How do you program 1000 robots?

Giovanni Beltrame - MIST Lab

7th PLOW, Montreal, 2017

Outline

Outline

Outline

1. From Swarm Intelligence to Swarm Robotics
2. Swarms as Programmable Machines
3. Buzz: Multi-Paradigm Swarm Programming

From Swarm Intelligence to Swarm Robotics

Natural Swarm Intelligence

Swarm Intelligence

From Natural to Artificial Swarm Intelligence

From Natural to Artificial Swarm Intelligence

Properties

From Natural to Artificial Swarm Intelligence

Properties

- Efficiency

From Natural to Artificial Swarm Intelligence

Properties

- Efficiency
- Robustness

From Natural to Artificial Swarm Intelligence

Properties

- Efficiency
- Robustness
- Parallelism

From Natural to Artificial Swarm Intelligence

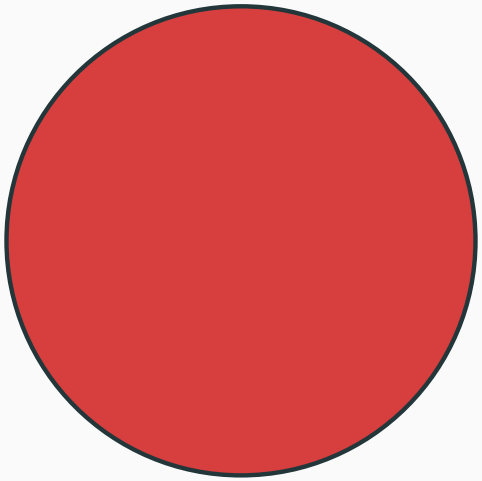
Properties

- Efficiency
- Robustness
- Parallelism
- Adaptivity

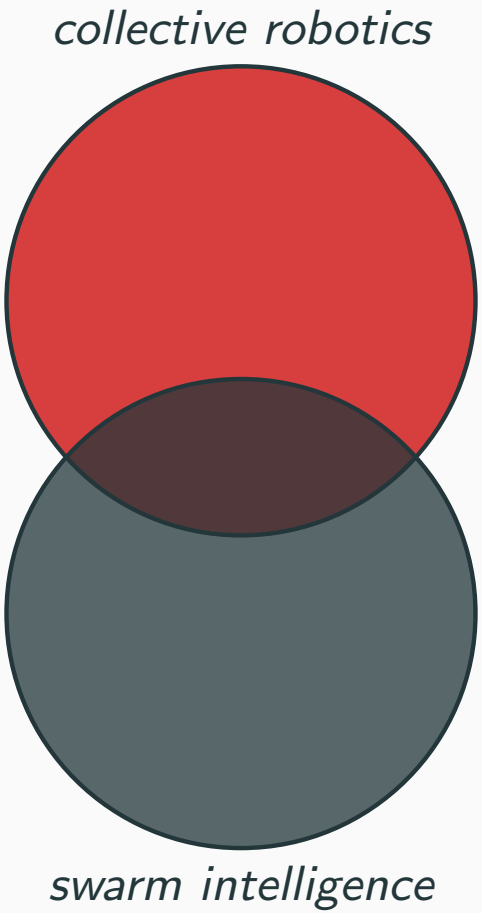
From Natural to Artificial Swarm Intelligence

From Natural to Artificial Swarm Intelligence

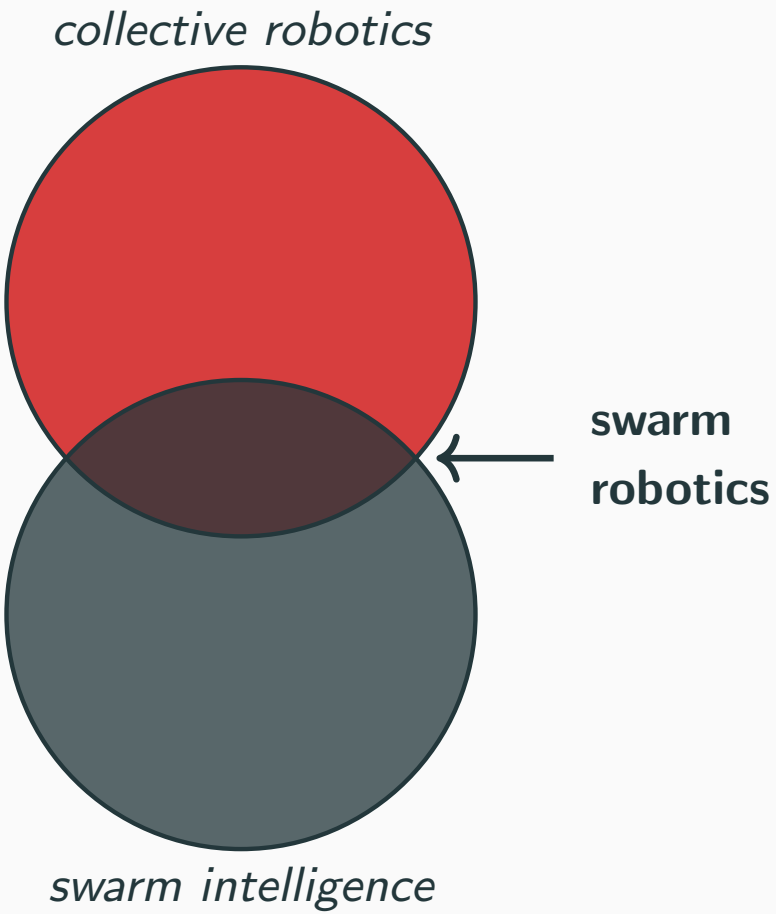
collective robotics



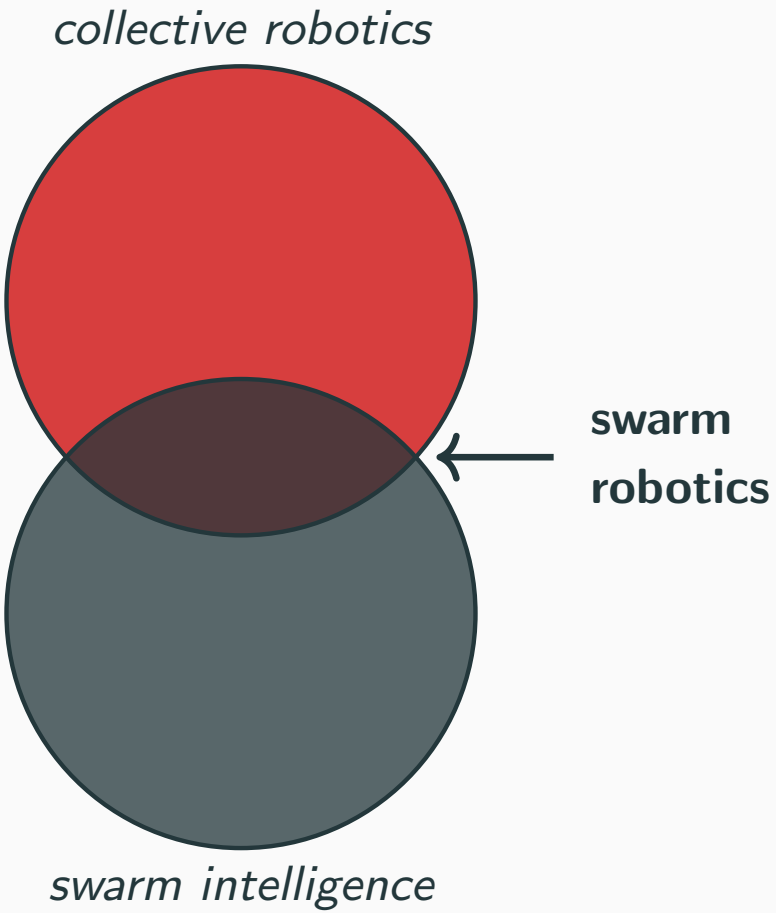
From Natural to Artificial Swarm Intelligence



From Natural to Artificial Swarm Intelligence

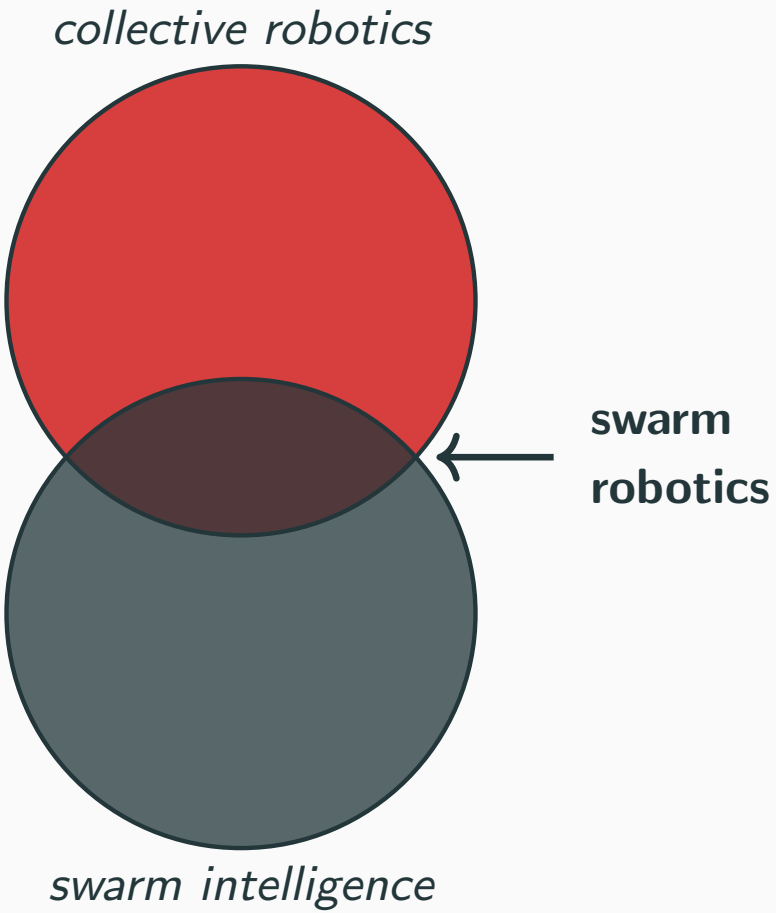


From Natural to Artificial Swarm Intelligence



Design principles

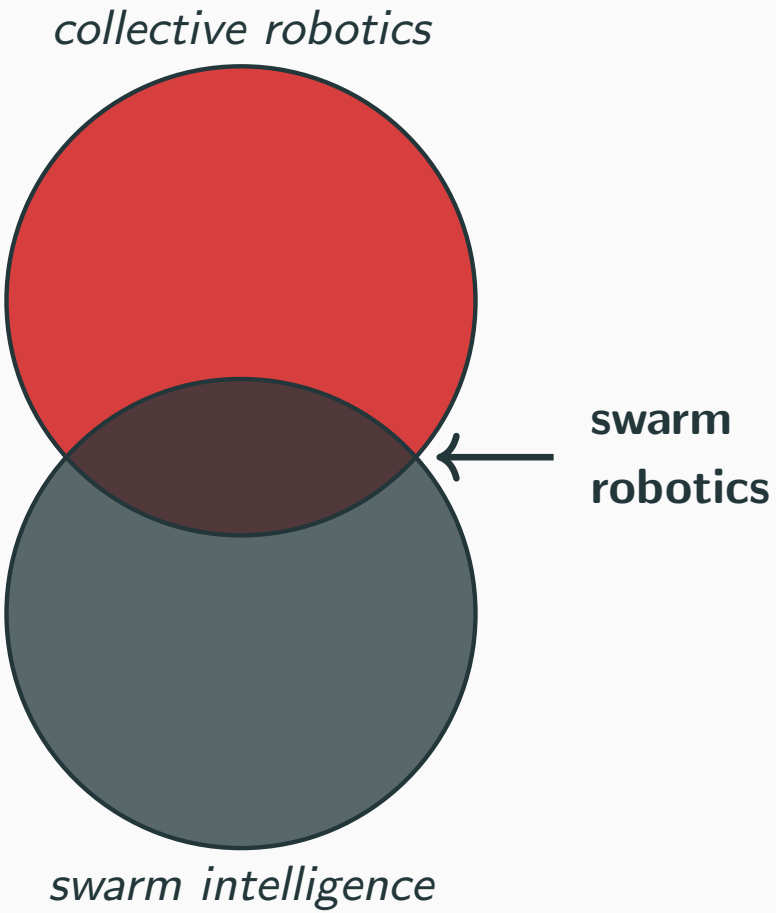
From Natural to Artificial Swarm Intelligence



Design principles

- Decentralized control

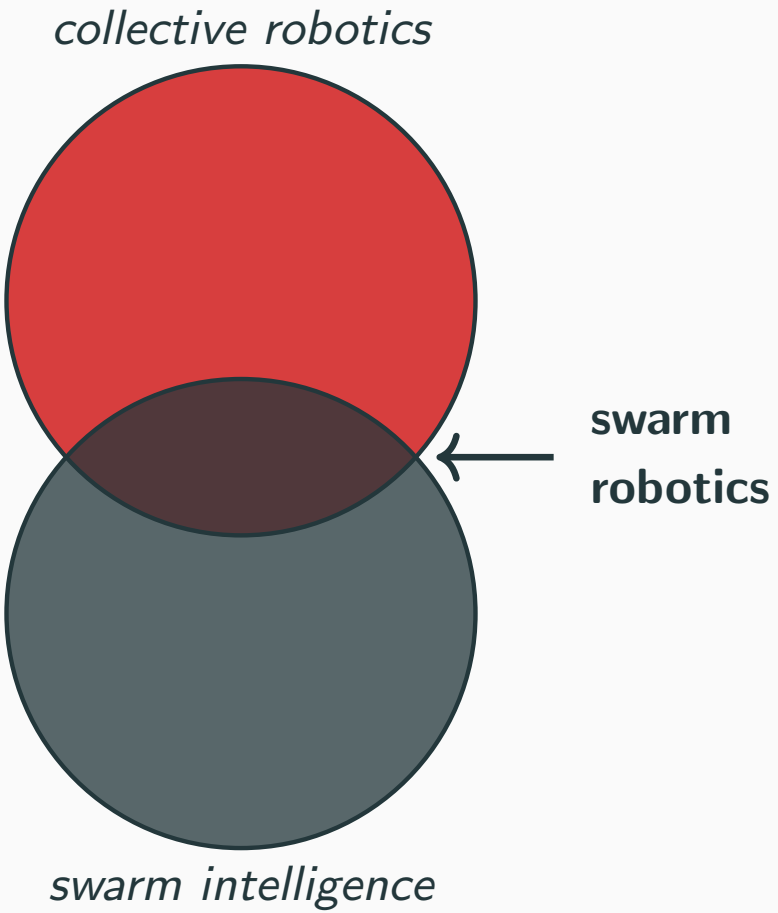
From Natural to Artificial Swarm Intelligence



Design principles

- Decentralized control
- No leaders

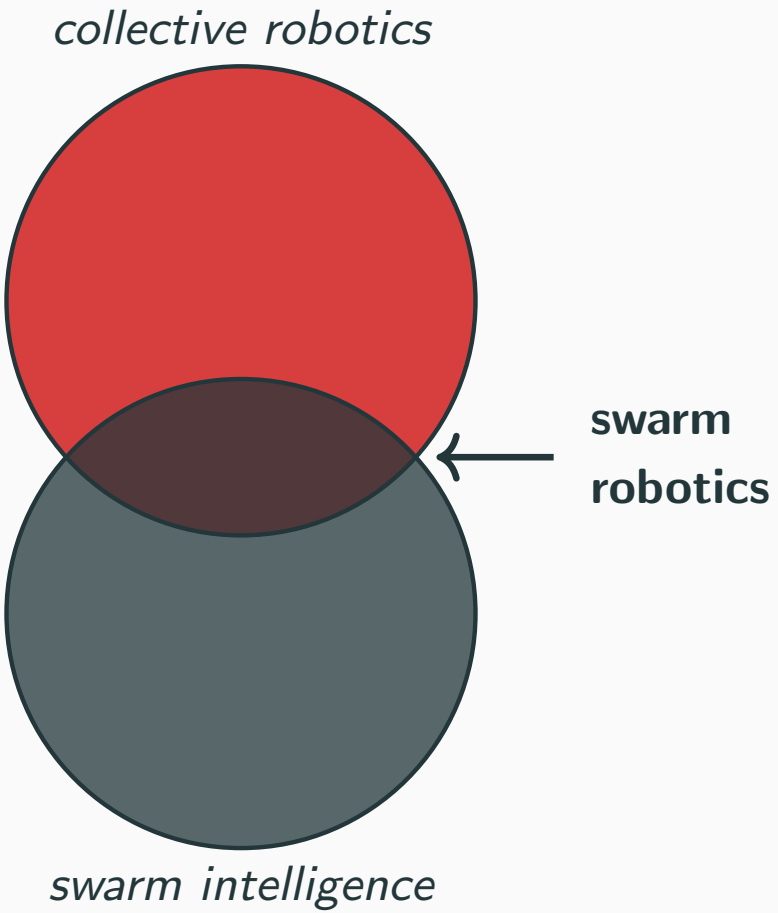
From Natural to Artificial Swarm Intelligence



Design principles

- Decentralized control
- No leaders
- No predefined roles

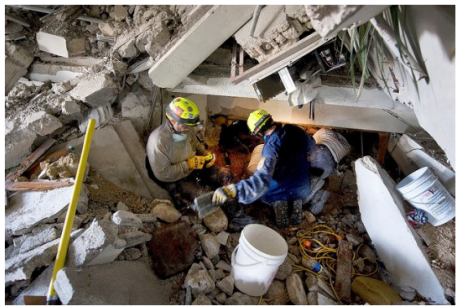
From Natural to Artificial Swarm Intelligence

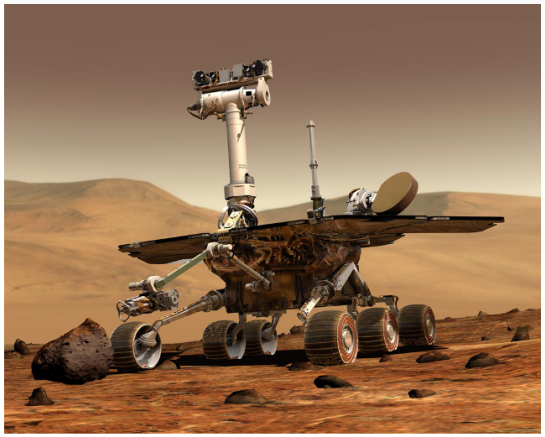


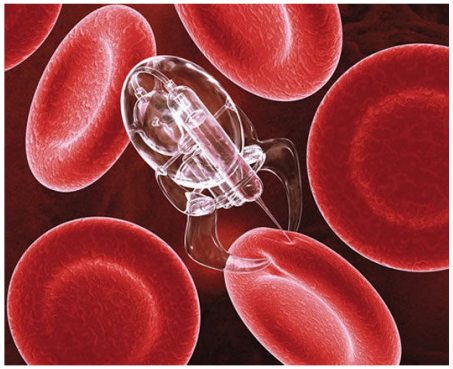
Design principles

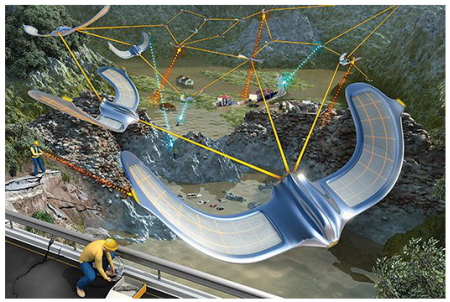
- Decentralized control
- No leaders
- No predefined roles
- Simple, local interactions

Swarm Robotics: Potential Applications









Swarm Design: The Global-to-Local problem

Swarm Design: The Global-to-Local problem

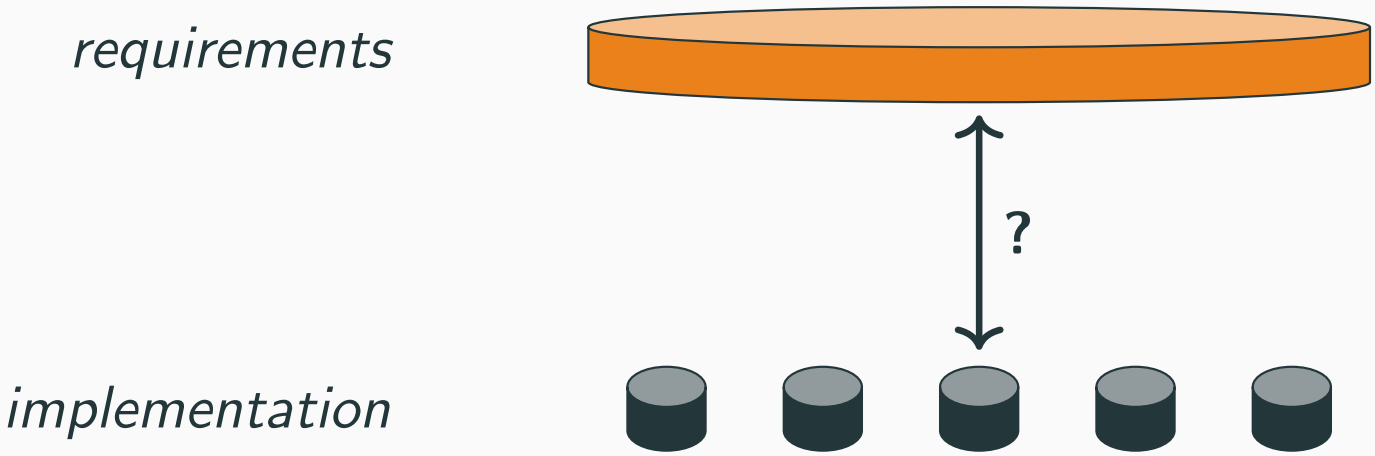
requirements

Swarm Design: The Global-to-Local problem

requirements

implementation

Swarm Design: The Global-to-Local problem



Approaches to Swarm Design

Automatic Methods

“Manual” Methods

Approaches to Swarm Design

Automatic Methods

- Focus on *what* rather than *how*

"Manual" Methods

Approaches to Swarm Design

Automatic Methods

- Focus on *what* rather than *how*
- Main approaches

"Manual" Methods

Approaches to Swarm Design

Automatic Methods

- Focus on *what* rather than *how*
- Main approaches
    - Neural networks + genetic algorithm

"Manual" Methods

Approaches to Swarm Design

Automatic Methods

- Focus on *what* rather than *how*
- Main approaches
  - Neural networks + genetic algorithm
  - Parametric FSA + optimization algorithm

"Manual" Methods

Approaches to Swarm Design

Automatic Methods

- Focus on *what* rather than *how*
- Main approaches
  - Neural networks + genetic algorithm
  - Parametric FSA + optimization algorithm
  - Behavioral trees

"Manual" Methods

Approaches to Swarm Design

Automatic Methods

- Focus on *what* rather than *how*
- Main approaches
  - Neural networks + genetic algorithm
  - Parametric FSA + optimization algorithm
  - Behavioral trees

"Manual" Methods

- Focus on *how* rather than *what*

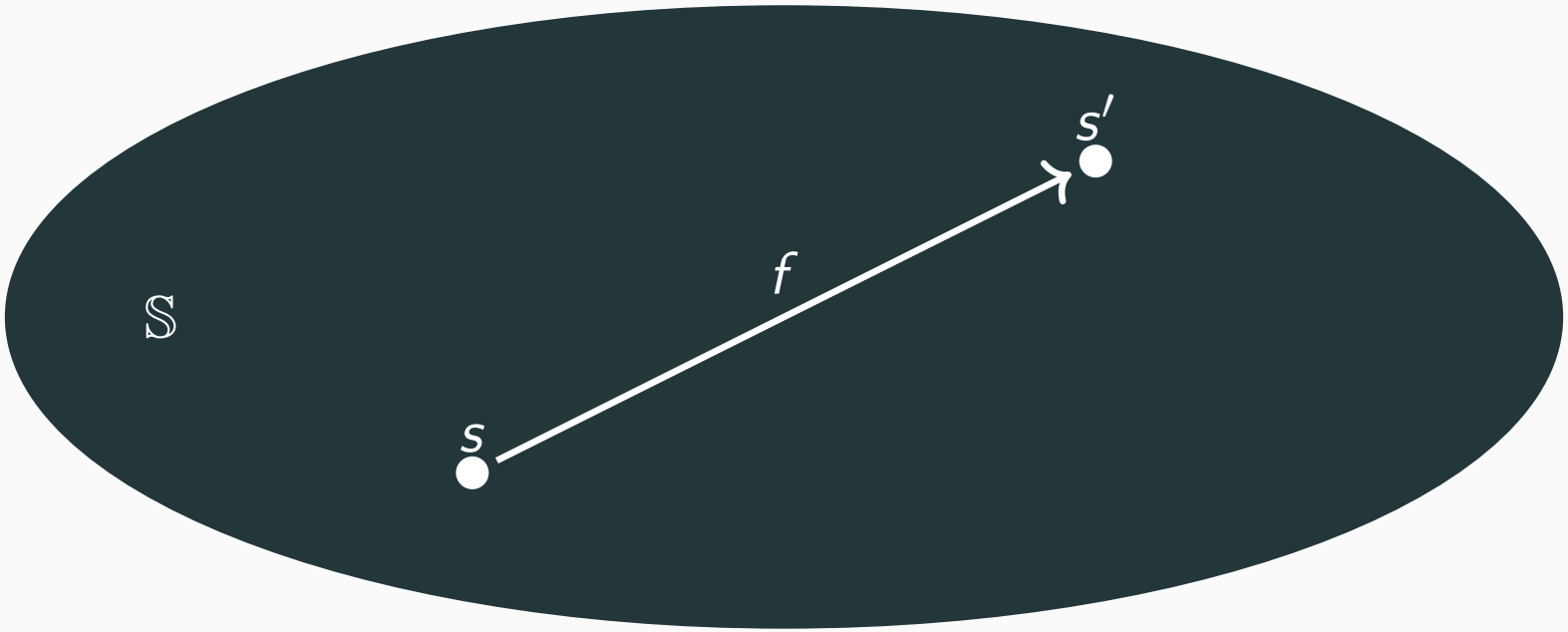
Swarms as Programmable Machines

Computation in a Swarm Machine

$$f : \mathbb{S} \to \mathbb{S}$$

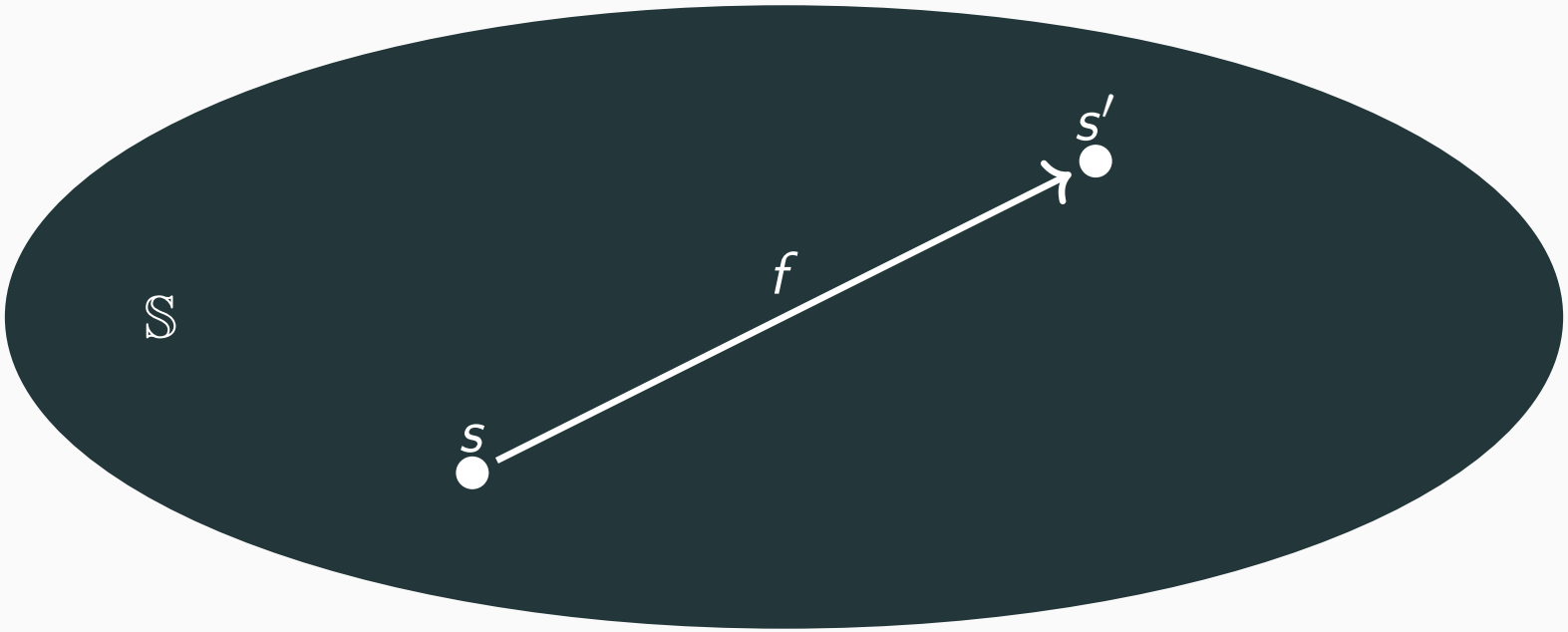
Computation in a Swarm Machine

$$f : \mathbb{S} \to \mathbb{S}$$



Computation in a Swarm Machine

$$f : \mathbb{S} \to \mathbb{S}$$

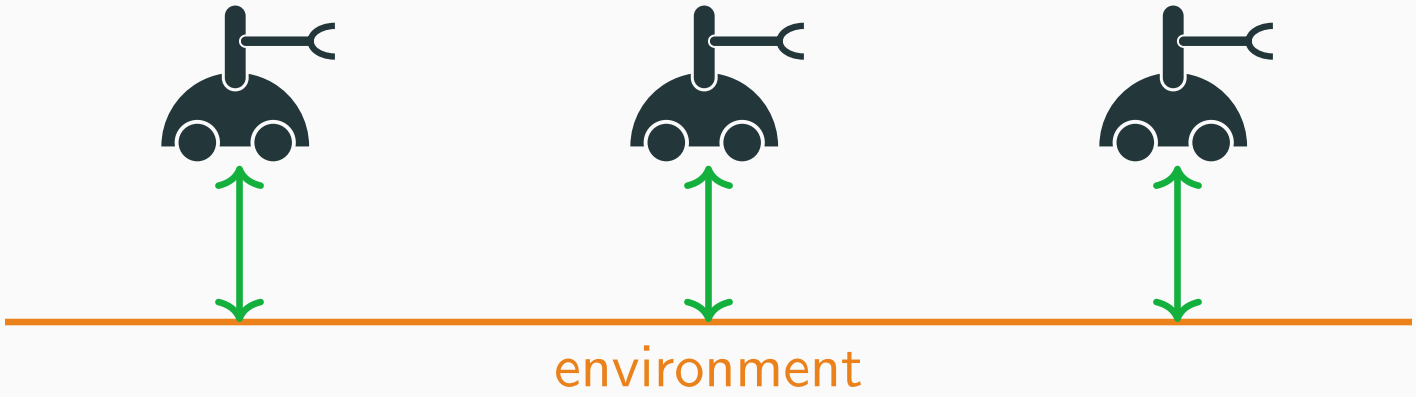


What is $\mathbb{S}$?

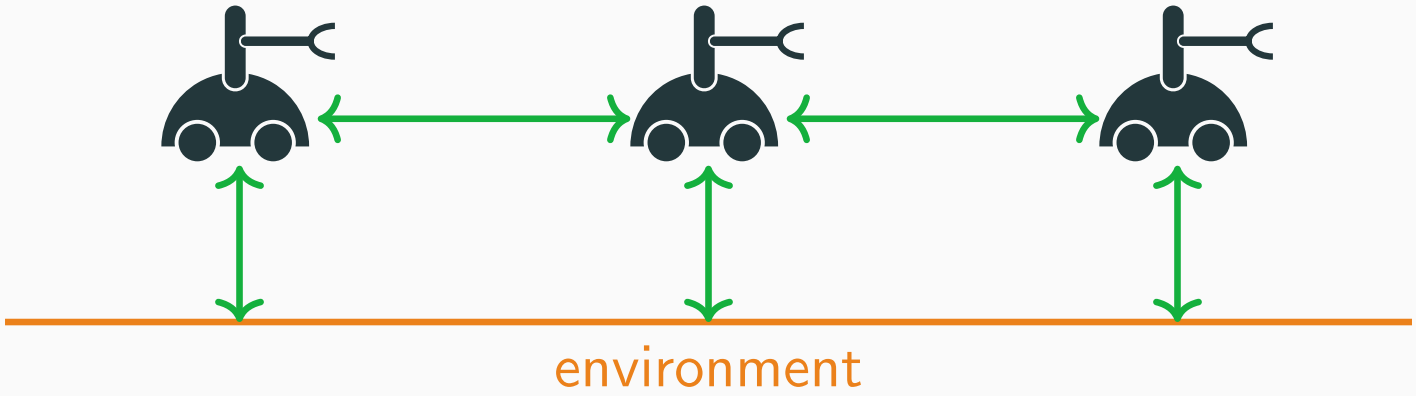
Information Flow in a Swarm Machine

environment

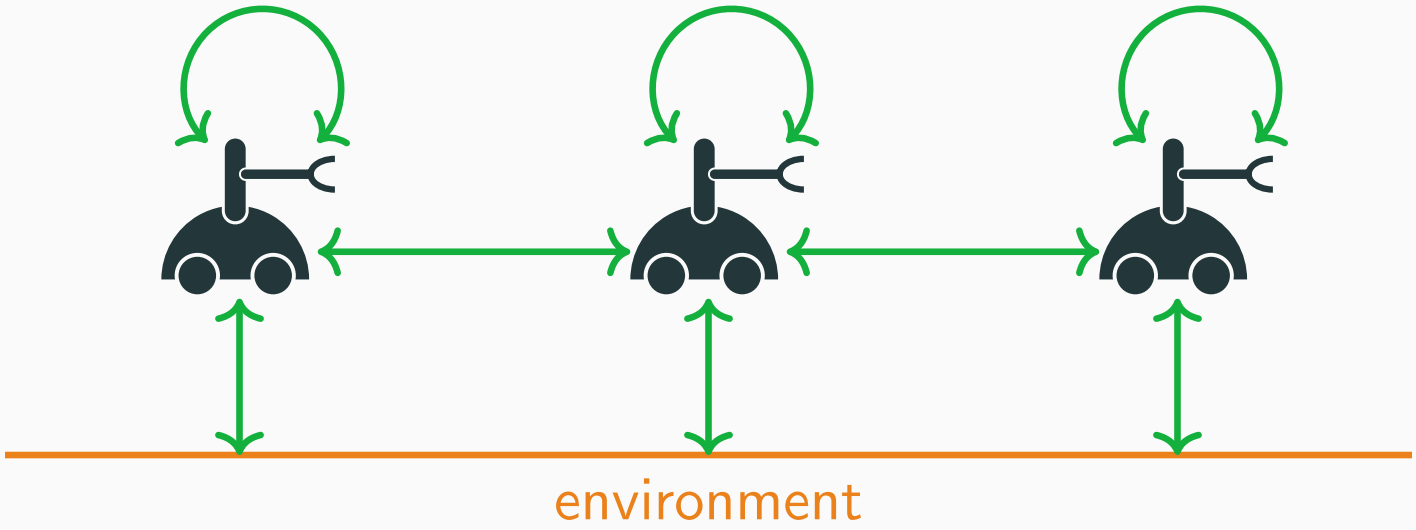
Information Flow in a Swarm Machine



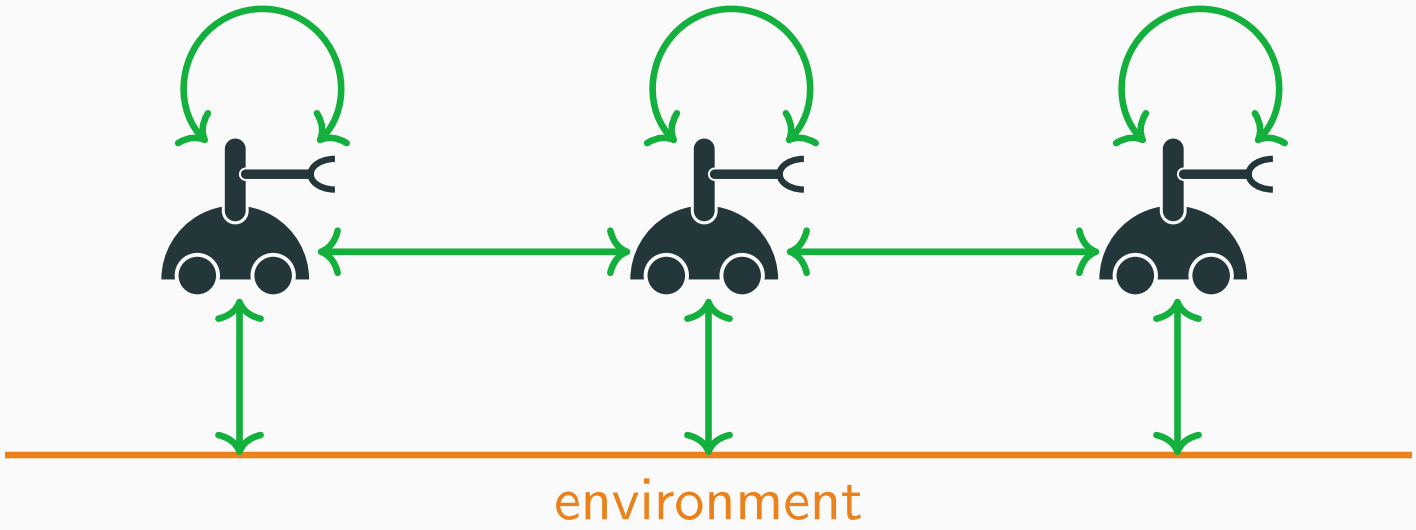
Information Flow in a Swarm Machine



Information Flow in a Swarm Machine

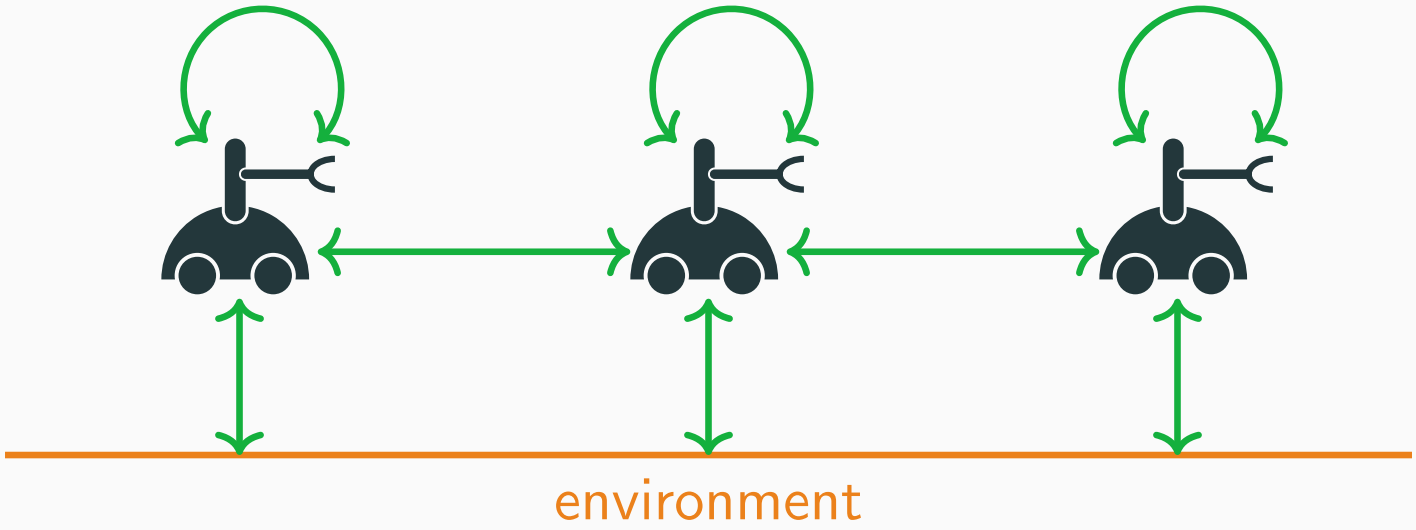


Information Flow in a Swarm Machine



$\mathbb{S}$ **includes:**

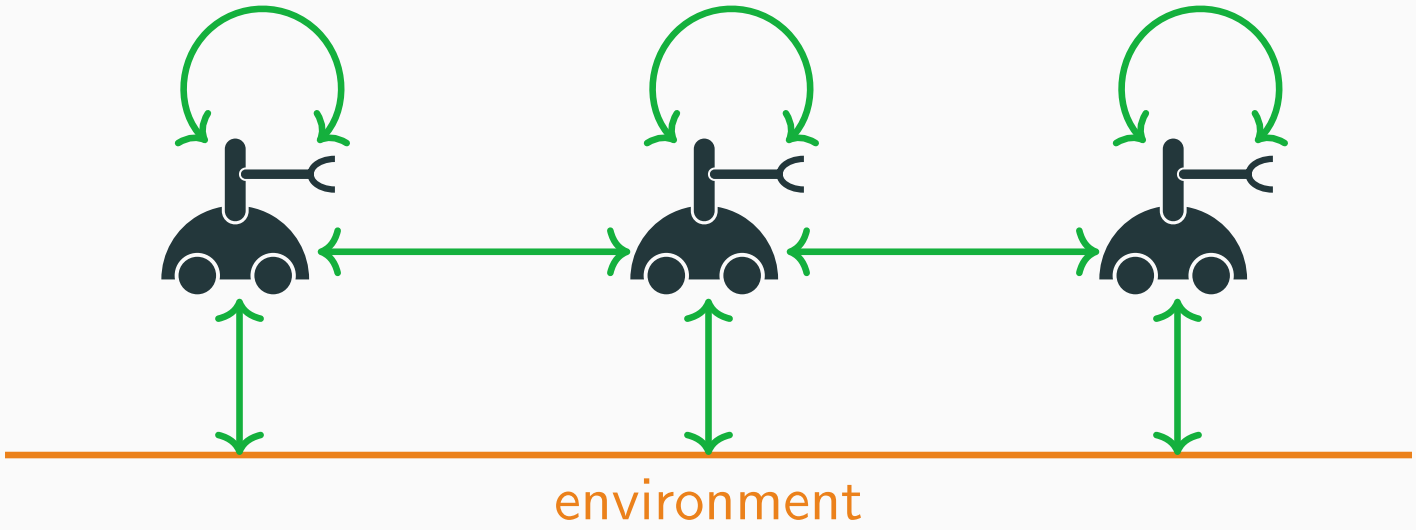
Information Flow in a Swarm Machine



$\mathbb{S}$ includes:

- Internal state of each robot (e.g., memory, battery level)

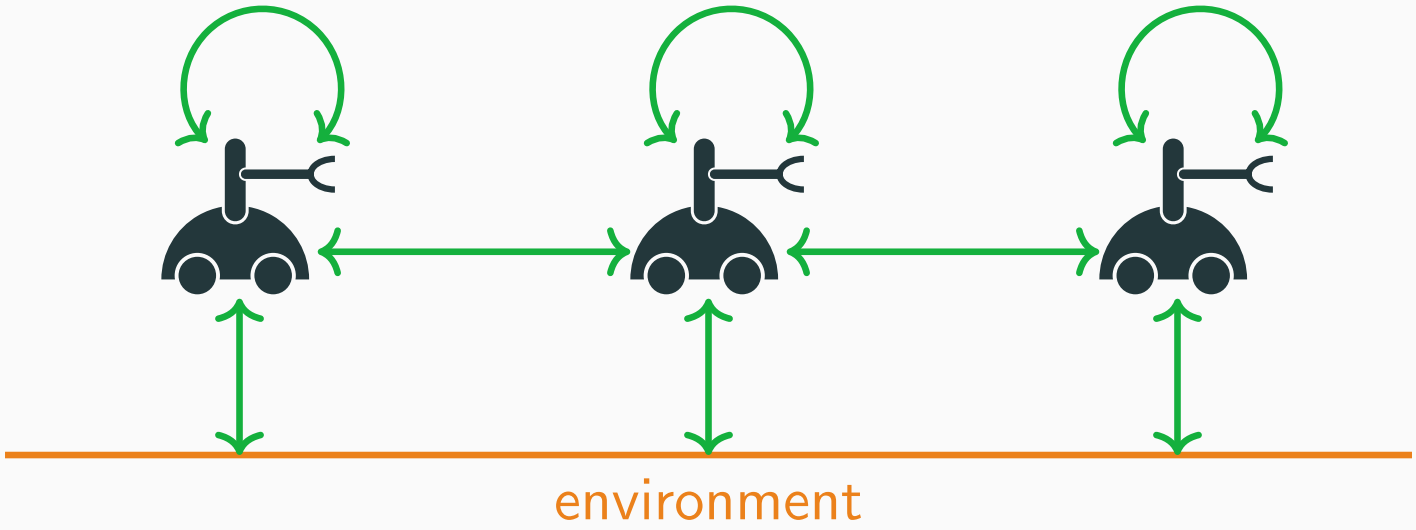
Information Flow in a Swarm Machine



$\mathbb{S}$ includes:

- Internal state of each robot (e.g., memory, battery level)
- Physical state of each robot (e.g., mechanics)

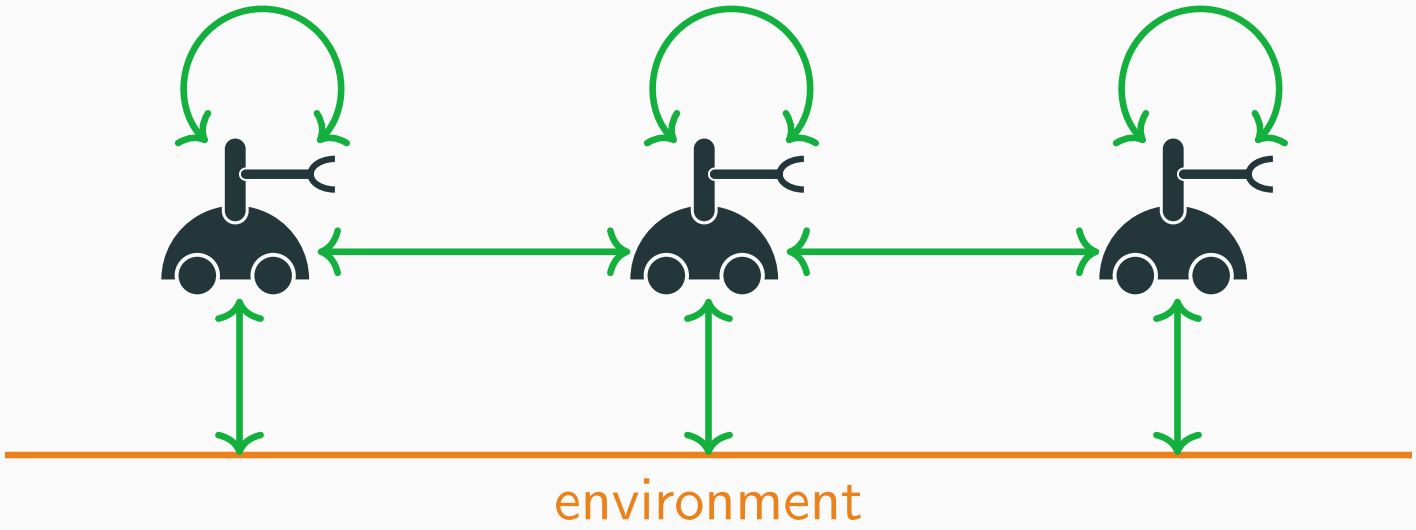
Information Flow in a Swarm Machine



$\mathbb{S}$ includes:

- Internal state of each robot (e.g., memory, battery level)
- Physical state of each robot (e.g., mechanics)
- Communication topology

Information Flow in a Swarm Machine



$\mathbb{S}$ includes:

- Internal state of each robot (e.g., memory, battery level)
- Physical state of each robot (e.g., mechanics)
- Communication topology
- Relevant environment state (e.g., stigmergy)

A Swarm is an “Open” Machine

The programmer can control only part of the state

A Swarm is an “Open” Machine

The programmer can control only part of the state

- Environment is dynamic and (partially) unknown

A Swarm is an “Open” Machine

The programmer can control only part of the state

- Environment is dynamic and (partially) unknown
- Swarm state is dynamic and (partially) unknown

A Swarm is an “Open” Machine

The programmer can control only part of the state

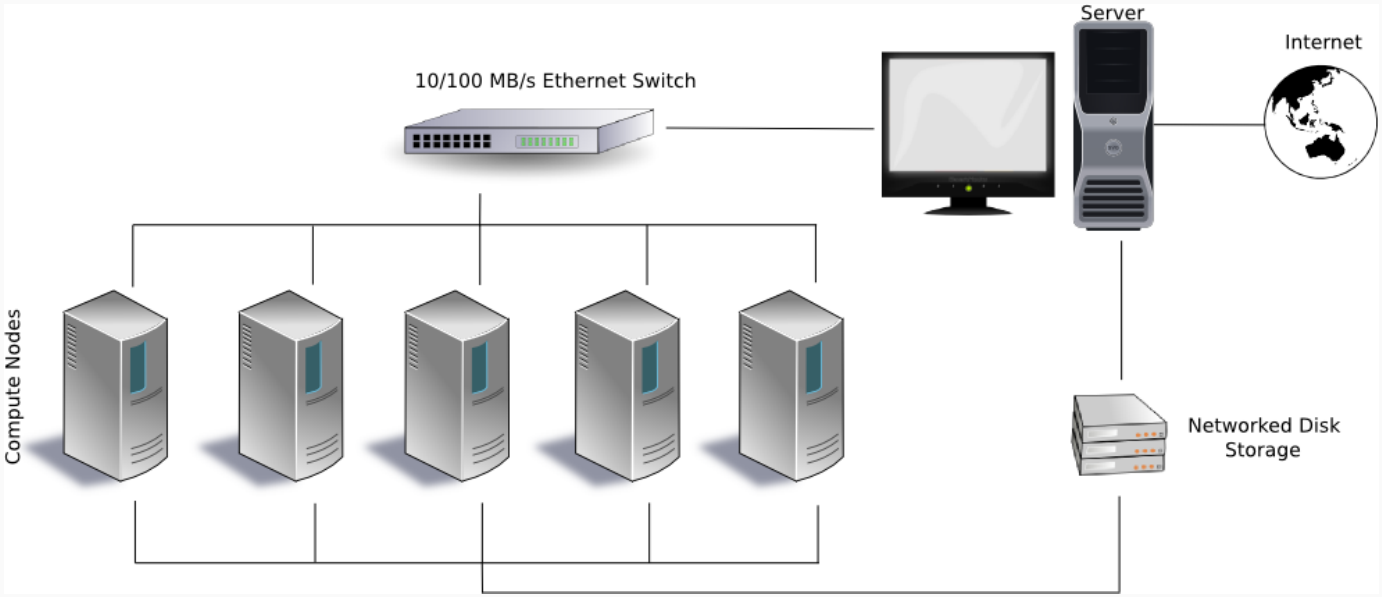
- Environment is dynamic and (partially) unknown
- Swarm state is dynamic and (partially) unknown
- Communication issues are likely

A Swarm is an "Open" Machine

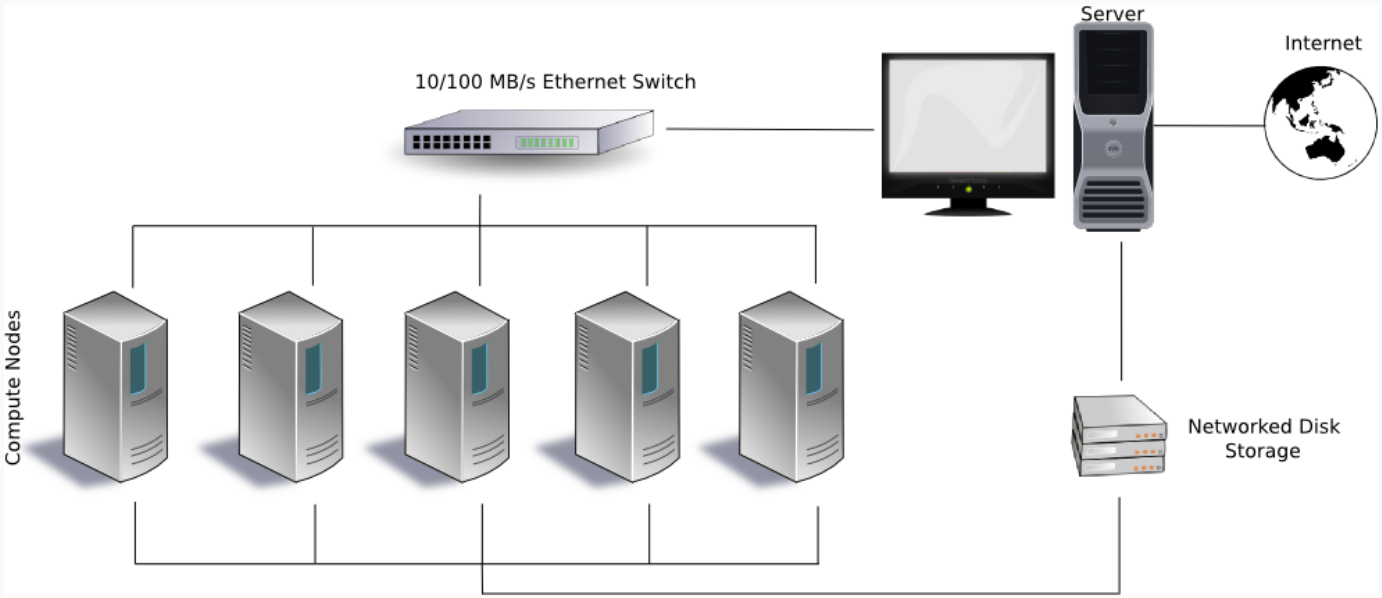
The programmer can control only part of the state

- Environment is dynamic and (partially) unknown
- Swarm state is dynamic and (partially) unknown
- Communication issues are likely
- Individual failures are likely (e.g., battery depletion)

Robot Swarm $\neq$ "Classical" Distributed System

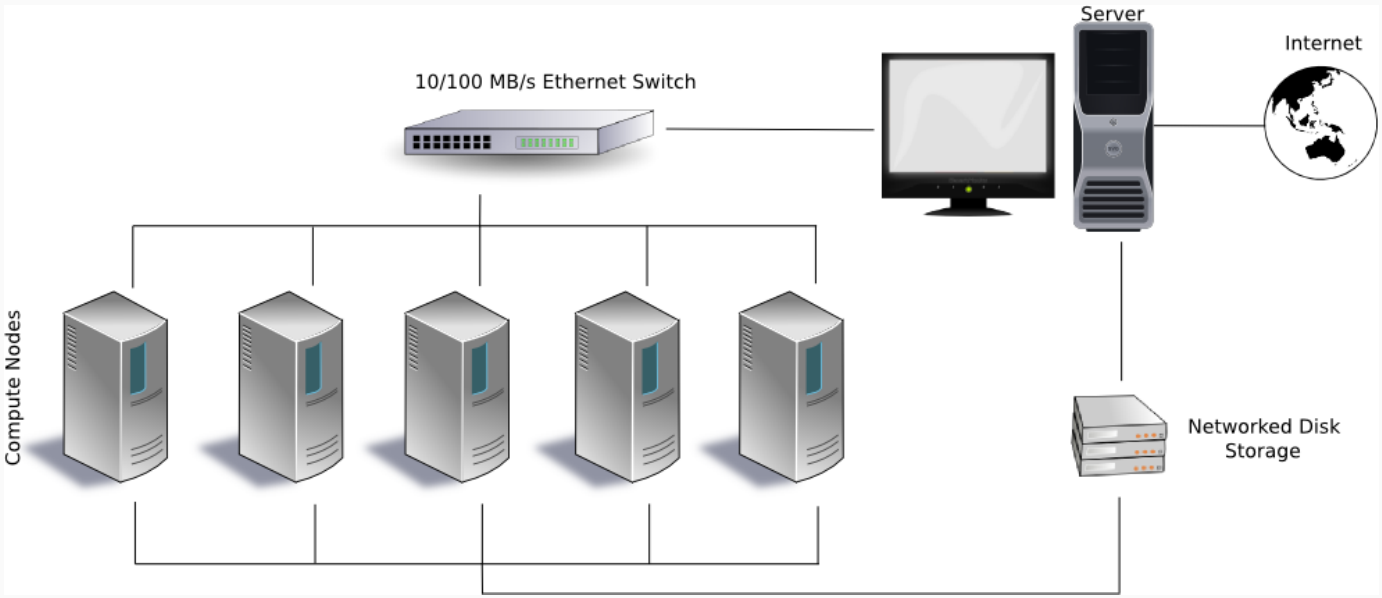


Robot Swarm $\neq$ "Classical" Distributed System



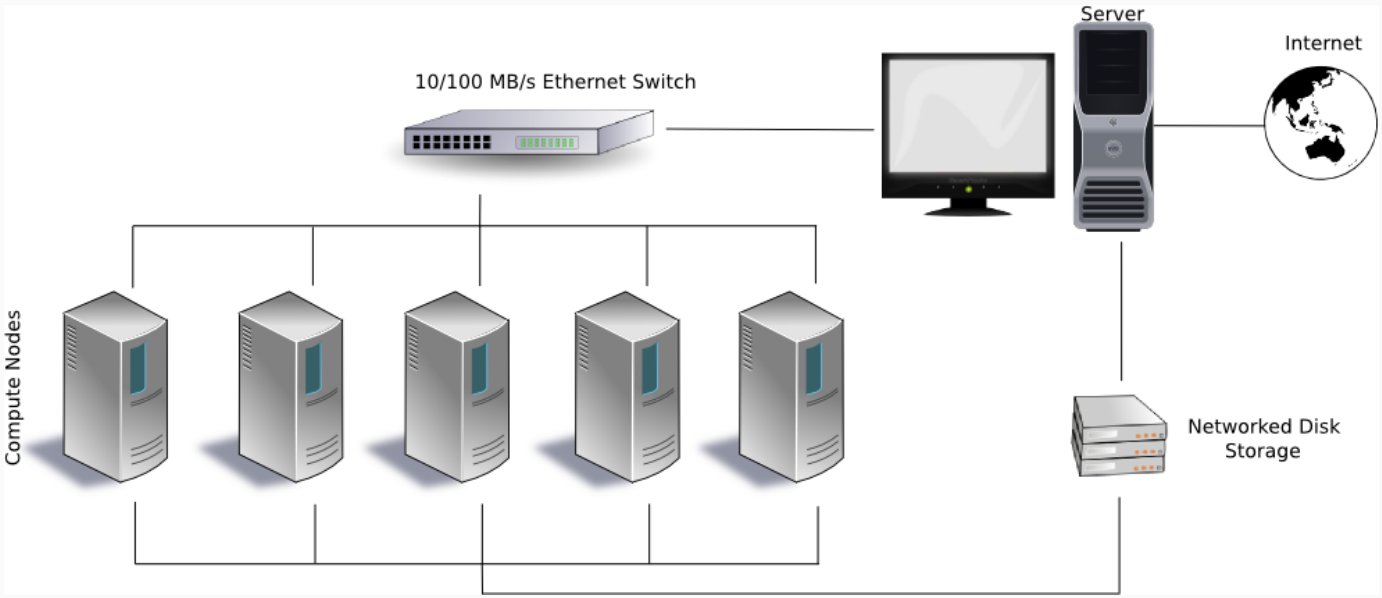
- Static topology

Robot Swarm $\neq$ "Classical" Distributed System



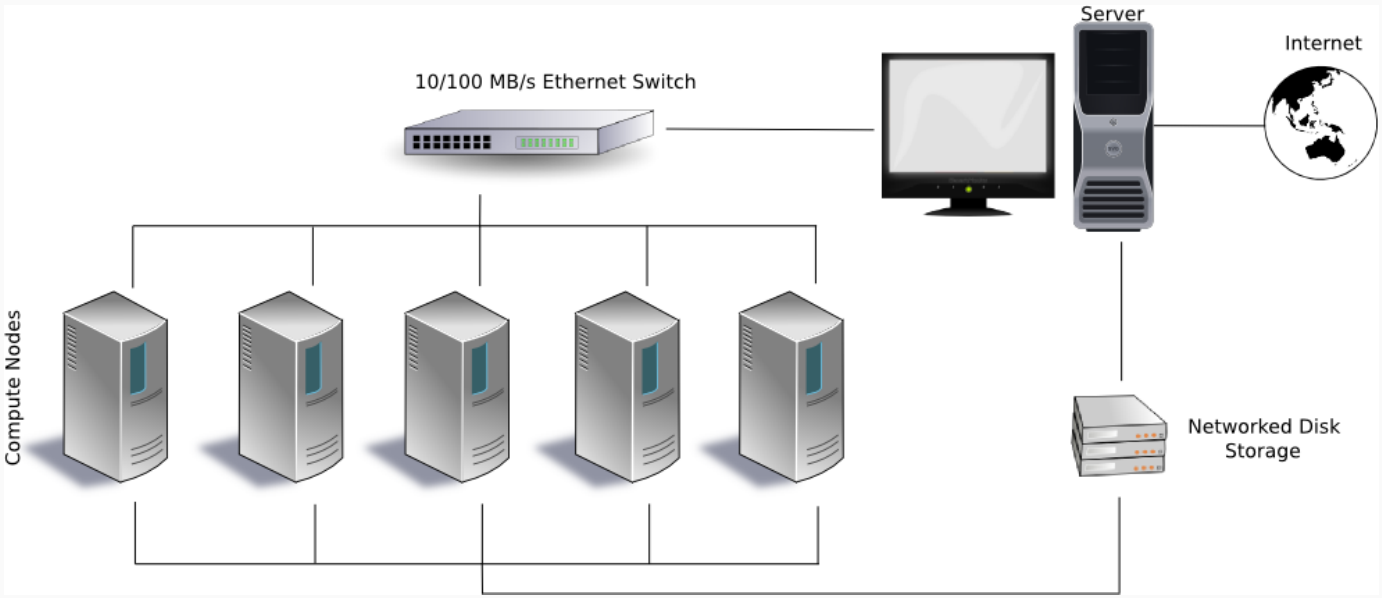
- Static topology
- Reliable, high-bandwidth communication

Robot Swarm $\neq$ "Classical" Distributed System



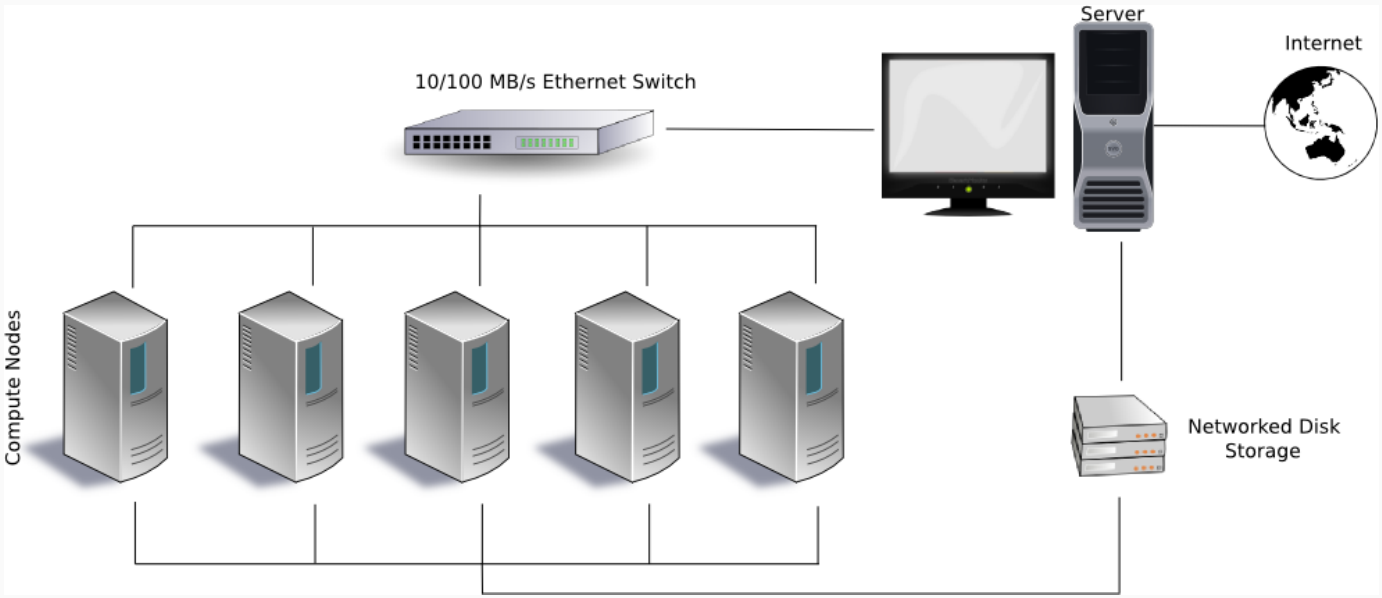
- Static topology
- Reliable, high-bandwidth communication
- Rare failures

Robot Swarm $\neq$ "Classical" Distributed System



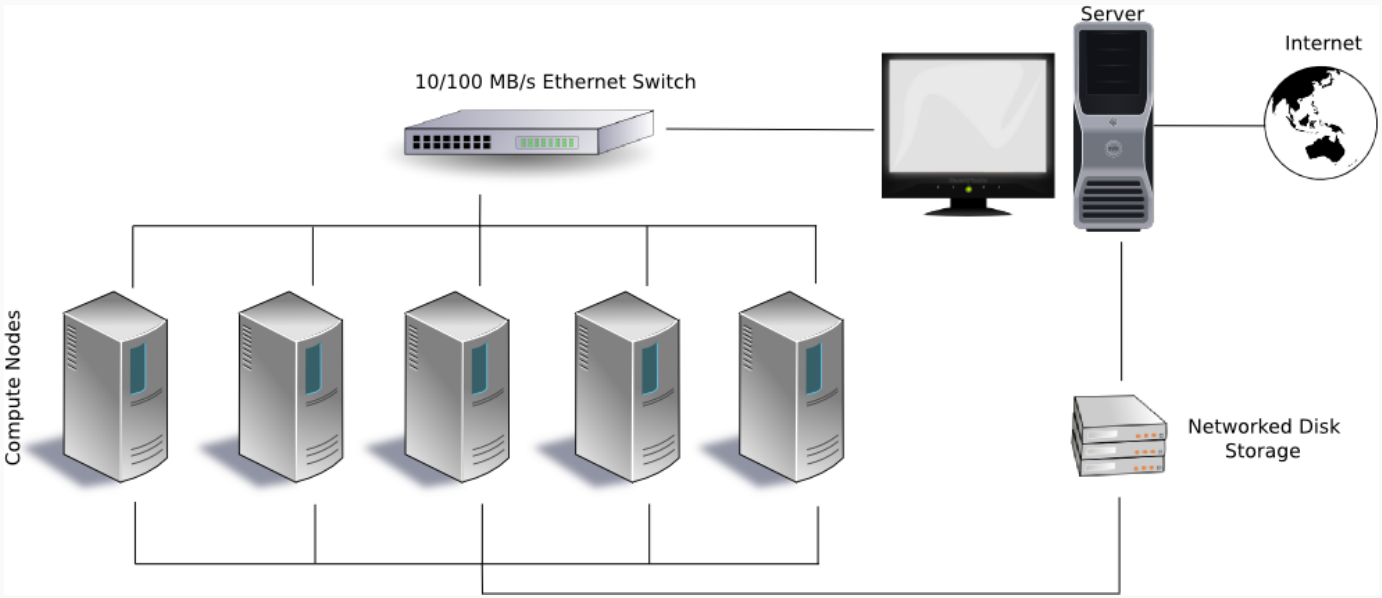
- Static topology
- Reliable, high-bandwidth communication
- Rare failures
- "Closed" machine

Robot Swarm $\neq$ "Classical" Distributed System



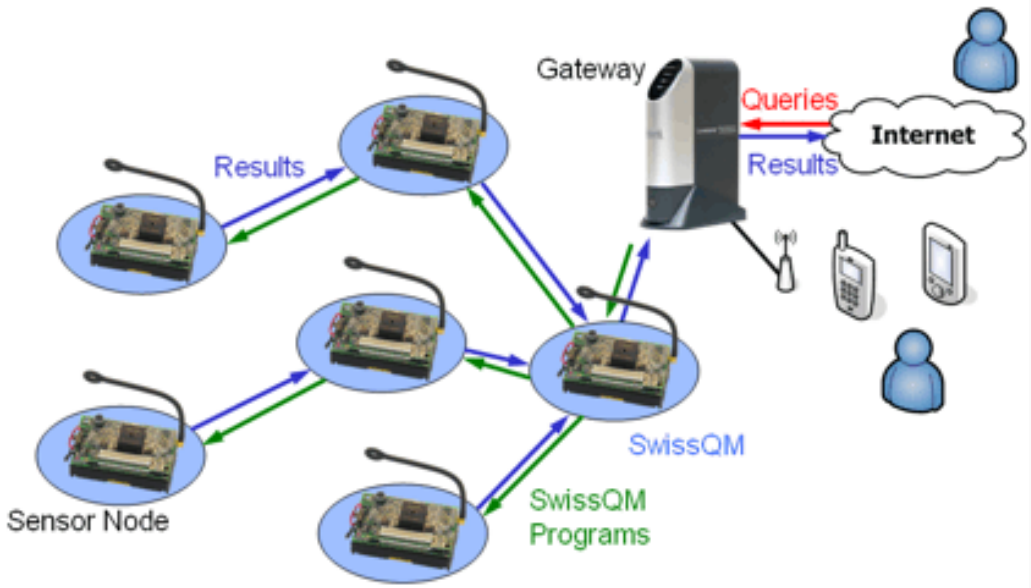
- Static topology
- Reliable, high-bandwidth communication
- Rare failures
- "Closed" machine
  - Only system itself affects state

Robot Swarm $\neq$ "Classical" Distributed System

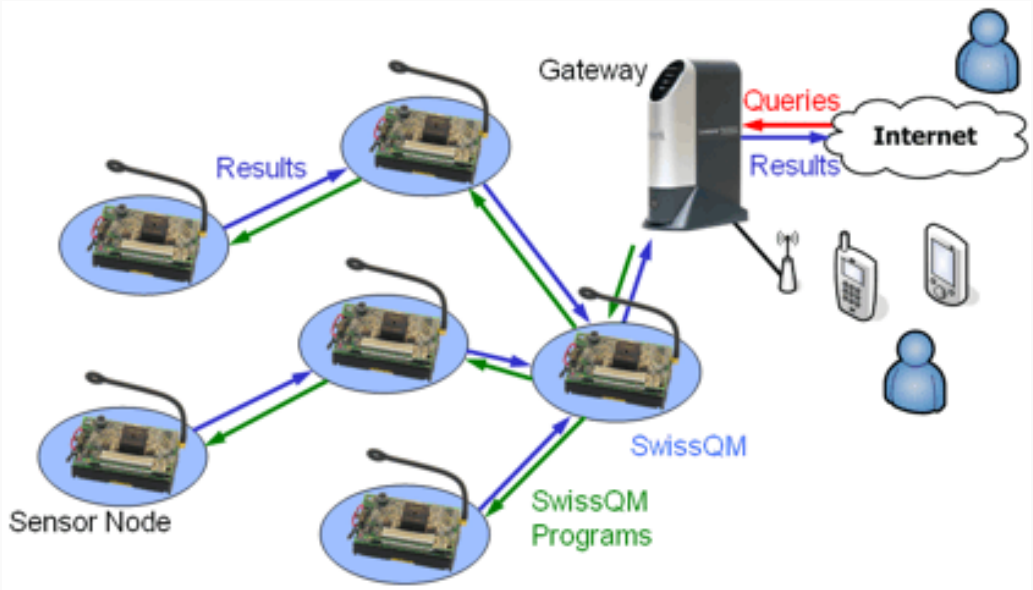


- Static topology
- Reliable, high-bandwidth communication
- Rare failures
- "Closed" machine
  - Only system itself affects state
  - Programmer has everything under control

Robot Swarm $\neq$ (Mobile) Sensor Network

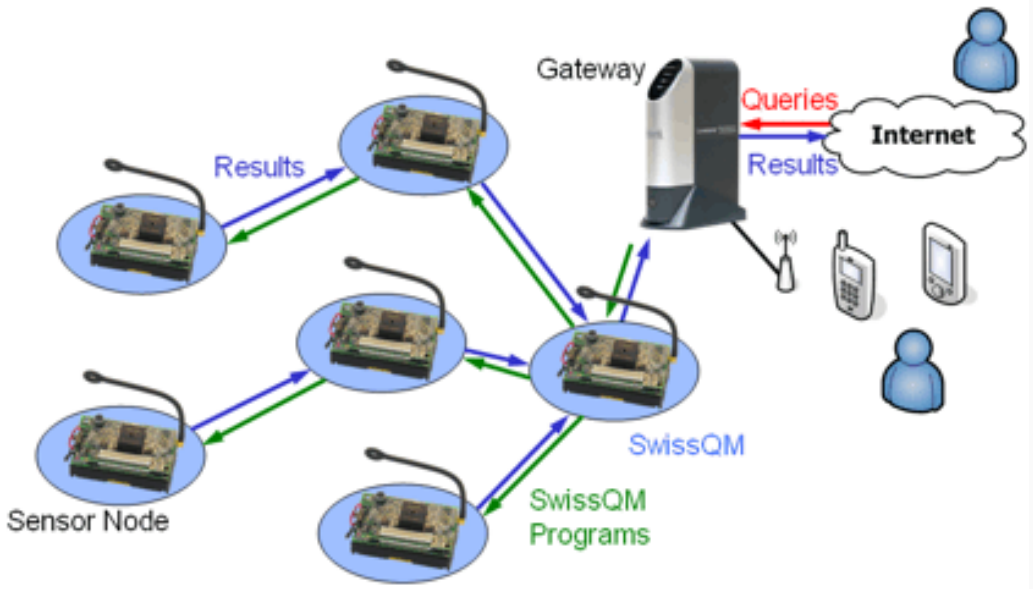


Robot Swarm $\neq$ (Mobile) Sensor Network



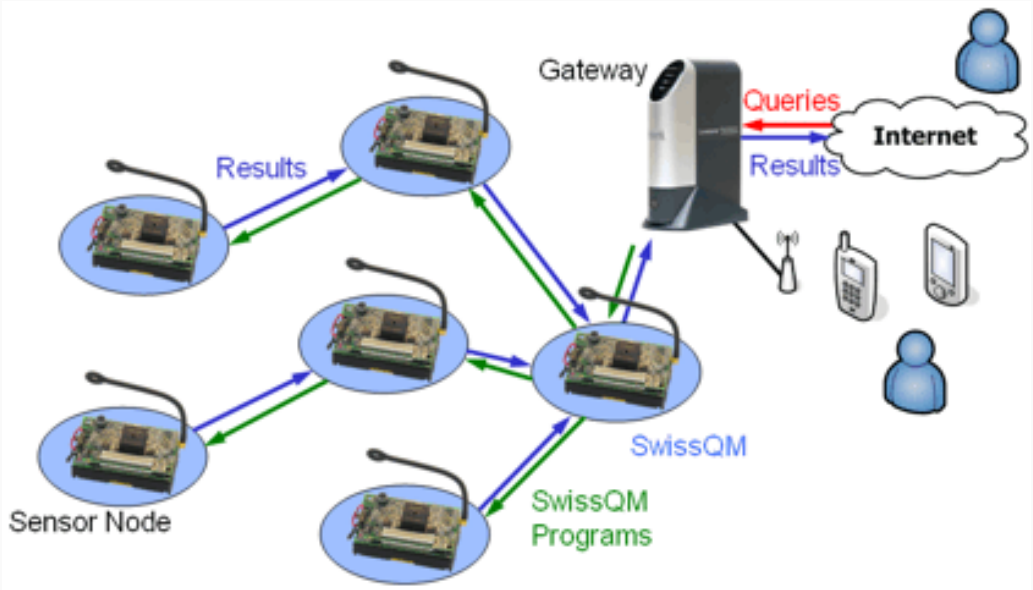
- Focus on data collection and aggregation

Robot Swarm $\neq$ (Mobile) Sensor Network



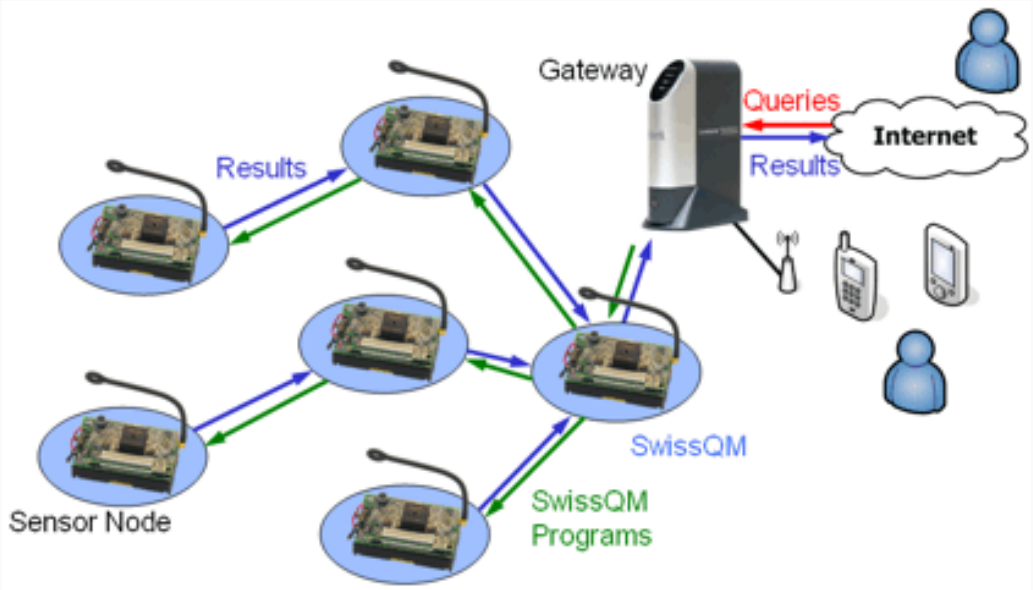
- Focus on data collection and aggregation
- No actuation

Robot Swarm $\neq$ (Mobile) Sensor Network



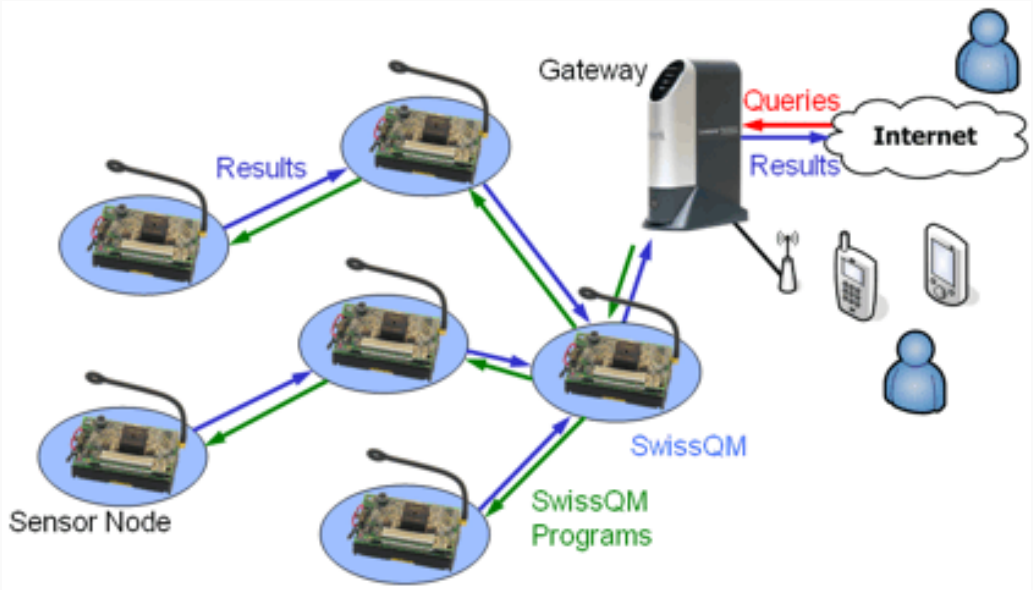
- Focus on data collection and aggregation
- No actuation
- "Ajar" machine

Robot Swarm $\neq$ (Mobile) Sensor Network



- Focus on data collection and aggregation
- No actuation
- "Ajar" machine
  - Environment has an unknown dynamics

Robot Swarm $\neq$ (Mobile) Sensor Network



- Focus on data collection and aggregation
- No actuation
- "Ajar" machine
  - Environment has an unknown dynamics
  - Programmer has part of logic under control, read-only

What Programming Paradigm for Swarm?

What Programming Paradigm for Swarm?

Required Language Features

- Composability
- Predictability
- Support for heterogeneous swarms

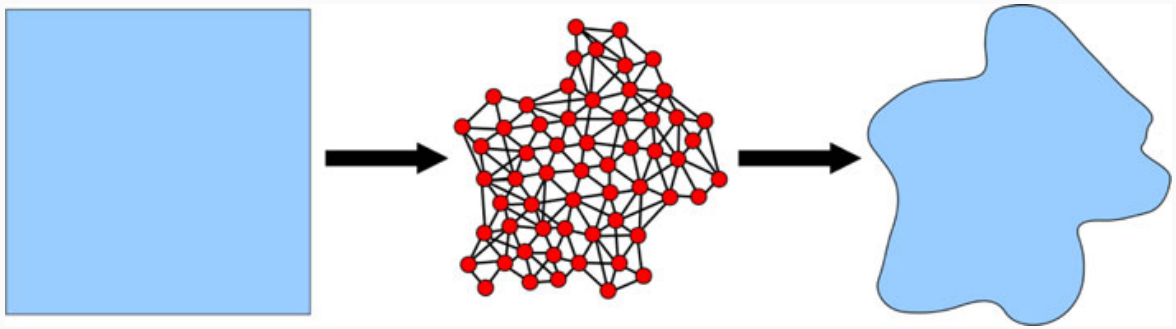
What Programming Paradigm for Swarm?

Robot-Oriented

- Focus on individual robots and their interactions
- C/C++/Python/JavaScript + ROS/MAVLink/...
- GOOD: Maximum level of control
- BAD: Overwhelming amount of detail

What Programming Paradigm for Swarm?

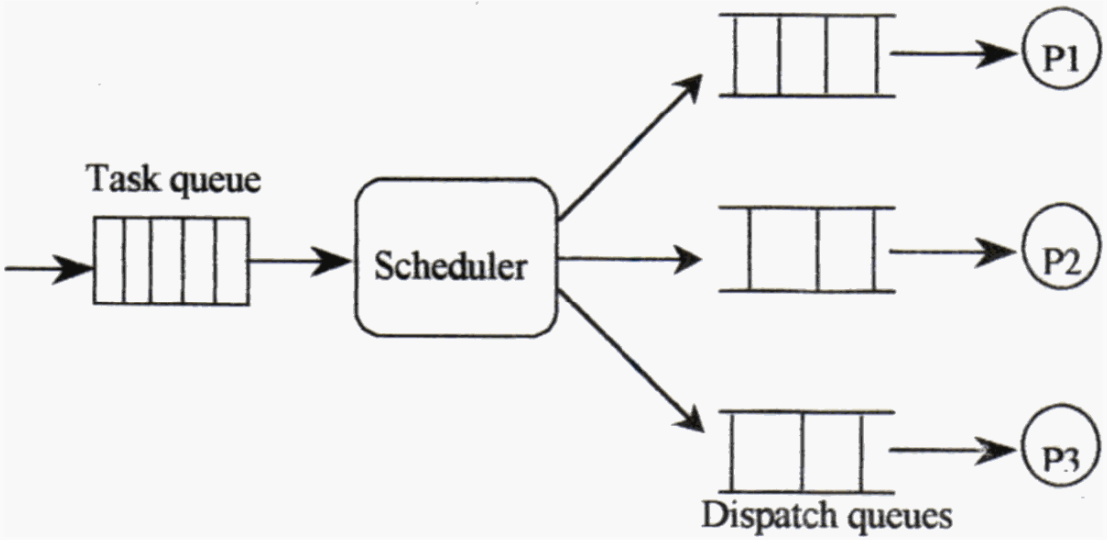
Spatial Computing/Computational Fields



- The swarm is a spatially continuous entity, each point associated to a tuple
- Proto/Protelis
- GOOD: Simple abstraction for space-time computation
- BAD: Concept of "robot" lost, no actuation, no heterogeneity

What Programming Paradigm for Swarm?

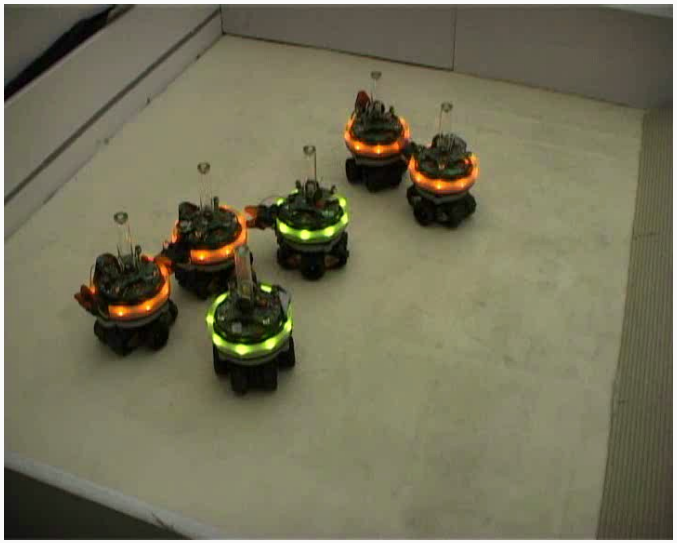
Task-Oriented



- Specify what to do for every task, scheduling is transparent
- Karma/Voltron
- GOOD: Well suited for static SRST scenarios
- BAD: Dynamic MR tasks not supported

What Programming Paradigm for Swarm?

Goal-Oriented



- Specify goal state rather than logic to achieve it
- SWARMMORPH, Termes
- GOOD: Well suited for spatially organizing behaviors
- BAD: Limited control in case of failures

Buzz: Multi-Paradigm Swarm Programming

Buzz at a Glance

Pinciroli, Beltrame. 2016. **Buzz: A Programming Language for Robot Swarms**. *IEEE Software*.

Pinciroli, Beltrame. 2016. **Buzz: An Extensible Programming Language for Self-Organizing Heterogeneous Robot Swarms**. *IROS 2016*.

Buzz at a Glance

Multiple Paradigms

Pinciroli, Beltrame. 2016. **Buzz: A Programming Language for Robot Swarms**. *IEEE Software*.

Pinciroli, Beltrame. 2016. **Buzz: An Extensible Programming Language for Self-Organizing Heterogeneous Robot Swarms**. *IROS 2016*.

Buzz at a Glance

Multiple Paradigms

- Robot-oriented primitives

Pinciroli, Beltrame. 2016. **Buzz: A Programming Language for Robot Swarms**. *IEEE Software*.

Pinciroli, Beltrame. 2016. **Buzz: An Extensible Programming Language for Self-Organizing Heterogeneous Robot Swarms**. *IROS 2016*.

Buzz at a Glance

Multiple Paradigms

- Robot-oriented primitives
- Swarm-oriented primitives

Pinciroli, Beltrame. 2016. **Buzz: A Programming Language for Robot Swarms**. *IEEE Software*.

Pinciroli, Beltrame. 2016. **Buzz: An Extensible Programming Language for Self-Organizing Heterogeneous Robot Swarms**. *IROS 2016*.

Buzz at a Glance

Multiple Paradigms

- Robot-oriented primitives
- Swarm-oriented primitives
- Neighbor management primitives (spatial computing)

Pinciroli, Beltrame. 2016. **Buzz: A Programming Language for Robot Swarms**. *IEEE Software*.

Pinciroli, Beltrame. 2016. **Buzz: An Extensible Programming Language for Self-Organizing Heterogeneous Robot Swarms**. *IROS 2016*.

Buzz at a Glance

Multiple Paradigms

- Robot-oriented primitives
- Swarm-oriented primitives
- Neighbor management primitives (spatial computing)
- Global consensus primitives (computational fields)

Pinciroli, Beltrame. 2016. **Buzz: A Programming Language for Robot Swarms**. *IEEE Software*.

Pinciroli, Beltrame. 2016. **Buzz: An Extensible Programming Language for Self-Organizing Heterogeneous Robot Swarms**. *IROS 2016*.

Buzz at a Glance

Multiple Paradigms

- Robot-oriented primitives
- Swarm-oriented primitives
- Neighbor management primitives (spatial computing)
- Global consensus primitives (computational fields)

Composability / Predictability

Pinciroli, Beltrame. 2016. **Buzz: A Programming Language for Robot Swarms**. *IEEE Software*.

Pinciroli, Beltrame. 2016. **Buzz: An Extensible Programming Language for Self-Organizing Heterogeneous Robot Swarms**. *IROS 2016*.

Buzz at a Glance

Multiple Paradigms

- Robot-oriented primitives
- Swarm-oriented primitives
- Neighbor management primitives (spatial computing)
- Global consensus primitives (computational fields)

Composability / Predictability

- Imperative/functional syntax (JavaScript, Lua, Python)

Pinciroli, Beltrame. 2016. **Buzz: A Programming Language for Robot Swarms**. *IEEE Software*.

Pinciroli, Beltrame. 2016. **Buzz: An Extensible Programming Language for Self-Organizing Heterogeneous Robot Swarms**. *IROS 2016*.

Buzz at a Glance

Multiple Paradigms

- Robot-oriented primitives
- Swarm-oriented primitives
- Neighbor management primitives (spatial computing)
- Global consensus primitives (computational fields)

Composability / Predictability

- Imperative/functional syntax (JavaScript, Lua, Python)

Support for Heterogeneous Swarms

Pinciroli, Beltrame. 2016. **Buzz: A Programming Language for Robot Swarms**. *IEEE Software*.

Pinciroli, Beltrame. 2016. **Buzz: An Extensible Programming Language for Self-Organizing Heterogeneous Robot Swarms**. *IROS 2016*.

Buzz at a Glance

Multiple Paradigms

- Robot-oriented primitives
- Swarm-oriented primitives
- Neighbor management primitives (spatial computing)
- Global consensus primitives (computational fields)

Composability / Predictability

- Imperative/functional syntax (JavaScript, Lua, Python)

Support for Heterogeneous Swarms

- Possibility to extend the language

Pinciroli, Beltrame. 2016. **Buzz: A Programming Language for Robot Swarms**. *IEEE Software*.

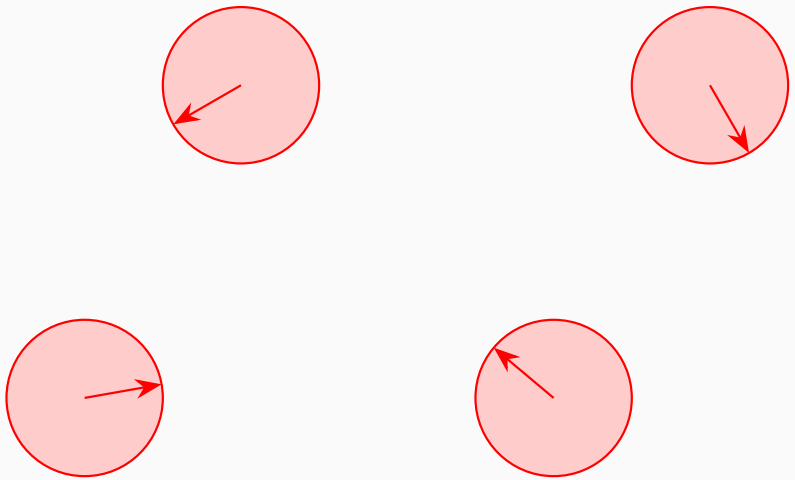
Pinciroli, Beltrame. 2016. **Buzz: An Extensible Programming Language for Self-Organizing Heterogeneous Robot Swarms**. *IROS 2016*.

Reference Model: Execution

Discrete swarm, step-wise execution

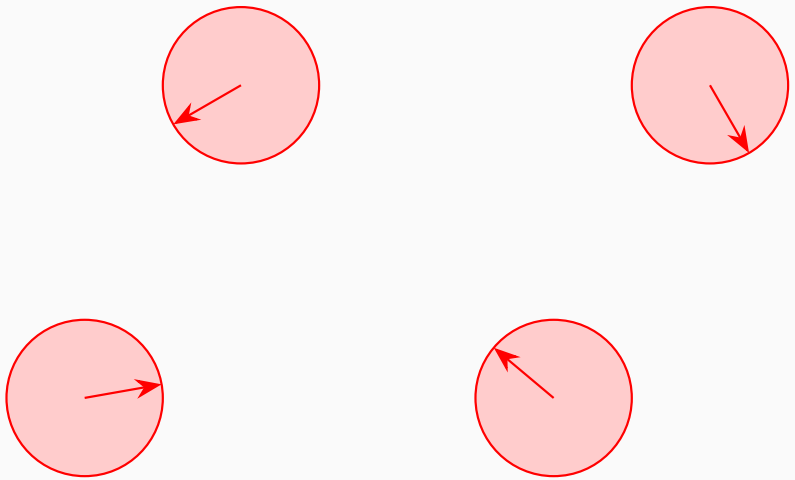
Reference Model: Execution

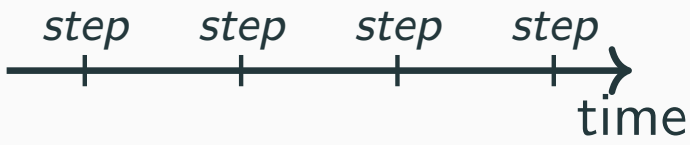
Discrete swarm, step-wise execution



Reference Model: Execution

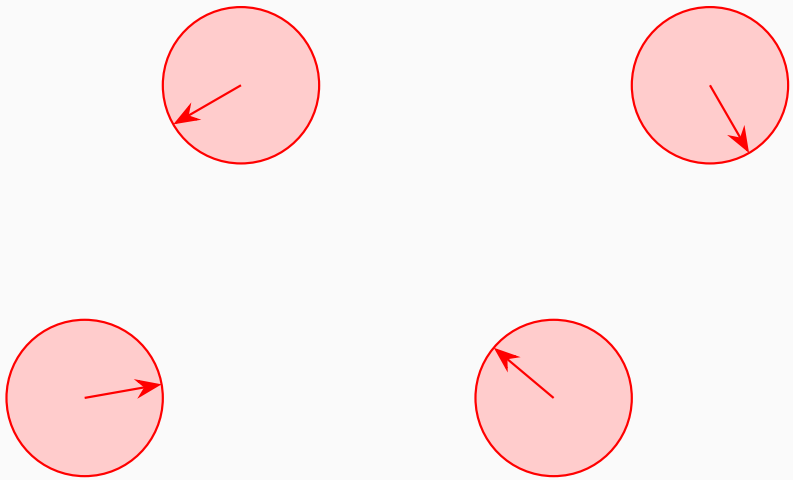
Discrete swarm, step-wise execution

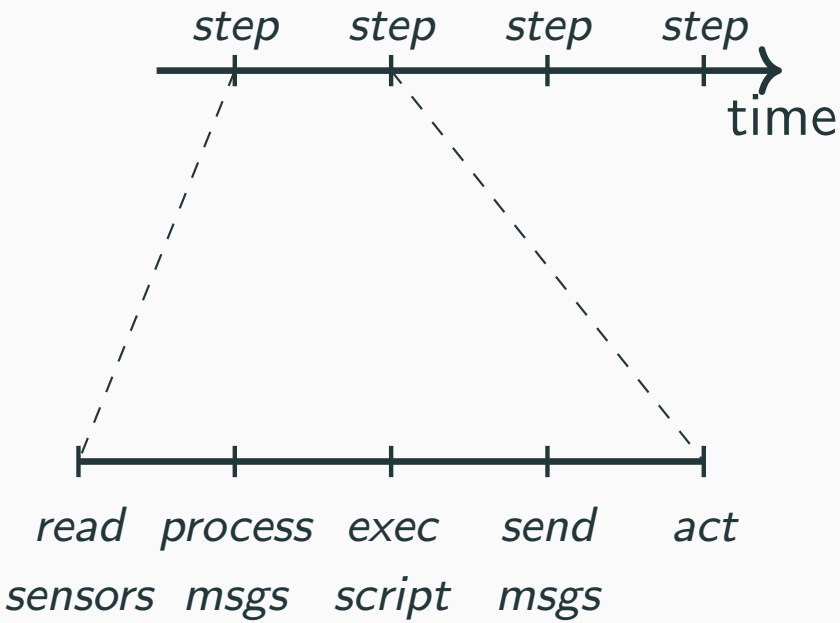




Reference Model: Execution

Discrete swarm, step-wise execution





Reference Model: Communication

Situated communication =

Local data broadcast + Source position data

Reference Model: Communication

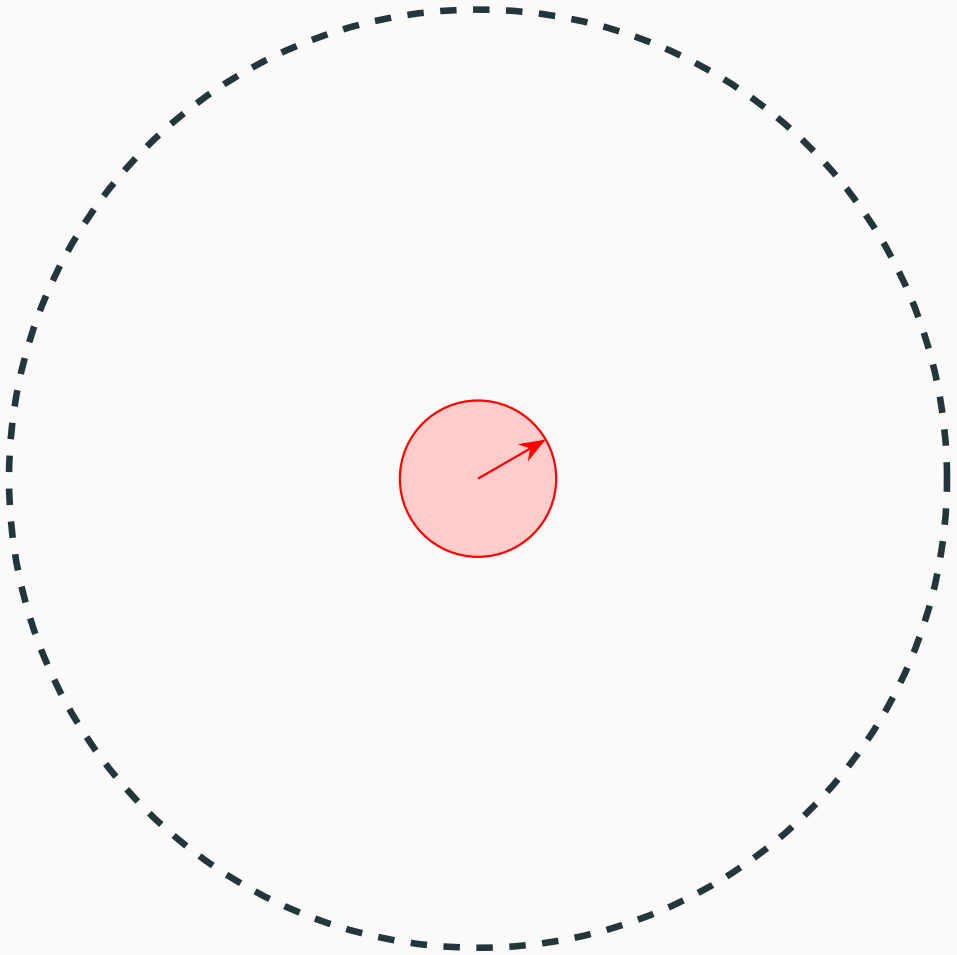
Situated communication =

Local data broadcast + Source position data

Reference Model: Communication

Situated communication =

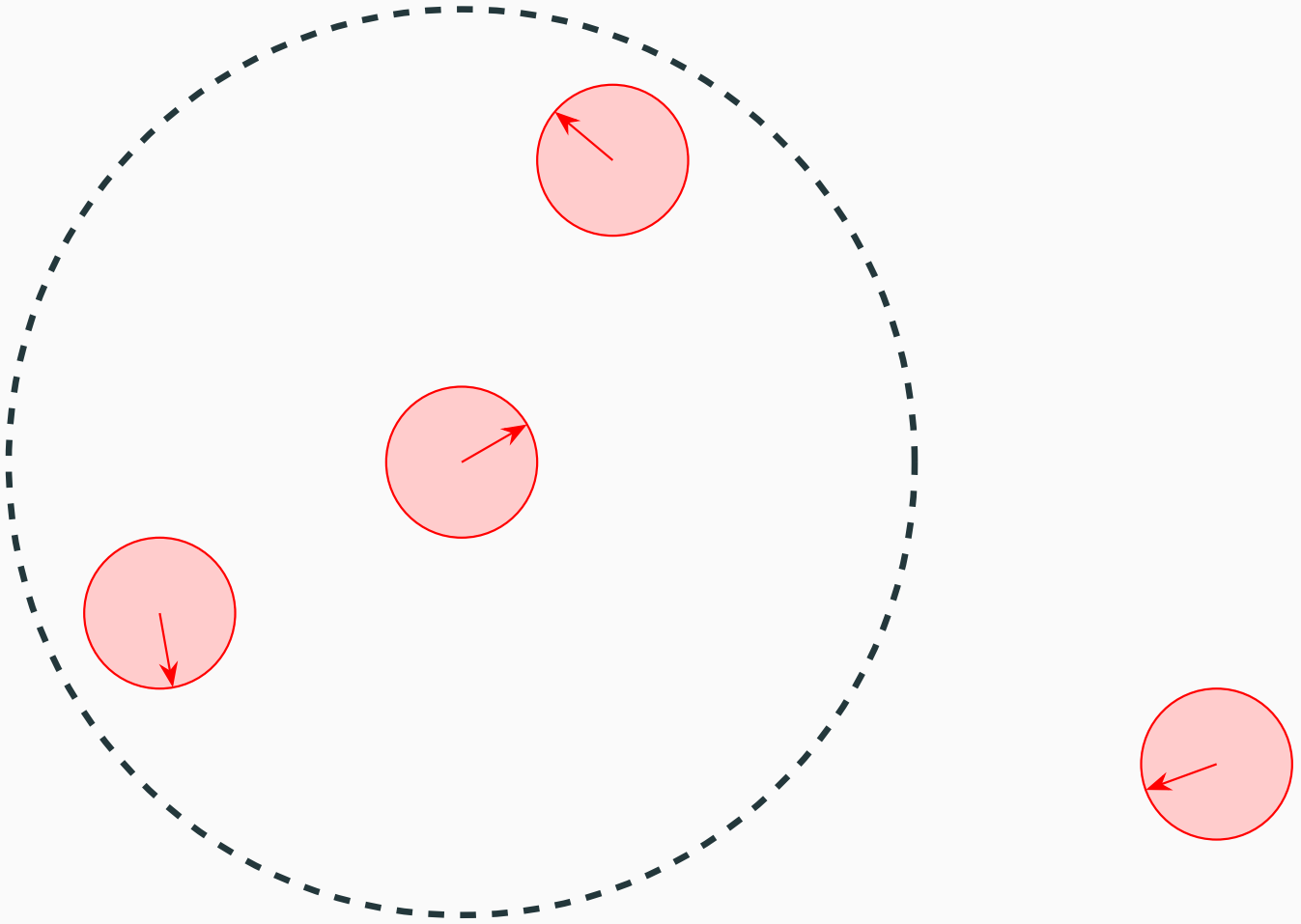
Local data broadcast + Source position data



Reference Model: Communication

Situated communication =

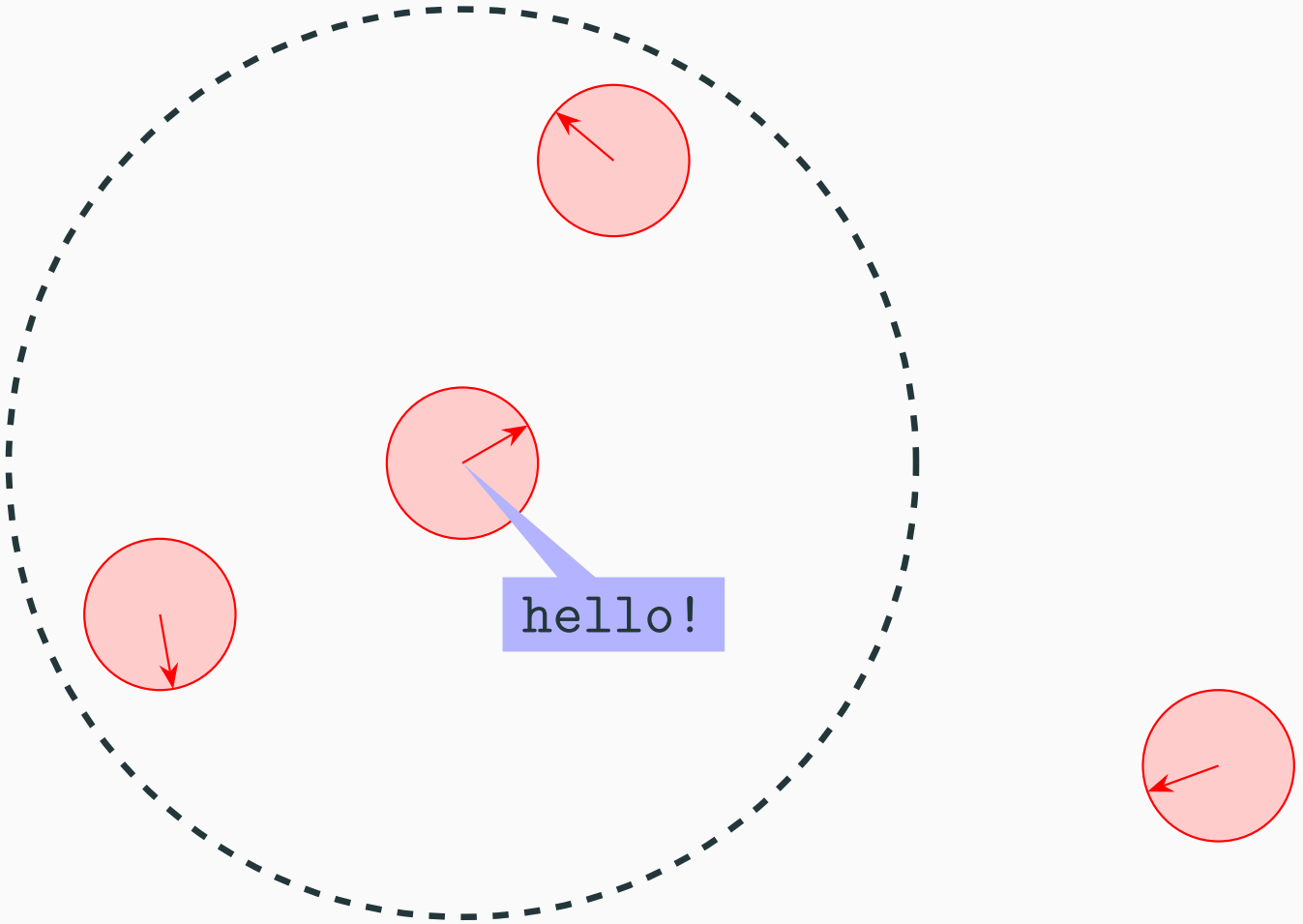
Local data broadcast + Source position data



Reference Model: Communication

Situated communication =

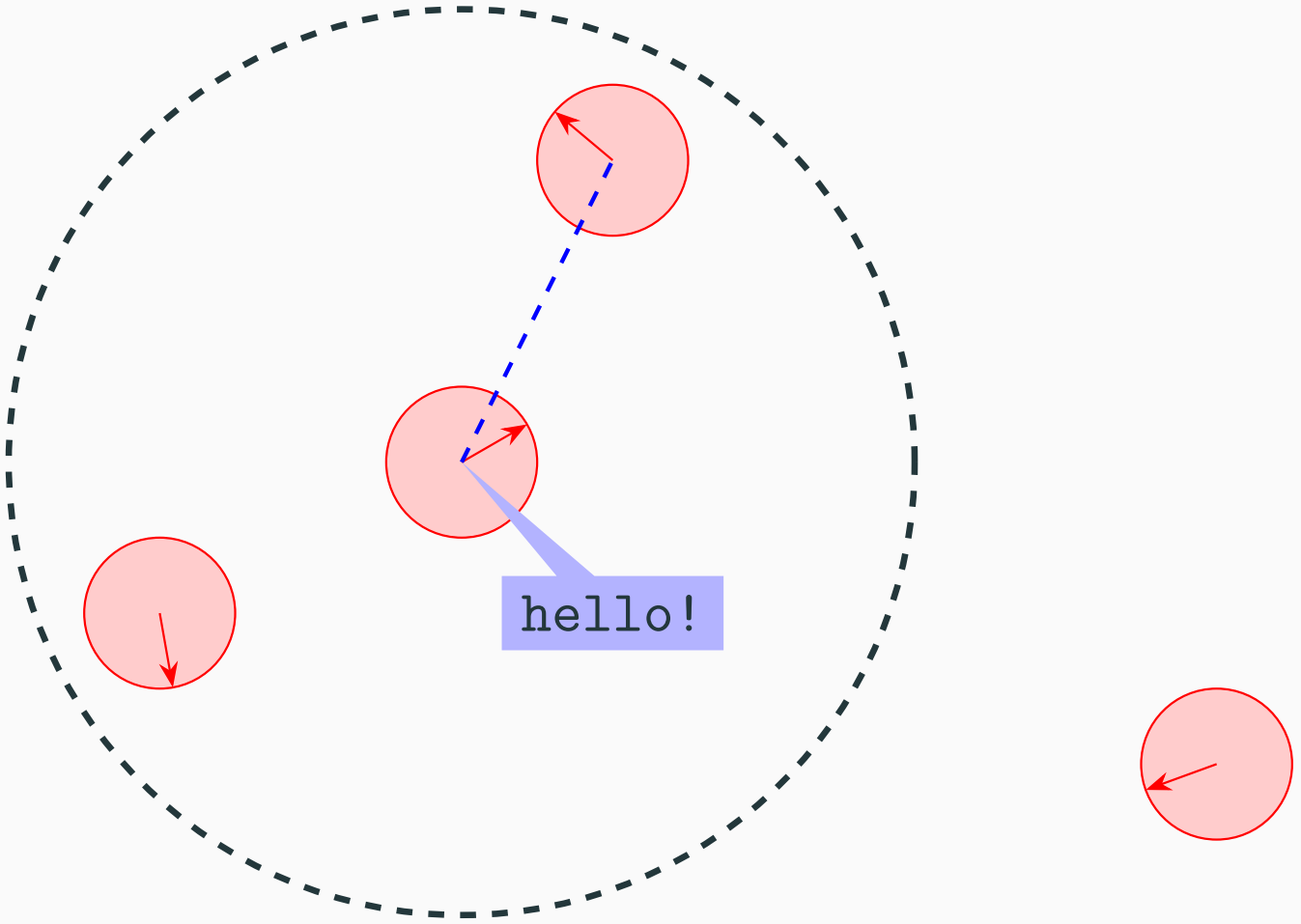
Local data broadcast + Source position data



Reference Model: Communication

Situated communication =

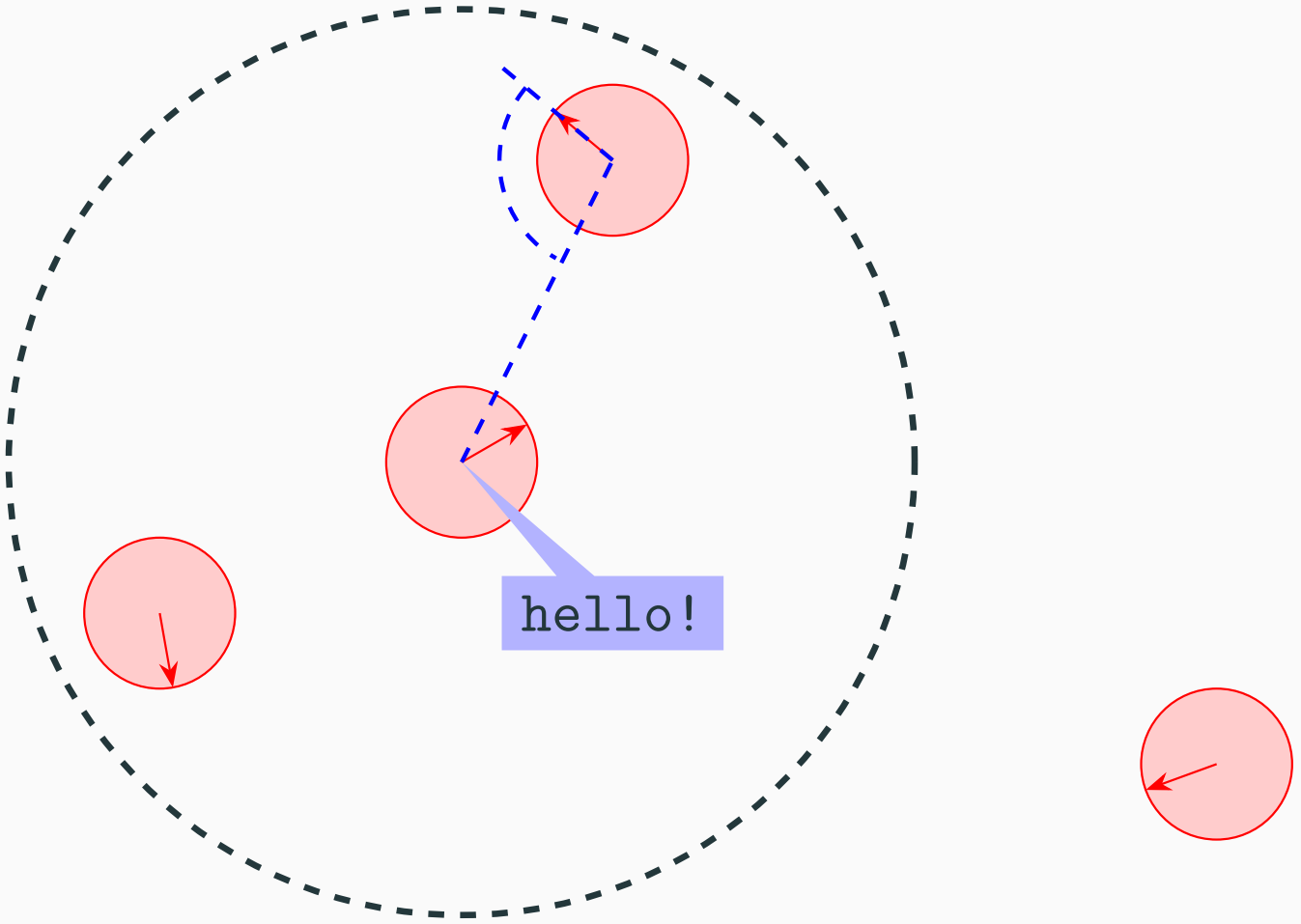
Local data broadcast + Source position data



Reference Model: Communication

Situated communication =

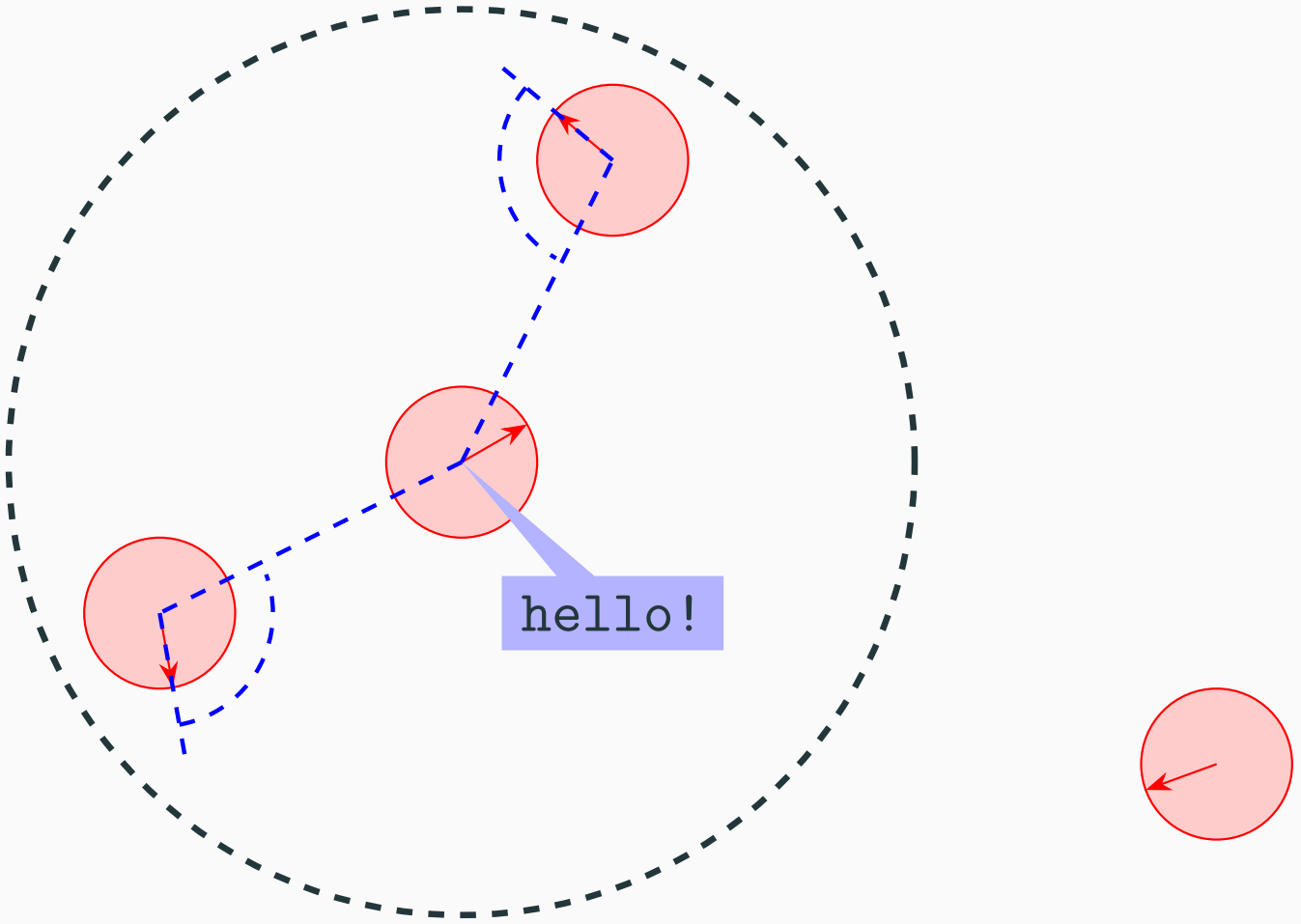
Local data broadcast + Source position data



Reference Model: Communication

Situated communication =

Local data broadcast + Source position data



Robot-wise Operations

Assignment, Looping, Branching

```
# Assignment and arithmetic operation
a = 3 + 7

# Looping
i = 0
while(i < a)
  i = i + 1

# Branching
if(a == 10)
  i = 0
```

Robot-wise Operations

Tables

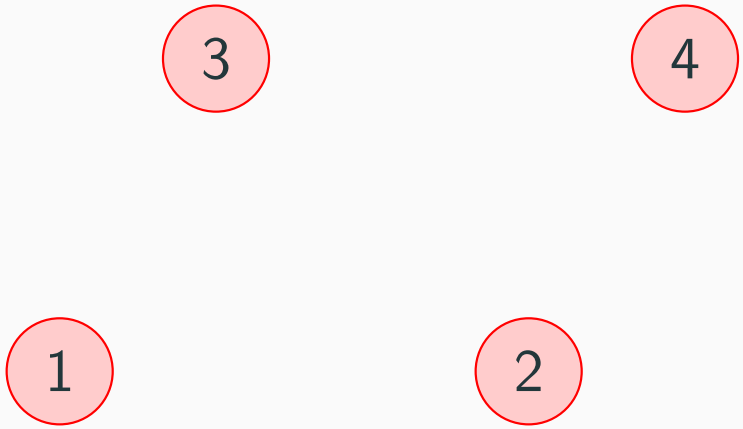
```
# Create an empty table
t = {}

# Index syntax, use as array
t[6] = 5

# Dot syntax, use as dictionary
t.b = 9

# Index syntax, use as dictionary
t["b"] = 10
```

Robot-wise Operations

Functions and Lambdas

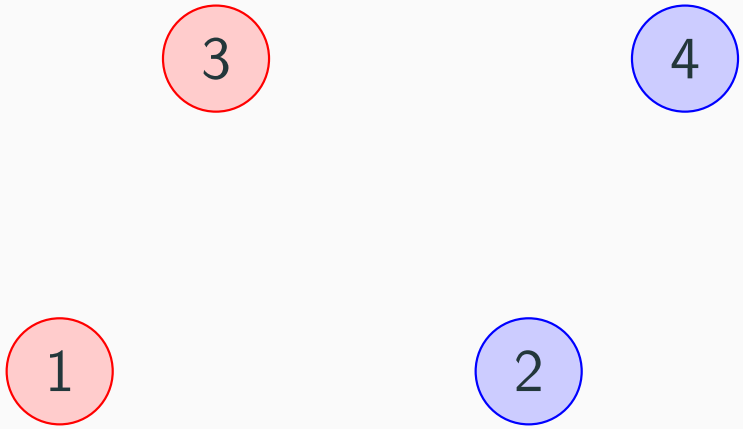
```
# Function definition
function f(a) {
  return a
}

# Function call
x = f(9)

# Closures can be defined anonymously (a.k.a. lambdas)
l = function(a,b) {
  return a+b
}
x = l(2,3) # x is set to 5
```

Creating a Swarm



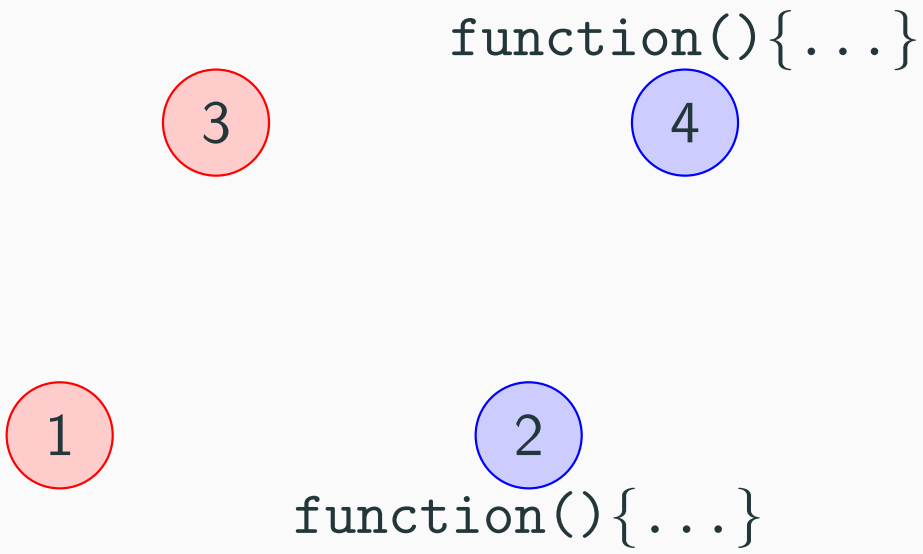

```
# Creation of a swarm with identifier 1
s = swarm.create(1)
```

Populating a Swarm



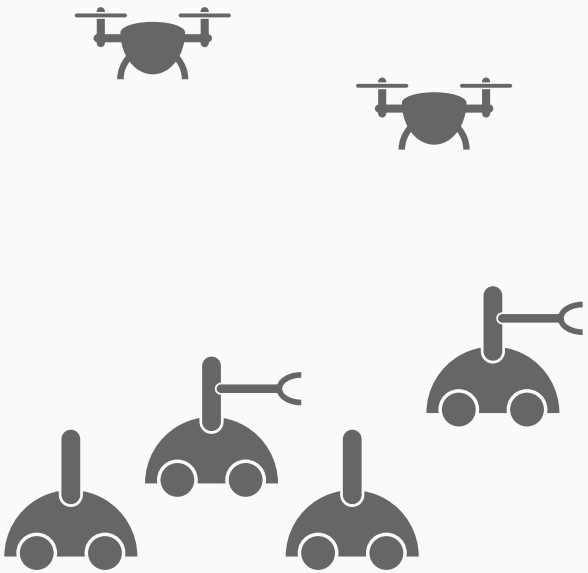

```
# Join the swarm if the robot identifier (id) is even
s.select(id % 2 == 0)
```

Swarm Task Assignment



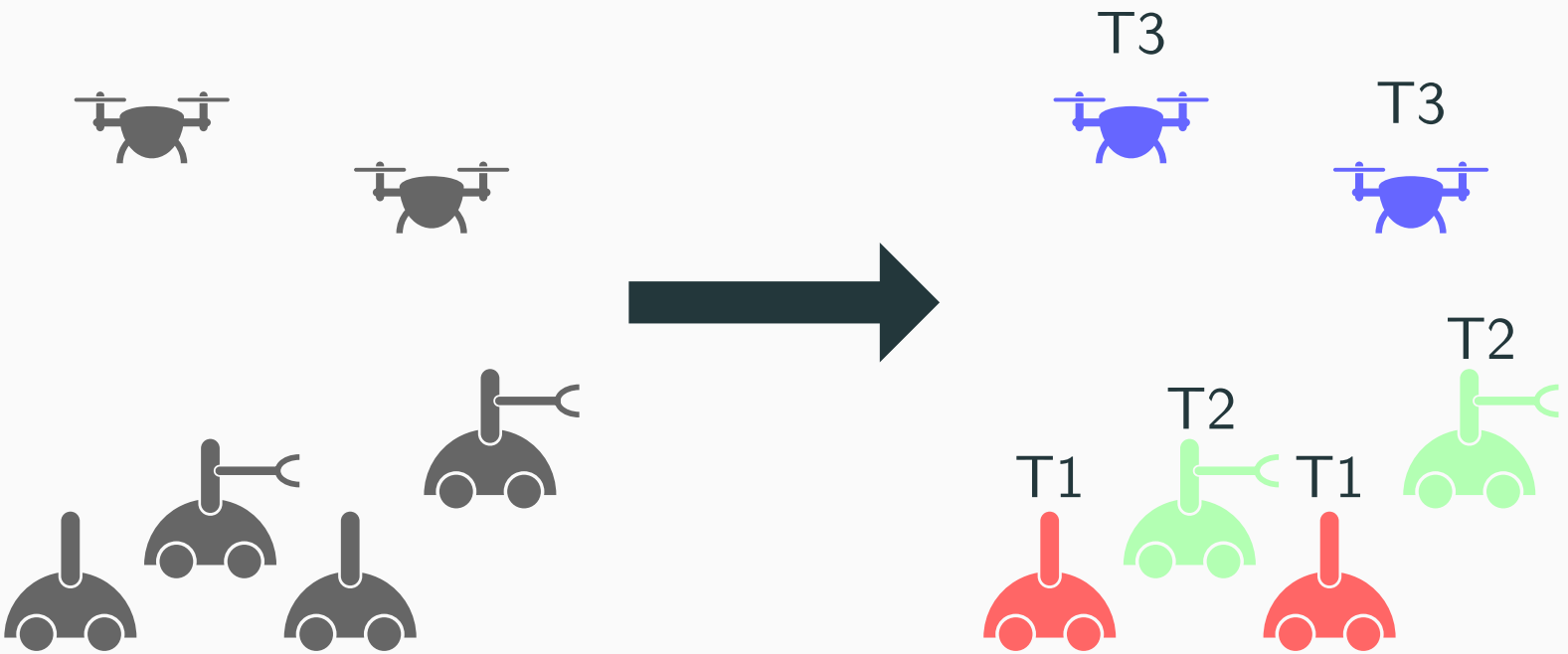

```
# Assigning a task to a swarm
s.exec(function() { ... })
```

Swarm-Oriented Development



Pinciroli, Beltrame. 2016. **Swarm-Oriented Programming of Distributed Robot Networks**. *IEEE Computer*.

Swarm-Oriented Development



Pinciroli, Beltrame. 2016. **Swarm-Oriented Programming of Distributed Robot Networks**. *IEEE Computer*.

Swarm Tables Propagation

Neighbor Management

Neighbor Management

Positional Data

The `neighbors` structure stores the relative positions of the neighbors of a robot (distance, azimuth, elevation)

Neighbor Management

Positional Data

The `neighbors` structure stores the relative positions of the neighbors of a robot (distance, azimuth, elevation)

Communication

The `neighbors` structure allows robots to query other robots, aggregate, and broadcast information.

Neighbor Communication

Neighbor Communication

Broadcast: `neighbors.broadcast("distance", mydist)`

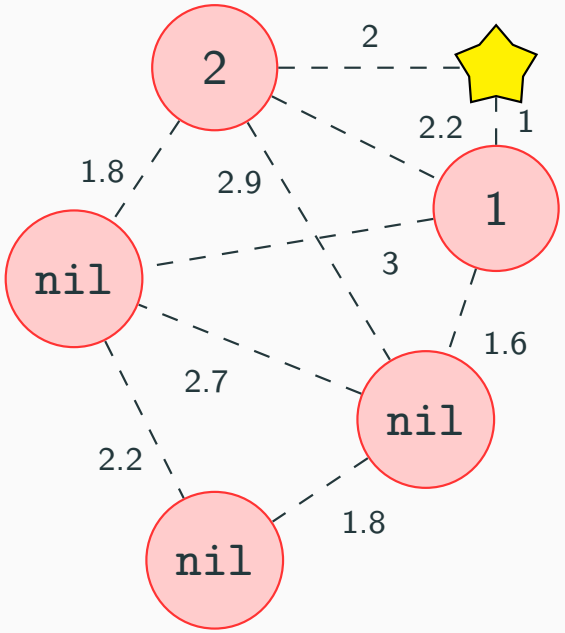
Neighbor Communication

Broadcast: `neighbors.broadcast("distance", mydist)`

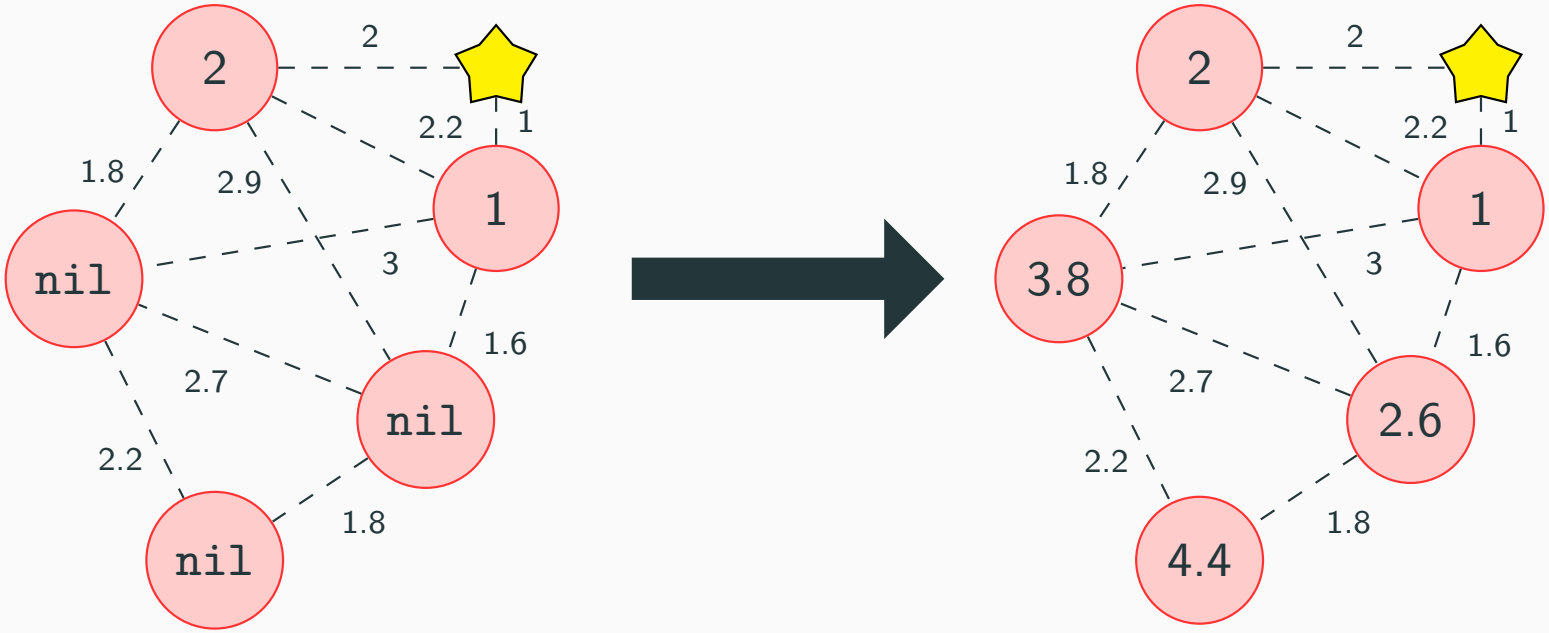
Listening: `neighbors.listen("distance", fun)`

```
# Gradient calculation
neighbors.listen("distance",
  # top: the topic ("distance")
  # nid: neighbor id
  # val: value broadcasted by neighbor
  function(top, val, nid) {
    mydist = math.min(mydist,
                      neighbors.get(nid).distance + val)
  })
```

Neighbor Communication



Neighbor Communication



Neighbor Data Management

Reduction: `x = neighbors.reduce(fun, accum)`

```
# Sum of the vectors connecting a robot to its neighbors (2D)
x = neighbors.reduce(function(rid, data, accum) {
    accum.x = accum.x + data.distance * math.cos(data.azimuth)
    accum.y = accum.y + data.distance * math.sin(data.azimuth)
    return accum
  }, {x=0, y=0})
```

Neighbor Data Management

Transformation: `x = neighbors.map(fun, accum)`

```
# Transformation example
x = neighbors.map(
  function(rid, data) {
    var c = {}
    c.x = data.distance * math.cos(data.elevation) *
          math.cos(data.azimuth)
    c.y = data.distance * math.cos(data.elevation) *
          math.sin(data.azimuth)
    c.z = data.distance * math.sin(data.elevation)
    return c })
```

Neighbor Data Management

Filtering by Data: x = neighbors.filter(fun)

```
# Filtering example
onemeter = neighbors.filter(function(rid, data) {
    # We assume the distance is expressed in centimeters
    return data.distance < 100 })
```

Neighbor Data Management

Filtering by Swarm Membership: `neighbors.kin()` and `neighbors.nonkin()`

```
# Calculate force field due to kin and non-kin robots
function direction() {
  var dir
  dir = neighbors.kin().reduce(force_sum_kin, {x=0,y=0})
  dir = neighbors.nonkin().reduce(force_sum_nonkin, dir)
  dir.x = dir.x / neighbors.count()
  dir.y = dir.y / neighbors.count()
  return dir
}
```

Virtual Stigmergy

Pinciroli, Lee-Brown, Beltrame. 2015. **A Tuple Space for Data Sharing in Robot Swarms**. *BICT 2015*.

Virtual Stigmergy

Natural Inspiration

Stigmergy: indirect, environment-mediated communication modality used by social insect colonies to build their nest and cluster food.

Pinciroli, Lee-Brown, Beltrame. 2015. **A Tuple Space for Data Sharing in Robot Swarms**. *BICT 2015*.

Virtual Stigmergy

Natural Inspiration

Stigmergy: indirect, environment-mediated communication modality used by social insect colonies to build their nest and cluster food.

Buzz Interpretation

Virtual stigmergy: a data structure that allows a robot swarm to agree globally on the value of a set of `(key, value)` pairs.

Pinciroli, Lee-Brown, Beltrame. 2015. **A Tuple Space for Data Sharing in Robot Swarms**. *BICT 2015*.

Virtual Stigmergy

Virtual Stigmergy Creation

```
v = stigmergy.create(1)
```

Virtual Stigmergy Creation

```
v = stigmergy.create(1)
```

Virtual Stigmergy Writing

```
v.put("a", 6)
```

Virtual Stigmergy

Virtual Stigmergy Creation

```
v = stigmergy.create(1)
```

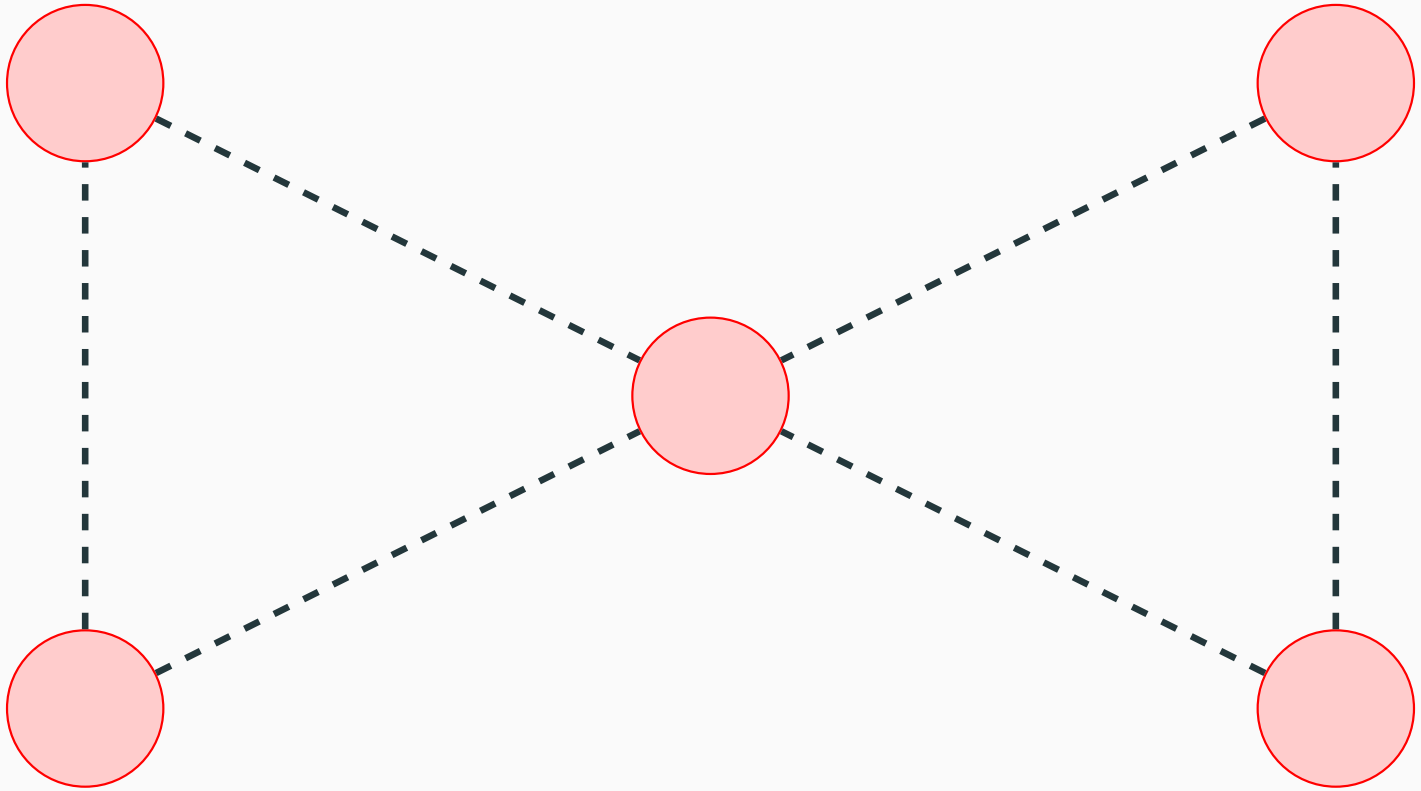
Virtual Stigmergy Writing

```
v.put("a", 6)
```

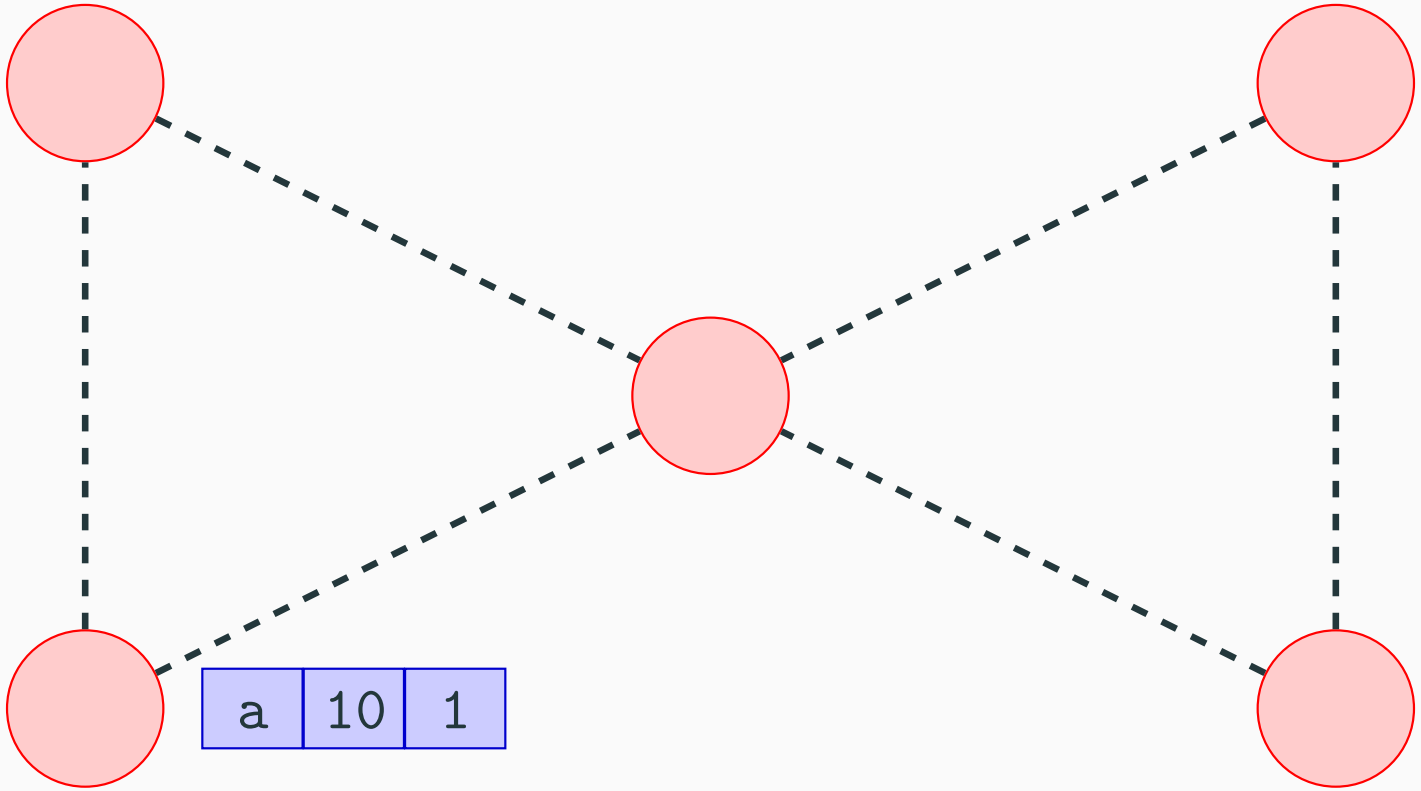
Virtual Stigmergy Reading

```
x = v.get("a")
```

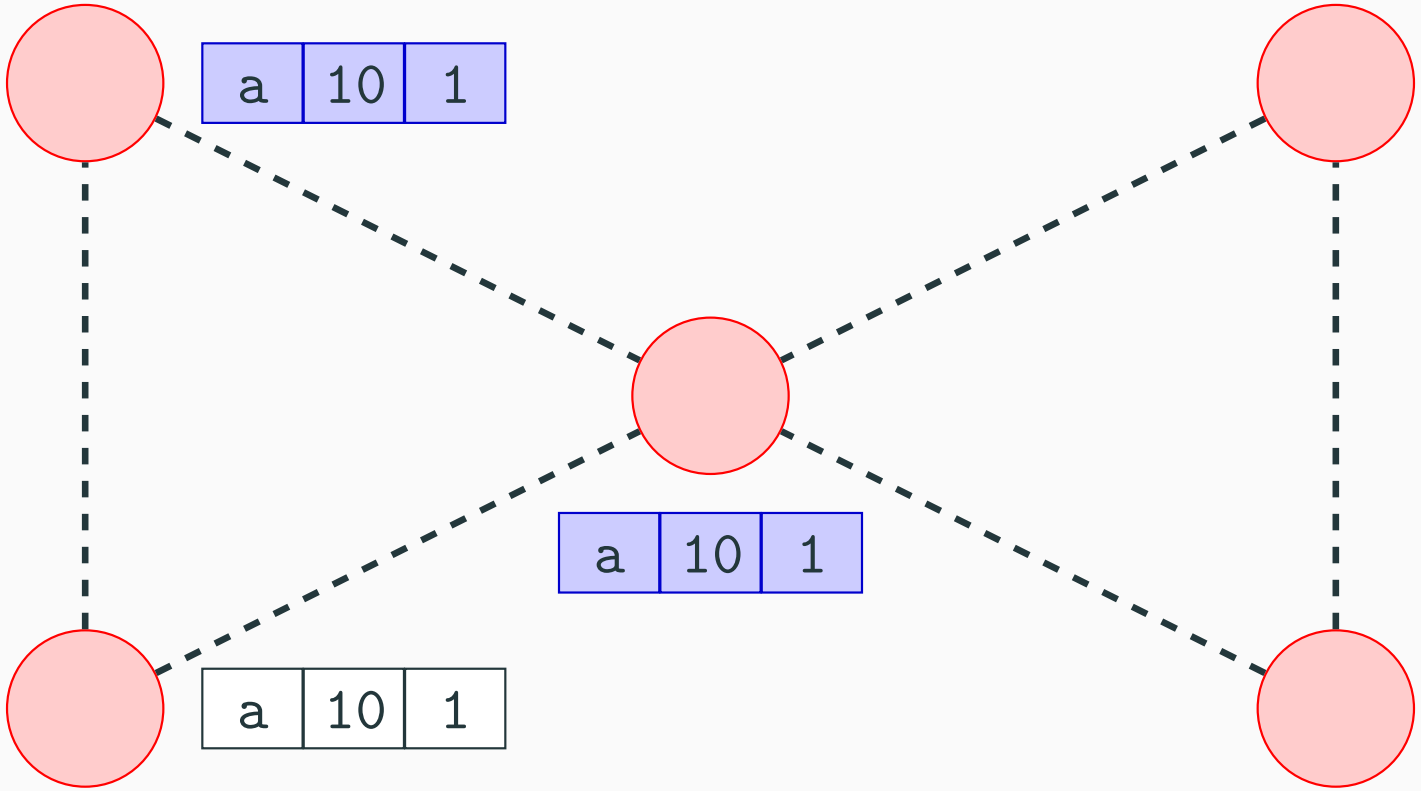
Information Propagation



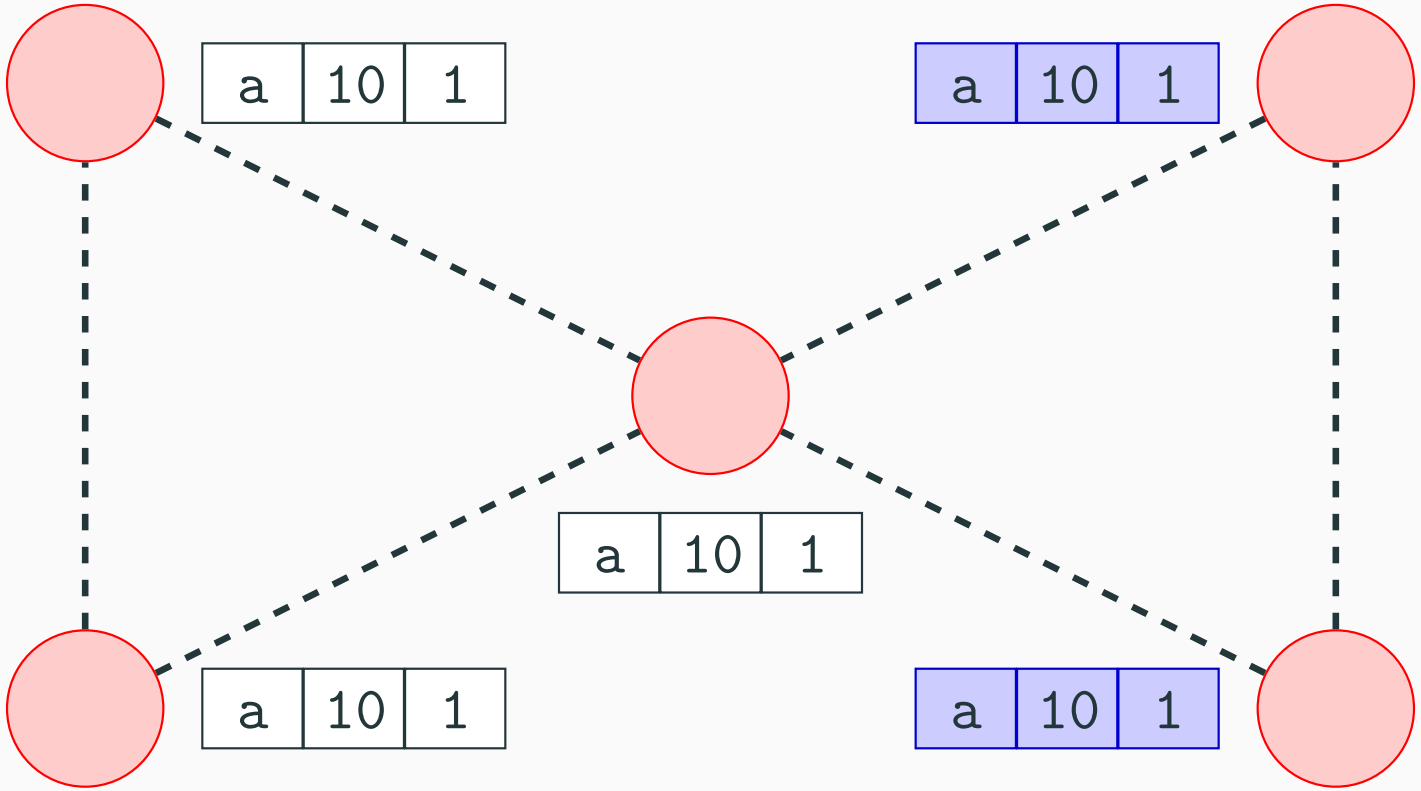
Information Propagation



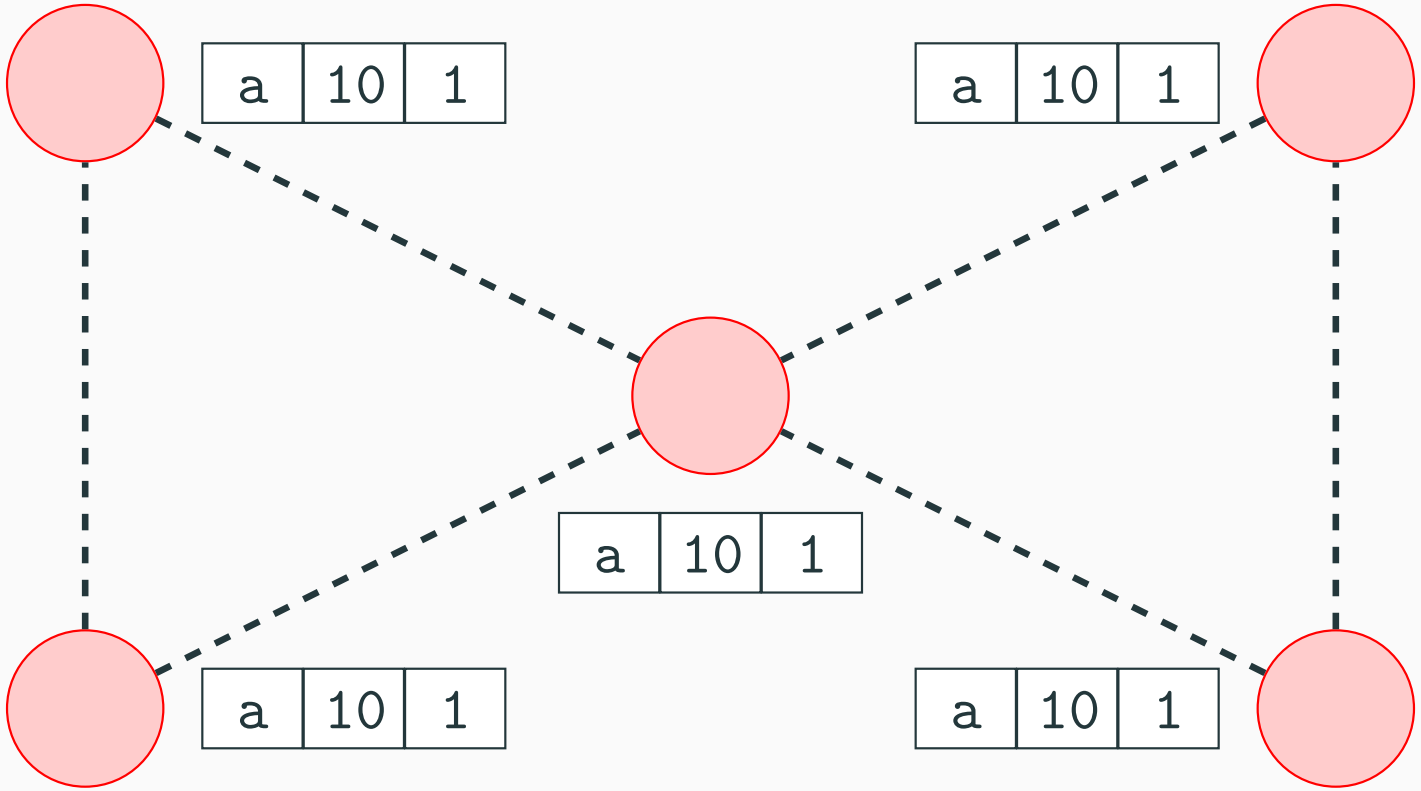
Information Propagation



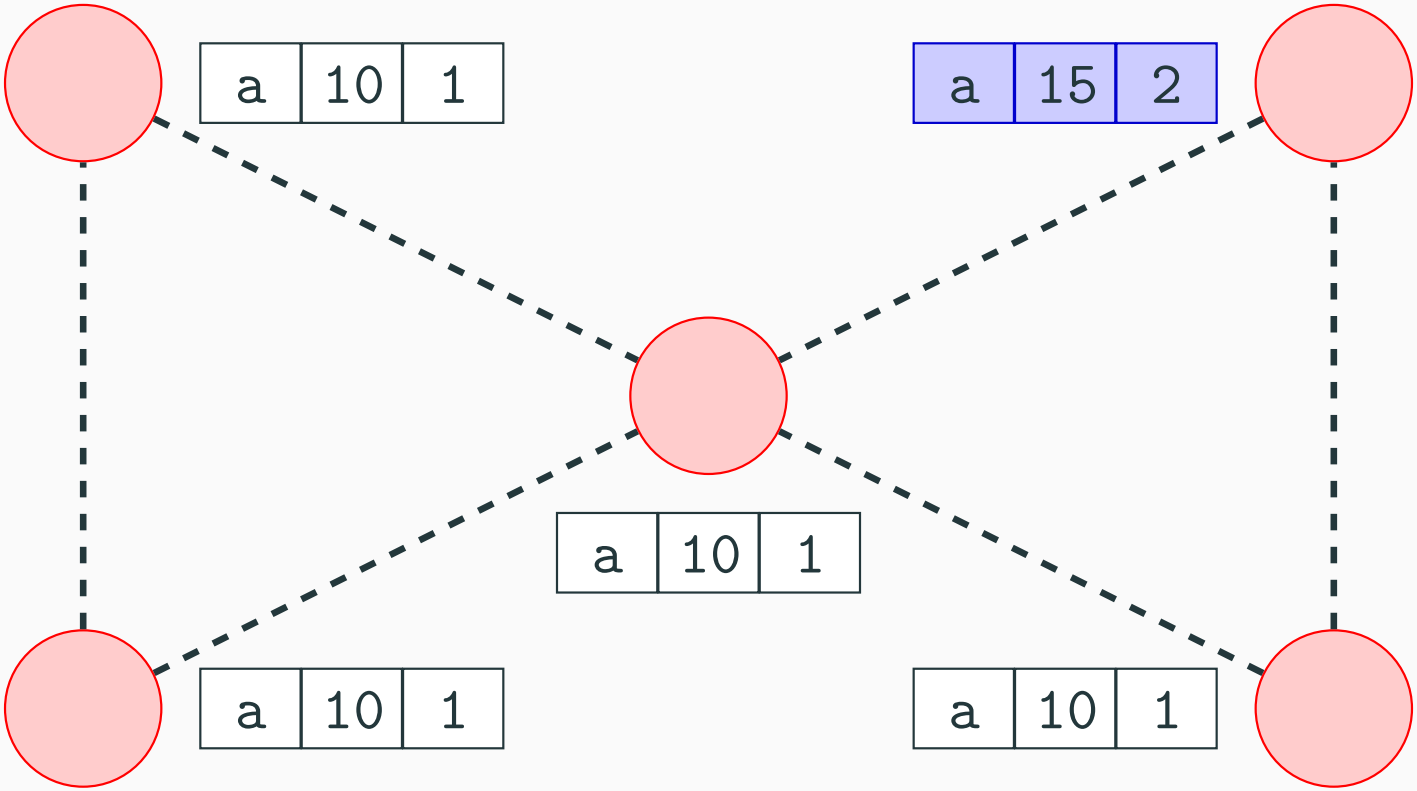
Information Propagation



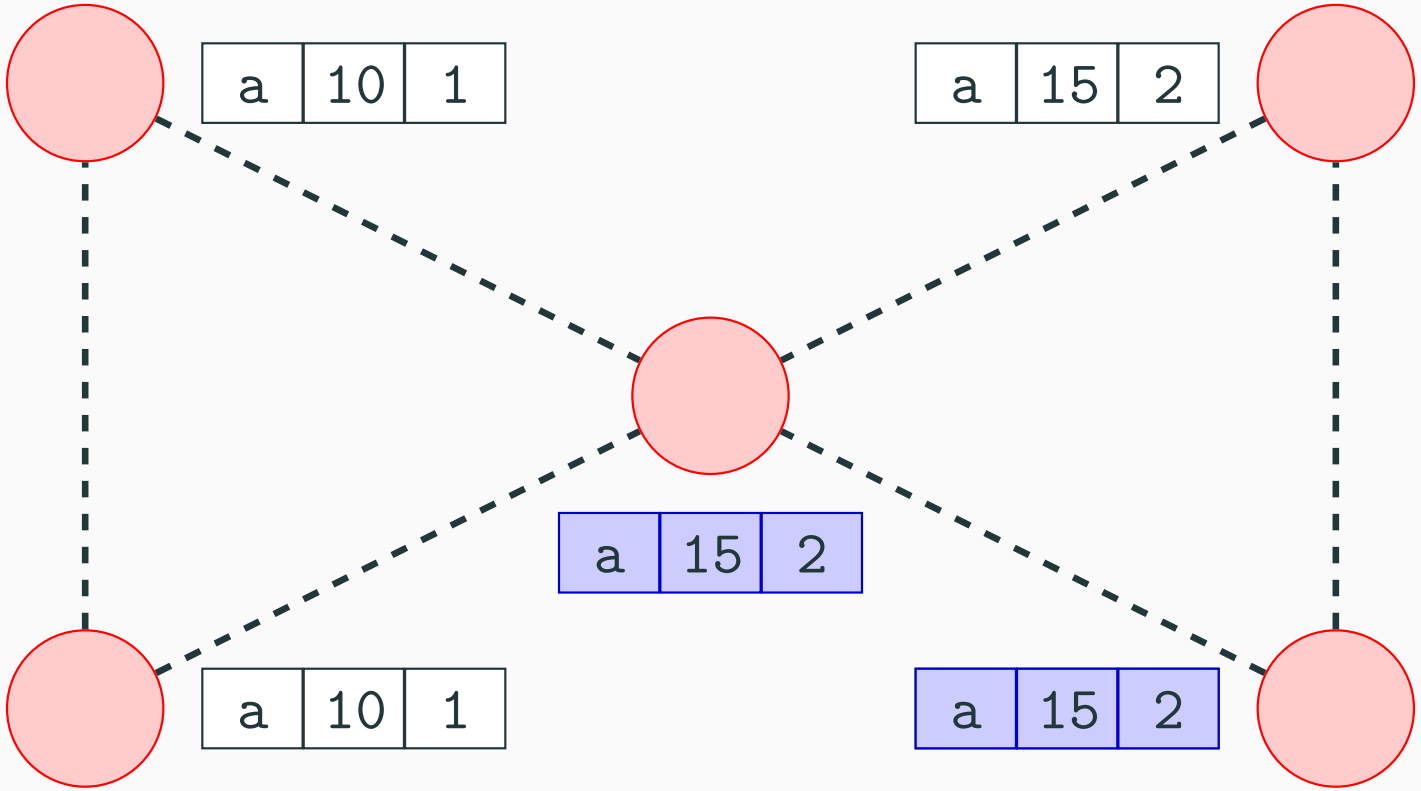
Information Propagation



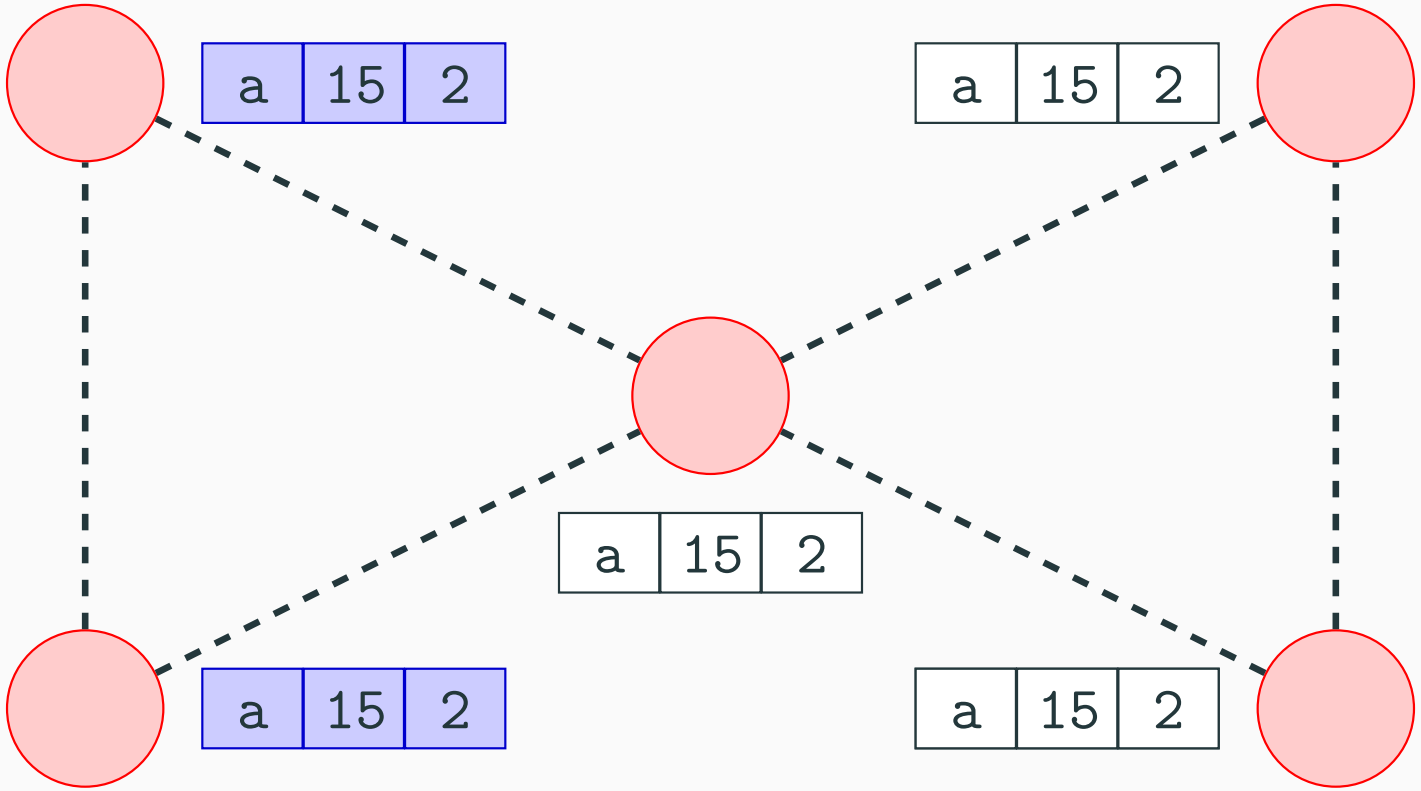
Information Propagation



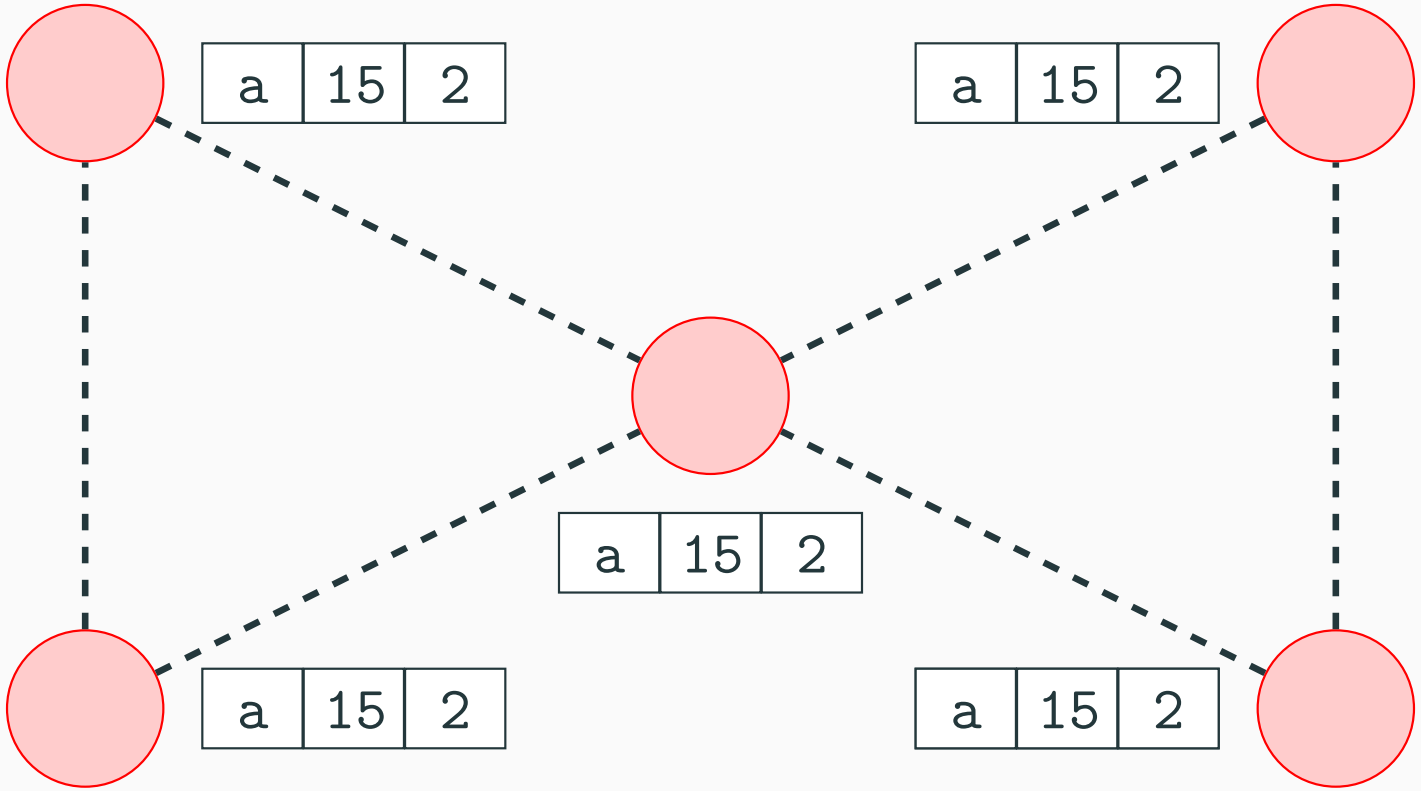
Information Propagation



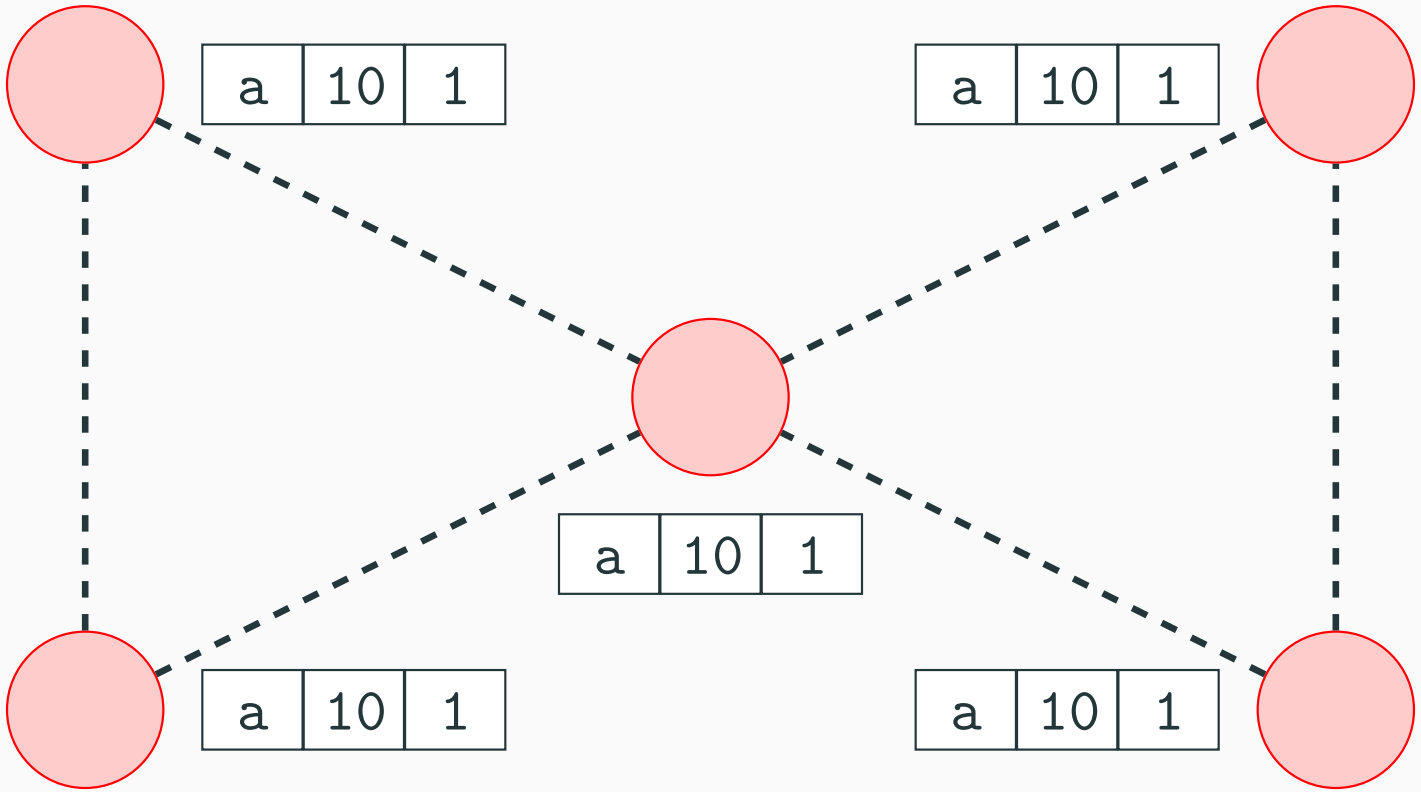
Information Propagation



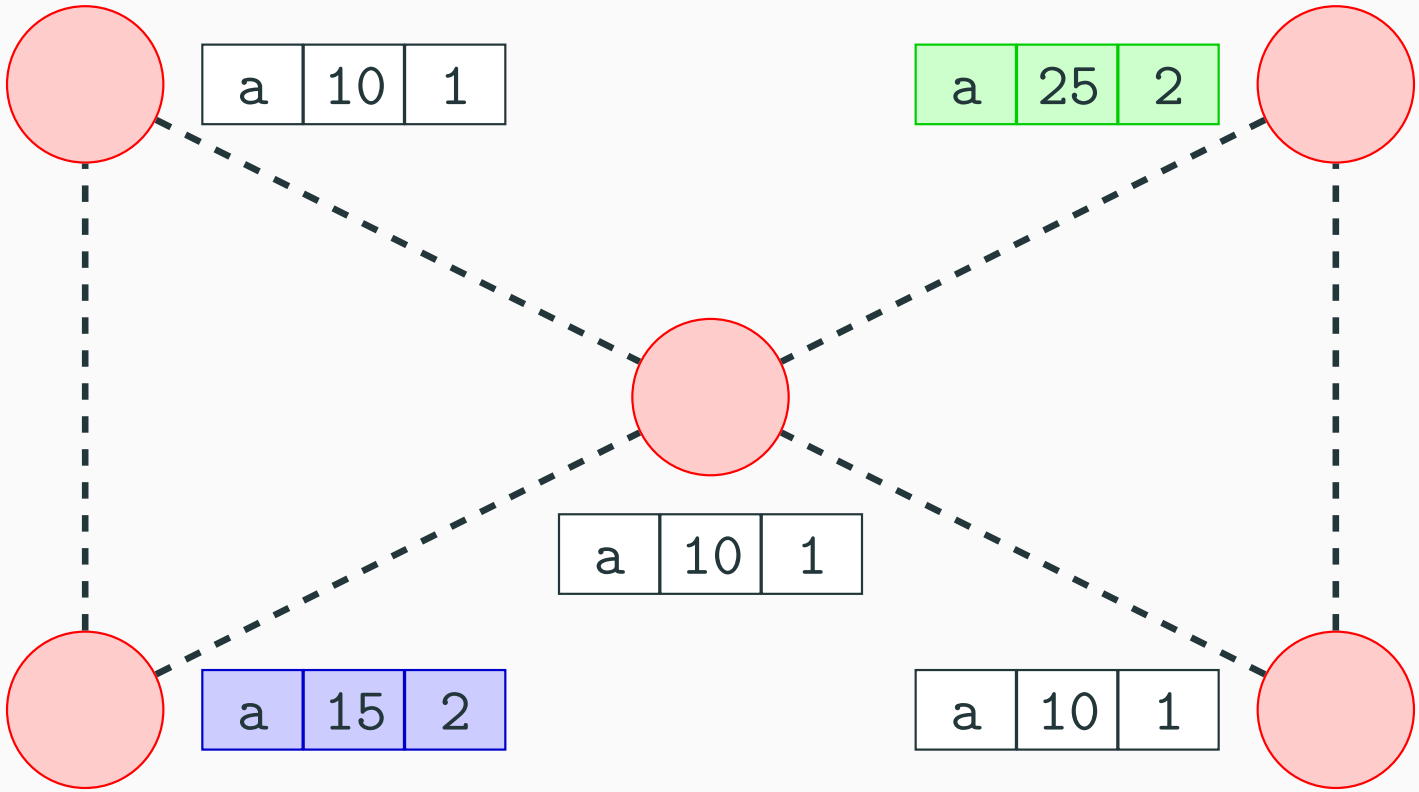
Information Propagation



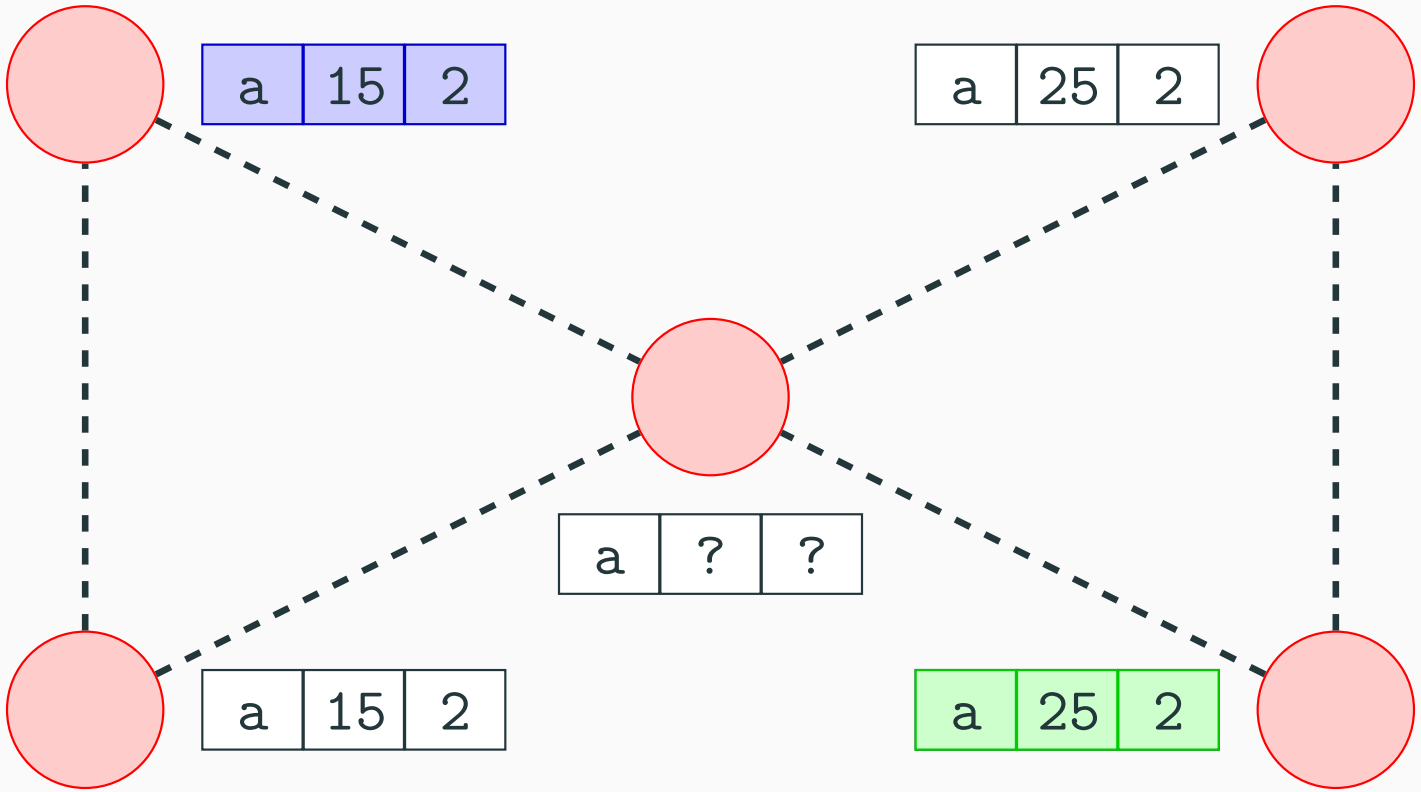
Conflict detection



Conflict detection



Conflict detection

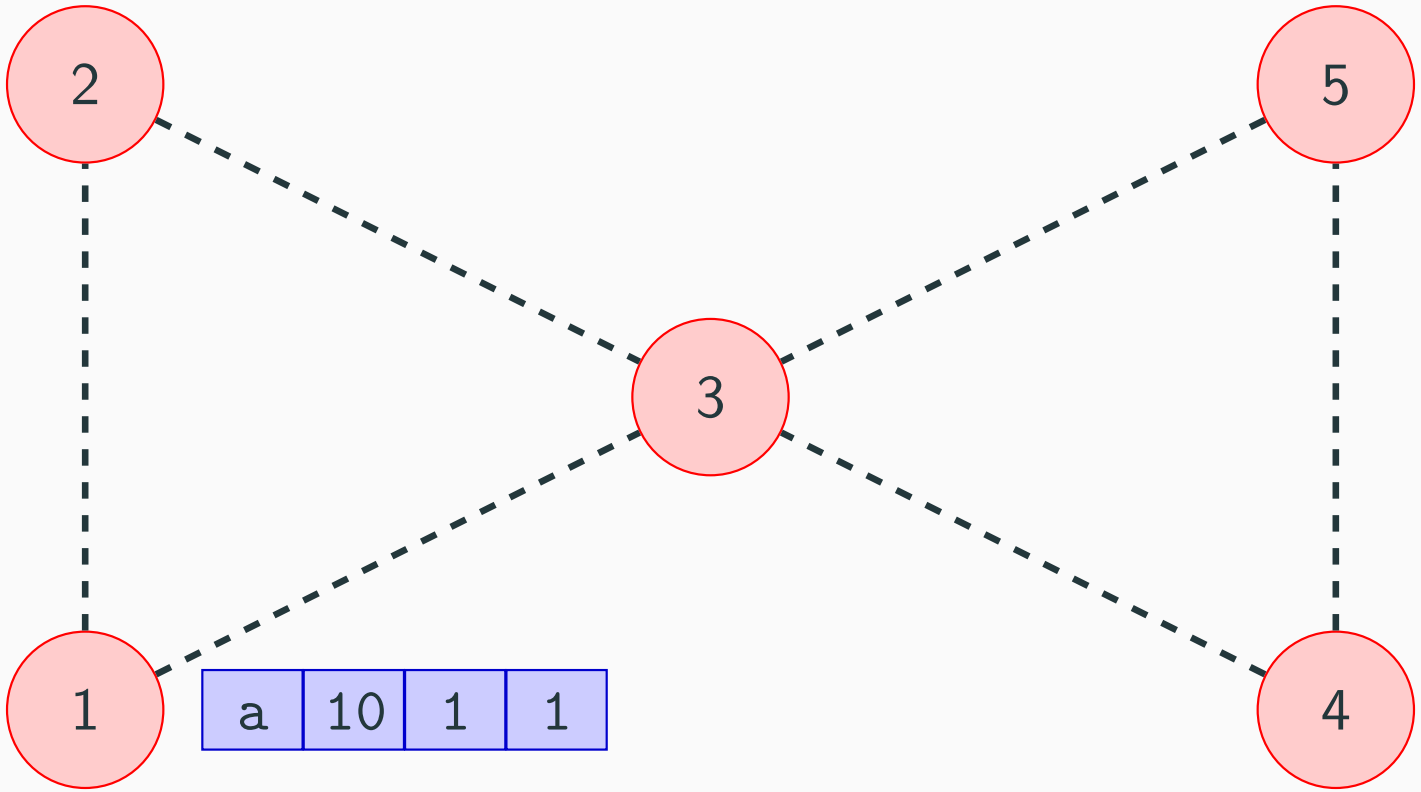


Virtual Stigmergy Operations

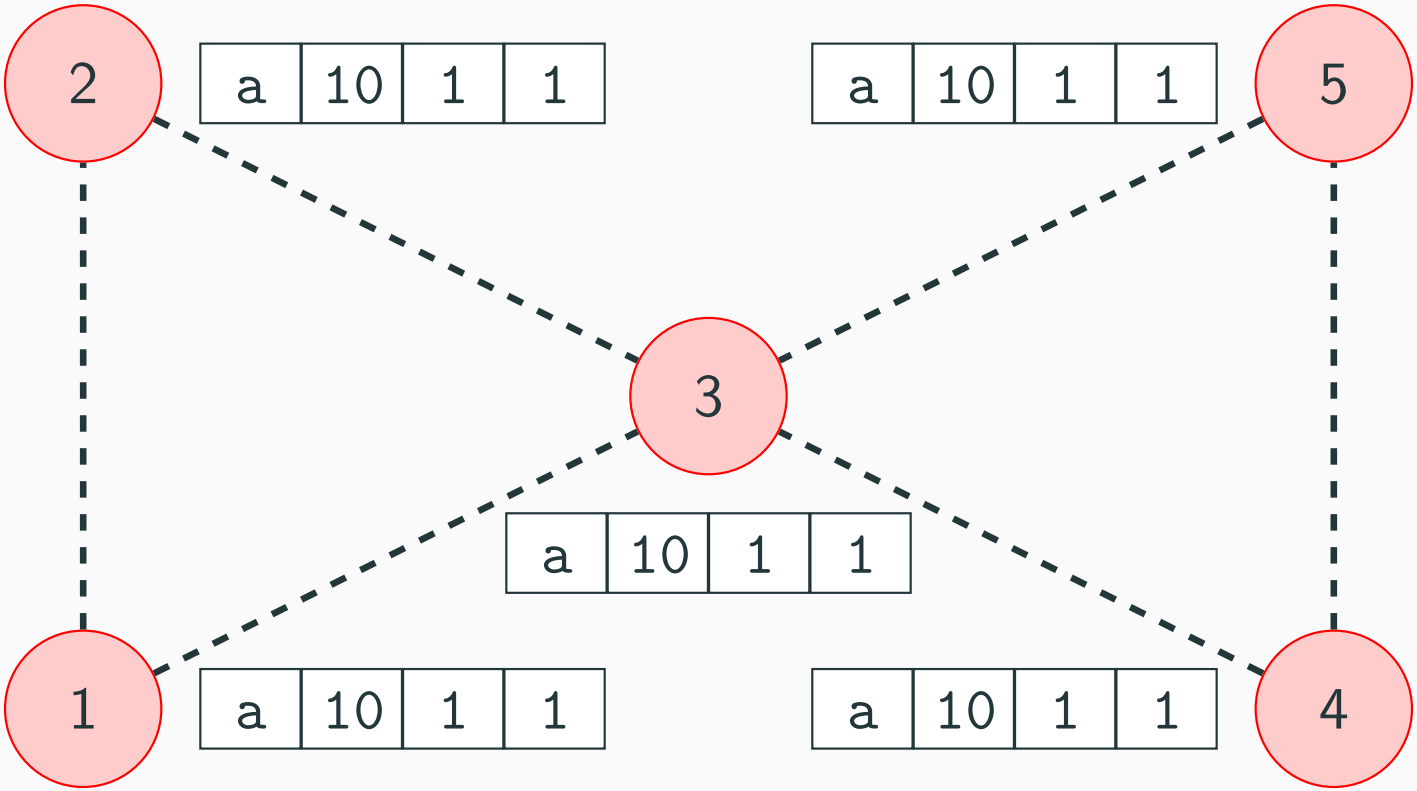
Conflict Manager Definition

```
# Create a new virtual stigmergy
v = stigmergy.create(1)

# Set onconflict manager
v.onconflict(function(k,l,r) {
  # Return local value if
  # - Remote value is smaller than local, OR
  # - Values are equal, robot of remote record is
  #   smaller than local one
  if(r.data < l.data or
    (r.data == l.data and
     r.robot < l.robot)) {
    return l
  }
  # Otherwise return remote value
  else return r
})
```

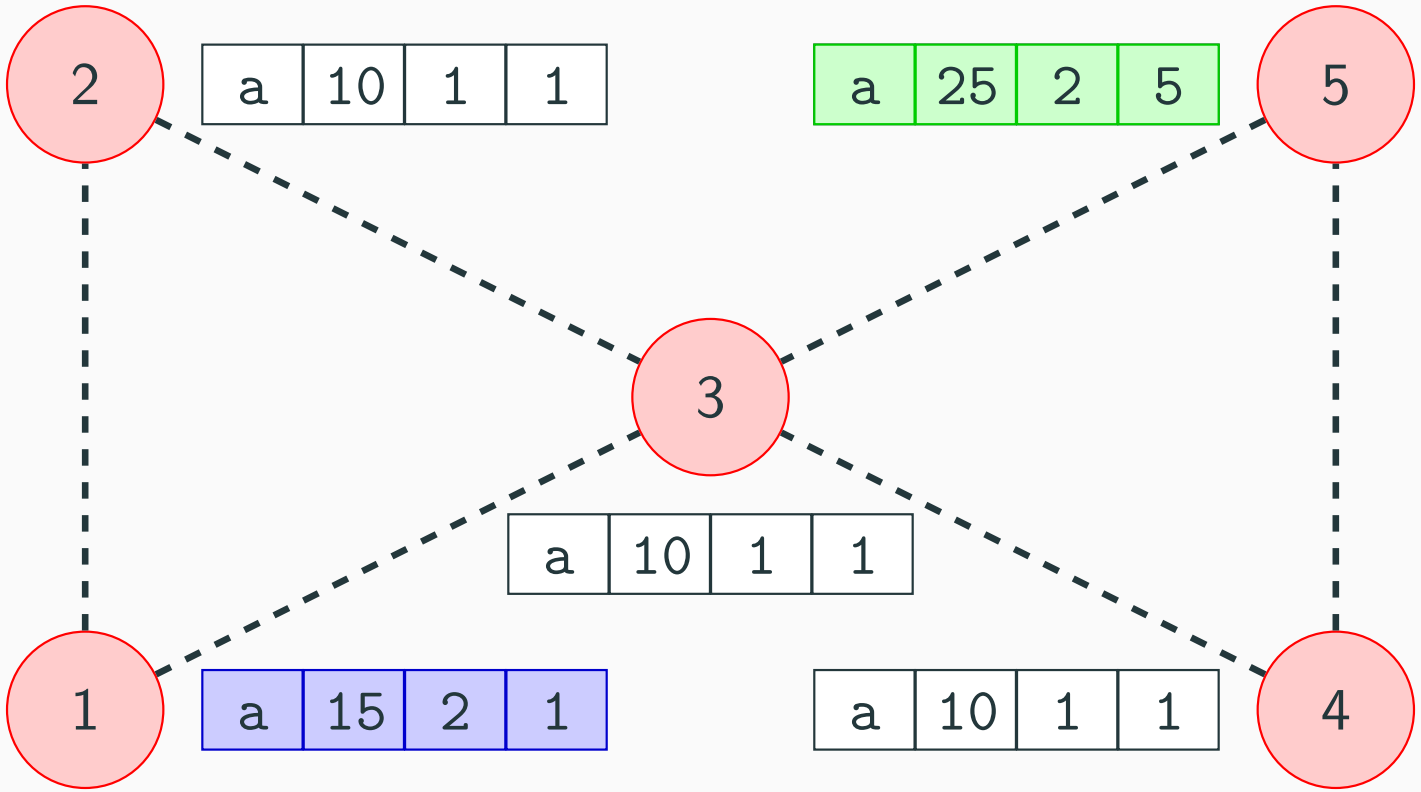
Conflict resolution



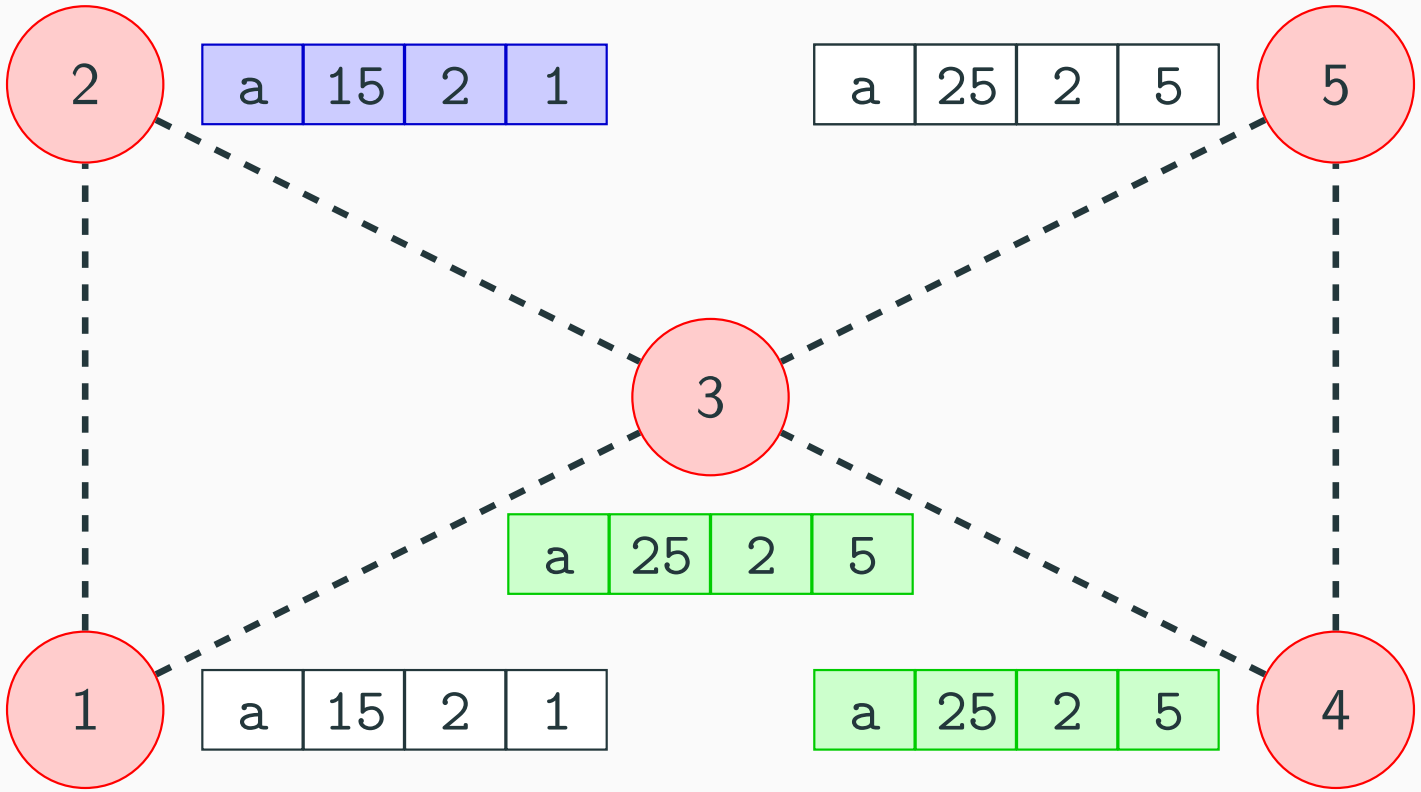
Conflict resolution



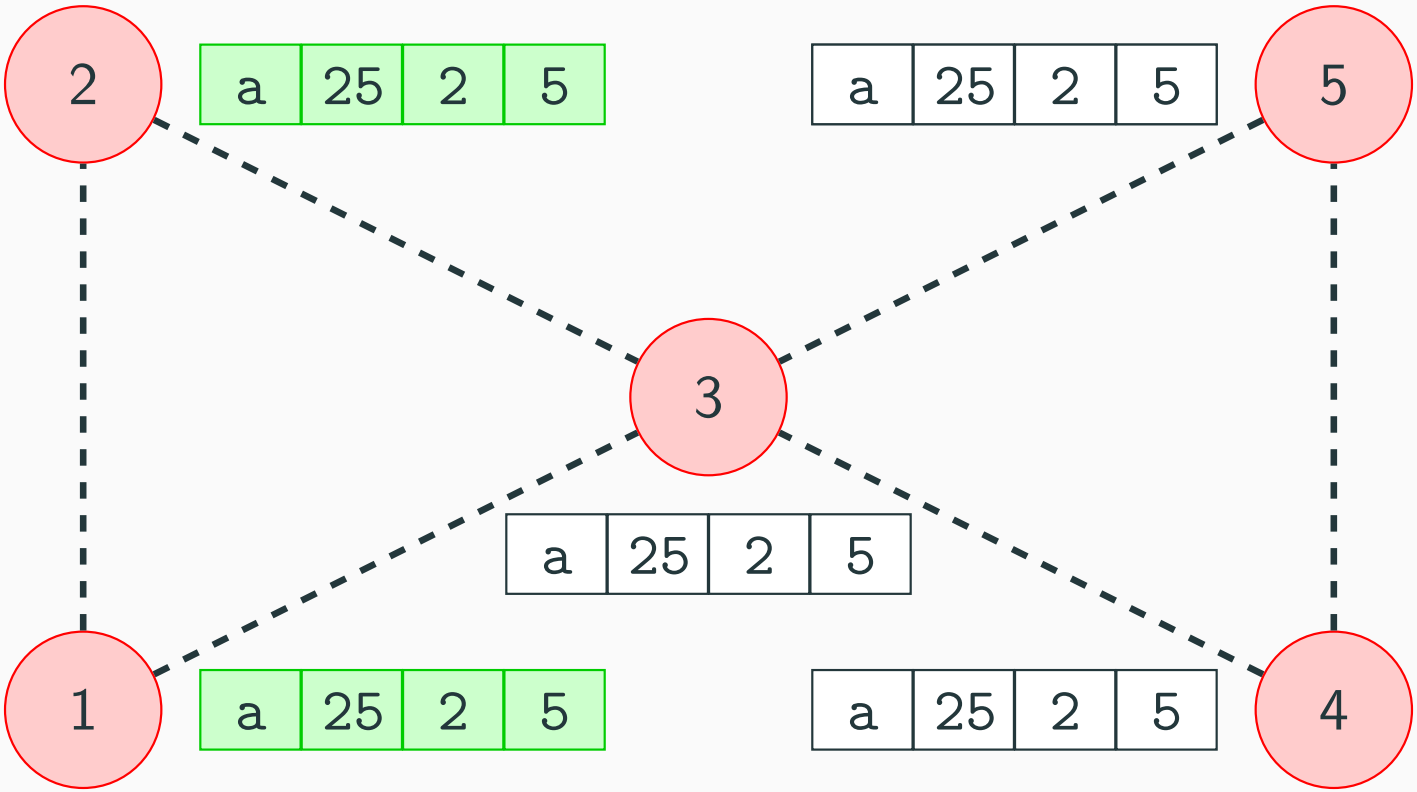
Conflict resolution



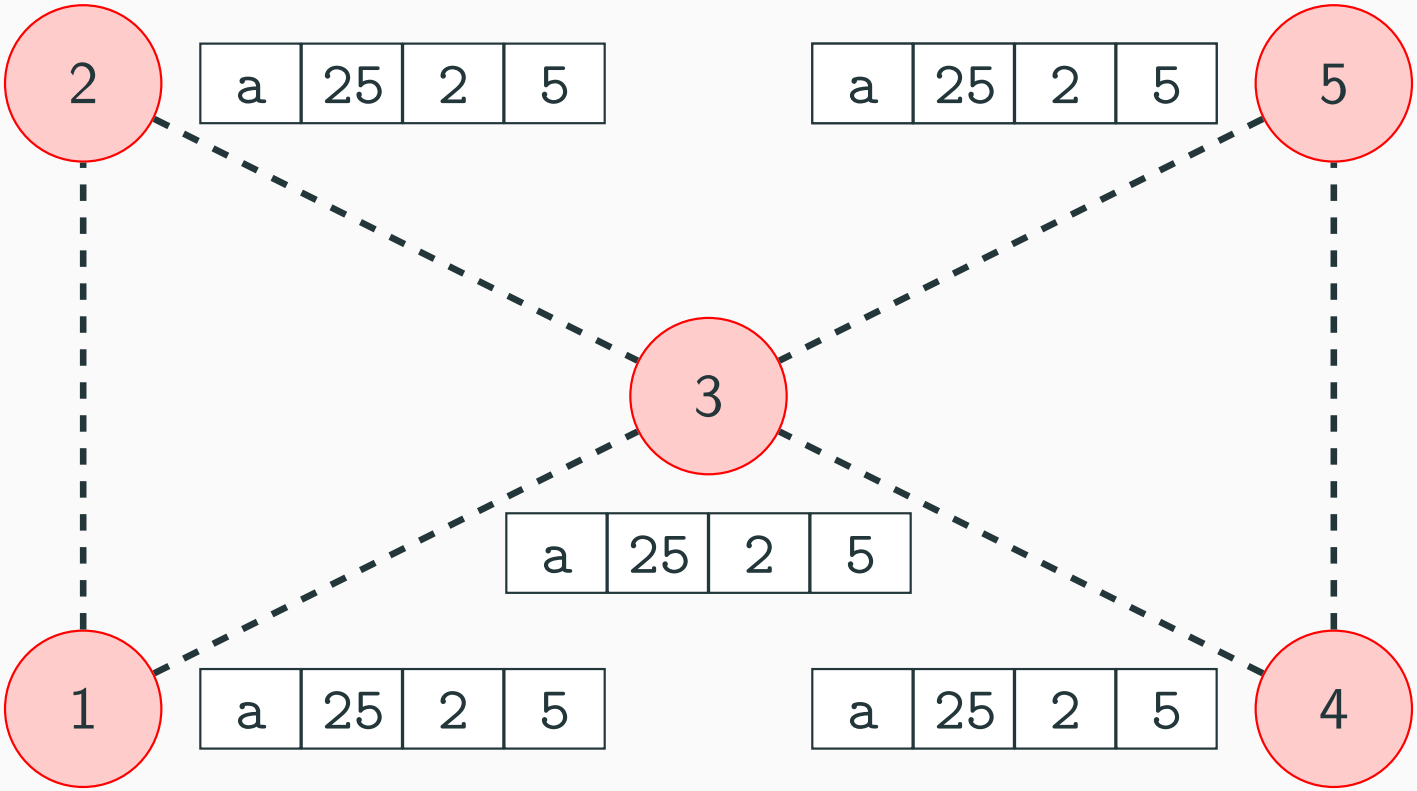
Conflict resolution



Conflict resolution



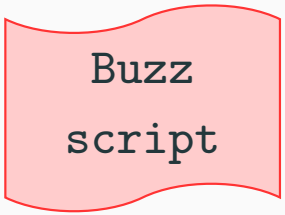
Conflict resolution



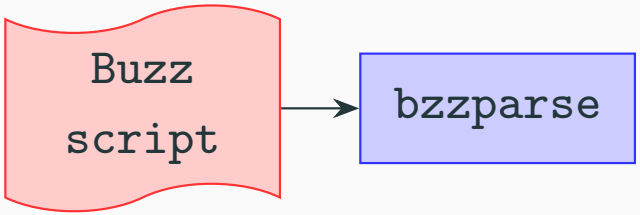
Stigmergy Propagation

Buzz Tool Set

Buzz Tool Set

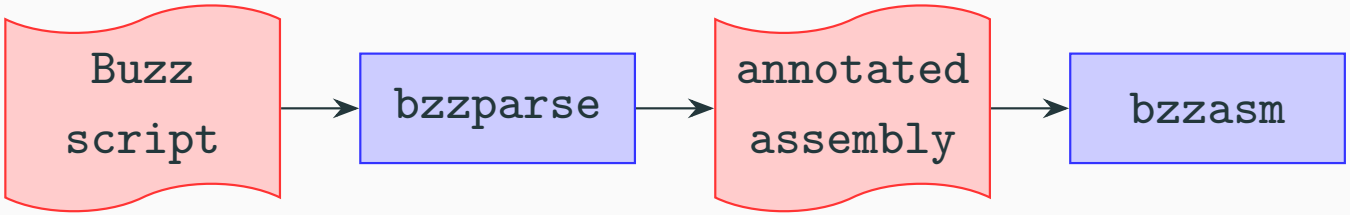


Buzz Tool Set

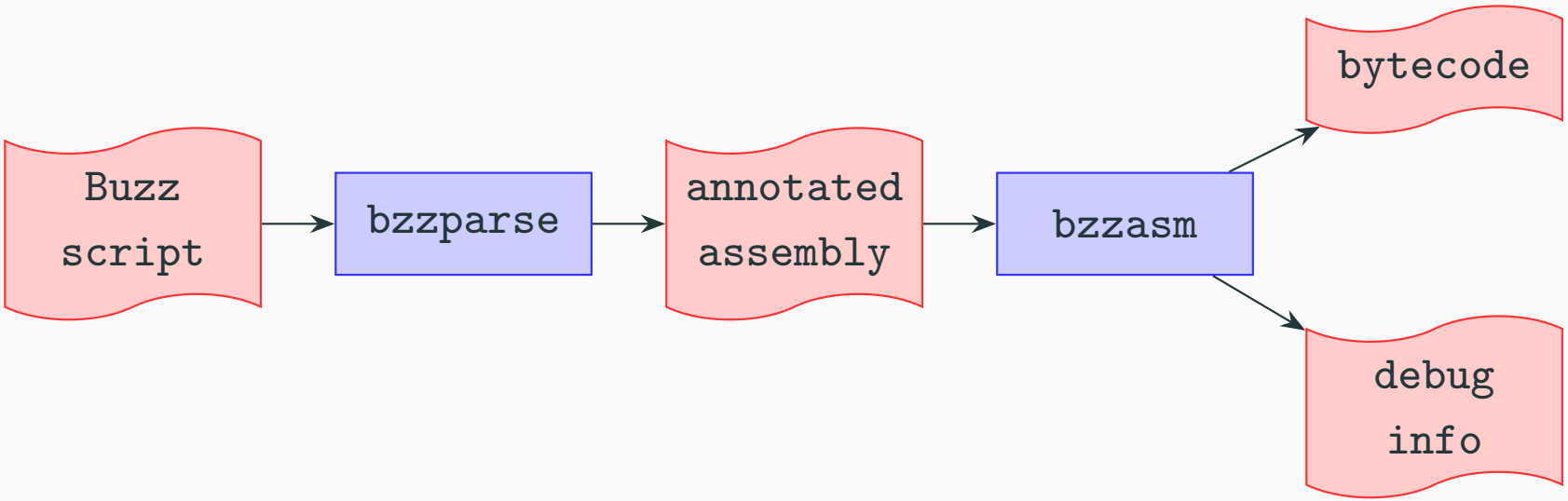


Buzz Tool Set

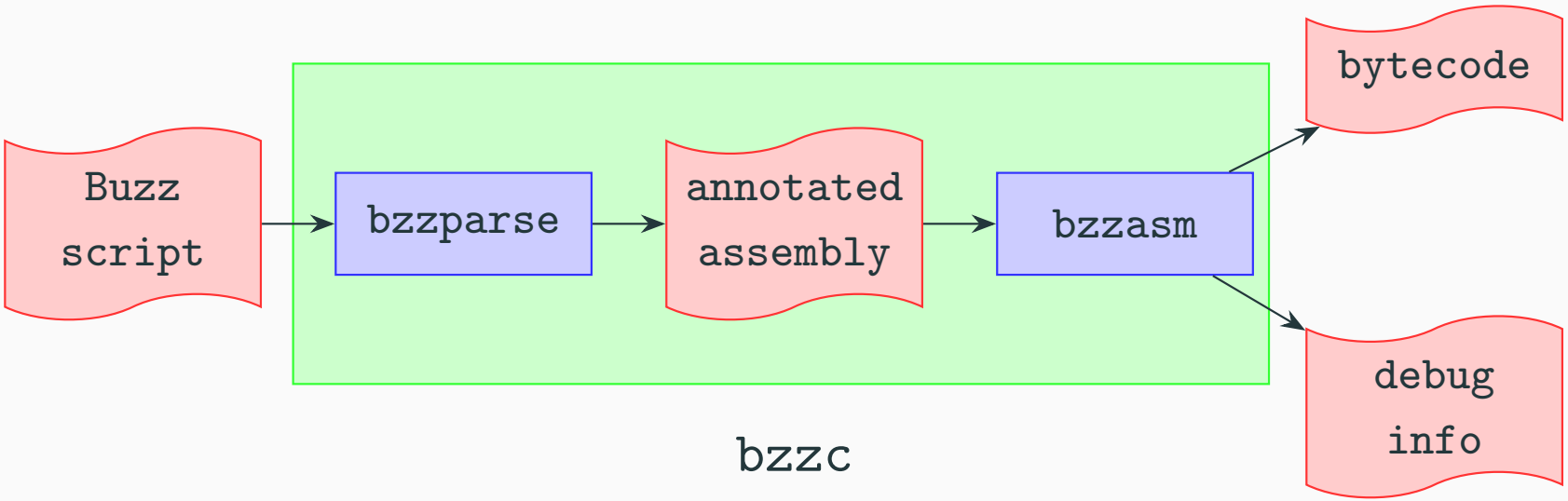
Buzz Tool Set



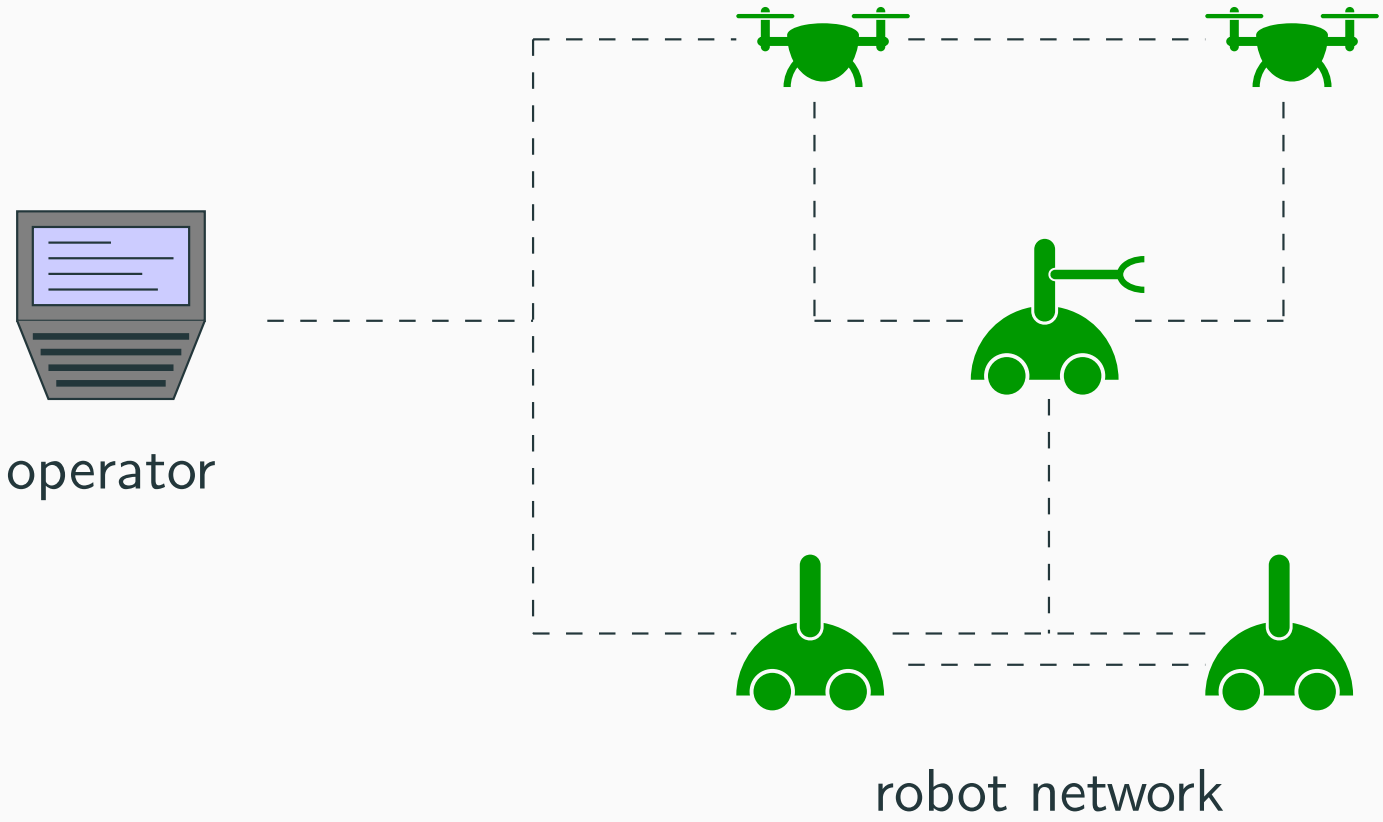
Buzz Tool Set



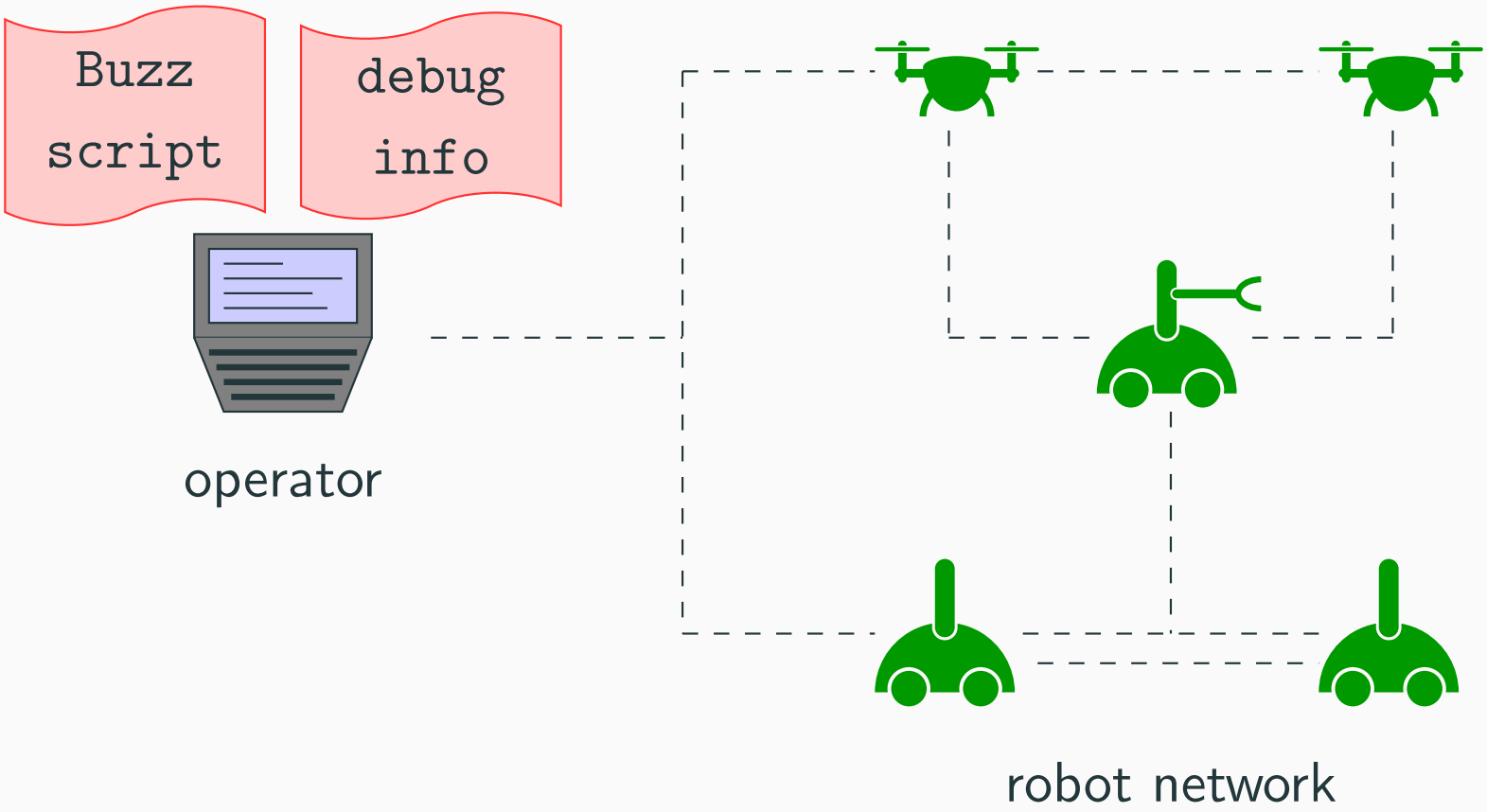
Buzz Tool Set



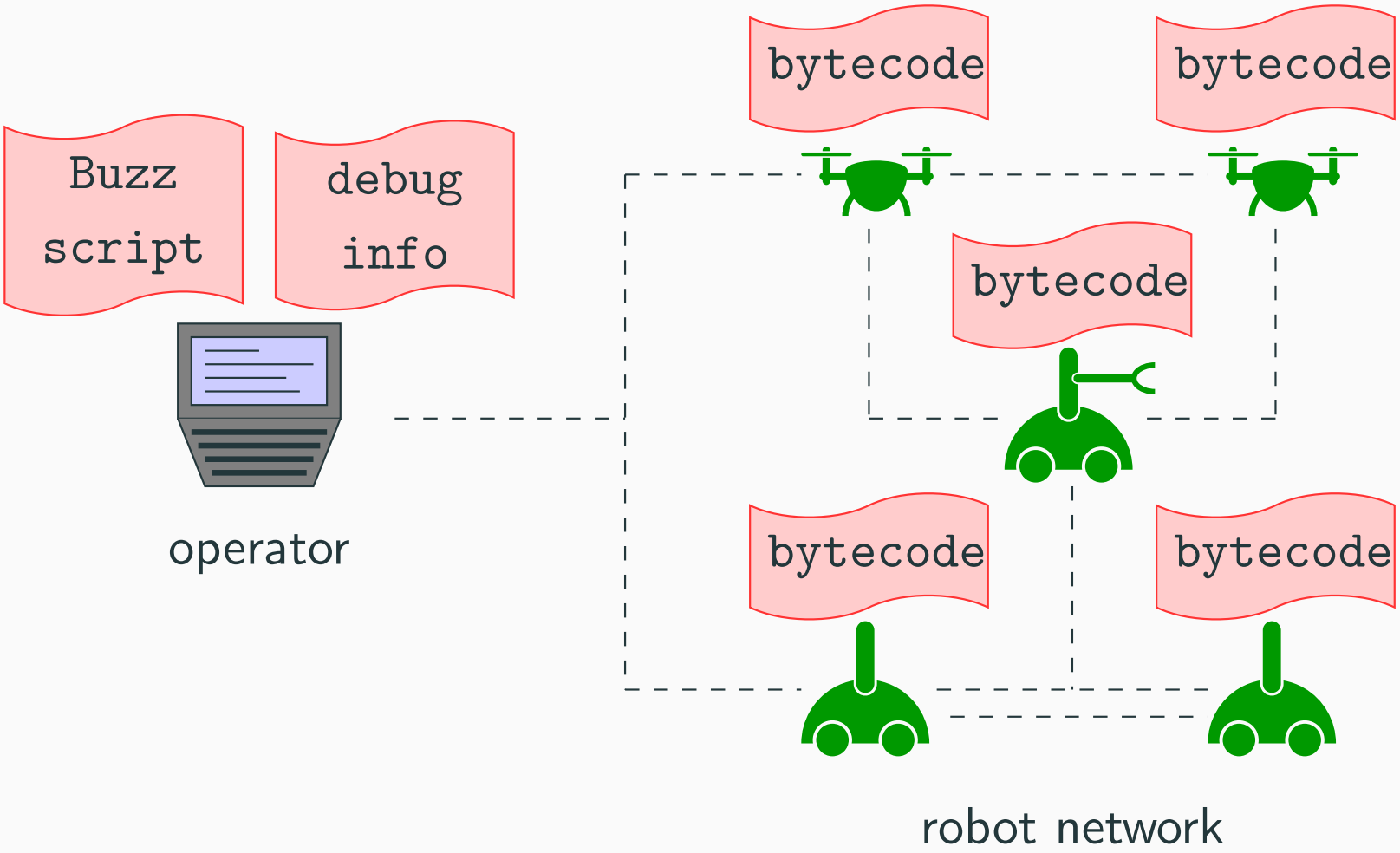
Buzz Deployment



Buzz Deployment



Buzz Deployment



Current Status

Current Status

Work Done

- Language definition
- Compilation toolchain
- Run-time platform (virtual machine)
- Integration with the ARGoS simulator
- Integration with Khepera IV robot

Current Status

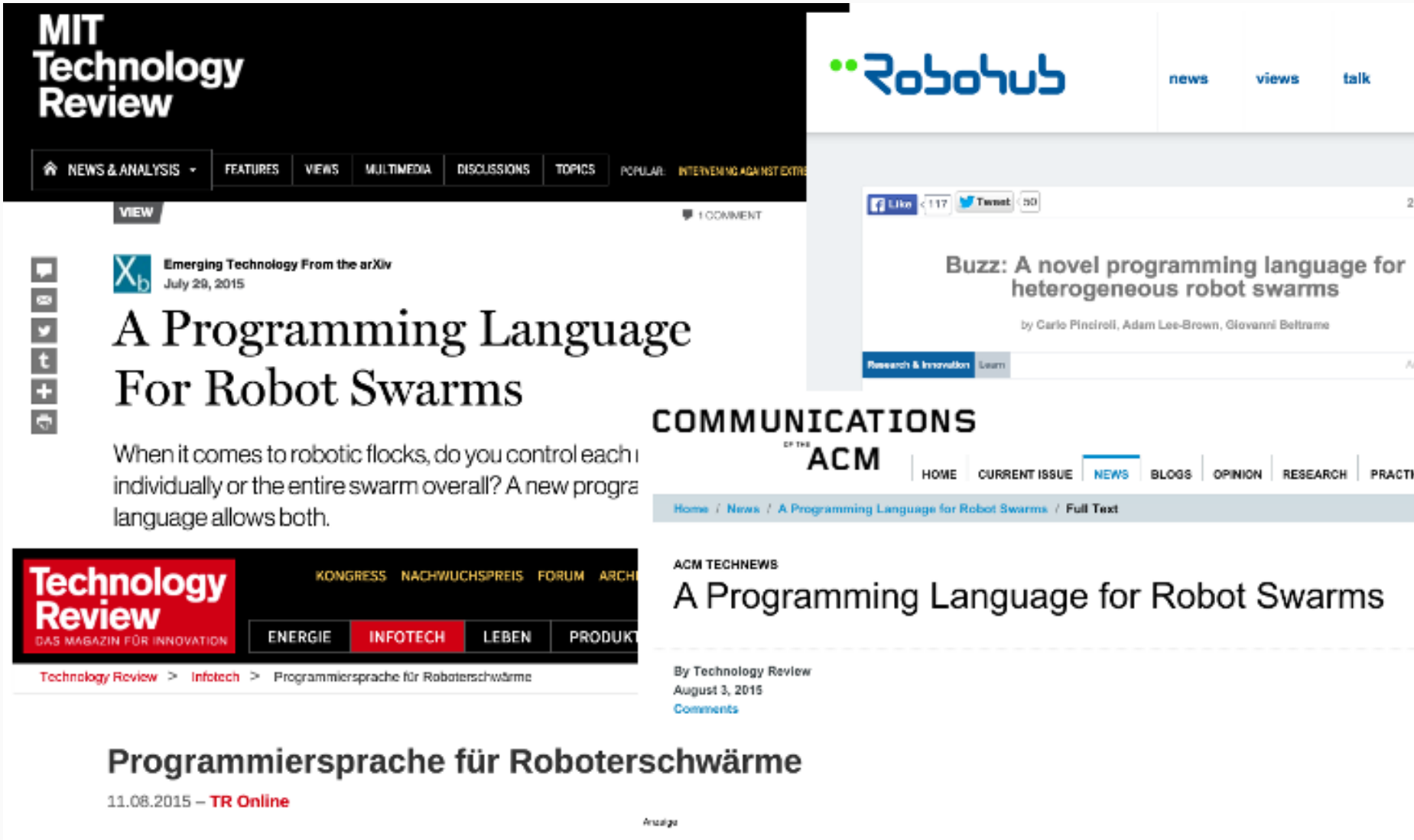
Work Done

- Language definition
- Compilation toolchain
- Run-time platform (virtual machine)
- Integration with the ARGoS simulator
- Integration with Khepera IV robot

Ongoing

- Integration with Mavlink and ROS
- Integration with DJI-SDK

Media Coverage



Thanks for Your Attention!

. . . questions?

`http://the.swarming.buzz/`

Extra Slides

Assumptions to Overcome

- Full network connectivity
  - How to maintain connectivity?
- No individual failures
  - How to manage *predictable* failures?
  - How to manage *unpredictable* failures?

Swarm Behavior Library

Goal: Build a library of common swarm behaviors.

Idea: Code annotations to define *behavioral traits*

```
trait(name:String) {
  impl(kinFunc:FunctionDecl,nonKinFunc:FunctionDecl)
  => FunctionDecl = {
    function $name() {
      var vsum = function(n,a) {math.vec2.add(n,a)}
      var vec_kin = $kinFunc
      var vec_nonkin = $nonKinFunc
      var tot = neighbors.kin().map(vec_kin)
            .reduce(vsum,math.vec2.new(0.0,0.0))
      tot = neighbors.nonkin().map(vec_nonkin)
                          .reduce(vsum, tot)
      math.vec2.scale(tot, 1.0/neighbors.count())
      return tot
    }
  }
 type = NeighborVector => NeighborVector => Pattern
}
```

Enhancing Virtual Stigmergy

Idea: Augment virtual stigmergy with aggregation functions, time-evolution functions, and positional tagging.

Applications:

- Dynamic task allocation
- Dynamic spatially organizing behaviors

Debugging

Problem: Bugs in programs for large swarms are hard to detect and fix.

- How to monitor a robot swarm?
- How to detect misbehaving robots?

Possible directions:

- Plain old logging
- Algebraic graph theory for topology
- Transfer entropy to model information flow
- Machine learning techniques

Deployment

Idea: Real-world robot swarms will be deployed progressively.

- How to design efficient behaviors for *growing* swarms?
- How to change code at runtime?