A Process Algebra for Wireless Mesh Networks

A. Fehnker, R. van Glabbeek, P. Höfner,
A. McIver, M. Portmann, W.-L. Tan

NICTA
UNSW
THE UNIVERSITY OF NEW SOUTH WALES
SYDNEY · CANBERRA · AUSTRALIA

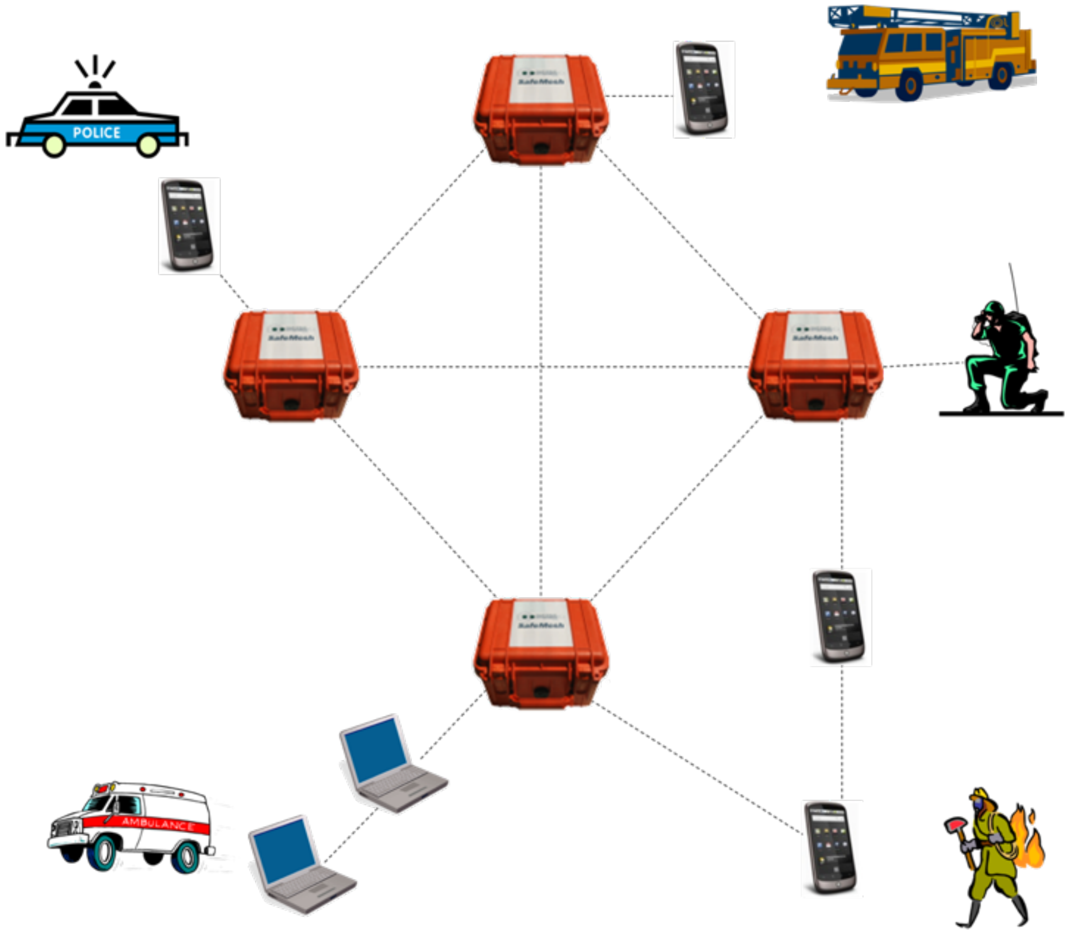
Australian Government
Department of Broadband, Communications and the Digital Economy
Australian Research Council

NICTA Members
NICTA Partners

What is the Problem?

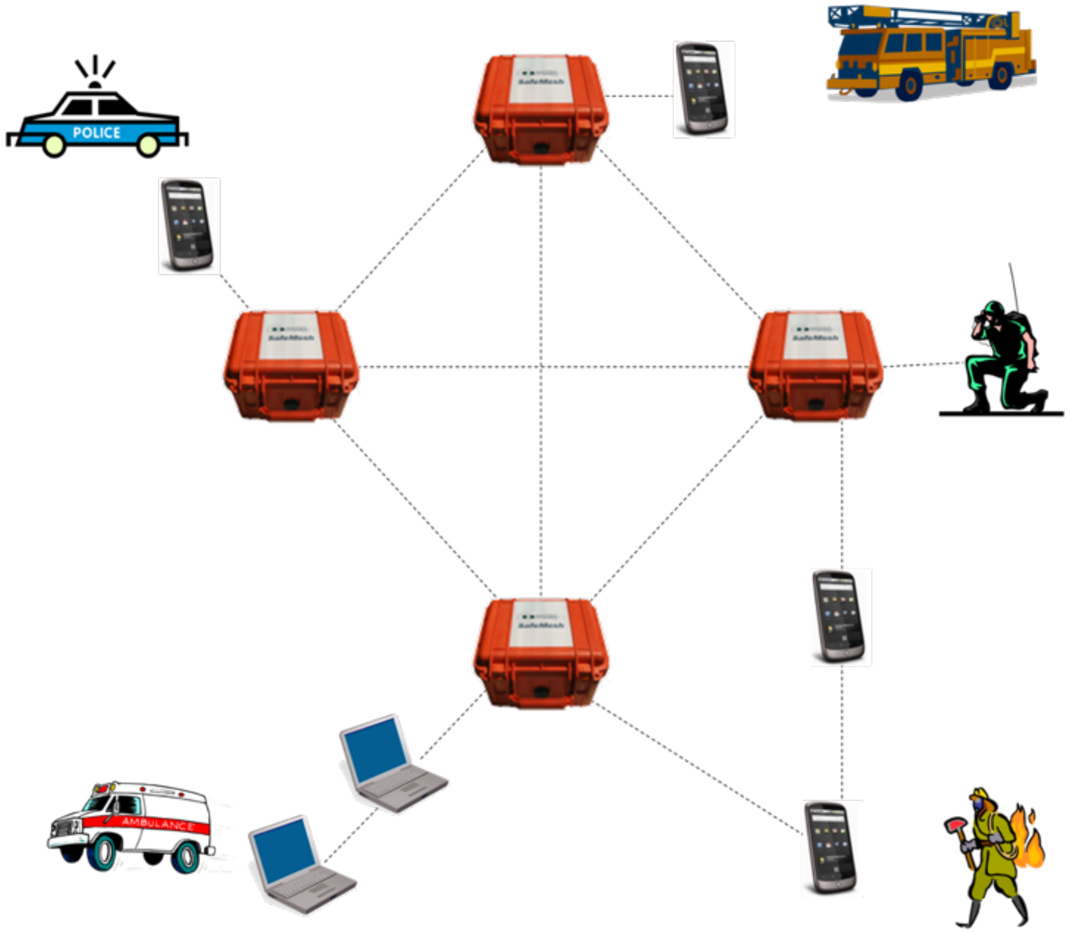
- Wireless Mesh Networks (WMNs)
  – key features: mobility, dynamic topology, wireless multihop backhaul
  – quick and low cost deployment

- Applications
  – public safety
  – emergency response, disaster recovery
  – transportation
  – mining
  – smart grid
  – ...



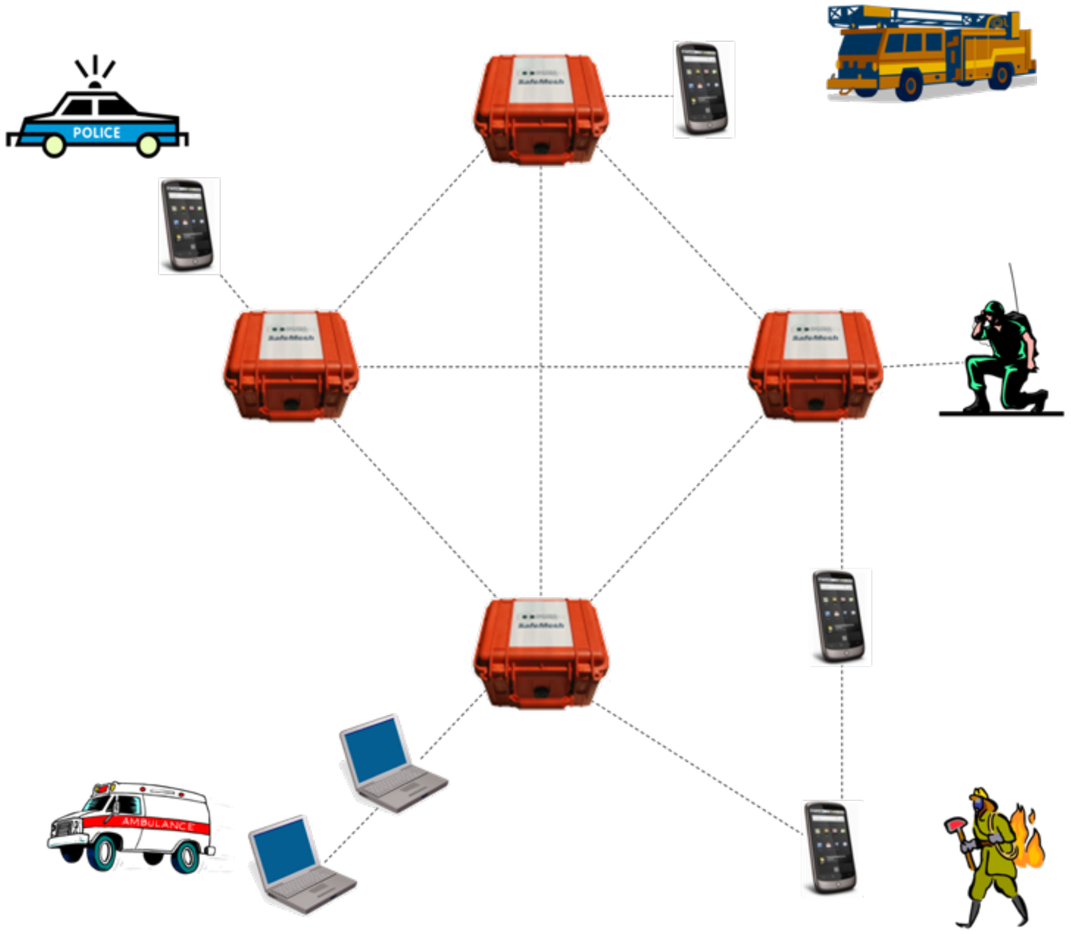
What is the Problem?

- WMNs promise to be fully
  - self-configuring
  - self-healing
  - self-optimising



What is the Problem?

- WMNs promise to be fully
  - self-configuring
  - self-healing
  - self-optimising
- **DOES NOT WORK** (in reality)
- Limitations in reliability and performance
- Limitations confirmed by
  - end users (e.g. police)
  - own experiments
    - Cisco, Motorola, Firetide, ...
  - industry



What is the Problem?

“Our requirement was for a system breadcrumb type deployment over at least 4 nodes and maintain a throughput of around 5Mbps–10Mbps to enable 'good' quality video to be passed. The commercial devices failed to meet our requirements [...]”

Rick Loebler, Applied Technology Manager,
NSW Police Force

Formal Methods for Mesh Networks

- Goal
  - model, analyse, verify, improve and increase the performance of wireless mesh protocols
  - develop suitable formal methods techniques

- Benefits
  - more reliable protocols
  - finding and fixing bugs
  - better performance
  - proving correctness
  - reduce “time-to-market”

Process Algebra

NICTA

$+ [\, (\texttt{oip}, \texttt{rreqid}) \notin \texttt{rreqs} \,]$ /* the RREQ is new to this node */

/* update the route to oip in rt */

$[\![\texttt{rt} := \texttt{update}(\texttt{rt}, (\texttt{oip}, \texttt{osn}, \texttt{valid}, \texttt{hops}+1, \texttt{sip}, \emptyset))]\!]$

/* update rreqs by adding (oip, rreqid) */

$[\![\texttt{rreqs} := \texttt{rreqs} \cup \{(\texttt{oip}, \texttt{rreqid})\}]\!]$

(

$[\, \texttt{dip} = \texttt{ip} \,]$ /* this node is the destination node */

/* update the sqn of ip by setting it to max(sqn(rt, ip), dsn) */

$[\![\texttt{rt} := \texttt{update}(\texttt{rt}, (\texttt{ip}, \texttt{dsn}, \texttt{valid}, 0, \texttt{ip}, \emptyset))]\!]$

/* 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 */

$[\![\texttt{dests} := \{(\texttt{rip}, \texttt{rsn}) \mid (\texttt{rip}, \texttt{rsn}, \texttt{valid}, *, \texttt{sip}, *) \in \texttt{rt}\}]\!]$

$[\![\texttt{pre} := \bigcup\{\texttt{precs}(\texttt{rt}, \texttt{rip}) \mid (\texttt{rip}, *) \in \texttt{dests}\}]\!]$

$[\![\text{for all } (\texttt{rip}, *) \in \texttt{dests} : \texttt{invalidate}(\texttt{rt}, \texttt{rip})]\!]$

groupcast(pre, rerr(dests, ip)) . AODV(ip, rt, rreqs, queues)

$+ [\, \texttt{dip} \neq \texttt{ip} \,]$ /* this node is not the destination node */

(

$[\, \texttt{dip} \in \texttt{aD}(\texttt{rt}) \land \texttt{dsn} \leq \texttt{sqn}(\texttt{rt}, \texttt{dip}) \land \texttt{sqn}(\texttt{rt}, \texttt{dip}) \neq 0 \,]$ /* valid route to dip that is fresh enough */

/* update rt by adding sip to precs(rt, dip) */

$[\![\texttt{r} := \texttt{addpre}(\sigma_{route}(\texttt{rt}, \texttt{dip}), \{\texttt{sip}\}); \texttt{rt} := \texttt{update}(\texttt{rt}, \texttt{r})]\!]$

Process Algebra – Achievements

- New: Algebra of Wireless Network (AWN)
  - language for formalising specifications of network protocols
  - key features:
    - guarantee local broadcast
    - conditional unicast
    - data handling
- Case study
  - full concise specification of AODV (without time)
  - classification of ambiguities and contradictions in the official specification (RFC)
  - verified/disproved properties, e.g. loop-freedom
  - found other shortcomings such as optimality
  - proposed improvements for some limitations
    - evaluation using model checking (TACAS 2012)

Process Algebra

- Inspired by
  - CCS, CSP, ACP, LOTOS, mCRL, π- Calculus
  - ω - Calculus

Structure of WMNs

NICTA

- User
  - Network as a “cloud”
- Collection of nodes
  - connect / disconnect / send / receive
  - “parallel execution” of nodes
- Nodes
  - data management
    - data packets, messages, IP addresses ...
  - message management (avoid blocking)
  - core management
    - broadcast / unicast / groupcast ...
  - “parallel execution” of sequential processes

Nodes (Sequential Process Expressions)

- Syntax of sequential process expressions

$$
\begin{aligned}
SP \; ::= \;& X(exp_1, \ldots, exp_n) \mid [\varphi]SP \mid [\![\mathtt{var} := exp]\!]SP \mid SP + SP \mid \\
& \alpha.SP \mid \mathbf{unicast}(dest, ms).SP \blacktriangleright SP \\
\alpha \; ::= \;& \mathbf{broadcast}(ms) \mid \mathbf{groupcast}(dests, ms) \mid \mathbf{send}(ms) \mid \\
& \mathbf{deliver}(data) \mid \mathbf{receive}(\mathtt{msg})
\end{aligned}
$$

Structural Operational Semantics I

- internal state determined by expression and valuation

$$\xi, \textbf{broadcast}(ms).p \xrightarrow{\textbf{broadcast}(\xi(ms))} \xi, p$$
$$\xi, \textbf{groupcast}(dests, ms).p \xrightarrow{\textbf{groupcast}(\xi(dests),\xi(ms))} \xi, p$$
$$\xi, \textbf{unicast}(dest, ms).p \blacktriangleright q \xrightarrow{\textbf{unicast}(\xi(dest),\xi(ms))} \xi, p$$
$$\xi, \textbf{unicast}(dest, ms).p \blacktriangleright q \xrightarrow{\neg\textbf{unicast}(\xi(dest))} \xi, q$$
$$\xi, \textbf{send}(ms).p \xrightarrow{\textbf{send}(\xi(ms))} \xi, p$$
$$\xi, \textbf{deliver}(data).p \xrightarrow{\textbf{deliver}(\xi(data))} \xi, p$$
$$\xi, \textbf{receive}(\texttt{msg}).p \xrightarrow{\textbf{receive}(m)} \xi[\texttt{msg} := m], p \qquad (\forall m \in \texttt{MSG})$$

Structural Operational Semantics II

- internal state determined by expression and valuation

$$\xi, [\![\mathtt{var} := exp]\!]p \xrightarrow{\tau} \xi[\mathtt{var} := \xi(exp)], p$$

$$\frac{\xi, p \xrightarrow{a} \zeta, p'}{\xi, p + q \xrightarrow{a} \zeta, p'} \qquad \frac{\xi, q \xrightarrow{a} \zeta, q'}{\xi, p + q \xrightarrow{a} \zeta, q'}$$

$$\frac{\xi \xrightarrow{\varphi} \zeta}{\xi, [\varphi]p \xrightarrow{\tau} \zeta, p}$$

Nodes (Parallel Processes)

- Syntax

$$PP \ ::= \ \xi, SP \ | \ PP \langle\!\langle \ PP \ ,$$

- Operational Semantics

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

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

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

Network

- node expressions:

$$M ::= \quad ip : P : R \quad | \quad M \| M$$

- Operational Semantics (snippet)

$$\frac{P \xrightarrow{\textbf{broadcast}(m)} P'}{ip : P : R \xrightarrow{R : \textbf{*cast}(m)} ip : P' : R}$$

$$\frac{P \xrightarrow{\textbf{groupcast}(D,m)} P'}{ip : P : R \xrightarrow{R \cap D : \textbf{*cast}(m)} ip : P' : R}$$

$$\frac{P \xrightarrow{\textbf{unicast}(dip,m)} P' \qquad dip \in R}{ip : P : R \xrightarrow{\{dip\} : \textbf{*cast}(m)} ip : P' : R}$$

$$\frac{P \xrightarrow{\neg\textbf{unicast}(dip)} P' \qquad dip \notin R}{ip : P : R \xrightarrow{\tau} ip : P' : R}$$

$$ip : P : R \xrightarrow{\textbf{connect}(ip,ip')} ip : P : R \cup \{ip'\}$$

$$ip : P : R \xrightarrow{\textbf{disconnect}(ip,ip')} ip : P : R - \{ip'\}$$

Network

- Operational Semantics (snippet II)

$$\frac{M \xrightarrow{R:\mathbf{*cast}(m)} M' \quad N \xrightarrow{H\neg K:\mathbf{listen}(m)} N'}{M\|N \xrightarrow{R:\mathbf{*cast}(m)} M'\|N' \qquad N\|M \xrightarrow{R:\mathbf{*cast}(m)} N'\|M'} \left(\begin{matrix} H \subseteq R \\ K \cap R = \emptyset \end{matrix}\right)$$

$$\frac{M \xrightarrow{H\neg K:\mathbf{listen}(m)} M' \quad N \xrightarrow{H'\neg K':\mathbf{listen}(m)} N'}{M\|N \xrightarrow{(H\cup H')\neg(K\cup K'):\mathbf{listen}(m)} M'\|N'}$$

$$\frac{M \xrightarrow{a} M'}{M\|N \xrightarrow{a} M'\|N} \qquad \frac{N \xrightarrow{a} N'}{M\|N \xrightarrow{a} M\|N'} \qquad (\forall a \in \{ip:\mathbf{deliver}(d), \tau\})$$

Encapsulation

- Syntax $\mathrm{N} ::= [\mathrm{M}]$
- Operational Semantics

$$\frac{M \xrightarrow{R:\,\mathbf{*cast}(m)} M'}{[M] \xrightarrow{\tau} [M']} \qquad \frac{M \xrightarrow{\{ip\}\neg K\,:\,\mathbf{listen}(\mathtt{newpkt}(d,dip))} M'}{[M] \xrightarrow{ip\,:\,\mathbf{newpkt}(d,dip)} [M']}$$

$$\frac{M \xrightarrow{\tau} M'}{[M] \xrightarrow{\tau} [M']} \qquad \frac{M \xrightarrow{ip\,:\,\mathbf{deliver}(d)} M'}{[M] \xrightarrow{ip\,:\,\mathbf{deliver}(d)} [M']}$$

$$\frac{M \xrightarrow{\mathbf{connect}(ip,ip')} M'}{[M] \xrightarrow{\mathbf{connect}(ip,ip')} [M']} \qquad \frac{M \xrightarrow{\mathbf{disconnect}(ip,ip')} M'}{[M] \xrightarrow{\mathbf{disconnect}(ip,ip')} [M']}$$

A Bit of Theoretical Results

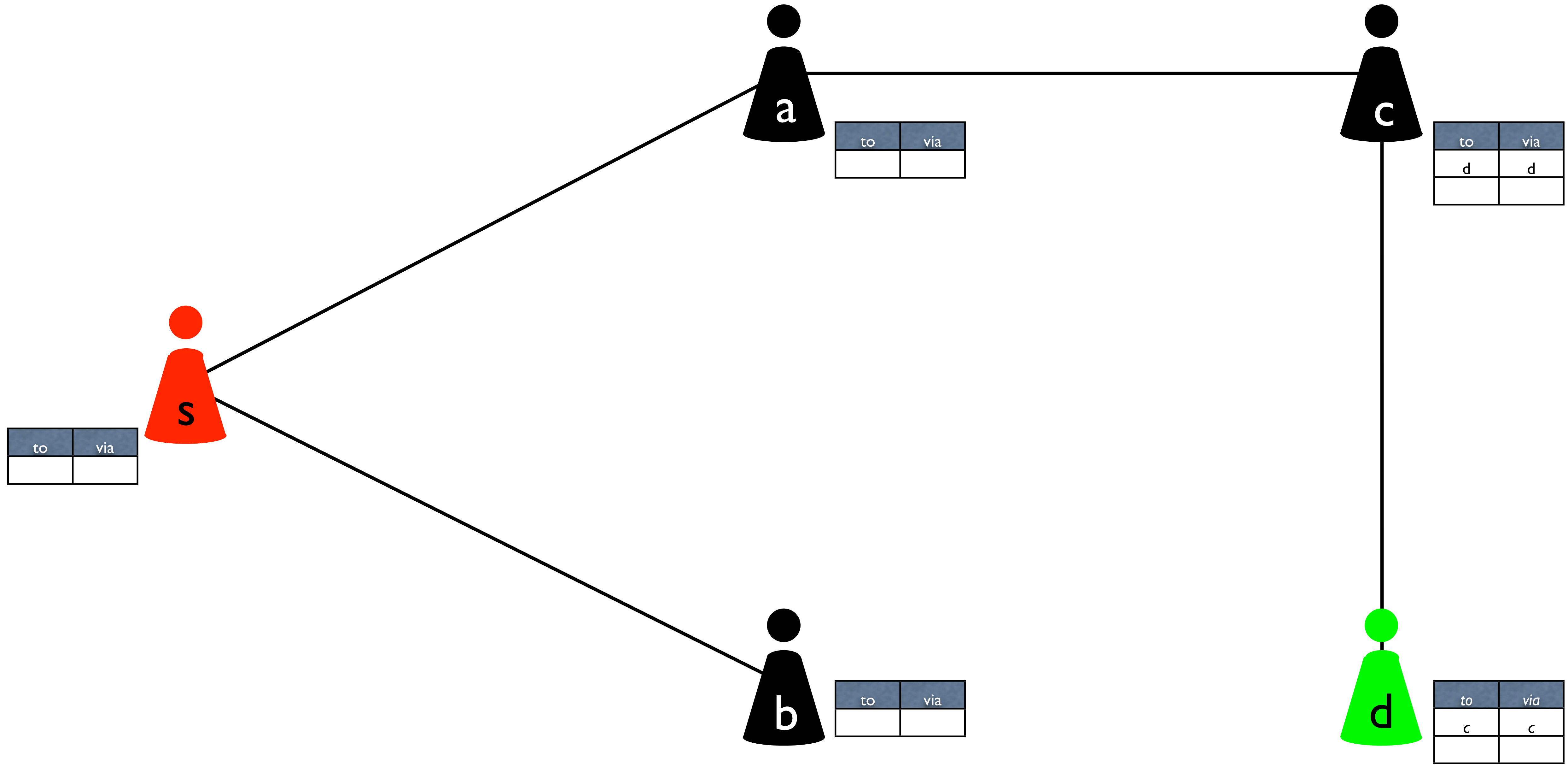
- process algebra is blocking (our model is non-blocking)
- process algebra is isomorphic to one without data structure --- a process for every substitution instance
- generates same transition system (up to strong bisimulation)
- resulting algebra is in *de Simone* format (by this strong bisimulation and other semantic equivalences are congruences)
- both parallel operators are associative (follows by a meta result of Cranen, Mousavi, Reniers)

A Formal Model for AODV

NICTA

- AODV: Ad-hoc On-Demand Distance Vector Routing Protocol
  - Ad hoc (network is not static)
  - On-Demand (routes are established when needed)
  - Distance (metric is hop count)
  - Vector (routing table has the form of a vector)
  - Developed 1997-2001 by Perkins, Beldig-Royer and Das (University of Cincinnati)
- Core components modelled
  - no time
  - no probability

AODV – An Example



s is looking for a route to d

Process Algebra – Snippet

Process 1 The basic routine

```
AODV(ip,rt,rreqs,store) ≝
    receive(msg) .
    /* depending on the message, the node calls different processes */
    (
      [ msg = newpkt(data, dip) ]        /* new DATA packet */
        PKT(data,dip,ip;ip,rt,rreqs,store)
      + [ msg = pkt(data, dip, oip) ]        /* incoming DATA packet */
        PKT(data,dip,oip;ip,rt,rreqs,store)
      + [ msg = rreq(hops, rreqid, dip, dsn, oip, osn, sip) ]        /* RREQ */
        /* update the route to sip in rt */
        ⟦rt := update(rt, (sip,0,val,1,sip,∅))⟧        /* 0 is the sequence number “unknown” */
        RREQ(hops,rreqid,dip,dsn,oip,osn,sip;ip,rt,rreqs,store)
      + [ msg = rrep(hops, dip, dsn, oip, sip) ]        /* RREP */
        /* update the route to sip in rt */
        ⟦rt := update(rt, (sip,0,val,1,sip,∅))⟧
        RREP(hops,dip,dsn,oip,sip;ip,rt,rreqs,store)
      + [ msg = rerr(dests, sip) ]        /* RERR */
        /* update the route to sip in rt */
        ⟦rt := update(rt, (sip,0,val,1,sip,∅))⟧
        RERR(dests,sip;ip,rt,rreqs,store)
    )
 + [ Let dip ∈ vD(rt) ∩ qD(store) ]        /* send a queued data packet if a valid route is known */
    ⟦data := head(σ_queue(store, dip))⟧
    unicast(nhop(rt,dip),pkt(data,dip,ip)) .
      /* the queue is only updated if the transmission was successful. */
      ⟦store := drop(dip, store)⟧
      AODV(ip,rt,rreqs,store)
    ▶ /* an error is produced and the routing table is updated */
      ⟦dests := {(rip,inc(sqn(rt,rip))) | rip ∈ vD(rt) ∧ nhop(rt,rip) = nhop(rt,dip)}⟧
      ⟦rt := invalidate(rt, dests)⟧
      ⟦pre := ⋃{precs(rt,rip) | (rip,∗) ∈ dests}⟧
      groupcast(pre,rerr(dests,ip)) . AODV(ip,rt,rreqs,store)
```

Ad Hoc On-Demand Distance Vector Protocol

- Invariant proofs
- temporal properties
- Properties of AODV
  - loop freedom ✓
  - route correctness ✓
  - route found ✗
  - packet delivery ✗

Process Algebra

- New process algebra developed
- Language for formalising specs of network protocols
- Key features:
  - guarantee broadcast
  - prioritised unicast
  - data handling
- Achievements
  - full concise specification of AODV (RFC 3561) (no time)
  - formally verified loop-freedom (without timeouts)
    - invariant proof
  - found several ambiguities, mistakes, shortcomings
  - found solutions for some limitations

Conclusion/Future Work

- Extend formal methods to other protocols
  - OSLR, DYMO, ...
- Add further necessary concepts
  - time
  - probability

NICTA

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