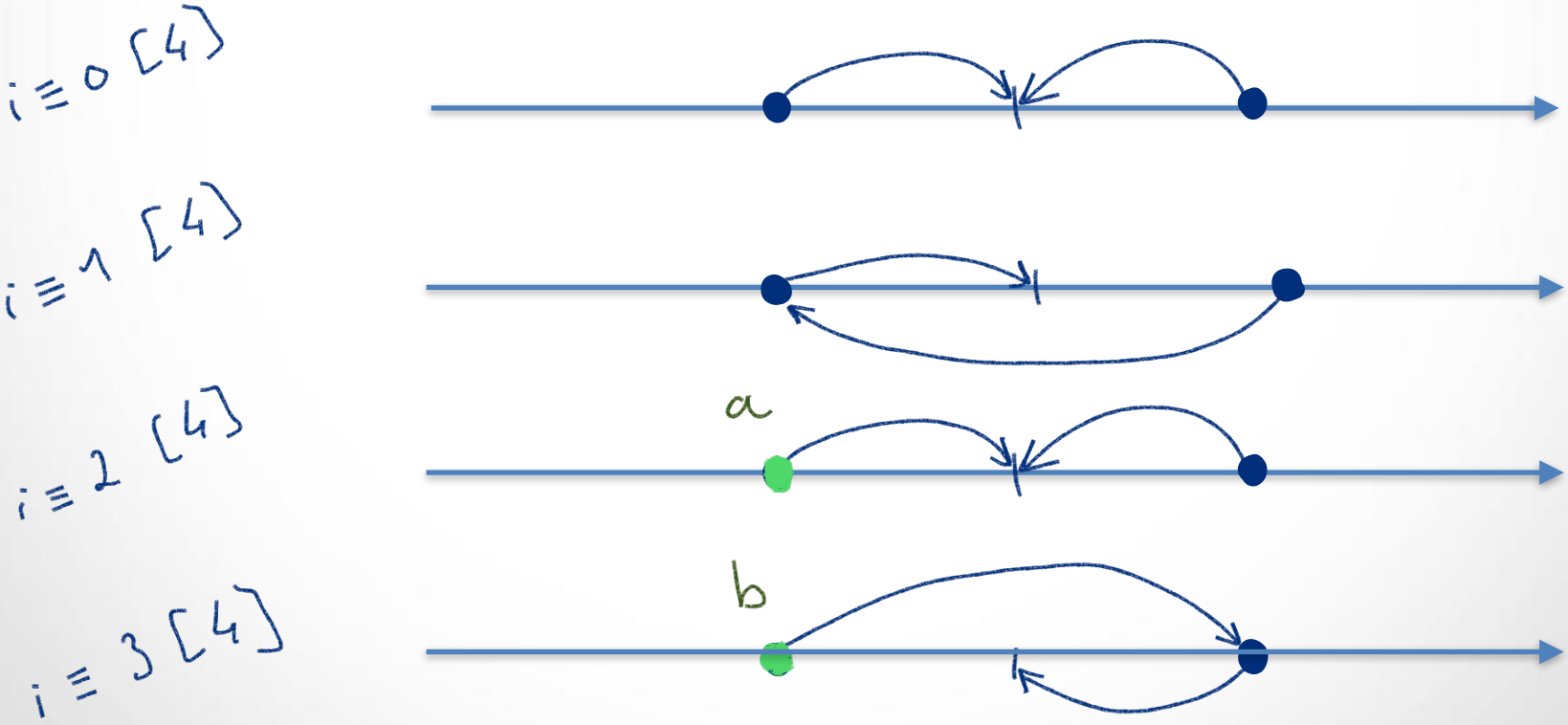
- ~~Are on an oriented line~~
- ~~Have a common unit distance~~
- ~~Have rigid movements~~



Solving The SUIR in FSYNC

Assume for now that robots :

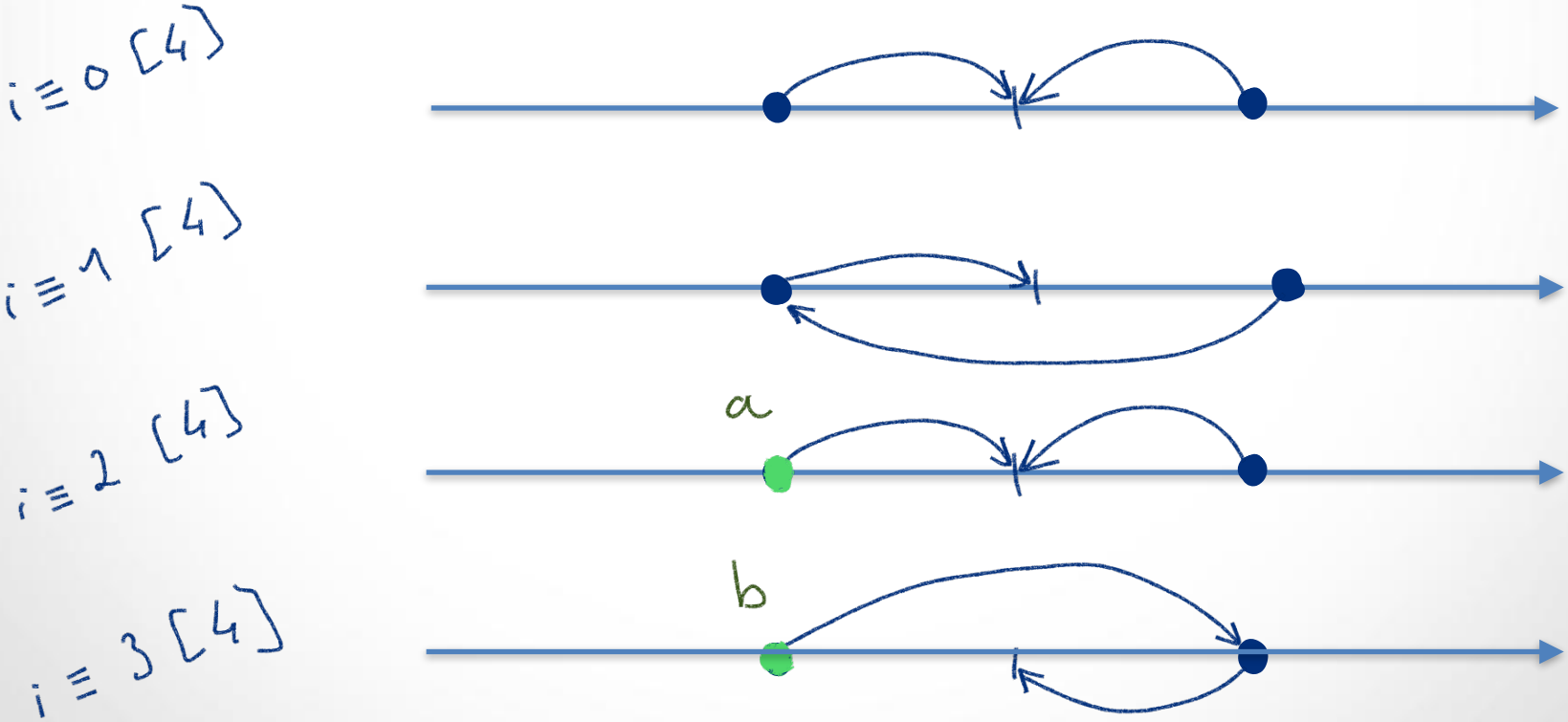
- ~~Are on an oriented line~~
- ~~Have a common unit distance~~
- ~~Have rigid movements~~



Solving The SUIR in FSYNC

Assume for now that robots :

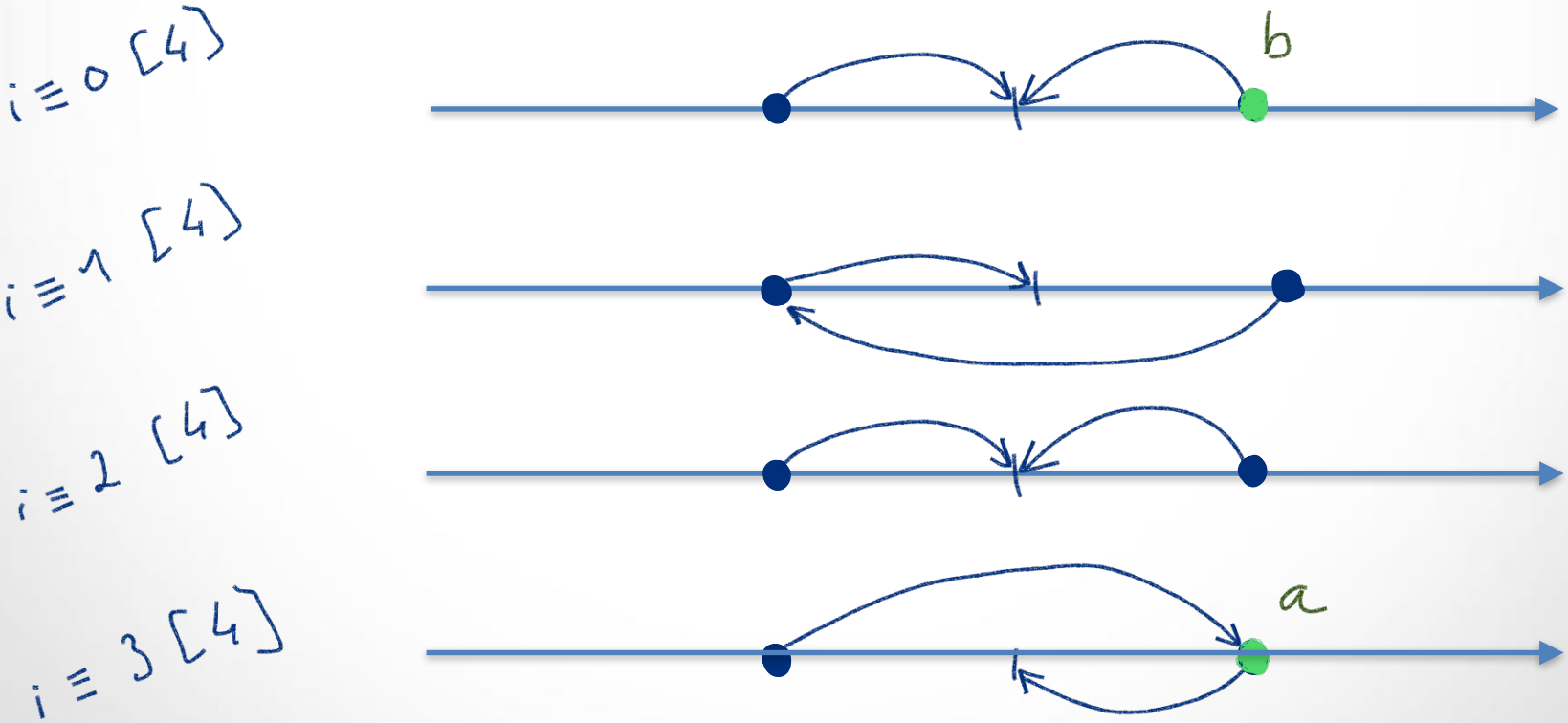
- ~~Are on an oriented line~~
- ~~Have a common unit distance~~
- ~~Have rigid movements~~



Solving The SUIR in FSYNC

Assume for now that robots :

- ~~Are on an oriented line~~
- ~~Have a common unit distance~~
- ~~Have rigid movements~~



Solving The SUIR in FSYNC

Assume for now that robots :

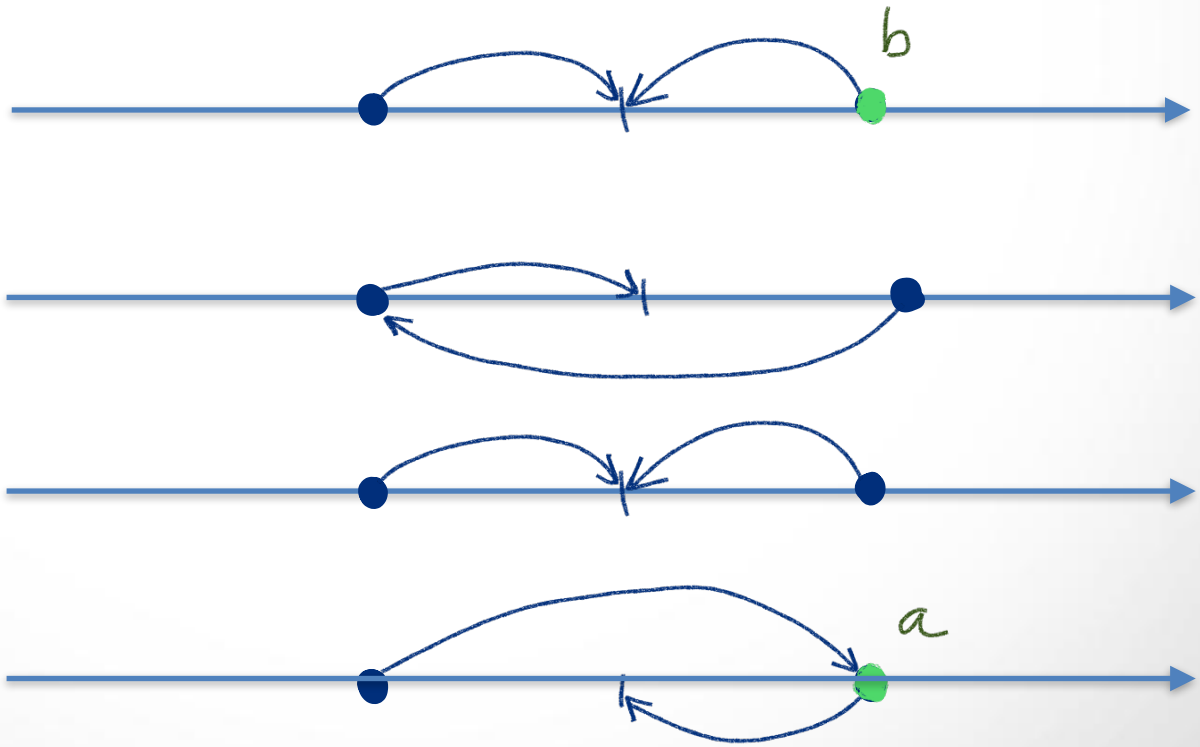
- ~~Are on an oriented line~~
- ~~Have a common unit distance~~
- ~~Have rigid movements~~

$i \equiv 0 \ [4]$

$i \equiv 1 \ [4]$

$i \equiv 2 \ [4]$

$i \equiv 3 \ [4]$



Conclusion

Conclusion

Conclusion

We defined an algorithm solving the SUIR problem, in FSYNC

Conclusion

We defined an algorithm solving the SUIR problem, in FSYNC

We showed that in SSYNC, the problem is not solvable, even with lights

Conclusion

We defined an algorithm solving the SUIR problem, in FSYNC

We showed that in SSYNC, the problem is not solvable, even with lights

Future Work

Conclusion

We defined an algorithm solving the SUIR problem, in FSYNC

We showed that in SSYNC, the problem is not solvable, even with lights

Future Work

Generalize our results with n robots tolerating at most one fault

Conclusion

We defined an algorithm solving the SUIR problem, in FSYNC

We showed that in SSYNC, the problem is not solvable, even with lights

Future Work

Generalize our results with n robots tolerating at most one fault

Thank you for your attention