Creating Correct Network Protocols
PhD Defence Oskar Wibling

Ansgar Fehnker

Introduction

Opponent

- Ansgar Fehnker

Affiliation

- National ICT Australia
- Managing Complexity
- Formal Methods

Research Interests

- Model checking and static analysis to support embedded software development (C/C++)
- Model checking to support protocol design in the wireless network domain

Introduction

National ICT Australia

- Australian Government backed research institute
- 5 Laboratories:
  * ATP, Sydney
  * Canberra Research Laboratory
  * The Neville Roach Laboratory, Sydney
  * Queensland Research Laboratory
  * Victoria Research Laboratory

726 Staff

- PhD and Masters Students: 301
- Corporate Staff: 107
- Researchers: 244

Creating Correct Network Protocols

Thesis Oskar Wibling

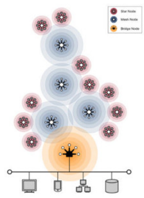
Content

- Networks
- Protocols
- Correctness
- Thesis
- Q&A

Networks

Applications

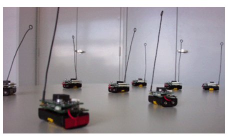
- Internet
- Home entertainment systems
- Power grid
- In-car network
- In-plane network
- Mobile phones
- WiFi
- Wireless sensor networks



Networks

Characteristics of Wireless Networks

- Ad-hoc
- Mobile
- Dynamic node creation
- Node failure
- Multi-hop communication
- Interference
- Resource constrained



Protocols

Protocols

NICTA

Purpose

- Protocols define the proper interaction between multiple components/ agents in a network.
- Protocols define the normal operating procedures
- Protocols should resilient to most failures

Protocols

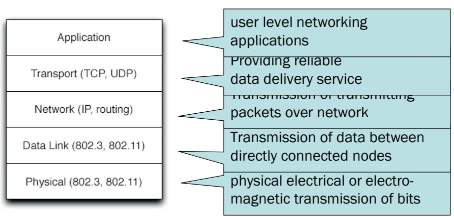
NICTA

Characteristics

- Protocol are typically organized in layers, the so-called *protocol stack*.
- Lower layers deal with the physical aspects of the network.
- Upper layers with more abstract applications.
- Upper layers assume that lower layers work correctly.

Protocols

NICTA



Page 16, Figure 1.2 *TCP/IP protocol stack.*

Routing Protocols

NICTA

Aim

- Used to set up correct routes, to transmit data from one node to another.
- Needs to find a series of intermediate nodes if sender and receiver are not directly connected

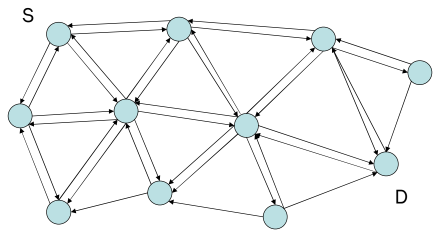
Routers

- Traditional networks uses router, i.e. dedicated nodes
- Routers provide a reliable “map” of the network.
- Ad hoc networks are more dynamic, no dedictaed routers.
- Every node has to act as a router.

Routing Protocols

NICTA

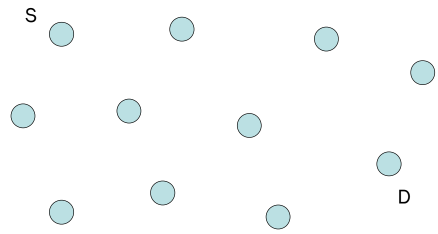
- Node S want to transmit to node D



Routing Protocols

NICTA

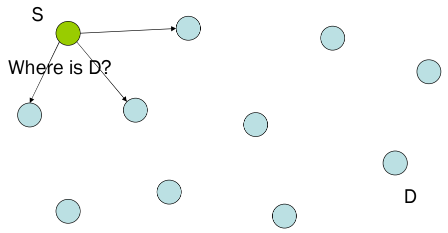
- Node S want to transmit to node D



Routing Protocols

NICTA

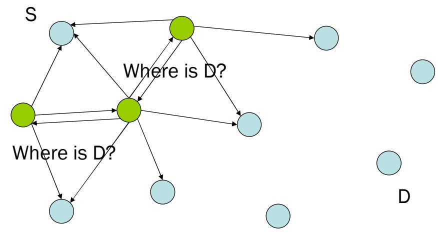
- Node S want to transmit to node D



Routing Protocols

NICTA

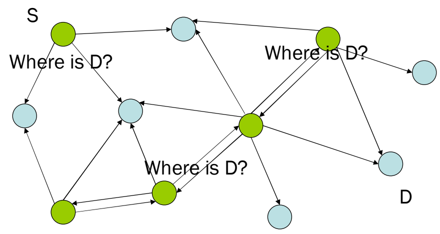
- Node S want to transmit to node D



Routing Protocols

NICTA

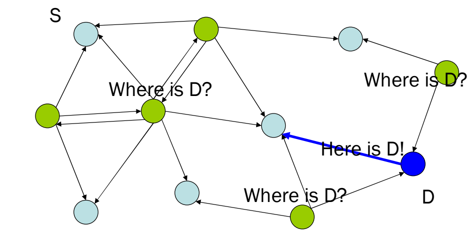
- Node S want to transmit to node D



Routing Protocols

NICTA

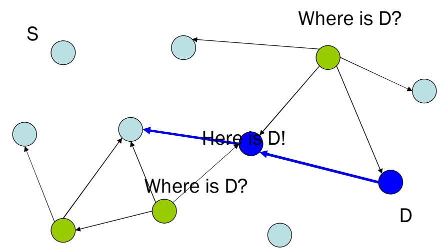
- Node S want to transmit to node D



Routing Protocols

NICTA

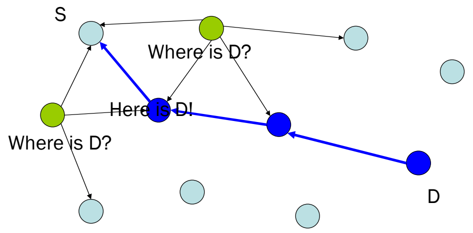
- Node S want to transmit to node D



Routing Protocols

NICTA

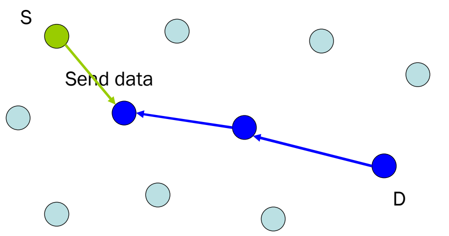
- Node S want to transmit to node D



Routing Protocols

NICTA

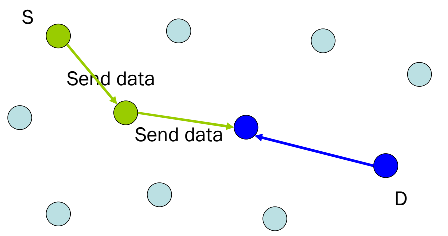
- Node S want to transmit to node D



Routing Protocols

NICTA

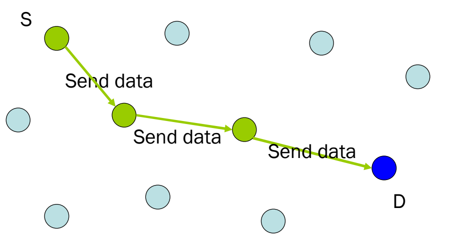
- Node S want to transmit to node D



Routing Protocols

NICTA

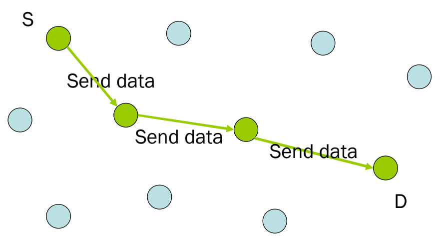
- Node S want to transmit to node D



Routing Protocols

NICTA

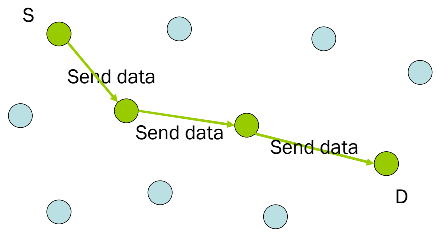
- Node S want to transmit to node D



Routing Protocols

NICTA

Challenges



Routing Protocols

NICTA

Challenges

- Nodes can move
- Nodes can fail
- Messages can get lost
- Messages can interfere/collide

Protocols

NICTA

Reactive protocols

- Create routing information as needed
- Examples are LUNAR and DYMO

Proactive

- Maintain routing information for later use.
- Examples are AODV, DSR, and OLSR

NICTA

Correctness

"We must not put mistakes into programs because of sloppiness, we have to do it systematically and with care."
(Edsger Dijkstra)

Correctness

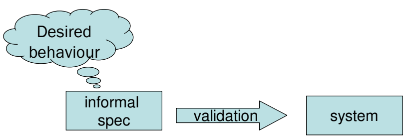
NICTA

Definition 1

- A system is correct if it cannot exhibit undesirable behaviour

Definition 2

- A system is correct if it exhibits only permissible behaviour



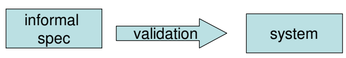
Correctness

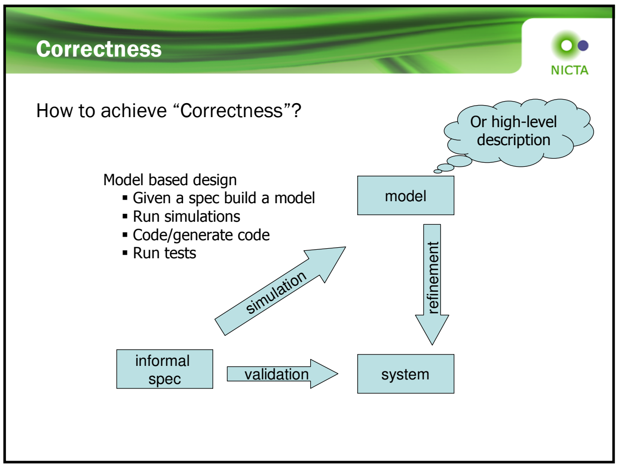
NICTA

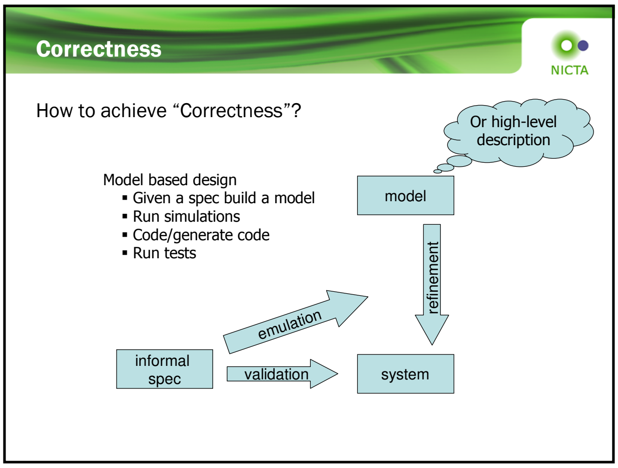
How to achieve "Correctness"?

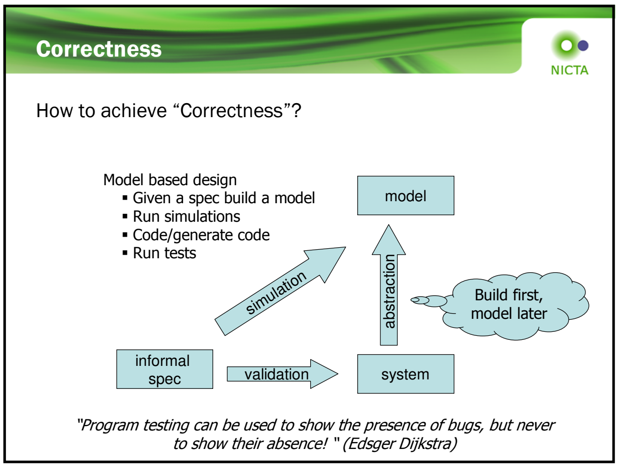
"Traditional" software engineering practice

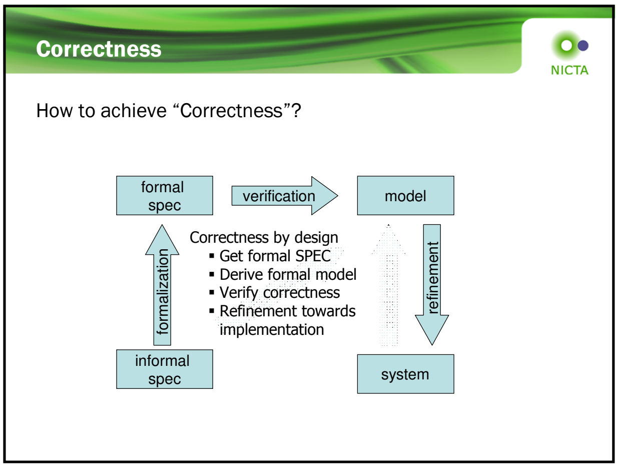
- Given a spec start coding
- Run test cases
- Code review
- Run more tests




Correctness
NICTA
How to achieve "Correctness"?
Or high-level description
Model based design
▪ Given a spec build a model
▪ Run simulations
▪ Code/generate code
▪ Run tests
model
simulation
refinement
informal spec
validation
system

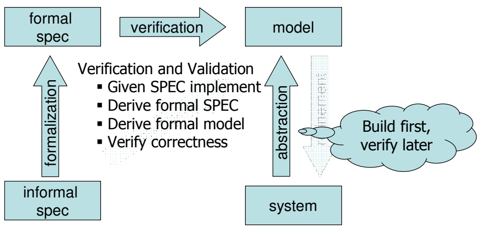

Correctness
NICTA
How to achieve "Correctness"?
Or high-level description
Model based design
▪ Given a spec build a model
▪ Run simulations
▪ Code/generate code
▪ Run tests
model
emulation
refinement
informal spec
validation
system


Correctness
NICTA
How to achieve "Correctness"?
Model based design
▪ Given a spec build a model
▪ Run simulations
▪ Code/generate code
▪ Run tests
model
simulation
abstraction
Build first, model later
informal spec
validation
system
"Program testing can be used to show the presence of bugs, but never to show their absence! " (Edsger Dijkstra)


Correctness
NICTA
How to achieve "Correctness"?
formal spec
verification
model
Correctness by design
▪ Get formal SPEC
▪ Derive formal model
▪ Verify correctness
▪ Refinement towards implementation
formalization
refinement
informal spec
system

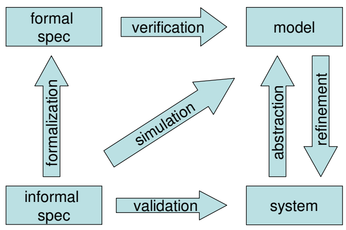
Correctness

Verification



Algorithmic Verification

How to achieve “Correctness”?



Creating Correct Network Protocols

Thesis Oskar Wibling

Thesis

Content and contributions

- Cross-platform protocol development
- Structured Live Testing
- Automata based protocol verification
- Graph-transformation system based (protocol) verification

Cross-Platform Protocol Development

LUNAR

- Lightweight Underlay Network Ad hoc Routing
- Discovers paths as needed
- Active paths are maintained
- Uses Propagating Localized Broadcast with Dampening (PLBD)
- Cross-platform implementation for Windows and Linux

Cross-Platform Protocol Development

Given

- User space implementation for Linux

Aim

- Kernel implementation for Windows and Linux

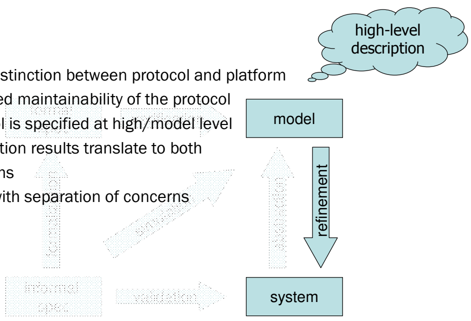
Approach

- Separate protocol logic from network and platform specific details
- Provide Windows versions of Linux kernel function calls

Cross-Platform Protocol Development

Benefits

- Clear distinction between protocol and platform
- Improved maintainability of the protocol
- Protocol is specified at high/model level
- Verification results translate to both platforms
- Helps with separation of concerns



Structured Live Testing

Comparative study

- Three different protocols: AODV, DSR and OLSR
- Three different evaluation methods: Simulation, emulation, real world testing
- Three different scenarios: End node swap, Relay node swap, Roaming node

=> Identified three ad hoc routing protocol problems: TCP backlash, Self Interference and Link cache poisoning

Structured Live Testing

NICTA

Simulation

- Simulating the protocol with ns-2
- No hardware
- Radio is simulated
- Mobility of nodes is simulated

Structured Live Testing

NICTA

Emulation

- Emulating the protocol using the APE testbed on identically configured laptops
- Stationary setup
- Uses actual radio and hardware
- Mobility is emulated using MAC filters
- Useful to study radio propagation effects when compared to simulation

Structured Live Testing

NICTA

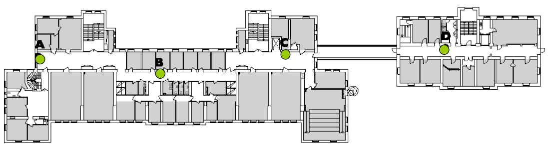
Real World Testing

- Running the protocol using the APE testbed on identically configured laptops
- Uses actual radio and hardware
- Mobility is achieved by humans carrying laptops
- To ensure repeatability carefully choreographed and scripted

Structured Live Testing

NICTA

Scenarios

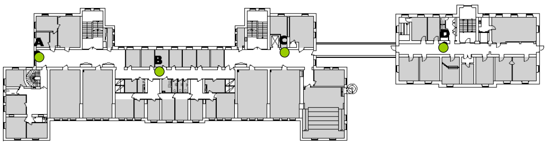


1. End node swap

Structured Live Testing

NICTA

Scenarios

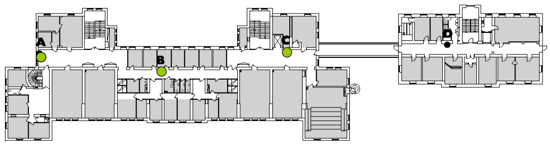


1. End node swap
2. Relay node swap

Structured Live Testing

NICTA

Scenarios



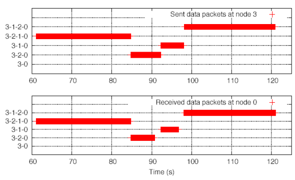
1. End node swap
2. Relay node swap
3. Roaming node

Structured Live Testing

NICTA

Results

- Comparing Simulation and Real-World points to sources for routing problems



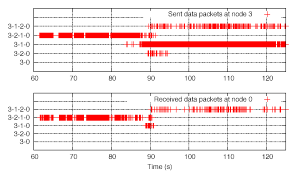
- Simulation for relay swap and DSR

Structured Live Testing

NICTA

Results

- Comparing Simulation and Real-World points to sources for routing problems



Link cache poisoning

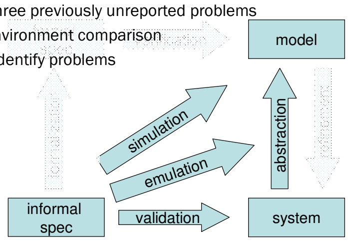
- Real world result for relay swap and DSR

Structured Live Testing

NICTA

Summary

- Used three different approaches to compare protocls
- Found three previously unreported problems
- Cross-environment comparison help to identify problems



Automata-based protocol verification

NICTA

Verification of LUNAR using SPIN and Uppaal

- Study protocol for network with finite number of nodes
- Subject to changes in topology.
- Correctness defined as guarantee that (1) the route will be set up and (2) the initial packet will be delivered
- Use time model in Uppaal to derive upper bounds for initial packet delivery

Automata-based protocol verification

NICTA

LUNAR

- The sender sends out a route request with Propagating Localized Broadcast with Dampening (PLBD)
  1. The initiating node tags the broadcast message with a unique ID
  2. Nodes ignore packets that they have received before
  3. Otherwise, if the node is not the destination, it will propagate the broadcast message.
- Once the destination node receives the request, it will send a unicast route reply along the discovered path.
- If the initiator receive the route reply it starts sending along the discovered path

Automata-based protocol verification

NICTA

Correctness property

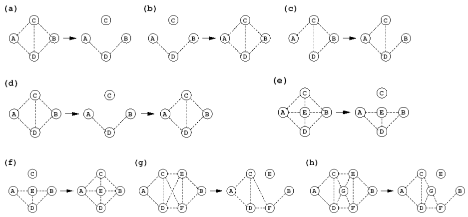
- If there at one point in time exists a path between two nodes, then the protocol must be able to find some path between the nodes.
- When a path has been found, it is possible to send packets along the path from the source node to the destination node, as long as the path remains valid.

Automata-based protocol verification

NICTA

Changes in topology

Prove that the protocol is resilient to changes in topology, due to link and/or node failure.



Automata-based protocol verification

NICTA

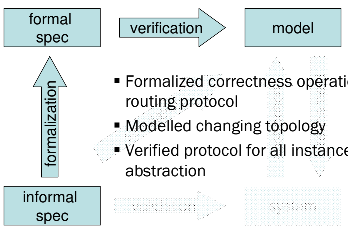
Broadcast Abstraction

- Improving the performance of model-checking by modelling PLBD as primitive operation, discarding many intermediate states and interleaving.
- Proving that the so-called “broadcast abstraction” is sound by provided
  - There exists a PLBD path
  - The PLBD path is unique
- Paper 3 gives proof that this is the case.

Automata-based protocol verification

NICTA

Summary



- Formalized correctness operation of an ad hoc routing protocol
- Modelled changing topology
- Verified protocol for all instances after broadcast abstraction

Graph Transformation System Verification

NICTA

Verification of DYMO and Heap operations using GBT

- A technique for modelling and verification based on graph transformation systems
- System configurations are modelled as hypergraphs
- Actions are modelled as graph rewrite rules
- Specification modelled as patterns
- Use backward reachability semi-algorithm to prove correctness
- Implemented as tool GBT

Graph Transformation System Verification

NICTA

Hypergraphs

- A hypergraph is a set of nodes with a set of hyperedges
- A hyperedge is a pair of an action label and an ordered tuple of nodes.

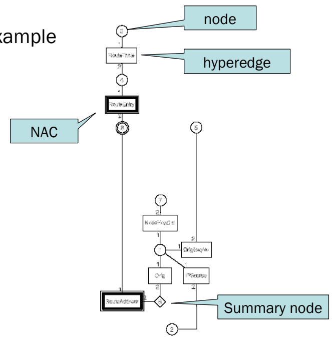
Patterns

- A pattern is a hypergraph, and represents all hypergraphs that have it as a subgraph.
- A pattern may include *negative application conditions,* which exclude all hypergraphs that have it as subgraph
- Introduction of *summary nodes*, to represent a non-empty set of nodes that have the same node type.

Graph Transformation System Verification

NICTA

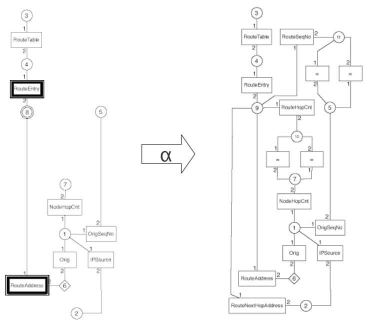
Example



Graph Transformation System Verification

NICTA

Example



Graph Transformation System Verification

NICTA

Backward Reachability

- Given a pattern representing all bad configurations (e.g. networks with loops)
- Compute the predecessor patterns, given all actions.
- Check if predecessor pattern is subsumed by a previously explored pattern.
- Stop if the initial configuration matches a predecessor pattern => Bad configurations are reachable.
- Stop if reachability analysis reaches a fix-point, i.e. find no new patterns => Bad configurations not reachable

Graph Transformation System Verification

NICTA

Verification Results

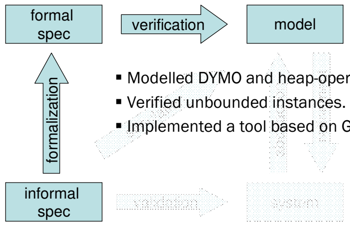
- Used the tool GBT to verify that the protocol DYMO guarantees absence of routing loops
- Verification took less than an hour
- Result holds for a network with an arbitrary number of network nodes.

- Verified the correctness of a heap-operation
- Made possible by introduction of summary nodes
- Verification took less than 20 minutes.
- Demonstrates the genral use of verification via Graph Transformation Systems

Graph Transformation System Verification

NICTA

Summary



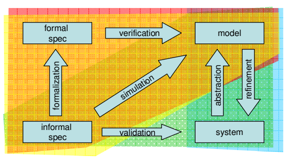
- Modelled DYMO and heap-operations as GTS
- Verified unbounded instances.
- Implemented a tool based on GTS

Summary

NICTA

The thesis achieved the following

- Cross-platform implementation of the LUNAR protocol
- Structured testing of 3 routing protocols
- Verification of bounded instances of a routing protocol using existing tools
- Developed a new tool to verify unbounded instances.



NICTA

Questions

Re: Verification results of paper II, III, V, VI.

Re: Correctness property in Definition 1, paper II and III.

Informal spec

- If there at one point in time exists a path between two nodes, then the protocol must be able to find some path between the nodes.
- When a path has been found, it is possible to send packets along the path from the source node to the destination node, as long as the path remains valid.

Formal spec

- A <> Lunar0.unic_rrep_rec
- A <> Lunar1.ip_rec_ok

Re: Results for GBT, table 5.3. p83.

Re: Correctness property for DYMO as hypergraph

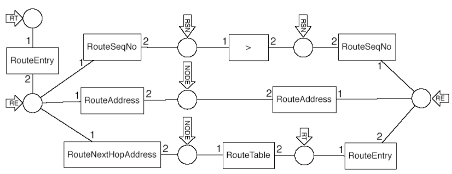


Figure 5.10: Pattern, φ, with node labels added.

Re: Paper IV, p 8. “The real world experiments suffer from (...) logging”.

Re: Page 40, Model checking. Classification of SPIN

Re: Paper III, p 3, “When using PLBD, the only possible paths (...) are disjoint.”

Re: p 90, paper VI, p15, CEGAR for GTS.

Re: Gap between simulation and real world experiment, p 57

Re: S/W development, p 25

Re: Impact of Network failure, p 15. “Driving to work or school”

Thanks