

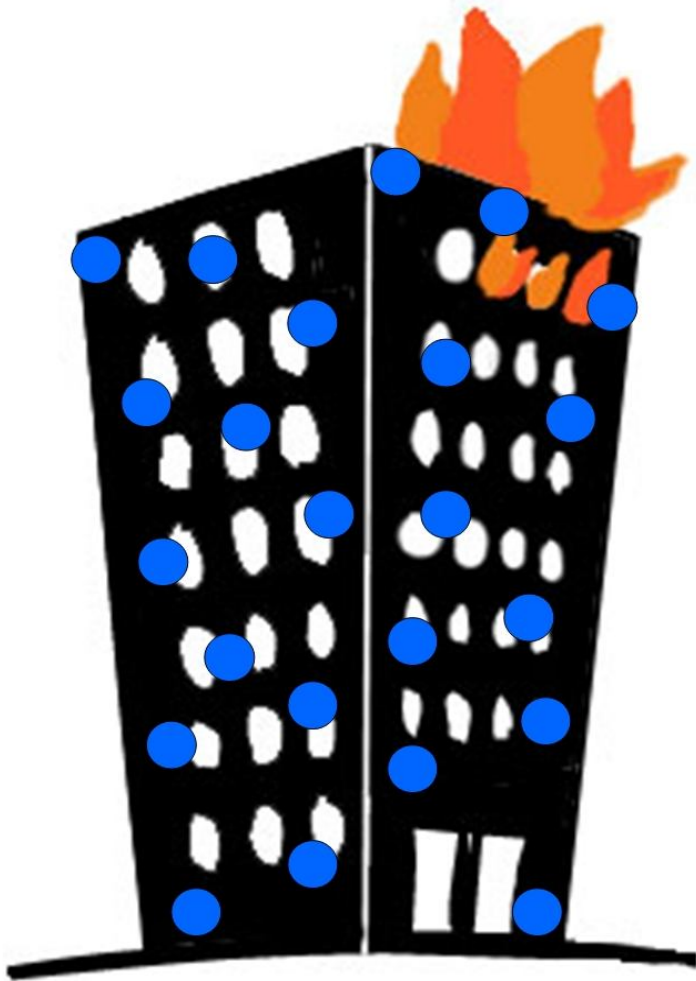
# Run Time Assurance of Application- Level Requirements in Wireless Sensor Networks

Yafeng Wu, Krasimira Kapitanova,  
Jingyuan Li, John A. Stankovic,  
Sang H. Son, and Kamin Whitehouse



IPSN 2010

# The Problem



- Continuous and reliable operation of WSN applications is hard to guarantee
- There could be changes in the operating conditions:
  - Structural building changes
  - Network topology
  - Nodes might stop functioning

# Current Solutions

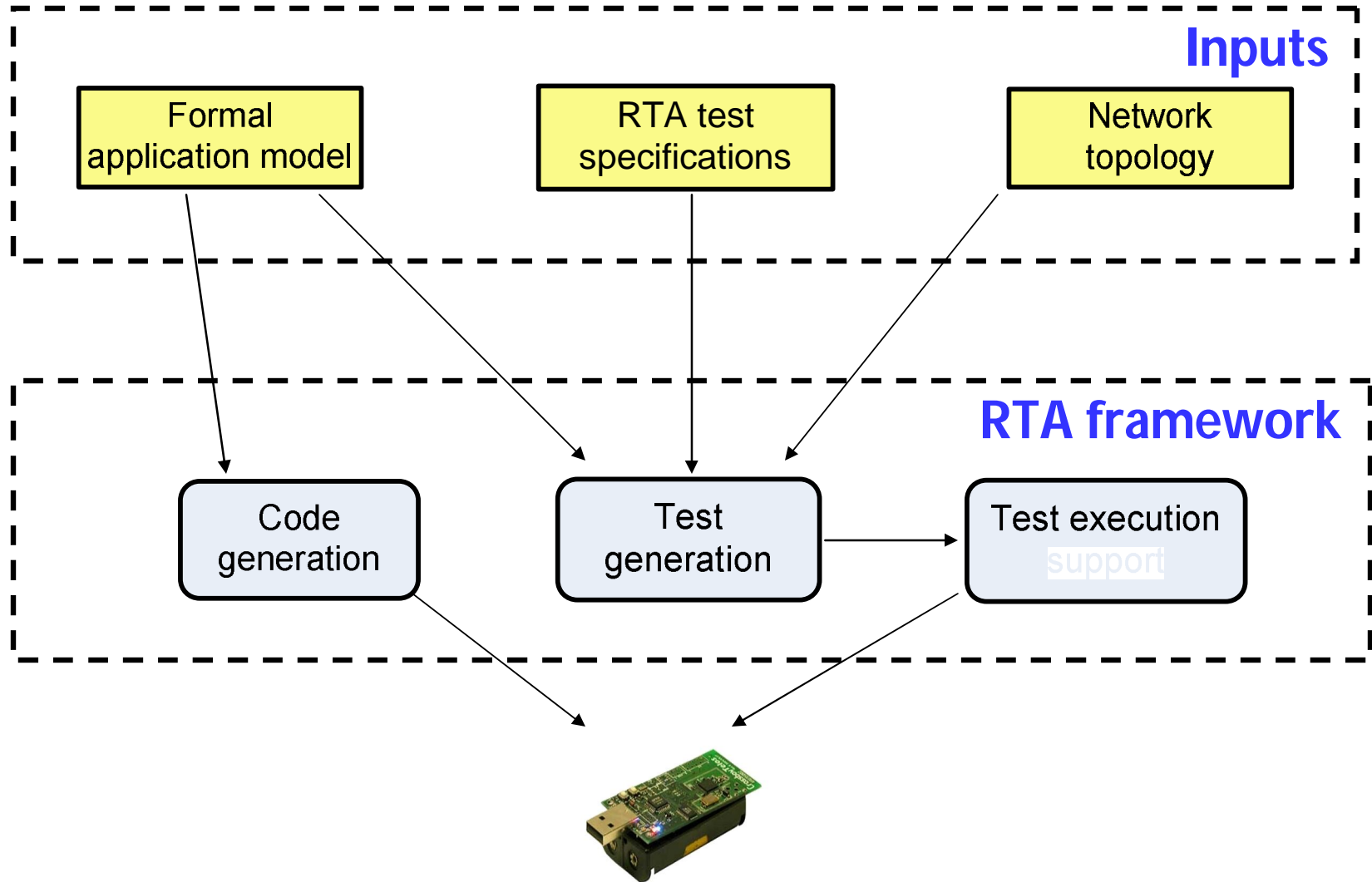
- Prior deployment analysis
  - Testing
  - Debugging
- *Post mortem* analysis
  - Debugging
- Monitoring low-level components of the system
  - network health monitoring system

**Disadvantage:** incur high levels of both **false positives** and **false negatives**

# Run Time Assurance

- Validate at run time that the WSN system functions correctly in terms of meeting its high-level application requirements, irrespective of any changes to the operating conditions.
- The validation is performed periodically at the application level
- The application is tested at run time
  - Mapping inputs to outputs

# RTA Framework



# Contributions

- A novel RTA methodology
- A prototype implementation based on a formal description language
- New test reduction techniques
- A quantitative evaluation of the RTA methodology:
  - comparison with an existing network health monitoring system

# Outline

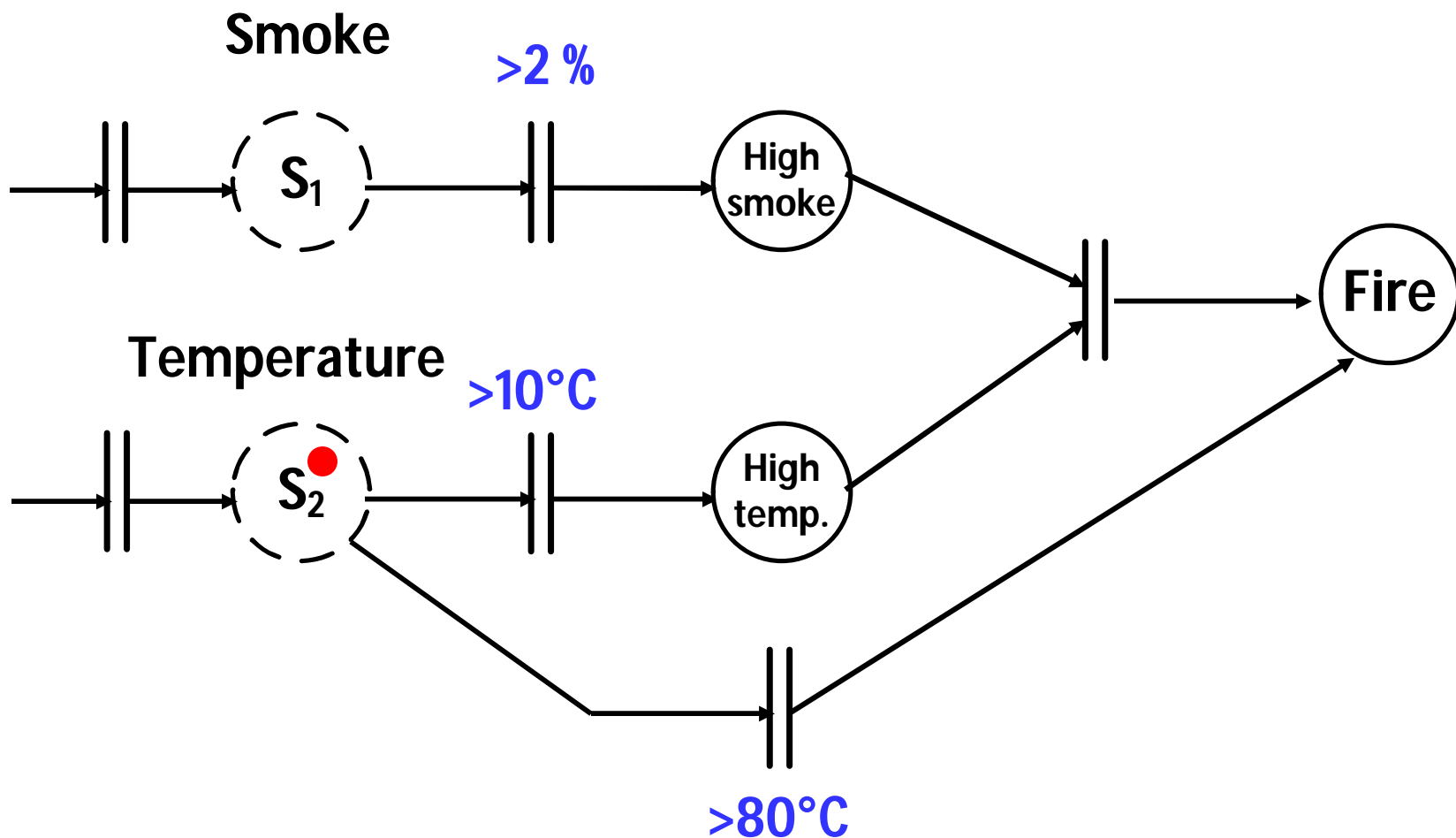
- Model-based application specification
- Code generation
- Test execution support
- Test generation
- Experimental results

# Outline

- Model-based application specification
- Code generation
- Test execution support
- Test generation
- Experimental results

# Model-based Application Specification

Sensor Network Event Description Language (SNEDL)



# Test Specification

//Declare the basic elements of the tests

**Time** T1;

**Region** R1, R2;

**Event** FireEvent;

//Define the elements

T1=07:00:00, \*/1/2010;

R1={Room1};

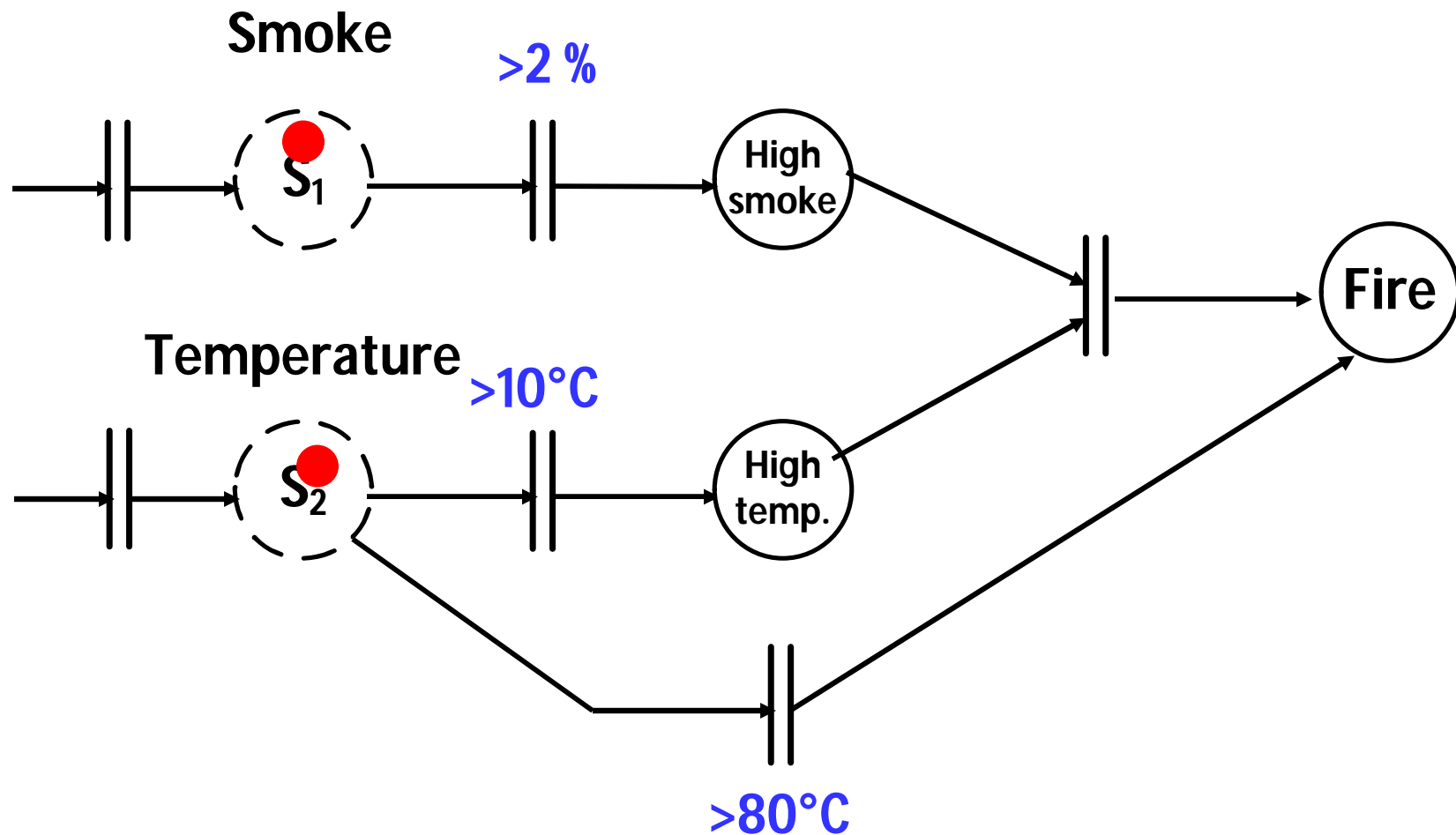
R2={Room2};

FireEvent = Fire @ T1;

# Outline

- Model-based application specification
- Code generation
- Test execution support
- Test generation
- Experimental results

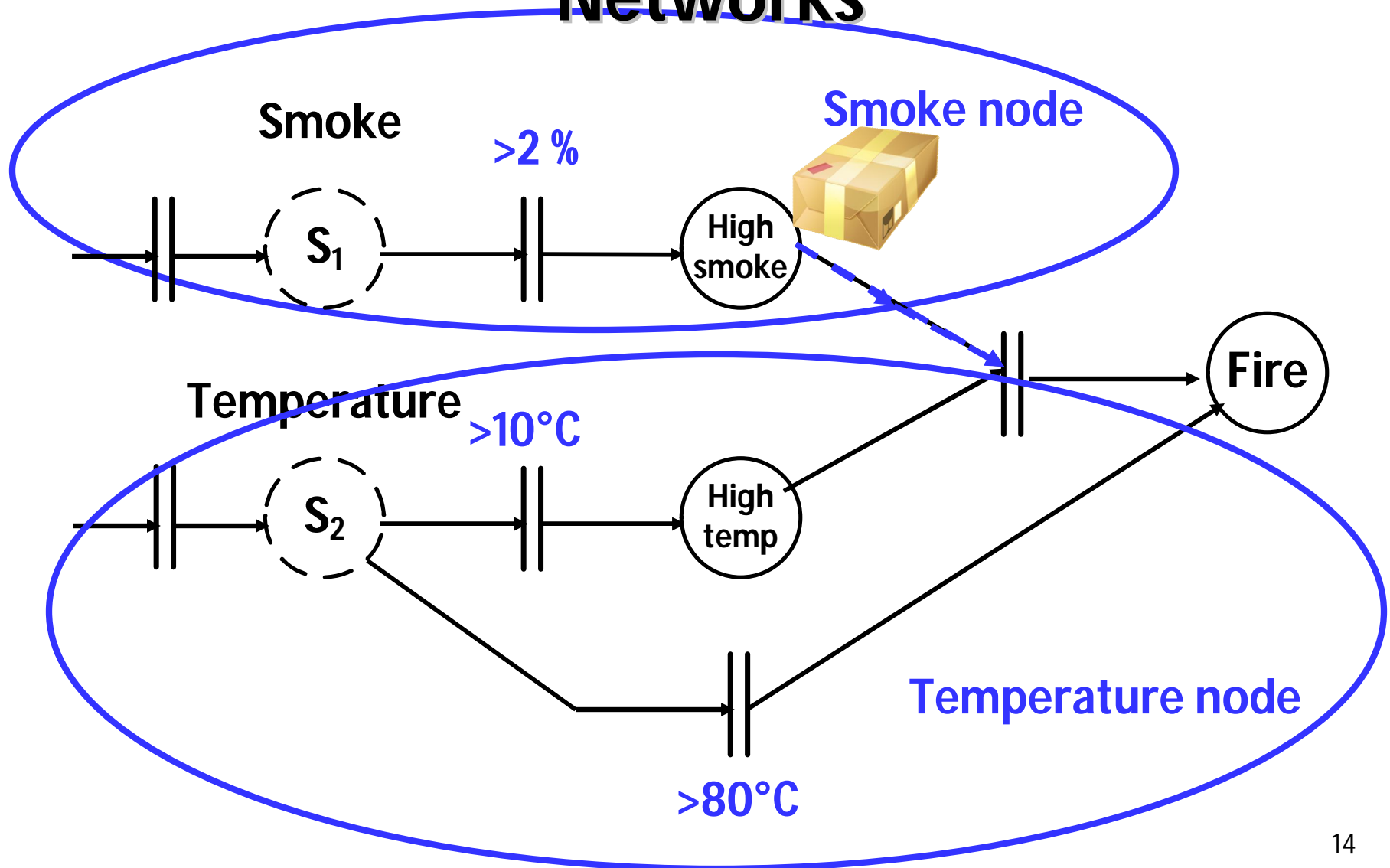
# Token Flow Program Execution



# Code Generation

- Code is automatically generated from the formal model
- Advantages of the token – flow model:
  - efficiently supports testing at run time
  - it is easy to monitor execution states and collect running traces
  - we can easily distinguish between real and test events

# Code Generation For Heterogeneous Networks



# Outline

- Model-based application specification
- Code generation
- Test execution support
- Test generation
- Experimental results

# Test Execution Support

- Fully automates the test execution
- Delivers the tests to the nodes
- Initiates the tests
- Retrieves the results

# Outline

- Model-based application specification
- Code generation
- Test execution support
- Test generation
- Experimental results

# Test Generation

Input-to-output test suite size:

$$R \cdot I \cdot S$$

- *R*: size of the sensing value range
- *I*: number of sensors per node
- *S*: number of nodes in the network

Example: 100 nodes

- temperature sensor 0-100C;
- smoke sensor 0-100 %;

Number of tests  **$10^{400}$**

# Test Reduction Techniques

## no reduction

- Each node **Number of tests:  $10^{16}$** 
  - Temperature (0-100° C)
  - Smoke (0-100%)

S	T
0	0
0	1
0	2
...	...
100	100



S	T
0	0
0	1
0	2
...	...
100	100



S	T
0	0
0	1
0	2
...	...
100	100



S	T
0	0
0	1
0	2
...	...
100	100

# Test Reduction Techniques

## static model analysis

- Each node **Number of tests:  $2^{16}$** 
  - Temperature (0-100° C)    - Smoke (0-100%)

Temp:  $\leq 75^{\circ}\text{C}$   
and  $> 75^{\circ}\text{C}$

Smoke:  $\leq 5\%$   
and  $> 5\%$

S	T
0	0
0	1
0	2
...	...
100	100



Temperature  
threshold :  $75^{\circ}\text{C}$

Smoke threshold: 5%

Temp:  $\leq 75^{\circ}\text{C}$   
and  $> 75^{\circ}\text{C}$

Smoke:  $\leq 5\%$   
and  $> 5\%$

S	T
0	0
0	1
0	2
...	...
100	100



Temp:  $\leq 75^{\circ}\text{C}$   
and  $> 75^{\circ}\text{C}$

Smoke:  $\leq 5\%$   
and  $> 5\%$

S	T
0	0
0	1
0	2
...	...
100	100



Temp:  $\leq 75^{\circ}\text{C}$   
and  $> 75^{\circ}\text{C}$

Smoke:  $\leq 5\%$   
and  $> 5\%$

S	T
0	0
0	1
0	2
...	...
100	100

20

# Test Reduction Techniques

## network topology

- Each node **Number of tests: 22**
  - Temperature (0-100° C) - Smoke (0-100%)

Temp:  $\leq 75^{\circ}\text{C}$   
and  $> 75^{\circ}\text{C}$

Smoke:  $\leq 5\%$   
and  $> 5\%$



Temperature  
threshold :  $75^{\circ}\text{C}$

Smoke threshold: 5%

Temp:  $\leq 75^{\circ}\text{C}$   
and  $> 75^{\circ}\text{C}$

Smoke:  $\leq 5\%$   
and  $> 5\%$



Temp:  $\leq 75^{\circ}\text{C}$   
and  $> 75^{\circ}\text{C}$

Smoke:  $\leq 5\%$   
and  $> 5\%$



Temp:  $\leq 75^{\circ}\text{C}$   
and  $> 75^{\circ}\text{C}$

Smoke:  $\leq 5\%$   
and  $> 5\%$



# Test Reduction Techniques

## node redundancy

- Each node **Number of tests: 82**
  - Temperature (0-100° C) - Smoke (0-100%)

Temp:  $\leq 75^{\circ}\text{C}$   
and  $> 75^{\circ}\text{C}$

Smoke:  $\leq 5\%$   
and  $> 5\%$



Temperature  
threshold :  $75^{\circ}\text{C}$

Smoke threshold: 5%

Temp:  $\leq 75^{\circ}\text{C}$   
and  $> 75^{\circ}\text{C}$

Smoke:  $\leq 5\%$   
and  $> 5\%$



Temp:  $\leq 75^{\circ}\text{C}$   
and  $> 75^{\circ}\text{C}$

Smoke:  $\leq 5\%$   
and  $> 5\%$



Temp:  $\leq 75^{\circ}\text{C}$   
and  $> 75^{\circ}\text{C}$

Smoke:  $\leq 5\%$   
and  $> 5\%$



# Outline

- Model-based application specification
- Code generation
- Test generation
- Test execution support
- Experimental results

# Experimental Setup

- We use a **fire detection application** to compare RTA to
  - Pure system with no RTA or HM support (FD system)
  - Network health monitoring system (FD system + HM)
- 21 TelosB nodes equipped with a light sensor
- Nodes are divided into different number of rooms (2-10)
- Poisson distribution used to:
  - Choose the rooms to test
  - Choose the nodes that stop executing the application
  - Choose the time the node failures occur

# Evaluation

- Test reduction techniques
- Robustness to failures
  - Node failures
  - Application-logic failures
- Overhead
  - Maintenance overhead
  - Message overhead

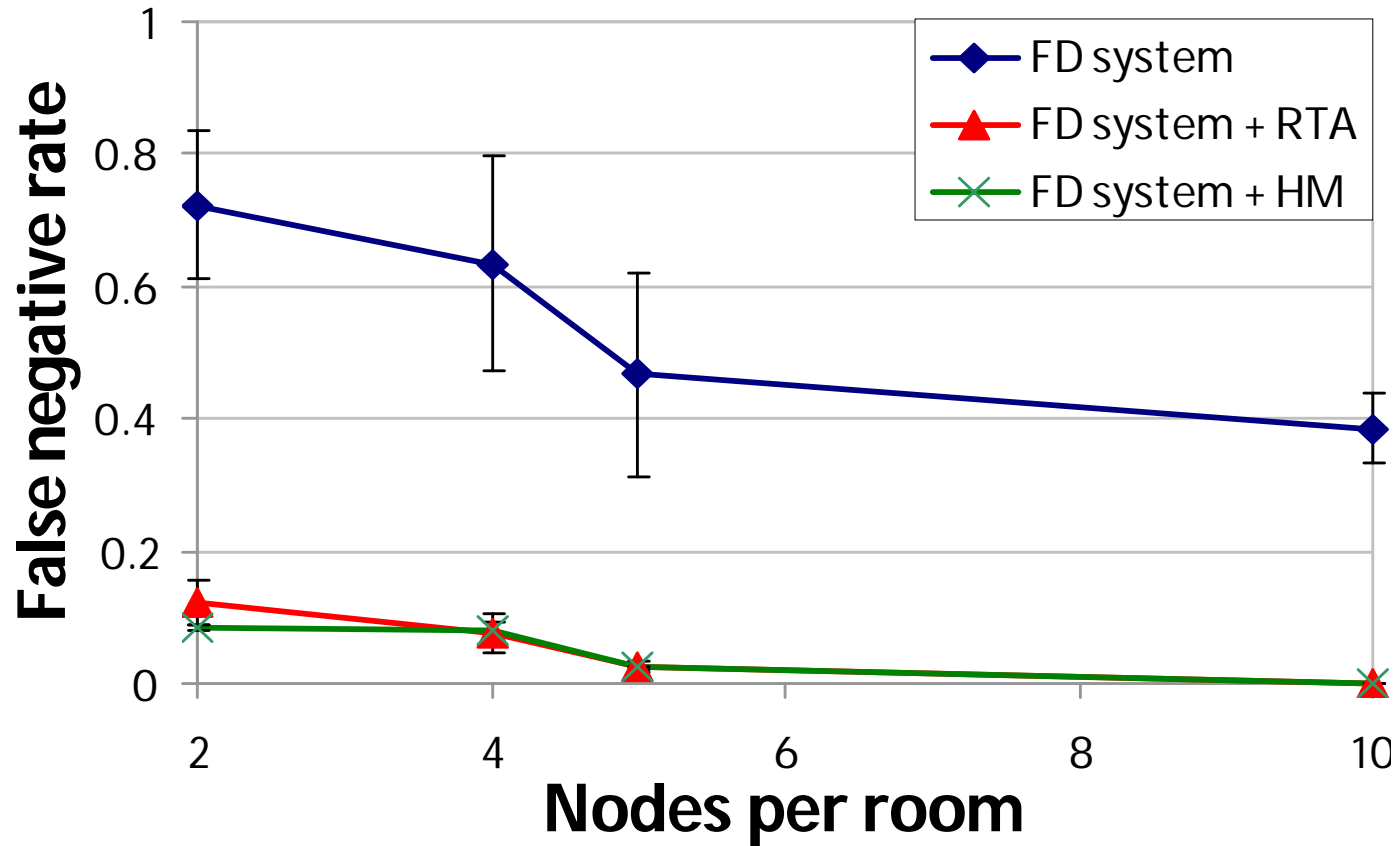
# Test Reduction

- 20 nodes each with a light sensor (0-100)

Rooms	2	4	5	10
Nodes per room	10	5	4	2
Baseline (no reduction)	$11 \cdot 10^{35}$	$11 \cdot 10^{35}$	$11 \cdot 10^{35}$	$11 \cdot 10^{35}$
Static analysis reduction	$11 \cdot 10^{11}$	$11 \cdot 10^{11}$	$11 \cdot 10^{11}$	$11 \cdot 10^{11}$
Topology reduction	$2 \cdot 10^7$	4096	1280	160
Redundancy reduction	8	16	20	40

# Robustness to Failure

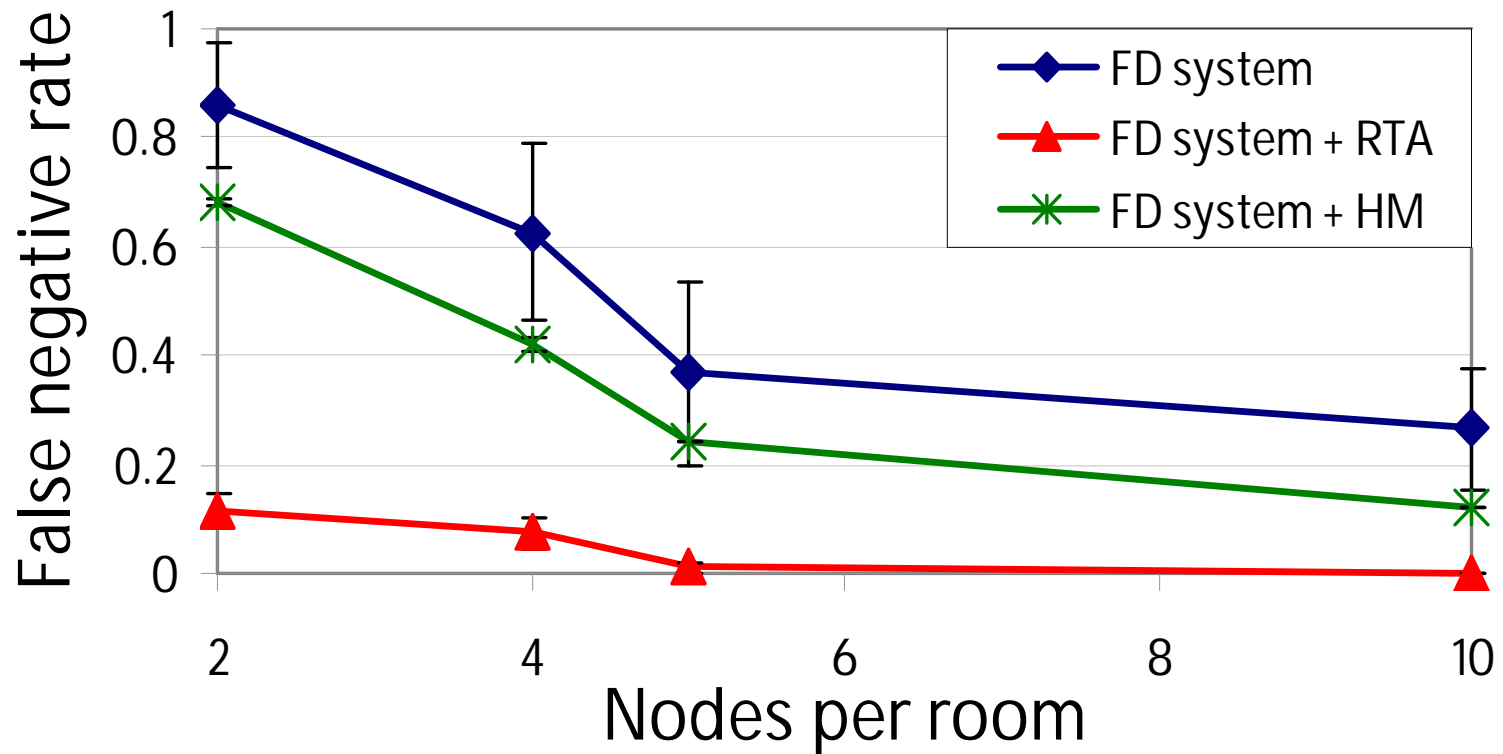
## node failure



RTA and HM achieve similar false negative rates.

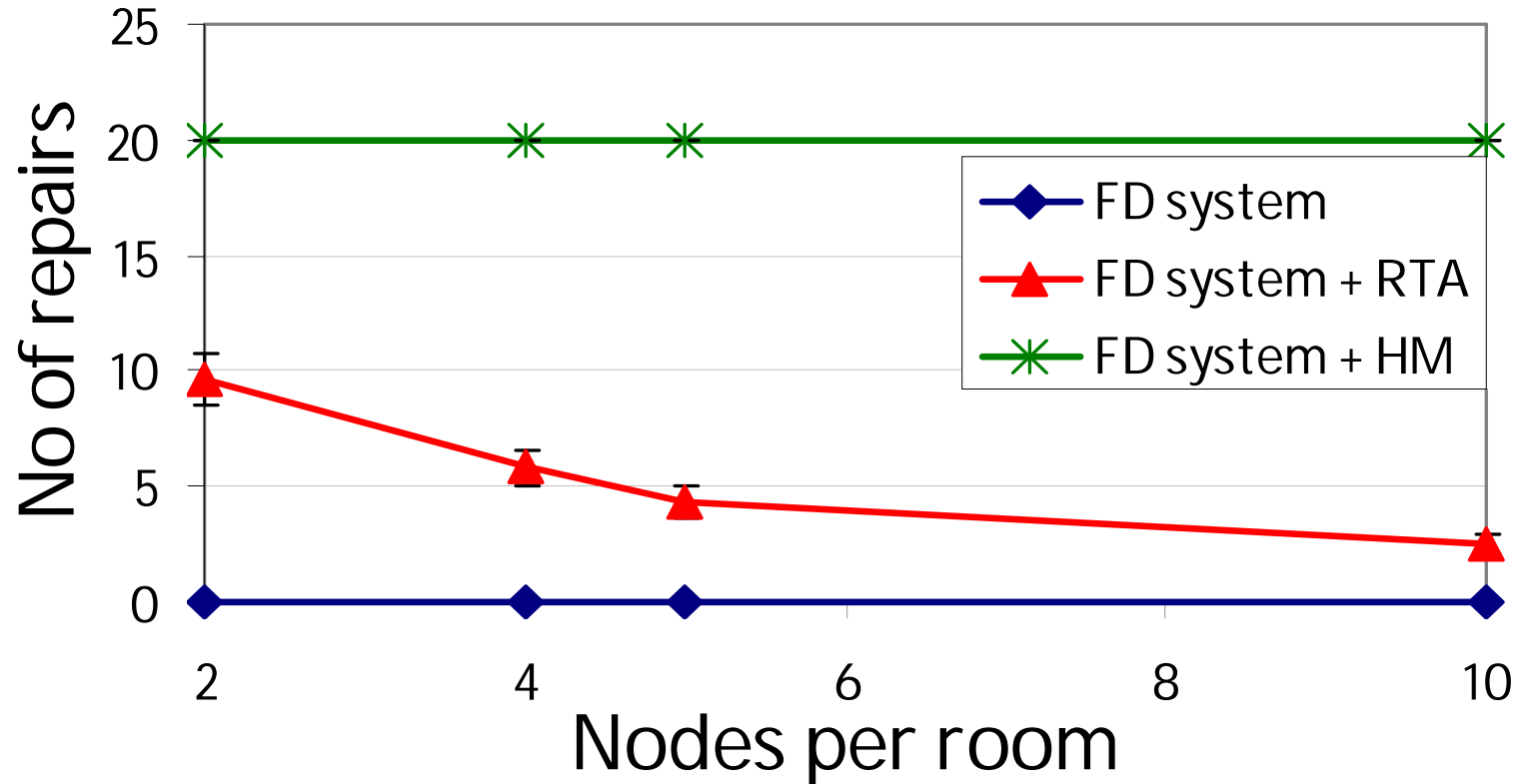
# Robustness to Failures

## location failure



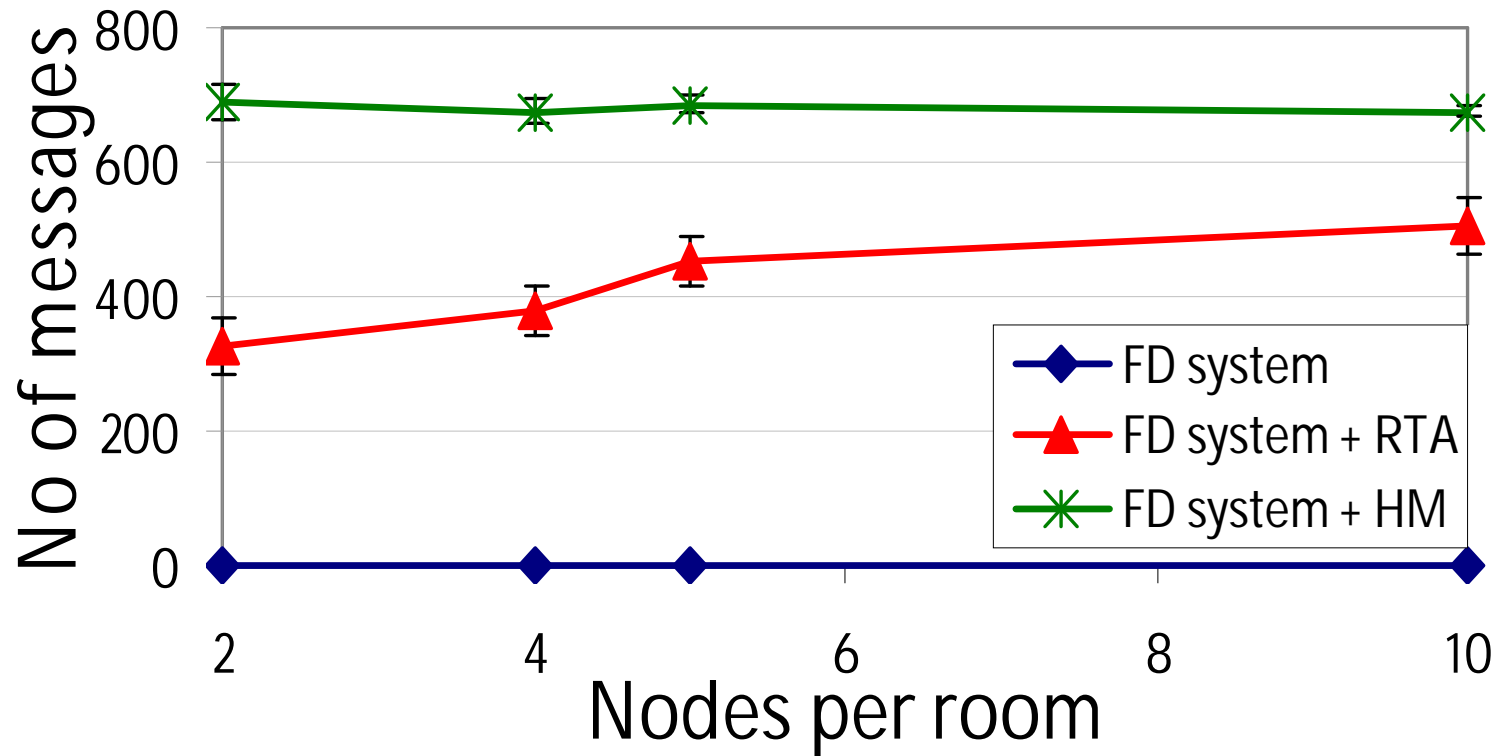
**RTA has 75% less false negatives than HM.**

# Maintenance Overhead



**RTA requires 50%-70% fewer repairs than HM.**

# Message Overhead



**RTA uses 33% less messages than HM.**

# Conclusions

- With RTA we can monitor the application-level behavior of the system.
- The RTA framework provides automatic code and test generation
- In comparison with network health monitoring, RTA:
  - misses 75% fewer system failures
  - produces 70% fewer maintenance dispatches
  - incurs 33% less messaging overhead

# Future Work

- Larger scale experiments
- Develop additional test reduction and prioritization schemes
- Apply the RTA methodology to other applications
- Apply the RTA methodology to system safety

**Questions?**