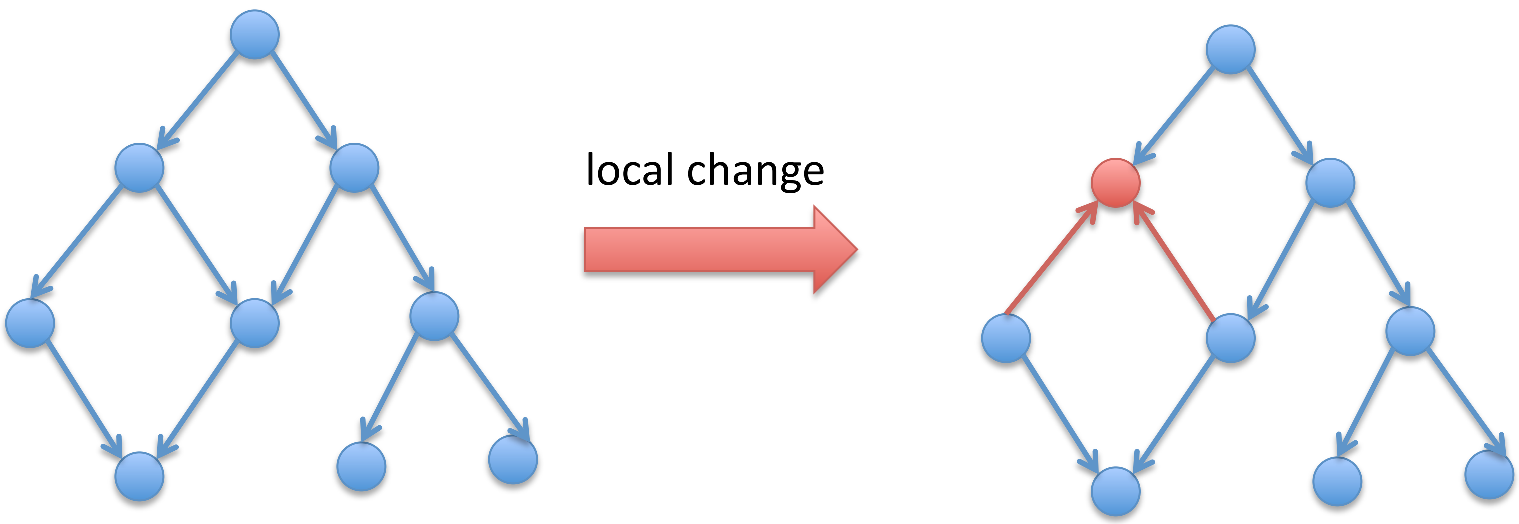
Verifying Distributed Systems: Examples, Tools, Limitations

Mani Chandy
California Institute of Technology

What's different about distributed systems?

- Local changes to global structures.

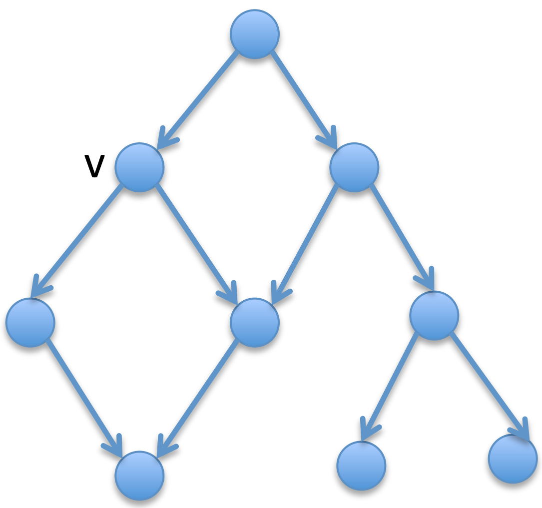


Global data structure: S

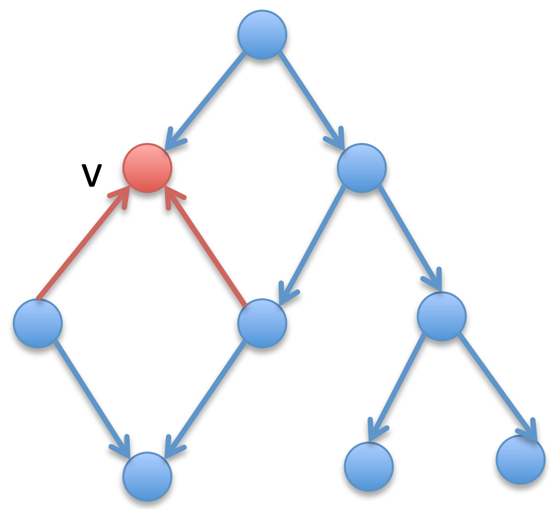
New global data structure: S'

Global properties of local transitions

Graph is acyclic $\xrightarrow{\text{global property}}$ Graph is acyclic



local change



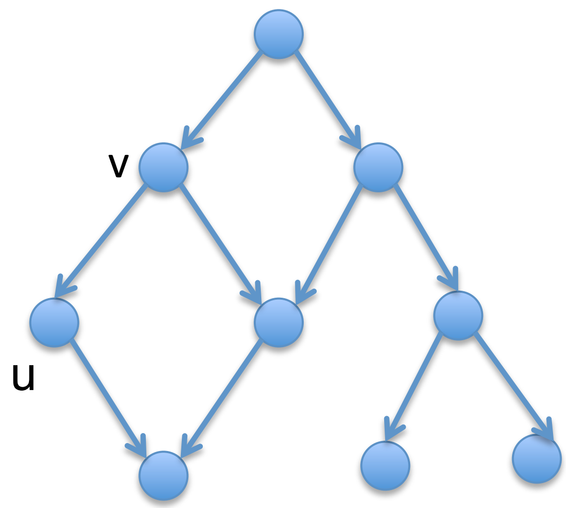
New global data structure: S’

Global properties of local transitions

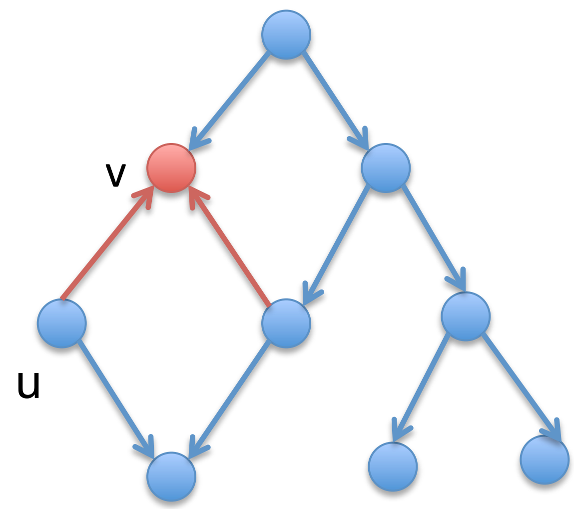
For vertexes other than v:
Number of higher
priority vertexes= k

global property

For vertexes other than v:
Number of higher
priority vertexes >= k



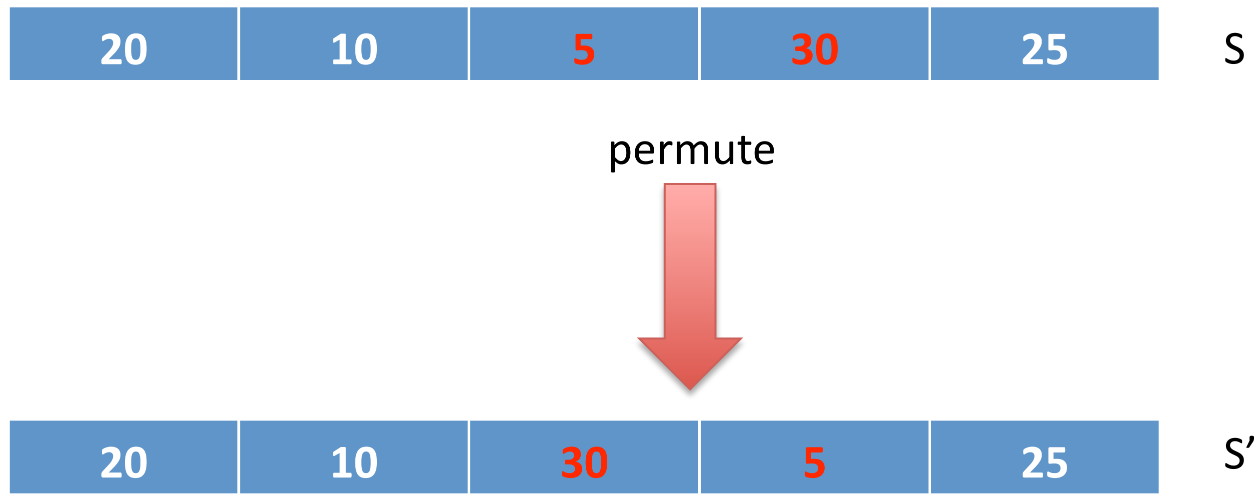
local change



Two vertexes with higher priority than u

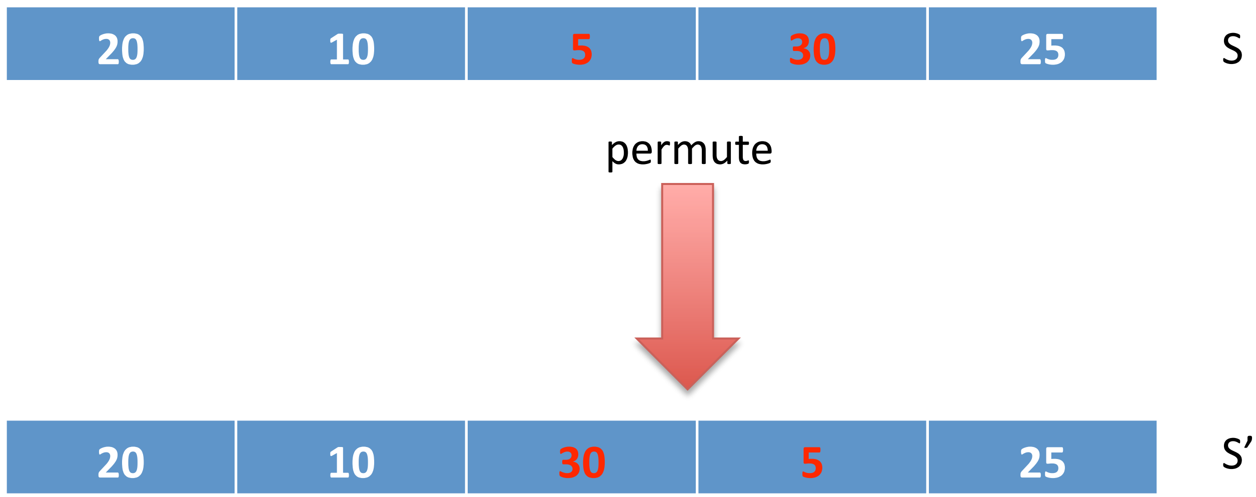
No vertex with higher priority than u

Local-Global Properties



Local transition is a permutation
IMPLIES
Global transition is a permutation

Local Global Properties



Number of out of order pairs decreases in a local transition

IMPLIES

Number of out of order pairs decreases in a global transition

4 pairs out of order before transition 3 pairs out of order after transition

Focus of our MURI research

- Data structures + algorithms = programs (Niklaus Wirth)

Focus of our MURI research

- Data structures +
  algorithms =
  programs

- Distributed data structures +
  distributed algorithms =
  distributed programs

Focus of our MURI research

- What is a distributed data structure?
- It is one that has local-global properties.

Focus of our MURI research

- What is a distributed data structure?

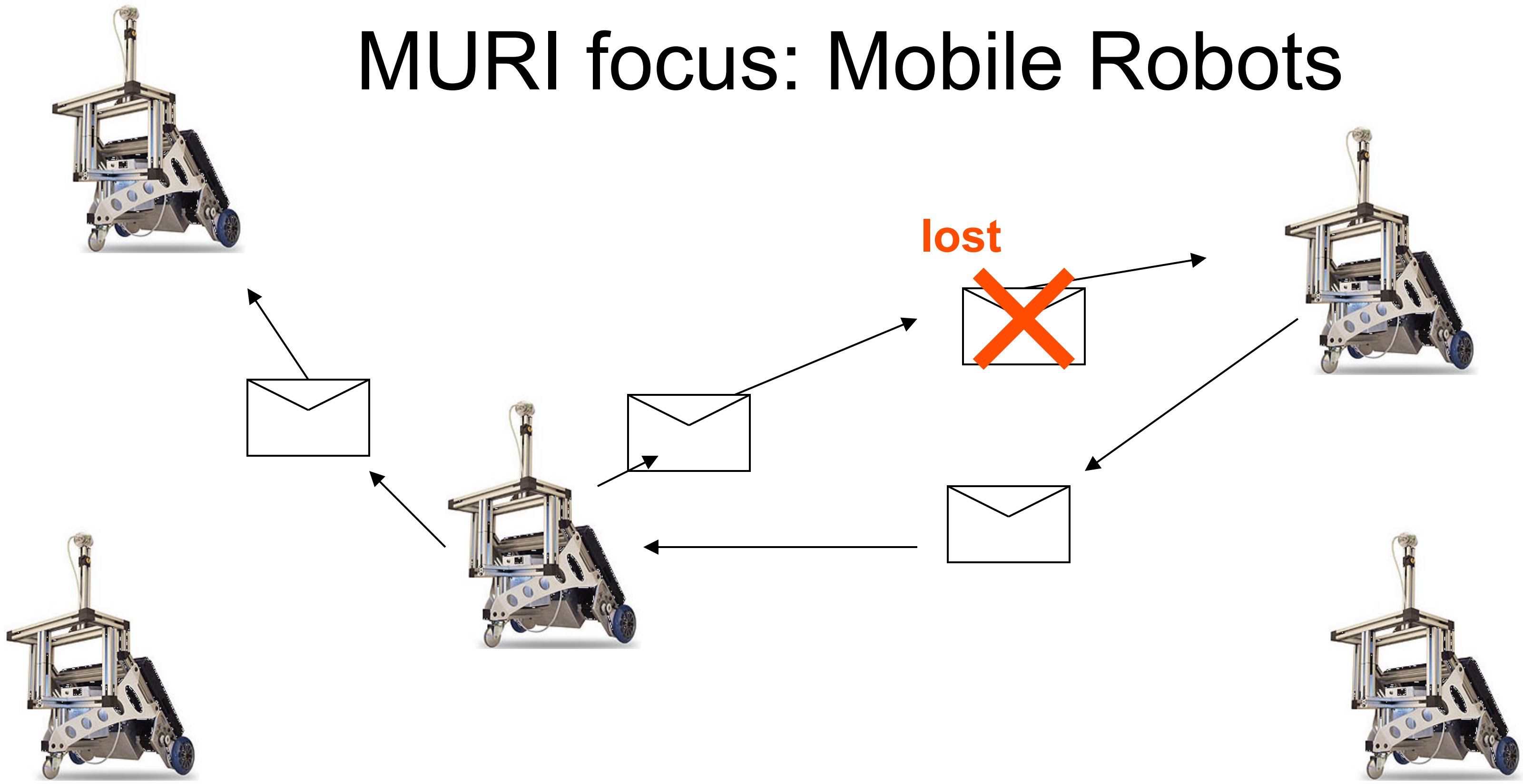
- It is one that has local-global properties.

- How do we formalize these properties?
- How do we verify them?
- How do we use them in practice?
- What lessons have we learned?

Focus of our MURI research

- Hybrid systems: continuous dynamics with discrete mode transitions.
- V&V of systems operating under adverse conditions:
  - Communication medium is faulty
  - Operating environment is faulty
  - Game theoretic models of adversaries

MURI focus: Mobile Robots



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

• Robots move into a specified formation (e.g., circle)

• Adversary controls communication medium.

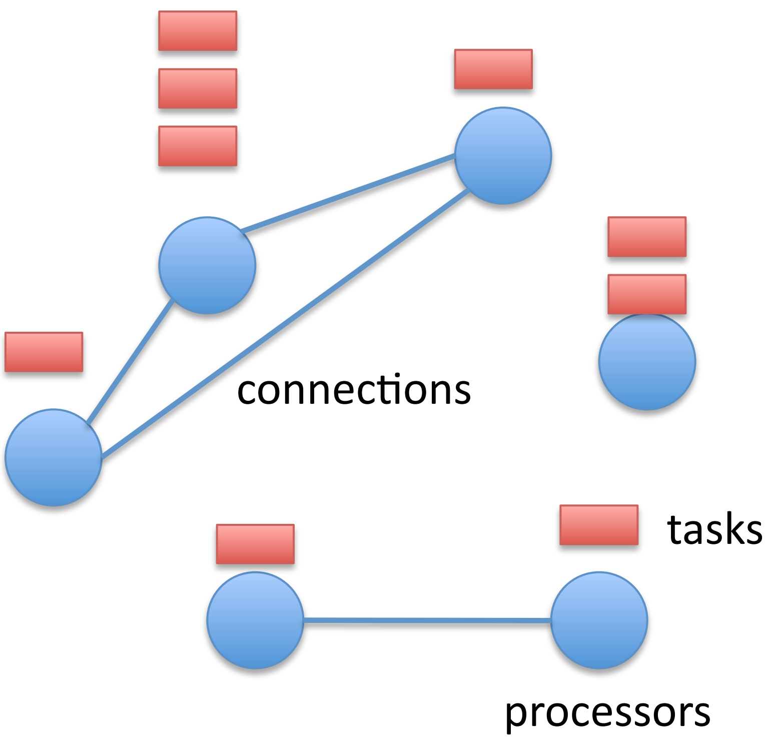
Formalizing Local-Global Concept

- Designers usually know the global properties that they want.
- Examples:
  - if s is an acyclic graph then s' is an acyclic graph.
  - The average of agents in s and s' is the same.
  - The sum of squares in s' is less than in s.

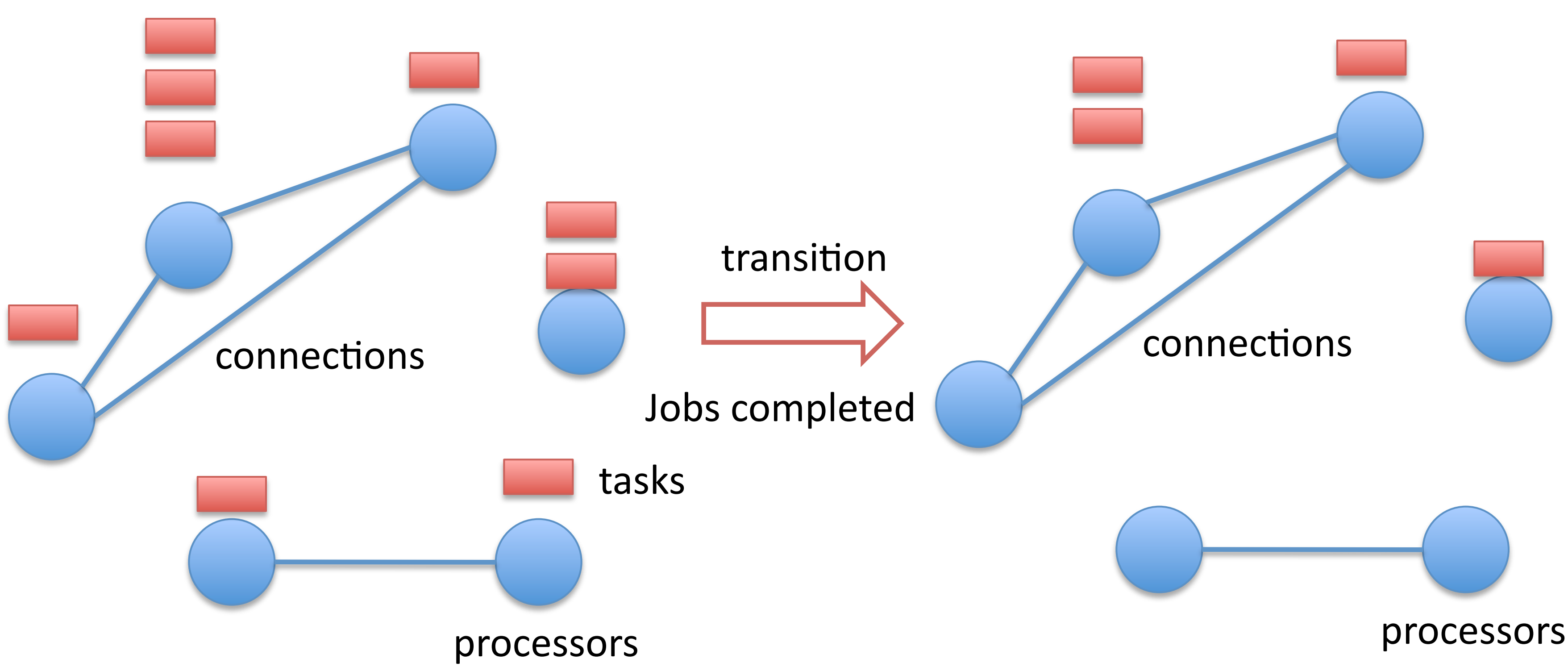
Formalizing Local-Global Concept

- Challenge: determine local actions that produce the desired global properties.

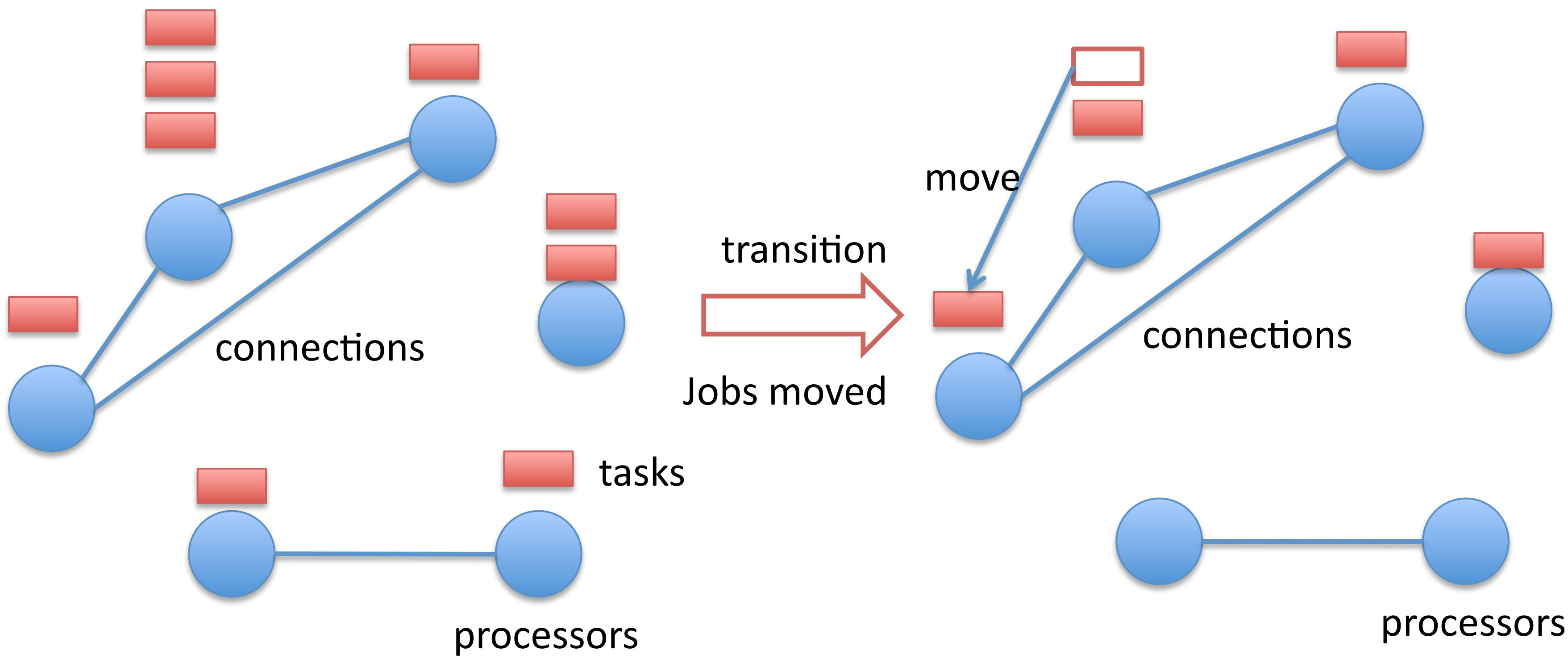
Running Example



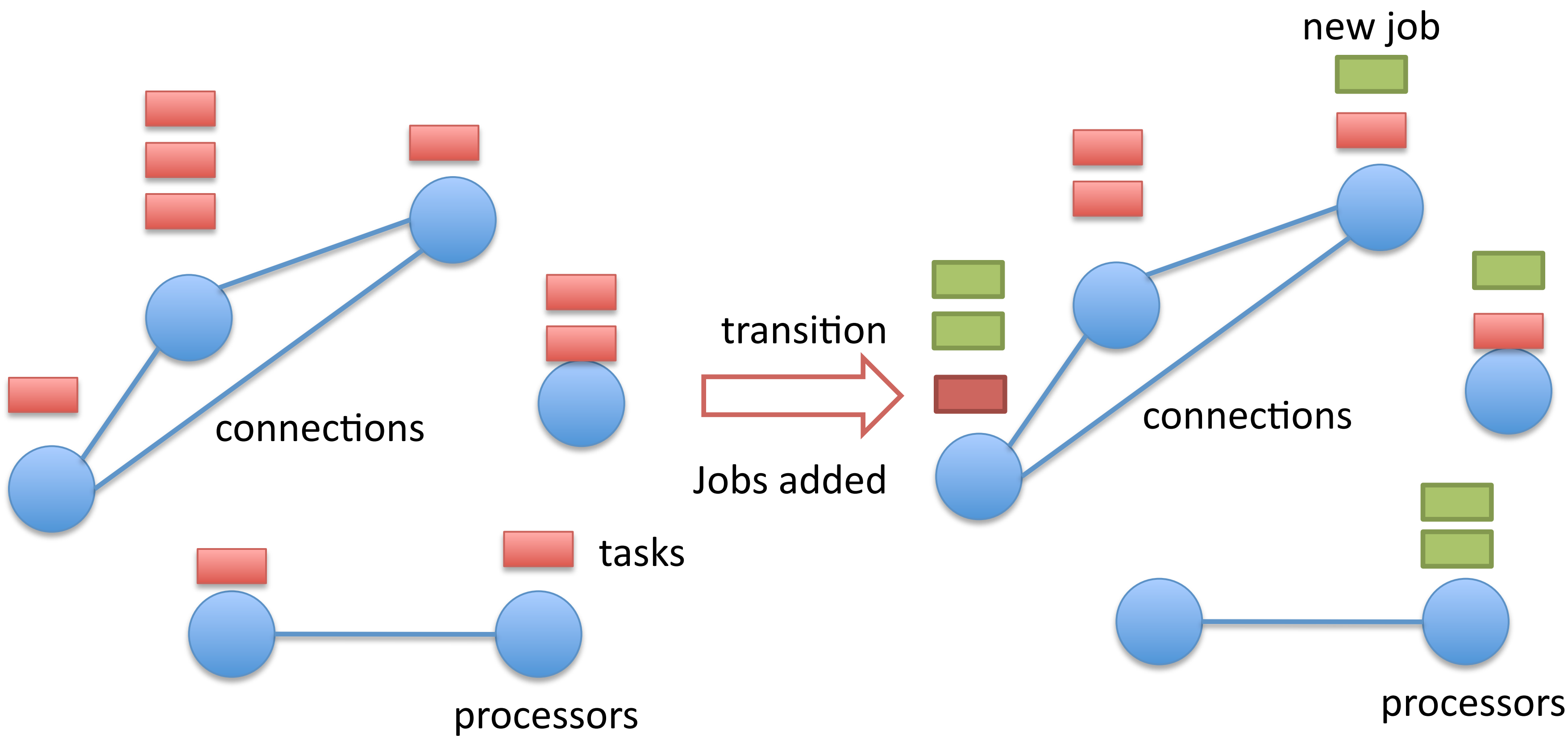
Running Example



Running Example



Running Example



MURI researchers and this problem

Richard Murray: Averaging

John Doyle & Vanessa Jonson: Dynamic load balancing

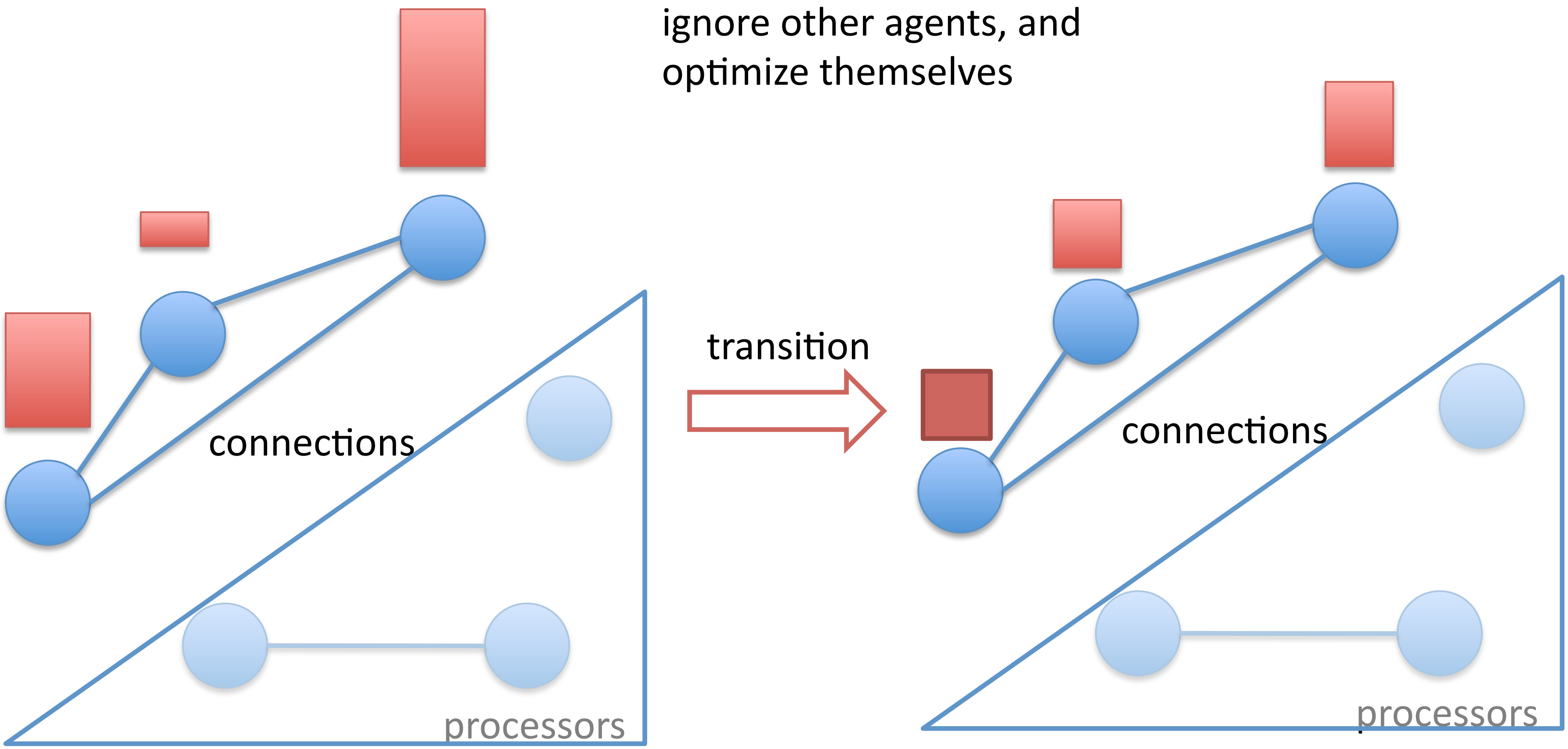
Erik Klavins: Movement of resources

Pablo Parillo: Sum of Squares

Chandy,Pilotto, White: Local-Global

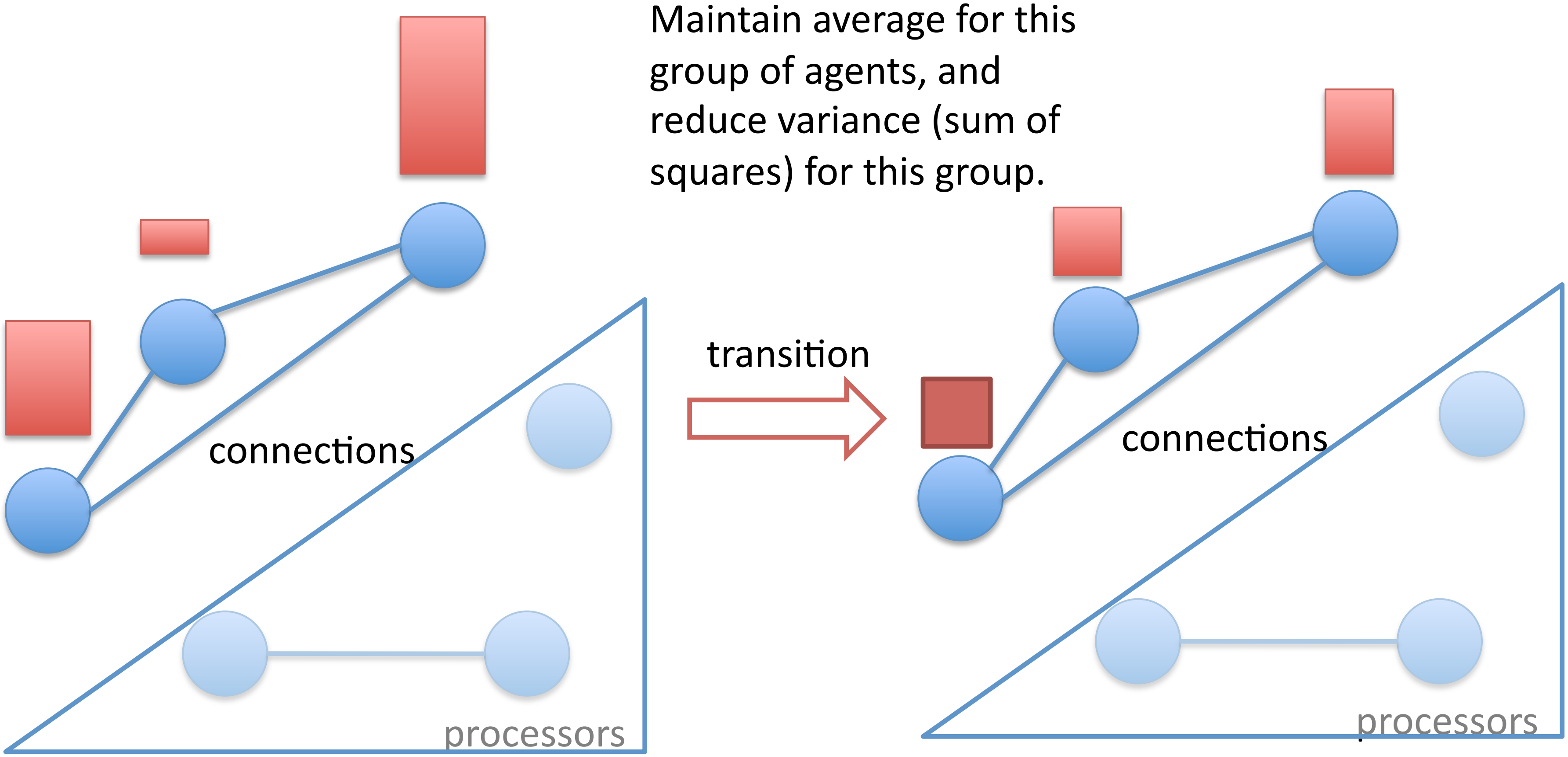
Running Example: Local-Global

Agents in an interaction ignore other agents, and optimize themselves

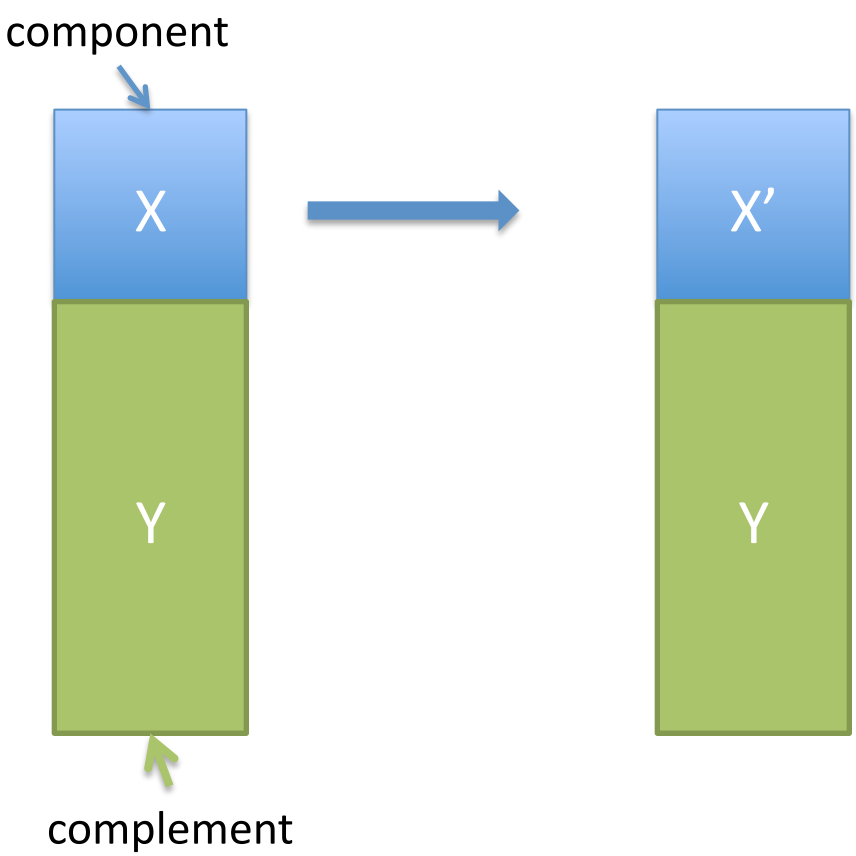


Running Example: Local-Global

Maintain average for this group of agents, and reduce variance (sum of squares) for this group.



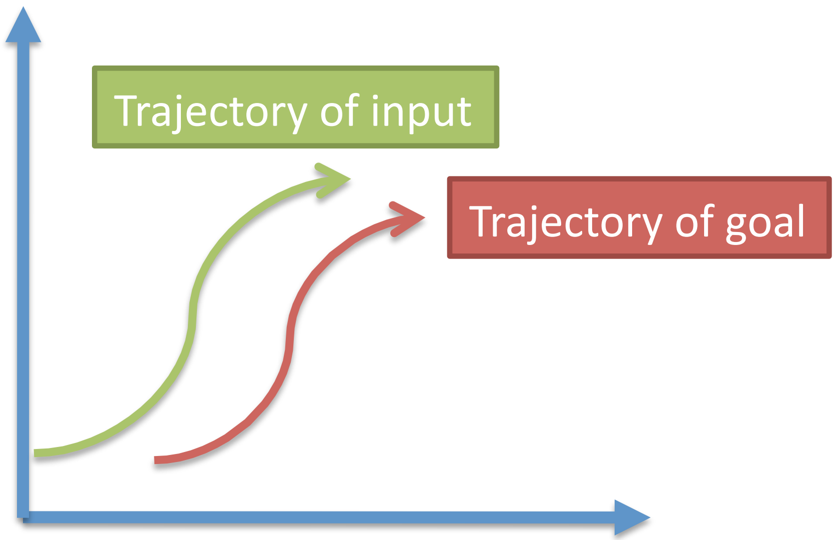
Local-Global Formalization



Local_relation(x, x')
IMPLIES
Global_relation(x#y, x'#y)

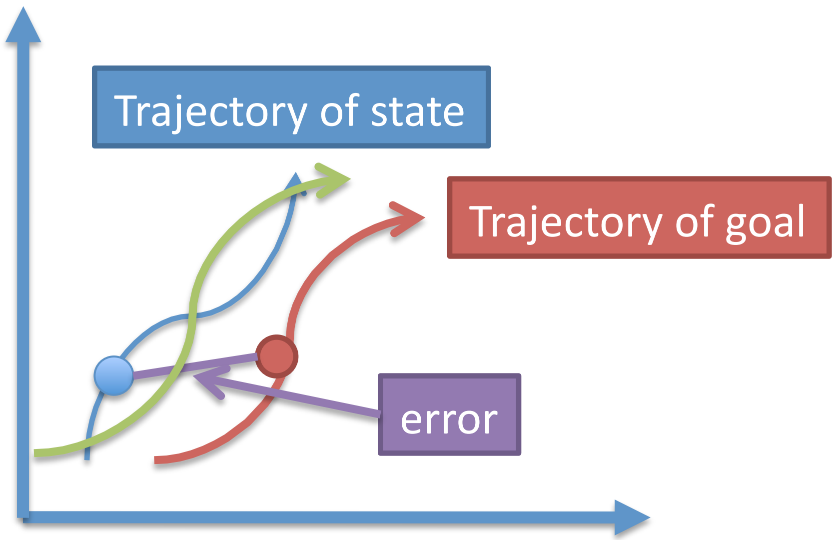
Example:
Average(x) = Average(x')
IMPLIES
Average(x#y) = Average(x'#y)

Running Example



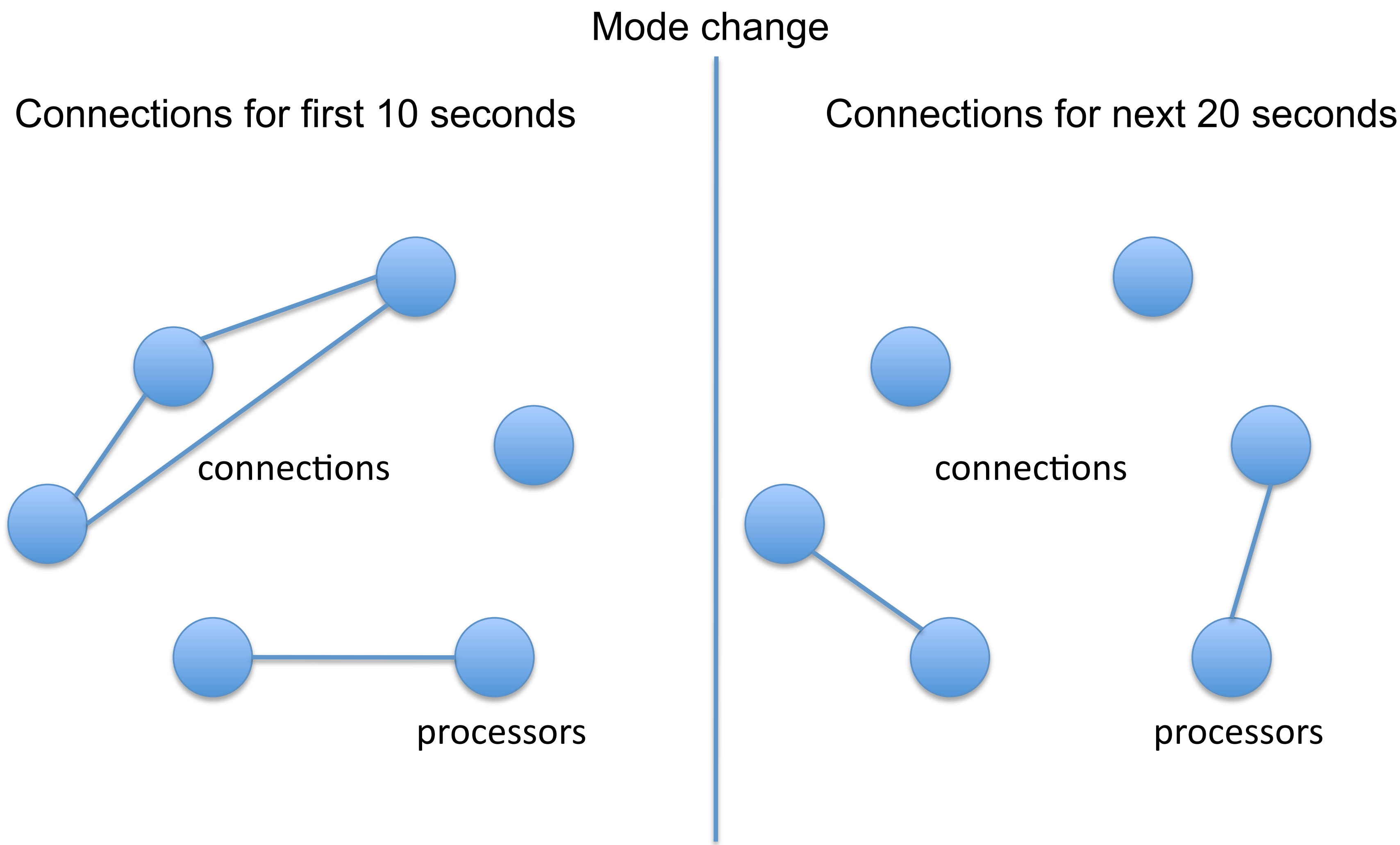
goal = f(input)

Running Example

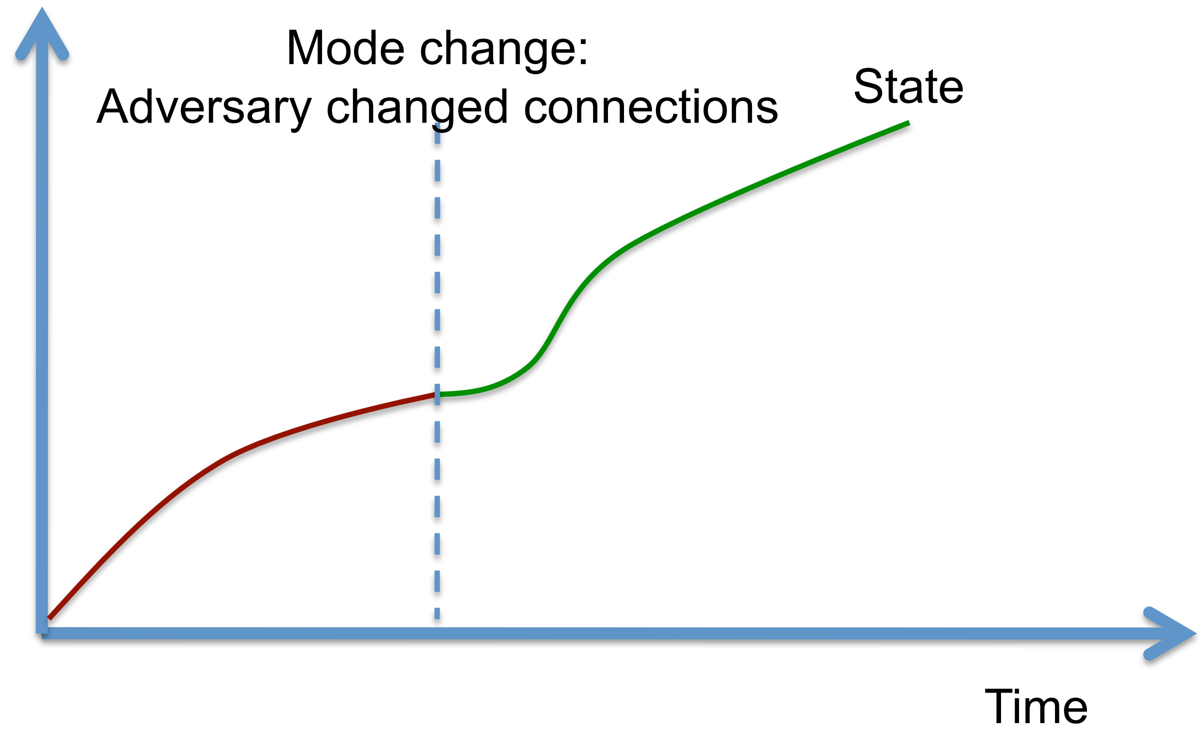


Under what conditions is the error bounded?

Running Example

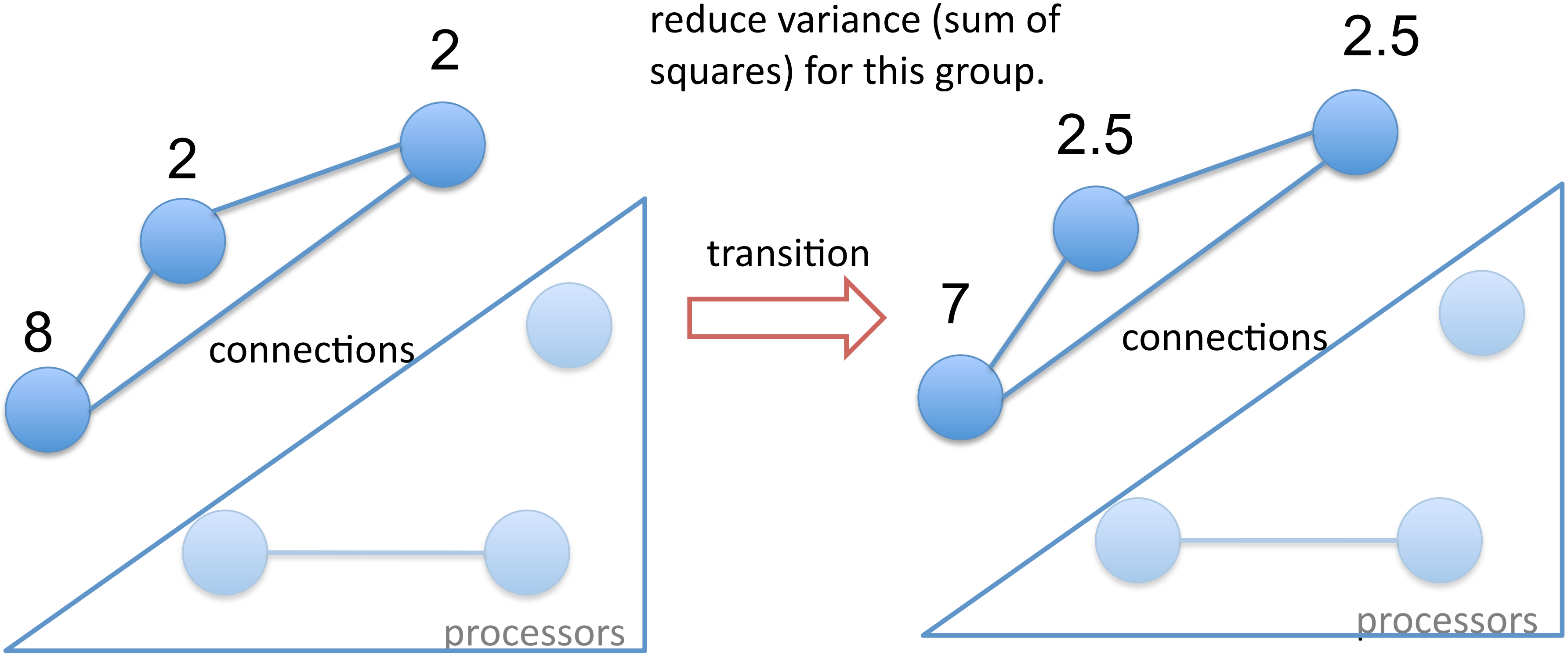


Running Example



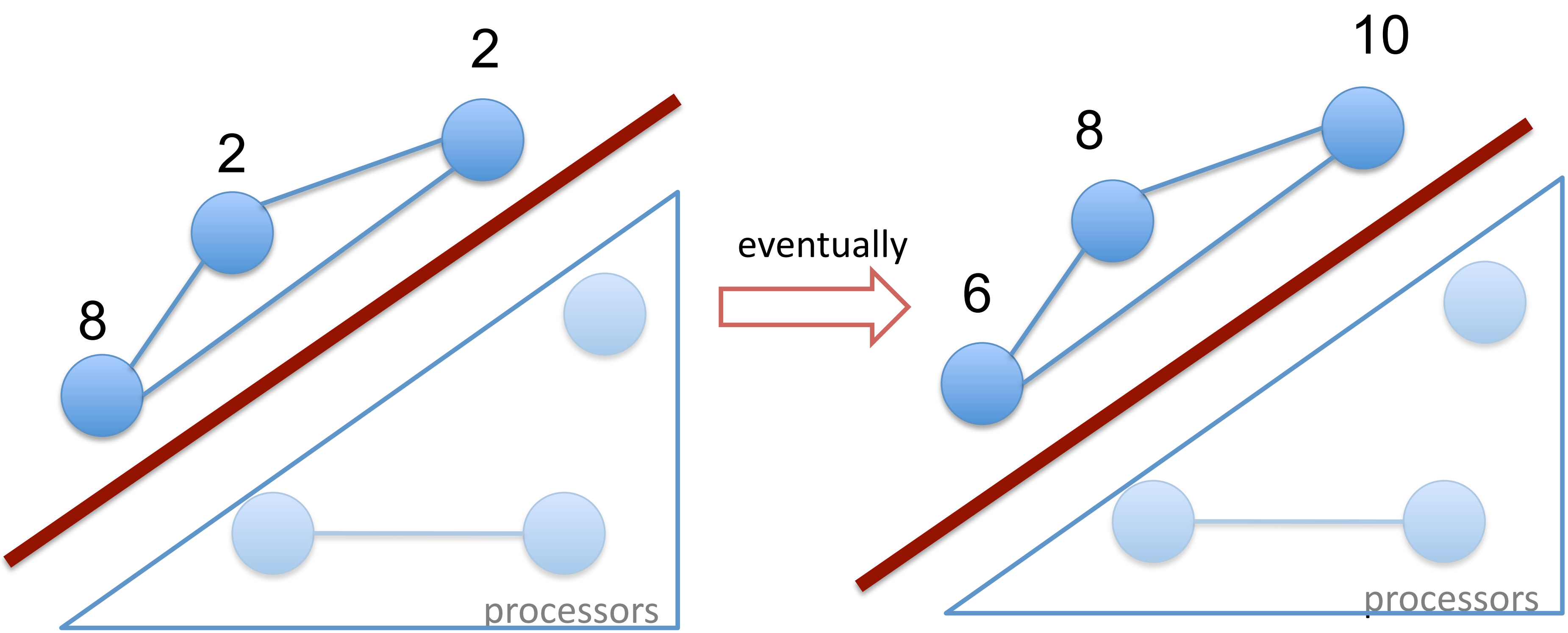
Local Operations & Global Properties

Maintain average for this group of agents, and reduce variance (sum of squares) for this group.



Results

If graph is permanently disconnected then error for each connected component C is bounded provided variance of the change in C in each epoch T is bounded



Our research in this MURI

1. Develop a library of local-global data structures and algorithms for continuous dynamics (convergence) and discrete systems.

- Examples:
  - trajectory following (e.g., average)
  - Movement of robots into formations
  - Fence polluted areas with smallest circle

Our research in this MURI

1. Prove properties of data structures and algorithms in a theorem prover (PVS). Develop a library of theorems. This verifies a collection of multi-agent programs written in PVS operating in adverse conditions.
   - multi-agent trajectory following
   - distributed solutions to systems of equations
   - multi-agent consensus

Our research in this MURI

2. Translate algorithms from PVS to Java, Erlang, C.
3. Develop a class library in Java.
4. Use tools such as static analyzers and model checkers such as SPIN.

Lesson learned:
Different tools for different purposes

- Theorem prover for abstract data types and algorithms. Verify algorithms for continuous state systems and continuous dynamics, and arbitrary numbers of agents.
- Type checking, debugging tools, model checking, code walkthrough tools, for target code.
- The combination results in V&V.

Playing devil's advocate

- Do mechanical proof checkers add value?
  - Yes, it makes you careful!
  - But, we hand-checked our proofs in painstaking detail before we used the mechanical verifier.
  - There were no surprises when we checked proofs with the mechanical proof checker.
  - Learning curve is substantial, but PVS notation is like math, and that helps
  - It takes a lot of effort.

Playing devil's advocate

- We need to make use of V&V tools efficient.
- How can we achieve efficiency?

Playing devil's advocate

- How can we achieve efficiency in tool usage?
- REUSE
  - Reuse data structures
  - Reuse classes of algorithms
  - Share libraries of theorems.
  - Repurpose theorems: change the problem a bit, and change the proof a bit (works sometimes).

Recommendations

- Use an arsenal of tools:
  - Mechanical proof checkers
  - Model checkers
  - Static analyzers and debugging tools
  - Design & code walkthrough tools
  - Emphasize reuse