NICTA

Formal Methods for Wireless (Mesh) Protocols

Peter Höfner

03/04/2012

Australian Government
Department of Broadband, Communications and the Digital Economy
Australian Research Council

NICTA Funding and Supporting Members and Partners

Australian National University
UNSW THE UNIVERSITY OF NEW SOUTH WALES
NSW GOVERNMENT Trade & Investment
State Government Victoria
THE UNIVERSITY OF MELBOURNE
THE UNIVERSITY OF SYDNEY
Queensland Government
Griffith UNIVERSITY
QUT Queensland University of Technology
THE UNIVERSITY OF QUEENSLAND AUSTRALIA

Project Structure

- Formal Methods for Routing Protocols of Wireless Mesh Networks (WMNs)

- Part of "**Mesh Protocols**"
- Across research groups
  - close cooperation with Software Systems Research Group
- Across research labs
  - NRL, QRL

- start November 2010

Project Team

- Formal Methods for WMNs @ NRG
  - Annabelle McIver
  - Marius Portmann
  - Wee Lum Tan

- Formal Methods for WMNs @ SSRG
  - Rob van Glabbeek
  - Peter Höfner

- ~2.5 FTEs

Today's Protocol Development

NICTA

- "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)

Research Challenges

- Is there a method which is more reliable and cost-efficient?
- Is there a way to compare variants of protocols or different protocols?
- **New engineering methods required (or finetune/extend existing ones)**

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

Problems

- Standards (IETF RFCs) are not precise
  - written in English
  - ambiguous (sometimes incomplete)
  - no formal specification or reasoning
- Compliant implementations
  - have different behaviours
  - are not compatible
  - have serious flaws
- Traditional evaluation techniques: simulation, test-bed experiments
  - expensive, time-consuming
  - limited to (a small number of) specific scenarios
  - protocol errors still found even after years of intensive evaluation (e.g. [MiskovicKnightly10])
  - barely any guarantee for properties such as route discovery

Internet Engineering Task Force (IETF)

NICTA

- “Formal languages are useful tools for specifying parts of protocols. However, as of today, there exists no well-known language that is able to capture the full syntax and semantics of reasonably rich IETF protocols.“

  [IETF]

- IETF’s requirements (for formal languages)
  - relatively easy to extract code
  - complete specification
  - implementation independent

Research Aims

NICTA

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

Key Research Outcomes (Summary)

- New languages and proof methodologies
  - process algebra AWN
  - routing algebra
- Modelling of AODV
  - process algebra: complete and detailed model (without time)
  - model checking: encoding of AWN specification
  - routing algebra: modelled parts of AODV
- Analysing/Verifying AODV
  - process algebra: proof methodology, invariant proofs
  - model checking: automatic finding of problematic behaviour e.g., no route discovery guarantee
  - analysed (all interpretations of) AODV

Formalisation of AODV

NICTA

Table 1 Excerpt of AWN spec for AODV. A few cases for RREQ handling.

```
AODV(ip,sn,rt,rreqs,store) ≝
/*depending on the message on top of the message queue, the node calls different processes*/
...
[ msg = rreq(hops, rreqid, dip, dsn, oip, osn, sip) ∧ (oip, rreqid) ∈ rreqs ]
    /*silently ignore RREQ, i.e. do nothing, except update the entry for the sender*/
    ⟦rt := update(rt, (sip, 0, val, 1, sip))⟧ .   /*update the route to sip*/
    AODV(ip,sn,rt,rreqs,store)
+ [ msg = rreq(hops, rreqid, dip, dsn, oip, osn, sip) ∧ (oip, rreqid) ∉ rreqs) ∧ dip = ip ]
    /*answer the RREQ with a RREP*/
    ⟦rt := update(rt, (oip, osn, val, hops + 1, sip))⟧   /*update the routing table*/
    ⟦rreqs := rreqs ∪ {(oip, rreqid)}⟧   /*update the array of already seen RREQ*/
    ⟦sn := max(sn, dsn)⟧   /*update the sqn of ip*/
    ⟦rt := update(rt, (sip, 0, val, 1, sip))⟧   /*update the route to sip*/
    unicast(nhop(rt,oip),rrep(0,dip,sn,oip,ip)) .
    AODV(ip,sn,rt,rreqs,store)
+ [ msg = rreq(hops, rreqid, dip, dsn, oip, osn, sip) ∧(oip, rreqid) ∉ rreqs) ∧ dip ≠ ip ∧
      (dip ∉ vD(rt) ∨ sqn(rt,dip) < dsn ∨ sqnf(rt,dip) = unk) ]
    /*forward RREQ*/
    ⟦rt := update(rt, (oip, osn, val, hops + 1, sip))⟧   /*update routing table*/
    ⟦rreqs := rreqs ∪ {(oip, rreqid)}⟧   /*update the array of already seen RREQ*/
    ⟦rt := update(rt, (sip, 0, val, 1, sip))⟧   /*update the route to the sender*/
    broadcast(rreq(hops + 1,rreqid,dip,max(sqn(rt, dip), dsn),oip,osn,ip)) .
    AODV(ip,sn,rt,rreqs,store)
+ [ rreq(hops, rreqid, dip, dsn, oip, osn, sip) ∧ ... ]
    ...
```

Loop Freedom

NICTA

- Idea (Common belief):
  Sequence numbers guarantee loop freedom if increased monotonically

- Case study: AODV (Ad hoc On-demand Distance Vector) routing protocol
  – RFC 3561:

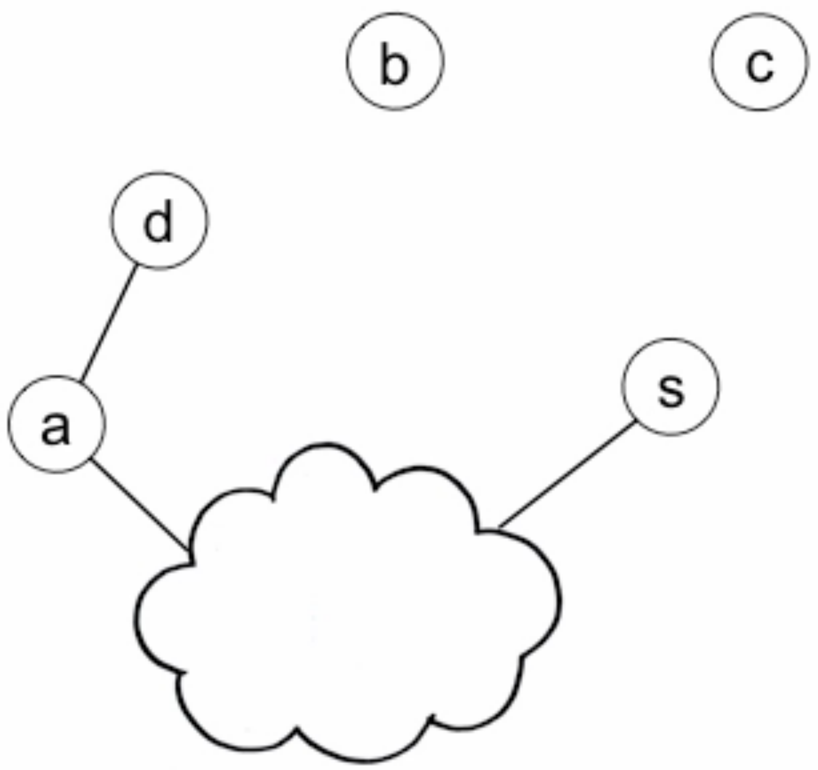
    “One distinguishing feature of AODV is its use of a destination sequence number for each route entry. The destination sequence number is created by the destination to be included along with any route information it sends to requesting nodes. **Using destination sequence numbers ensures loop freedom and is simple to program.**”

  – “Proofs”
    - [PerkinsRoyer97]: proof sketch; missing cases - no error handling
    - [ZhouEtAl09]: over abstraction; it is not a proof for AODV

Loop Example

- Loop freedom does not only depend on sequence numbers, but also on
  - error handling
  - self entries
- Loop freedom of AODV is not guaranteed by the RFC
  - depends on the interpretation of the RFC
  - depends on the experience of the software engineer
- Some compliant implementations, such as ns2-AODV, contain loops

- Details
  - 2 nodes moving
  - 4 route requests



Research Outcomes (Process Algebra)

NICTA

- Algebra for Wireless Networks (AWN)
  - novel treatment of data structures, conditional unicast und local broadcast (w.r.t. to previous process algebras such as LOTOS)
  - formalisation and (dis)proof of key aspects of routing protocols, e.g. loop freedom, packet delivery
- Case study
  - Ad-hoc On Demand Distance Vector Protocol (AODV)
    - model the standard
    - **first formal and complete proof of loop freedom (for particular interpretations)**
    - analysed more key properties such as packet delivery or route discovery
  - Analysed variants/interpretations of AODV
    - all reasonable interpretations of the standard (RFC) analysed (more than 128)
- Publications

[1] A Process Algebra for Wireless Mesh Networks. In European Symposium on Programming (ESOP 2012), Lecture Notes in Computer Science, Springer, 2012. (to appear)

[2] A Process Algebra for Wireless Mesh Networks used for Modelling, Verifying and Analysing AODV. Technical report 5513, NICTA, 2012

Ambiguities and Loop Freedom

1.	**Updating the Unknown Sequence Number in Response to a Route Reply**	
1a.	the destination sequence number (DSN) is copied from the RREP message (Sect 6.7)	decrement of sequence numbers and loops
1b.	the routing table is not updated when the information inside is "fresher" (Sect. 6.1)	loop free
2.	**Updating with the Unknown Sequence Number (Sect. 6.5)**	
2a.	no update occurs	loop free, but opportunity to improve routes is missed.
2b.	overwrite any routing table entry by an update with an unknown DSN	decrement of sequence numbers and loops
2c.	use the new entry with the old DSN	loop free
3.	**More Inconclusive Evidence on Dealing with the Unknown Sequence Number (Sect. 6.2)**	
3a.	update when *incoming* sequence number is unknown	supports Interpretations 2b or 2c above
3b.	update when *existing* sequence number is unknown	decrement of sequence numbers and loops
3c.	update when no *existing* sequence number is known	supports Interpretation 2a above
4.	**(Dis)Allowing Self-Entries**	
4a.	allow self-entries	loop free if used with appropriate `invalidate`
4b.	disallow self-entries; if self-entries would occur, ignore mess.	loop free
4c.	disallow self-entries; if self-entries would occur, forward	loop free
5.	**Storing the Own Sequence Number**	
5a.	store sequence number as separate value	loop free
5b.	store sequence number inside routing table	excludes non-trivial self-entries (4b–c)
6.	**Invalidating Routing Table Entries in Response to a RERR message**	
6a.	copy DSN from RERR message (Sect. 6.11)	decrement of sequence numbers and loops (when allowing self-entries (Interpretation 4a))
6b.	no action if the DSN in the routing table is larger than the one in the RERR mess. (Sect. 6.1 & 6.11)	loops (when allowing self-entries)
6c.	take the maximum of the DSN of the routing table and the one from the RERR message	loops (when allowing self-entries)
6d.	take the maximum of the increased DSN of the routing table and the one from the RERR mess.	loop free

Table 2: Analysis of Different Interpretations of the RFC 3561 (AODV)

Research Landscape (w.r.t. AWN)

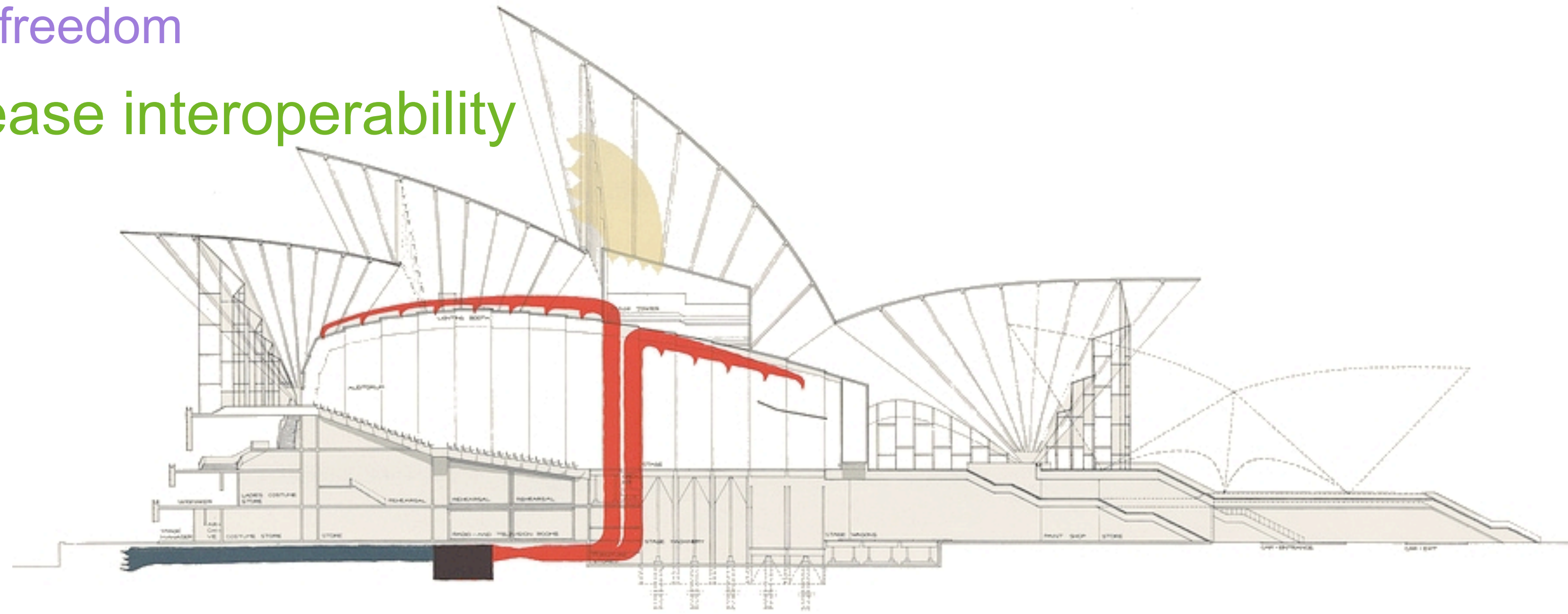
Approach	Description	Features	Point of difference
AWN	process algebra for WMNs (specification language + proof methodology)	broadcast unicast data structure translation to UPPAAL	WMN primitives readable
LOTOS (CCS, CSP, ACP)	general-purpose process algebra	first algebra with data	no assignment broadcast not a primitive (encoding less readable)
ESTELLE	based on abstract data types and finite automata	everything is a data structure (e.g., communication)	only testing and static analysis available
Model checking (e.g. UPPAAL)	method to check properties in a given scenarios (topology)	formal semantics automatic and executable scenarios	not designed for WMN
Petri nets	model of concurrency	graphical and intuitive interpretation explicit concurrency	no specification language specification much larger (hence less readable)
SysML	general-purpose modelling and specification languages	based on UML	usually no proof methodology
SDL	general-purpose modelling and specification languages	based on finite automata graphical version	usually no proof methodology

Key Research Outcomes (Summary)

- New languages and proof methodologies
  - process algebra AWN
  - routing algebra
- Modelling of AODV
  - process algebra: complete and detailed model (without time)
  - model checking: encoding of AWN specification
  - routing algebra: modelled parts of AODV
- Analysing/Verifying AODV
  - process algebra: proof methodology, invariant proofs
  - model checking: automatic finding of problematic behaviour e.g., no route discovery guarantee
  - analysed (all interpretations of) AODV

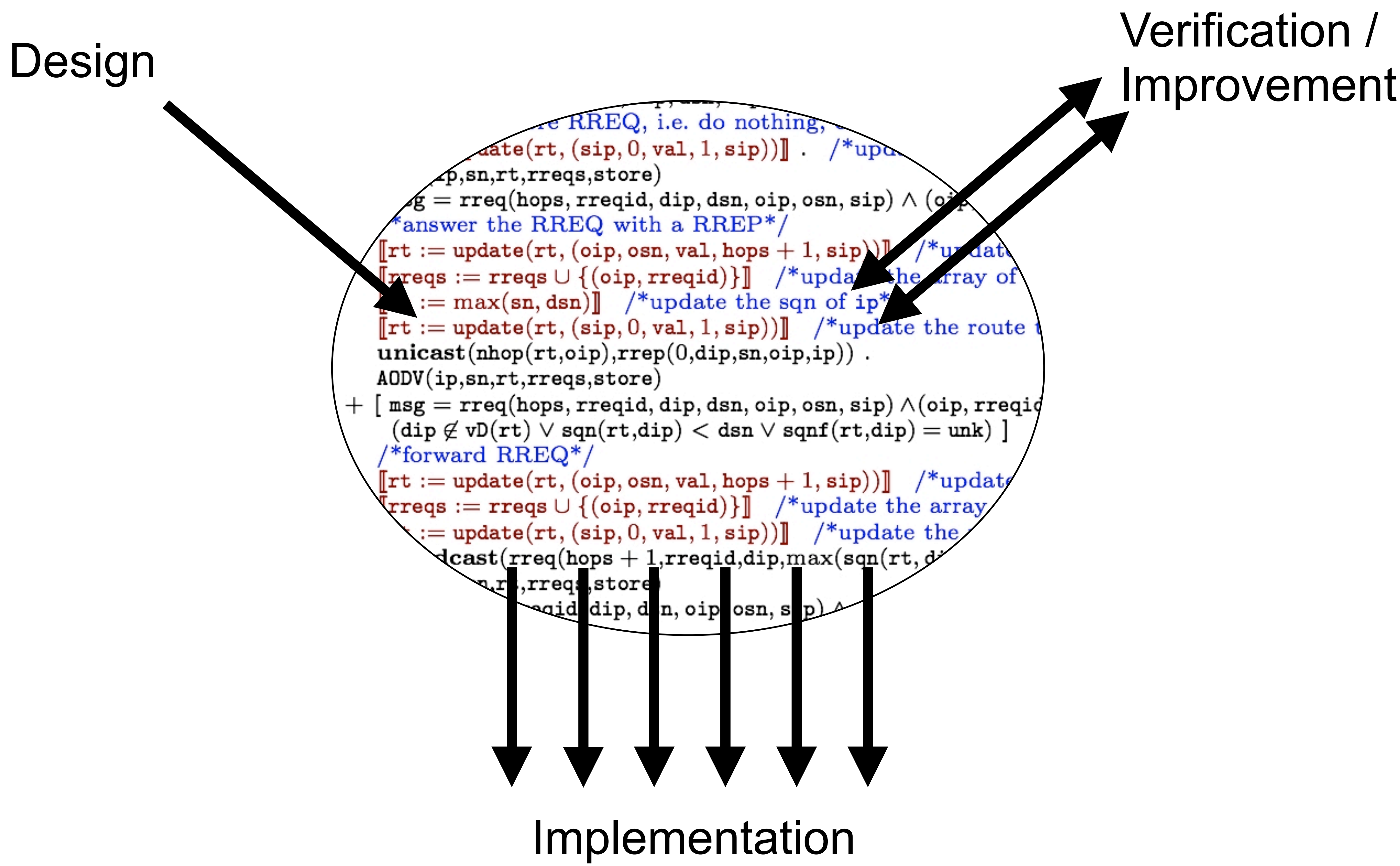
Vision

- Provide practical methods and tools for WMN protocols that
  - are used for specification and analysis/verification
  - have high usability and are intuitive
    - help (network) researchers/engineers to achieve their tasks and to tackle their problems
  - have expressive power to model wireless networks (e.g. broadcast)
  - are unambiguous and concise
- Key Goals
  - understand, formalise, analyse and solve network problems;
    - e.g. what is meant by loop freedom
  - remove ambiguities, increase interoperability
  - higher level of assurance
- Reduce “time-to-market”



Vision - Practical Protocol Engineering

NICTA



Future Work

- Extend languages and proof methodologies
  - process algebra, model checking: time, probability
  - routing algebra: complete expressive power
- Proof automatisation
  - process algebra: Isabelle/HOL
  - routing algebra: Prover9
- Specification vs. Implementation
  - check real implementations against (correct) specification
- Application of developed formal methods to new protocols
  - adaptive, modular protocols for WMNs

Links / Engagement

- Within NICTA
  - software systems research group
    - proof automatisation (Isabelle/HOL)
- Academic cooperation
  - Cambridge, Stanford, Stony Brook, Nijmegen, ...
- Industry partner
  - Firetide
    - current main focus on channel allocation

Global research competitive position

NICTA

Research Group	Key staff	Scale of effort	Point of difference
NICTA Mesh protocols	Rob van Glabbeek Peter Höfner	2 researchers	rigorous formal methods application to relevant protocols
Cambridge University Metarouting	Timothy G. Griffin	4 researchers and students	focus on analysis of internet protocols (BGP)
AT&T Labs Research	Pamela Zave	numbers vary	focus on higher-level protocols (e.g. SIP)
Stony Brook University	Scott A. Smolka C.R. Ramakrishnan	3 researchers	no close collaboration with network engineers
University of Pennsylvania NetDB@Penn	Boon Thau Loo	2 researchers and 8 PhD students	distributed systems, analysis of BGP, no wireless
Radboud University Model-Based System Develop.,	Frits Vaandrager	4 researchers and students	no focus on networks, no close collaboration with network engineers

Selected Publications

NICTA

Title	Conference	Year
Sequence Numbers Do Not Guarantee Loop Freedom —AODV Can Yield Routing Loops—	submitted to Sigcomm 2012	2012
A Process Algebra for Wireless Mesh Networks	European Symposium on Programming (ESOP 12)	2012
Automated Analysis of AODV using AODV	Tools and Algorithms for the Construction and Analysis of Systems (TACAS 12)	2012
A Process Algebra for Wireless Mesh Networks used for Modelling Verifying and Analysing AODV.	Technical Report, NICTA	2012
Modelling and Analysis of AODV in UPPAAL	Workshop on Rigorous Protocol Engineering (W-Ripe 11, ICNP-workshop)	2011
Towards an Algebra of Routing Tables	Relational and Algebraic Methods in Computer Science (RAMiCS 11)	2011

NICTA

Questions, Comments ?