

Distributed Algorithms for Large-scale Robotic Ensembles: Centrality, Synchronization and Self-reconfiguration

André Naz¹
Thesis Defense
December, 4 2017

Supervisors: Julien Bourgeois, Seth C. Goldstein and Benoît Piranda

¹Univ. Bourgogne Franche-Comté, University of Franche-Comté
FEMTO-ST Institute, CNRS



The original presentation contains videos that are not embedded in this PDF.
Please visit <https://github.com/nazandre/phdthesis-defense>
to visualize the videos.

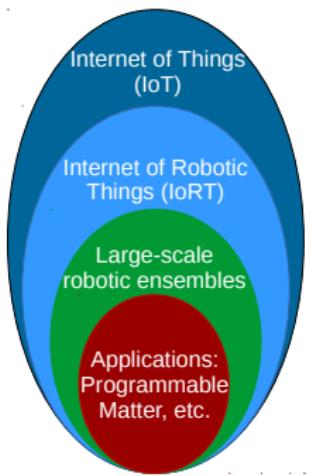


Funded by the ANR/RGC
(contracts ANR-12-IS02-0004-01
and 3-ZG1F)

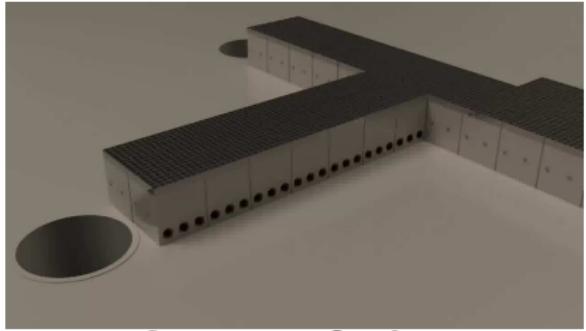
Outline

- 1 Introduction
- 2 Research Environment: Hardware and Simulation Tools
- 3 Centrality-based Leader Election
- 4 Time Synchronization
- 5 Self-Reconfiguration
- 6 Conclusion and Future Works

Large-scale Robotic Ensembles



- Modular robotics, swarm robotics, distributed MEMS, robotic materials
- Macro-scale: reconfigurable IoRT objects
- Micro-scale: self-organized ensembles
 - ▶ up to 10^6 units
 - ▶ small and resource-constrained (energy, memory, computation) units
- Advantages: versatility, robustness, cheaper

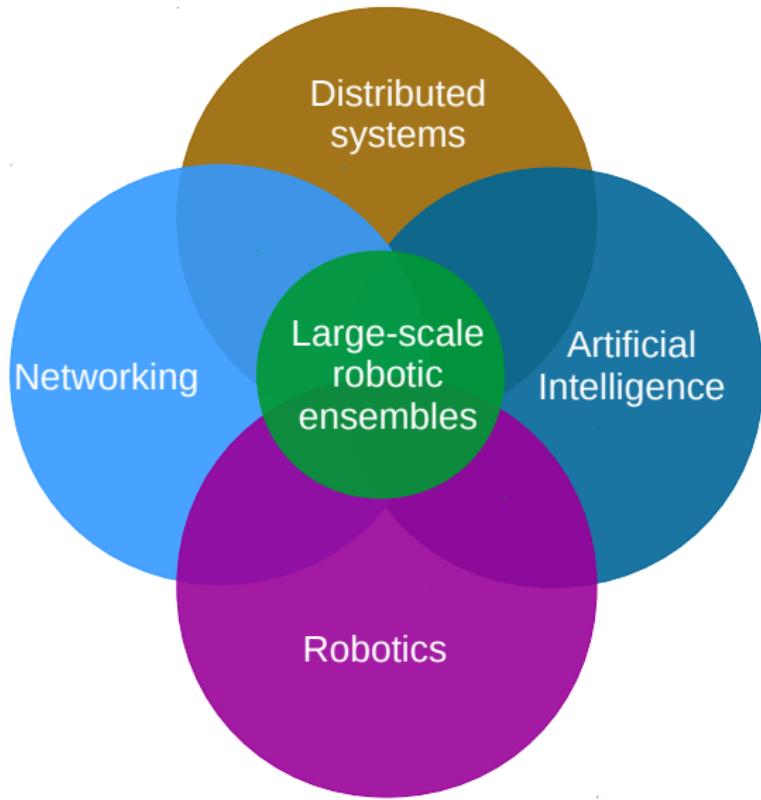


Conveying Surface
Smart Blocks project (FEMTO-ST)



Programmable Matter
Claytronics project (CMU)

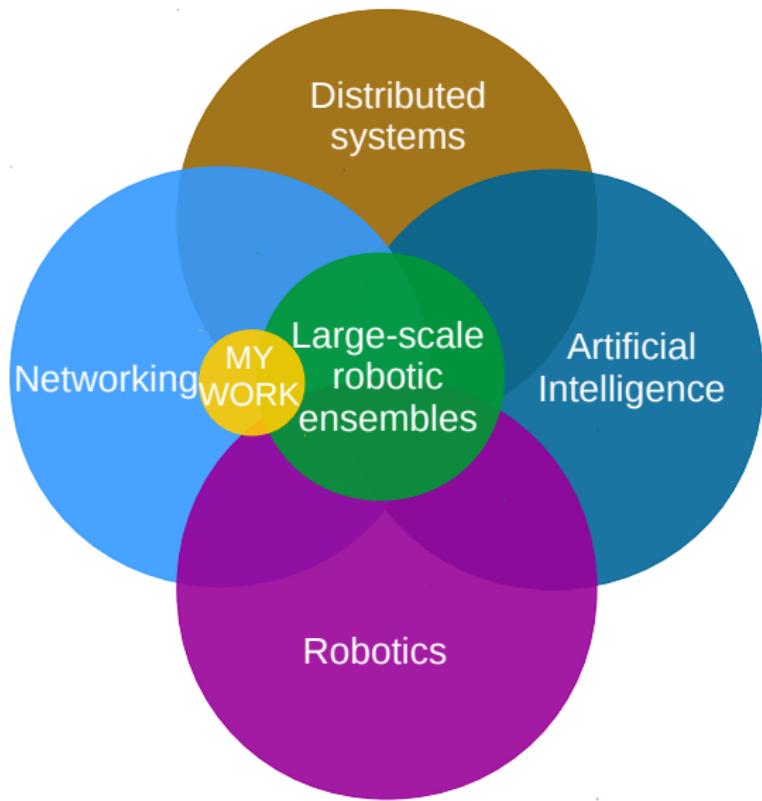
An Inter-Disciplinary Domain



Challenges:

- Hardware: Fabrication
- Algorithmic: Distributed coordination

An Inter-Disciplinary Domain



Challenges:

- Hardware: Fabrication
- Algorithmic: Distributed coordination

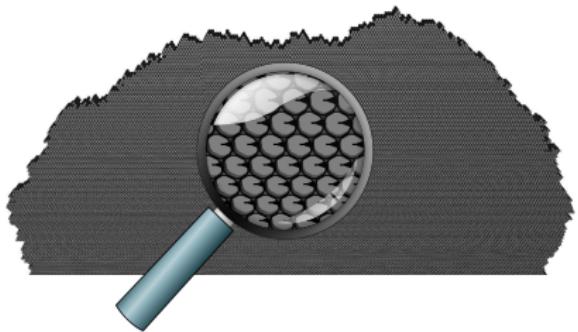
my work

Problem Statement

- Coordination of large-scale ensembles of resource-constrained robots
 - ⇒ New algorithmic challenges
 - ⇒ My postulate: we need to identify and design high-level primitives to help the coordination of these ensembles

Problem Statement

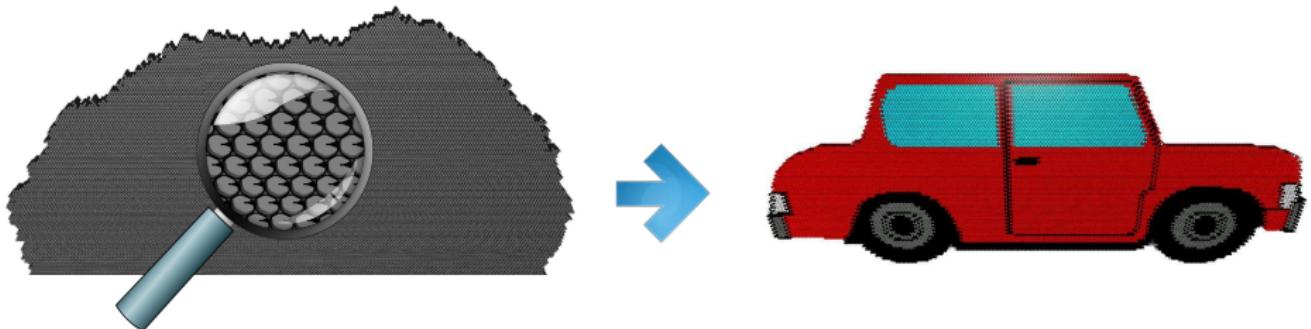
- Coordination of large-scale ensembles of resource-constrained robots
 - ⇒ New algorithmic challenges
 - ⇒ My postulate: we need to identify and design high-level primitives to help the coordination of these ensembles
- Application scenario*:



*Image based on modified versions of images released in the public domain under the Creative Commons CC0 license which were downloaded from Pixabay and Fotomelia.

Problem Statement

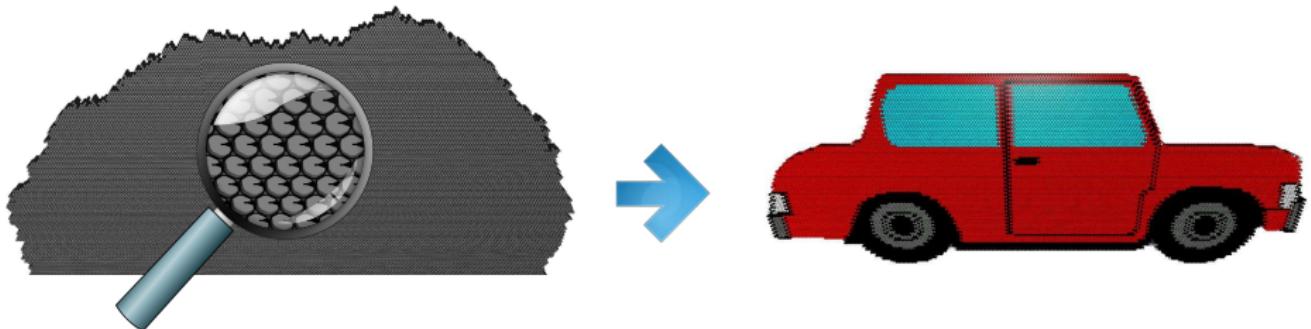
- Coordination of large-scale ensembles of resource-constrained robots
 - ⇒ New algorithmic challenges
 - ⇒ My postulate: we need to identify and design high-level primitives to help the coordination of these ensembles
- Application scenario*:



*Image based on modified versions of images released in the public domain under the Creative Commons CC0 license which were downloaded from Pixabay and Fotomelia.

Problem Statement

- Coordination of large-scale ensembles of resource-constrained robots
 - ⇒ New algorithmic challenges
 - ⇒ My postulate: we need to identify and design high-level primitives to help the coordination of these ensembles
- Application scenario*:



- Research problems (proposed high-level primitives):
 - ① Shape construction
 - ② Time synchronization
 - ③ Centrality-based leader election

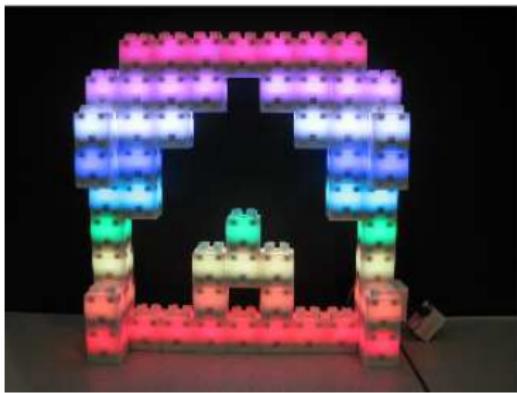
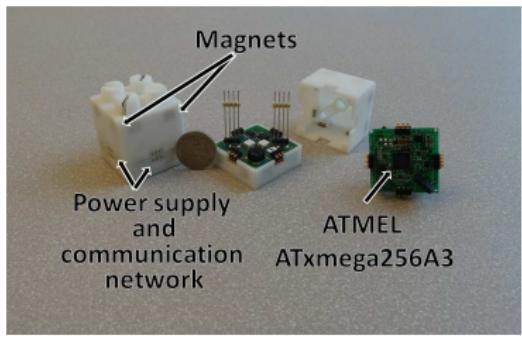
*Image based on modified versions of images released in the public domain under the Creative Commons CC0 license which were downloaded from Pixabay and Fotomelia.

Section Outline

- 1 Introduction
- 2 Research Environment: Hardware and Simulation Tools
- 3 Centrality-based Leader Election
- 4 Time Synchronization
- 5 Self-Reconfiguration
- 6 Conclusion and Future Works

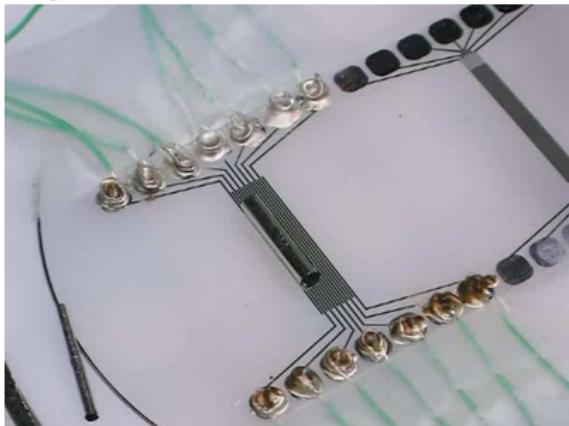
The Blinky Blocks

- Several generations:
 - ▶ Originally developed at Carnegie Mellon University [Kirby et al., 2011]
 - ▶ Last generations fabricated by FEMTO-ST
- Hardware features
 - ▶ micro-controller: ATMEAL ATxmega256A3-AU 8/16-bits 32MHz
 - ▶ memory: 256KB ROM and 16KB RAM
 - ▶ Network: 6 serial interfaces configured at 38.4 kbit/s
 - ▶ RGB leds
 - ▶ Local Clock: Real-Time Counter driven by a RC oscillator
 - 1% accuracy and 1ms resolution

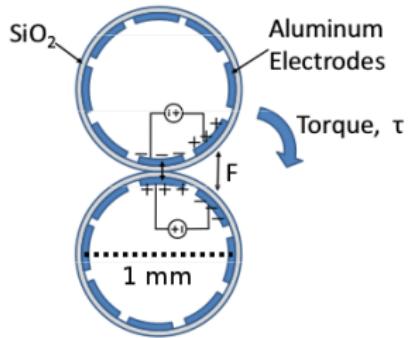


The 2D Catoms

Partially validated 2D Catom hardware prototype [Karagozler et al., 2009]



Hardware prototype



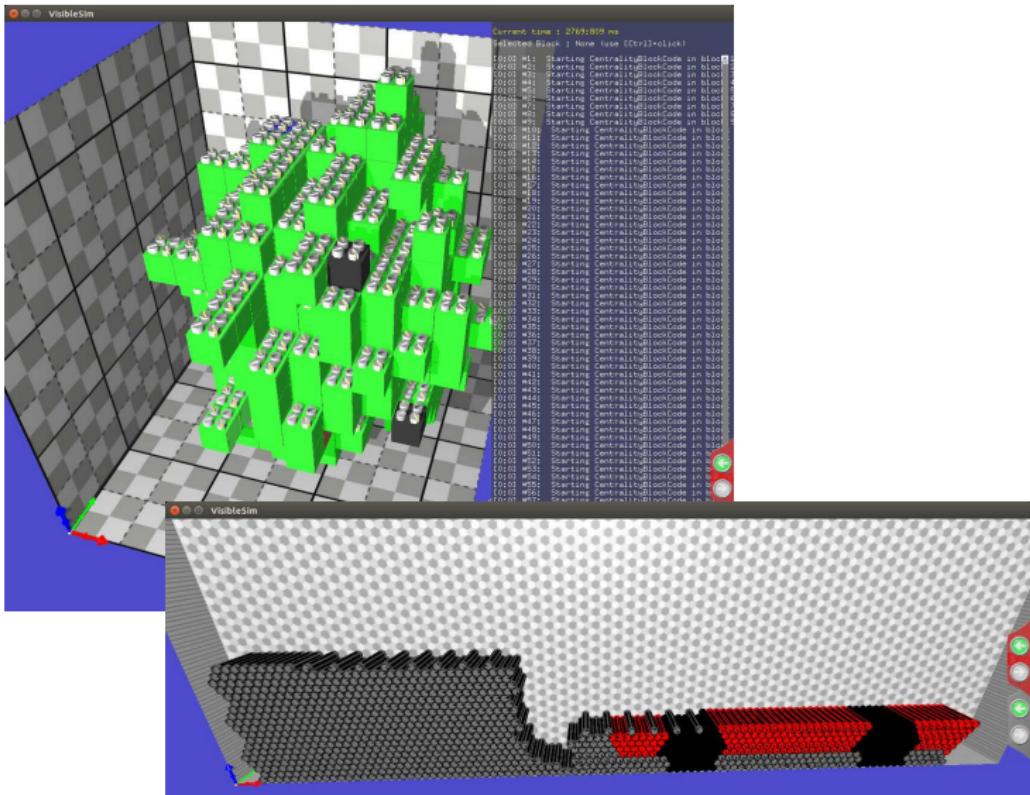
Actuation scheme

Partially validated:

- Only motion on the ground
- No communication, nor module arrangement, nor color for now

VisibleSim

Simulator of modular robotic ensembles [Dhoutaut et al., 2013]

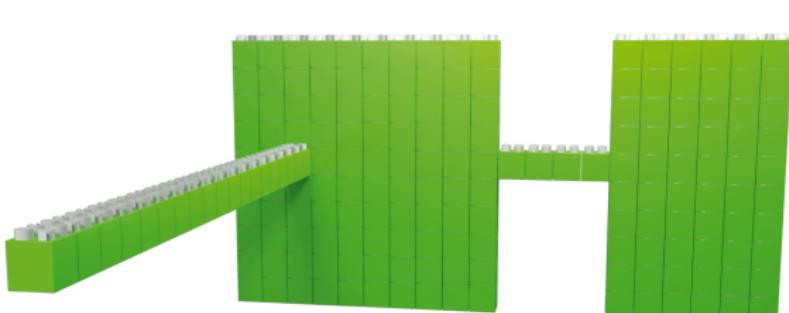


Section Outline

- 1 Introduction
- 2 Research Environment: Hardware and Simulation Tools
- 3 Centrality-based Leader Election
 - Problem and Motivations
 - Related Work and Contributions
 - ABC-CenterV2
 - Evaluation
 - Conclusion
- 4 Time Synchronization
- 5 Self-Reconfiguration
- 6 Conclusion and Future Works

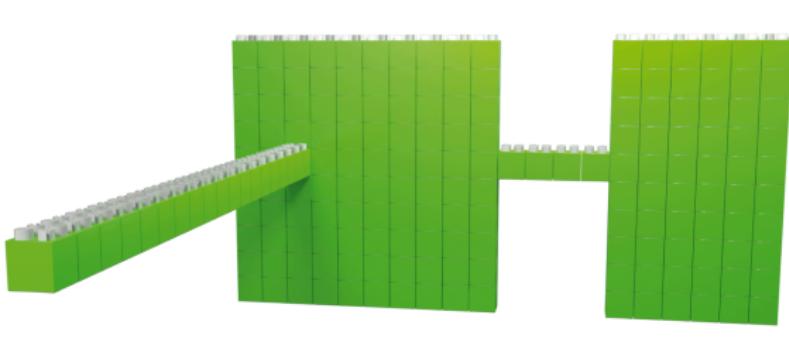
Problem and Motivations

- Network graph $G(V, E)$: V = modules, E = connections
 - ▶ Neighbor-to-Neighbor communications, unique node identifiers
 - ▶ $d(v_i, v_j)$ = hop distance between modules v_i and v_j



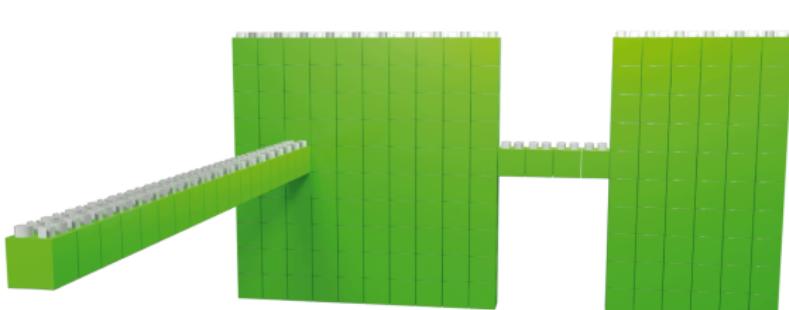
Problem and Motivations

- Network graph $G(V, E)$: V = modules, E = connections
 - ▶ Neighbor-to-Neighbor communications, unique node identifiers
 - ▶ $d(v_i, v_j)$ = hop distance between modules v_i and v_j
- Problem: efficient election of an approximate central node



Problem and Motivations

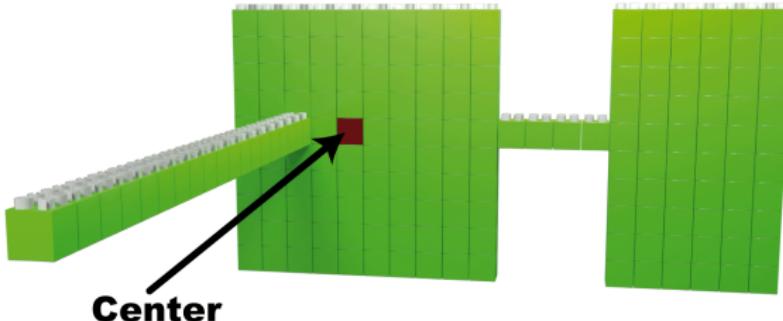
- Network graph $G(V, E)$: V = modules, E = connections
 - ▶ Neighbor-to-Neighbor communications, unique node identifiers
 - ▶ $d(v_i, v_j)$ = hop distance between modules v_i and v_j
- Problem: efficient election of an approximate central node
- Central nodes?



Problem and Motivations

- Network graph $G(V, E)$: V = modules, E = connections
 - ▶ Neighbor-to-Neighbor communications, unique node identifiers
 - ▶ $d(v_i, v_j)$ = hop distance between modules v_i and v_j
- Problem: efficient election of an approximate central node
- Central nodes?
 - ▶ Center: minimizes the maximum distance to all the others

$$\text{Center} = \operatorname{argmin}_{v_i \in V} \text{ecc}(v_i) = \operatorname{argmin}_{v_i \in V} \max_{v_j \in V} d(v_i, v_j)$$



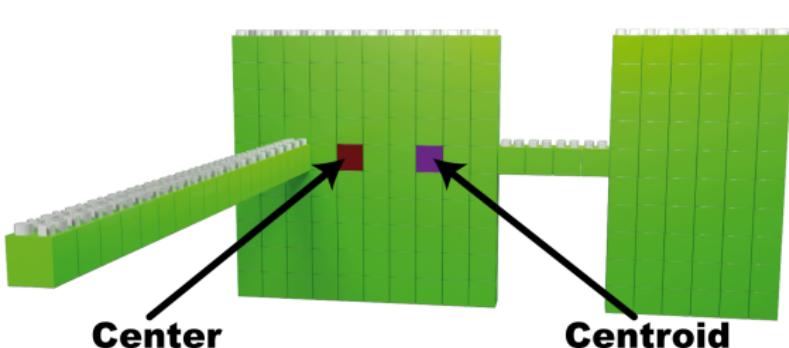
Problem and Motivations

- Network graph $G(V, E)$: V = modules, E = connections
 - ▶ Neighbor-to-Neighbor communications, unique node identifiers
 - ▶ $d(v_i, v_j)$ = hop distance between modules v_i and v_j
- Problem: efficient election of an approximate central node
- Central nodes?
 - ▶ Center: minimizes the maximum distance to all the others

$$\text{Center} = \operatorname{argmin}_{v_i \in V} \text{ecc}(v_i) = \operatorname{argmin}_{v_i \in V} \max_{v_j \in V} d(v_i, v_j)$$

- ▶ Centroid: minimizes the average distance to all the others

$$\text{Centroid} = \operatorname{argmin}_{v_i \in V} \frac{1}{|V|} \sum_{v_j \in V} d(v_i, v_j)$$



Problem and Motivations

- Network graph $G(V, E)$: V = modules, E = connections
 - ▶ Neighbor-to-Neighbor communications, unique node identifiers
 - ▶ $d(v_i, v_j)$ = hop distance between modules v_i and v_j

- Problem: efficient election of an approximate central node
- Central nodes?

- ▶ Center: minimizes the maximum distance to all the others

$$\text{Center} = \operatorname{argmin}_{v_i \in V} \text{ecc}(v_i) = \operatorname{argmin}_{v_i \in V} \max_{v_j \in V} d(v_i, v_j)$$

- ▶ Centroid: minimizes the average distance to all the others

$$\text{Centroid} = \operatorname{argmin}_{v_i \in V} \frac{1}{|V|} \sum_{v_j \in V} d(v_i, v_j)$$

- Challenge: requires a distributed all-pair shortest path computation
 - ▶ Large time, message and/or storage costs.

Problem and Motivations

- Network graph $G(V, E)$: V = modules, E = connections
 - ▶ Neighbor-to-Neighbor communications, unique node identifiers
 - ▶ $d(v_i, v_j)$ = hop distance between modules v_i and v_j
- Problem: efficient election of an approximate central node
- Central nodes?
 - ▶ Center: minimizes the maximum distance to all the others

$$\text{Center} = \operatorname{argmin}_{v_i \in V} \text{ecc}(v_i) = \operatorname{argmin}_{v_i \in V} \max_{v_j \in V} d(v_i, v_j)$$
 - ▶ Centroid: minimizes the average distance to all the others

$$\text{Centroid} = \operatorname{argmin}_{v_i \in V} \frac{1}{|V|} \sum_{v_j \in V} d(v_i, v_j)$$
- Challenge: requires a distributed all-pair shortest path computation
 - ▶ Large time, message and/or storage costs.
- Motivation: ideal nodes for communications with all the others
 - ▶ Example: service providers
 - Time synchronization: the precision decreases with the hop distance to the time master

Subsection Outline

3 Centrality-based Leader Election

- Problem and Motivations
- Related Work and Contributions
- ABC-CenterV2
- Evaluation
- Conclusion

Related Work

Approach	Accuracy (experiments)	Time and message cost	Memory cost
Naive and exhaustive (e.g., [Korach et al., 1984, Mamei et al., 2005])	exact	+++++ (potential congestion)	+ to +++++
Graph specific (e.g., [Bruell et al., 1999] for trees)	exact (but not for arbitrary graphs)	+	+
Approximation (e.g., [Wehmuth and Ziviani, 2013, Dissler et al., 2016, Kim and Wu, 2013])	++ to +++	++	+ to +++

Thesis contribution:

3 approximation algorithms	+++ to ++++	+++	+
----------------------------	----------------	-----	---

⇒ A good cost-accuracy trade-off

My Contributions

- 3 distributed algorithms:
 - ▶ k -BFS SumSweep
 - ▶ ABC-Center (two versions)
 - ▶ Probabilistic Counter based Central Leader Election (PC2LE)
- Inspired from existing external-graph analysis algorithms
- All based on intuitive heuristics
- Experimental evaluation of the accuracy
- Formal analysis of the performance:

Name	Type of center	Time	Memory (per module)	Message
k -BFS SumSweep	center, centroid	$O(k \times d)$	$O(\Delta)$	$O(m \times n^2)$
ABC-CenterV2	center	$O(\#steps \times d)$	$O(\Delta)$	$O(m \times n^2)$
PC2LE	center, centroid	$O(d)$	$O(\Delta + \text{probabilistic counter})$	$O(m \times n^2)$

Notation:

$n = \#\text{modules}$, $m = \#\text{ links}$, $d = \text{diameter}$, $\Delta = \text{maximum number of neighbors}$

Subsection Outline

3 Centrality-based Leader Election

- Problem and Motivations
- Related Work and Contributions
- ABC-CenterV2
 - General Idea
 - Algorithm
 - Examples
 - Tricky Case
- Evaluation
- Conclusion

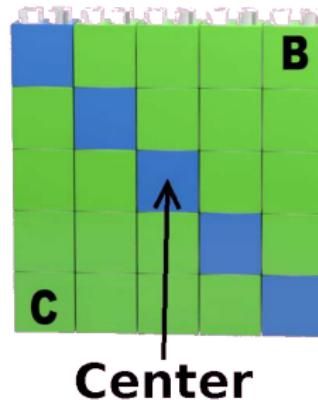
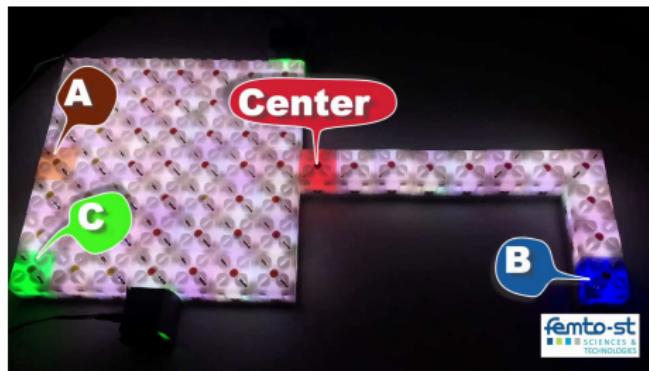
General Idea

- Extends the sequential MiniMax algorithm [Handler, 1973] for tree graphs
- Idea: the center stands at the middle of a diameter (i.e., longest path)



General Idea

- Extends the sequential MiniMax algorithm [Handler, 1973] for tree graphs
- Idea: the center stands at the middle of a diameter (i.e., longest path)



Algorithm

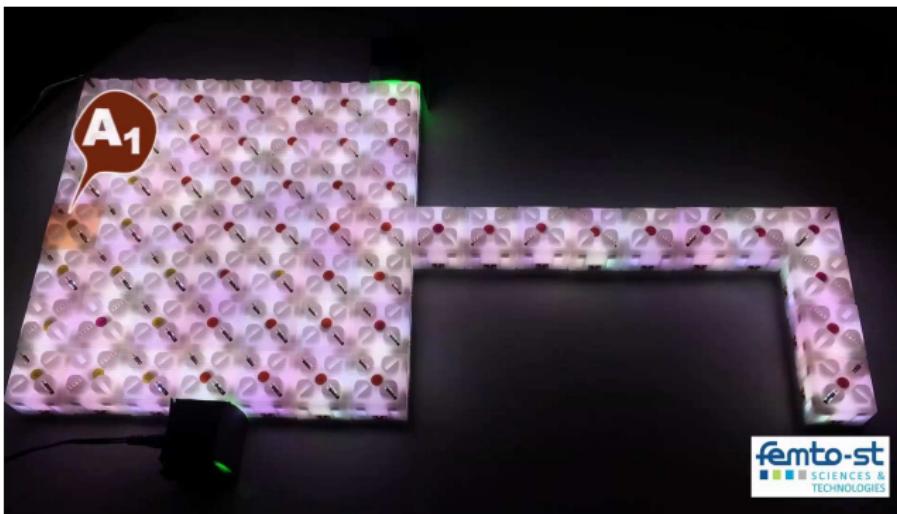
```

Candidates  $\leftarrow$  all modules;
while  $|Candidates| > 2$  do
     $A \leftarrow v_i \in Candidates$ ; // A is a random candidate
     $B \leftarrow v_i \in \operatorname{argmax}_{v_j \in Candidates} d(A, v_j)$ ; // B a farthest candidate from A
     $C \leftarrow v_i \in \operatorname{argmax}_{v_j \in Candidates} d(B, v_j)$ ; // C a farthest candidate from B
    // Most equi-distant nodes from B to C remain candidate:
     $Candidates \leftarrow \operatorname{argmin}_{v_j \in Candidates} |d(B, v_j) - d(C, v_j)|$ ;
    // B and C are eliminated (if not already purged by the previous
    // line):
     $Candidates \leftarrow Candidates - \{B, C\}$ ;
end
 $central \leftarrow v_i \in Candidates$ ; // a random candidate

```

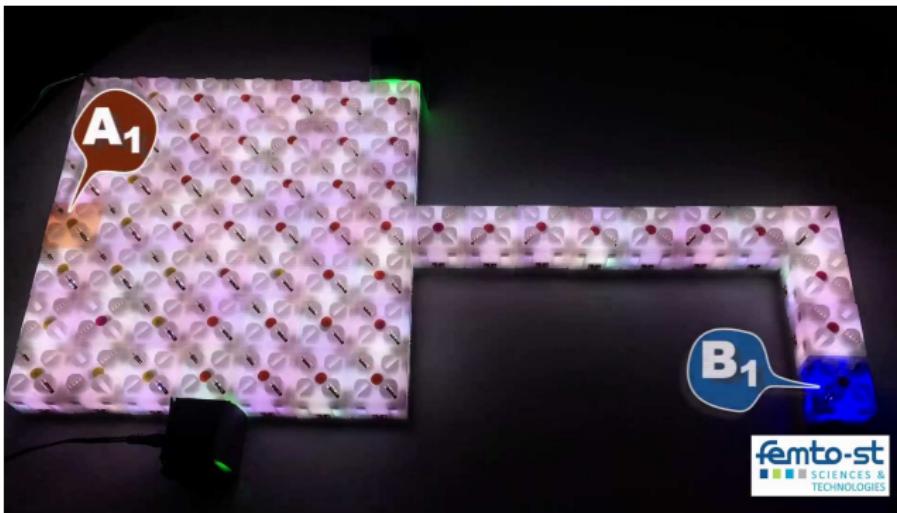
Single-source shortest paths are computed using distributed breadth-first network traversals (based on [Cheung, 1983]).

Examples - One-step 2D Shape



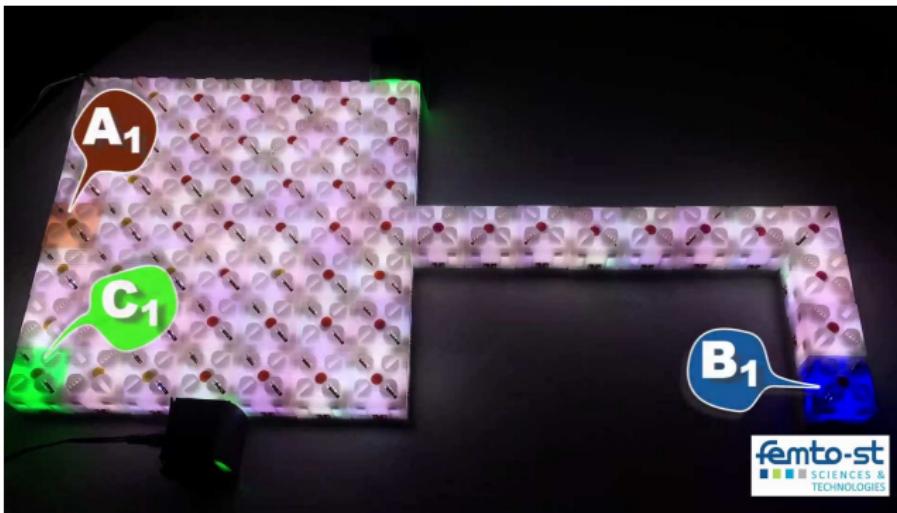
*60 Blinky Blocks

Examples - One-step 2D Shape



*60 Blinky Blocks

Examples - One-step 2D Shape



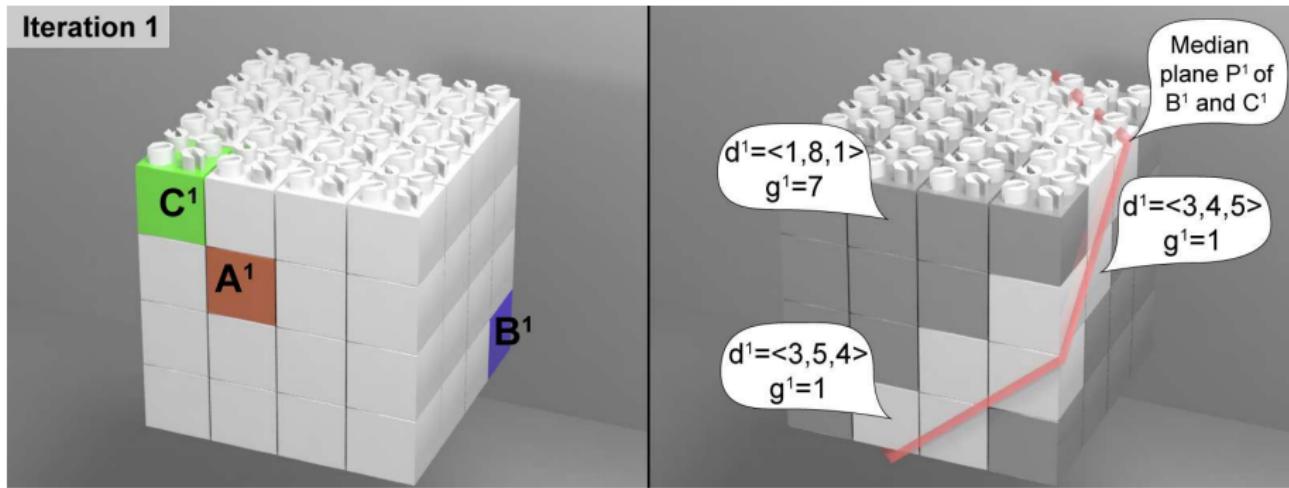
*60 Blinky Blocks

Examples - One-step 2D Shape



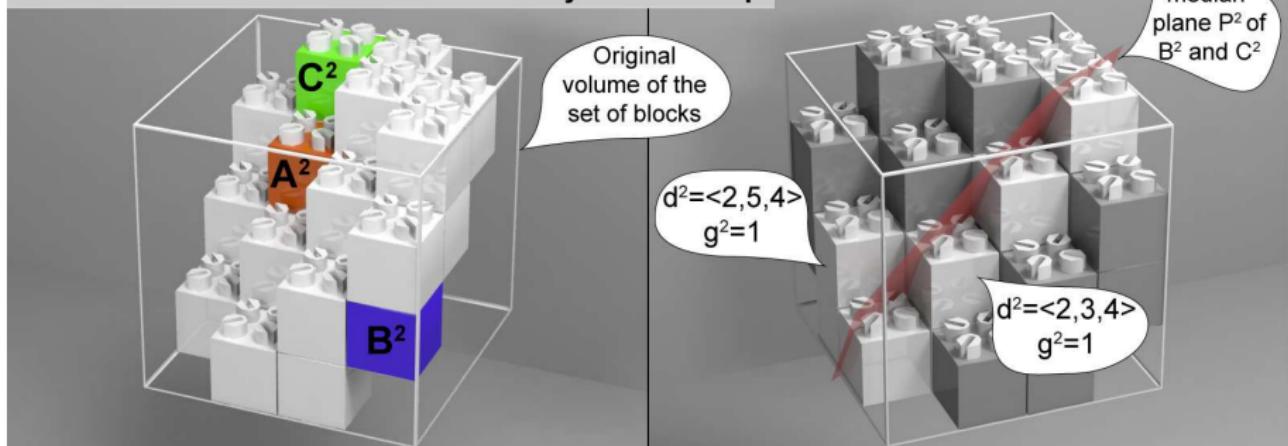
*60 Blinky Blocks

Examples - More complex



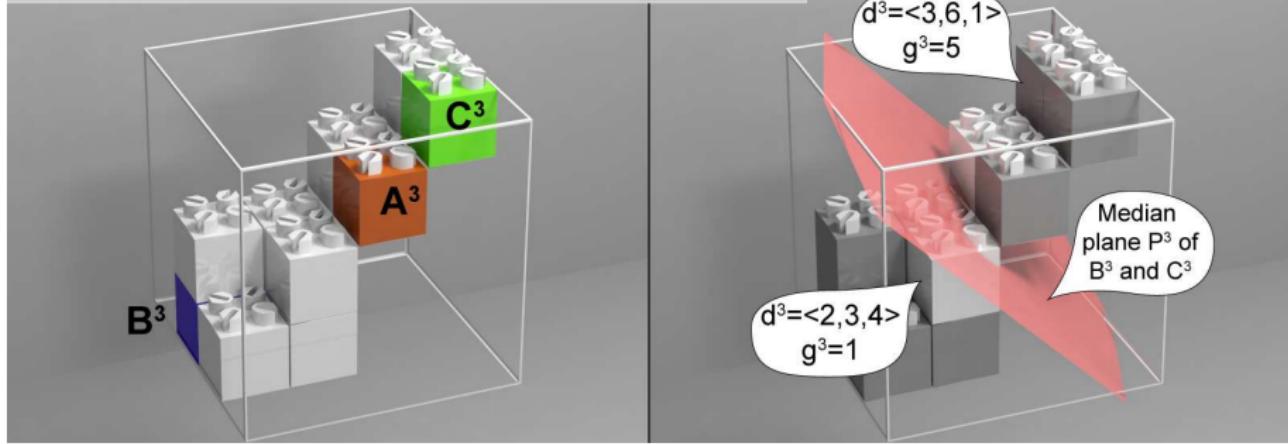
Examples - More complex

Iteration 2: We hide blocks eliminated by the first step.



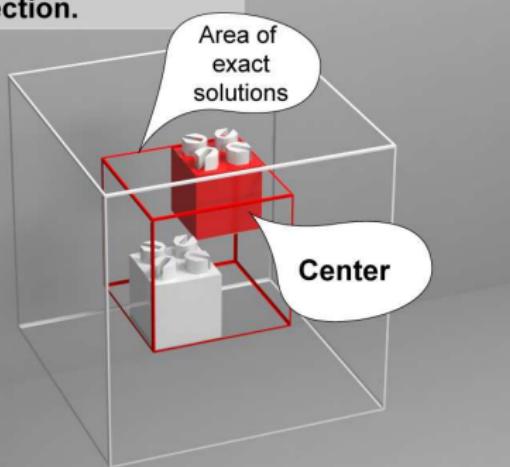
Examples - More complex

Iteration 3: We hide blocks eliminated by previous steps.



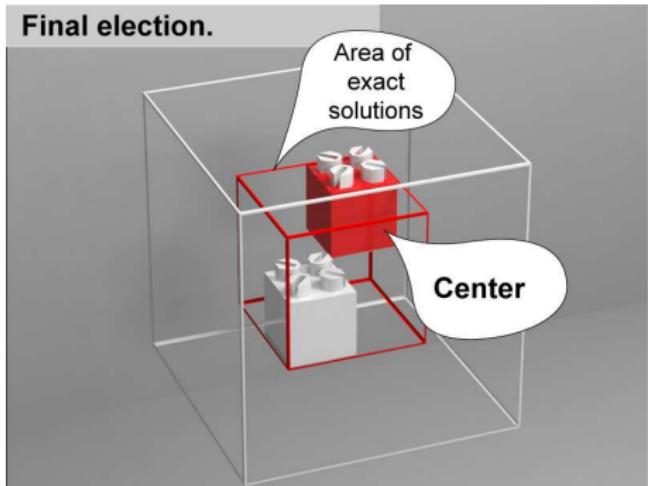
Examples - More complex

Final election.



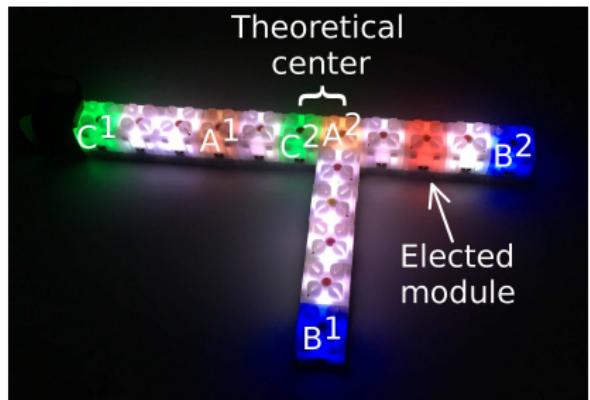
Examples - More complex

Final election.

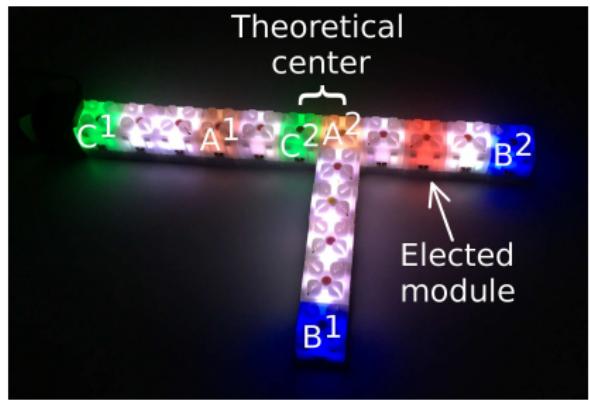


- ⇒ # steps increases with the thickness of the diameter
- ⇒ ABC-Center is more efficient in low-degree networks

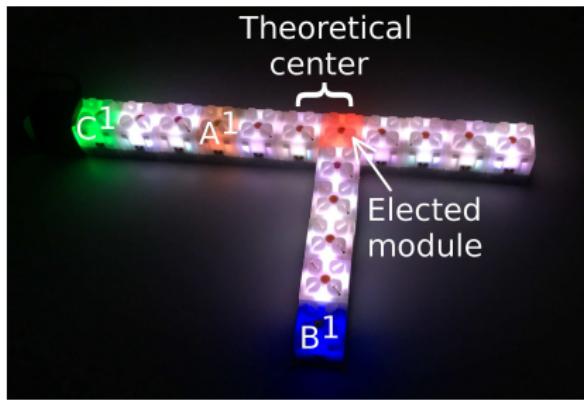
Tricky Case for Future Work



Tricky Case for Future Work

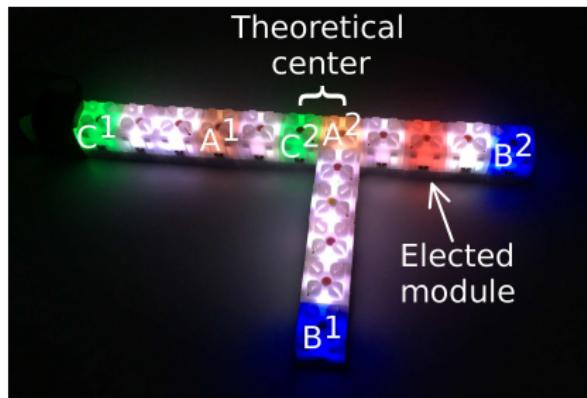


ABC-CenterV1/V2

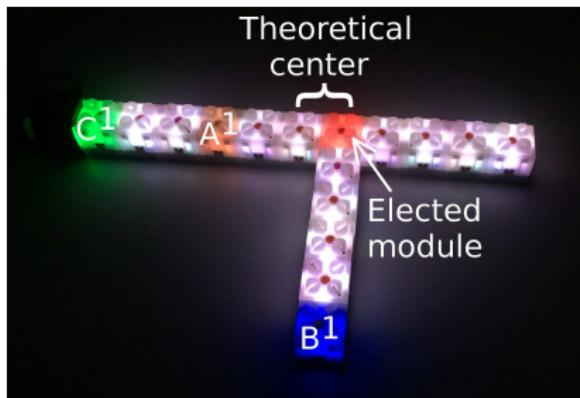


Envisioned but abandoned approach

Tricky Case for Future Work



ABC-CenterV1/V2



Envisioned but abandoned approach

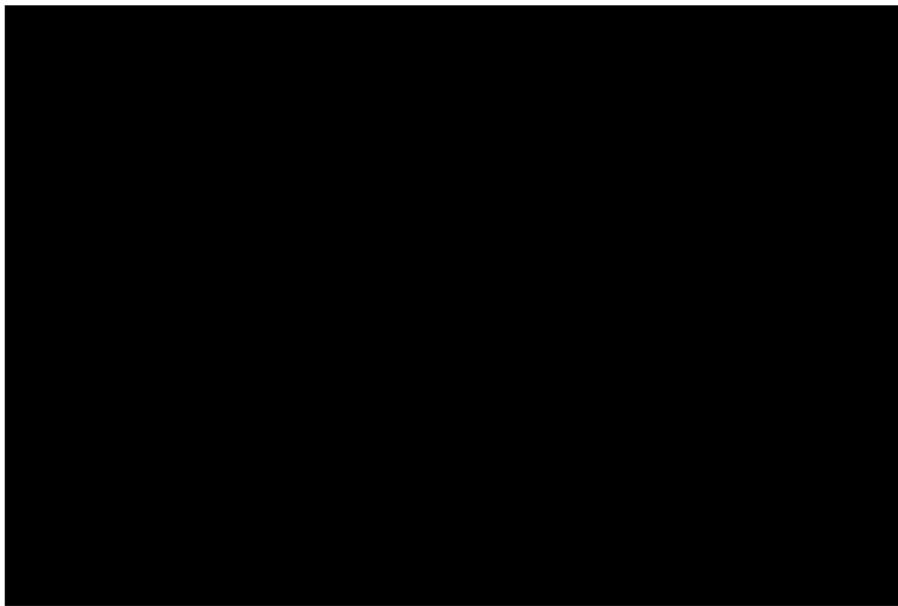
- ⇒ The envisioned approach solves this case but leads to less accuracy in experiments with random systems
- ⇒ This specific case should be investigated in future work!

Subsection Outline

3 Centrality-based Leader Election

- Problem and Motivations
- Related Work and Contributions
- ABC-CenterV2
- Evaluation
 - ABC-CenterV1 on Hardware
 - Simulation Fidelity
 - Simulation Evaluation
- Conclusion

ABC-CenterV1 on Hardware



VisibleSim Simulator Fidelity

Statistical simulation model based on measurements

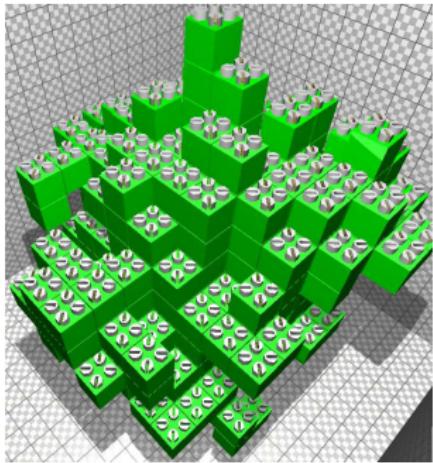
- Processing time
- Communication time

Shape	Size (module)	average ABC-CenterV1 execution time ± standard-deviation (ms)		Relative average simulation precision
		Hardware	Simulator	
Line	5	234 ± 1	244 ± 3	95.7%
	10	545 ± 5	544 ± 5	99.8%
	50	2873 ± 23	2885 ± 17	99.6%
Square	9	598 ± 45	588 ± 14	98.3%
	25	1117 ± 30	1119 ± 27	99.8%
	49	1684 ± 48	1686 ± 44	99.9%
Cube	27	1229 ± 56	1214 ± 31	98.8%
	64	1927 ± 51	1941 ± 33	99.3%
Dumbbell	59	1262 ± 56	1252 ± 57	99.2%

→ VisibleSim accurately simulates our algorithm on the Blinky Blocks

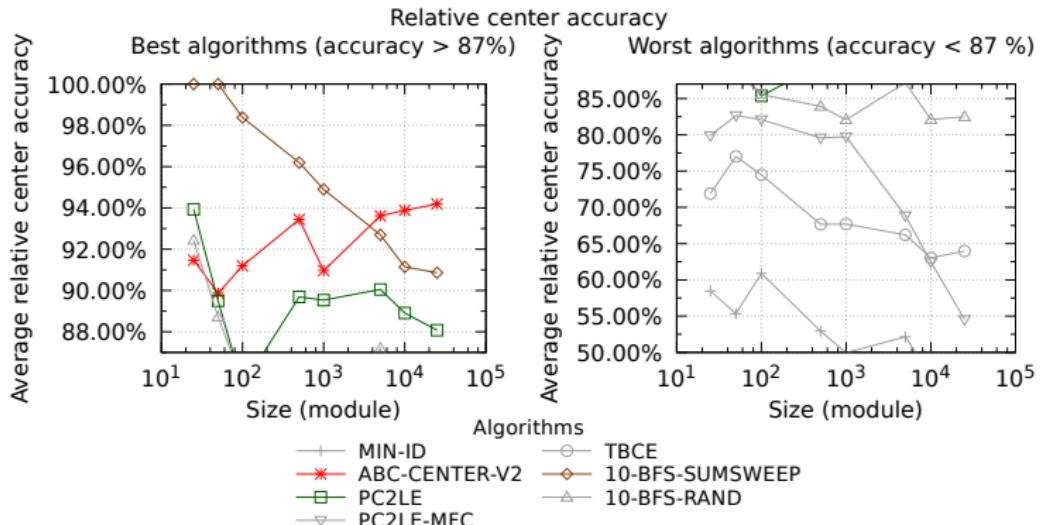
Simulation Evaluation

- Simulations using VisibleSim
- Comparisons with other algorithms
 - ▶ Random: MIN-ID [Raynal, 2013]
 - ▶ Exhaustive: BARYCENTER [Mamei et al., 2005]
 - ▶ Approximations:
 - TBCE [Kim and Wu, 2013]
 - k -BFS-RAND, our distributed version of [Eppstein and Wang, 2001]
 - PC2LE with [Garin et al., 2012] 's probabilistic counter
- Systems
 - ▶ Random compact shapes
 - ▶ Different sizes: 10 to 25,000 modules
 - ▶ Statistics on multiple independent runs
- Evaluation criteria
 - ▶ Accuracy
 - ▶ Cost
 - Execution time
 - Number of messages
 - Memory usage



Relative accuracy

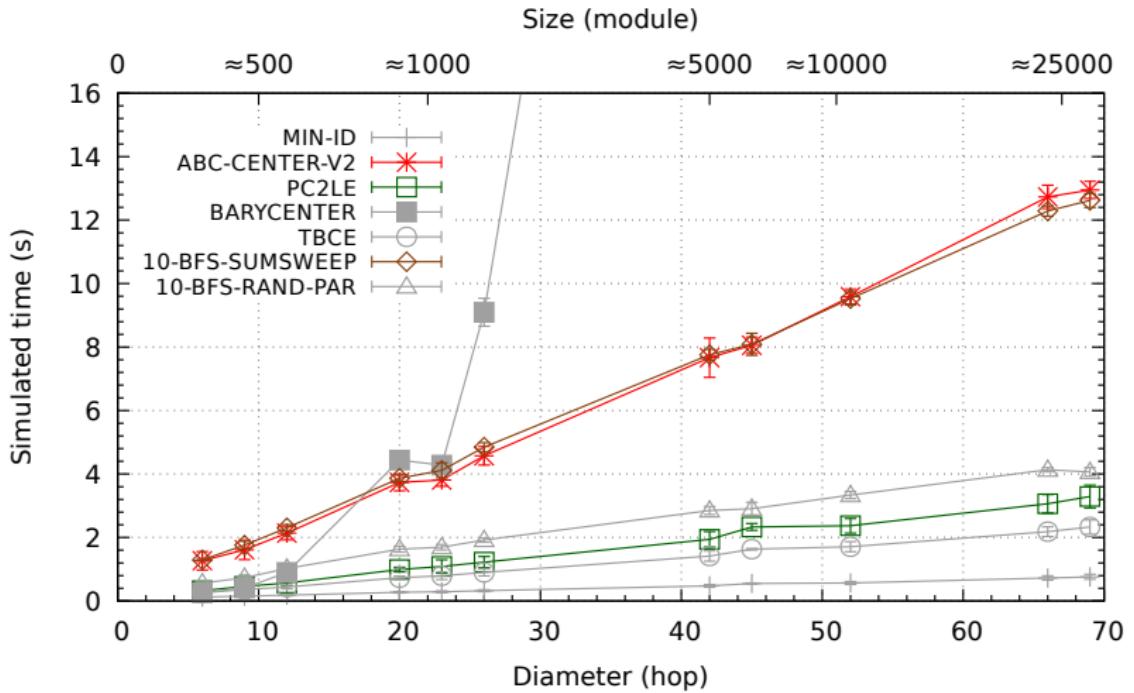
$$\text{relative center accuracy} = 1 - \left| \frac{\text{ecc}(\text{center}) - \text{ecc}(\text{elected node})}{\text{ecc}(\text{center})} \right|$$



⇒ Our algorithms:

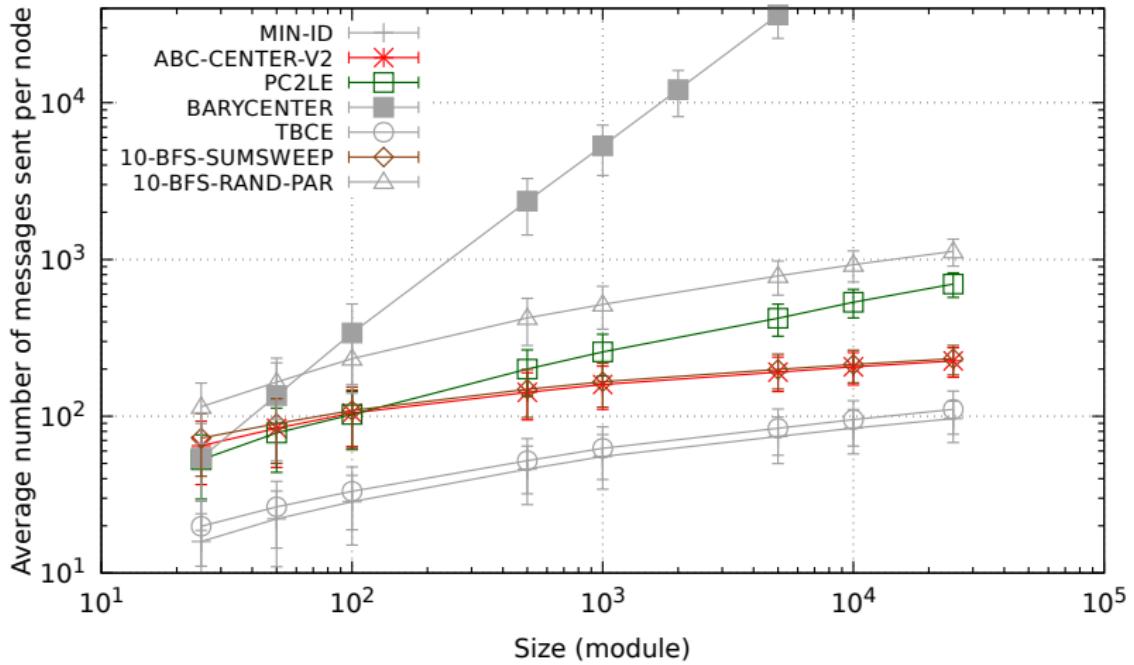
most precise approximation algorithms (accuracy > 85%)

Execution Time



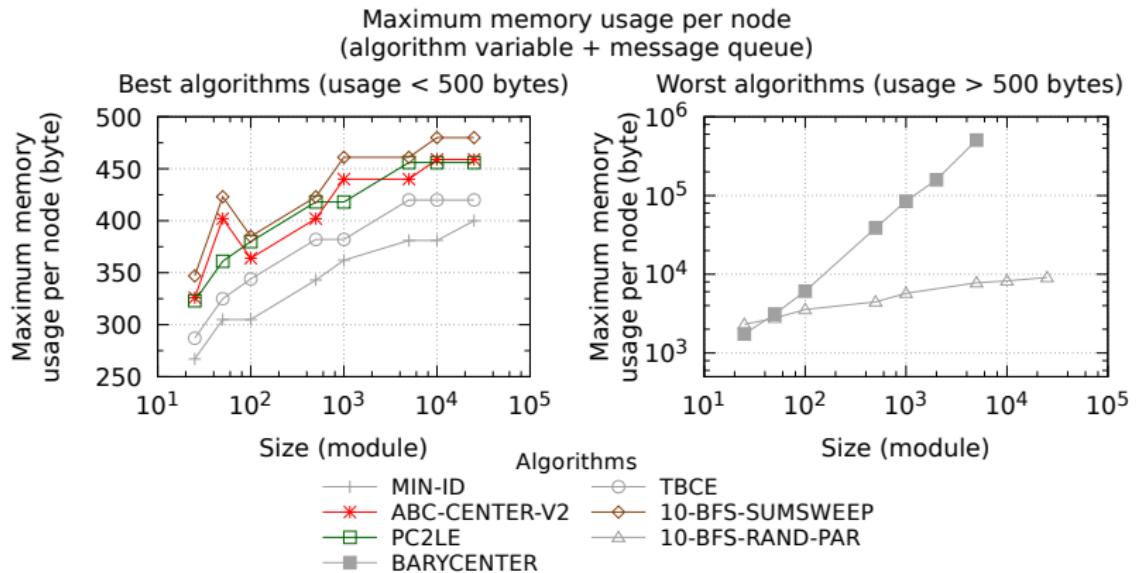
⇒ A good trade-off between cost and accuracy

Average number of messages per module



⇒ A good trade-off between cost and accuracy

Memory usage



⇒ Our algorithms have a bounded and limited memory usage (< 500 bytes per module)

Conclusion

- 3 efficient distributed centrality-based leader election algorithms
 - ▶ k -BFS SumSweep, ABC-Center and PC2LE
- Robustness to network dynamics
- Formal analysis of the performance (time, message and memory)
- Evaluation
 - ▶ Hardware: 5 to 64 modules
 - ▶ Simulations: 10 to 25,000 modules
 - Most precise approximation algorithms ($> 85\%$ relative center accuracy)
 - Good cost-accuracy trade-off
- Limits
 - ▶ Experimental evaluation of the accuracy
 - ▶ Bad cases?
 - ▶ Efficiency in other platforms with different network structures?
 - Our algorithms: inspired from external-graph analysis algorithms used to study large-scale real-world networks (e.g., YahooWeb, AS networks)

Section Outline

- 1 Introduction
- 2 Research Environment: Hardware and Simulation Tools
- 3 Centrality-based Leader Election
- 4 Time Synchronization
 - Problem and Motivation
 - Assumptions
 - Related Work
 - Contribution: The Modular Robot Time Protocol (MRTP)
 - Evaluation
 - Conclusion
- 5 Self-Reconfiguration
- 6 Conclusion and Future Works

Problem

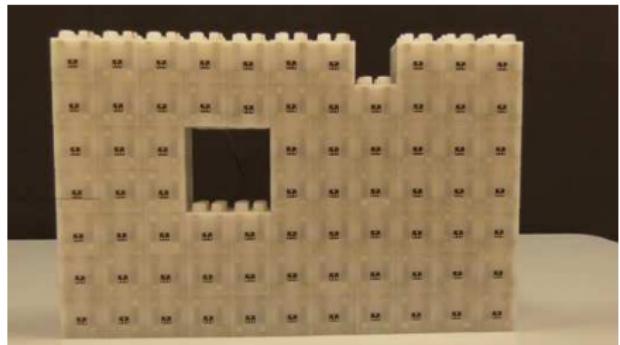
- Problem: maintain a global timescale through the system



- Challenges:
 - ▶ Imperfect hardware clocks (clock drift, skew, offset and noise)
 - ▶ Handling communication delays

Motivation

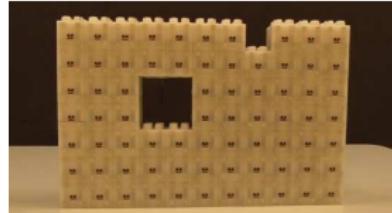
Distributed coordination (e.g., distributed bitmap scroller)



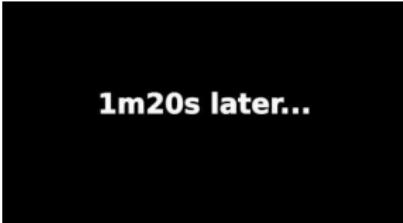
72-Blinky-Blocks scroller
synchronized with our protocol
(MRTP)

Unsynchronized scroller:

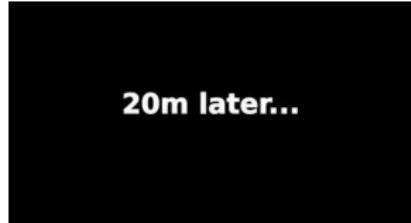
Start:



1min20:



20min:



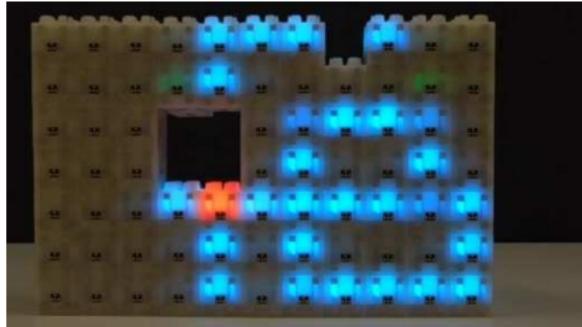
⇒ This application requires a global timescale

Assumptions

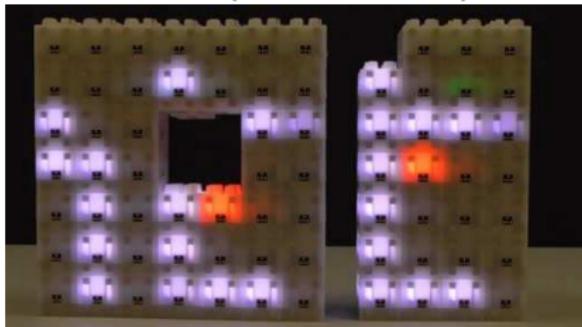
Illustrated on the Blinky Blocks modular robot, but suitable for networked embedded devices with:

- Local clocks
 - ▶ Nearly identical precision/resolution (potentially poor)
 - Blinky Blocks: 1% precision, 1 ms resolution
- Network
 - ▶ Neighbor-to-Neighbor communications
 - ▶ Potentially: large network size and large diameter
 - ▶ Fairly stable

Split:



Merge (after 2 hours):



Related Works

Name	Domain	Architecture	Infras-structure	Synchronization Technique	Clock Skew Compensation
NTP [Mills, 1991]	Computer Networks	Master/Slave Master(s): pre-configured	Tree	(Multi-hop) round-trip messages with frame-level timestamps and statistics	Phase/frequency locked loops
PTP [IEEE, 2008]		Master/slave Master: best clock		Round-trip with low-level timestamps and per-hop delay compensation	
RBS [Elson et al., 2002]	Sensor Networks	Master/Slave	Clustering	Reference broadcast	Linear model
ATS [Schenato et al., 2011]		Fully distributed	/	Average-based consensus. Byte-level timestamps	
MTS [He et al., 2014a, He et al., 2014b]				Extremum-value based consensus. Byte-level timestamps	
TPSN + MLE [Leng and Wu, 2010]		Master/slave	Tree	Recursive per-hop synchronization. Round-trip with frame-level timestamps and statistics	
FTSP [Maróti et al., 2004]		Master/slave Master: minimum-id implicit election	/	Periodic asynchronous broadcasts. Byte-level timestamps and statistics	
PulseSync [Lenzen et al., 2015]				Fast flooding using broadcast. Byte-level timestamps	

Contribution: MRTP	Modular Robotic	Master/Slave Master: central	Breadth-first spanning-tree	Fast recursive per-hop synchronization. Compensation comm. delays: best for target system	Linear model
-----------------------	--------------------	---------------------------------	--------------------------------	--	--------------

Subsection Outline

4 Time Synchronization

- Problem and Motivation
- Assumptions
- Related Work
- Contribution: The Modular Robot Time Protocol (MRTP)
 - Workflow
 - The Predictive Method to Compensate for Communication Delays
- Evaluation
- Conclusion

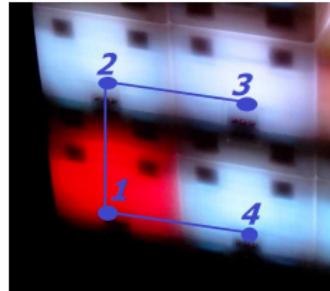
Contribution: The Modular Robot Time Protocol (MRTP)

① Initialization

- ① Central time master election
- ② Breadth first spanning-tree construction

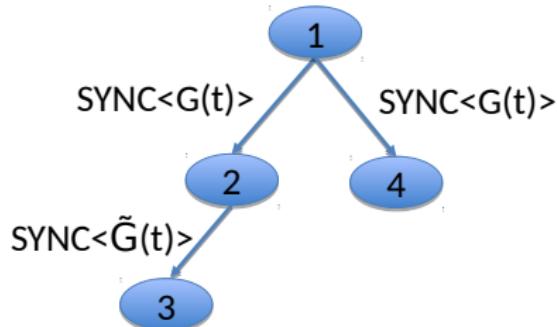
② Periodic synchronization

- ▶ Best-suited method to compensate for communication delays
- ▶ Clock skew compensation using linear regression

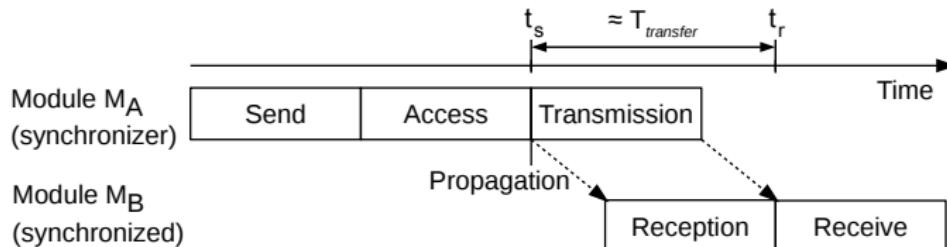


$G(t)$: global time, held by the time master

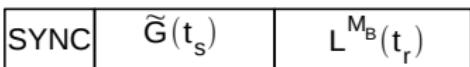
$\tilde{G}(t)$: disseminated estimation of $G(t)$



The Predictive Method to Compensate for Communication Delays



Message
format :



$$\tilde{G}(t_r) = \tilde{G}(t_s) + T_{transfer}$$

Linear regression on a window of synchronization points $< \tilde{G}(t), L^{M_B}(t) >$

$$G^{M_B}(t) = a^{M_B}(t) * L^{M_B}(t) + b^{M_B}(t)$$

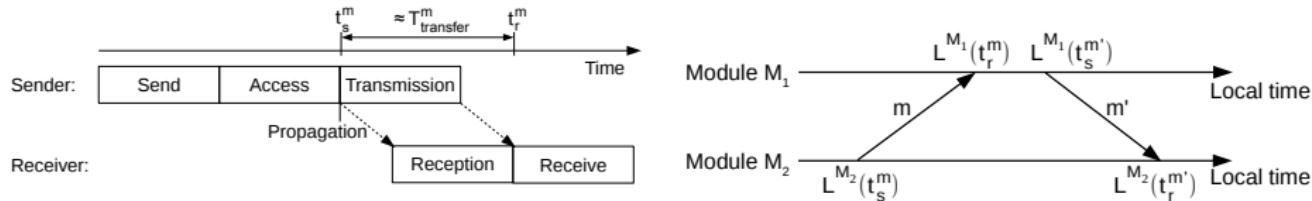
Subsection Outline

4

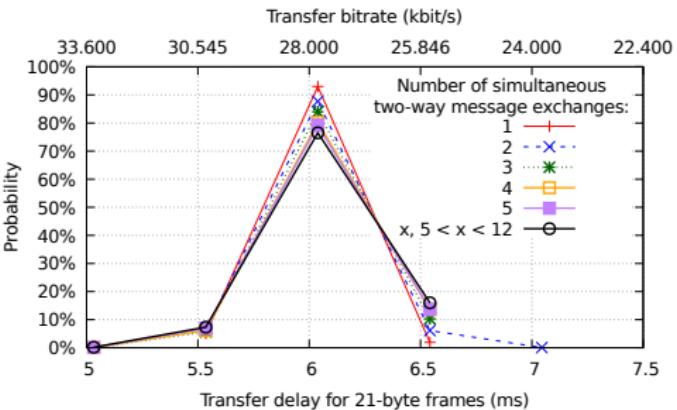
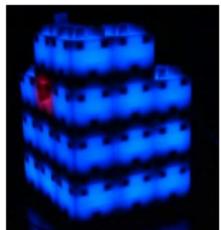
Time Synchronization

- Problem and Motivation
- Assumptions
- Related Work
- Contribution: The Modular Robot Time Protocol (MRTP)
- Evaluation
 - Blinky Blocks: Compensation for Communication Delays?
 - Hardware Results
 - Simulation Fidelity
 - Simulation Results
- Conclusion

The Blinky Blocks Communication delays



$$T_{transfer} \approx \frac{(L^{M_2}(t_r^{m'}) - L^{M_2}(t_s^m)) - (L^{M_1}(t_s^{m'}) - L^{M_1}(t_r^m))}{2}$$



300,000 two-way exchanges

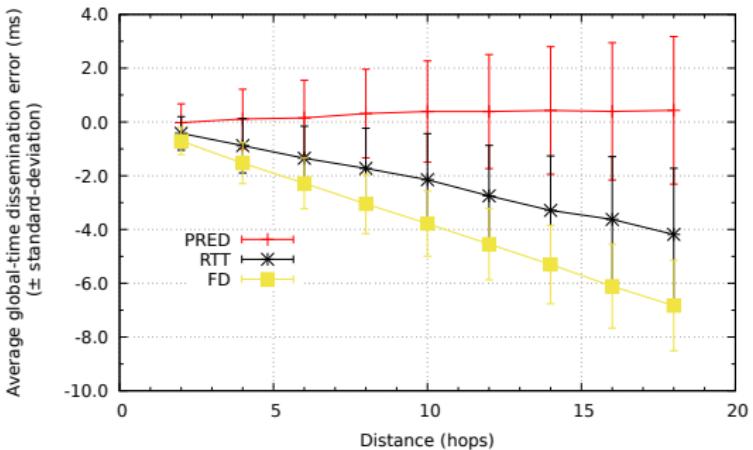
$$T_{transfer} \approx \frac{\text{frame size}}{28.00}$$

⇒ Predictable transfer delays

Best Method to Compensate for the Blinky Blocks Communication Delays?

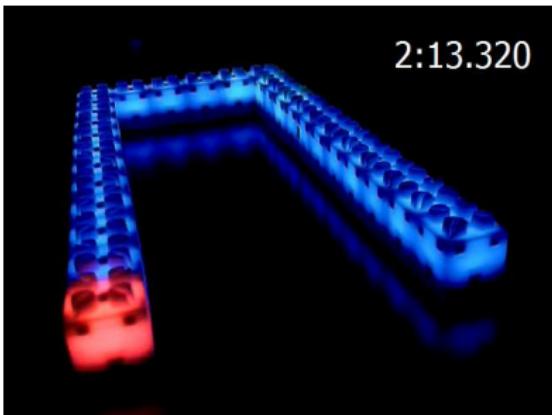
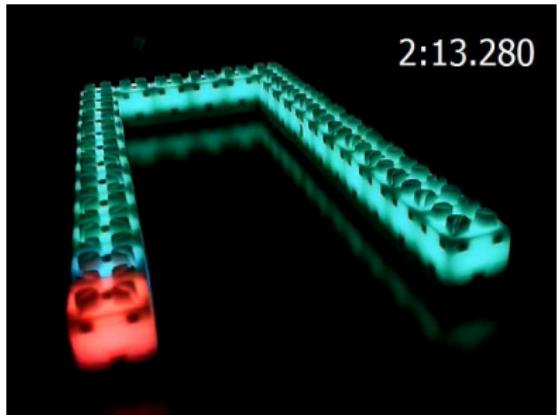
Methods applicable on the Blinky Blocks

- Predictive (PRED)
- Round-Trip Time (RTT): used in TPSN
- Frame Delimiter (FD): used in ATS and MTS



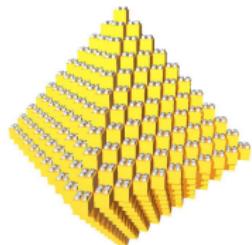
⇒ PRED is the most accurate and uses a single message per hop

Synchronizable Radius



⇒ The time master synchronizes modules in a 27-hop radius to < 40ms

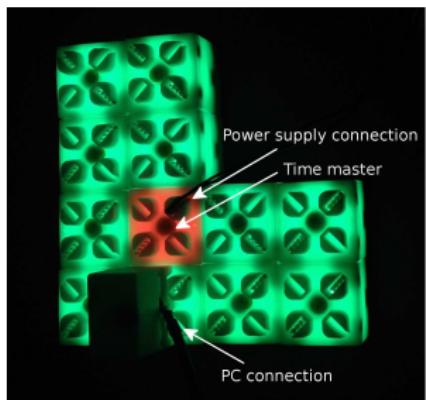
⇒ Extrapolation: with a central time master, MRTP can synchronize the 27-hop-radius ball system (octahedral shape with 27,775 modules) to < 40ms.



10-hop-radius ball
1561 Blinky Blocks

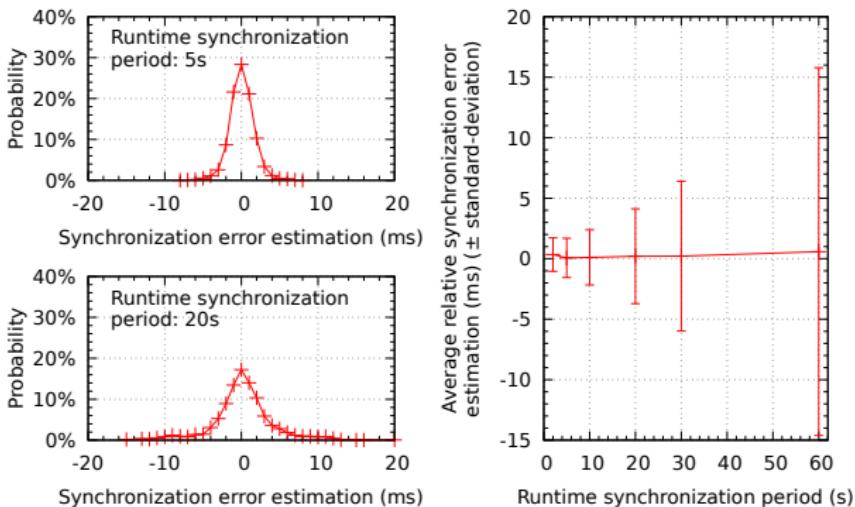
⇒ We will use simulations to show it!

Impact of the Synchronization Period



Duration: 1 hour

Window size: 5



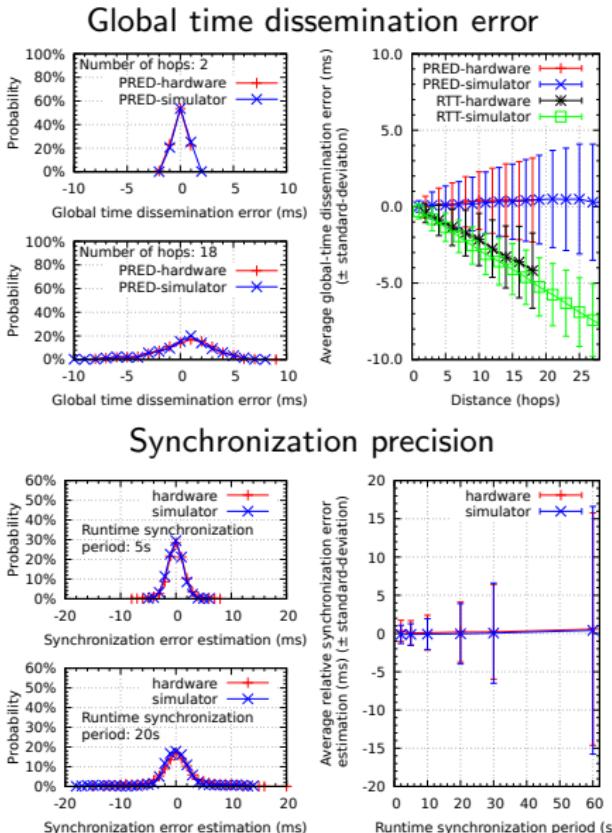
- ⇒ MRTP keeps the system synchronized to a few milliseconds even with a synchronization period of 60 seconds
- ⇒ The synchronization period depends on the desired precision. A long period uses less system resources.

VisibleSim Simulator Fidelity

Statistical models:

- Clock
 - ▶ Skew
 - ▶ Drift
 - ▶ Noise (replayed)
- Communication time
- Processing time

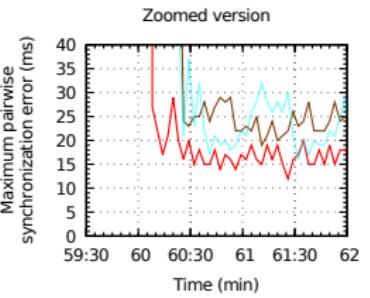
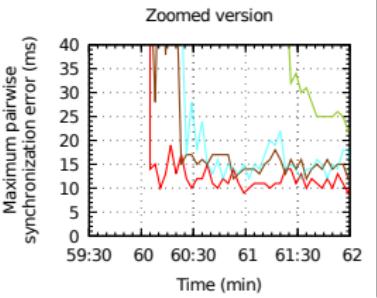
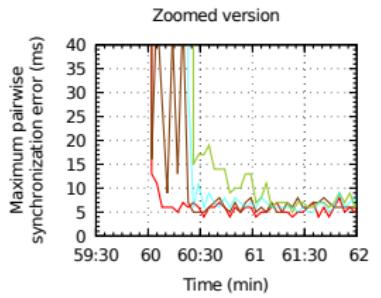
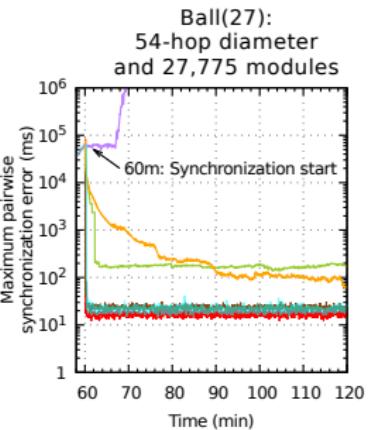
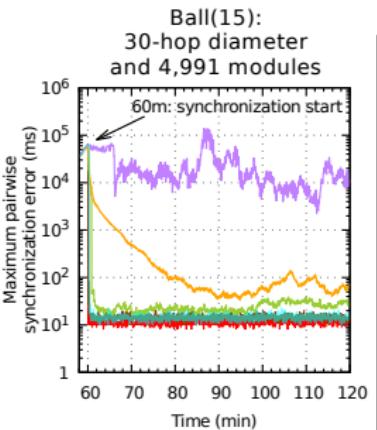
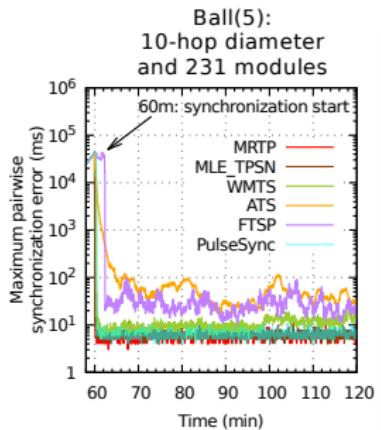
⇒ Accurate simulation



Simulation Evaluation

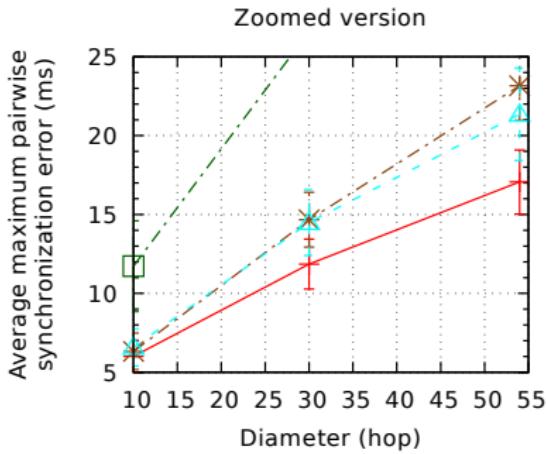
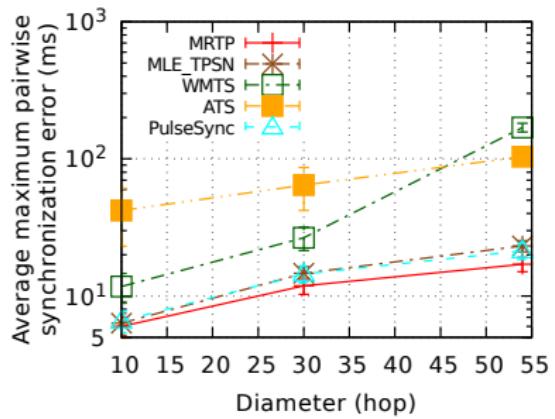
- Simulations using VisibleSim
- Comparisons with ported version of existing protocols
 - ▶ Decentralized: ATS [Schenato et al., 2011], WMTS [He et al., 2014b]
 - ▶ Master/Slave with tree: TPSN + MLE [Leng and Wu, 2010]
 - ▶ Master/Slave but infrastructureless: PulseSync [Lenzen et al., 2015]
- Experiments:
 - ▶ Ball systems (octahedrally-shaped)
 - Size: 231 to 27,775 modules
 - Diameter: 10 to 54 hops
 - ▶ Scenario:
 - Left unsynchronized for 1 hour
 - Then synchronization every 5 seconds during 1 hour
 - Measurements the maximum pairwise synchronization error every 3 seconds
- Evaluation criteria
 - ▶ Time of convergence
 - ▶ Synchronization precision
 - ▶ Messages

Convergence Time



⇒ MRTP converges quickly

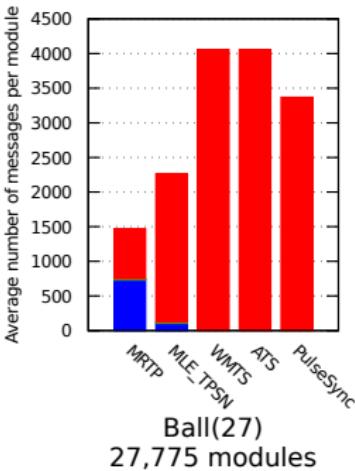
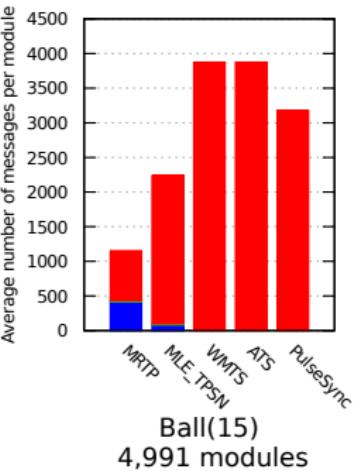
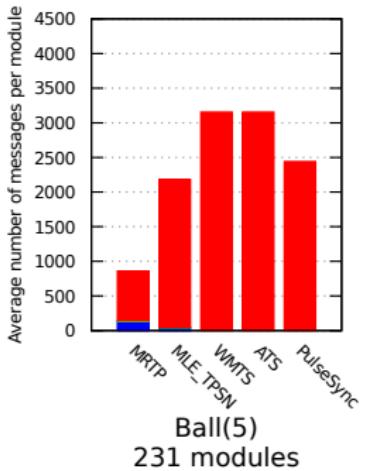
Synchronization Precision after Convergence



⇒ MRTP is the most precise protocol

⇒ MRTP synchronizes the 27,775 module system to 17ms in average

Messages



Leader Election

Types of message:
Infrastructure
Synchronization

⇒ MRTP uses few messages

Conclusion

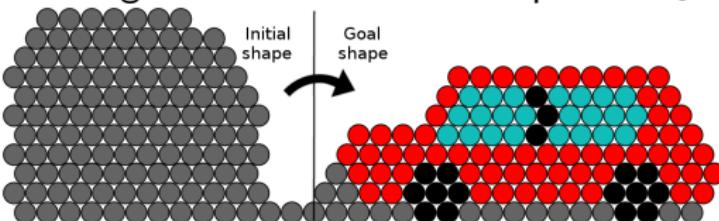
- The Modular Robot Time Protocol (MRTP)
- Evaluation
 - ▶ Hardware
 - ▶ Simulations
 - Most precise protocol with a fast convergence
 - Synchronizes a 27,775 module system with a 54-hop diameter to $< 20ms$
 - Uses less messages in compact systems than existing algorithms
- Limit
 - ▶ Targets fairly stable systems

Section Outline

- 1 Introduction
- 2 Research Environment: Hardware and Simulation Tools
- 3 Centrality-based Leader Election
- 4 Time Synchronization
- 5 Self-Reconfiguration
 - Problem and Motivations
 - System Model and Assumptions
 - Related Work
 - Contribution: The Cylindrical-Catoms Self-Reconfiguration (C2SR) Algorithm
 - Evaluation
 - Conclusion
- 6 Conclusion and Future Works

Problem and Motivations

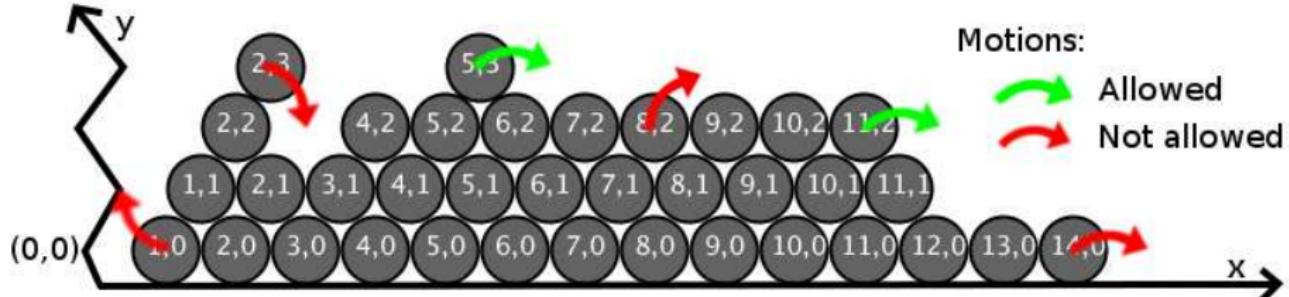
- Given an initial shape \mathcal{I} , a goal shape \mathcal{G} and the mechanical constraints
- Problem: self-reconfigure the robot from shape \mathcal{I} to \mathcal{G}



- Challenge: Distributed coordination
 - Effectiveness: reach \mathcal{G}
 - Prevent collisions
 - Prevent deadlock
 - Efficiency
 - Execution time
 - Motion
 - Communication
- Motivation: programmable matter

System Model and Assumptions

- Hexagonal lattice:
 - ▶ Every module knows its coordinates and its neighbor ones
- Communications:
 - ▶ Asynchronous
 - ▶ Neighbor-to-neighbor only
- Motions:
 - ▶ Asynchronous
 - ▶ Mechanical constraints
 - ▶ No presence sensor: prevent collisions using communications
- Failure-free environment (modules, communications, motions, lattice)
- Every module stores a complete representation of \mathcal{G}



Related Work

Cite	Shapes	Constraints on module movement	Collision and deadlock avoidance
[Walter et al., 2000]	chain to chain	relaxed	centralized pre-computation, synchronous rounds
[Walter et al., 2005, Bateau et al., 2012]	chain to 2D	relaxed	centralized pre-computation, synchronous rounds
[Lakhlef et al., 2015]	chain to square	relaxed and very-relaxed	predefined shape construction
[Lakhlef et al., 2014]	arbitrary to square	relaxed	predefined shape construction
[Wong et al., 2015]	compact 2D to chain	relaxed	touch sensors, synchronous rounds, single direction
[De Rosa et al., 2006]	2D	relaxed	
[Rubenstein et al., 2014]	horizontal 2D compact	very relaxed	collision allowed (swarm robotic)
Contribution: C2SR	vertical 2D compact	strong	messages, single direction

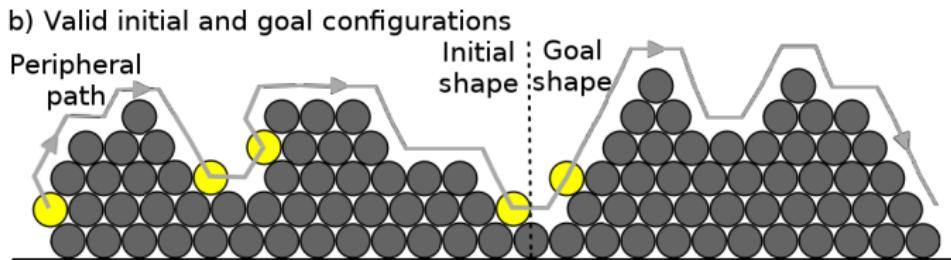
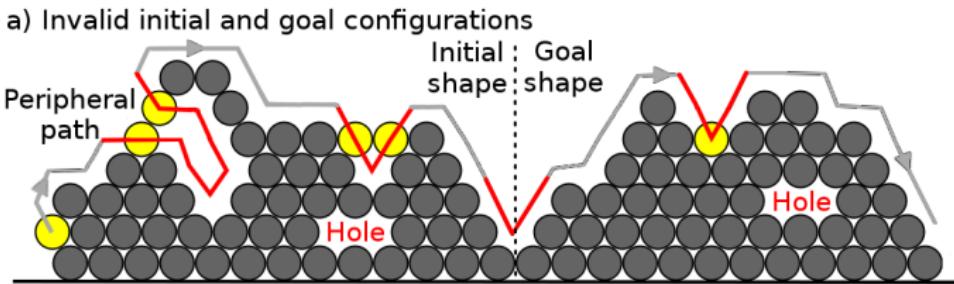
Subsection Outline

5 Self-Reconfiguration

- Problem and Motivations
- System Model and Assumptions
- Related Work
- Contribution: The Cylindrical-Catoms Self-Reconfiguration (C2SR) Algorithm
 - Shape Admissibility Conditions
 - Algorithm
- Evaluation
- Conclusion

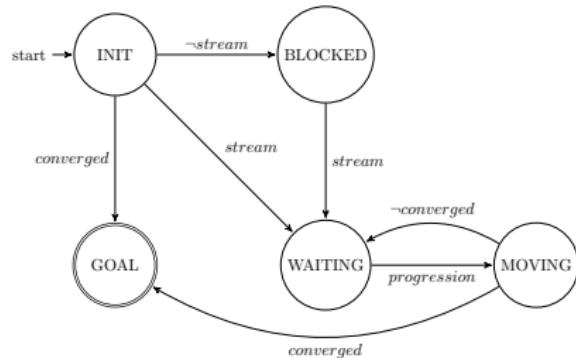
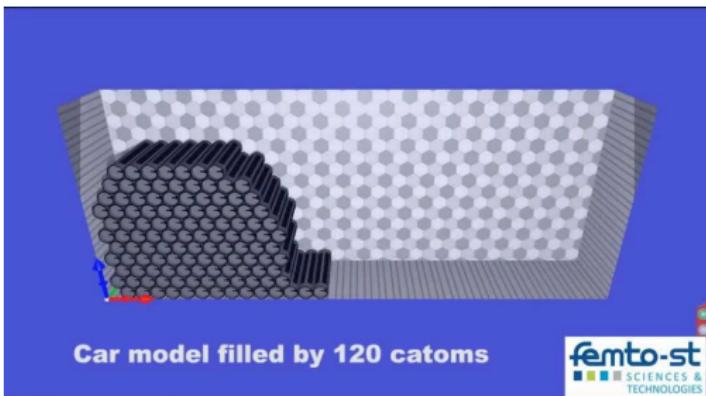
Shape Admissibility Conditions

- $|\mathcal{I}| \geq |\mathcal{G}|$
- \mathcal{I} and \mathcal{G} are next to each other and share some bottom cells
- No hole
- Wide peripheral path with no narrow passage



C2SR at a Glance

- Stream of moving modules on the periphery
- Maintains an empty cell between moving modules
- Layer-by-layer deconstruction/construction (for shapes with only continuous horizontal layers)
- Decentralized with local interactions only (< 5 hops)
 - ▶ Communications with modules geographically around the next position in the stream



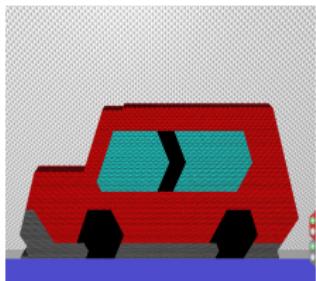
Subsection Outline

5 Self-Reconfiguration

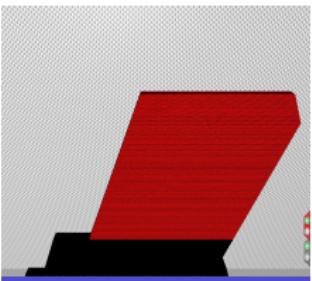
- Problem and Motivations
- System Model and Assumptions
- Related Work
- Contribution: The Cylindrical-Catoms Self-Reconfiguration (C2SR) Algorithm
- Evaluation
 - Experimental Setup
 - Simulation Results
- Conclusion

Experiment Setup

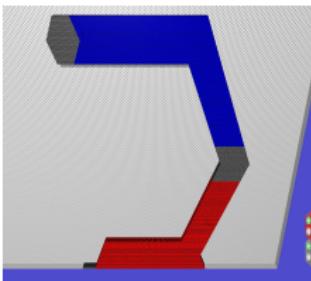
- VisibleSim
- Parameters (unless otherwise mentioned)
 - ▶ 4 shapes with different scale from 10^1 to 10^4 modules
 - ▶ Communication rate $\rightarrow \mathcal{N}(38.9\text{ kpbs}, 389\text{ bps})$ ($[3ms, 8ms]$)
 - ▶ Motion speed $\rightarrow \mathcal{N}(1.88mm \cdot s^{-1}, 0.0188mm \cdot s^{-1})$ ($[273ms, 285ms]$)
- Evaluation criteria
 - ▶ Effectiveness
 - ▶ Efficiency
 - Time
 - Motion
 - Communication



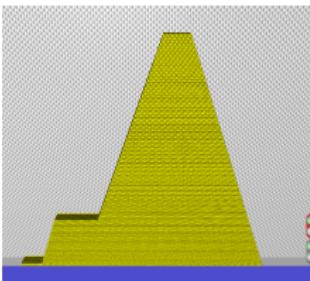
Car



Flag

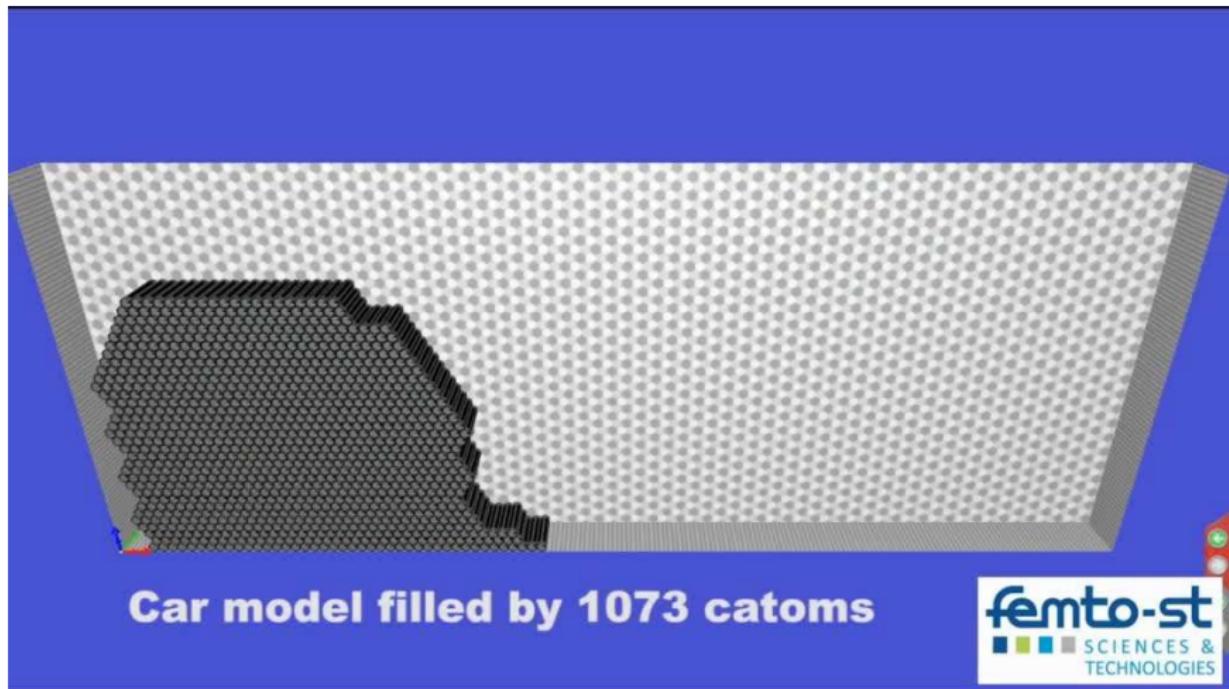


Magnet



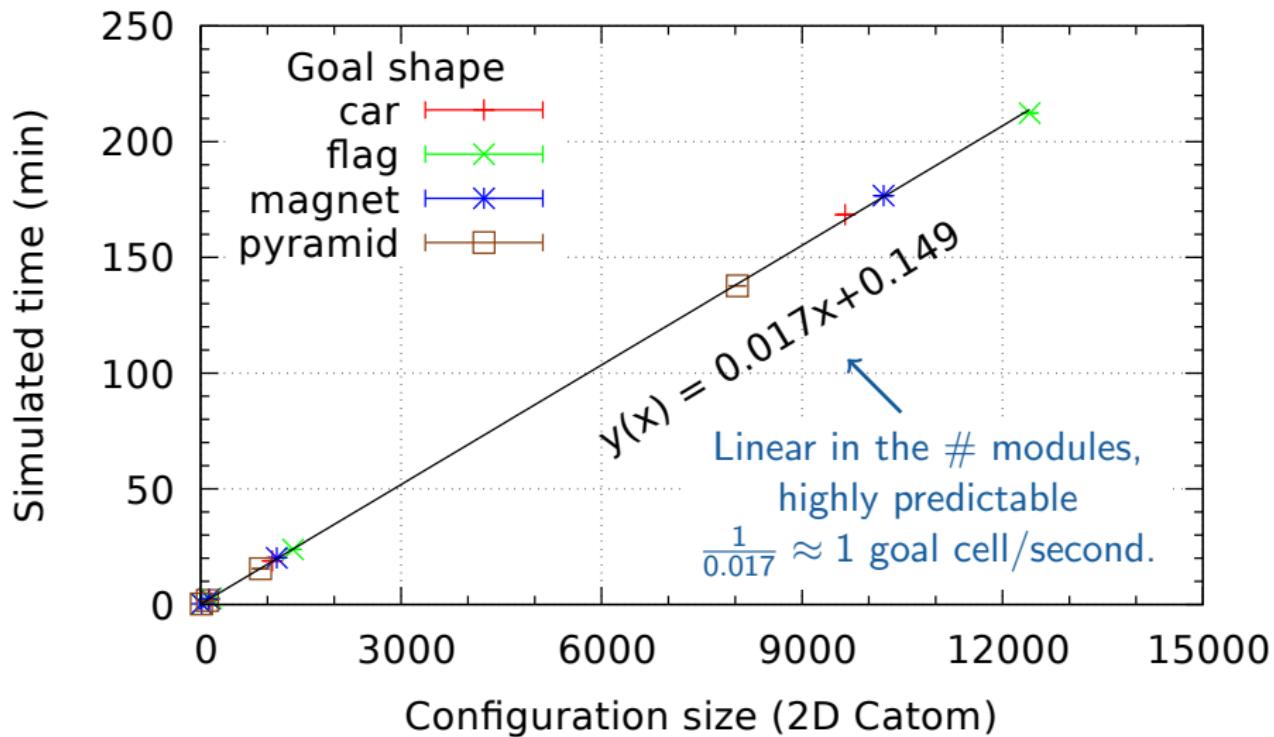
Pyramid

Effectiveness: C2SR in Video



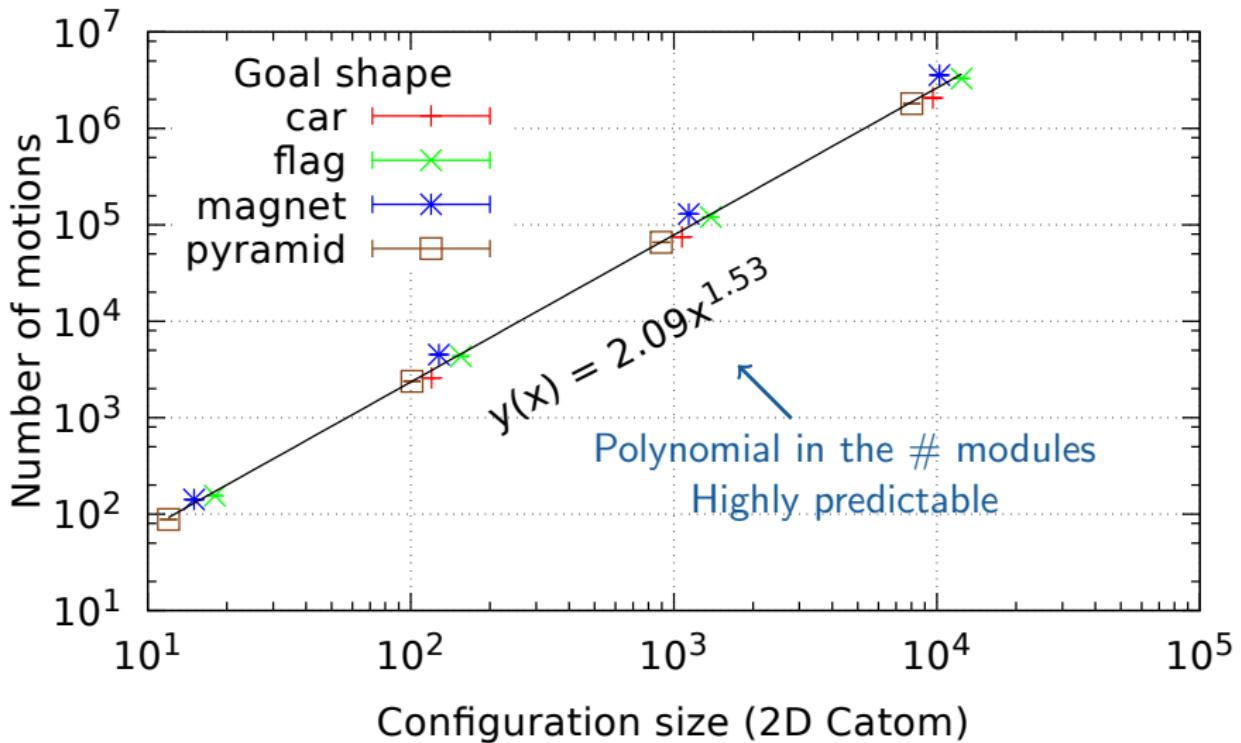
*Speed: x3

Execution Time (\pm standard-deviation)



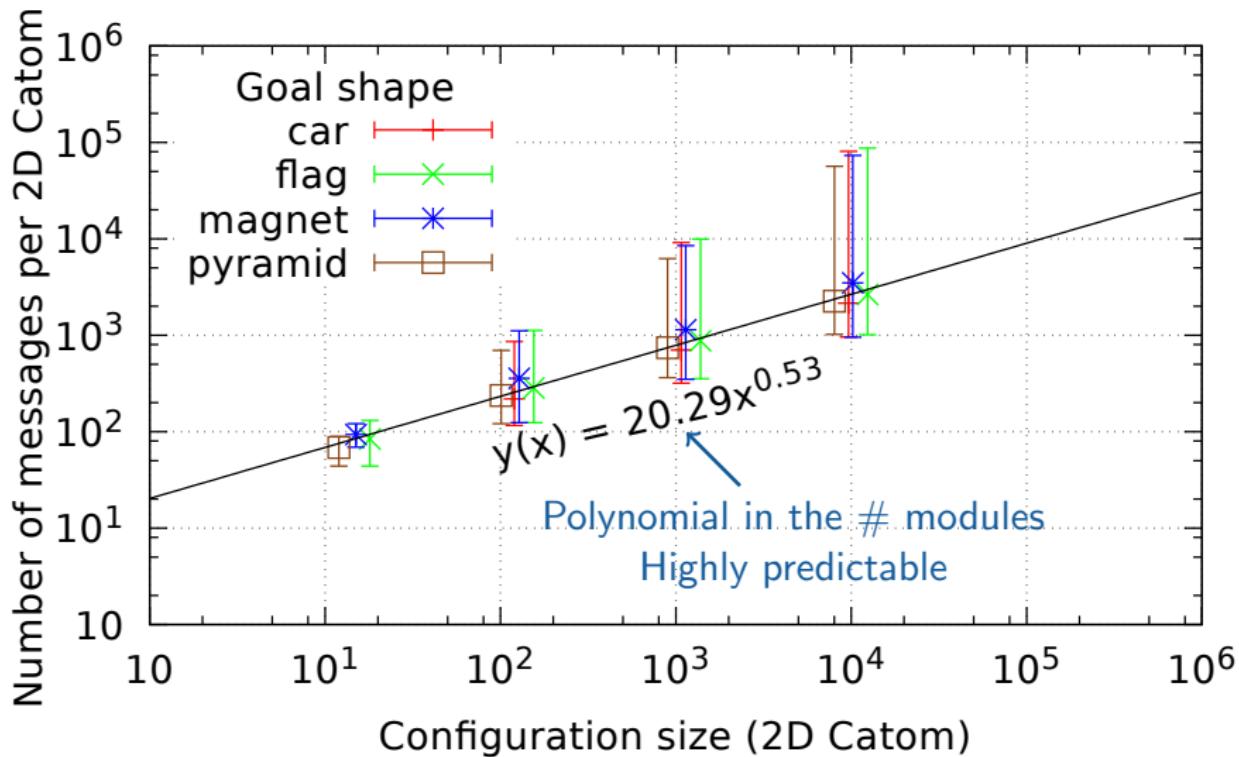
*Each point: 10 executions

Average Total Number of Motions (\pm standard-deviation)



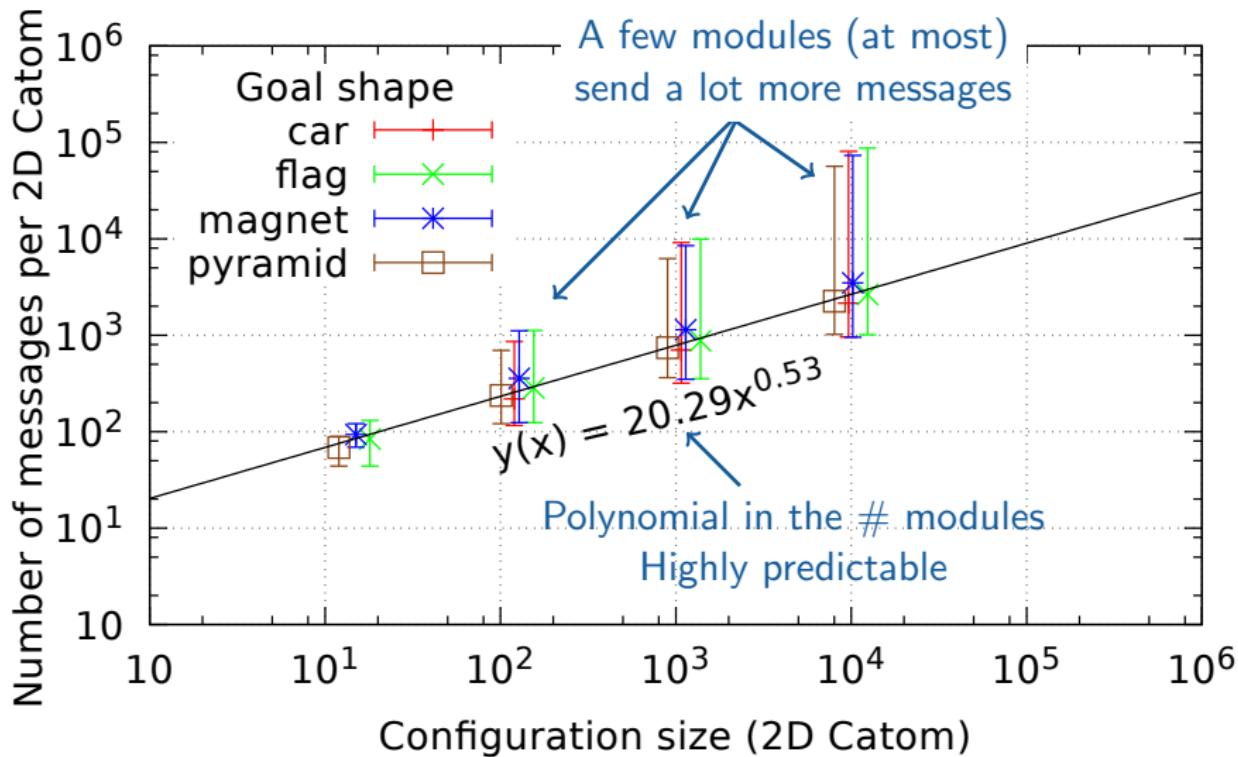
*Each point: 10 executions

Average Number of Messages per Catom (\pm min/max)



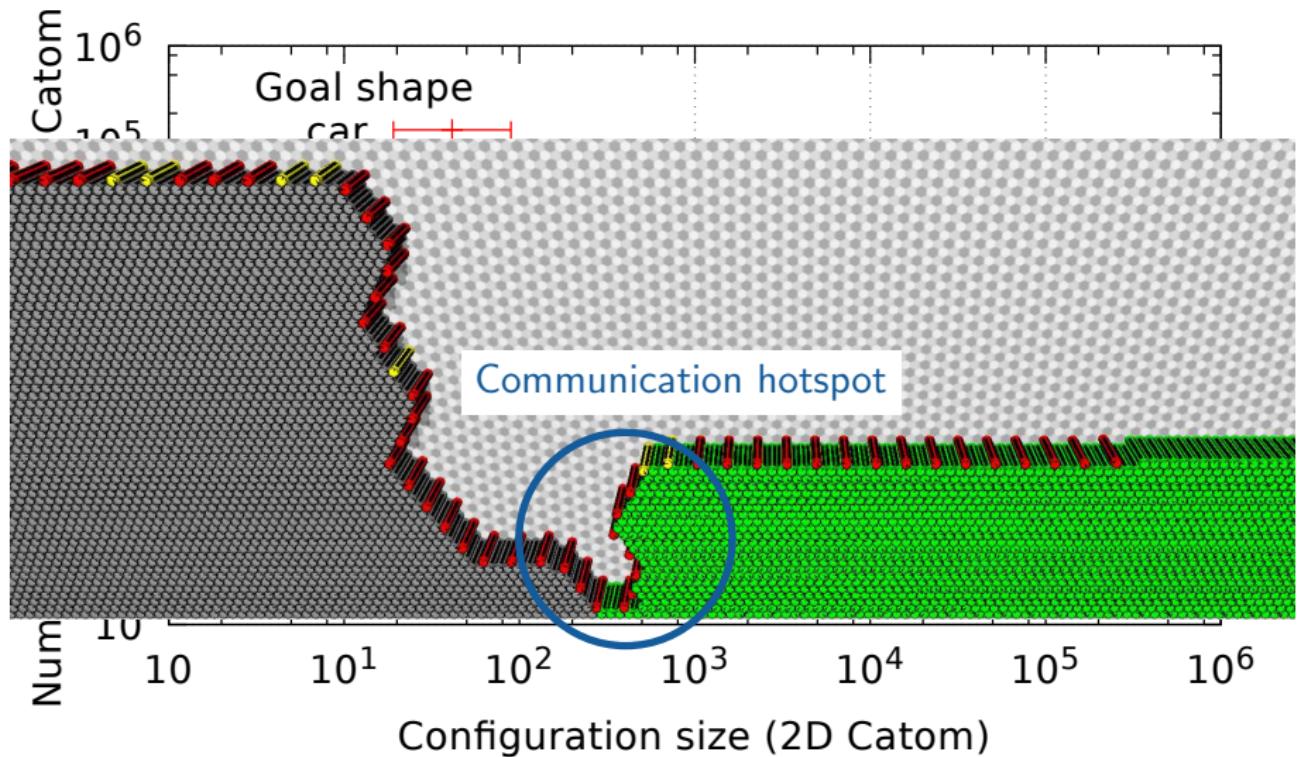
*Each point: 10 executions

Average Number of Messages per Catom (\pm min/max)



*Each point: 10 executions

Average Number of Messages per Catom (\pm min/max)



*Each point: 10 executions

Conclusion

- The Cylindrical-Catoms Self-Reconfiguration (C2SR) Algorithm
 - ▶ Effective: tested on different types of shapes
 - ▶ Scalable: tests with more than 10,000 modules
 - ▶ Nice properties:
 - Execution time seems linear in the size of the goal shape
 - Highly predictable: number of motions and messages
- Limits
 - ▶ Correctness and performance only shown experimentally
 - ▶ 2D systems only

Section Outline

- 1 Introduction
- 2 Research Environment: Hardware and Simulation Tools
- 3 Centrality-based Leader Election
- 4 Time Synchronization
- 5 Self-Reconfiguration
- 6 Conclusion and Future Works

Conclusion and Future Works (1)

- Contributions

- ▶ Three efficient algorithms for distributed centrality-based leader election
- ▶ The Modular Robot Time Protocol (MRTP)
- ▶ The 2D-Catoms Self-Reconfiguration (C2SR) algorithm

- Future works

- ▶ Centrality
 - Proofs on the accuracy of our algorithms
 - Deployment on other platforms with different network structures
 - More efficient management of network dynamics
- ▶ Time synchronization
 - Deployment on ensembles with more precise clocks
 - Consider both clock stability and centrality to select the time master
 - Time synchronization in more dynamic ensembles
 - Synchronized self-reconfiguration?
 - For now, PulseSync [Lenzen et al., 2015] can be used

Conclusion and Future Works (2)

- Future works (cont'd)

- ▶ Self-reconfiguration
 - Proof of correctness
 - Demonstration of the execution time linearity
 - Other models
 - Other mechanical constraints
 - 3D systems

Questions

Thank you for your attention!



Any question?

This work has been funded by the ANR/RGC
(contracts ANR-12-IS02-0004-01 and 3-ZG1F)