Distributed Watchpoints: Debugging Very Large Ensembles of Robots

De Rosa, Goldstein, Lee, Campbell, Pillai

Aug 19, 2006

Motivation

- Distributed errors are hard to find with traditional debugging tools
- Centralized snapshot algorithms
  - Expensive
  - Geared towards detecting one error at a time
- Special-purpose debugging code is difficult to write, may itself contain errors

Expressing and Detecting Distributed Conditions

"How can we represent, detect, and trigger on distributed conditions in very large multi-robot systems?"

- Generic detection framework, well suited to debugging
- Detect conditions that are not observable via the local state of one robot
- Support algorithm-level debugging (not code/HW debugging)
- Trigger arbitrary actions when condition is met
- Asynchronous, bandwidth/CPU-limited systems

Distributed / Parallel Debugging: State of the Art

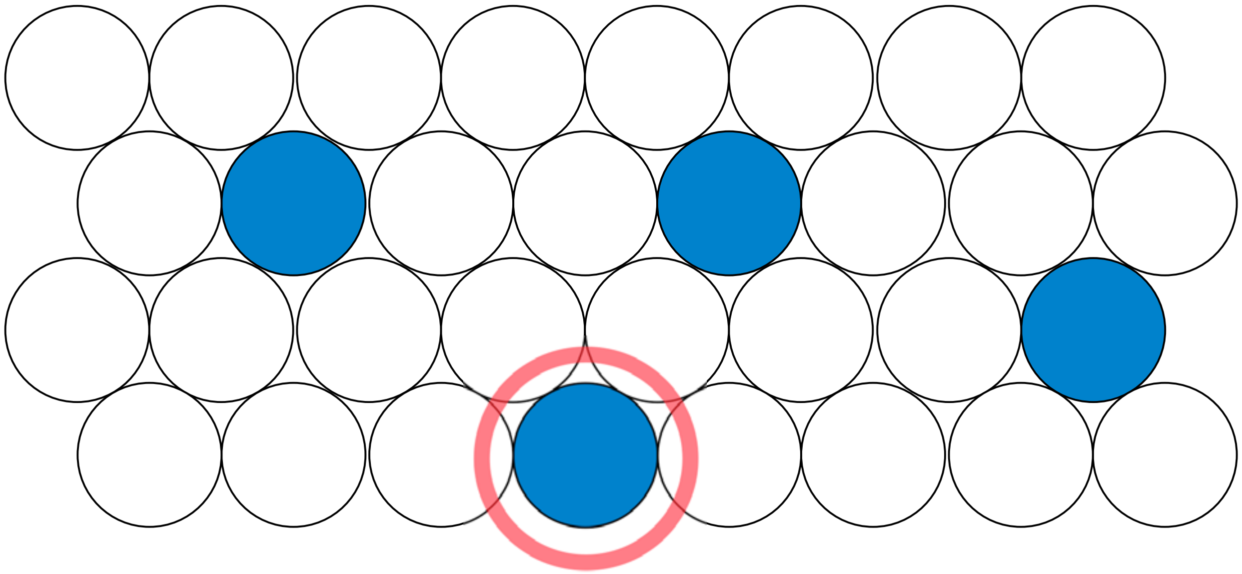
Modes:

- Parallel: powerful nodes, regular (static) topology, shared memory
- Distributed: weak, mobile nodes

Tools:

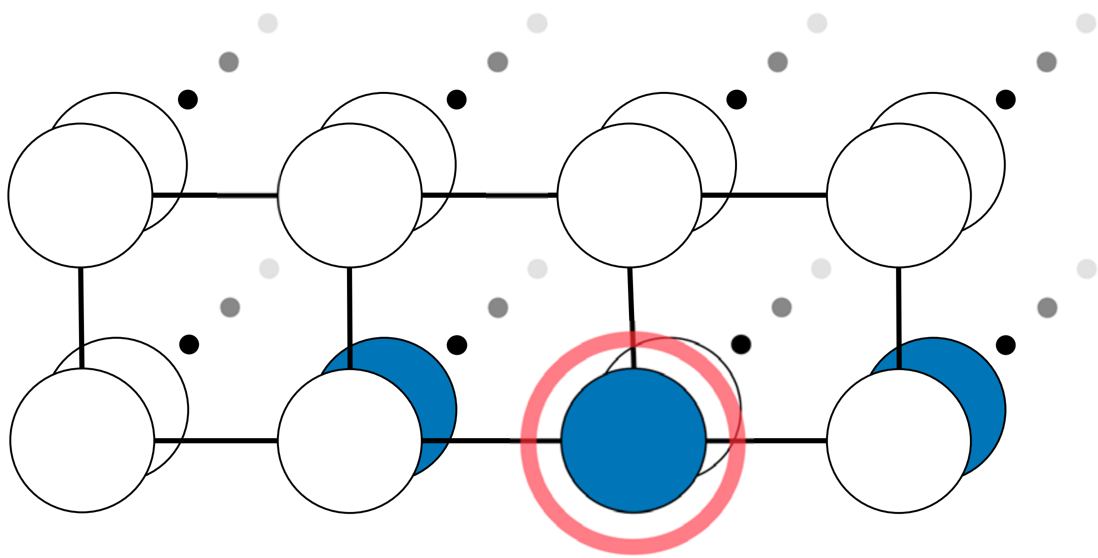
- GDB
- printf()
- Race detectors
- Declarative network systems with debugging support (ala P2)

Example Errors: Leader Election



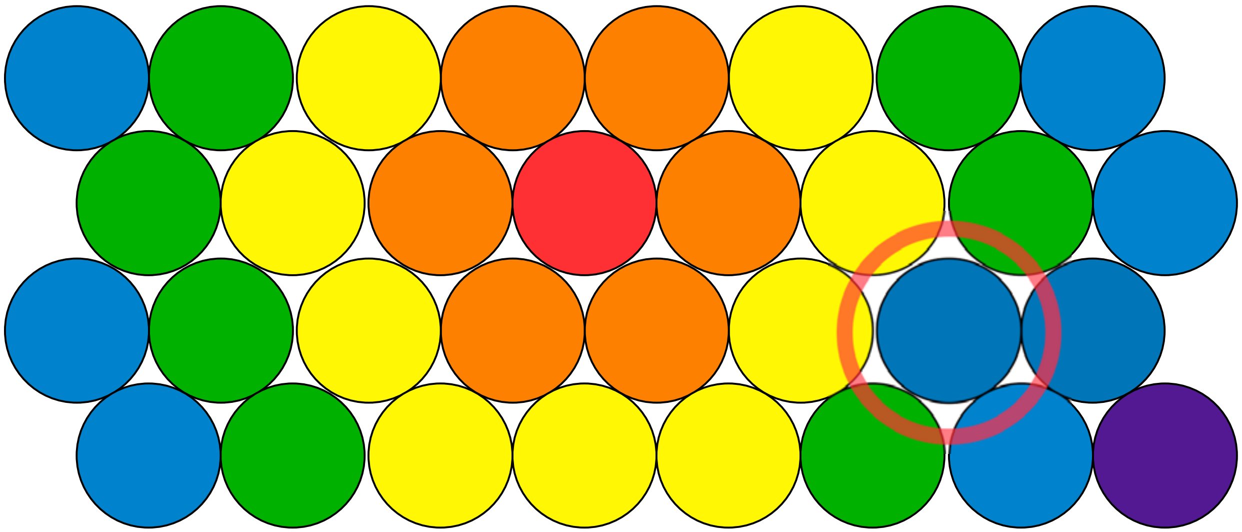
Scenario: One Leader Per Two-Hop Radius

Example Errors: Token Passing



Scenario: If a node has the token, exactly one of it's neighbors must have had it last timestep

Example Errors: Gradient Field



Scenario: Gradient Values Must Be Smooth

Expressing Distributed Error Conditions

Requirements:

- Ability to specify shape of trigger groups
- Temporal operators
- Simple syntax (reduce programmer effort/learning curve)

A Solution:

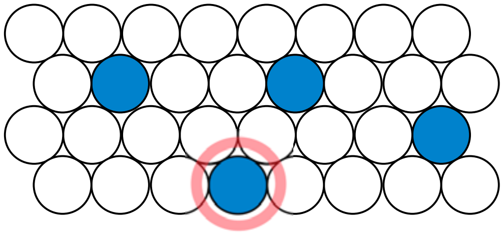
- Inspired by Linear Temporal Logic (LTL)
  - A simple extension to first-order logic
  - Proven technique for single-robot debugging [Lamine01]
- Assumption: Trigger groups must be connected
  - For practical/efficiency reasons

Watchpoint Primitives

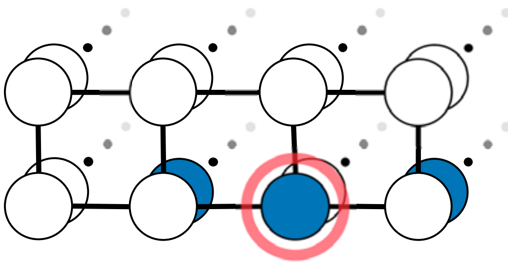
nodes(a,b,c); n(b,c) & (a.var > b.var) & (c.prev.var != 2)

- Modules (implicitly quantified over all connected sub-ensembles)
- Topological restrictions (pairwise neighbor relations)
- Boolean connectives
- State variable comparisons (distributed)
- Temporal operators

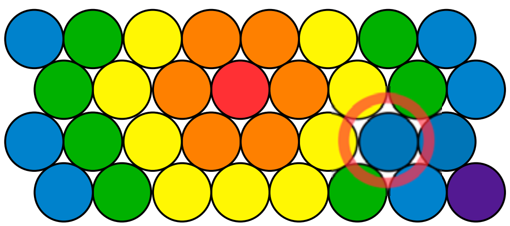
Distributed Errors: Example Watchpoints



nodes(a,b,c);n(a.b) & n(b,c) & (a.isLeader == 1) & (c.isLeader == 1)



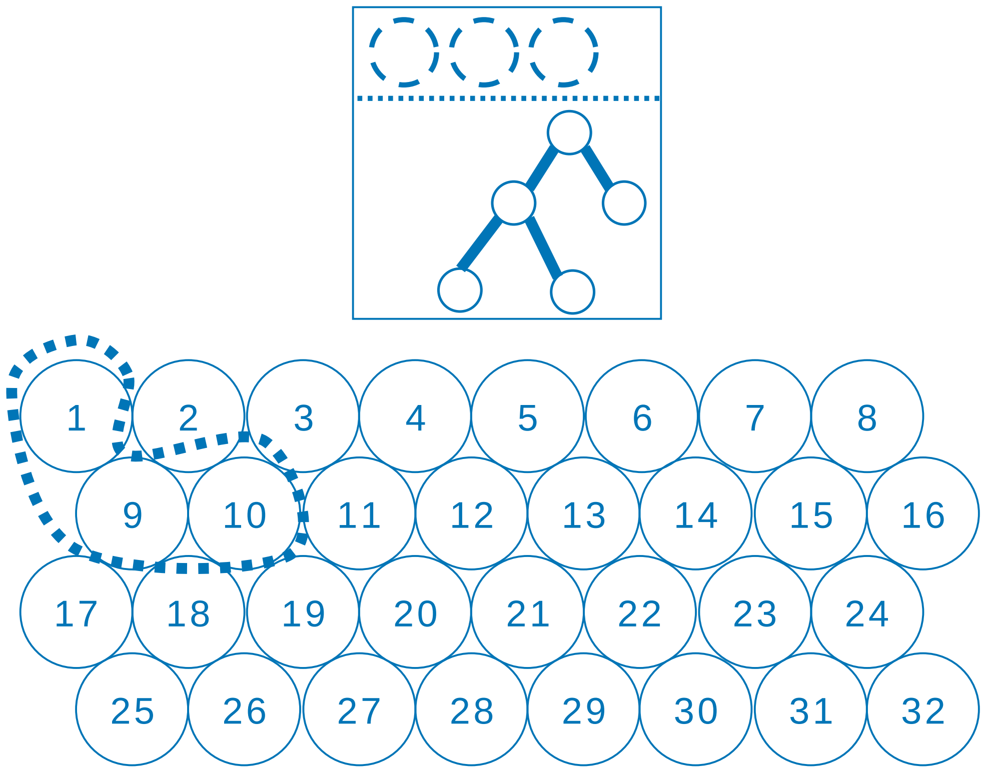
nodes(a,b,c);n(a,b) & n(a,c) & (a.token == 1) & (b.prev.token == 1) & (c.prev.token == 1)

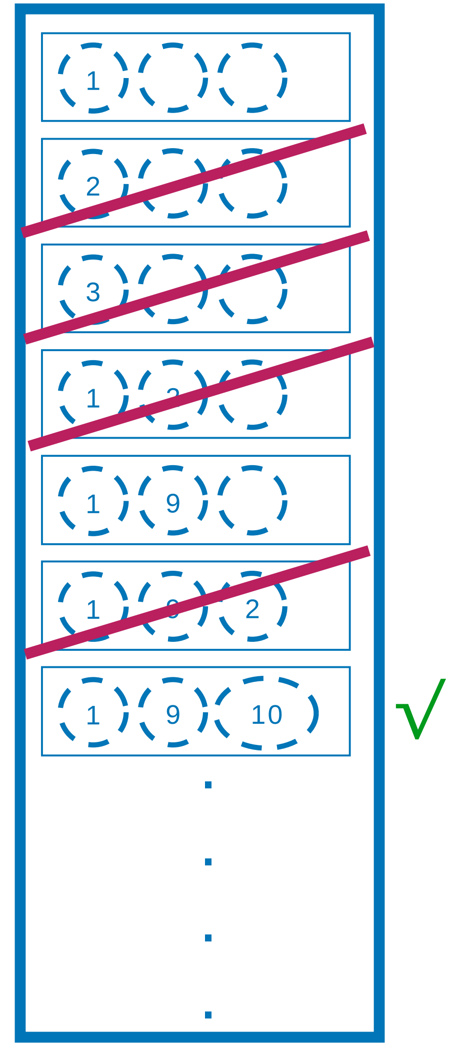


nodes(a,b);(a.state - b.state > 1)

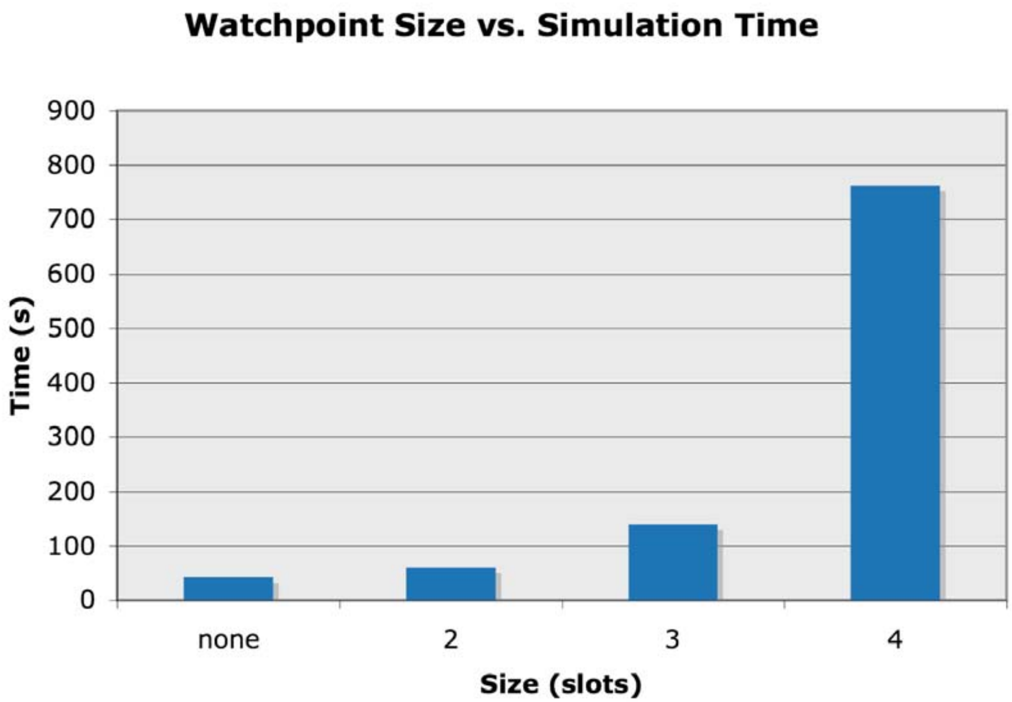
Watchpoint Execution

nodes(a,b,c)...



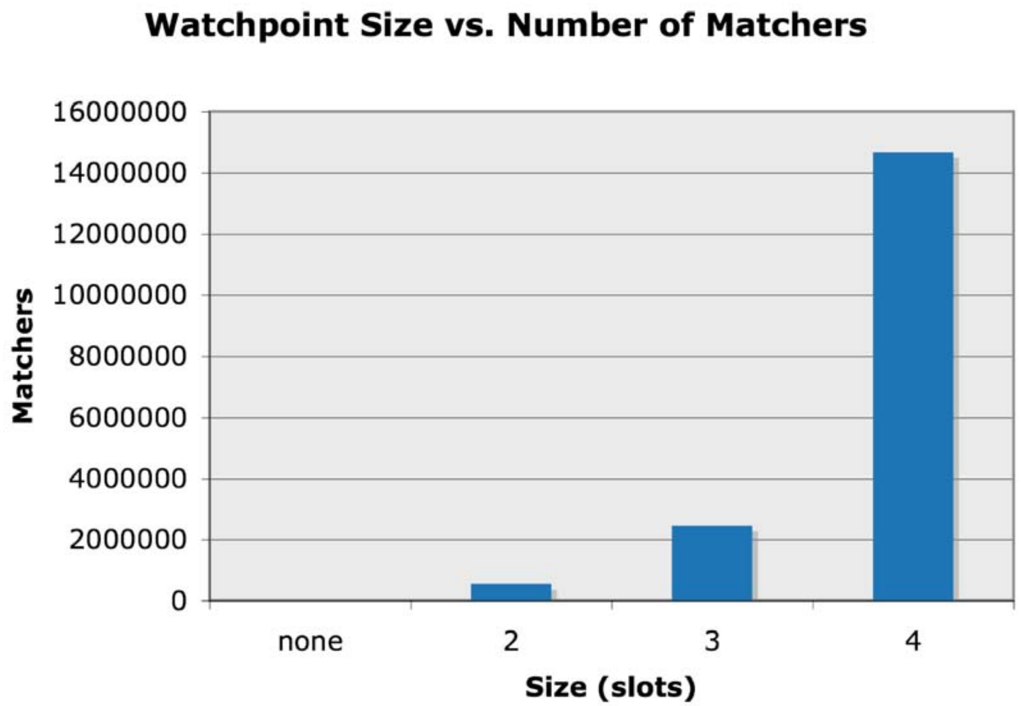


Performance: Watchpoint Size



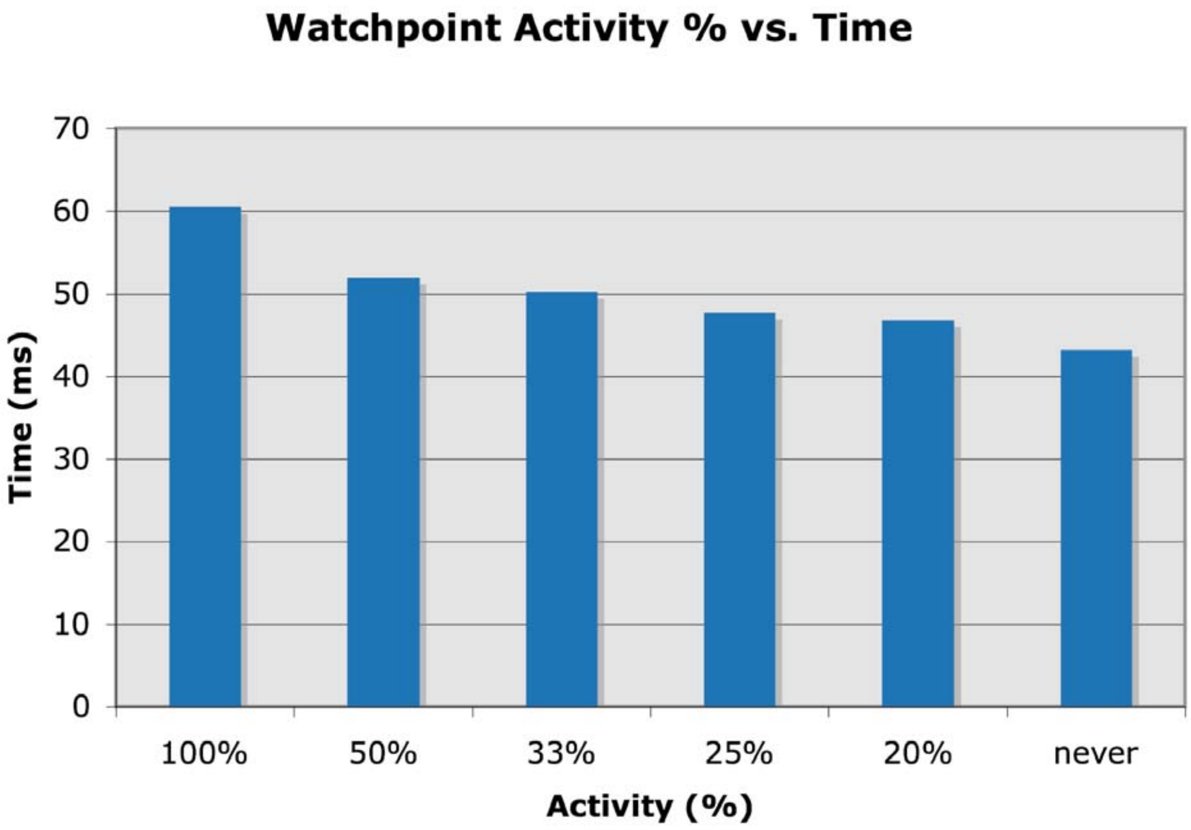
- 1000 modules, running for 100 timesteps
- Simulator overhead excluded
- Application: data aggregation with landmark routing
- Watchpoint: are the first and last robots in the watchpoint in the same state?

Performance: Number of Matchers



- This particular watchpoint never terminates early
- Number of matchers increases exponentially
- Time per matcher remains within factor of 2
- Details of the watchpoint expression more important than size

Performance: Periodically Running Watchpoints



Future Work

- Distributed implementation
- More optimization
- User validation
- Additional predicates

Conclusions

- Simple, yet highly descriptive syntax
- Able to detect errors missed by more conventional techniques
- Low simulation overhead

Thank You

Backup Slides

Optimizations

- Temporal span
- Early termination
- Neighbor culling
- (one slide per)