Automated Verification: Accomplishments Overview

- Integrate verification concepts and tools used in computer science to verify problems in distributed control.

- Explore the use of concepts from game theory to represent cooperative (and competitive) situations in a distributed computing notation used in an automatic theorem proving system

AFOSR MURI Automated Verification Accomplishments: 2007-08

1. Verified canonical problems studied in controls papers
    1. Groups of mobile robots
    2. Analog systems that converge to averages
2. Mechanically checked proofs (PVS) of these algorithms
3. Developed theorem to reduce state space of algorithms for control problems to enable use of model checkers.
4. Increased productivity of verification by reuse of mechanically verified theorems and algorithms.
5. Organizing an “open” course around theorem reuse.

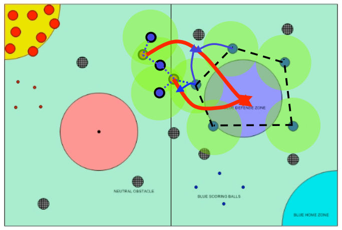
Problem Scope (Aug 06)

Overall Goal:

Develop methods and tools for designing control policies, specifying the properties of the resulting distributed embedded system and the physical environment, and proving that the specifications are met

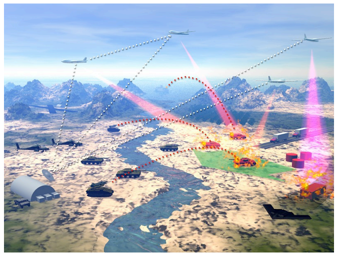
Specification

- How does the user specify---in a single formalism---continuous and discrete control policies, communications protocols and environment models (including faults)?



Design and reasoning

- How can engineers reason that their designs satisfy the specifications?
- In particular, can engineers reason about the performance of computations and communication, and incorporate real-time constraints, dynamics, and uncertainty into that reasoning?



Implementation

- What are the best ways of mapping detailed designs to hardware artifacts, running on specific operating systems? What languages are suitable for specifying systems so that the specifications can be verified more easily?

Students and Postdoctoral Fellows

1. Concetta Pilotto (PhD); graduating 2008 – 09
2. Jerome White (PhD); graduating 2008 - 09
3. Annie Liu (PhD); graduating 2010 -11
4. Brian Go (BS); graduating 2009

Gerard Holzmann's PhD students

1. Cheng Hu
2. Mihai Florian

Postdoctoral fellow: Sayan Mitra, now Asst Prof. UIUC

Papers: page 1

1. · Towards Verified Distributed Software through Refinement of Formal Archetypes: Chandy, Go, Mitra, White. Verified Software: Theories, Tools and Experiments (VSTTE 2008), Toronto
2. Convergence Verification: From Shared Memory to Partially Synchronous Systems K. Mani Chandy, Sayan Mitra and Concetta Pilotto; 6th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS 08), St. Malo, France, September 2008
3. · A Formalized Theory for Verifying Convergence and Stability of Automata in PVS Sayan Mitra and K. Mani Chandy; 21st International Conference on Theorem Proving in Higher Order Logics, (TPHOLS 2008), Montreal, 18-21 August, 2008

Papers: page 2

1. · Networked Sensing Systems for Detecting People Carrying Radioactive Material K. Mani Chandy, Concetta Pilotto and Ryan McLean; Fifth International IEEE Conference on Networked Sensing Systems (INSS 2008), June 17 - 19, 2008, Kanazawa, Japan
2. Towards a Theory of Events K. Mani Chandy, M. Charpentier, A. Capponi; Distributed Event Based Systems (DEBS 07) Conference; 2007
3. Periodically Controlled Hybrid Systems: Verifying a Controller for an Autonomous Vehicle T. Wongpiromsarn, S. Mitra, R. M. Murray, A.Lamperski, Hybrid Systems Computation and Control (submitted)

Bridge Differences in CS & Control Theory

- Control theory is based on differential equations.
- Distributed computing is based on discrete state transitions.

- Control theory is based on convergence
- Distributed computing is based on termination.

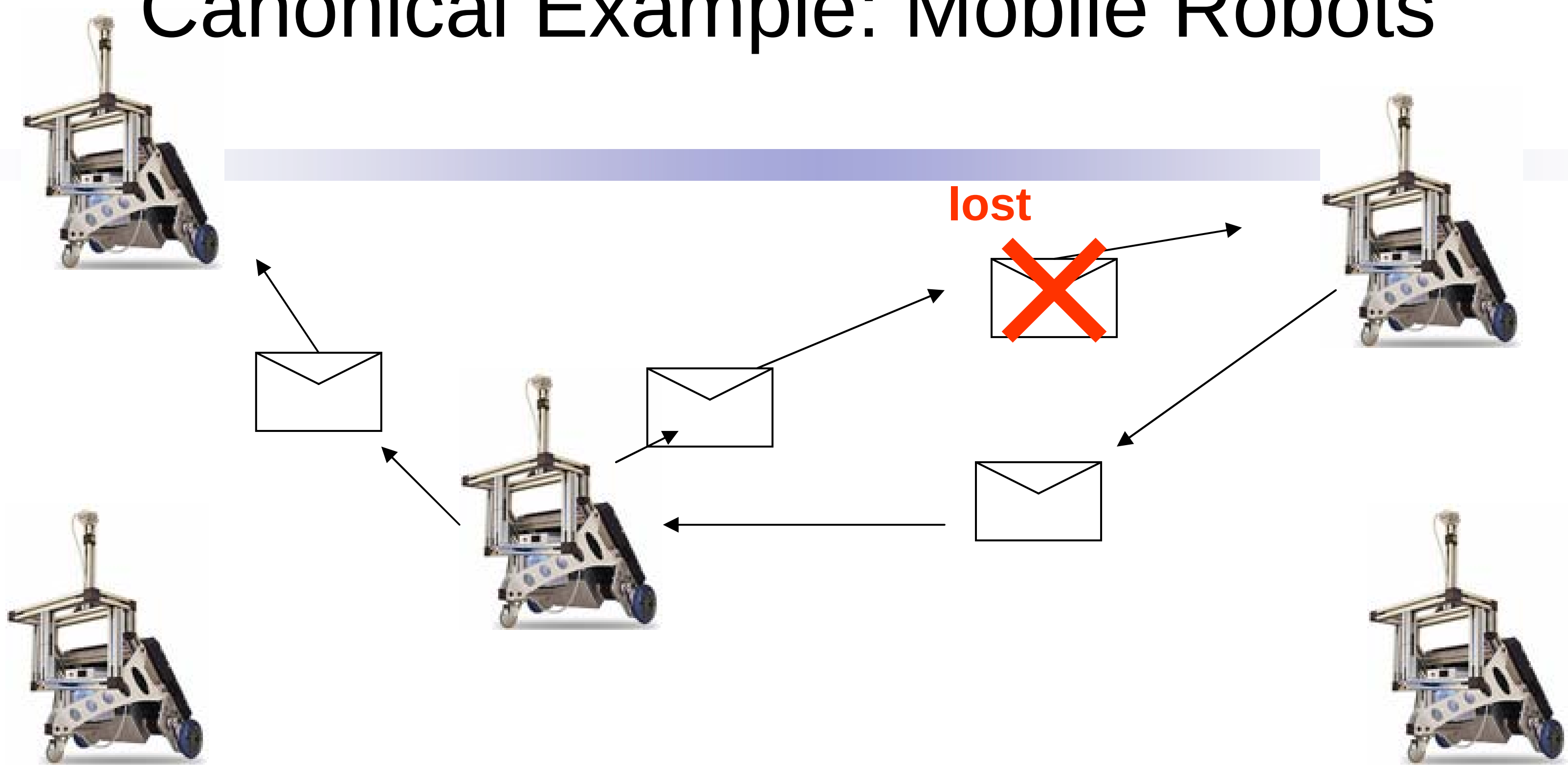
- In controls agents operate in actual time
- Distributed computing often deals with “eventuality”

- Controls proofs are “checked” using Matlab, Mathematica, …
- Distributed computing proofs are checked with theorem provers and model checkers

Canonical Problems at the Intersection of Distributed Controls and Computing

- Multi-agent systems in which agents communicate using messages through a faulty network.
- Agents operate in continuous state spaces.
- **Examples**: Groups of mobile robots; sensor networks; intrusion detection systems; distributed asynchrononous games

Canonical Example: Mobile Robots

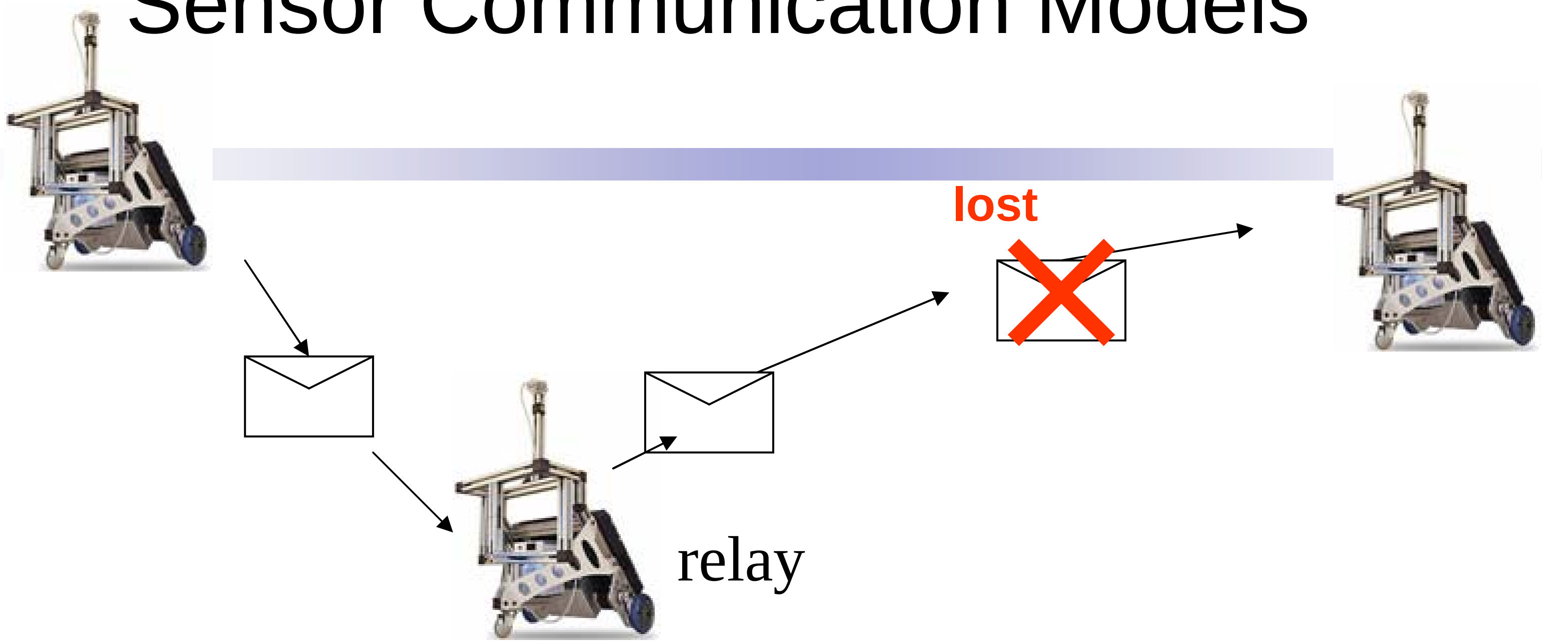


• Robots communicate by messages that may be lost & delayed.

• Each robot moves to the midpoint of the locations in the messages it last received from each neighbor.

• Will the robots eventually form an equi-spaced straight line?

Sensor Communication Models



- Each agent repeatedly sends messages containing its current state.
- Agents may relay messages received to other agents.
- Messages may get lost.

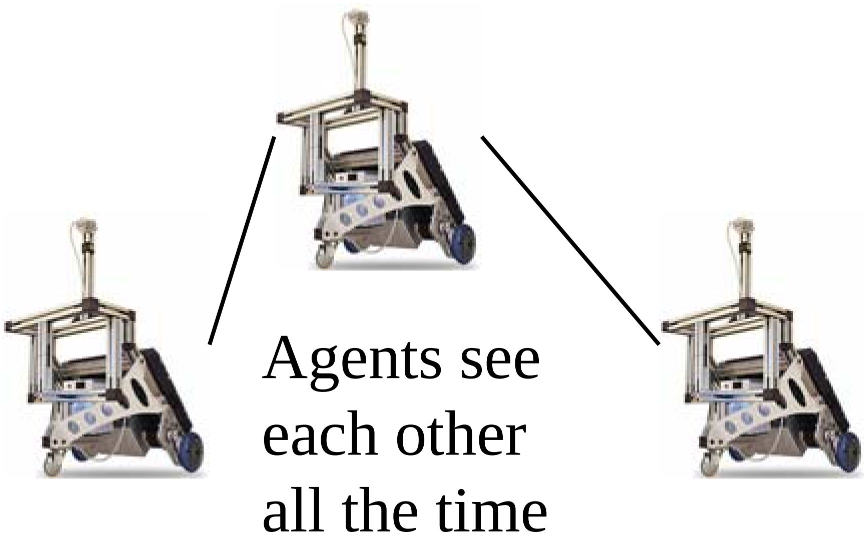
Control Theory Dynamics

1. $$\forall j \text{ where } 0 < j < N :$$

$$\frac{dX[j]}{dt} = \frac{(X[j-1] - X[j+1])}{2}$$

2. $$X[0] = C$$

3. $$X[N] = K$$



Control Theory Approaches

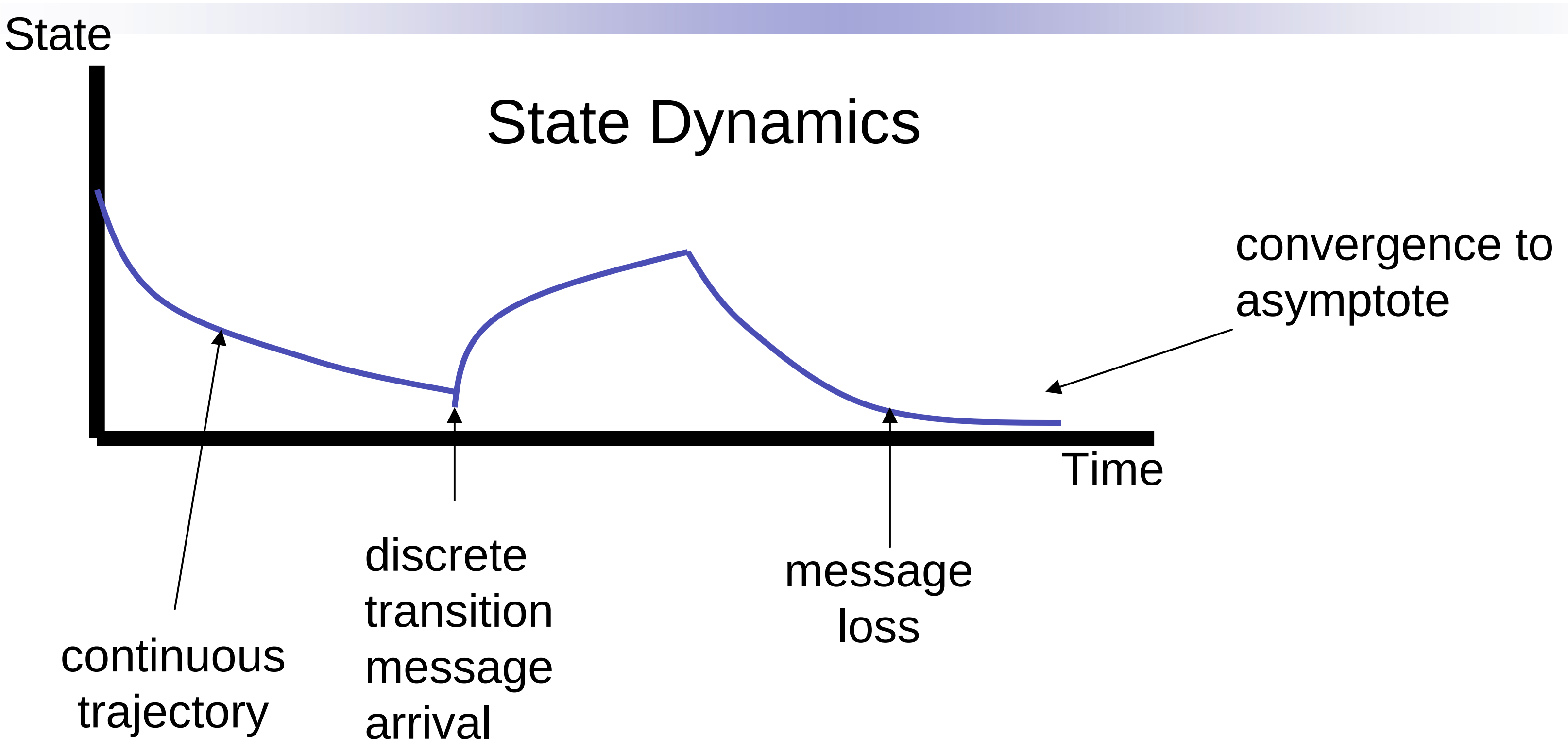
Solve $$\frac{dX}{dt} = A.X$$

More complex equations with feedback.

Use analysis of Eigen values; Bode plots; Lyapunov functions.

But these approaches don’t work with discrete lossy messages

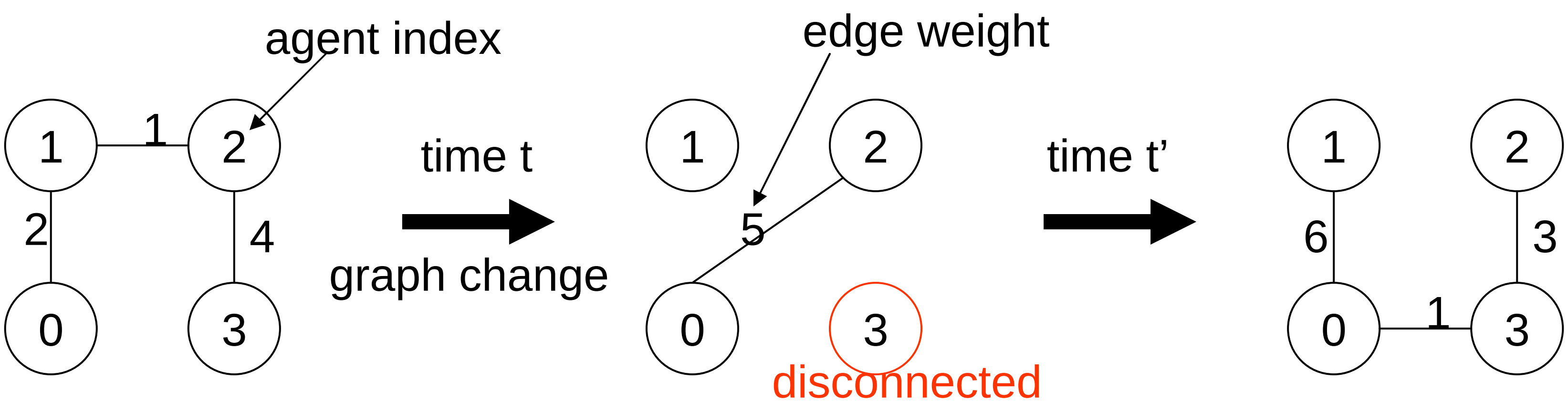
Convergence



Typical question: Will the system state converge? Or terminate? Or ..?

Another Canonical Example from Controls: Networked Multi-Agent Systems

Olfati-Saber and Murray, IEEE Proc. 2007 give results of what happens in this case



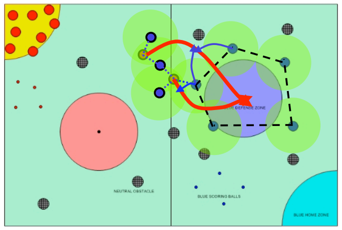
State trajectory $$\frac{dX[j]}{dt} = \sum_{k \neq j} W[j,k].(X[k] - X[j])$$

Problem Scope (Aug 06)

Overall Goal:

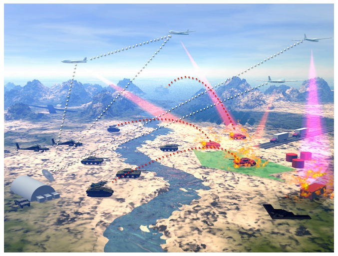
Develop methods and tools for designing control policies, specifying the properties of the resulting distributed embedded system and the physical environment, and proving that the specifications are met

Specification



- How does the user specify---in a single formalism---continuous and discrete control policies, communications protocols and environment models (including faults)?

Design and reasoning

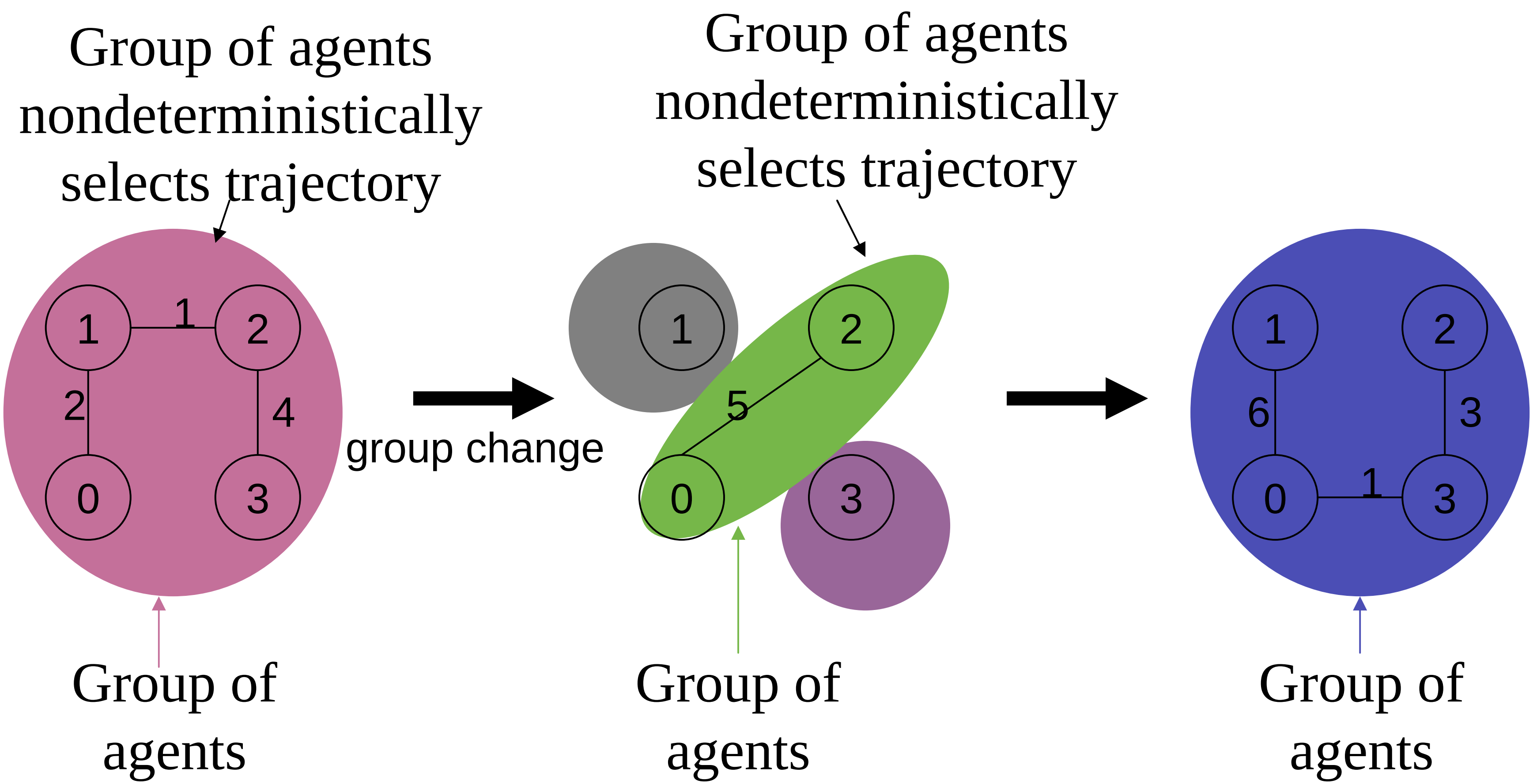


- How can engineers reason that their designs satisfy the specifications?
- In particular, can engineers reason about the performance of computations and communication, and incorporate real-time constraints, dynamics, and uncertainty into that reasoning?

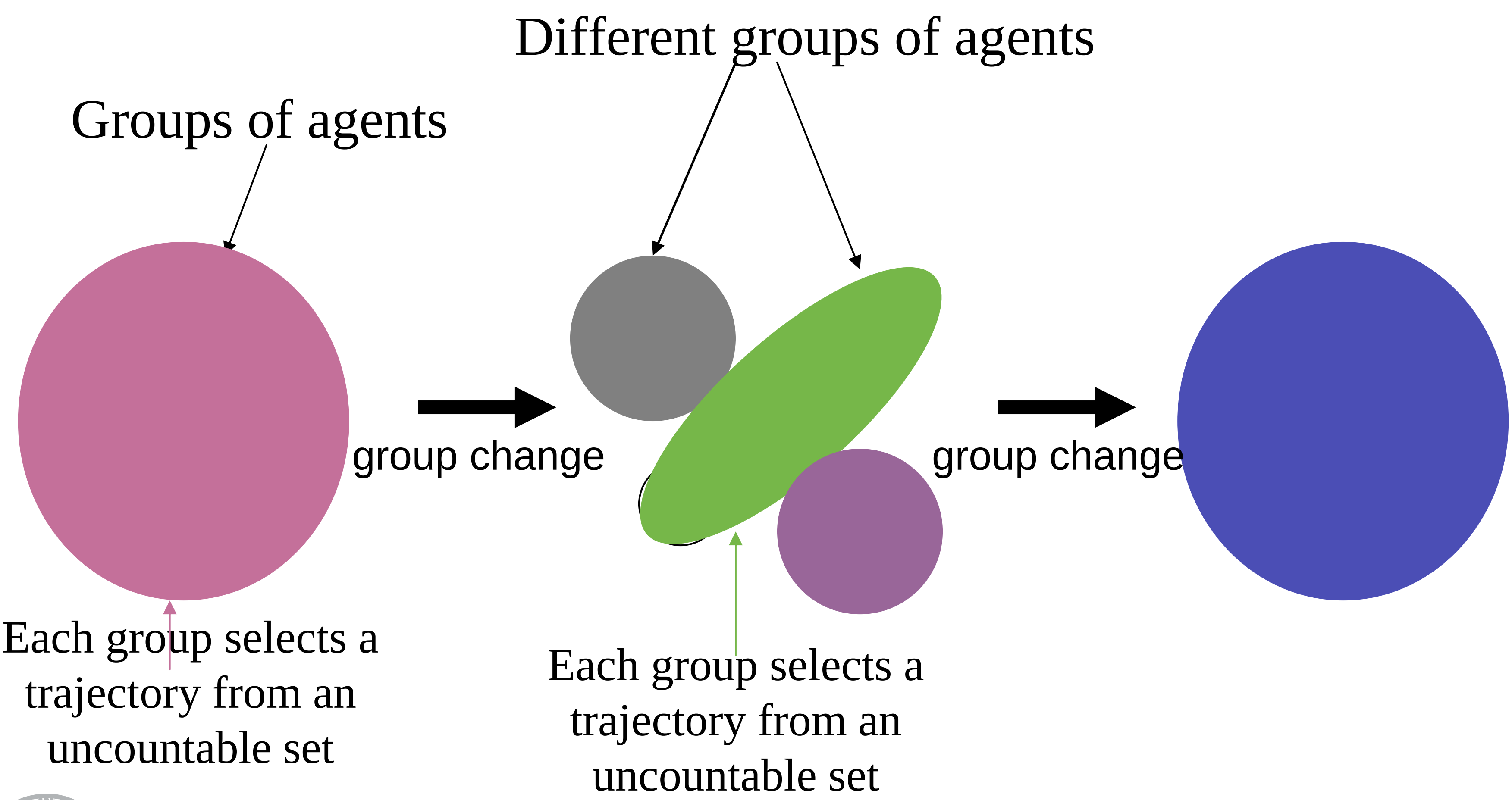
Implementation

- What are the best ways of mapping detailed designs to hardware artifacts, running on specific operating systems? What languages are suitable for specifying systems so that the specifications can be verified more easily?

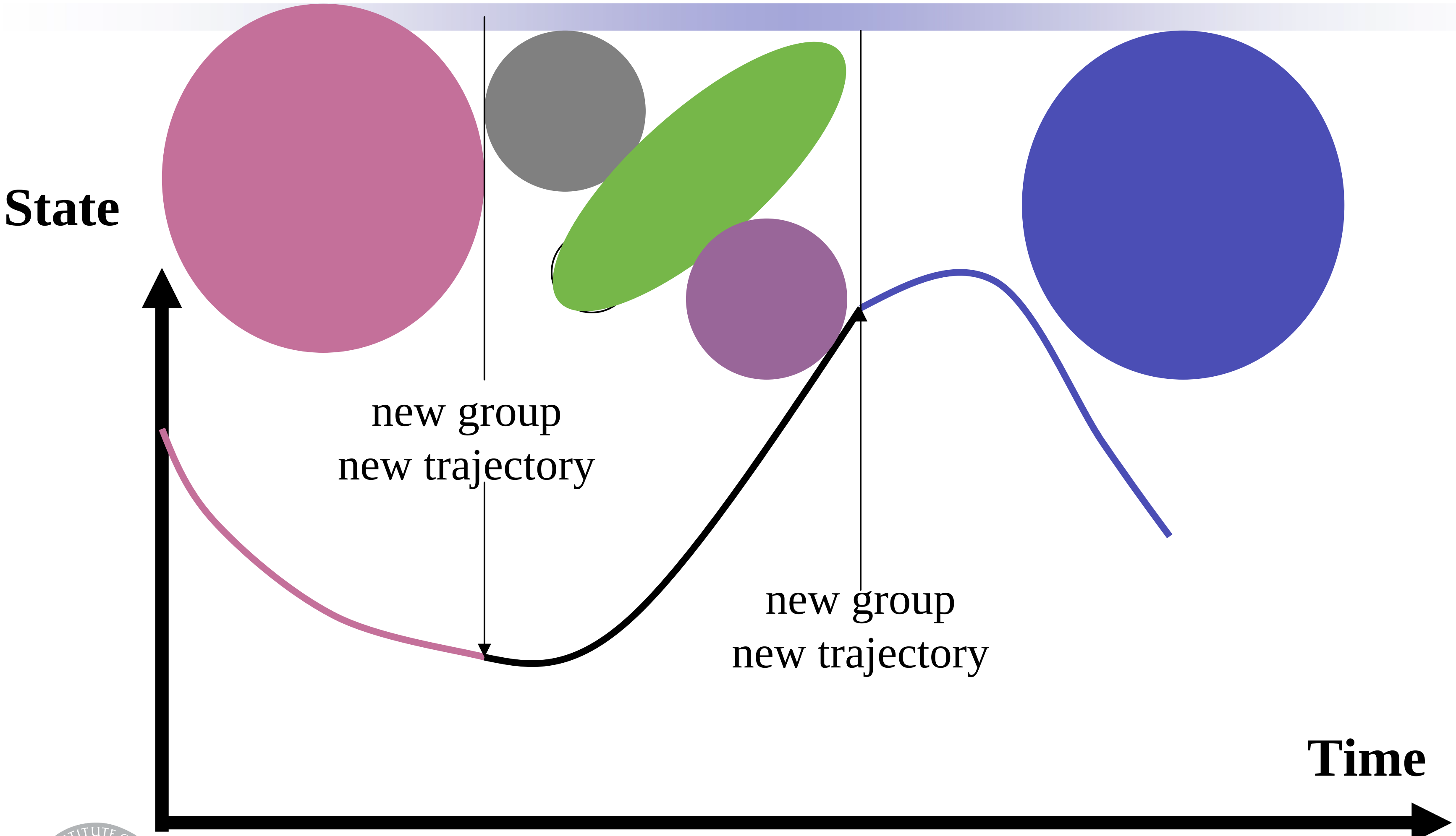
CS View and PVS Proofs of Control Results on Networked Multi-Agent Systems



CS View and PVS Proofs of Control Results on Networked Multi-Agent Systems



System Trajectory



Example of Theorem Prover

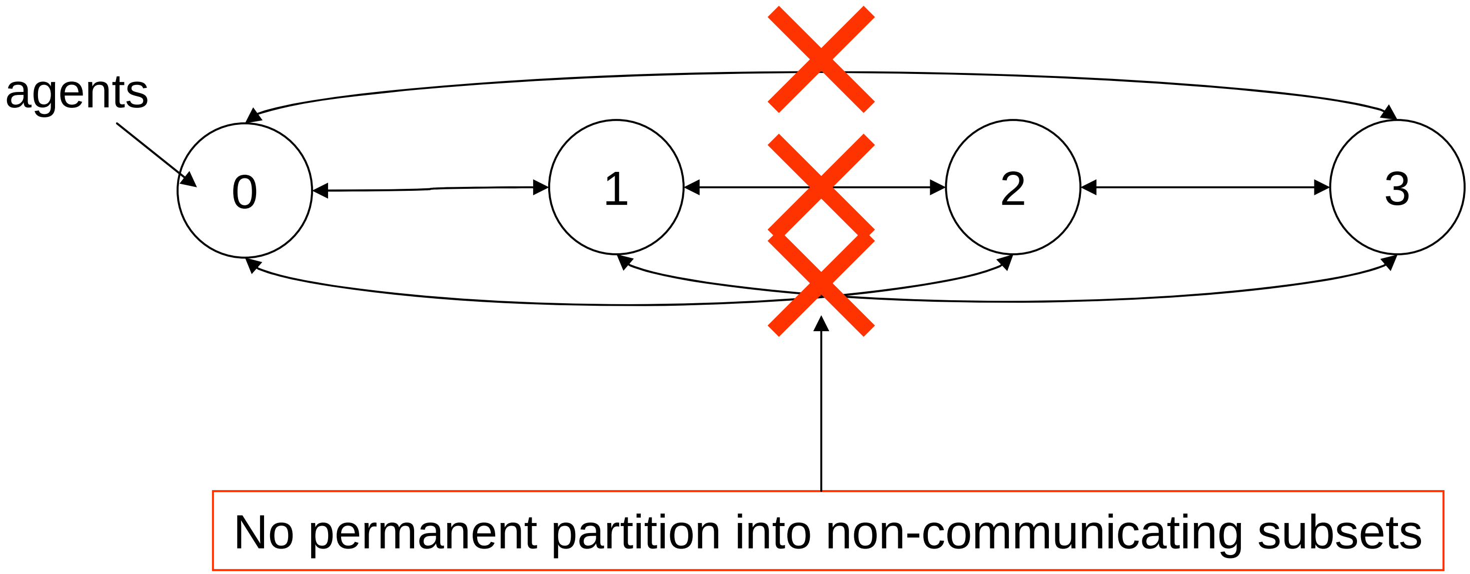
Given problem: Converge to the average of initial values

constraint $$\sum_j X[j] = K$$

Generalization: $$\circ f_j(X[j]) = K$$

Where the operator $\circ$ is associative and commutative.

Example Motivating Fairness



F = { {0,2}, {0, 3}, {1, 2}, {1, 3} }

Conservation Laws in PVS

Example: Let set of all agents be partitioned into subsets

$J_1, J_2, \ldots, J_M$

If each group of agents conserves $\circ X[j]$ then so does the entire system.

$$\forall k \in 1, \ldots, M : (\circ_{j \in J_k} \; X'[j]) = (\circ_{j \in J_k} \; X'[j])$$

$$\Rightarrow$$

$$\circ_j \; X'[j] = \circ_j \; X[j]$$

Generic Progress Proofs in PVS

If $\circ$ is strictly monotone and no group increases $\circ_{j \in J_k} X[j]$

and at least one group decreases it, then the total decreases.

$$\forall k \in 1, \ldots, M : (\circ_{j \in J_k} \; X'[j]) \leq (\circ_{j \in J_k} \; X[j])$$

$$\wedge$$

$$\exists k \in 1, \ldots, M : (\circ_{j \in J_k} \; X'[j]) < (\circ_{j \in J_k} \; X[j])$$

$$\Rightarrow$$

$$(\circ_j \; X'[j]) \; < \; (\circ_j \; X[j])$$

Example: Converge to the Average

- Monotone non increasing $\sum_j X[j]^2$
- Conserve $\sum_j X[j]$
- Proofs added to PVS library by Jerome White and Brian Go

AFOSR MURI Automated Verification Accomplishments: 2007-08

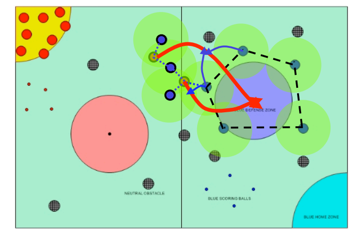
1. Verified canonical problems studied in controls papers
    1. Groups of mobile robots
    2. Analog systems that converge to averages
2. Mechanically checked proofs (PVS) of these algorithms
3. <u>*Developed theorem to reduce state space of algorithms for control problems to enable use of model checkers.*</u>
4. Increased productivity of verification by reuse of mechanically verified theorems and algorithms.
5. Organizing an “open” course around theorem reuse.

Problem Scope (Aug 06)

Overall Goal:

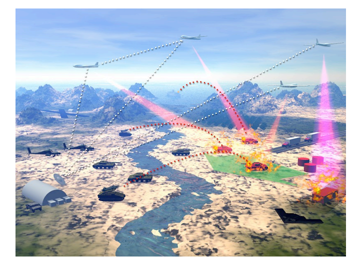
Develop methods and tools for designing control policies, specifying the properties of the resulting distributed embedded system and the physical environment, and proving that the specifications are met

Specification



- How does the user specify---in a single formalism---continuous and discrete control policies, communications protocols and environment models (including faults)?

Design and reasoning

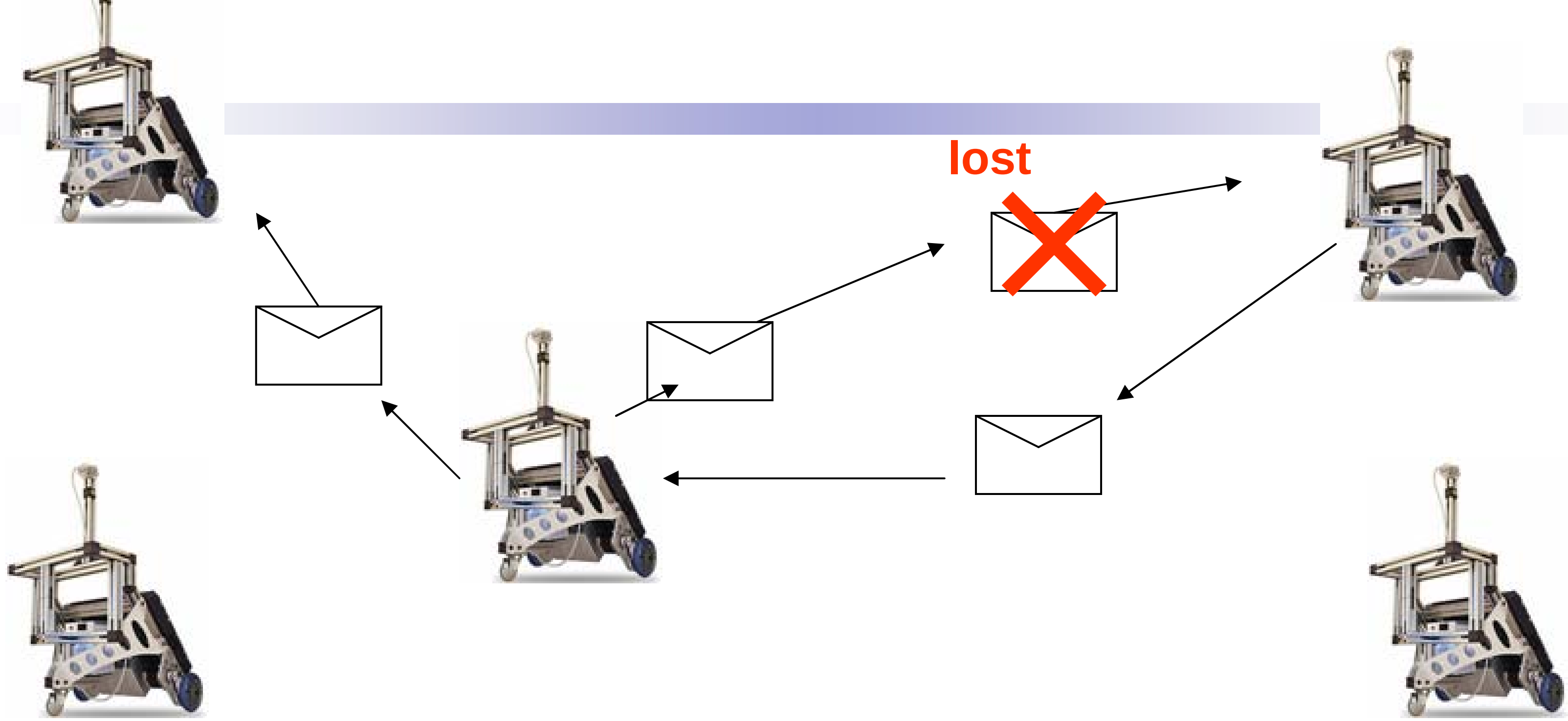


- How can engineers reason that their designs satisfy the specifications?
- In particular, can engineers reason about the performance of computations and communication, and incorporate real-time constraints, dynamics, and uncertainty into that reasoning?

Implementation

- What are the best ways of mapping detailed designs to hardware artifacts, running on specific operating systems? What languages are suitable for specifying systems so that the specifications can be verified more easily?

Wireless Communication Model

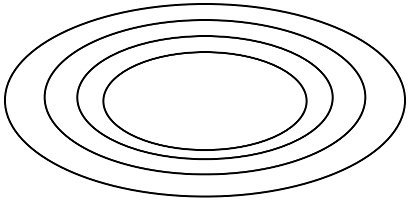


- Each agent sends its state (e.g., location) periodically
- Messages may be lost
- Messages may be relayed by intermediate agents.
- Delays unknown

Theorem

If there is a proof that a shared-state system converges then the wireless-communication system converges too provided the predicates in the proof are conjunctive.

Tsitiklis necessary and sufficient conditions for convergence



Telescoping sets

Condition: Sets defined by conjunction of agent-state predicates

Benefits of the theorem

- Enables model-checking to be used for distributed systems in which agents communicate using the wireless model and where agent state space is uncountable.

- Simplifies proofs and enables checking by automatic theorem prover.

AFOSR MURI Automated Verification Accomplishments: 2007-08

1. Verified canonical problems studied in controls papers
    1. Groups of mobile robots
    2. Analog systems that converge to averages
2. Mechanically checked proofs (PVS) of these algorithms
3. Developed theorem to reduce state space of algorithms for control problems to enable use of model checkers.
4. *Increased productivity of verification by reuse of mechanically verified theorems and algorithms.*
5. Organizing an “open” course around theorem reuse.

Ongoing Project: Reuse PVS Proofs

Dynamic Game Theory

↓ Refinement in PVS

Linear dynamic game theory

↓ Refinement in PVS

Mobile agents in shared state space

↓ Program transformation in PVS

Mobile agents with message passing in PVS

↓ **Hand transformation from PVS to robot code**

Mobile robots

Ongoing Project: Reuse PVS Proofs

Consensus

↓

Consensus: Associative, Commutative, Idempotent Operators
Shared State

↓ Program transformation

Consensus: Associative, Commutative, Idempotent Operators
Message Passing

↓

Consensus: Message-Passing *max*, *min*, *gcd*,… in PVS

↓ **Hand transformation from PVS to Java**

Agents coded in Java

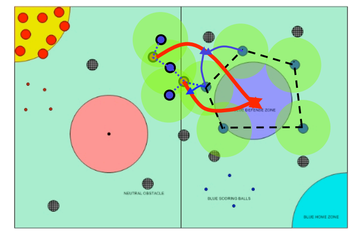
Problem Scope (Aug 06)

Overall Goal:

Develop methods and tools for designing control policies, specifying the properties of the resulting distributed embedded system and the physical environment, and proving that the specifications are met

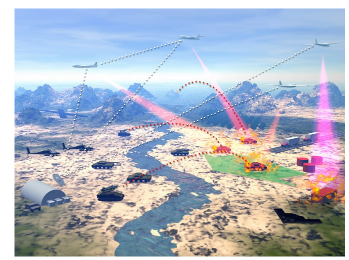
Specification

- How does the user specify---in a single formalism---continuous and discrete control policies, communications protocols and environment models (including faults)?



Design and reasoning

- How can engineers reason that their designs satisfy the specifications?
- In particular, can engineers reason about the performance of computations and communication, and incorporate real-time constraints, dynamics, and uncertainty into that reasoning?



Implementation

- What are the best ways of mapping detailed designs to hardware artifacts, running on specific operating systems? What languages are suitable for specifying systems so that the specifications can be verified more easily?

AFOSR MURI Automated Verification Accomplishments: 2007-08

1. Verified canonical problems studied in controls papers
    1. Groups of mobile robots
    2. Analog systems that converge to averages
2. Mechanically checked proofs (PVS) of these algorithms
3. Developed theorem to reduce state space of algorithms for control problems to enable use of model checkers.
4. Increased productivity of verification by reuse of mechanically verified theorems and algorithms.
5. <u>*Organizing an "open" course around theorem reuse.*</u>

Planned Experimental Course on V&V

- Experimental course on distributed computation
- The course emphasizes V&V using tools: automatic theorem proving systems (PVS) and model checking
- All course material will be available on a public courseware site (Moodle) except for class discussions and homework questions.
- Course offered every term. No formal lectures. Weekly homework tutorial discussions.
- No prerequisites. Suitable for students in all disciplines.
- Proposed start: 3rd term of this academic year.

Accomplishments in Teaching

- Established sequence of 3 courses on verification taught by JPL Lab for Reliable Software:
- CS 116: Reasoning about program correctness
- CS 118: Logic model checking for formal software verification
- CS 119: Reliable software testing and monitoring.

AFOSR MURI Automated Verification Accomplishments: 2007-08

1. Verified canonical problems studied in controls papers
   1. Groups of mobile robots
   2. Analog systems that converge to averages
2. Mechanically checked proofs (PVS) of these algorithms
3. Developed theorem to reduce state space of algorithms for control problems to enable use of model checkers.
4. Increased productivity of verification by reuse of mechanically verified theorems and algorithms.
5. Organizing an “open” course around theorem reuse.

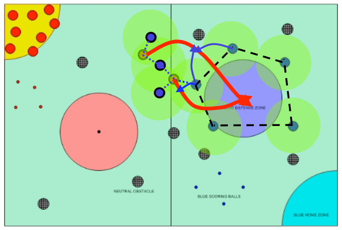
Problem Scope (Aug 06)

Overall Goal:

Develop methods and tools for designing control policies, specifying the properties of the resulting distributed embedded system and the physical environment, and proving that the specifications are met

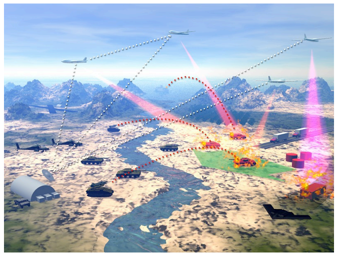
Specification

- How does the user specify---in a single formalism---continuous and discrete control policies, communications protocols and environment models (including faults)?



Design and reasoning

- How can engineers reason that their designs satisfy the specifications?
- In particular, can engineers reason about the performance of computations and communication, and incorporate real-time constraints, dynamics, and uncertainty into that reasoning?



Implementation

- What are the best ways of mapping detailed designs to hardware artifacts, running on specific operating systems? What languages are suitable for specifying systems so that the specifications can be verified more easily?