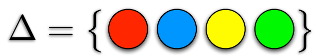
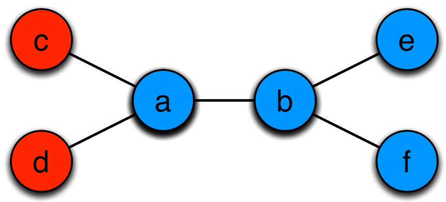
Scheduling

- **Scheduler** (a.k.a. **Daemon**): the daemon chooses among activatable processes those that will execute their actions
  - can be seen as an *adversary* whose role is to prevent stabilization

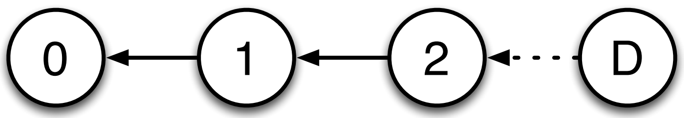
Spatial Scheduling

$$true \rightarrow color_i := Min\{\Delta \setminus \{color_j | j \in Neighbors_i\}\}$$





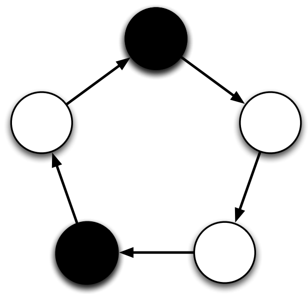
Spatial Scheduling



Temporal Scheduling

$$token \rightarrow pass\ token\ to\ left\ neighbor\ with\ probability\ \frac{1}{2}$$

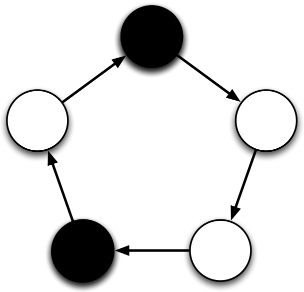
$token =$ ● $\quad no\ token =$ ○



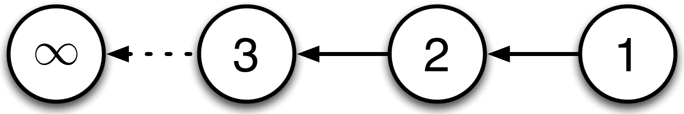
Temporal Scheduling

$token \rightarrow pass\ token\ to\ left\ neighbor\ with\ probability\ \frac{1}{2}$

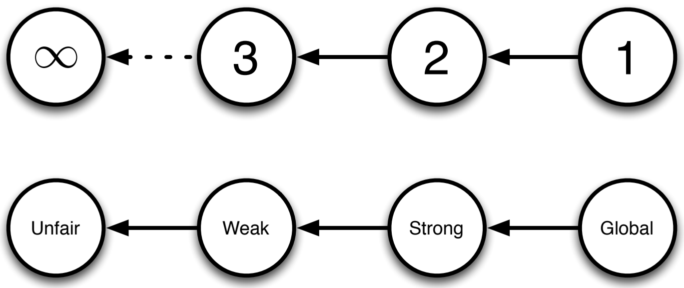
$token =$ ● $no\ token =$ ○



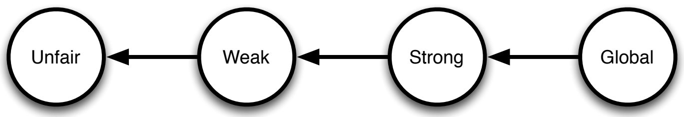
Temporal Scheduling



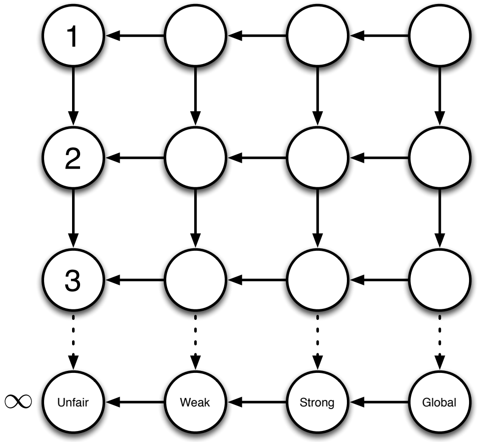
Temporal Scheduling



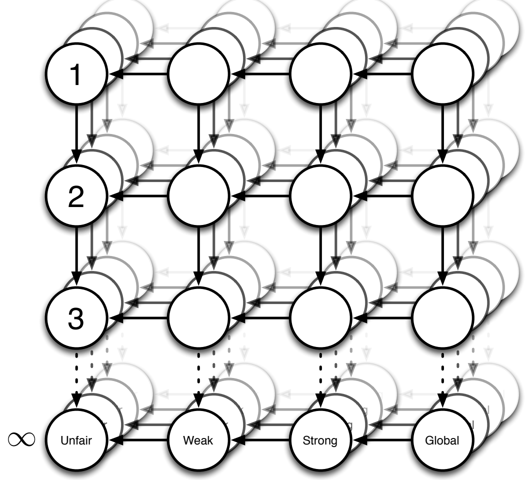
A Map of Daemons



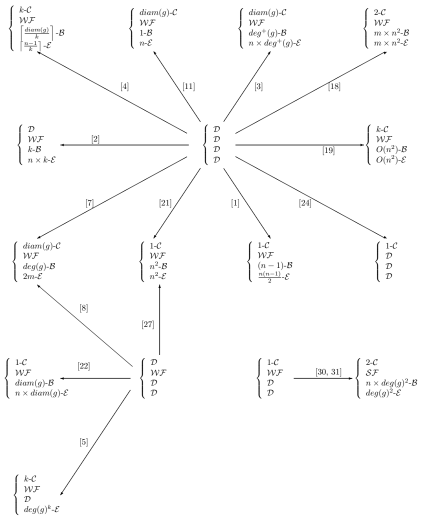
A Map of Daemons



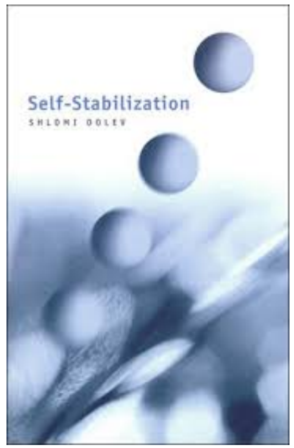
A Map of Daemons



Daemon Transformers



Self-stabilization



Population Protocols

Population Protocols

Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, René Peralta: Computation in networks of passively mobile finite-state sensors. Distributed Computing 18(4): 235-253 (2006)

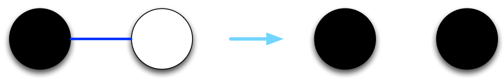
Population Protocols

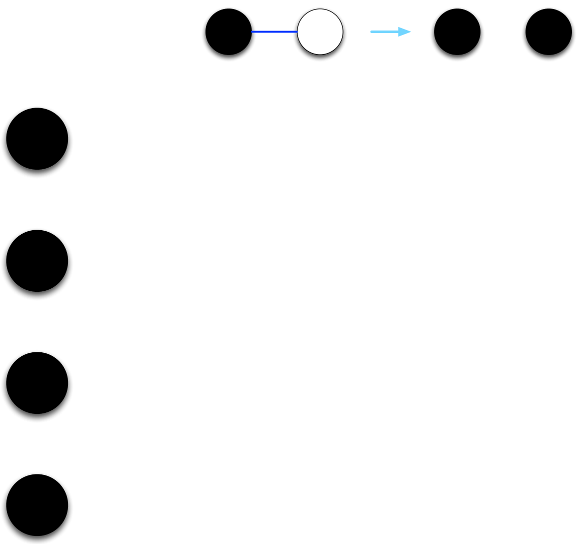
Population Protocols

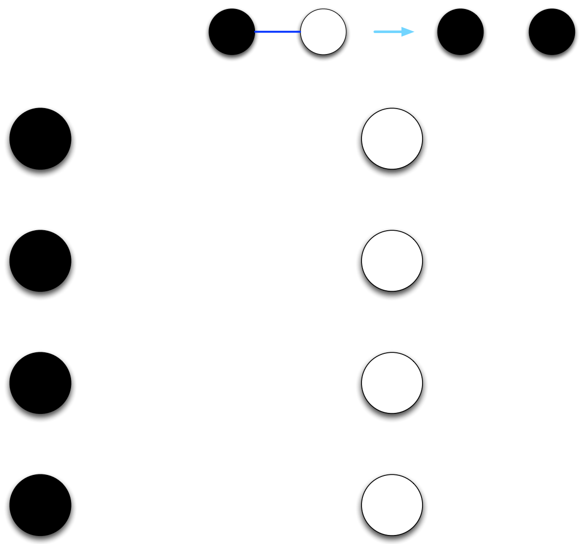
Population Protocols

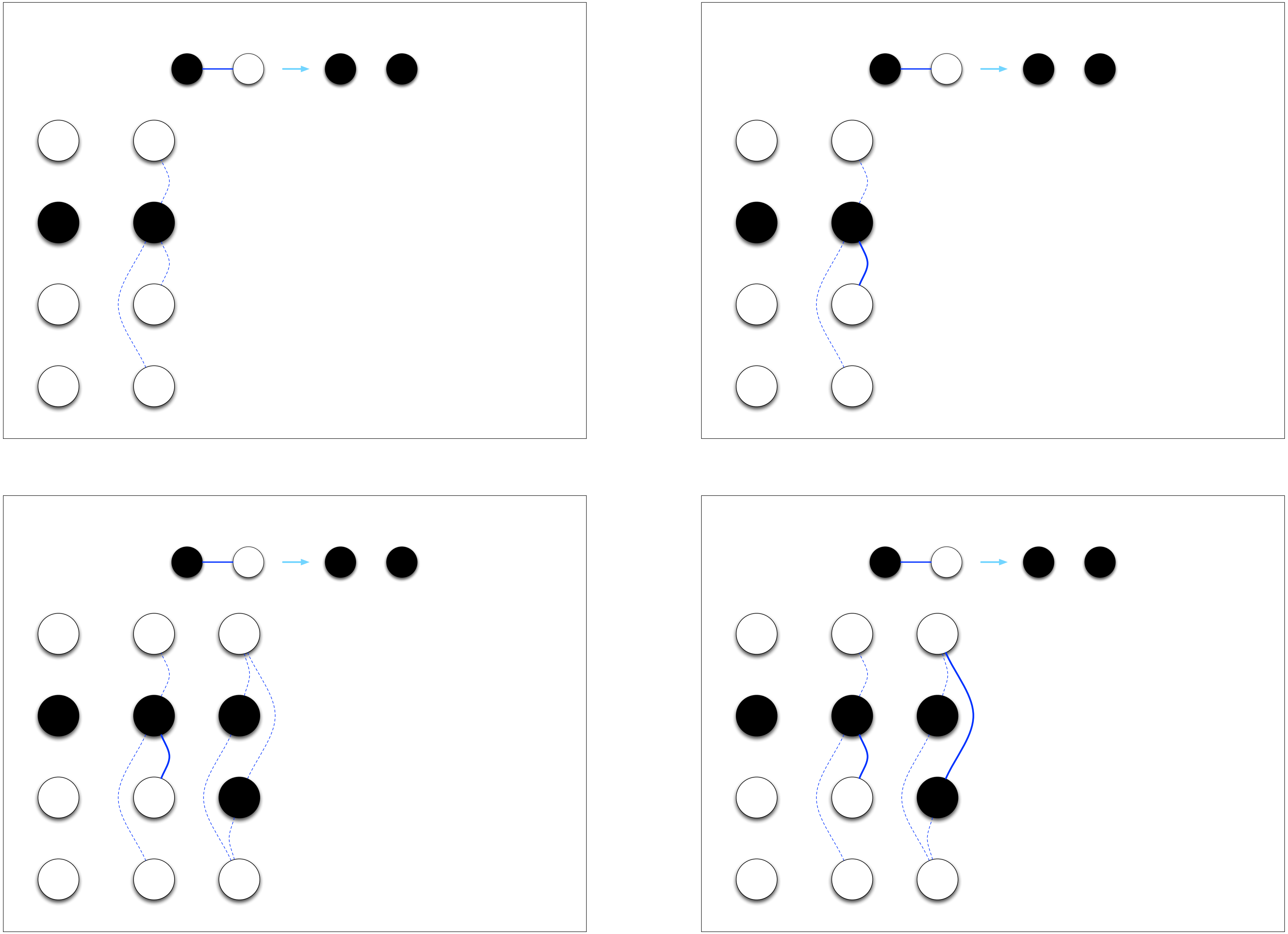
- **Definition**
  - A *Population Protocol* is a 6-tuple **(X,Y,Q,I,O,T)**
    - **X**: Set of inputs
    - **Y**: Set of outputs
    - **Q**: Set of states
    - **I**: Input mapping function, $X \longrightarrow Q$
    - **O**: Output mapping function, $Q \longrightarrow Y$
    - **T**: Transition function, $Q \times Q \longrightarrow Q \times Q$

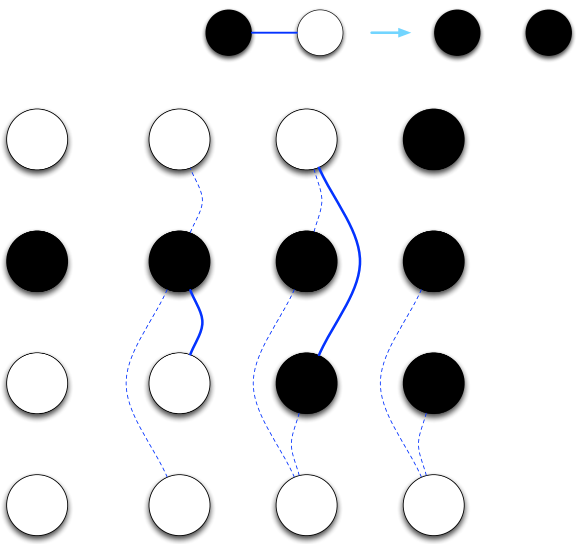
Example 1

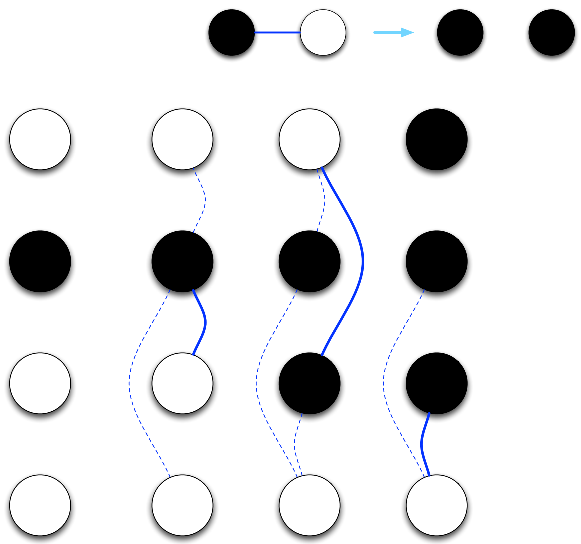


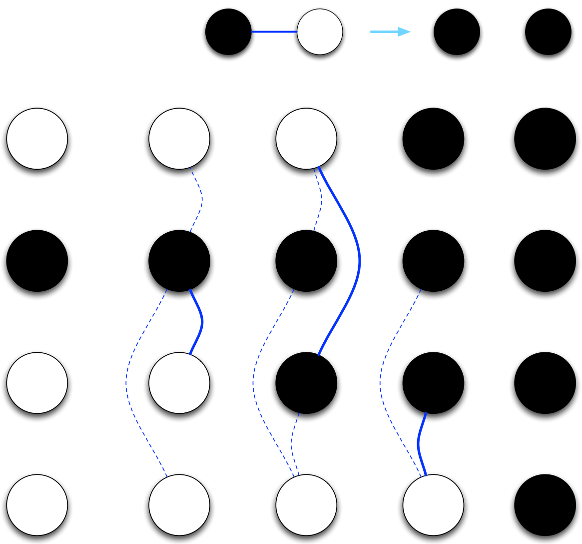




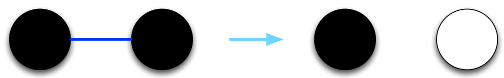


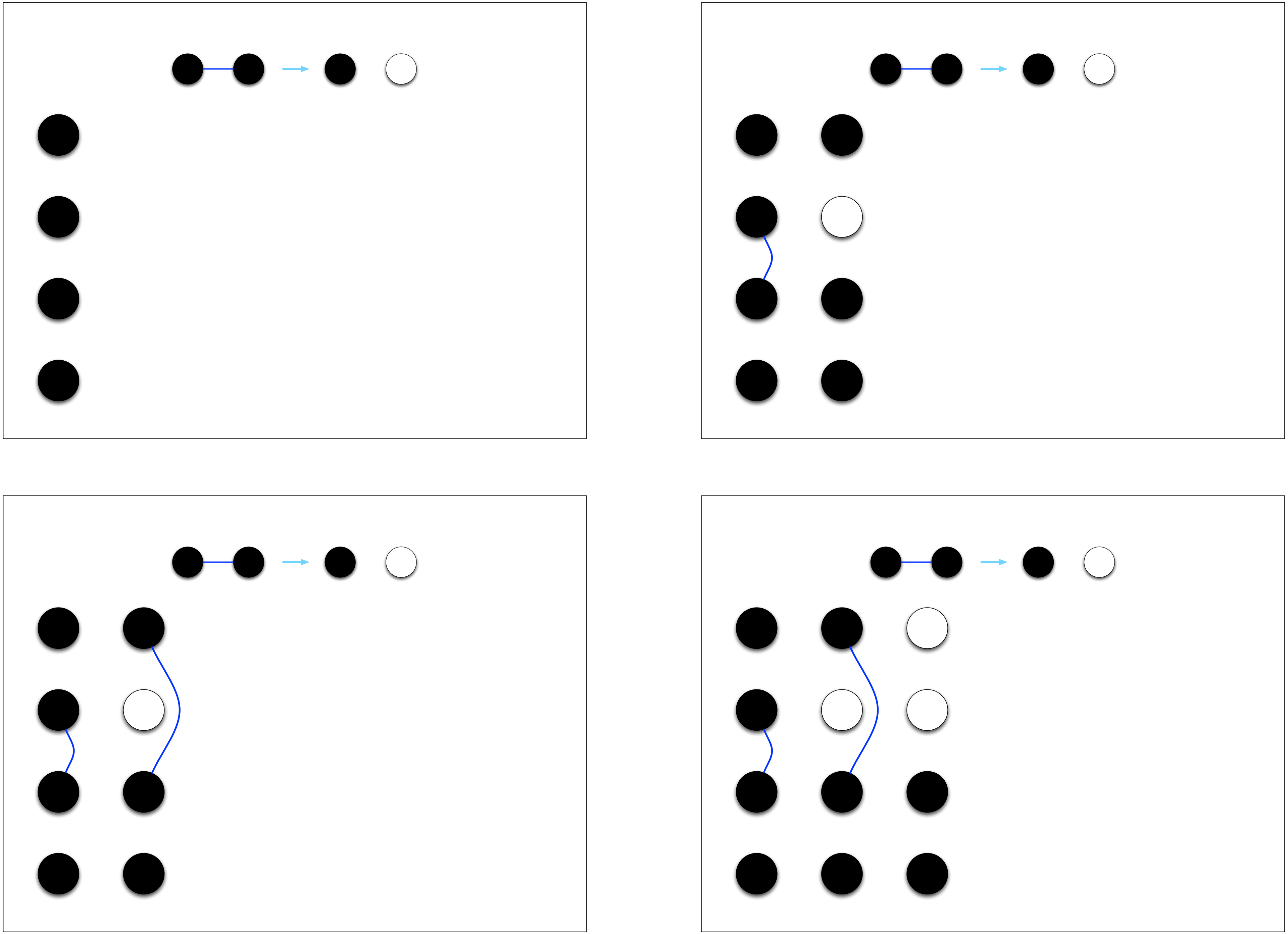


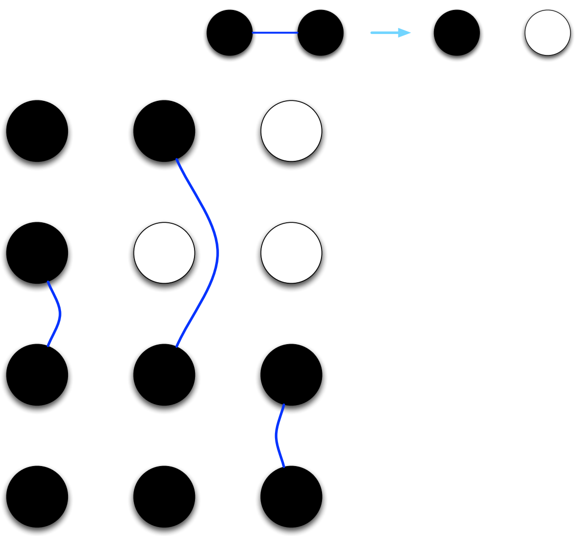


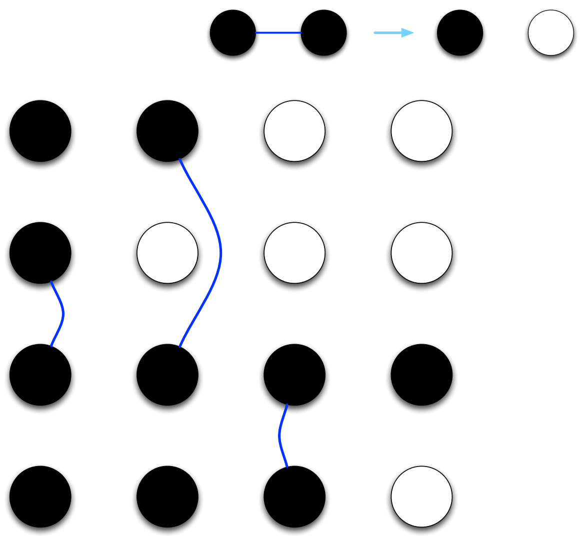


Example 1b

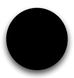
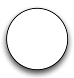




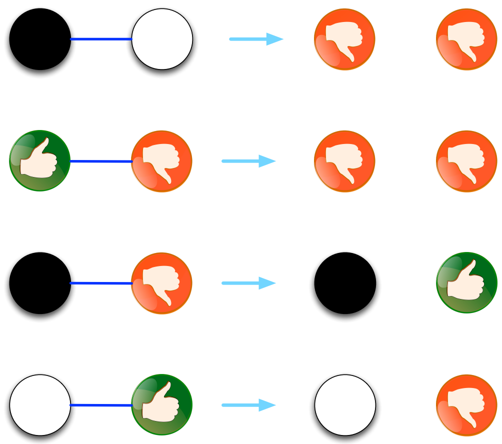


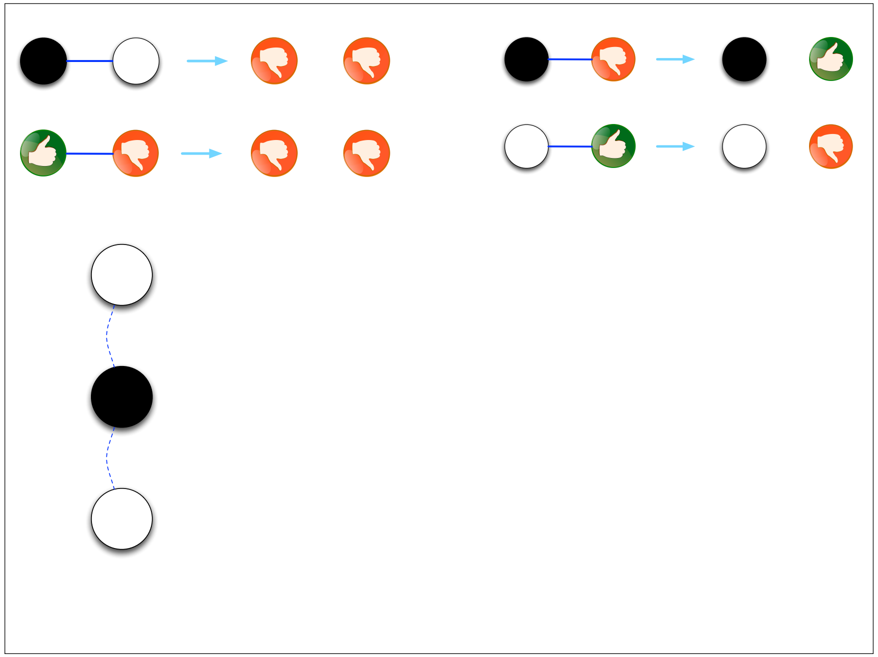
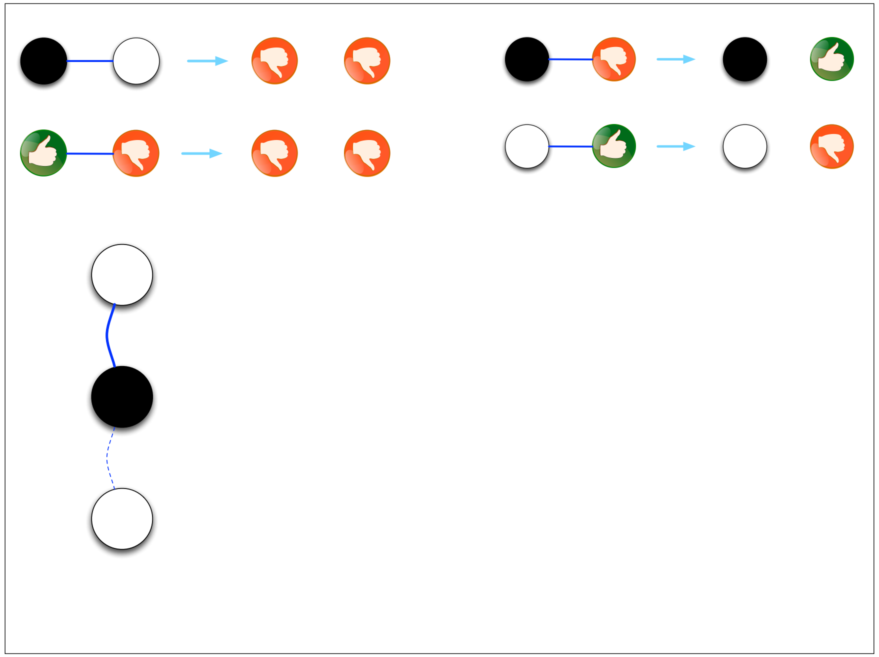
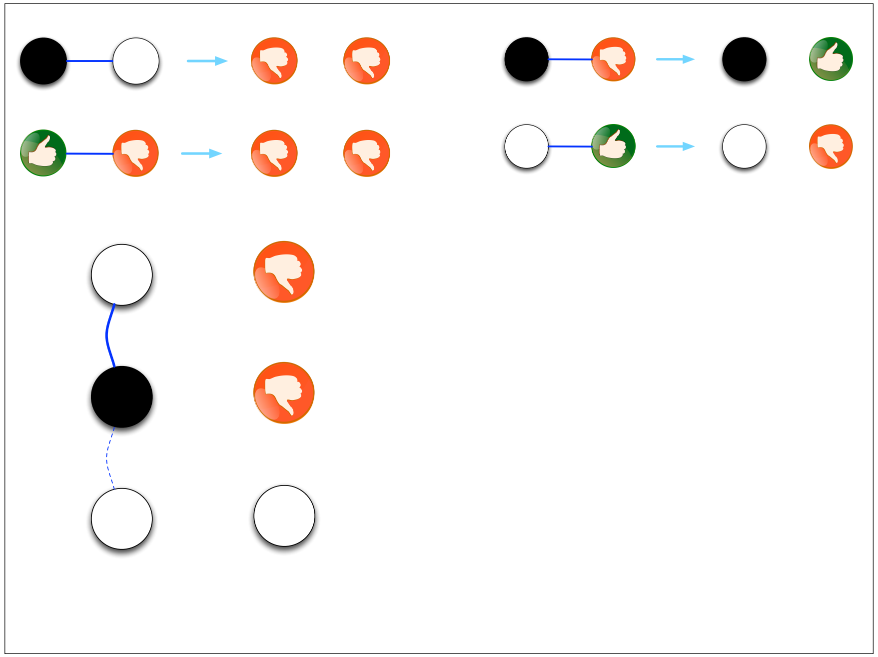
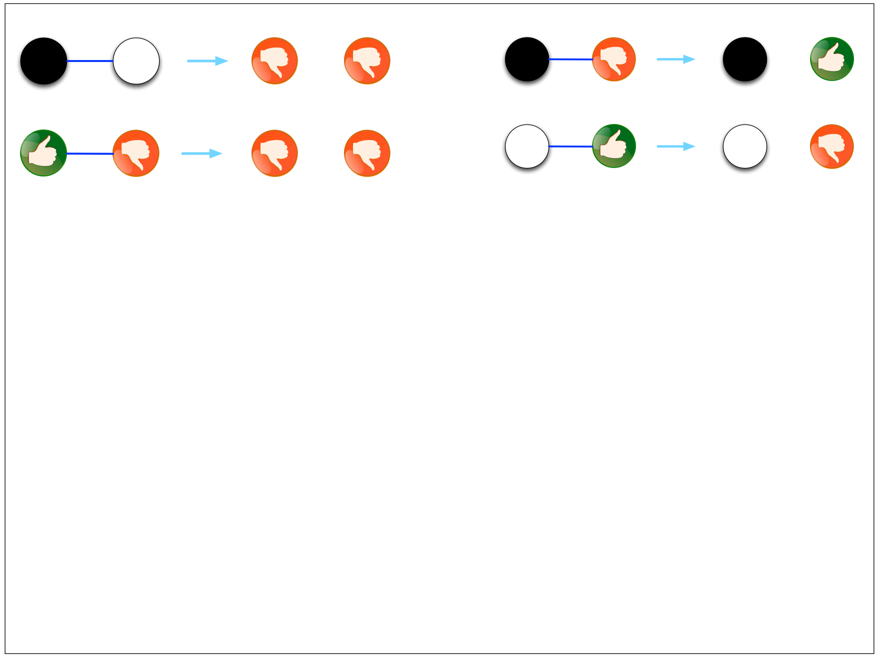


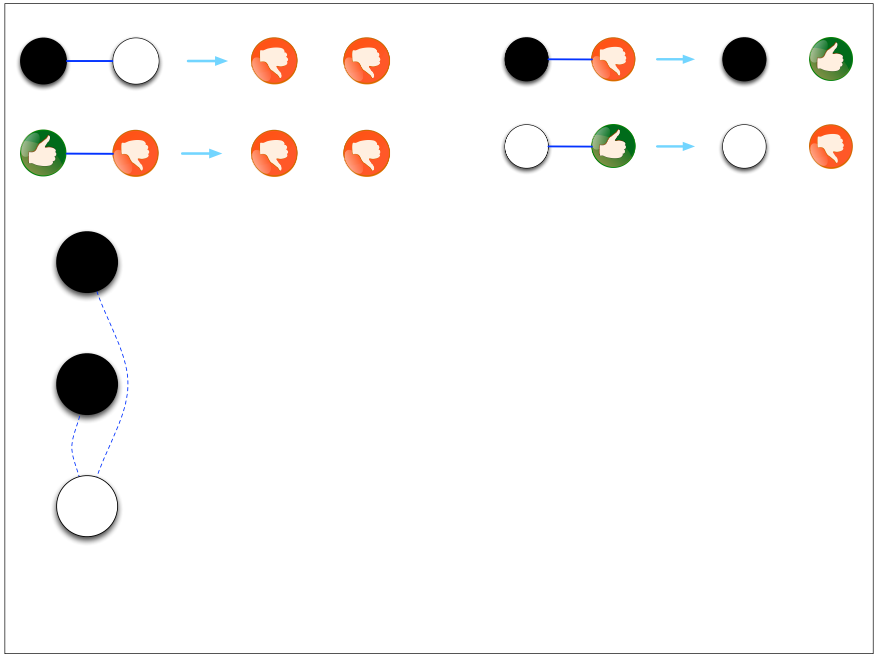
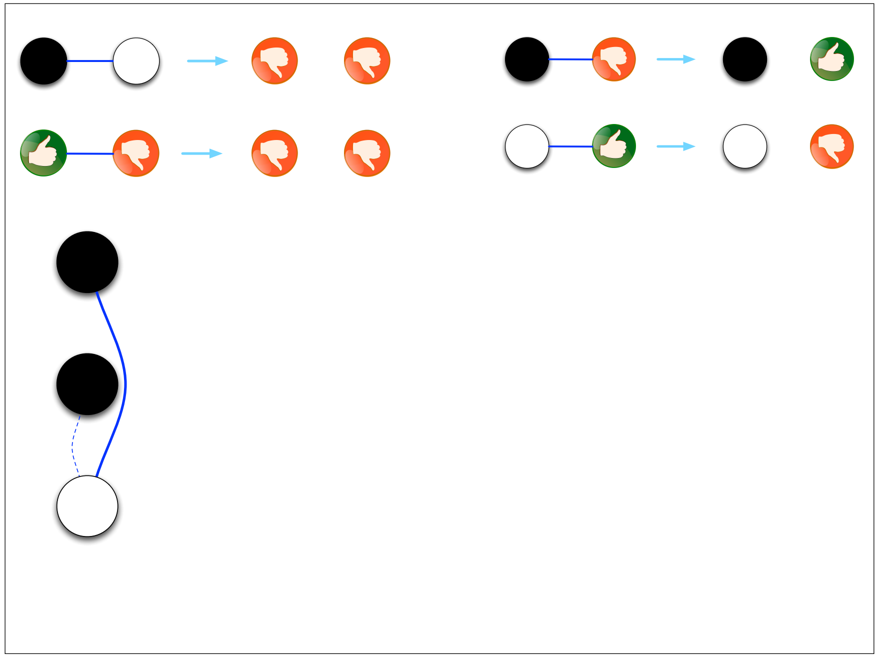
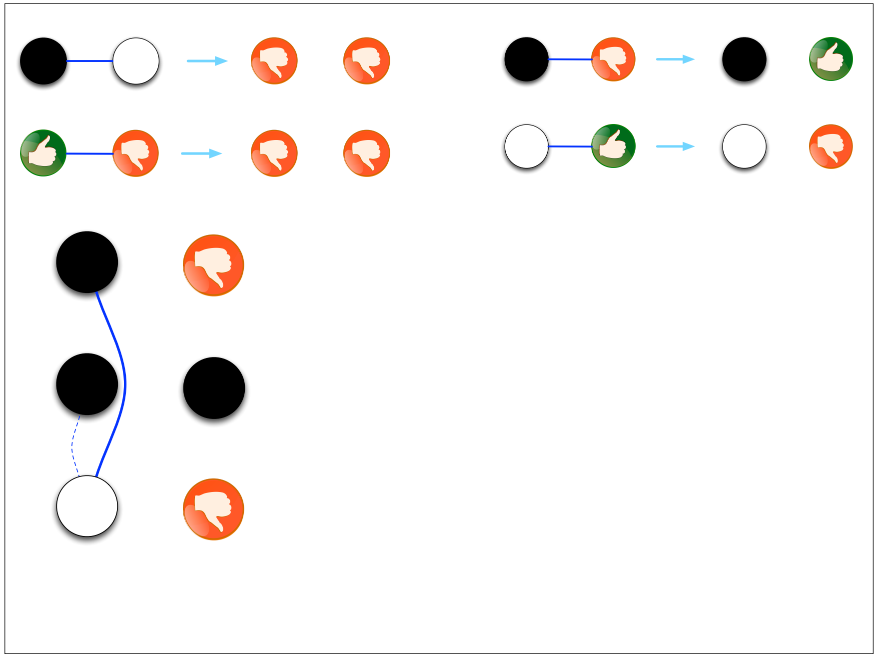
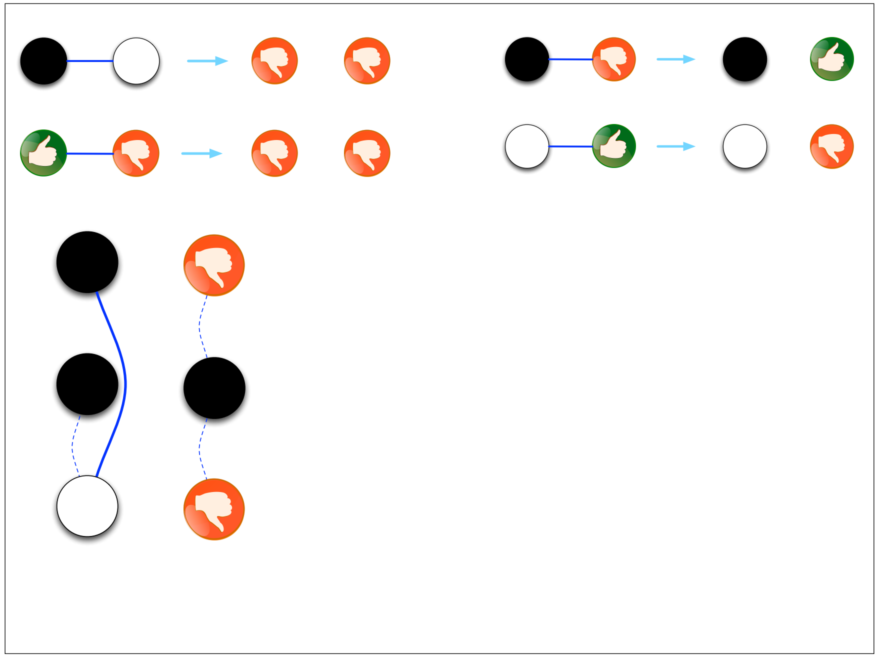
Example 2

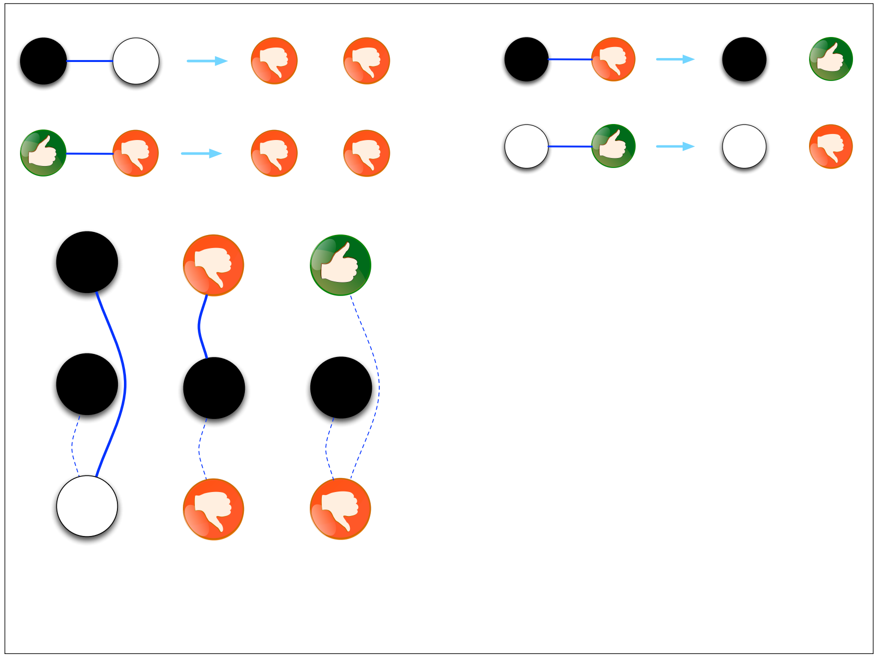
- **Inputs**: 
- **Outputs**: 
- **#**  **> #**  **?**

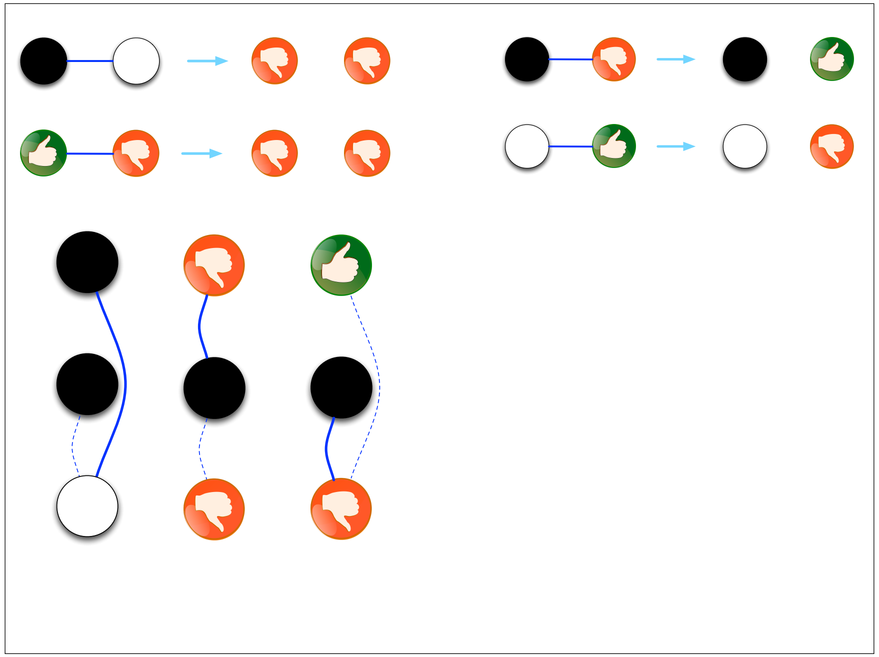
Example 2

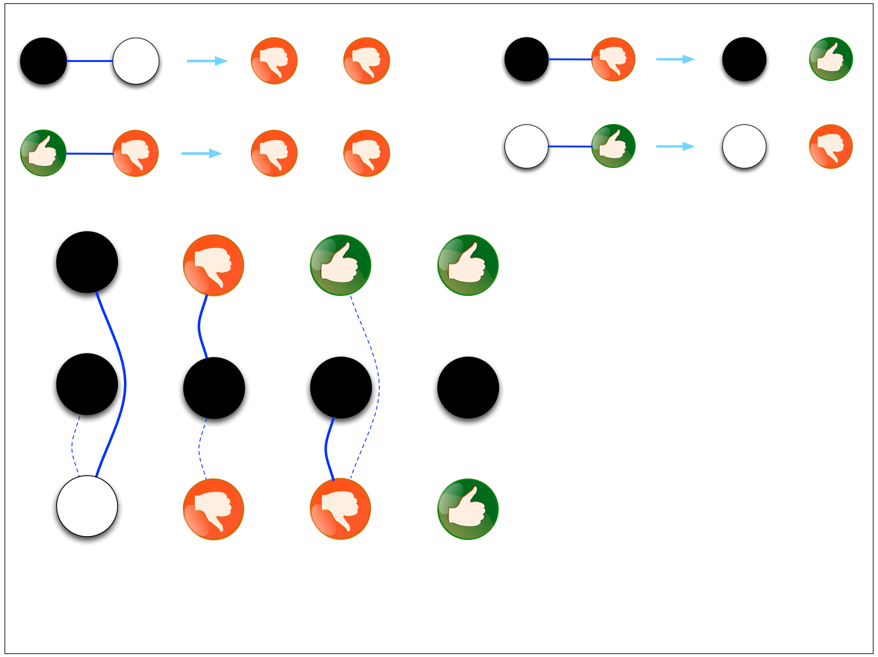


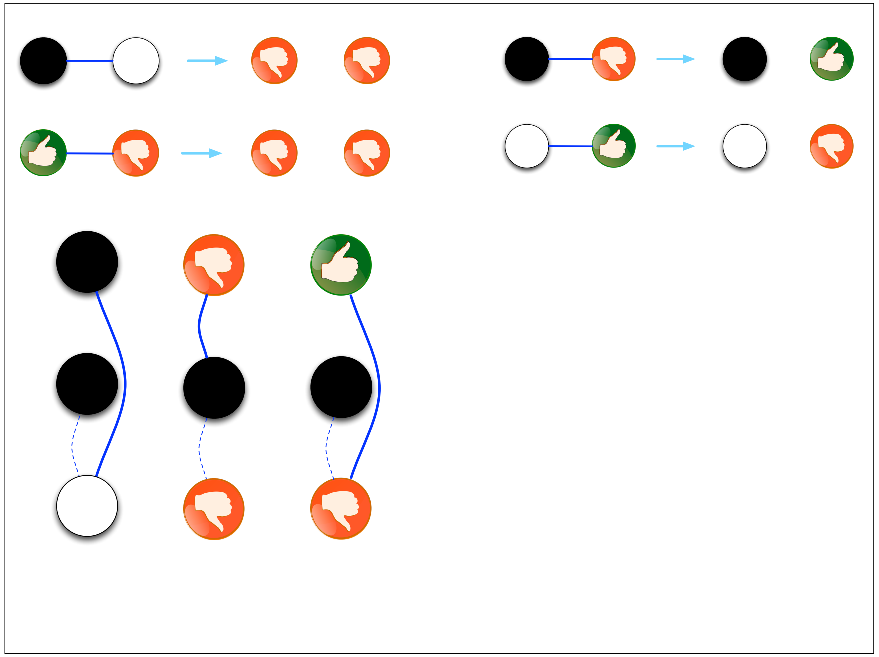





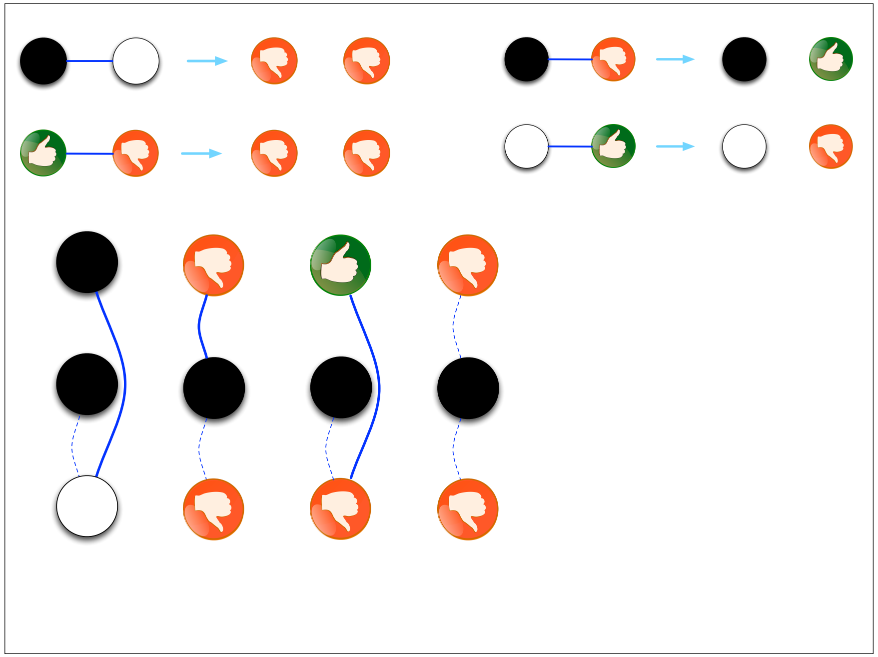





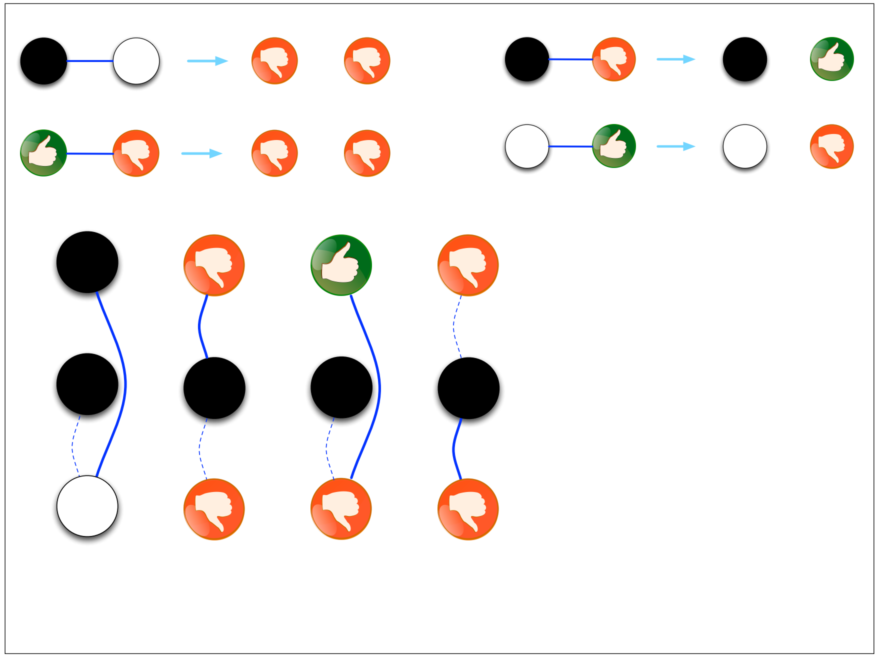


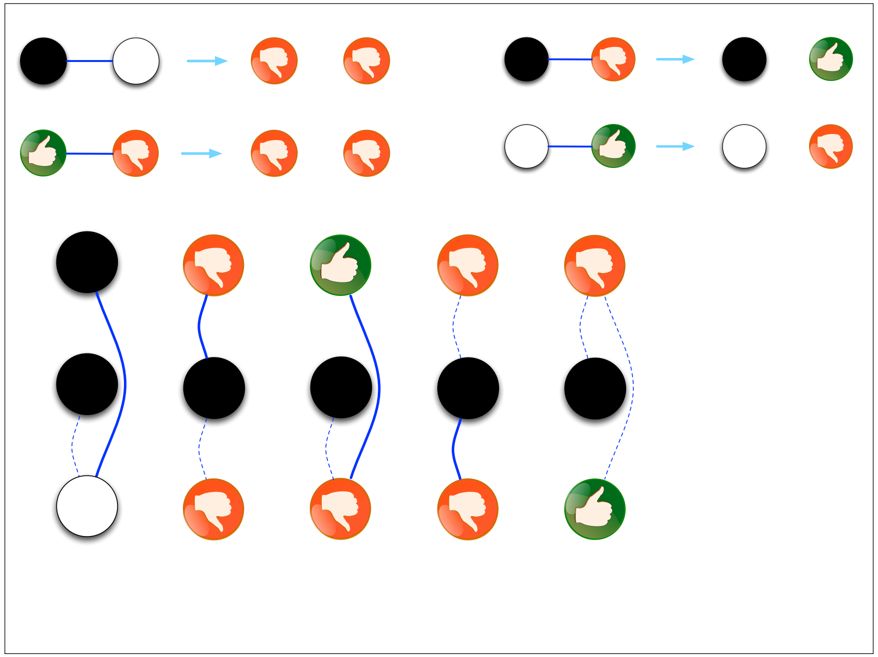


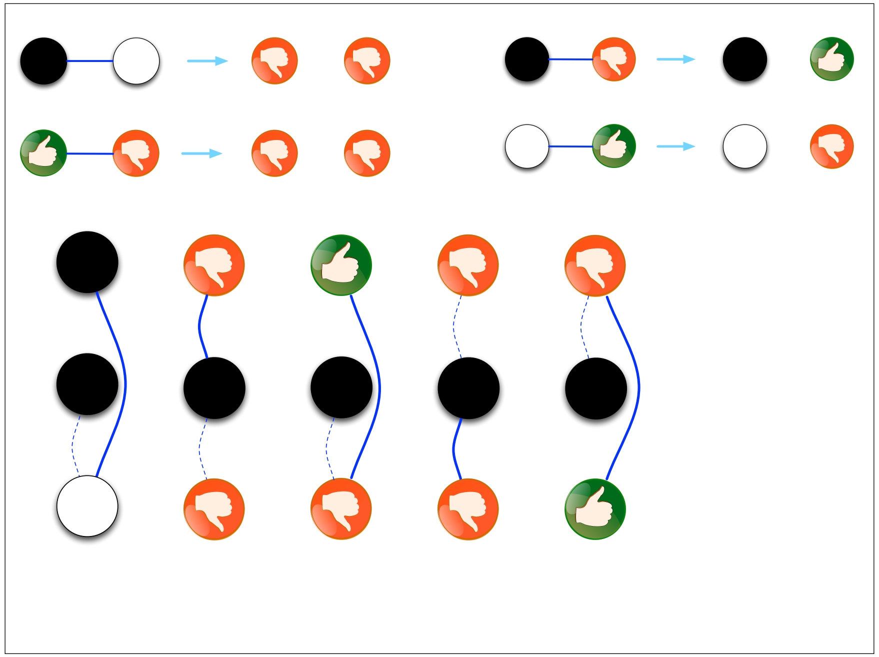


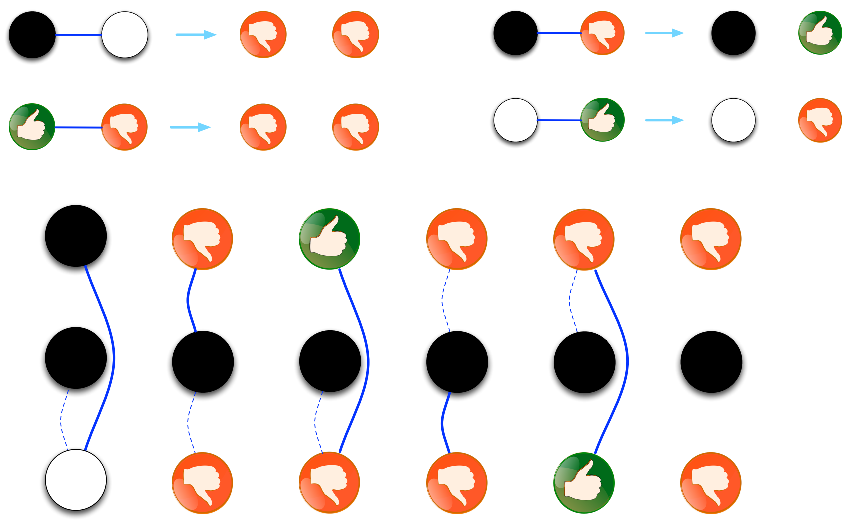








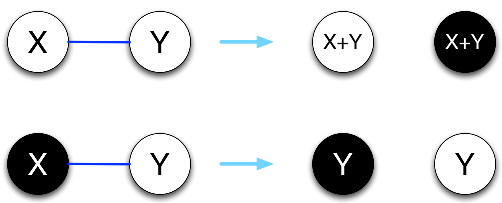


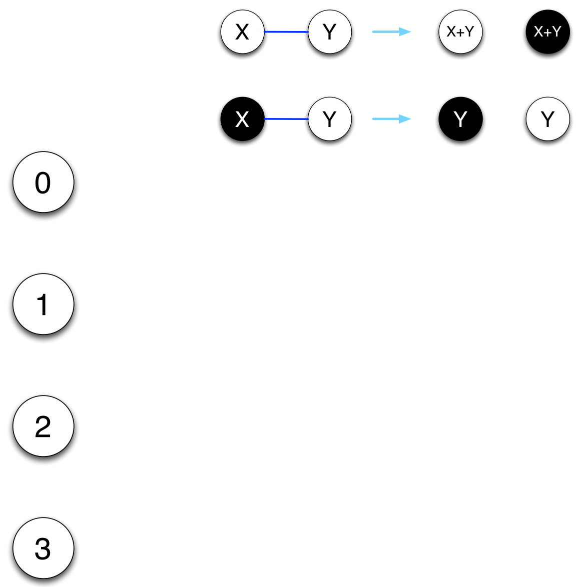


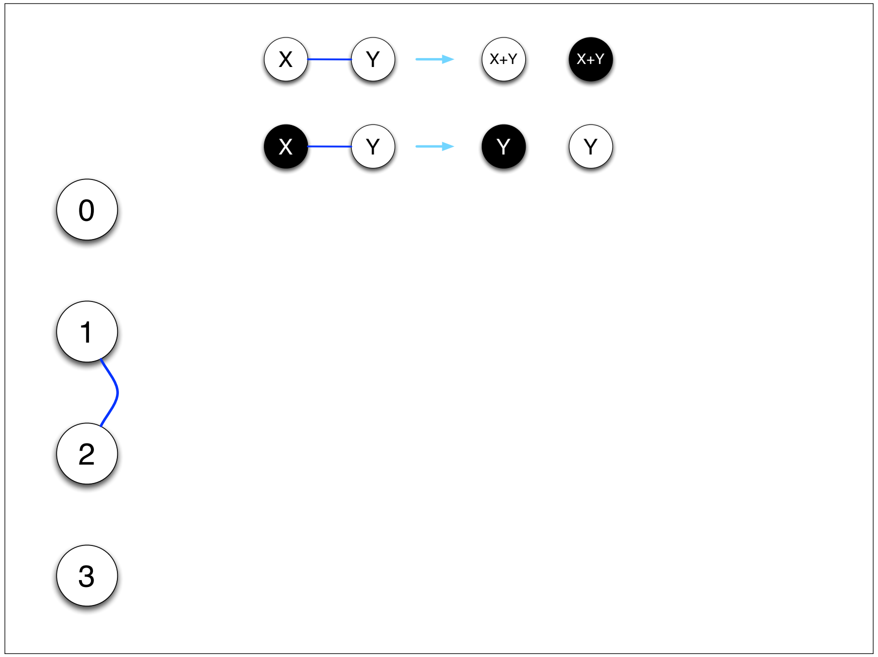
Example 3

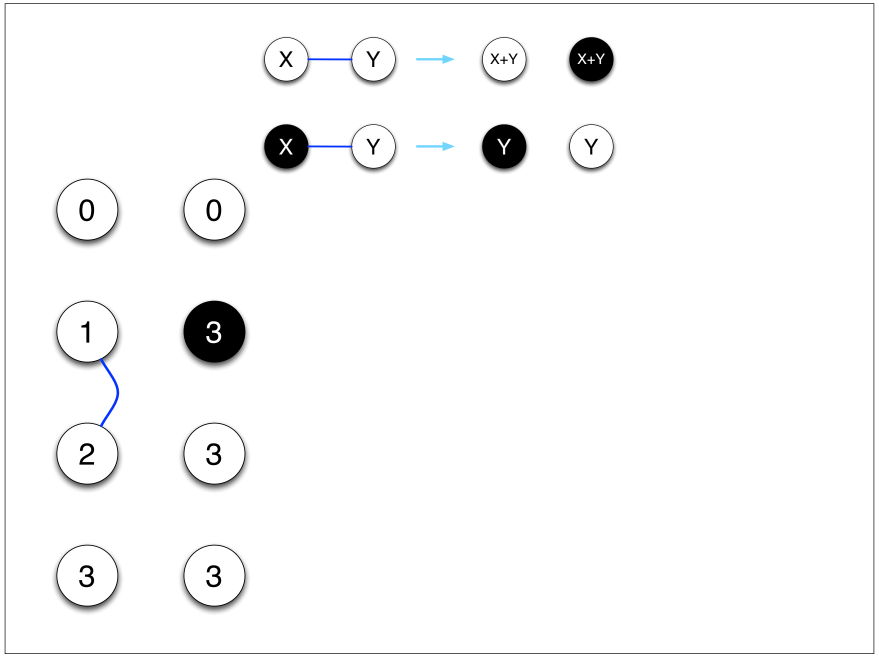
- **Inputs**: 0 1 2 3
- **Outputs**: 0 1 2 3 0 1 2 3
- **Sum mod 4 ?**

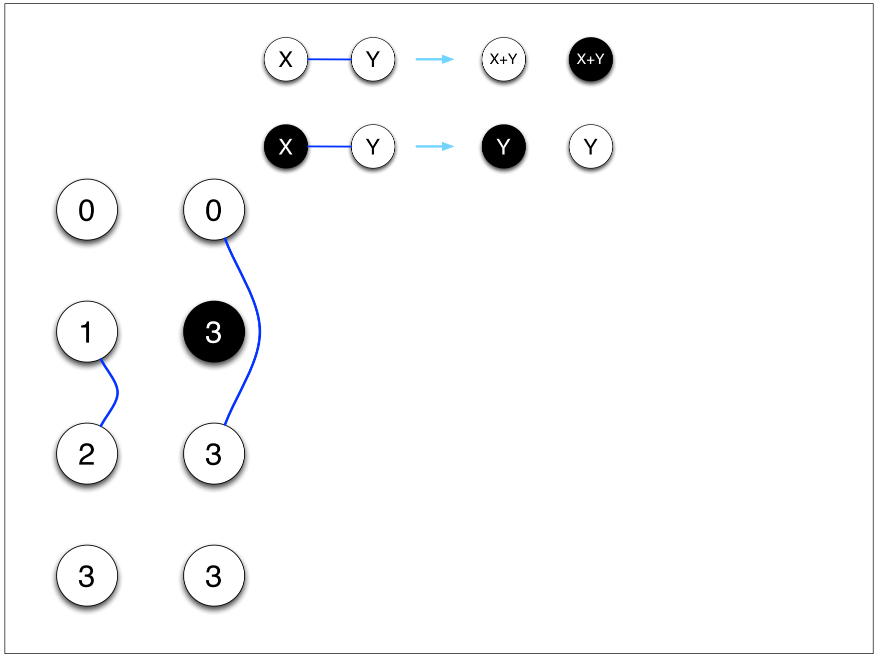
Example 3

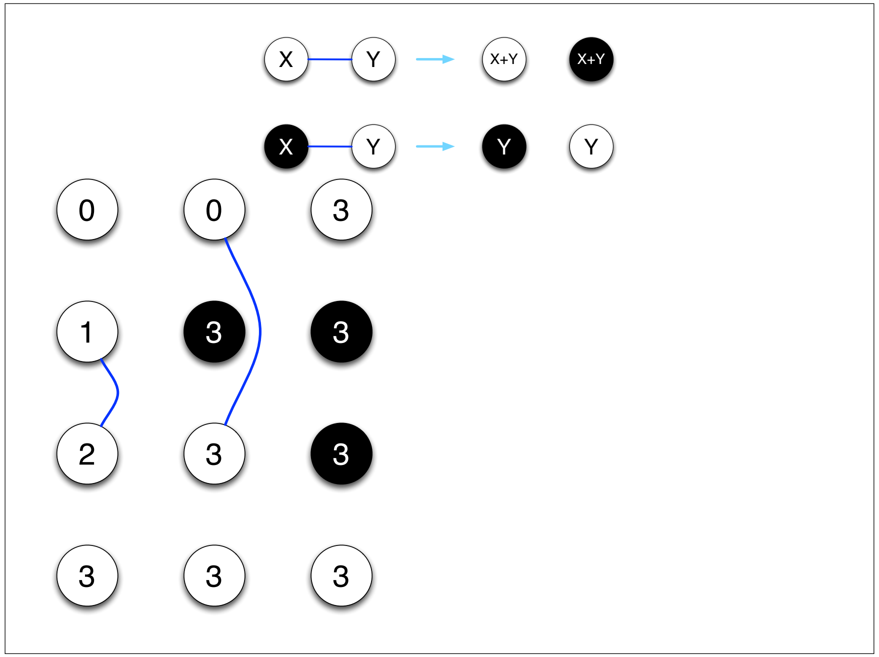


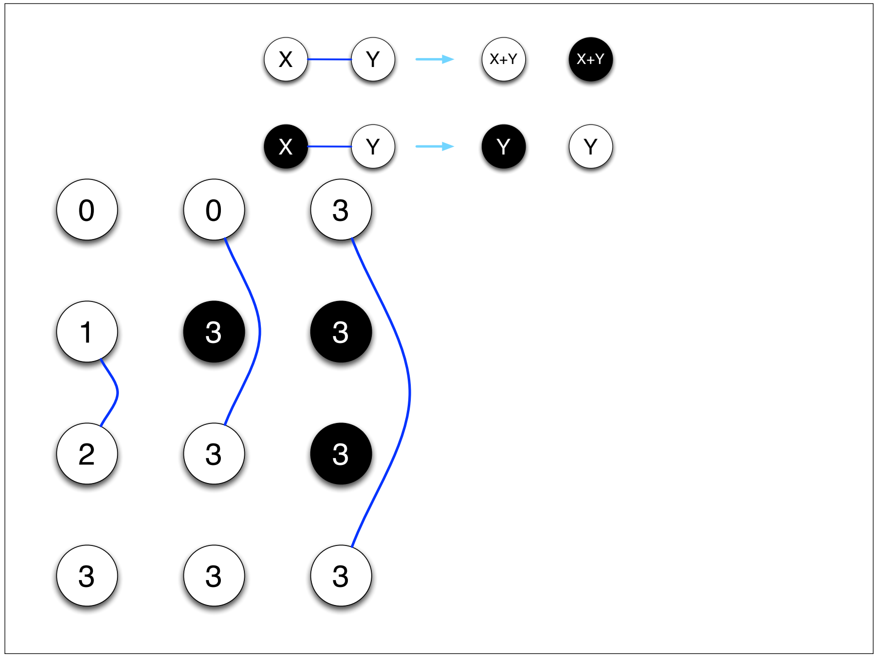


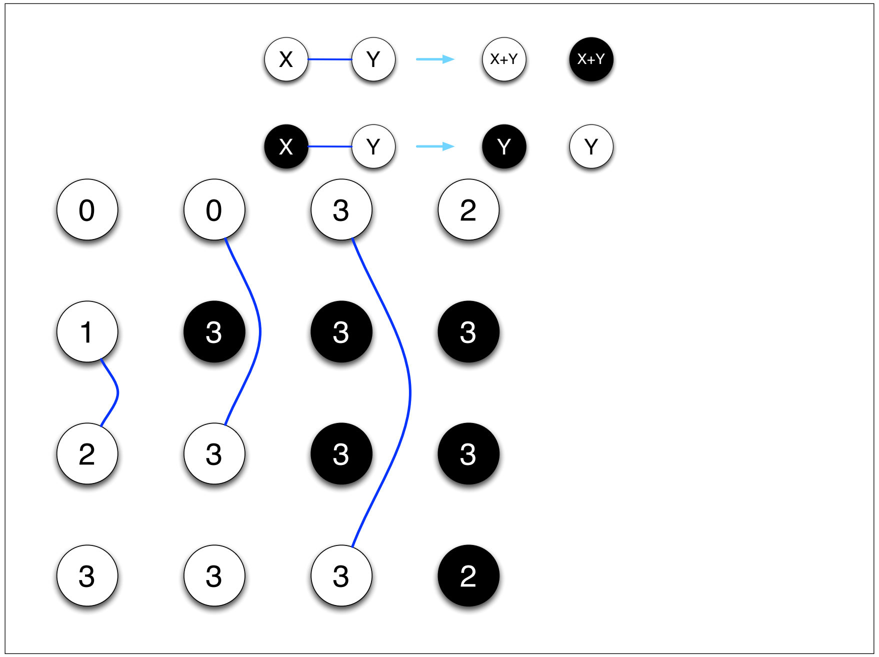

X Y X+Y X+Y
X Y Y Y
0
1
2
3

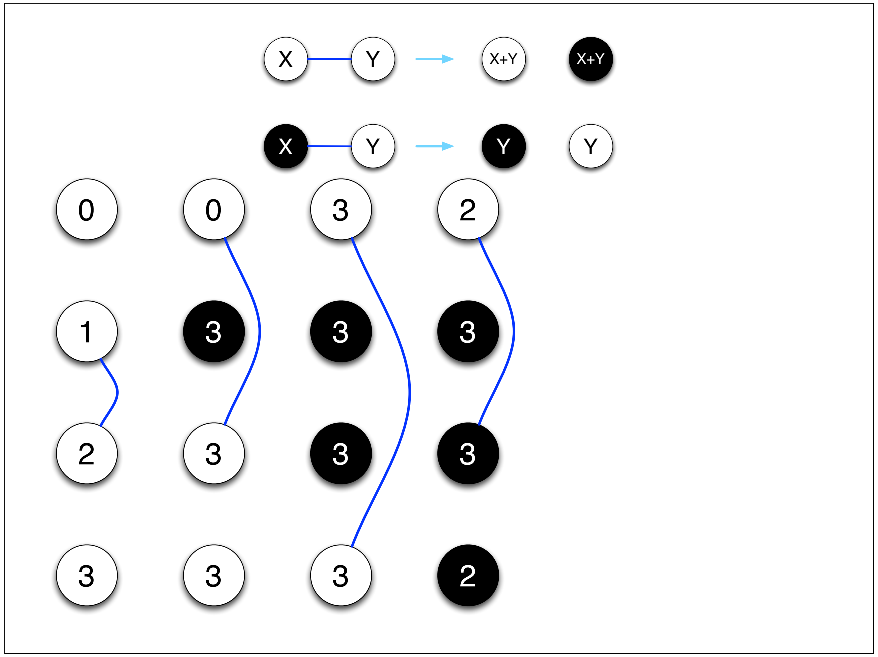

X Y X+Y X+Y
X Y Y Y
0 0
1 3
2 3
3 3

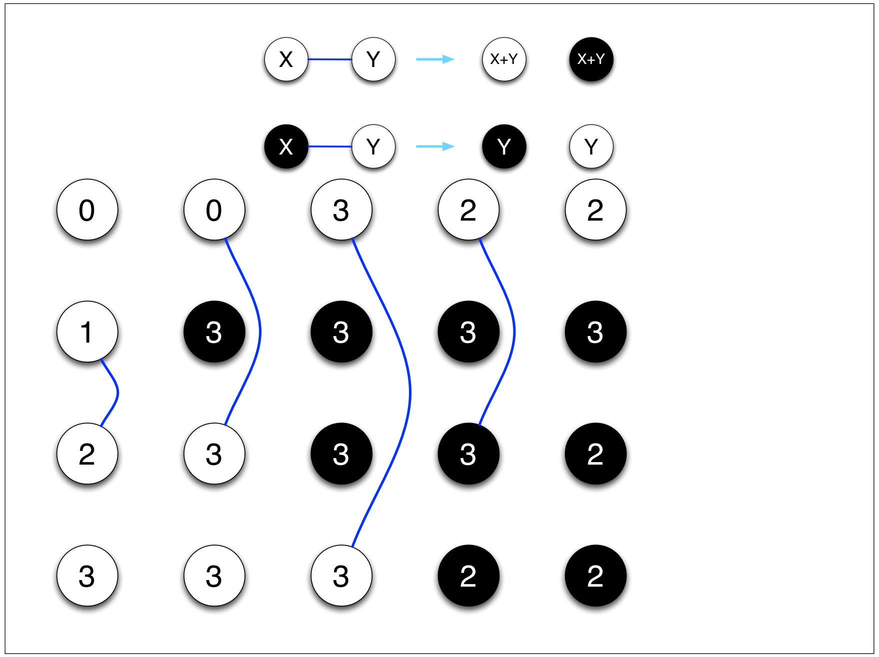

X Y X+Y X+Y
X Y Y Y
0 0
1 3
2 3
3 3

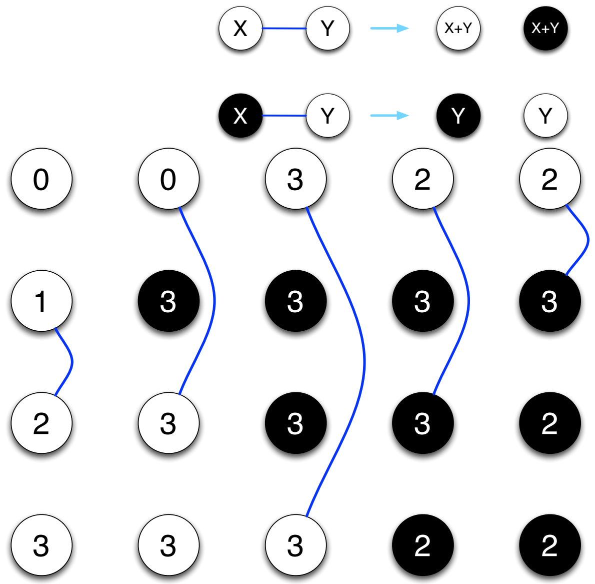

X Y X+Y X+Y
X Y Y Y
0 0 3
1 3 3
2 3 3
3 3 3

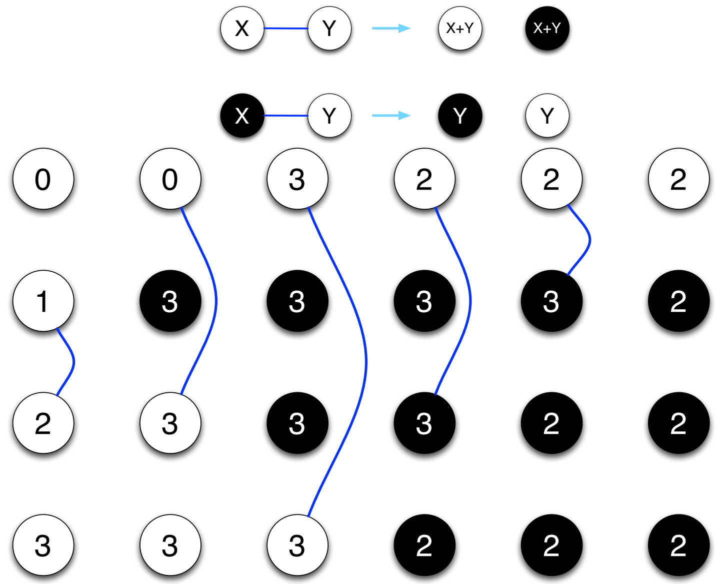

X Y X+Y X+Y
X Y Y Y
0 0 3
1 3 3
2 3 3
3 3 3


X Y X+Y X+Y
X Y Y Y
0 0 3 2
1 3 3 3
2 3 3 3
3 3 3 2

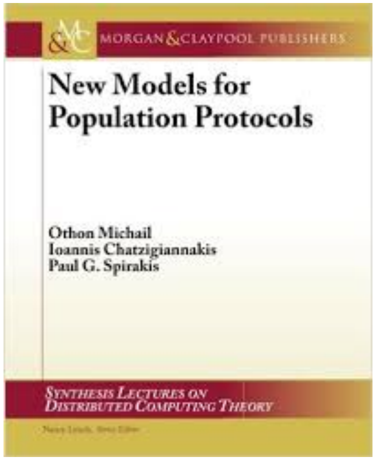

X Y X+Y X+Y
X Y Y Y
0 0 3 2
1 3 3 3
2 3 3 3
3 3 3 2


X Y X+Y X+Y
X Y Y Y
0 0 3 2 2
1 3 3 3 3
2 3 3 3 2
3 3 3 2 2





Population Protocols



Dynamic Graphs

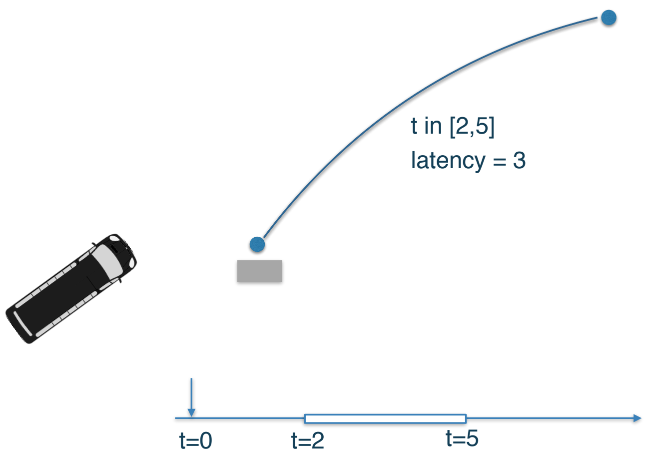
Time-varying Graphs

- A *time-varying graph* (TVG) is a 5-tuple (**V**,**E**,**T**,**p**,**l**)
  - **V**: set of nodes
  - **E**: (labelled) set of edges
  - **T**: lifetime, $\mathbf{T} \subseteq \mathcal{T}$
  - **p**: presence function, $\mathbf{E} \times \mathbf{T} \longrightarrow \{0,1\}$
  - **l**: latency function, $\mathbf{E} \times \mathbf{T} \longrightarrow \mathcal{T}$

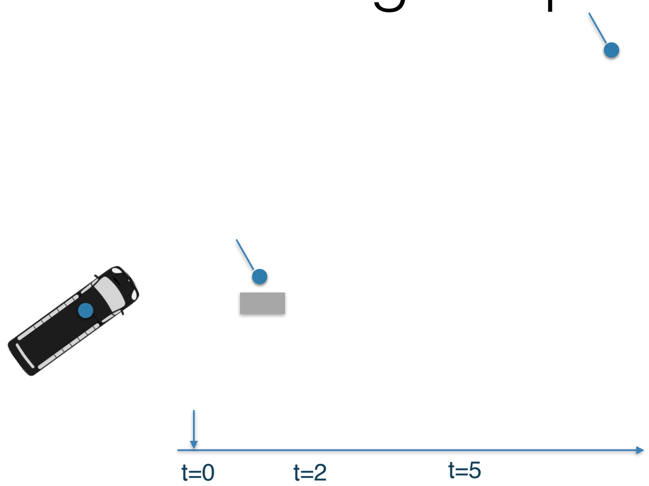
Time-varying Graphs

- A *time-varying graph* (TVG) is a 5-tuple (**V**,**E**,**T**,**p’**,**l’**)
  - **V**: set of nodes
  - **E**: (labelled) set of edges
  - **T**: lifetime, $\mathbf{T} \subseteq \mathcal{T}$
  - **p’**: *node* presence function, $\mathbf{V} \times \mathbf{T} \longrightarrow \{0,1\}$
  - **l’**: *node* latency function, $\mathbf{V} \times \mathbf{T} \longrightarrow \mathcal{T}$

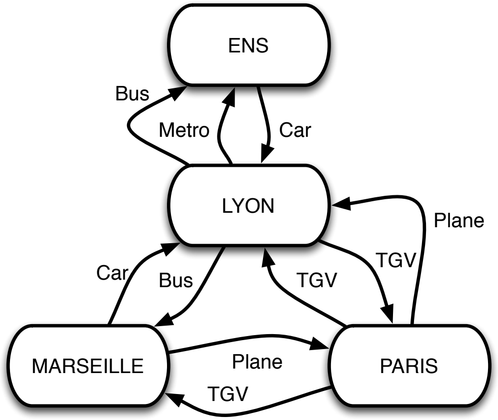
Time-varying Graphs



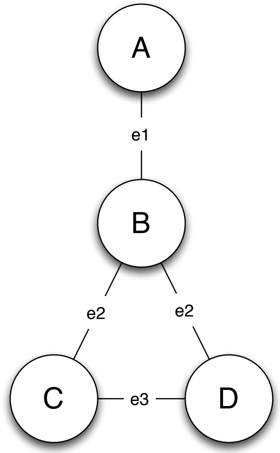
Evolving Graphs



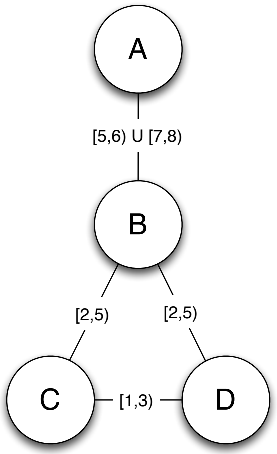
Example

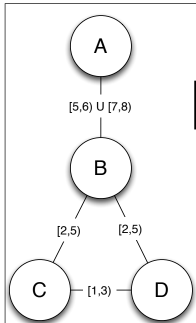


Example



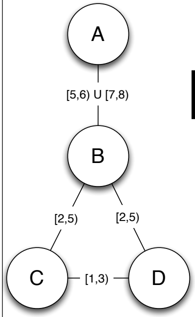
Example

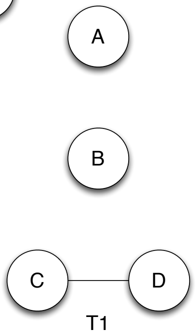




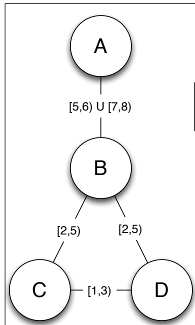
Evolving Graphs

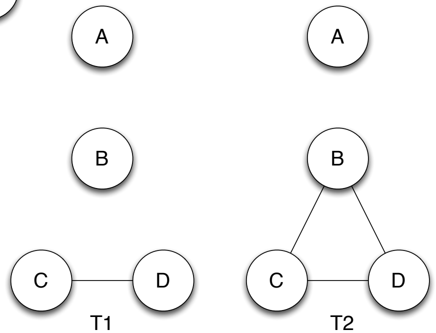
Evolving Graphs



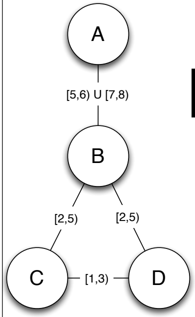


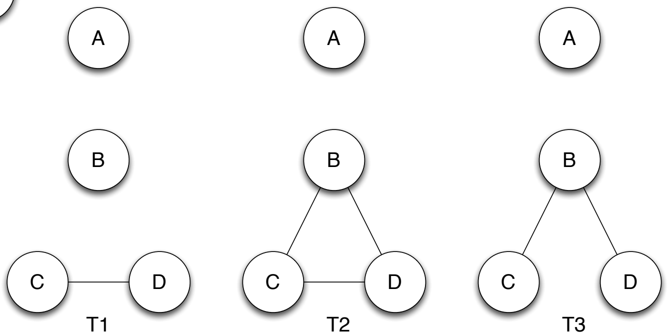
Evolving Graphs



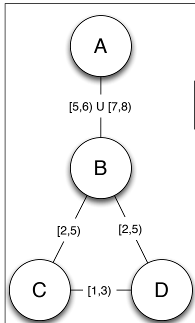


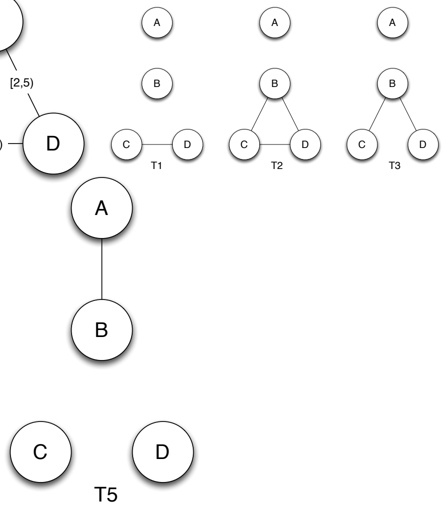
Evolving Graphs



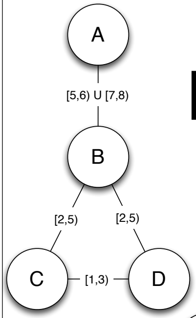


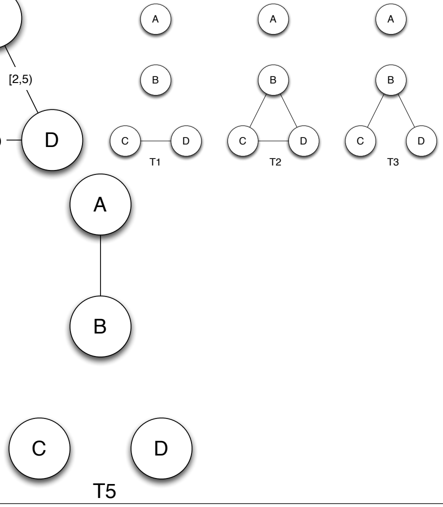
Evolving Graphs



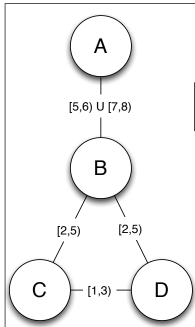


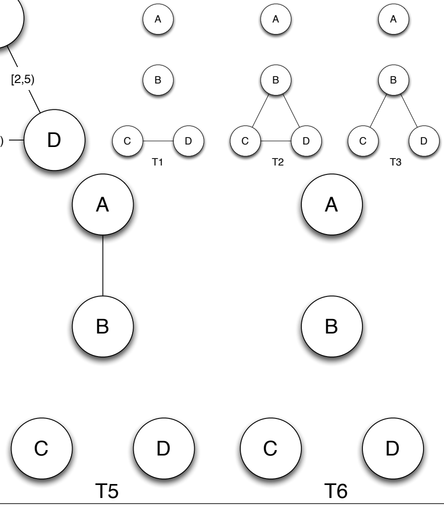
Evolving Graphs



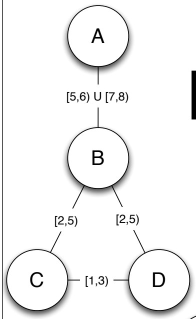


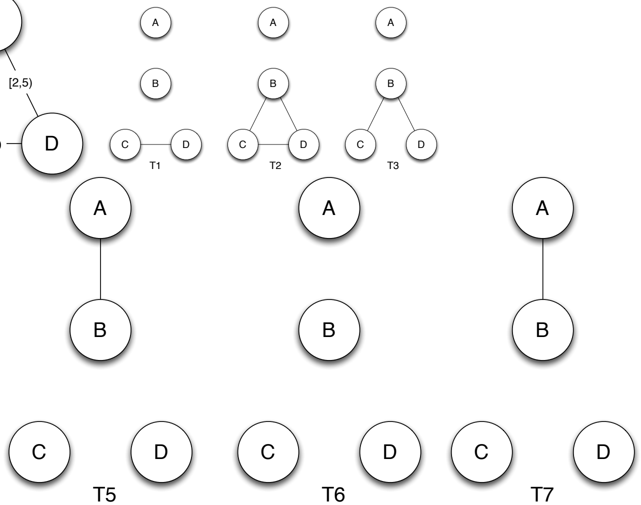
Evolving Graphs



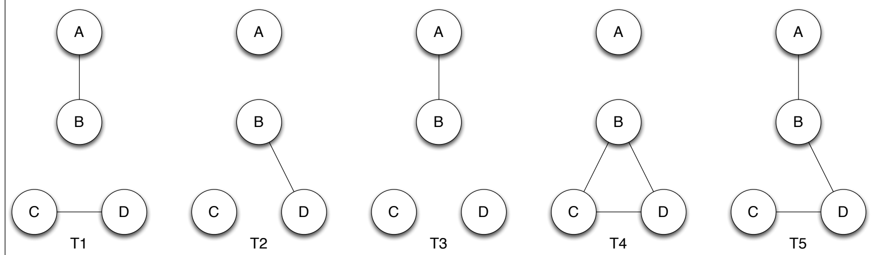


Evolving Graphs

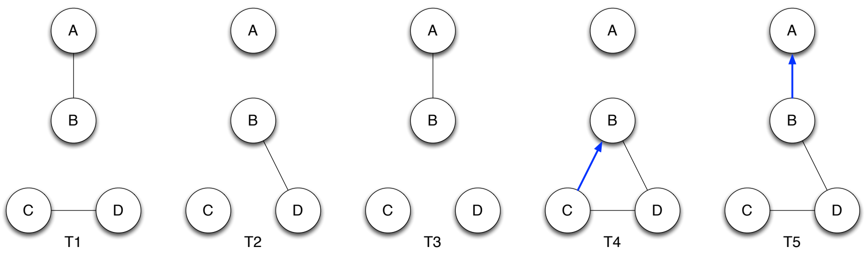




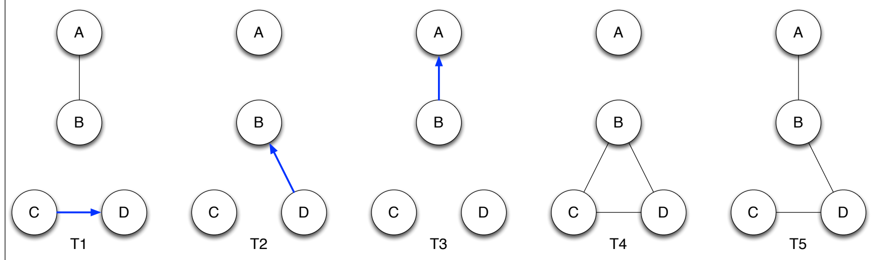
Journeys from C to A



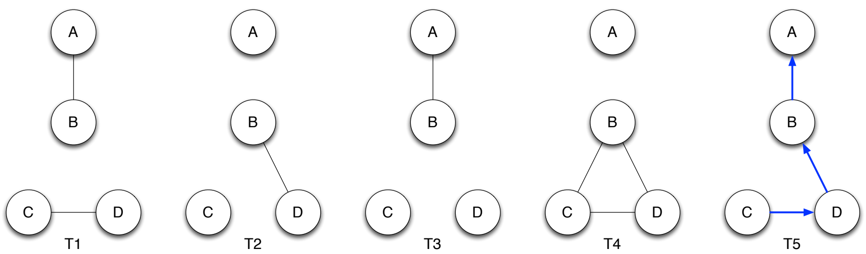
Shortest Journey



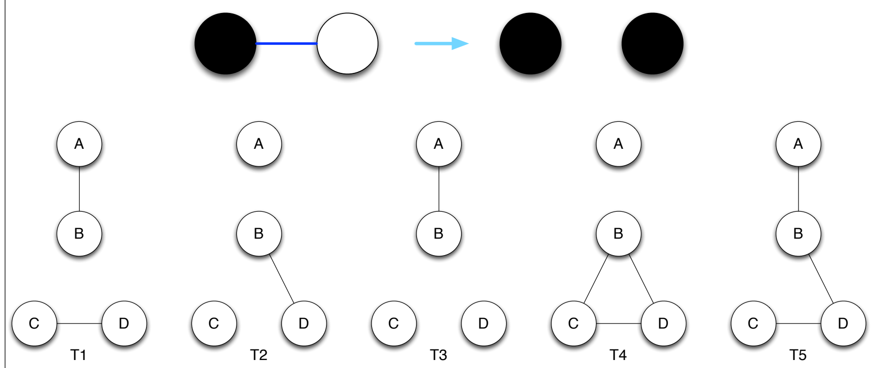
Foremost Journey



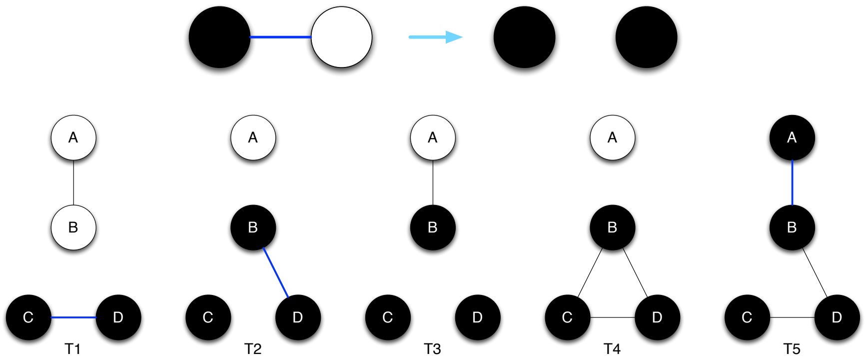
Fastest Journey



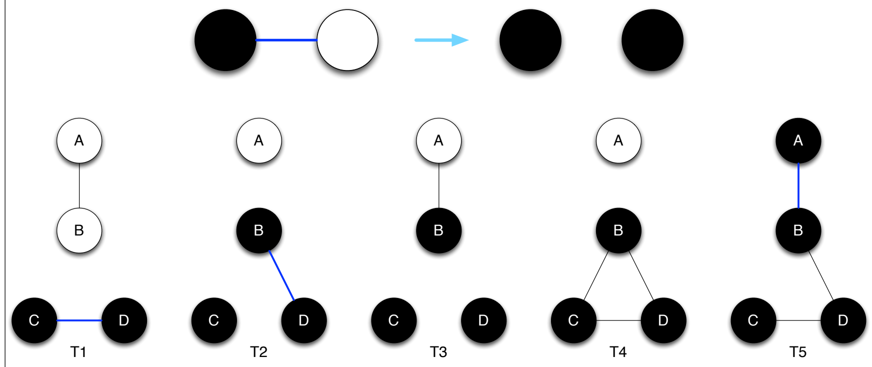
Condition for Broadcast?



Condition for Broadcast?

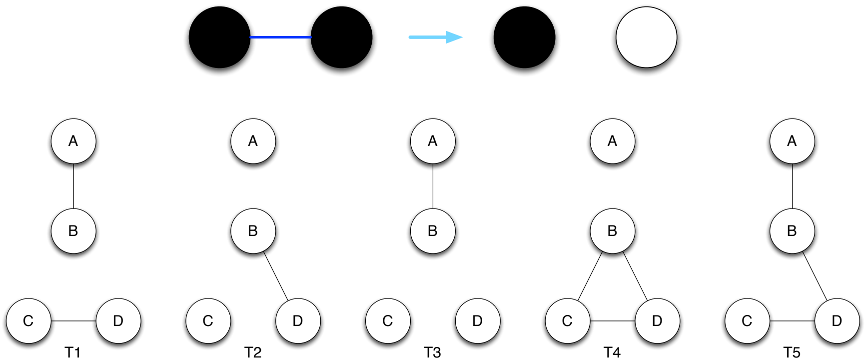


Condition for Broadcast?

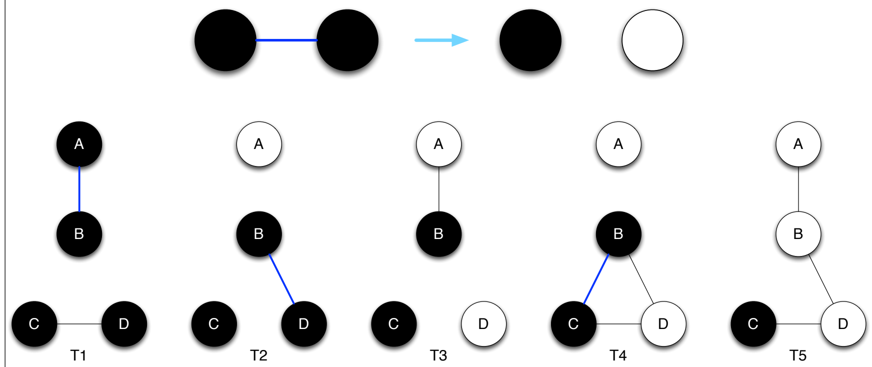


There exists a node (**C**) from which a journey reaches every other node

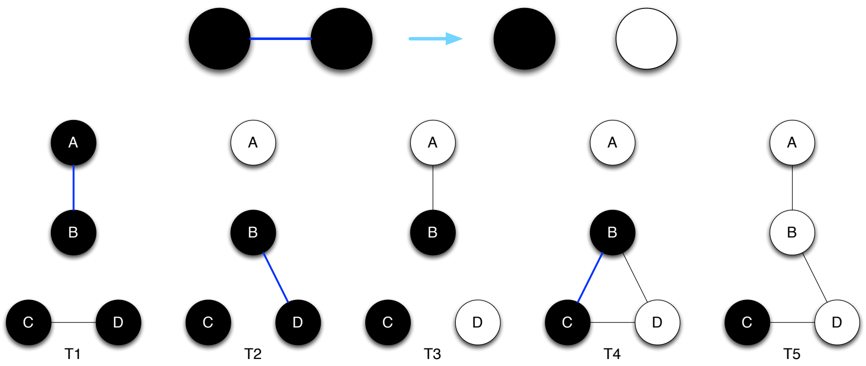
Condition for Election?



Condition for Election?

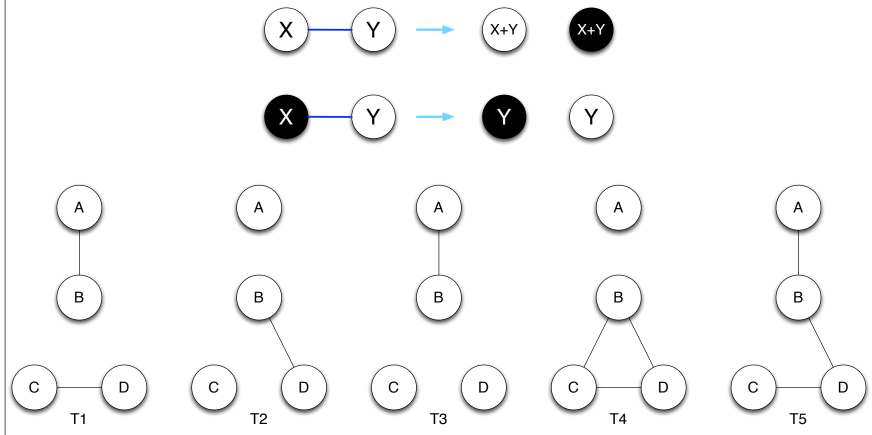


Condition for Election?

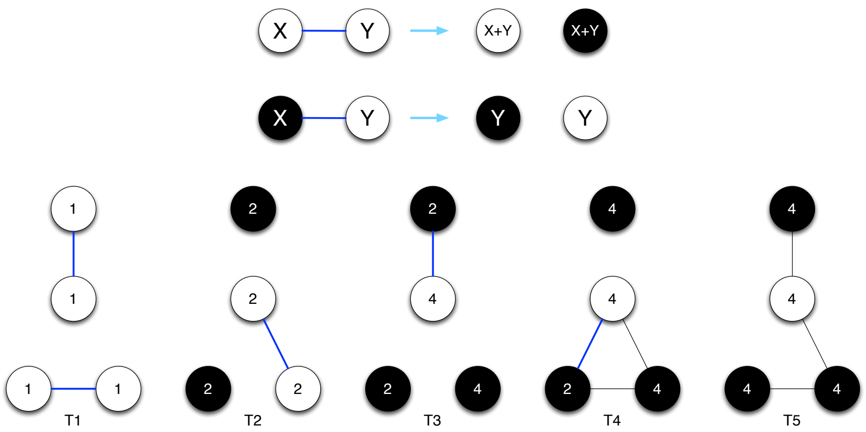


There exists a node (**C**) such that there exists a journey from every other node to it

Condition for Global Calculus?

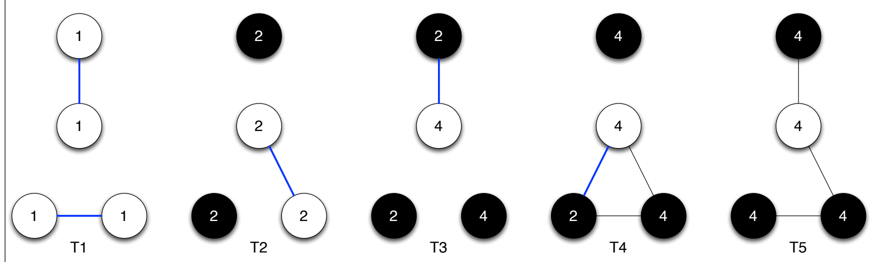


Condition for Global Calculus?



Condition for Global Calculus?

There exists a node (**Center**) such that there exists a journey from every other node to it *and back*



Connectivity Classes

- There exists a node *r* from which a journey reaches every other node $1 \rightsquigarrow *$
- There exists a node *r* such that there exists a journey from every other node to it $* \rightsquigarrow 1$
- There exists a node *r* such that there exists a journey from every other node to to and back $1 \overset{\leftsquigarrow}{\rightsquigarrow} *$

Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, Nicola Santoro: Time-varying graphs and dynamic networks. IJPEDS 27(5): 387-408 (2012)

More Classes

- There exists a journey between any two nodes $* \rightsquigarrow *$
- There exists a roundtrip journey between any two nodes $* \overset{\rightsquigarrow}{\leftsquigarrow} *$
- There exists a journey between any two nodes infinitely often $* \overset{\mathcal{R}}{\rightsquigarrow} *$
- Every edge appears infinitely often $\bullet \overset{\mathcal{R}}{-} \bullet$

Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, Nicola Santoro: Time-varying graphs and dynamic networks. IJPEDS 27(5): 387-408 (2012)

More Classes

- Every edge appears infinitely often, and there is an upper bound between between two occurrences $\bullet \overset{\mathcal{B}}{-} \bullet$
- Every edge appears infinitely often with some period *p* $\bullet \overset{\mathcal{P}}{-} \bullet$

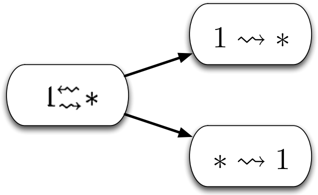
Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, Nicola Santoro: Time-varying graphs and dynamic networks. IJPEDS 27(5): 387-408 (2012)

More Classes

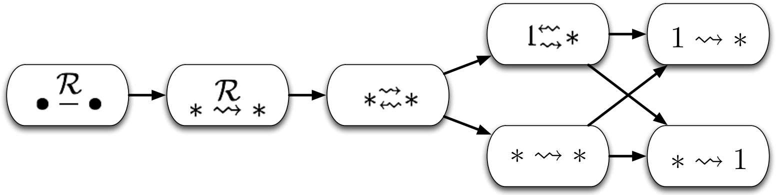
- At any time, the graph is connected
- Every spanning subgraph lasts at least T time units
- Every edge appears infinitely often, and the underlying graph is a clique $* \overset{\mathcal{R}}{-} *$

Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, Nicola Santoro: Time-varying graphs and dynamic networks. IJPEDS 27(5): 387-408 (2012)

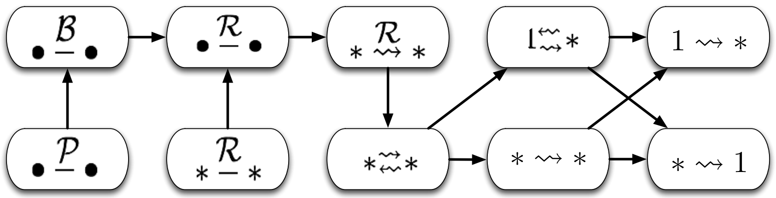
A Classification



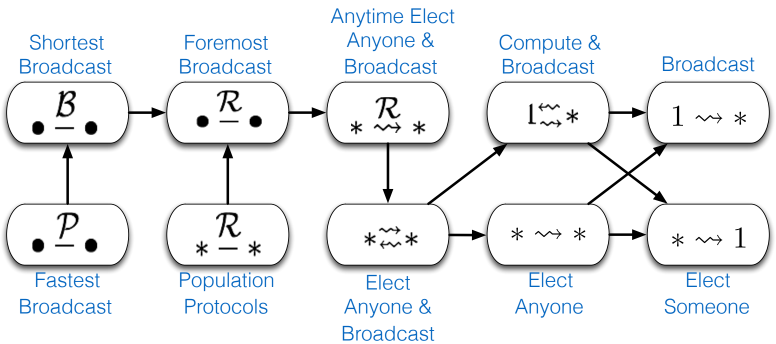
A Classification



A Classification



A Classification

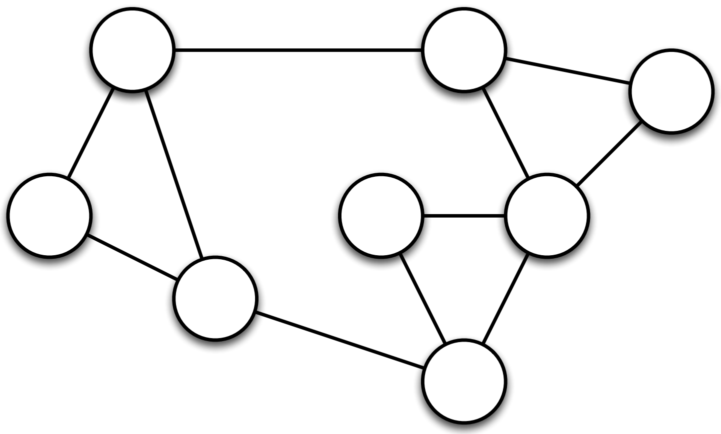


Arnaud Casteigts, Paola Flocchini, Bernard Mans, Nicola Santoro: Shortest, Fastest, and Foremost Broadcast in Dynamic Networks. Int. J. Found. Comput. Sci. 26(4): 499-522 (2015)

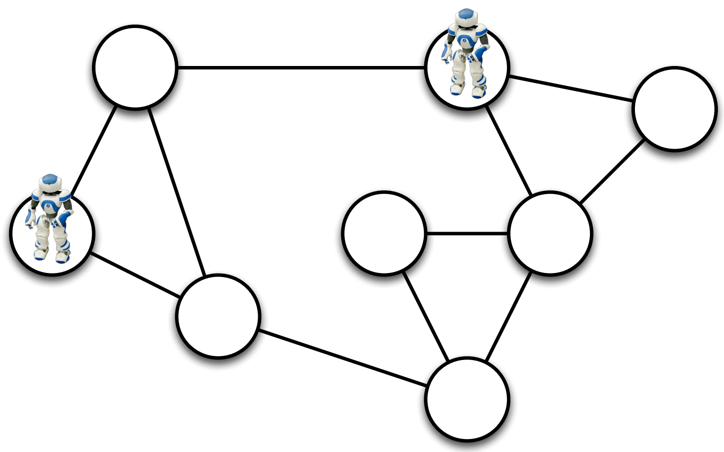
Actively Mobile Networks

Mobile Agents

Mobile Agents



Mobile Agents



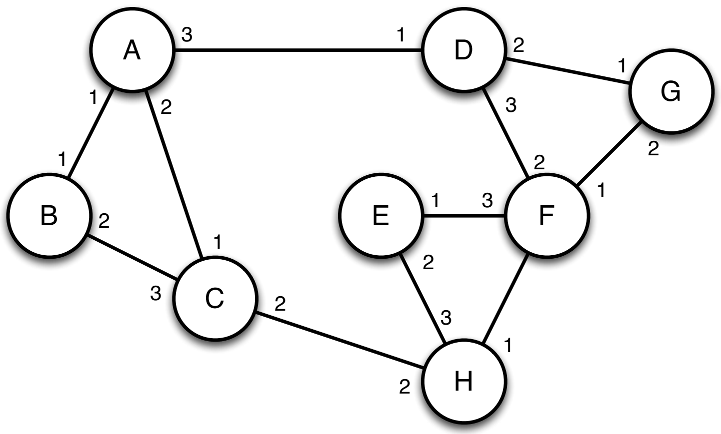
Problems to Solve

- **Exploration** (perpetual or with stop)
- **Mapping**
- **Rendez-vous**
- **Black hole search**
- **Capturing an intruder**

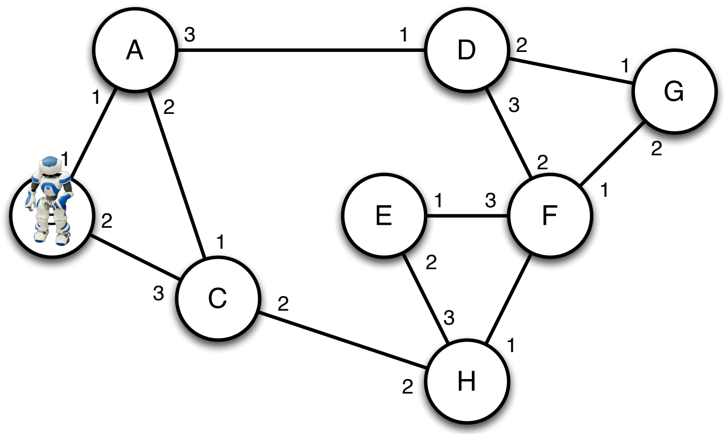
Models

- **Network** (anonymous *vs.* ID based)
- **Agents** (anonymous *vs.* ID based)
- **Synchrony**
- **Initial** (structural) **knowledge**
- **Communications** (none, peebles, whiteboards)
- Agent **memory** (infinite, bounded, constant)

Mapping



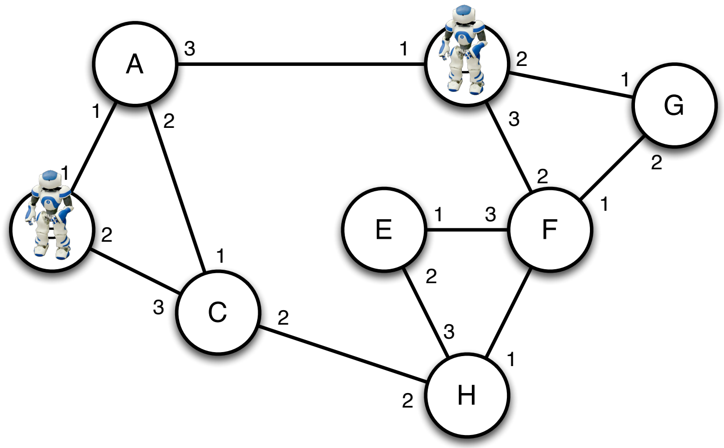
Mapping



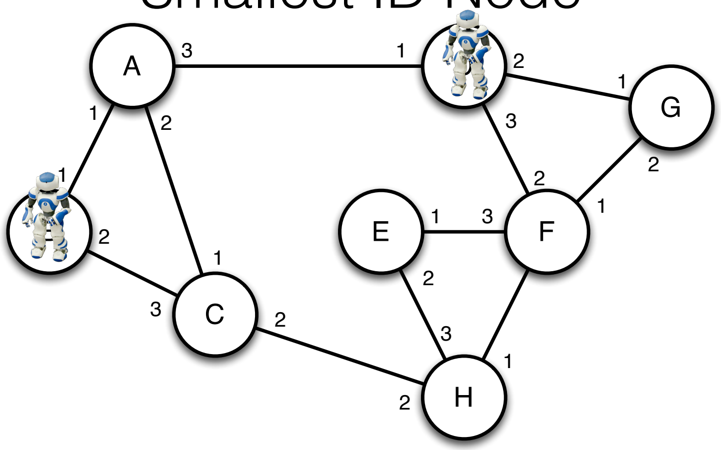
Rendez-vous

- **Two** (or more) **mobile agents must meet** in a graph
- They start on **distinct** locations
- **Computability**?
- **Complexity**?

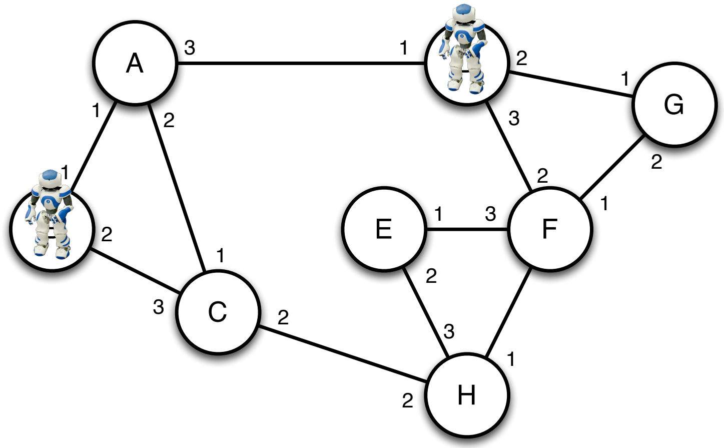
Rendez-vous in ID Graphs



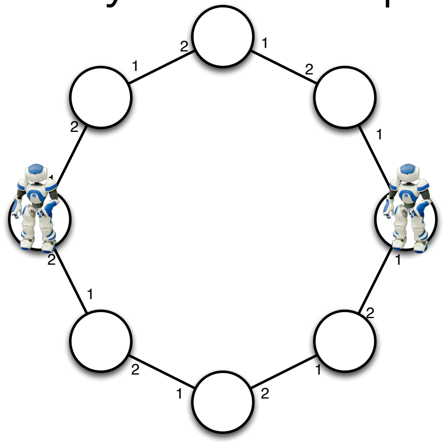
DFS to find Smallest ID Node



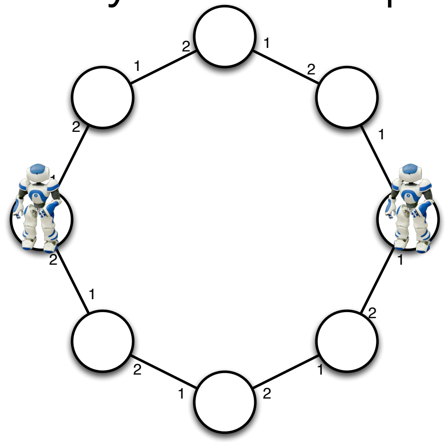
Rendez-vous in A



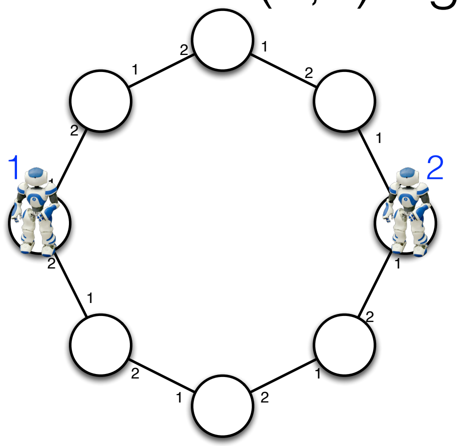
Rendez-vous in Anonymous Graphs



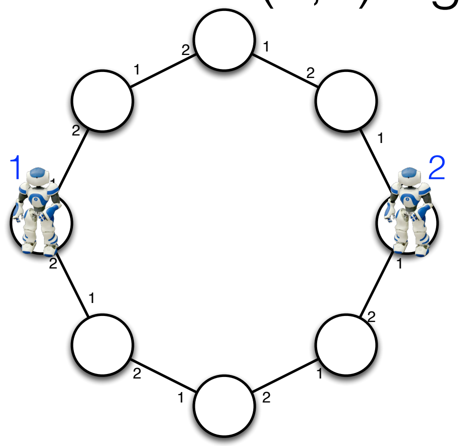
Rendez-vous in Anonymous Graphs



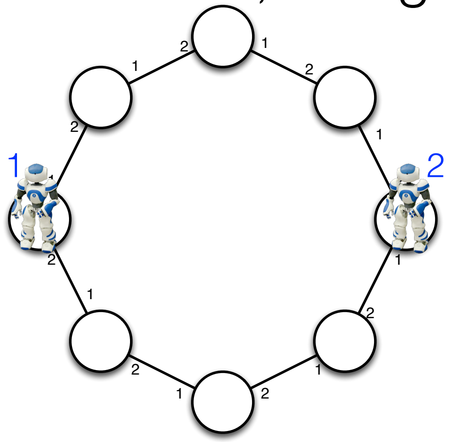
Anonymous Graphs with Known ID (1,2) Agents



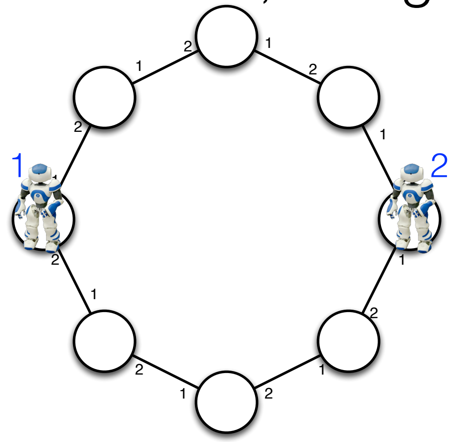
Anonymous Graphs with Known ID (1,2) Agents



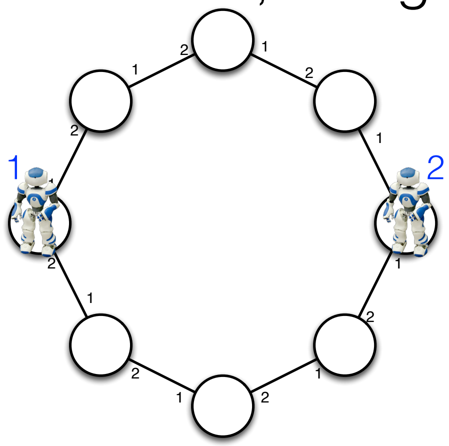
Anonymous Graphs with Known N, ID Agents



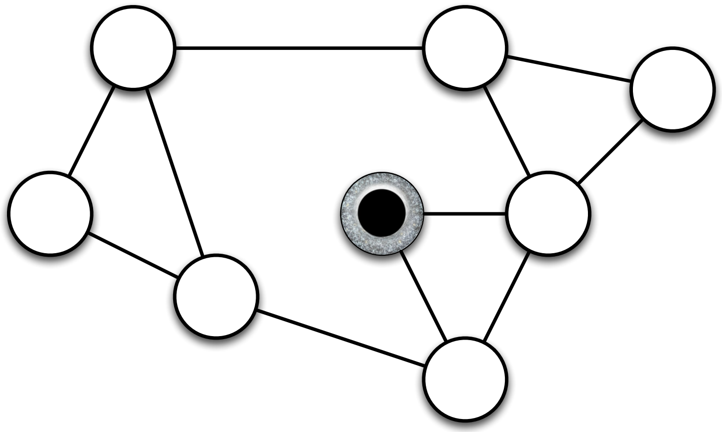
Anonymous Graphs with Known N, ID Agents



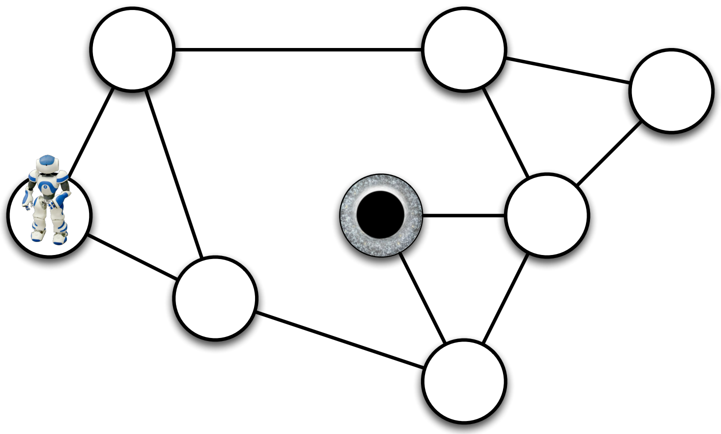
Anonymous Graphs with Known N, ID Agents



Black Hole Search



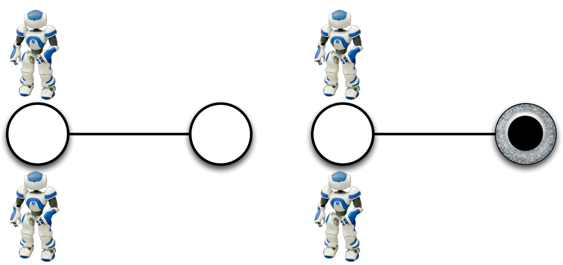
Black Hole Search



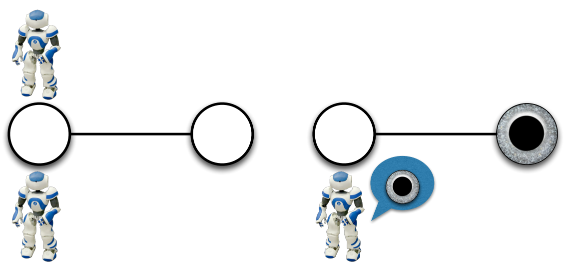
Black Hole Search

- A **single** black hole in the graph
- The black hole **does not disconnect** the graph
- Identify each **adjacent edge**
- **Minimize** #agents, #moves

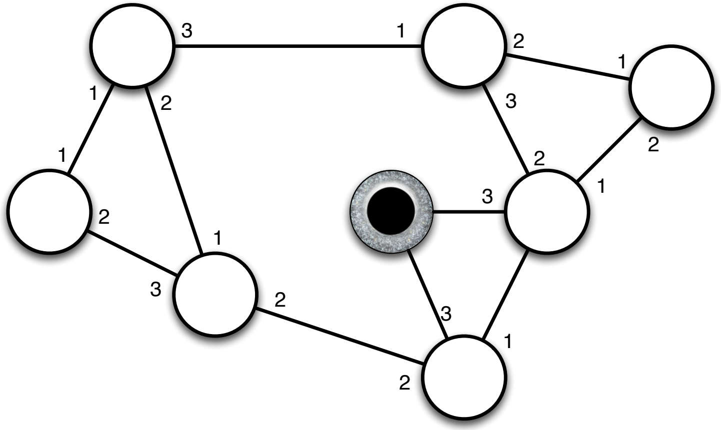
Synchronous Agents



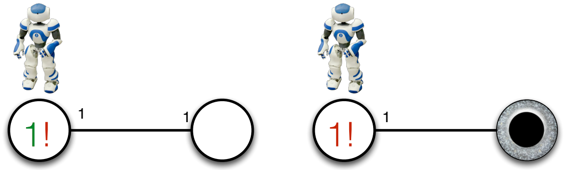
Synchronous Agents



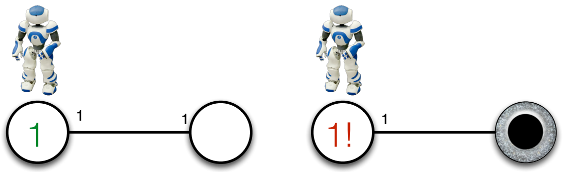
Asynchronous Black Hole Search



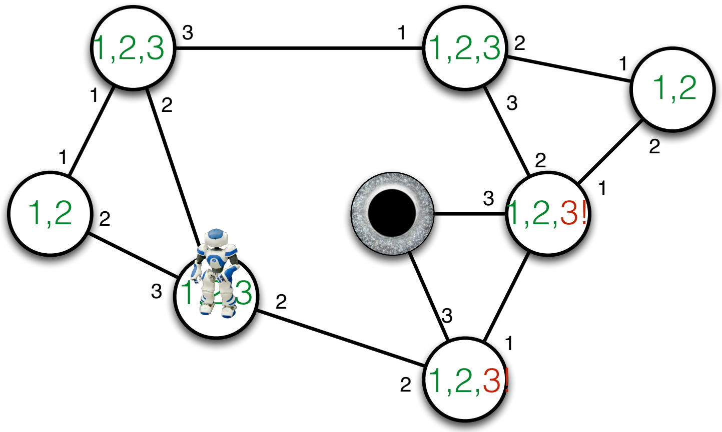
Asynchronous Black Hole Search



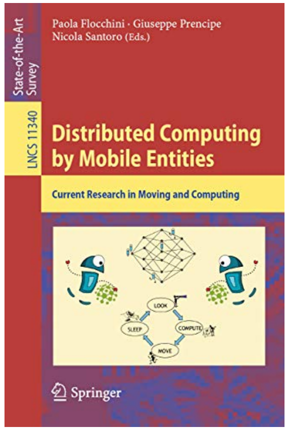
Asynchronous Black Hole Search



Asynchronous Black Hole Search

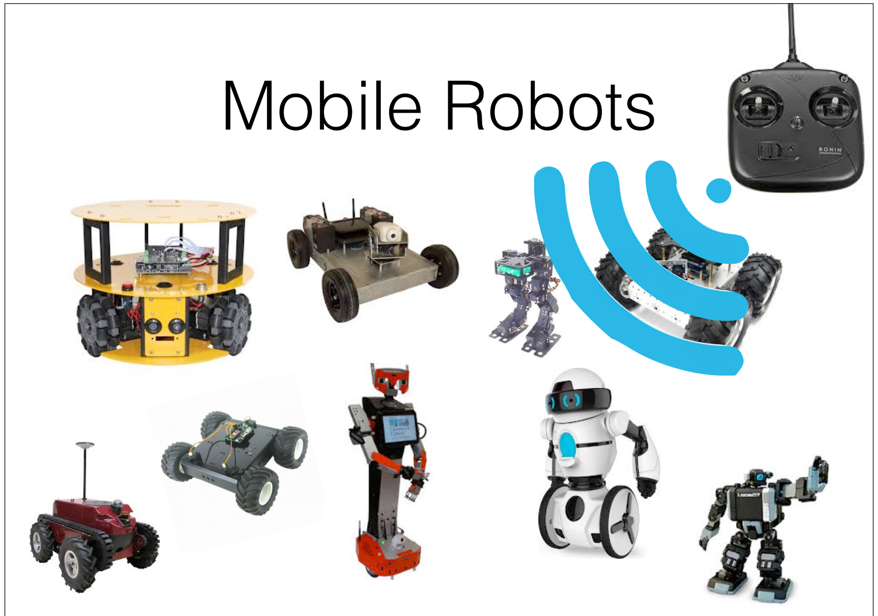


Mobile Agents

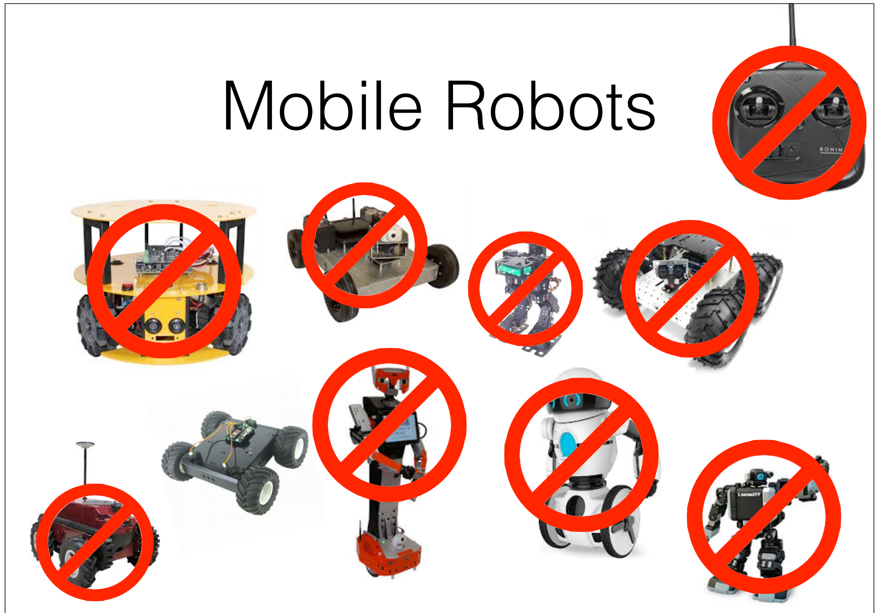


Mobile Robots

Mobile Robots



Mobile Robots

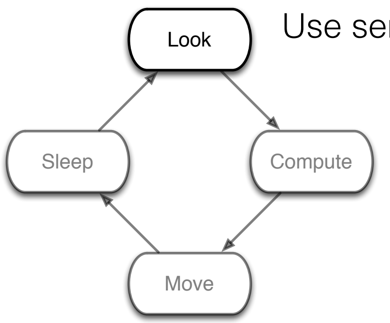


Mobile Robots

Mobile Robots

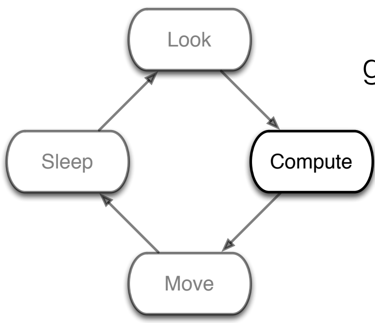
- **Autonomous** (no central control)
- **Homogeneous** (run same algorithm)
- **Identical** (indistinguishable)
- **Silent** (no explicit communication)

Robot Life Cycle



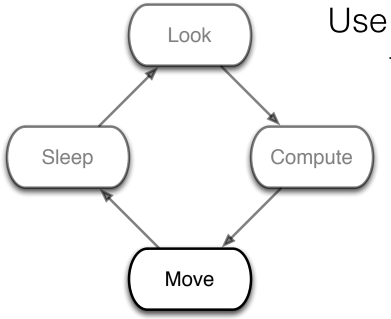
Use sensors to observe the world, get a **snapshot**

Robot Life Cycle



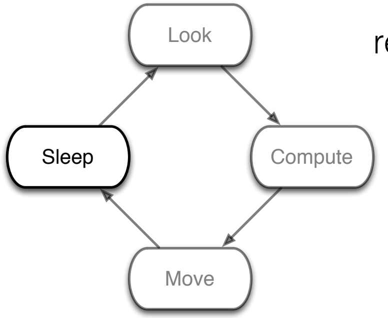
Execute the algorithm, get a **destination point**

Robot Life Cycle

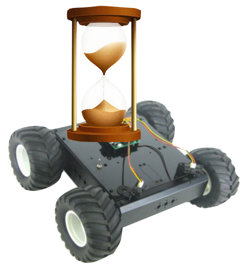


Use motors to **move toward** the destination point

Robot Life Cycle

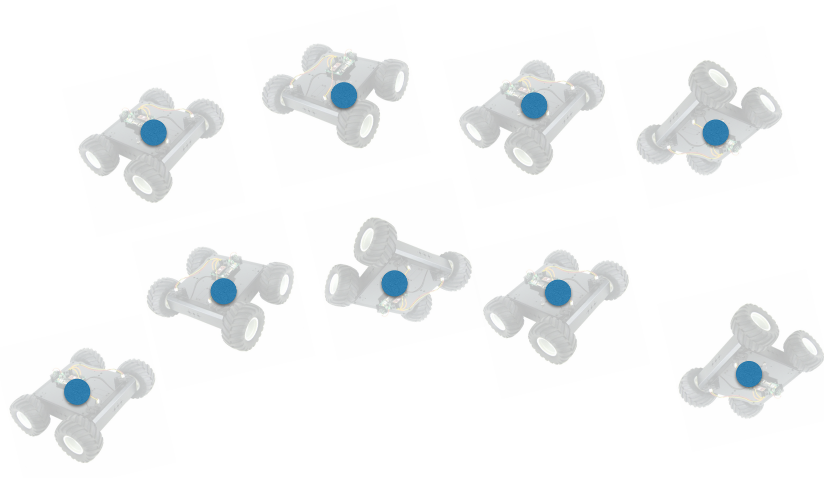


remain **idle** for a while

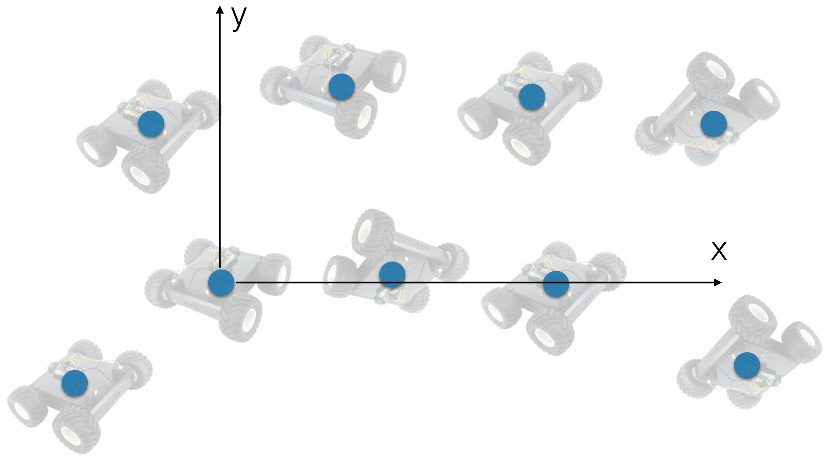


Visibility

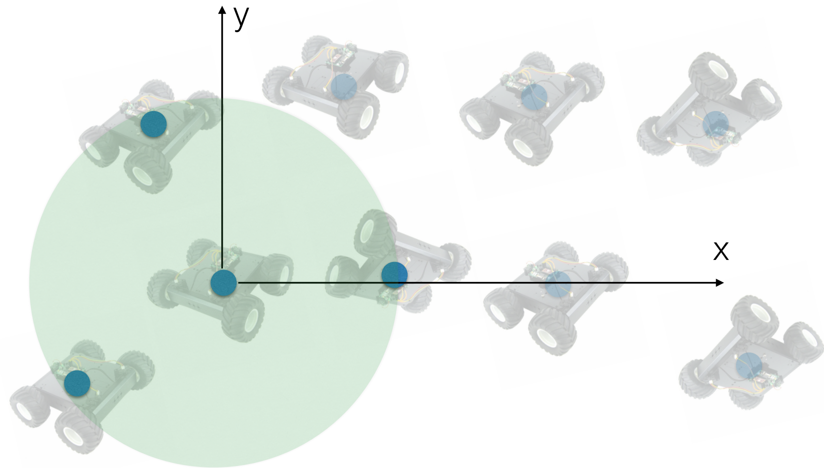
Visibility



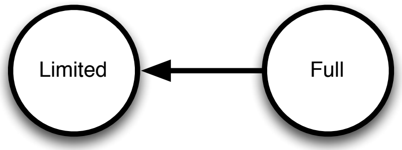
Visibility



Limited Visibility



Visibility

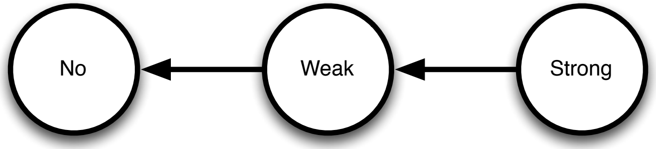


Multiplicity Detection

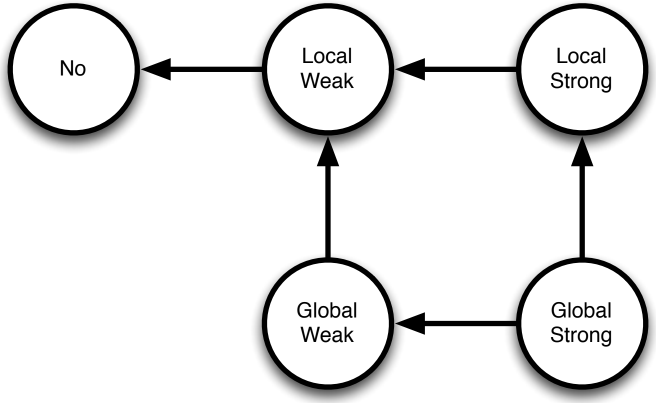
How many robots do you see?

- No detection 1
- Weak multiplicity detection >1
- Strong multiplicity detection

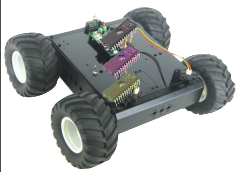
Multiplicity



Multiplicity



Memory

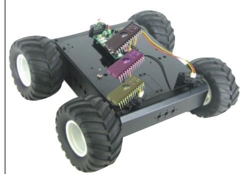


Algorithm

Persistent Memory

Volatile Memory

Oblivious Robot Memory

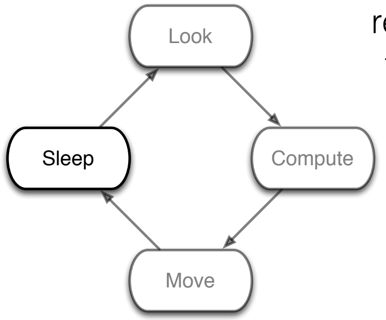


Algorithm

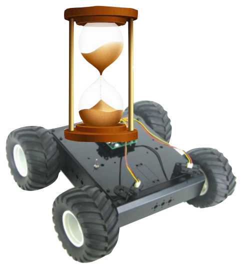
Persistent Memory

Volatile Memory

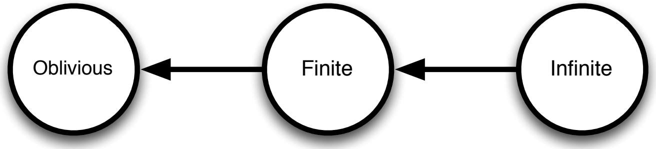
Oblivious Robot Life Cycle



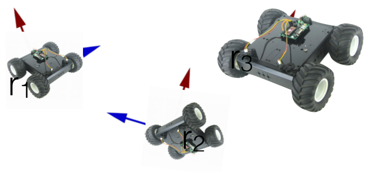
remain **idle** for a while,
forget about the past



Memory



Scheduling

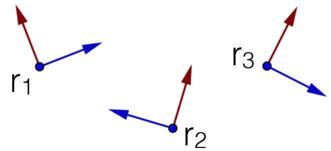


↺ Look→Compute→Move

FSYNC

	1	2	3	4	5	6
r_1	↺	↺	↺	↺	↺	↺
r_2	↺	↺	↺	↺	↺	↺
r_3	↺	↺	↺	↺	↺	↺

Scheduling

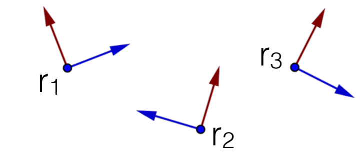


↺ Look→Compute→Move

SSYNC

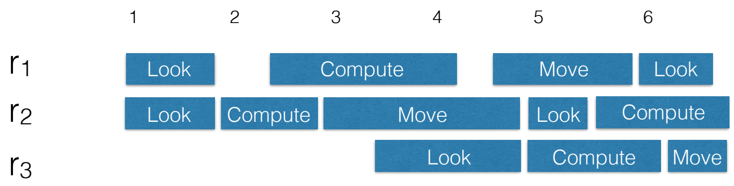
	1	2	3	4	5	6
r_1	↺		↺	↺	↺	↺
r_2	↺	↺		↺	↺	
r_3	↺		↺			↺

Scheduling

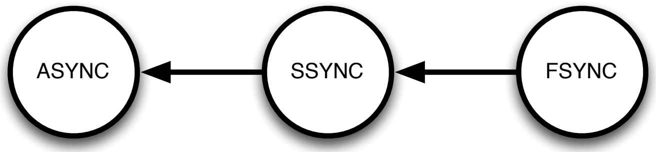


Look→Compute→Move

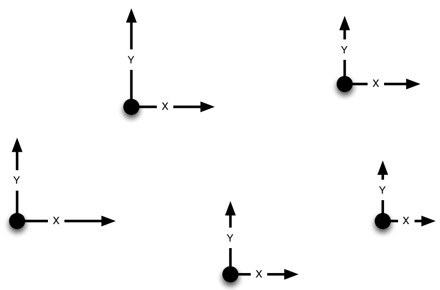
ASYNC



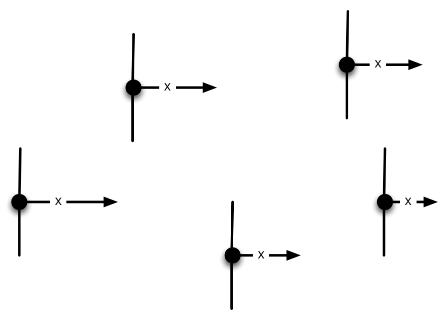
Scheduling



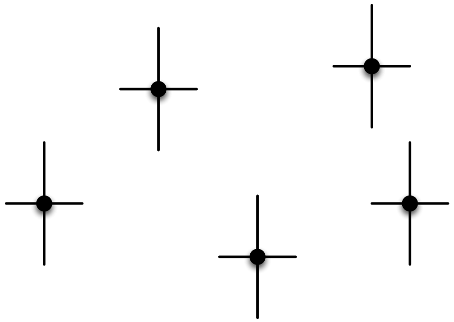
Two Axes
Direction and Orientation



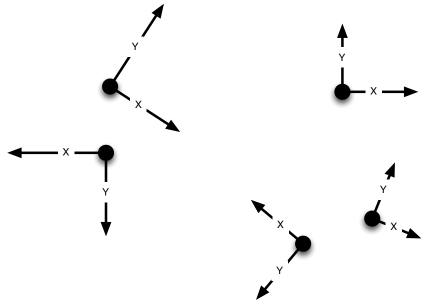
One Axis
Direction and Orientation



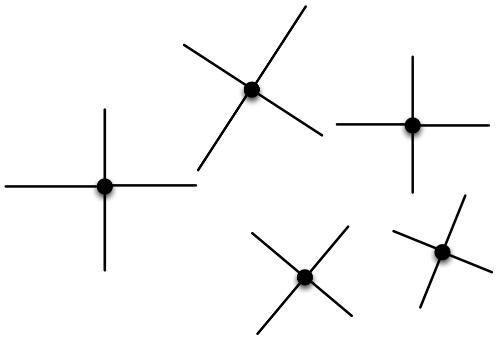
Two Axes Direction



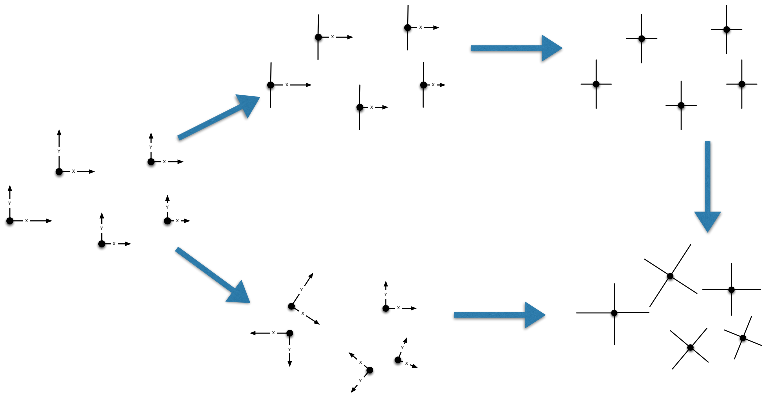
Chirality



No Agreement



Overview

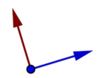


Scattering

Scattering

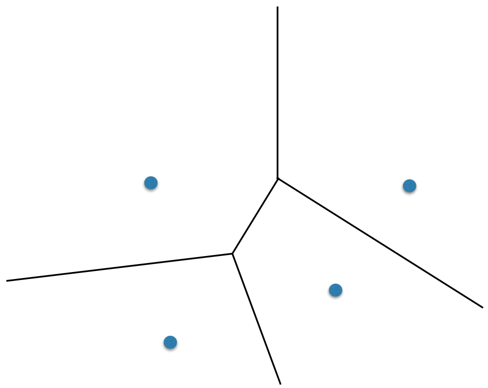
No two robots should occupy the same position

- No deterministic solution



- No termination without multiplicity detection

Scattering

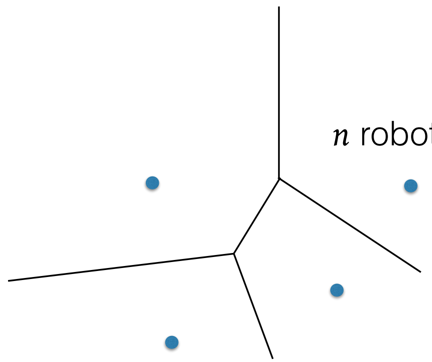


1 toss

$O(log(n)loglog(n))$ rounds

Yoann Dieudonné and Franck Petit. *Scatter of robots*. Parallel Processing Letters 19.01 (2009): 175-184.

Scattering



n robots: $2n^2$ destinations

$O(1)$ rounds

Julien Clément, Xavier Défago, Maria Potop-Butucaru, et al. *The cost of probabilistic agreement in oblivious robot networks*. Information Processing Letters, 2010, vol. 110, no 11, p. 431-438.

How Many Tosses?

Minimum?

Influence of multiplicity detection?

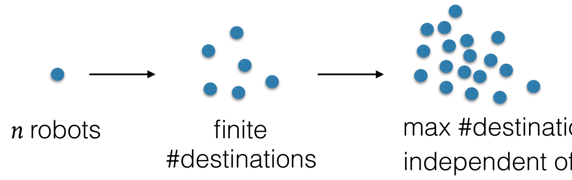
Relationship with scattering speed?

Optimal Speed

With strong multiplicity detection:

Algorithm with optimal #tosses terminates in $O(1)$ rounds

Without strong multiplicity detection:



$O(1)$ rounds scattering of n robots is impossible

How fast can we go?

Scattering

	Scattering	Scattering +MD
FSYNC		Yes $O(1)$ rounds
SSYNC		Yes $O(1)$ rounds
ASYNC		

Julien Clément, Xavier Défago, Maria Potop-Butucaru, et al. *The cost of probabilistic agreement in oblivious robot networks.* Information Processing Letters, 2010, vol. 110, no 11, p. 431-438.

Scattering

	Scattering	Scattering +MD
FSYNC	Yes $O(f(n))$ rounds	Yes $O(1)$ rounds
SSYNC	Yes $O(f(n))$ rounds	Yes $O(1)$ rounds
ASYNC		

Quentin Bramas and Sébastien Tixeuil. *The Ramdom Bit Complexity of Mobile Robot Scattering.* Int. J. Found. Comput. Sci. 28(2): 111-134 (2017)

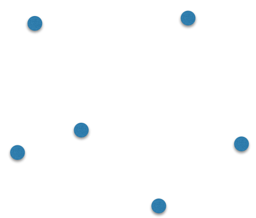
Scattering

	Scattering	Scattering +MD
FSYNC	Yes *O(f(n))* rounds	Yes *O(1)* rounds
SSYNC	Yes *O(f(n))* rounds	Yes *O(1)* rounds
ASYNC	Yes ?	Yes ?

Ulysse Léchine and Sébastien Tixeuil. *Asynchronous Scattering. ArXiv 2019*

Gathering

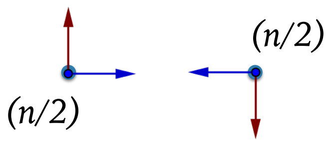
Gathering



Gathering

Gathering

Impossible for two robots



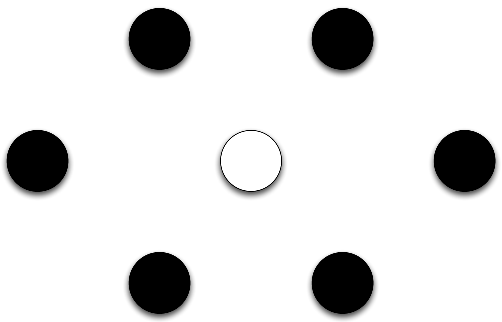
A bivalent configuration

Gathering vs. Convergence

- **Gathering**: robot must **reach** the same point in finite time
- **Convergence**: robots must get **closer** at time goes by

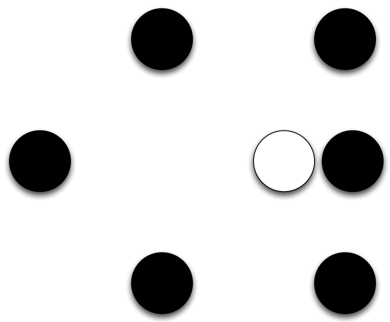
Center of Gravity

$$\vec{c}[t] = \frac{1}{n}\sum_{i=1}^{n}\vec{r_i}[t]$$



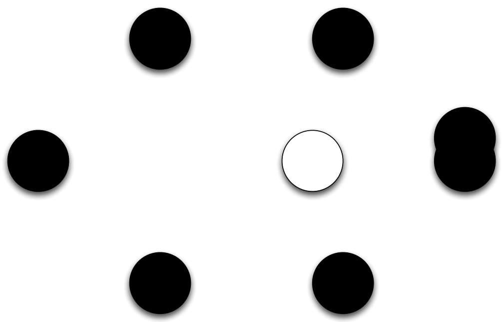
Center of Gravity

$$\vec{c}[t] = \frac{1}{n}\sum_{i=1}^{n}\vec{r_i}[t]$$



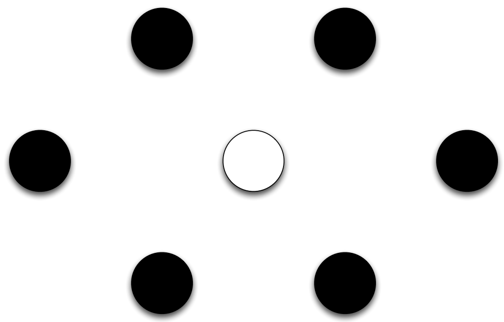
Center of Gravity

$$\vec{c}[t] = \frac{1}{n}\sum_{i=1}^{n}\vec{r_i}[t]$$



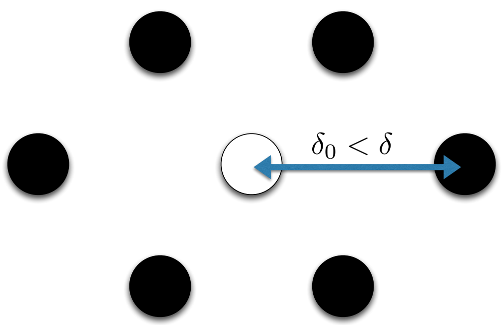
Center of Gravity of Positions

$$\vec{c}[t] = \frac{1}{p}\sum_{i=1}^{p}\vec{p_i}[t]$$



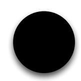
FSYNC Gathering

$$\vec{c}[t] = \frac{1}{p}\sum_{i=1}^{p}\vec{p_i}[t]$$

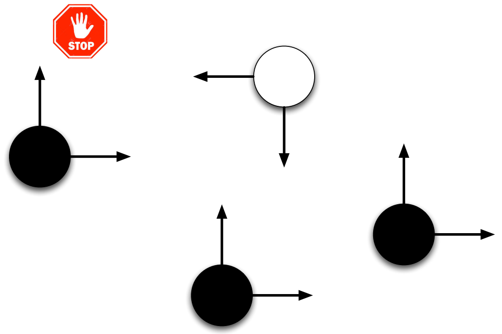


FSYNC Gathering

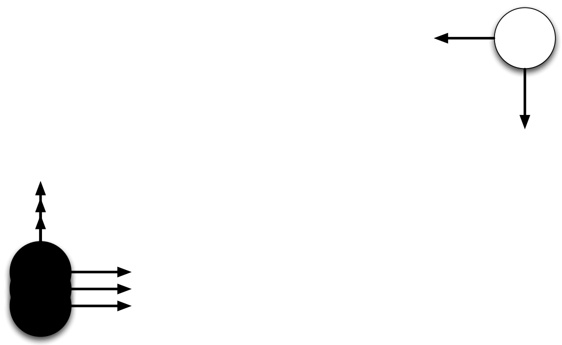
$$\vec{c}[t] = \frac{1}{p}\sum_{i=1}^{p}\vec{p_i}[t]$$



SSYNC Gathering?



SSYNC Gathering?



Convergence & Gathering

	Convergence	2-Gathering	n-Gathering	n-Gathering +MD	n-Gathering +MD+WF
FSYNC	Yes	Yes	Yes	Yes	Yes
SSYNC	Yes	No	No	Yes	Yes
ASYNC	Yes	No	No	Yes	?

Convergence & Gathering

	Convergence	2-Gathering	n-Gathering	n-Gathering +MD	n-Gathering +MD+WF
FSYNC	Yes	Yes	Yes	Yes	Yes
SSYNC	Yes	No	No	Yes	Yes
ASYNC	**Yes**	No	No	Yes	?

Reuven Cohen and David Peleg. *Convergence Properties of the Gravitational Algorithm in Asynchronous Robot Systems.* SIAM J. Comput. 34(6): 1516-1528 (2005)

Convergence & Gathering

	Convergence	2-Gathering	n-Gathering	n-Gathering +MD	n-Gathering +MD+WF
FSYNC	Yes	**Yes**	**Yes**	Yes	**Yes**
SSYNC	Yes	**No**	**No**	**Yes**	**Yes**
ASYNC	**Yes**	No	**No**	**Yes**	**?**

Ichiro Suzuki, Masafumi Yamashita: Distributed Anonymous Mobile Robots: Formation of Geometric Patterns. SIAM J. Comput. 28(4): 1347-1363 (1999)

Convergence & Gathering

	Convergence	2-Gathering	n-Gathering	n-Gathering +MD	n-Gathering +MD+WF
FSYNC	Yes	**Yes**	**Yes**	Yes	**Yes**
SSYNC	Yes	**No**	**No**	**Yes**	**Yes**
ASYNC	**Yes**	No	No	**Yes**	**?**

Guiseppe Prencipe. *Impossibility of gathering by a set of autonomous mobile robots. Theor. Comput. Sci. 384(2-3): 222-231 (2007)*

Convergence & Gathering

	Convergence	2-Gathering	n-Gathering	n-Gathering +MD	n-Gathering +MD+WF
FSYNC	Yes	**Yes**	**Yes**	Yes	**Yes**
SSYNC	Yes	**No**	**No**	**Yes**	**Yes**
ASYNC	**Yes**	No	No	**Yes**	**?**

Thibaut Balabonski, Amélie Delga, Lionel Rieg, Sébastien Tixeuil, Xavier Urbain: Synchronous Gathering Without Multiplicity Detection: A Certified Algorithm. SSS 2016: 7-19

Convergence & Gathering

	Convergence	2-Gathering	n-Gathering	n-Gathering +MD	n-Gathering +MD+WF
FSYNC	Yes	**Yes**	**Yes**	Yes	**Yes**
SSYNC	Yes	**No**	**No**	Yes	**Yes**
ASYNC	**Yes**	No	No	**Yes**	**?**

Mark Cieliebak, Paola Flocchini, Giuseppe Prencipe, Nicola Santoro. *Distributed Computing by Mobile Robots: Gathering*. SIAM J. Comput. 41(4): 829-879 (2012)

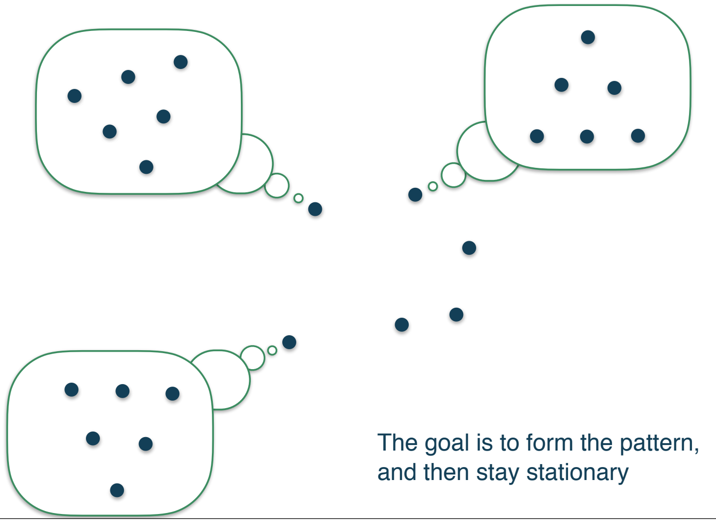
Convergence & Gathering

	Convergence	2-Gathering	n-Gathering	n-Gathering +MD	n-Gathering +MD+WF
FSYNC	Yes	**Yes**	**Yes**	Yes	Yes
SSYNC	Yes	**No**	**No**	Yes	**Yes**
ASYNC	**Yes**	No	No	**Yes**	**?**

Quentin Bramas, Sébastien Tixeuil. *Wait-Free Gathering Without Chirality*. SIROCCO 2015: 313-327

Pattern Formation

Pattern Formation

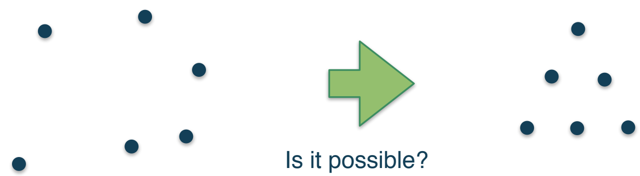


The goal is to form the pattern, and then stay stationary

Pattern Formation

Initial configuration

Pattern to form

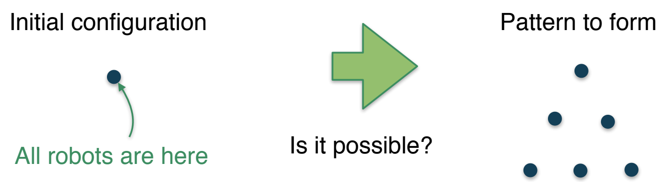


Is it possible?

In general, NO

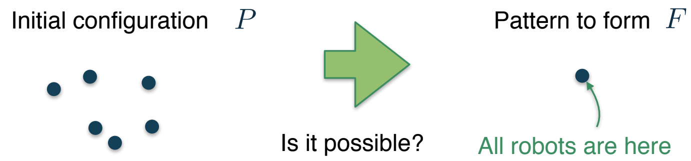
What about restricting the set of initial configurations?
What about adding conditions on the pattern?
What about adding capabilities to robots?

Pattern Formation



No, so from now, we assume the initial configuration does not have points of multiplicity

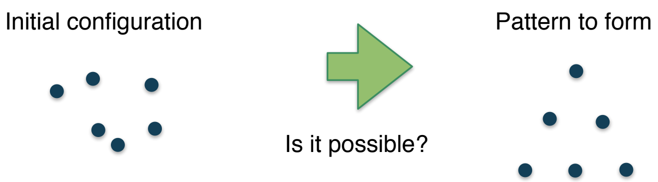
Pattern Formation



NO

Guiseppe Prencipe. *Impossibility of gathering by a set of autonomous mobile robots. Theor. Comput. Sci. 384(2-3): 222-231 (2007)*

Pattern Formation

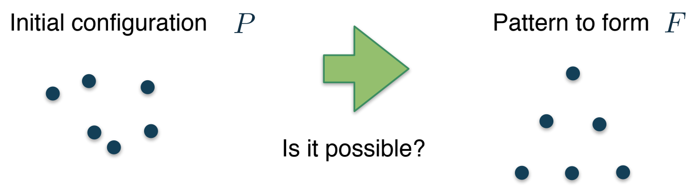


Yes, if robots agree on a common North and a common Right

Yes, if robots agree on a common North and n is odd

Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, Peter Widmayer: Arbitrary pattern formation by asynchronous, anonymous, oblivious robots. Theor. Comput. Sci. 407(1-3): 412-447 (2008)

Pattern Formation



…assuming a common chirality, and F does not have multiplicity points

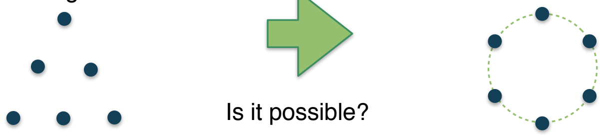
Yes, if $\rho(P) \mid \rho(F)$ where $\rho(P)$ is the symmetricity of P, the maximum integer such that the rotation by $2\pi/\rho(P)$ is invariant for P

No, otherwise

Nao Fujinaga, Yukiko Yamauchi, Hirotaka Ono, Shuji Kijima, Masafumi Yamashita: Pattern Formation by Oblivious Asynchronous Mobile Robots. SIAM J. Comput. 44(3): 740-785 (2015)

Pattern Formation

Initial configuration P Pattern to form F



…assuming a common chirality, and F does not have multiplicity points

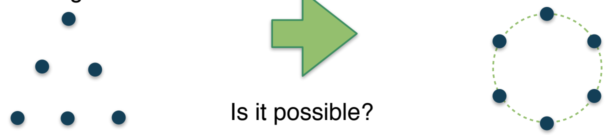
Yes, if $\rho(P) \mid \rho(F)$ where $\rho(P)$ is the symmetricity of P, the maximum integer such that the rotation by $2\pi/\rho(P)$ is invariant for P

No, otherwise

Nao Fujinaga, Yukiko Yamauchi, Hirotaka Ono, Shuji Kijima, Masafumi Yamashita:
Pattern Formation by Oblivious Asynchronous Mobile Robots. SIAM J. Comput. 44(3): 740-785 (2015)

Pattern Formation

Initial configuration P Pattern to form F



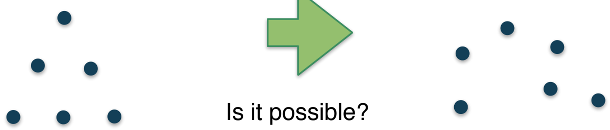
…assuming a common chirality, and F does not have multiplicity points

Yes, if $\rho(P) \mid \rho(F)$ where $\rho(P)$ is the symmetricity of P, the maximum integer such that the rotation by $2\pi/\rho(P)$ is invariant for P

Nao Fujinaga, Yukiko Yamauchi, Hirotaka Ono, Shuji Kijima, Masafumi Yamashita:
Pattern Formation by Oblivious Asynchronous Mobile Robots. SIAM J. Comput. 44(3): 740-785 (2015)

Pattern Formation

Initial configuration P Pattern to form F



…assuming a common chirality, and F does not have multiplicity points

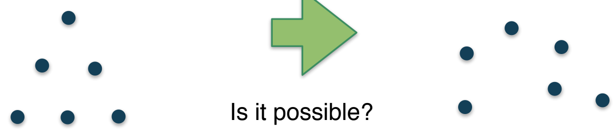
Yes, if $\rho(P) \mid \rho(F)$ where $\rho(P)$ is the symmetricity of P, the maximum integer such that the rotation by $2\pi/\rho(P)$ is invariant for P

No, otherwise

Nao Fujinaga, Yukiko Yamauchi, Hirotaka Ono, Shuji Kijima, Masafumi Yamashita:
Pattern Formation by Oblivious Asynchronous Mobile Robots. SIAM J. Comput. 44(3): 740-785 (2015)

Pattern Formation

Initial configuration P Pattern to form F

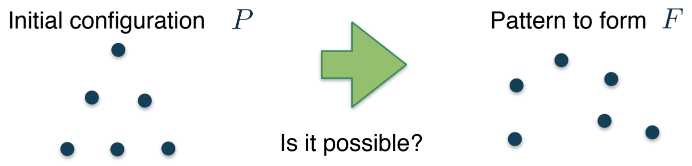


…assuming a common chirality, and F does not have multiplicity points

No, otherwise

Nao Fujinaga, Yukiko Yamauchi, Hirotaka Ono, Shuji Kijima, Masafumi Yamashita:
Pattern Formation by Oblivious Asynchronous Mobile Robots. SIAM J. Comput. 44(3): 740-785 (2015)

Pattern Formation



…assuming a common chirality, and F does not have multiplicity points

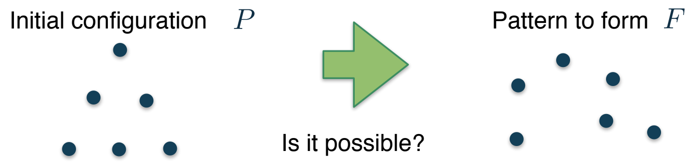
Yes, with a randomized algorithm

… assuming robots do not "pause" while moving

… and using infinitely many random bits per activation

Yukiko Yamauchi, Masafumi Yamashita: Randomized Pattern Formation Algorithm for Asynchronous Oblivious Mobile Robots. DISC 2014: 137-151

Pattern Formation



~~…assuming a common chirality, and F does not have multiplicity points~~

Yes, with a randomized algorithm

F is not a point

… ~~assuming robots do not "pause" while moving~~ really asynchronous

… ~~and using infinitely many random bits per activation~~

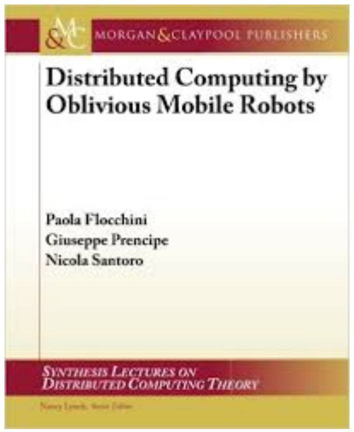
only one random bit

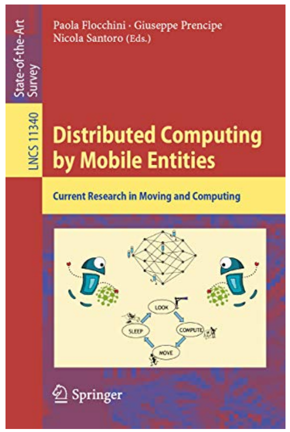
Quentin Bramas, Sébastien Tixeuil: Brief Announcement: Probabilistic Asynchronous Arbitrary Pattern Formation. PODC 2016: 443-445

ASYNC Pattern Formation

Pattern	Agreement	Chirality	Randomization
Point	Yes	No	?
Divide Symmetricity	Yes	Yes	Yes
No Multiplicity	Yes	No	Yes
Not a Point	Yes	No	Yes
Arbitrary	Yes	No	?

Mobile Robots





Conclusion

Static Networks

- **Fundamental**, well established model
  - **Space-centric**, *complexity* results
  - **Time-centric**, *computability* results

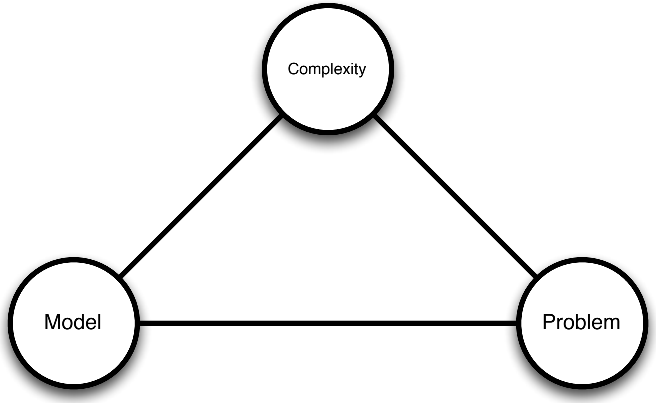
Mobility as an Adversary

- Can corrupt the distributed state of a network
- Can reduces communication capacity
- Can increase uncertainty
- Can increase protocol complexity

Mobility as a Friend

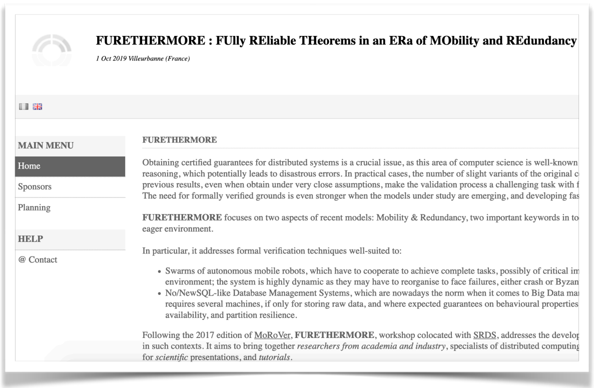
- Mobility can be the solution to the problem
- Mobility can improve efficiency
- Mobility can promote simplicity

Distributed Computing



Thank You

Furethermore 2019


FURETHERMORE : FUlly REliable THeorems in an ERa of MObility and REdundancy

1 Oct 2019 Villeurbanne (France)

MAIN MENU
- Home
- Sponsors
- Planning

HELP
- @ Contact

FURETHERMORE

Obtaining certified guarantees for distributed systems is a crucial issue, as this area of computer science is well-known
reasoning, which potentially leads to disastrous errors. In practical cases, the number of slight variants of the original c
previous results, even when obtain under very close assumptions, make the validation process a challenging task with f
The need for formally verified grounds is even stronger when the models under study are emerging, and developing fas

FURETHERMORE focuses on two aspects of recent models: Mobility & Redundancy, two important keywords in to
eager environment.

In particular, it addresses formal verification techniques well-suited to:

- Swarms of autonomous mobile robots, which have to cooperate to achieve complete tasks, possibly of critical im
environment; the system is highly dynamic as they may have to reorganise to face failures, either crash or Byzan
- No/NewSQL-like Database Management Systems, which are nowadays the norm when it comes to Big Data ma
requires several machines, if only for storing raw data, and where expected guarantees on behavioural properties
availability, and partition resilience.

Following the 2017 edition of MoRoVer, **FURETHERMORE**, workshop colocated with SRDS, addresses the develop
in such contexts. It aims to bring together *researchers from academia and industry*, specialists of distributed computing
for *scientific* presentations, and *tutorials*.

https://furethermore.sciencesconf.org/