NICTA

From **imagination** to **impact**

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

A Mechanized Proof of Loop Freedom of the (untimed) AODV routing protocol

Timothy Bourke, Peter Höfner, Rob van Glabbeek

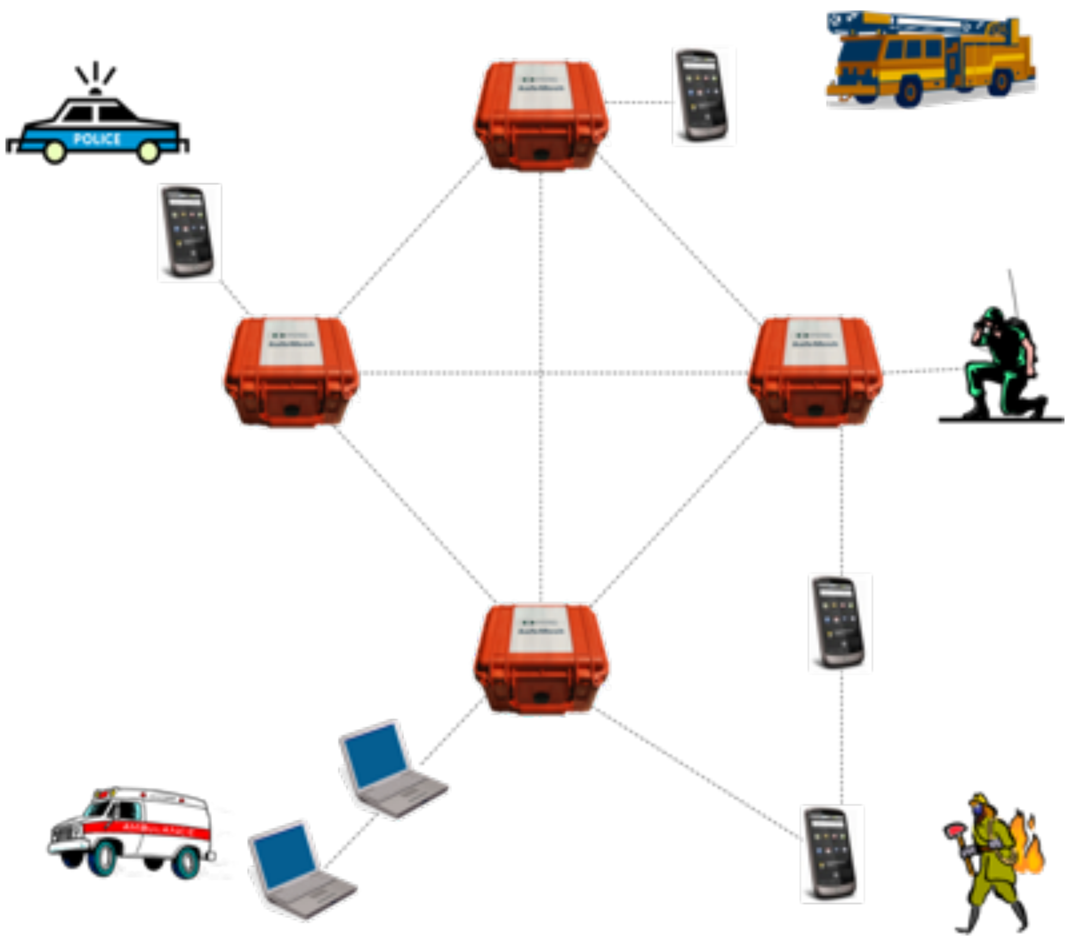
Australian Government
Department of Broadband, Communications and the Digital Economy
Australian Research Council

NICTA Funding and Supporting Members and Partners

NICTA Partners

Wireless Mesh Networks

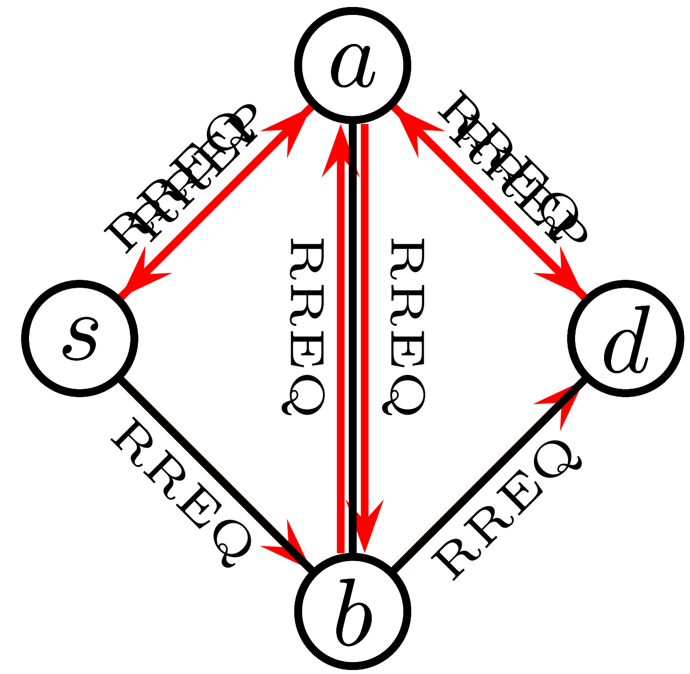
- Wireless Mesh Networks (WMNs)
  - key features: mobility, dynamic topology, wireless multihop backhaul
  - quick and low cost deployment
- Applications
  - public safety
  - emergency response, disaster recovery
  - transportation
  - mining
  - smart grid
  - ...
- Limitations in reliability and performance



Ad Hoc On-Demand Distance Vector Protocol

NICTA

- 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
- Essential Data structure
  - a routing table
    - local knowledge
    - entries:
      $(dip, dsn, dsk, val, hops, nhip, pre)$



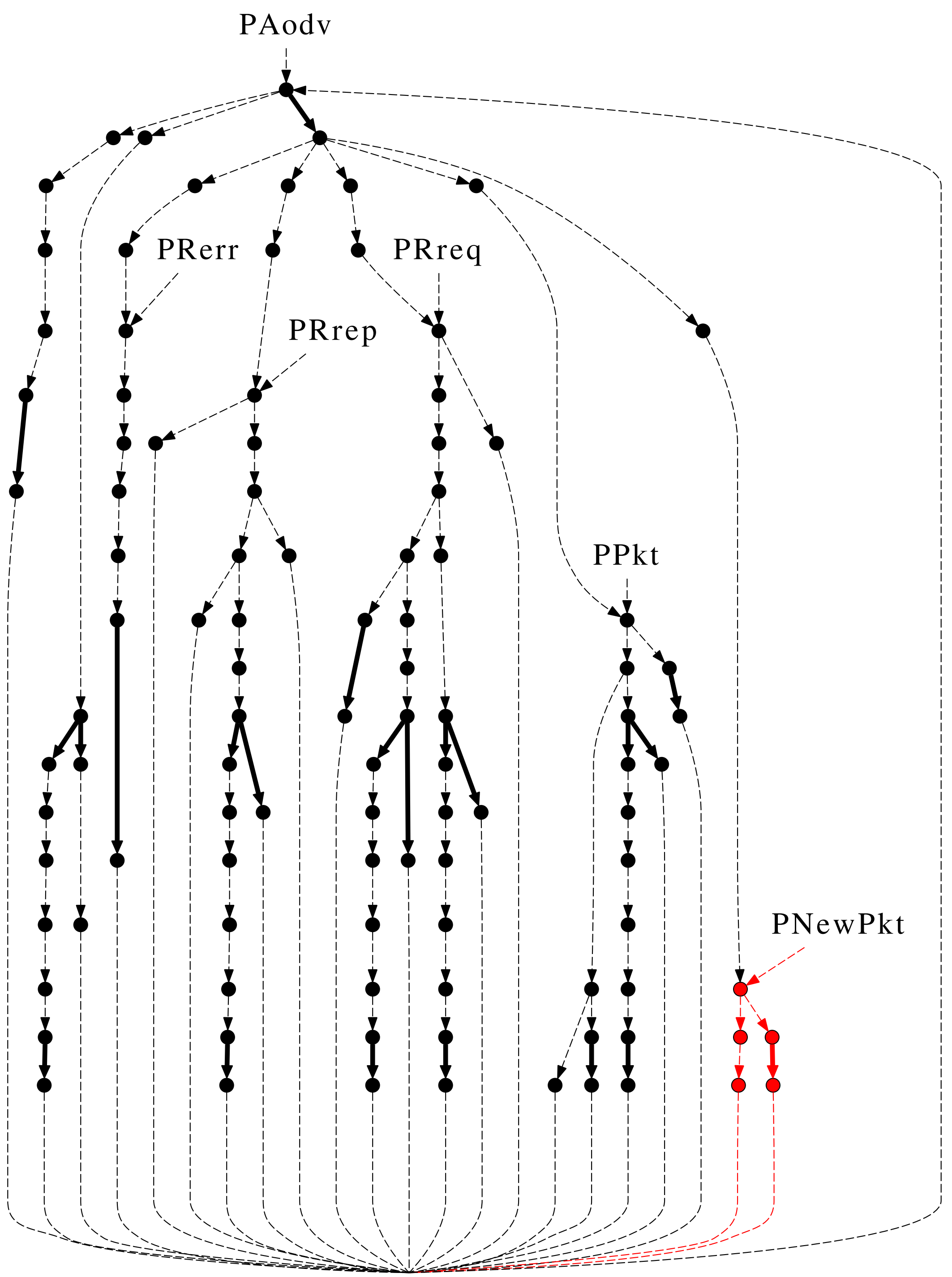
Ad Hoc On-Demand Distance Vector Protocol

NICTA

- Properties of AODV
  - route correctness ✓
  - loop freedom
  - route discovery ✗
  - packet delivery ✗

Specification in Process Algebra

- AODV in Process Algebra AWN
  - standard process algebra
  - with data structure
  - network-specific primitives, such as
    - (local) broadcast
    - (conditional) unicast
  - layered structure (processes, nodes, network of nodes, encapsulation)



- Model of AODV
  - 6 processes
  - about 150 lines of specification

Snippet of AODV

```
+ [ (oip, rreqid) ∉ rreqs ]        /* the RREQ is new to this node */
  /* update the route to oip in rt */
  [[rt := update(rt, (oip, osn, valid, hops + 1, sip, ∅))]]
  /* update rreqs by adding (oip, rreqid) */
  [[rreqs := rreqs ∪ {(oip, rreqid)}]]
  (
    [ dip = ip ]        /* this node is the destination node */
      /* update the sqn of ip by setting it to max(sqn(rt, ip), dsn) */
      [[rt := update(rt, (ip, dsn, valid, 0, ip, ∅))]]
      /* unicast a RREP towards oip of the RREQ; next hop is sip */
      unicast(sip, rrep(0, dip, sqn(rt, ip), oip, ip)) . AODV(ip, rt, rreqs, queues)
      ▶ /* If the packet transmission is unsuccessful, a RERR message is generated */
        [[dests := {(rip, rsn) | (rip, rsn, valid, ∗, sip, ∗) ∈ rt}]]
        [[pre := ⋃{precs(rt, rip) | (rip, ∗) ∈ dests}]]
        [[for all (rip, ∗) ∈ dests : invalidate(rt, rip)]]
        groupcast(pre, rerr(dests, ip)) . AODV(ip, rt, rreqs, queues)
    + [ dip ≠ ip ]        /* this node is not the destination node */
      (
        [ dip ∈ aD(rt) ∧ dsn ≤ sqn(rt, dip) ∧ sqn(rt, dip) ≠ 0 ]        /* valid route to dip that is
        fresh enough */
          /* update rt by adding sip to precs(rt, dip) */
          [[r := addpre(σ_route(rt, dip), {sip}); rt := update(rt, r)]]
```

A Pen-and-Paper Proof for the RFC 3561

- There existed a pen-and-paper proof
  - around 20 pages
  - about 40 invariants
    - state invariants
    - transition invariants
    - talking about one or more nodes

Mechanizing the Proof

- done in Isabelle/HOL
- Mechanization of the process algebra AWN
  - details see ITP’14
  - some crucial parts are discussed below
- Mechanization of the loop-freedom proof
  - 360 lemmas of which 40 are invariants
  - “usual” overhead

Node Properties

- Often straight forward

- Example:
  "all routing table entries have a hop count greater than or equal to one"

$$(*, *, *, *, hops, *, *) \in \xi_N^{ip} \;\Rightarrow\; 1 \leq hops$$

$$\mathsf{paodv}\ \mathsf{i} \mathrel{|\!\!\models} \mathsf{onl}\ \Gamma_{\mathsf{aodv}}\ (\lambda(\xi, \text{-}).\ \forall \mathsf{ip} \in \mathsf{kD}(\mathsf{rt}\ \xi).\ 1 \leq \mathsf{the}\ (\mathsf{dhops}\ (\mathsf{rt}\xi)\ \mathsf{ip}))$$

Network Properties for Single Nodes

- stating network properties already gets complicated
- proving even more sophisticated
  (but also for the pen-and-paper proof)

Network Properties for Single Nodes

- Example:
  "the quality of the routing table entries for a destination *dip* is strictly increasing along a route towards *dip*"

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

$$\begin{aligned}
&\mathsf{opaodv\ i} \Vdash (\mathsf{otherwith}\ (\mathsf{op} =)\ \{\mathsf{i}\} \\
&\qquad (\mathsf{orecvmsg}\ (\lambda\sigma\ m.\ \mathsf{msg_fresh}\ \sigma\ m\ \wedge\ \mathsf{msg_zhops}\ m)), \\
&\qquad \mathsf{otherquality_increases}\{\mathsf{i}\}\ \rightarrow) \\
&\qquad \mathsf{onl}\ \Gamma_{\mathsf{aodv}}\ (\lambda(\sigma, \text{-}).\ \forall \mathsf{dip}.\ \mathsf{let\ nhip} = \mathsf{the}\ (\mathsf{nhop}\ (\mathsf{rt}(\sigma\ \mathsf{i}))\ \mathsf{dip}) \\
&\qquad\qquad \mathsf{in\ dip} \in \mathsf{vD}\ (\mathsf{rt}\ (\sigma\ \mathsf{i}))\ \cap\ \mathsf{vD}\ (\mathsf{rt}\ (\sigma\ \mathsf{nhip})) \\
&\qquad\qquad\quad \wedge\ \mathsf{nhip} \neq \mathsf{dip} \\
&\qquad\qquad\quad \longrightarrow (\mathsf{rt}\ (\sigma\ \mathsf{i}))\ \sqsubset_{\mathsf{dip}}\ (\mathsf{rt}\ (\sigma\ \mathsf{nhip})))
\end{aligned}$$

Why Mechanisation?

NICTA

- Did we waste our time
  - there was a pen-and-paper proof before
  - took about 1 person-year
    (building up infrastructure for AWN, etc.)
  - more confidence
  - found one missing case and some typos
- Do we gain anything when we analyse variants

Variants

- Variants might occur
  - change in specification
    (not yet standardized)
  - optimizations found
  - different Interpretations of the Specification
    (written in English)

Change in Specifications

```
+ [ (oip, rreqid) ∉ rreqs ]      /* the RREQ is new to this node */
  /* update rreqs by adding (oip, rreqid) */
  [[rreqs := rreqs ∪ {(oip, rreqid)}]]
  /* update the route to oip in rt */
  [[rt := update(rt, (oip, osn, valid, hops + 1, sip, ∅))]]
  (
   [ dip = ip ]      /* this node is the destination node */
     /* update the sqn of ip by setting it to max(sqn(rt, ip), dsn) */
     [[rt := update(rt, (ip, dsn, valid, 0, ip, ∅))]]
     /* unicast a RREP towards oip of the RREQ; next hop is sip */
     unicast(sip, rrep(0, dip, sqn(rt, ip), oip, ip)) . AODV(ip, rt, rreqs, queues)
     ▶ /* If the packet transmission is unsuccessful, a RERR message is generated */
       [[dests := {(rip, rsn) | (rip, rsn, valid, ∗, sip, ∗) ∈ rt}]]
       [[pre := ⋃{precs(rt, rip) | (rip, ∗) ∈ dests}]]
       [[for all (rip, ∗) ∈ dests : invalidate(rt, rip)]]
       groupcast(pre, rerr(dests, ip)) . AODV(ip, rt, rreqs, queues)
   + [ dip ≠ ip ]      /* this node is not the destination node */
     (
       [ dip ∈ aD(rt) ∧ dsn ≤ sqn(rt, dip) ∧ sqn(rt, dip) ≠ 0 ]      /* valid route to dip that is
       fresh enough */
         /* update rt by adding sip to precs(rt, dip) */
         [[r := addpre(σ_route(rt, dip), {sip}); rt := update(rt, r)]]
```

Different Reading of a Standard

Analysing Variants

- Overview
- Analysed 5 Variants
  - from simple optimisations
  - to “bug fixing”
- An Interactive Theorem Prover can try to replay the proof
  - points a points where proofs/invariants break down
  - hope it’s easy to fix
  - if you cannot fix it, you don’t know anything

Variants of AODV

- Variant A: Skipping route request identifiers
  - small optimisation
  - RFC uses unnecessary data structure

  - modification in specification took about 5 minutes
  - proof went basically through

Variants of AODV

- Variant B: Forwarding route replies
  - “bug” fix of RFC
  - modification includes deletion of 3 lines of the spec
  - out of 400-odd lemmas only 7 broke down
    - 4 were easily fixed (broken references to line numbers)
    - about 3 hours to repair (for a novice)

Variants of AODV

- Variant C: From groupcast to broadcast
  - groupcast failed to inform some nodes
  - using boadcast is (in some sense) more efficient
  - as a consequence: simplifying data structure

  - modification includes new guard
  - about 75 lemmas broke down
  - 74 simple fixes
    - delete references to dropped data structure
    - fix line references
  - basically 1 lemma broke which could be fixed

Variants of AODV

- Variant D: Forwarding route requests
  - requests are not send to all nodes
  - missed opportunity to establish routes
  - performance improvement (maybe)

  - 8 lines were changed in the specification
  - 17 lemmas broke down
  - one proof needed major rework

Variants of AODV

- Variant E: All changes discussed above
  - basically merging all proof changes
  - no conflicting proofs/conditions showed up

The Other Side of the Medal

- Is every variant fix so simple
  - probable yes, IF the lemmas/invariants stay valid
    (which was the case for the presented variants)
  - no, IF lemmas do not hold any longer
    Variant F: Updating with the Unknown Sequence Number
    - since lemmas are not valid any more deep expertise is needed
      to see that the lemma is incorrect
      to provide an alternative repair

Conclusion/Future Work

- Mechanised Proof of AODV and variants
  - based on process algebra
  - mechanised in Isabelle
  - about 1 hour verification time (with 4 cores)
  - both mechanisation of process algebra and AODV can be found in the Archive of Formal Proofs
- Variants are often so small that the proof can be “replayed”

- Optimise Mechanisation
  - simple changes might be automated (e.g. reference to line numbers)
- Extend Formalism and Model
  - add time
  - add probabilities and quantities

Questions

NICTA

From **imagination** to **impact**