Performance Analysis with Prism

Formal Verification and Simulation for Performance Analysis for Probabilistic Broadcast Protocols

Ansgar Fehnker,

joint work with Peng Gao

(accepted at Ad-Hoc Now 2006)

The Content

- The project
- The problem
- The protocol
- The assumptions
- The approach
- The results
- The future

The PEWNA project

Wireless sensor networks

Aggregate of small, portable devices

- battery-operated computing power
- gather sensor information distributedly
- wireless communications
- multi-hop communication

The PEWNA project

Challenges for network and application design

- Unpredictable behaviour of the environment.
- Dynamic nature of network with respect to spatial distribution and ad hoc addition of nodes.
- Resilience to message loss and node failure.
- Power efficiency to maximise battery life and network lifetime.

The PEWNA project

Project Goals

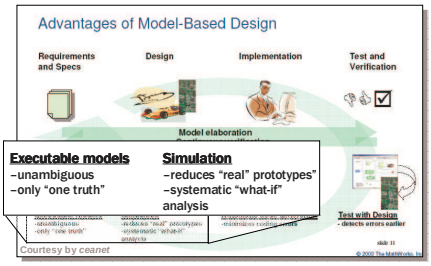
- Notations, analysis tools and reusable formal models for wireless network protocols.
- Application of probabilistic and hybrid model checking techniques for performance evaluation.
- Abstraction techniques to scale probabilistic and hybrid model checking techniques.

The problem

Simulation is an imperfect tool

Current Best Practice

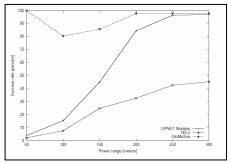
Model-Based Design



Related Work

On the Accuracy of MANET Simulators
Cavin, Sasson and Schiper (2002)

- Different simulators give different answers
- Even for simple protocols
- Semantics defined by simulator.
- Low level details as important as high level protocol



Related Work

Experimental Evaluation of Wireless Simulation Assumptions
Kotz, Newport et al. (2004)

- Assumptions render results useless
- Common assumptions:
  1. The earth is flat
  2. The transmission area is circular.
  3. All radios have equal range.
  4. If I can hear you, you can hear me.
  5. If I can hear you at all, I can hear you perfectly.
  6. Signal strength is a simple function of distance.

The Problem

Common Solution

- More details
- Hardware in the loop simulation
- Precise specific assumptions
- Specific results for specific instance of a system

The Problem

Our Approach

- More abstract model
- Formal model with well defined semantics
- Precise assumptions
- General results for a range of systems

The Protocol

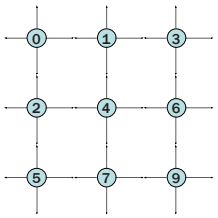
Gossiping is simple

The Protocol

Flooding and Gossiping

- Simple protocols
- Commonly used in some phase of wireless protocols
- Many ambiguous and conflicting simulation models

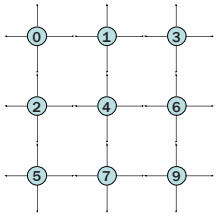
Flooding and Gossiping



Flooding protocol
- listen to medium
- if you receive a message
  - send message
- go to sleep

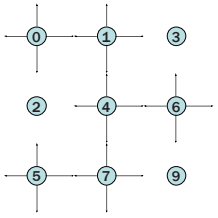
Properties of flooding
- simple
- used for routing
- redundant
- prone to collisions
- inefficient

Flooding and Gossiping



Gossiping protocol
- listen to medium
- if you receive a message
  - send message with probability p
- go to sleep

Flooding and Gossiping



Gossiping protocol
- listen to medium
- if you receive a message
  - send message with probability p
- go to sleep

Properties of gossiping
- still simple
- reduces redundancy
- reduces collisions
- improves efficiency

The assumptions

What else?

Common assumption

Common assumptions

- Absence of collisions
- Perfectly synchronous execution
- No clock drift
- Perfect medium

Our approach

For a simple protocol show

- The effect of common assumptions on performance results
- How to model a wireless protocol
  - a less ambiguous model
  - with explicit assumptions
  - abstracting from low level detail
- How to use a probabilistic model checker (PRISM)
- How to obtain performance style results

Modelchecking

PRISM

A Probabilistic Symbolic Model Checker (Uni Birmingham)

Supports:
- Discrete-Time Markov Chains (DTMCs)
- Continuous-Time Markov Chains (CTMCs)
- Markov Decision Processes (MDPs)

Checks:
- Probabilistic temporal logic (PCTL)
- Example: node 7 receives a message with probability > 0.8

Formal Model

Model protocols as formal models (in Prism)

- Well defined semantics,
- Abstraction from low level detail

Example

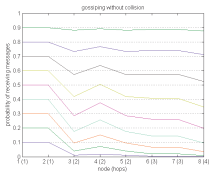
```
module node4
act4:  bool init true;
send4: bool init false;

[tick]  act4 & !send4 & (send1|send3|send5|send7)
        -> psend: (act4'=true)&(send4'=true) + (1-psend):(act4'=false)&(send4'=false);
[tick]  act4 & !send4 & !(send1|send3|send5|send7)-> (act4'=true) & (send4'=false);
[tick]  act4 &  send4 -> (act4'=false) & (send4'=false);
[tick] !act4          -> (act4'=false) & (send4'=false);
endmodule
```

PRISM model of gossiping protocol

Performance Evaluation

Model protocol as formal models

- Well defined semantics, allows abstraction from low level detail
- Verification produces exact answers
- Makes assumptions explicit



- **PRISM results for gossiping w/o collision**
- **Results are exact probabilities rather than approximations**

Common assumption

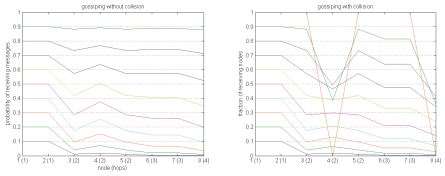
Common assumptions

- Absence of collisions
- Perfectly synchronous execution
- No clock drift
- Perfect medium
- Random choice equals lossy channel

Collisions

Model protocol as formal models

- Well defined semantics, allows abstraction from low level detail
- Verification produces exact answers
- Makes assumptions explicit



Common assumption

Common assumptions

- Absence of collisions
- Perfectly synchronous execution
- No clock drift
- Perfect medium

Synchronization

Model protocols as formal models

- Well defined semantics, allows abstraction from low level detail
- Verification produces exact answers
- Makes assumptions explicit
- Clean understanding of concurrency

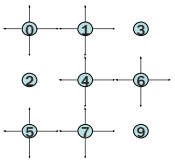
```
for i=find(sending) %for all nodes that are sending

   if(rand <= p)
        for j=1:4
            if(C(i,j) ~= 0) % if it has a neighbour
                        receive(C(i,j)) = 1; %then neighbour receives a msg
            end
        end;
   end;
end;
```

Matlab code for Monte-Carlo simulation: *implicitly synchronous*

Synchronization

Model protocols as formal models

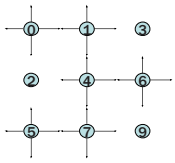
- Well defined semantics, allows abstraction from low level detail
- Verification produces exact answers
- Makes assumptions explicit
- Clean understanding of concurrency



Synchronization

Model protocols as formal models

- Well defined semantics, allows abstraction from low level detail
- Verification produces exact answers
- Makes assumptions explicit
- Clean understanding of concurrency



Does it matter?

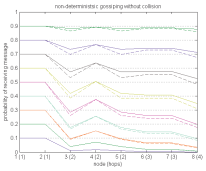
The imagination driving Australia's ICT future.

NATIONAL ICT AUSTRALIA

Synchronization

Model protocols as formal models

- Well defined semantics, allows abstraction from low level detail
- Verification produces exact answers
- Makes assumptions explicit
- Clean understanding of concurrency



- PRISM results for gossiping w/o collision
- Model with non-deterministic execution order
- Minimal and maximal probability cover all possible execution orders

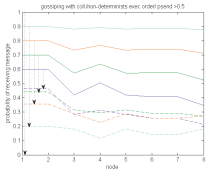
The imagination driving Australia's ICT future.

NATIONAL ICT AUSTRALIA

Synchronization

Model protocols as formal models

- Well defined semantics, allows abstraction from low level detail
- Verification produces exact answers
- Makes assumptions explicit
- Clean understanding of concurrency



- PRISM results for gossiping with collision
- Model with non-deterministic execution order
- Minimal and maximal probability cover all possible execution orders
- Model checking gives guarantees that simulation cannot.

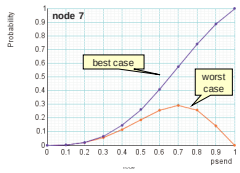
The imagination driving Australia's ICT future.

NATIONAL ICT AUSTRALIA

Synchronization

Model protocols as formal models

- Well defined semantics, allows abstraction from low level detail
- Verification produces exact answers
- Makes assumptions explicit
- Clean understanding of concurrency



- PRISM results for gossiping with collision
- Model with non-deterministic execution order
- Minimal and maximal probability cover all possible execution orders
- Model checking gives guarantees that simulation cannot.

The imagination driving Australia's ICT future.

NATIONAL ICT AUSTRALIA

Common assumption

Common assumptions

- Absence of collisions
- Perfectly synchronous execution
- No clock drift
- Perfect medium

Timing

Unreliable timing

- Modelled as probabilistic waiting
- Compared different variants of the timing model

Example

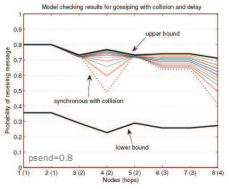
```
[tick] active4=1 & send4=0 & send1+send3+send5+send7 =1
                           ->   (1-psend):        (active4'=0)&(send4'=0)
                           +    psend*(1-pdelay): (active4'=1)&(send4'=1)
                           +    psend*   pdelay:  (active4'=2)&(send4'=0);
[tick] active4=2 ->(1-pdelay): (active4'=1)&(send4'=1)
                           +    pdelay:  (active4'=2)&(send4'=0);
```

PRISM model of simple delay

Timing

Unreliable timing

- Modelled as probabilistic waiting
- Compared different variants of the timing model



- **PRISM results for simple timing model**
- **Effect of collisions vanishes with an increasing probability for delay**
- **Firm upper and lower bound provided by non-deterministic model**

Common assumption

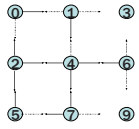
Common assumptions

- Absence of collisions
- Perfectly synchronous execution
- No clock drift
- Perfect medium

Imperfect Medium

Model medium by lossy channels

- Abstract from actual topology
- Nodes a connected by channels
- Messages are received with a certain probability

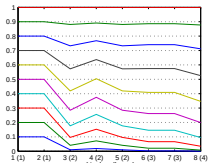


Rather than sending with a certain probability

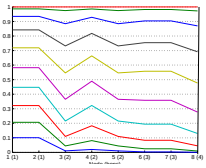
Imperfect Medium

Model medium by lossy channels

- Performance of lossy channels better than probabilistic broadcast



Probabilistic broadcast



Lossy channel

Intermediate Summary

Results

- Prism model for gossiping protocols
- Well defined semantics
- Abstraction from low level detail
- Verification produces exact answers
- Makes assumptions explicit
- Results for small networks (up to 20 nodes)

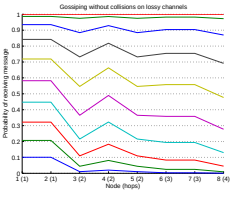
What about large networks (1000 nodes)?

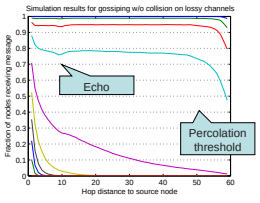
Matlab model

Monte-Carlo simulation

- Derived a simulation model (manually) from PRISM model
- Observed effects are only visible for large models/models

Baseline



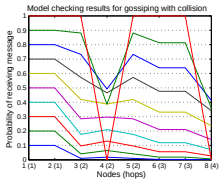


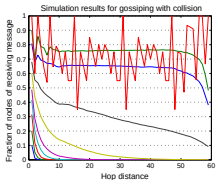
Matlab model

Monte-Carlo simulation

- Derived a simulation model (manually) from PRISM model
- Observed effects are only visible for large models/models

Collision





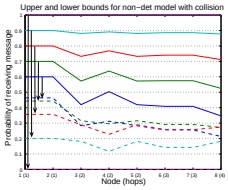
The imagination driving Australia's ICT future.

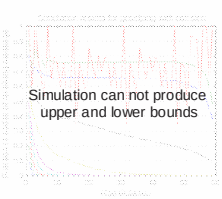
Matlab model

Monte-Carlo simulation

- Derived a simulation model (manually) from PRISM model
- Observed effects are only visible for large models/models

Execution order





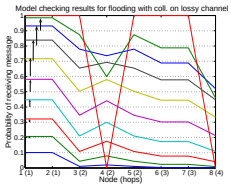
The imagination driving Australia's ICT future.

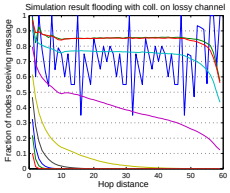
Matlab model

Monte-Carlo simulation

- Derived a simulation model (manually) from PRISM model
- Observed effects are only visible for large models/models

Lossy channel





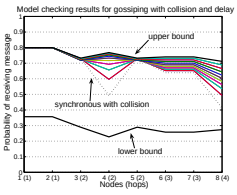
The imagination driving Australia's ICT future.

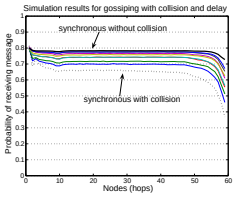
Matlab model

Monte-Carlo simulation

- Derived a simulation model (manually) from PRISM model
- Observed effects are only visible for large models/models
- Only approximate results

Delay





The imagination driving Australia's ICT future.

Observations

The imagination driving Australia's ICT future.

Observations

Gossiping

- Collisions matter
- Collisions are worst in synchronous networks
- Clock jitter mitigates effect of collisions
- Lossy channels is not random broadcast

The imagination driving Australia's ICT future.

Observations

Verification

- Simulation models based on implicit assumptions
- Different behaviour for "same" protocol
- Formal model helps to reveal hidden assumptions
- Formal model can serve as "golden" model
- Model checking can be used for performance evaluation

The imagination driving Australia's ICT future.

Observations

Simulation and Verification

- Simulation complements model checking
- We are currently two different models (artifacts) Prism/Simulink
  - Straightforward translation yields inefficient simulator
  - Prism sparse matrix model infeasible for large networks
  - Sparse matrix model hides the structure of the problem
- The problem can be solved
- Future work

  Prism to of-the-shelf simulator translation

The imagination driving Australia's ICT future.

The End

Towards part 2

Model checking

For debugging
- Better descriptions for improved protocols

For performance evaluation
- Exact results under known assumptions

For optimization
- Ask Annabelle

Baseline Model for Packet Collision

```
module node4
act4:  bool init true;
send4: bool init false;

[tick]  act4 & !send4 & (send1+send3+send5+send7=1)
        -> psend: (act4'=true)&(send4'=true) +
       (1-psend):(act4'=false)&(send4'=false);
[tick]  act4 & !send4 & !(send1+send3+send5+send7=1)
        -> (act4'=true) & (send4'=false);
[tick]  act4 &  send4
        -> (act4'=false) & (send4'=false);
[tick] !act4
        -> (act4'=false) & (send4'=false);

endmodule
```


Summary

Project focus
- Notations, analysis tools and reusable formal models
- Model checking techniques for performance evaluation.
- Abstraction techniques to scale probabilistic and hybrid model checking techniques.

Tasks
- Identification of case studies
- Formalisation of network behaviour
- Analysis with existing model checkers
- Modelling notation and semantics
- Mapping to existing tools
- Integration with proof-based techniques

Summary

Project focus
- Notations, analysis tools and reusable formal models
- Model checking techniques for performance evaluation.
- Abstraction techniques to scale probabilistic and hybrid model checking techniques.

Tasks
- Identification of case studies
- Formalisation of network behaviour
- Analysis with existing model checkers
- Modelling notation and semantics
- Mapping to existing tools
- Integration with proof-based techniques