Using Process Algebra to Design Better Protocols

Peter Höfner

Why Better Protocols are Needed

- Routing Protocols are Broken

Why Better Protocols are Needed

- Routing Protocols are Broken
  - *Routing Protocols establish* ***non-optimal routes***

Routing Primitives for Wireless Mesh Networks: Design, Analysis and Experiments

Stanislav Miskovic and Edward W. Knightly

Why Better Protocols are Needed

- Routing Protocols are Broken
  - *Routing Protocols establish* ***non-optimal routes***
  - *AODV Routing Protocol sends packets in* ***loops***

Sequence Numbers Do Not Guarantee Loop Freedom —AODV Can Yield Routing Loops—

Rob van Glabbeek
NICTA, Australia
University of New South Wales, Australia
rvg@cs.stanford.edu

Peter Höfner
NICTA, Australia
University of New South Wales, Australia
Peter.Hoefner@nicta.com.au

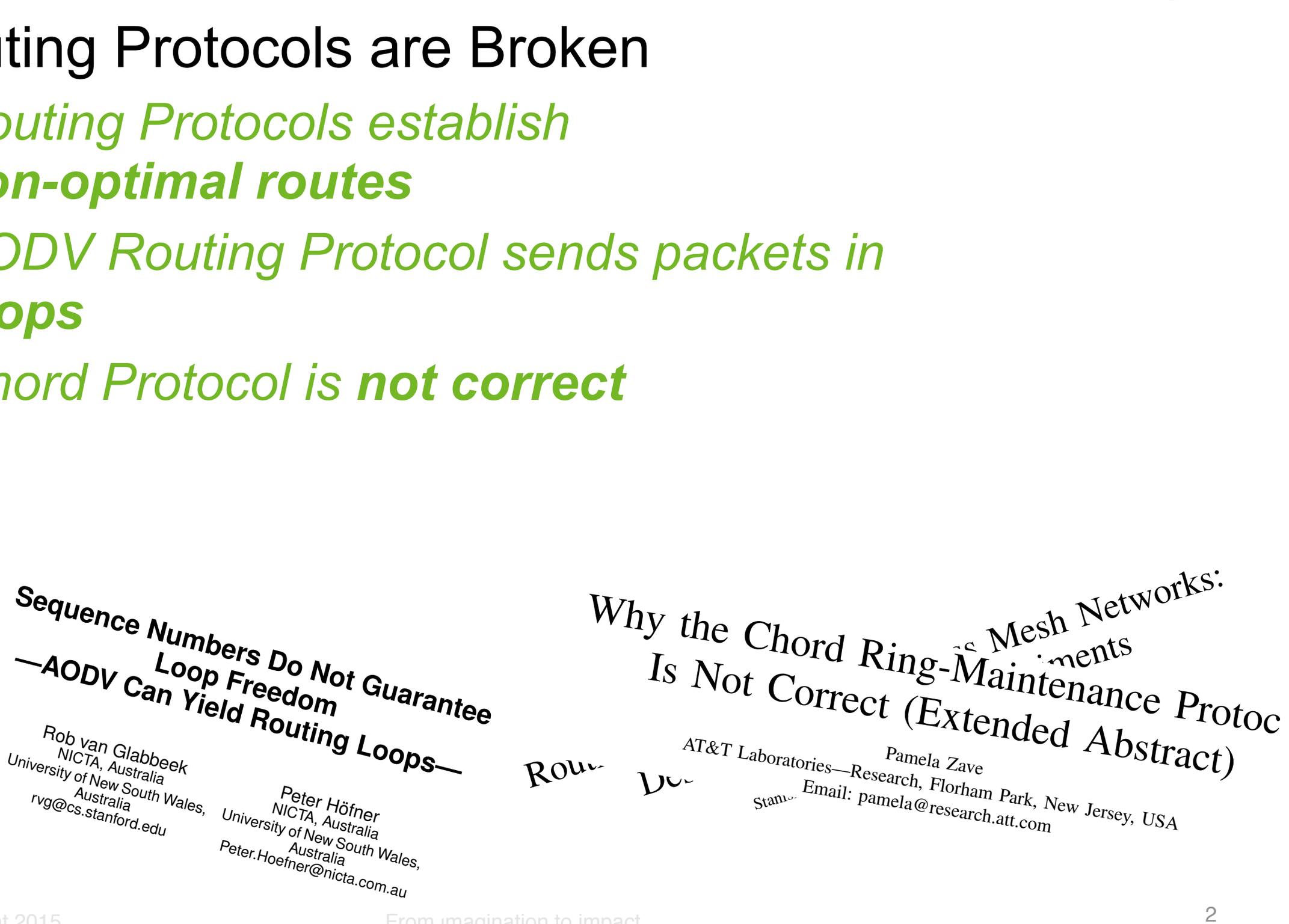
Routing Primitives for Wireless Mesh Networks: Design, Analysis and Experiments

Stanislav Miskovic and Edward W. Knightly

Why Better Protocols are Needed

NICTA

- Routing Protocols are Broken
  - *Routing Protocols establish* ***non-optimal routes***
  - *AODV Routing Protocol sends packets in* ***loops***
  - *Chord Protocol is* ***not correct***



Sequence Numbers Do Not Guarantee Loop Freedom —AODV Can Yield Routing Loops—

Rob van Glabbeek
NICTA, Australia
University of New South Wales, Australia
rvg@cs.stanford.edu

Peter Höfner
NICTA, Australia
University of New South Wales, Australia
Peter.Hoefner@nicta.com.au

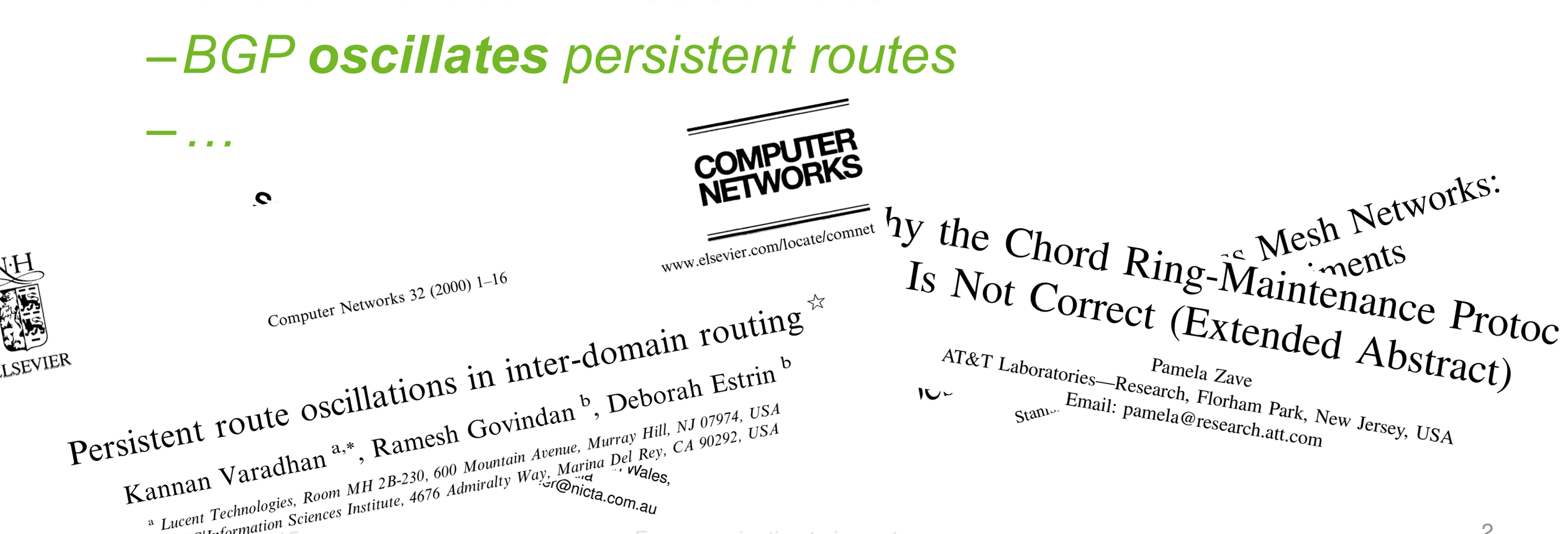
Why the Chord Ring-Maintenance Protoc Is Not Correct (Extended Abstract)

Pamela Zave
AT&T Laboratories—Research, Florham Park, New Jersey, USA
Email: pamela@research.att.com

Why Better Protocols are Needed

NICTA

- Routing Protocols are Broken
  - *Routing Protocols establish **non-optimal routes***
  - *AODV Routing Protocol sends packets in **loops***
  - *Chord Protocol is **not correct***
  - *BGP **oscillates** persistent routes*
  - *…*



Today's Protocol Development

- IETF: "Rough Consensus and Running Code" (Trial and Error)
  - start with a good idea
  - build a protocol out of it (implementation)
    - run tests (over several years)
      - find limitations, flaws, etc...
      - fix problems
    - build a new version of the protocol
    - at some point people agree on an RFC

Beauvais Cathedral
(~300 years to build, at least 2 collapses)

Better Protocols are Needed Now!

NICTA

- We cannot afford this approach
  - to expensive w.r.t. time
  - to expensive w.r.t. money
  - we are not working in a lab, i.e., sometimes we have one try only (e.g. BGP)
- Is there a method which is more reliable and cost efficient

The original design was so boldly conceived that it was found structurally impossible to build.

What's the Problem? (1)

NICTA

- Specifications are (excessively) long
  - the Session Initiation Protocol is 268 pages long (and not even self contained - by 2009 142 additional documents were required)
  - IEEE 802.11 is 2.793 pages long

What's the Problem? (2)

- Specifications are
  - underspecified
  - contradictory
  - erroneous, and
  - ambiguous

What's the Problem? (3)

NICTA

- Specifications are written in English Prose
  - in case of AODV there are 5 different implementations all compliant to the standard

What's the Problem? (3)

NICTA

- Specifications are written in English Prose
  - in case of AODV there are 5 different implementations all compliant to the standard

Aims

NICTA

- Provide complete and practical formal methods
  - expressive (mobility, dynamic topology, types of communication,…)
  - usable and intuitive
  - description language + proof methodology + automation
- Specification, verification and analysis of protocols
  - formalise relevant standard protocols
  - analyse the protocols w.r.t. key requirements
  - analyse compliant implementations
- Development of improved protocols
  - assured protocol correctness
  - improve reliability and performance

Developed Process Algebra

NICTA

- Description Language (Syntax)

$X(exp_1, \ldots, \exp_n)$	process calls
$P + Q$	nondeterministic choice
$[\varphi]P$	if-construct (guard)
$[\![\texttt{var} := exp]\!]P$	assignment followed by
$\mathbf{broadcast}(ms).P$	broadcast
$\mathbf{groupcast}(dests, ms).P$	groupcast
$\mathbf{unicast}(dest, ms).P \blacktriangleright Q$	unicast
$\mathbf{send}(ms).P$	send
$\mathbf{receive}(\texttt{msg}).P$	receive
$\mathbf{deliver}(data).P$	deliver

Developed Process Algebra

NICTA

- Description Language (Syntax)

$P \lll Q$	parallel operator on nodes

- Do we need more?

$[\varphi]P + [\neg\varphi]Q$	deterministic choice
$P(n) = [\![n := n + 1]\!].P(n)$	loops

Case Study: AODV

NICTA

```
+ [ (oip, rreqid) ∉ rreqs ]        /* the RREQ is new to this node */
   ⟦rt := update(rt,(oip,osn,kno,val,hops+1,sip,∅))⟧        /* update the route to oip in rt */
   ⟦rreqs := rreqs ∪ {(oip,rreqid)}⟧        /* update rreqs by adding (oip, rreqid) */
   (
      [ dip = ip ]        /* this node is the destination node */
         ⟦sn := max(sn,dsn)⟧        /* update the sqn of ip */
         /* unicast a RREP towards oip of the RREQ */
          unicast(nhop(rt,oip),rrep(0,dip,sn,oip,ip)) . AODV(ip,sn,rt,rreqs,store)
          ▶ /* If the transmission is unsuccessful, a RERR message is generated */
            ⟦dests := {(rip, inc(sqn(rt,rip))) | rip ∈ vD(rt) ∧ nhop(rt,rip) = nhop(rt,oip)}⟧
         ⟦rt := invalidate(rt,dests)⟧
         ⟦store := setRRF(store,dests)⟧
         ⟦pre := ⋃{precs(rt,rip) | (rip,*) ∈ dests}⟧
         ⟦dests := {(rip,rsn) | (rip,rsn) ∈ dests ∧ precs(rt,rip) ≠ ∅}⟧
         groupcast(pre,rerr(dests,ip)) . AODV(ip,sn,rt,rreqs,store)
   + [ dip ≠ ip ]        /* this node is not the destination node */
      (
         [dip ∈ vD(rt) ∧ dsn ≤ sqn(rt,dip) ∧ sqnf(rt,dip) = kno ]        /* valid route to dip that is fresh enough */
            /* update rt by adding precursors */
            ⟦rt := addpreRT(rt,dip,{sip})⟧
            ⟦rt := addpreRT(rt,oip,{nhop(rt,dip)})⟧
            /* unicast a RREP towards the oip of the RREQ */
             unicast(nhop(rt,oip),rrep(dhops(rt,dip),dip,sqn(rt,dip),oip,ip)) .
```

Developed Process Algebra

NICTA

- Semantics
  - not used by a software engineer
  - internal state determined by expression and valuation

$$\xi, \mathbf{broadcast}(ms).p \xrightarrow{\mathbf{broadcast}(\xi(ms))} \xi, p$$

$$\xi, \mathbf{groupcast}(dests, ms).p \xrightarrow{\mathbf{groupcast}(\xi(dests),\xi(ms))} \xi, p$$

$$\xi, \mathbf{unicast}(dest, ms).p \blacktriangleright q \xrightarrow{\mathbf{unicast}(\xi(dest),\xi(ms))} \xi, p$$

$$\xi, \mathbf{unicast}(dest, ms).p \blacktriangleright q \xrightarrow{\neg\mathbf{unicast}(\xi(dest))} \xi, q$$

$$\xi, \mathbf{send}(ms).p \xrightarrow{\mathbf{send}(\xi(ms))} \xi, p$$

$$\xi, \mathbf{deliver}(data).p \xrightarrow{\mathbf{deliver}(\xi(data))} \xi, p$$

$$\xi, \mathbf{receive}(\mathtt{msg}).p \xrightarrow{\mathbf{receive}(m)} \xi[\mathtt{msg} := m], p \qquad (\forall m \in \mathtt{MSG})$$

Developed Process Algebra

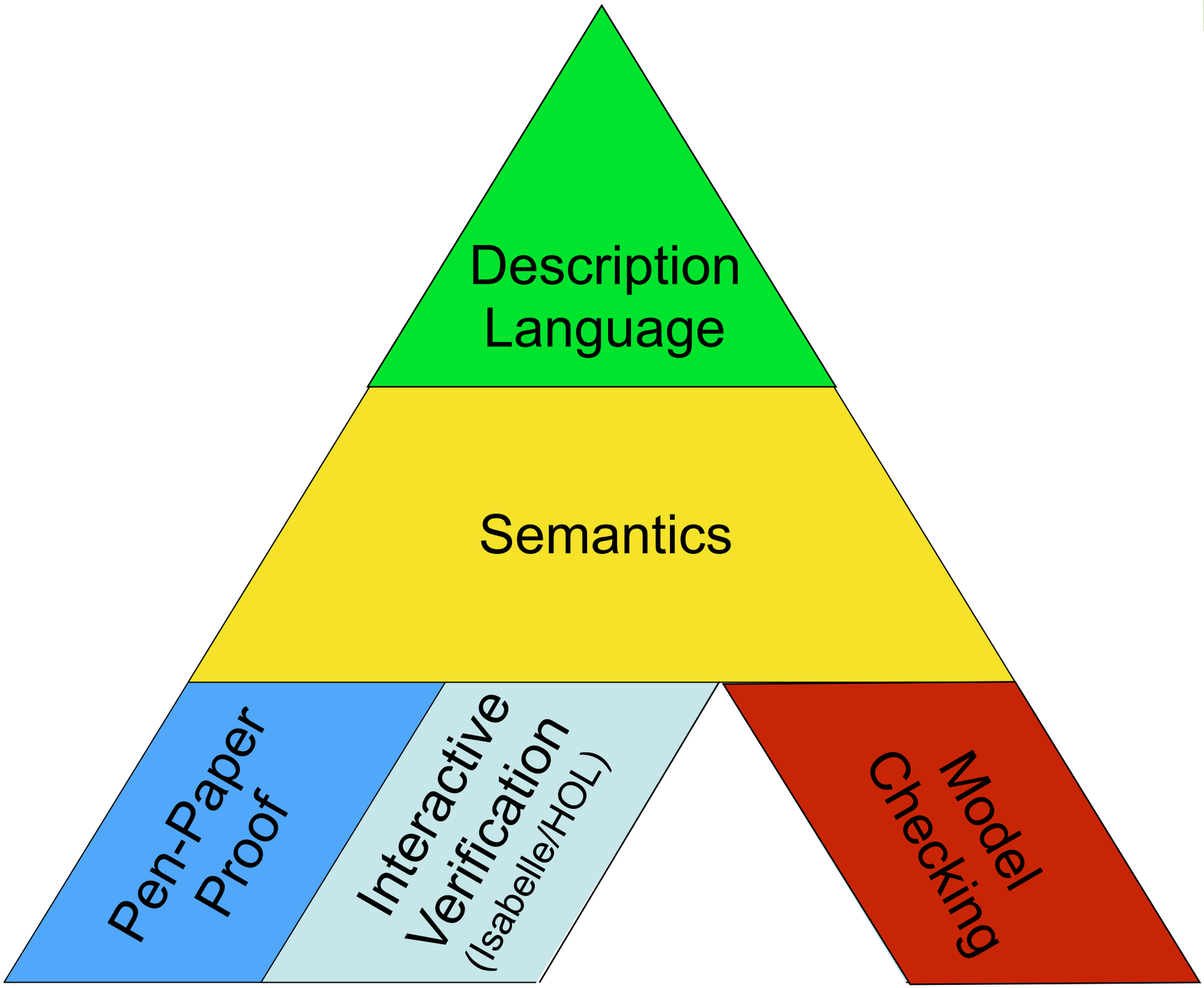
- Semantics cont'd

$$\frac{P \xrightarrow{a} P'}{P \langle\!\langle Q \xrightarrow{a} P' \langle\!\langle Q} \quad (\forall a \neq \mathbf{receive}(m))$$

$$\frac{Q \xrightarrow{a} Q'}{P \langle\!\langle Q \xrightarrow{a} P \langle\!\langle Q'} \quad (\forall a \neq \mathbf{send}(m))$$

$$\frac{P \xrightarrow{\mathbf{receive}(m)} P' \qquad Q \xrightarrow{\mathbf{send}(m)} Q'}{P \langle\!\langle Q \xrightarrow{\tau} P' \langle\!\langle Q'} \quad (\forall m \in \mathtt{MSG})$$

Backbone Support

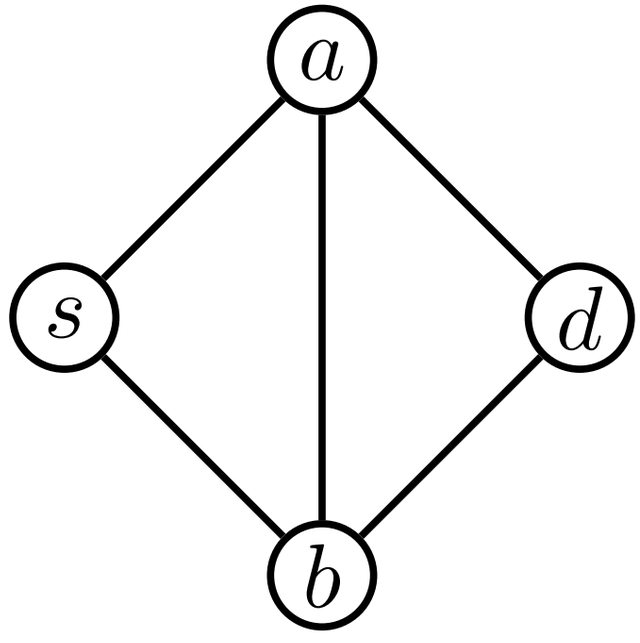


Case Study: AODV

- Ad Hoc On-Demand Distance Vector Protocol
  - routing protocol for wireless mesh networks (wireless networks without wired backbone)
  - Ad hoc (network is not static)
  - On-Demand (routes are established when needed)
  - Distance (metric is hop count)
  - developed 1997-2001 by Perkins, Beldig-Royer and Das (University of Cincinnati)
  - one of the four protocols standardised by the IETF MANET working group (IEEE 802.11s)

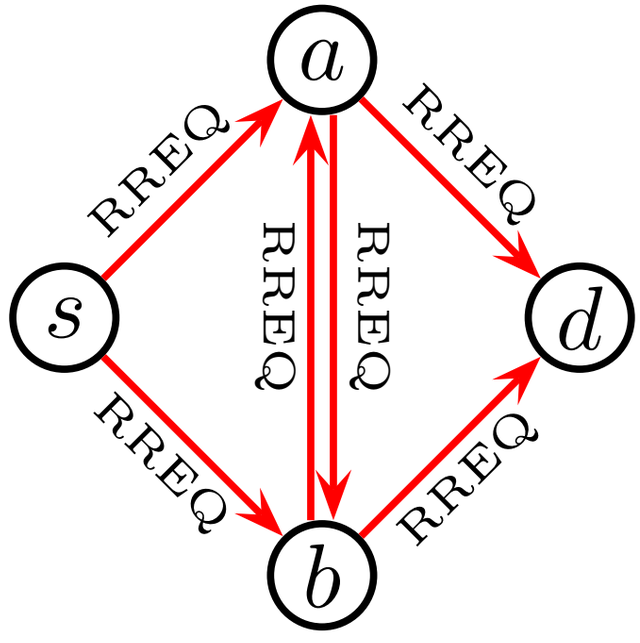
Case Study

- Main Mechanism
  - if route is needed
    BROADCAST RREQ
  - if node has information about a destination
    UNICAST RREP
  - if unicast fails or link break is detected
    GROUPCAST RERR
  - performance improvement via intermediate route reply



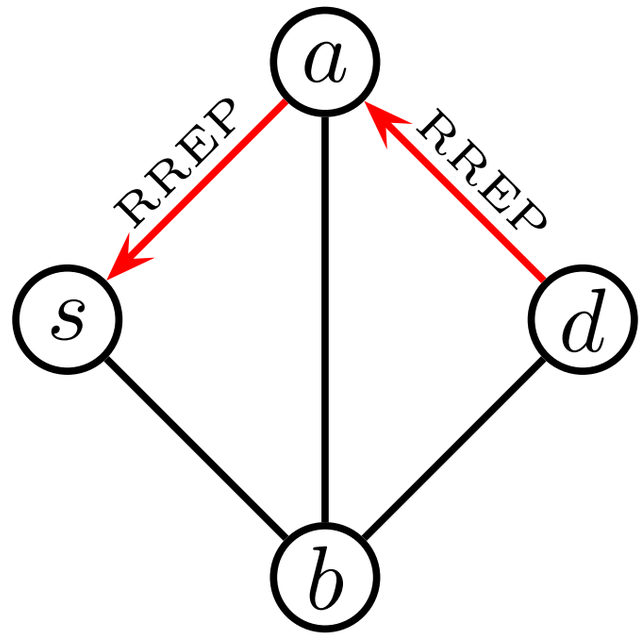
Case Study

- Main Mechanism
  - if route is needed
    BROADCAST RREQ
  - if node has information about a destination
    UNICAST RREP
  - if unicast fails or link break is detected
    GROUPCAST RERR
  - performance improvement via intermediate route reply



Case Study

- Main Mechanism
  - if route is needed
    BROADCAST RREQ
  - if node has information about a destination
    UNICAST RREP
  - if unicast fails or link break is detected
    GROUPCAST RERR
  - performance improvement via intermediate route reply



Case Study: AODV

NICTA

$+$ [$(\texttt{oip},\texttt{rreqid}) \notin \texttt{rreqs}$] /* the RREQ is new to this node */
⟦$\texttt{rt} := \texttt{update}(\texttt{rt},(\texttt{oip},\texttt{osn},\texttt{kno},\texttt{val},\texttt{hops}+1,\texttt{sip},\emptyset))$⟧ /* update the route to oip in rt */
⟦$\texttt{rreqs} := \texttt{rreqs} \cup \{(\texttt{oip},\texttt{rreqid})\}$⟧ /* update rreqs by adding (oip, rreqid) */
(
[$\texttt{dip} = \texttt{ip}$] /* this node is the destination node */
⟦$\texttt{sn} := \max(\texttt{sn},\texttt{dsn})$⟧ /* update the sqn of ip */
/* unicast a RREP towards oip of the RREQ */
unicast(nhop(rt,oip),rrep(0,dip,sn,oip,ip)) . AODV(ip,sn,rt,rreqs,store)
▶ /* If the transmission is unsuccessful, a RERR message is generated */
⟦$\texttt{dests} := \{(\texttt{rip},\texttt{inc}(\texttt{sqn}(\texttt{rt},\texttt{rip}))) \mid \texttt{rip} \in \texttt{vD}(\texttt{rt}) \wedge \texttt{nhop}(\texttt{rt},\texttt{rip}) = \texttt{nhop}(\texttt{rt},\texttt{oip})\}$⟧
⟦$\texttt{rt} := \texttt{invalidate}(\texttt{rt},\texttt{dests})$⟧
⟦$\texttt{store} := \texttt{setRRF}(\texttt{store},\texttt{dests})$⟧
⟦$\texttt{pre} := \bigcup\{\texttt{precs}(\texttt{rt},\texttt{rip}) \mid (\texttt{rip},*) \in \texttt{dests}\}$⟧
⟦$\texttt{dests} := \{(\texttt{rip},\texttt{rsn}) \mid (\texttt{rip},\texttt{rsn}) \in \texttt{dests} \wedge \texttt{precs}(\texttt{rt},\texttt{rip}) \neq \emptyset\}$⟧
groupcast(pre,rerr(dests,ip)) . AODV(ip,sn,rt,rreqs,store)
$+$ [$\texttt{dip} \neq \texttt{ip}$] /* this node is not the destination node */
(
[$\texttt{dip} \in \texttt{vD}(\texttt{rt}) \wedge \texttt{dsn} \leq \texttt{sqn}(\texttt{rt},\texttt{dip}) \wedge \texttt{sqnf}(\texttt{rt},\texttt{dip}) = \texttt{kno}$] /* valid route to dip that is fresh enough */
/* update rt by adding precursors */
⟦$\texttt{rt} := \texttt{addpreRT}(\texttt{rt},\texttt{dip},\{\texttt{sip}\})$⟧
⟦$\texttt{rt} := \texttt{addpreRT}(\texttt{rt},\texttt{oip},\{\texttt{nhop}(\texttt{rt},\texttt{dip})\})$⟧
/* unicast a RREP towards the oip of the RREQ */
unicast(nhop(rt,oip),rrep(dhops(rt,dip),dip,sqn(rt,dip),oip,ip)) .

Case Study: AODV

- full specification of AODV (IETF Standard)
- specification details
  - around 5 types and 30 functions
  - around 120 lines of specification
    (in contrast to 40 pages English prose)

Case Study: AODV - Analysis

- *Properties of AODV*
  - *route correctness*
  - *loop freedom*
  - *route discovery*
  - *packet delivery*

Case Study: AODV - Analysis

- *Properties of AODV*
  - *route correctness* ✓
  - *loop freedom* ✓ (at least for some interpretations)
  - *route discovery* ✗
  - *packet delivery* ✗

Case Study: Analysis

- Loop Freedom
  - invariant proof
    based on about 35 invariants, e.g.

If a route reply is sent by a node ip_c, different from the destination of the route, then the content of ip_c's routing table must be consistent with the information inside the message.

$$N \xrightarrow{R:*\mathbf{cast}(\mathtt{rrep}(hops_c,dip_c,dsn_c,*,ip_c))}_{ip} N' \ \wedge\ ip_c \neq dip_c$$
$$\Rightarrow\quad dip_c \in \mathtt{kD}_N^{ip_c} \ \wedge\ \mathtt{sqn}_N^{ip_c}(dip_c) = dsn_c \ \wedge\ \mathtt{dhops}_N^{ip_c}(dip_c) = hops_c \ \wedge\ \mathtt{flag}_N^{ip_c}(dip_c) = \mathtt{val}$$

  - ultimately we defined quality on routes
    the quality strictly increases

$$dip \in \mathtt{vD}_N^{ip} \cap \mathtt{vD}_N^{nhip} \ \wedge\ nhip \neq dip \ \Rightarrow\ \xi_N^{ip}(\mathtt{rt}) \sqsubset_{dip} \xi_N^{nhip}(\mathtt{rt})$$

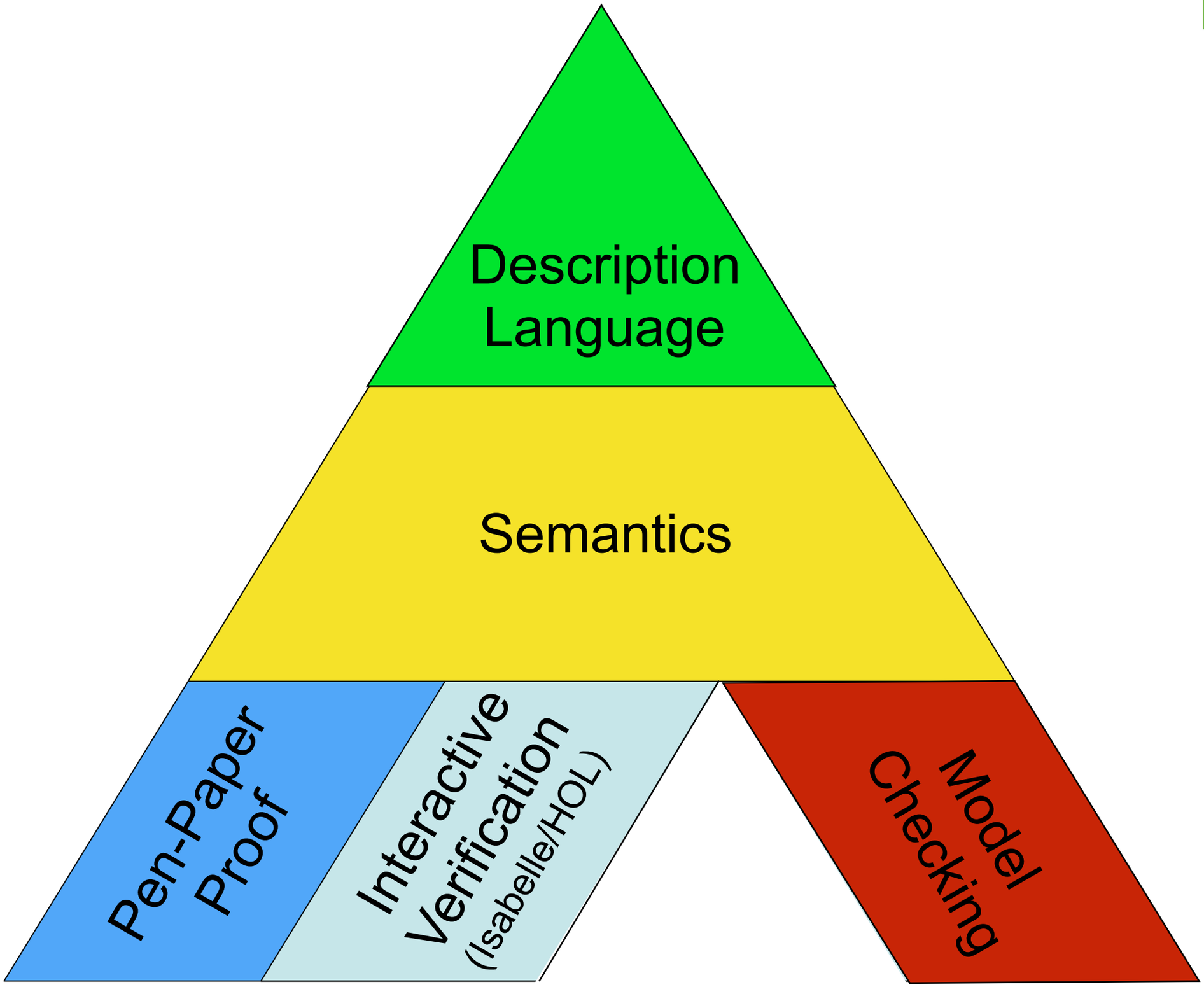
  - first rigorous and complete proof of loop freedom of AODV
    (for some interpretations)

Case Study: Analysis

- Loop Freedom
  - 5184 possible interpretations due to ambiguities
  - 5006 of these readings of the standard contain loops
  - 3 out of 5 open-source implementations contain loops

- Found other shortcomings
  - e.g. non-optimal routing information
  - we proposed solutions and proved them correct

Computer-Aided Verification



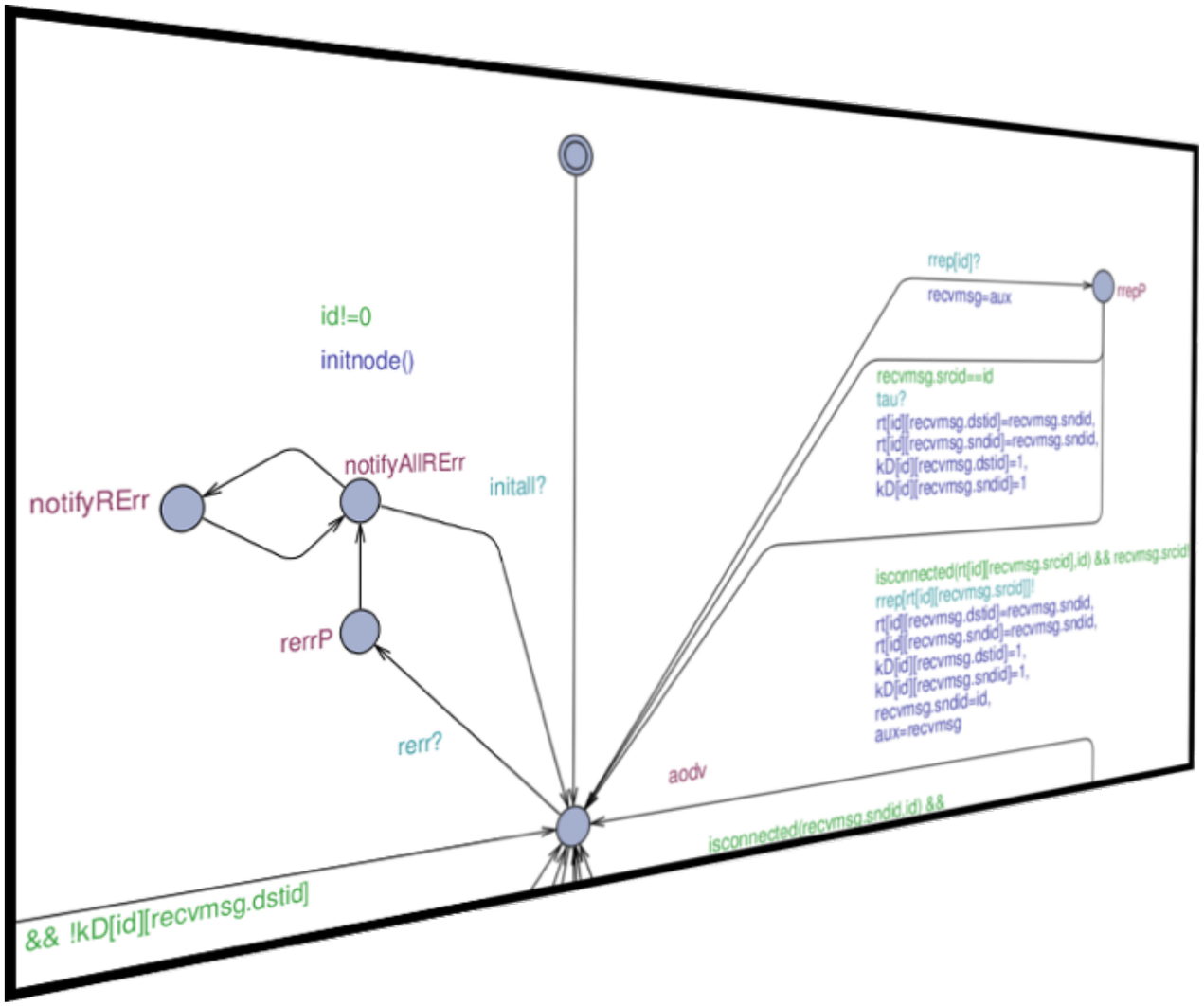
Computer-Aided Verification

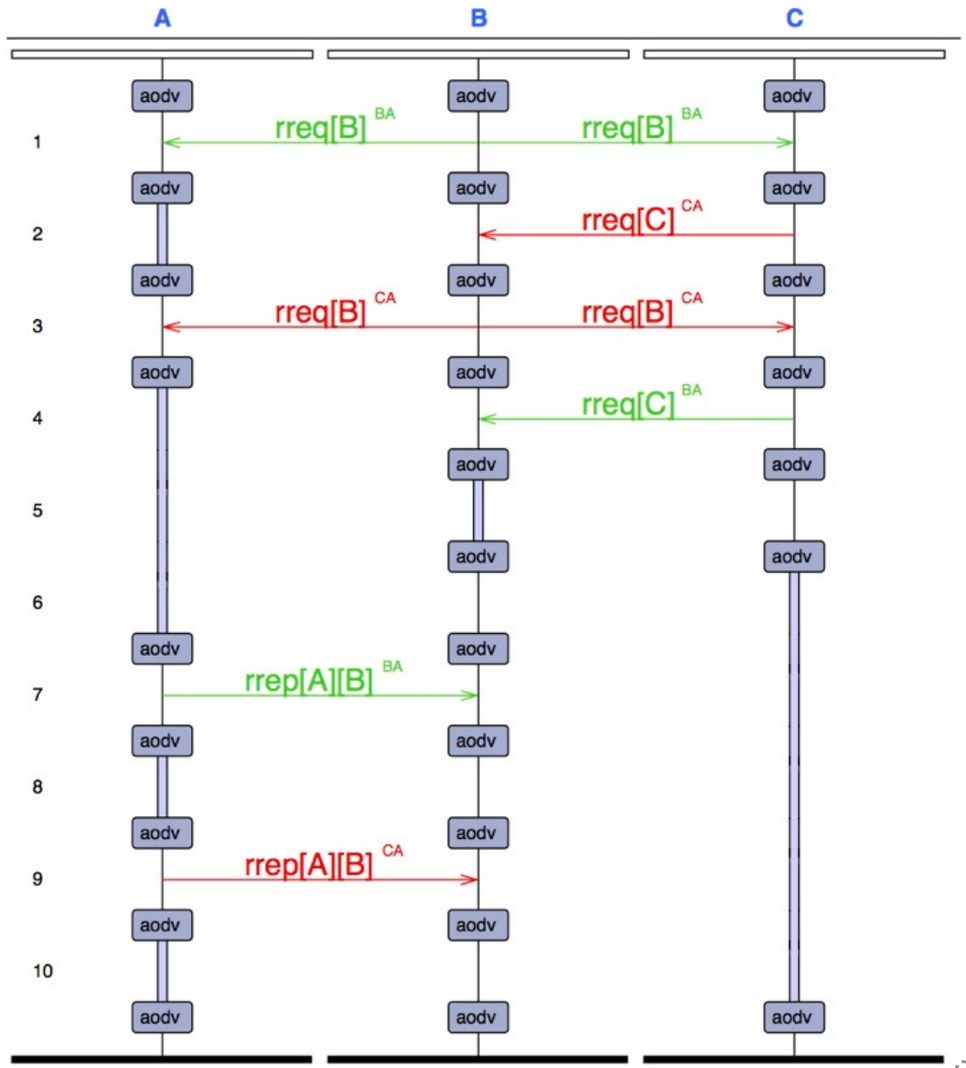
NICTA

- Model Checking
  - quick feedback for development
  - cannot be used for full verification

- (Interactive) Theorem Proving
  - Isabelle/HOL
    - replay proofs
      - proof verification
      - robust against small changes in specification

Model Checking





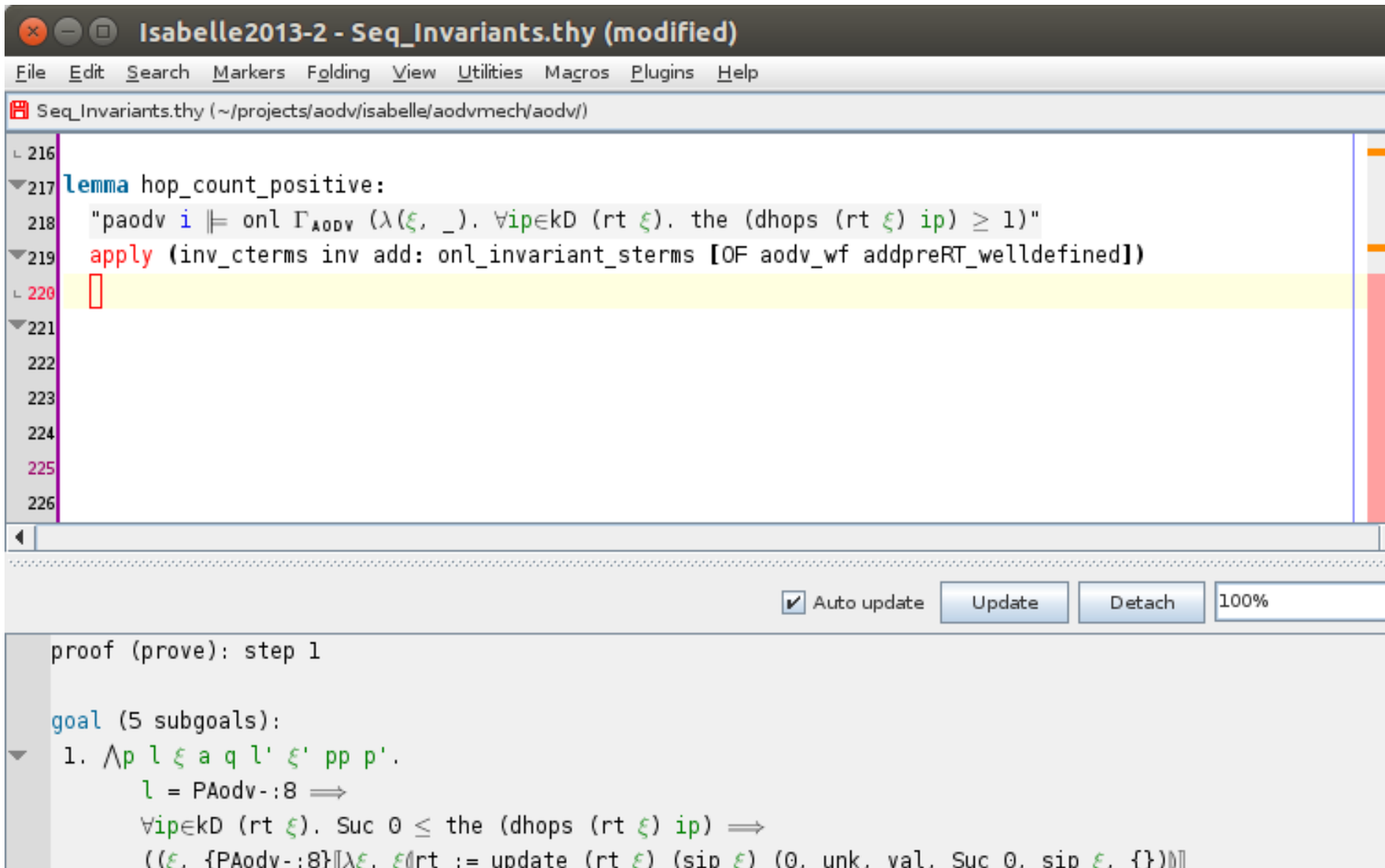
Model Checking

NICTA

- Model checking routing algorithms
  - executable models (generated from process-algebraic specification)
- Complementary to process algebra
  - find bugs and typos in process-algebraic model
  - check properties of specification applied to particular topology
  - easy adaption in case of change
  - automatic verification
- Achievements
  - implemented process algebra specification of AODV
  - found/replayed shortcomings

Isabelle/HOL

NICTA



Isabelle/HOL

NICTA

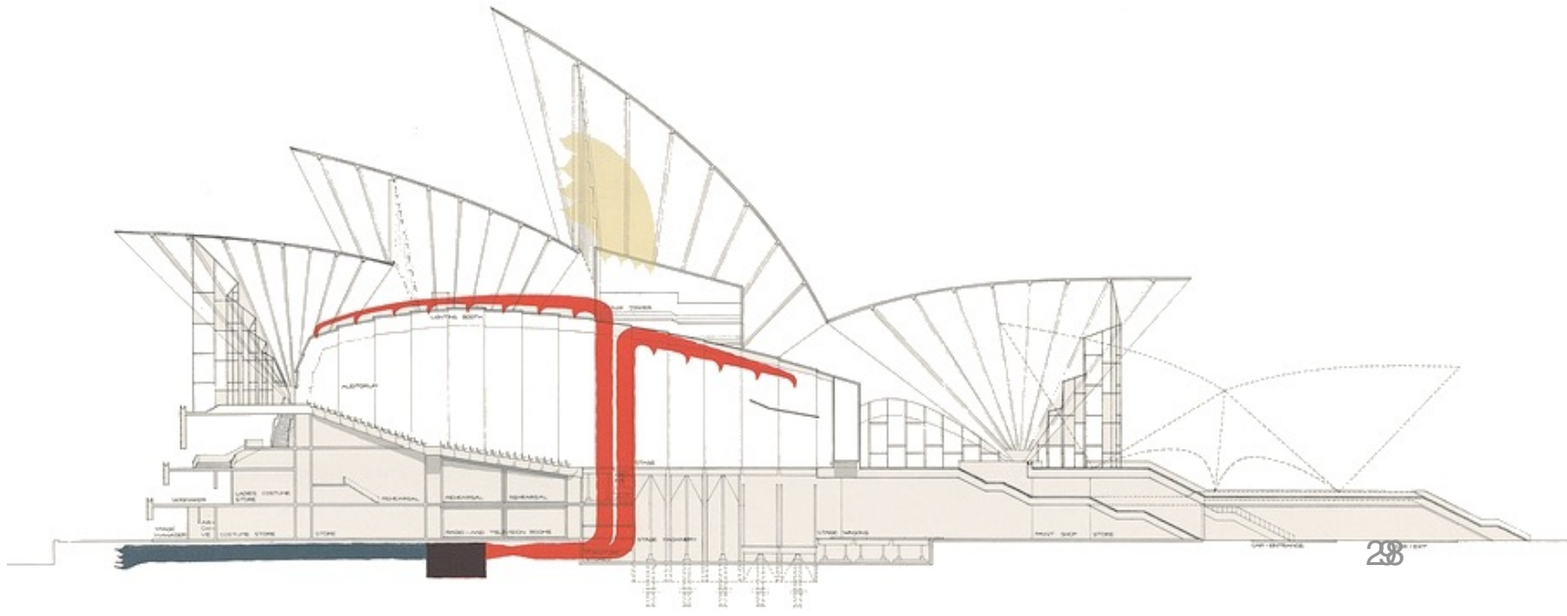
- Generic proof assistant
- We implemented
  - developed process algebra
  - AODV invariant proofs

- Advantages
  - proof verification
  - speed up of analysis of protocol variants
    - analysed variants/improvements more or less automatically
  - quick proof adaption
    - reply of proofs
    - necessary for protocol development

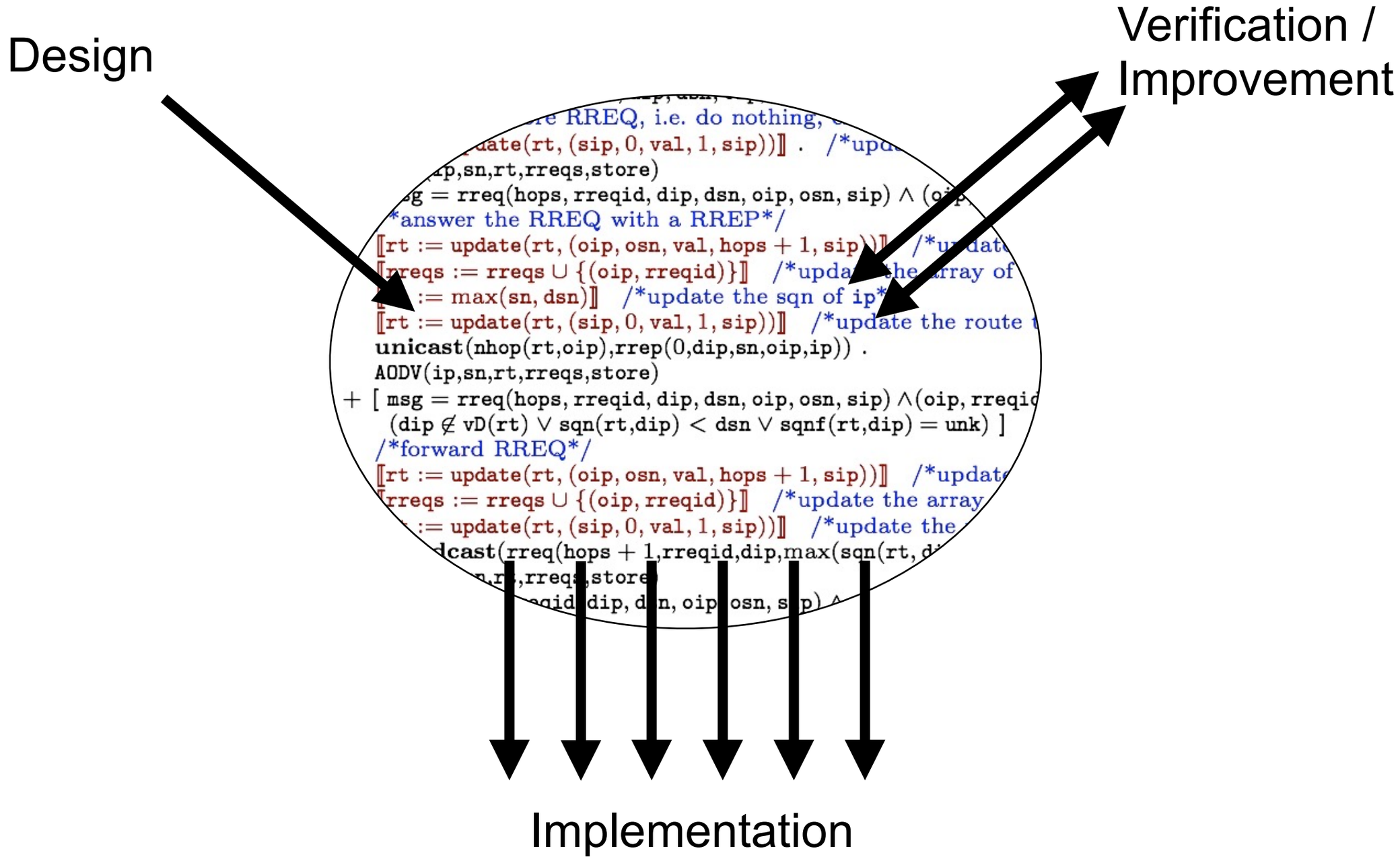
Key Research Outcomes

NICTA

- New languages and proof methodologies
  - process algebra
- Case Study AODV
  - complete and detailed model (without time)
  - model checking: quick check for counterexamples
  - theorem proving: verification and proof automation



Vision - Practical Protocol Engineering



Future Work

- Research (1)
  - timed analysis
  - build tool suite
  - better tool support (more proof automation)

- Research (2)
  - code generation
  - code verification

- Training
  - train network engineers to use our approach
  - hardest to achieve

Questions?

“Despite the maturity of formal description languages and formal methods for analyzing them, the description of real protocols is still overwhelmingly informal. The consequences of informal protocol description drag down industrial productivity and impede research progress”.

Pamela Zave (AT&T)