A Rigorous Analysis of AODV and its Variants

Peter Höfner, Rob van Glabbeek, Wee Lum Tan,
Marius Portman, Annabelle McIver, Ansgar Fehnker

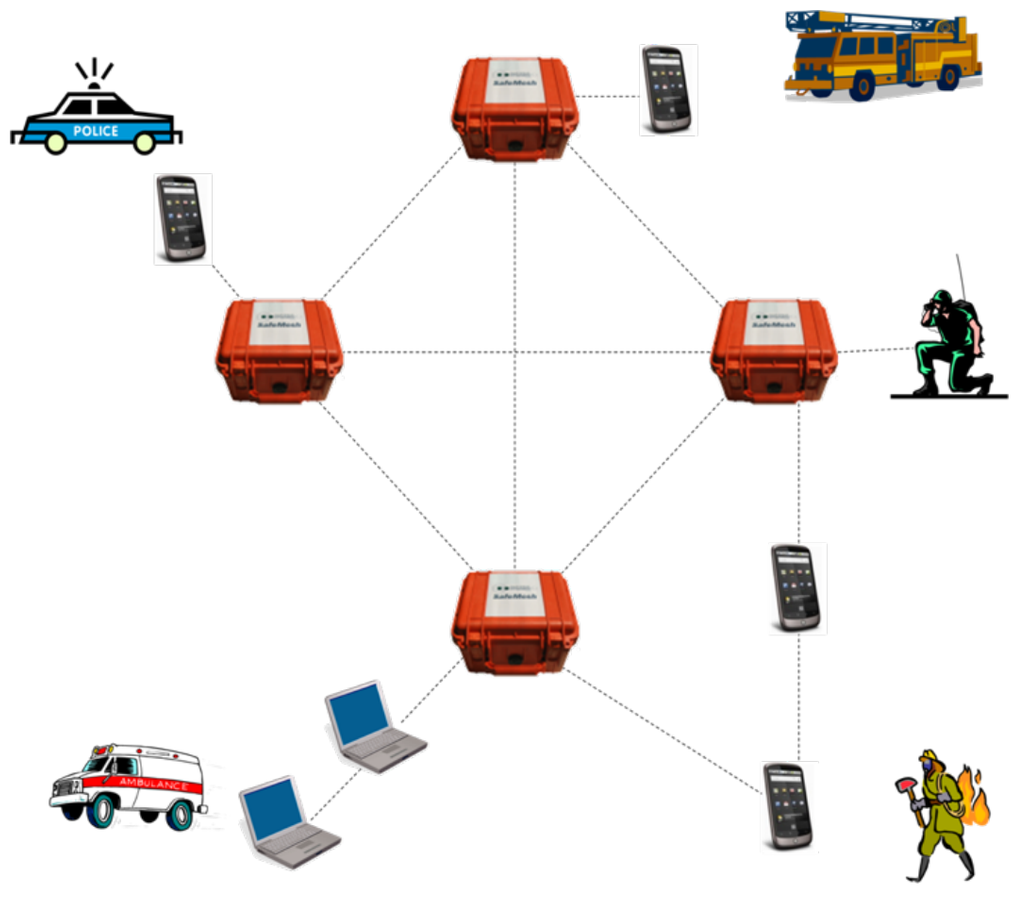
Paphos, Cyprus
October 23, 2012

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

NICTA Members
NICTA Partners

MANETs and WMNs

- Mobile Ad Hoc Networks (MANETs)
  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
  - smart grid
  - London buses
  - ...
- Limitations in reliability and performance



Formal Methods for Mesh Networks

NICTA

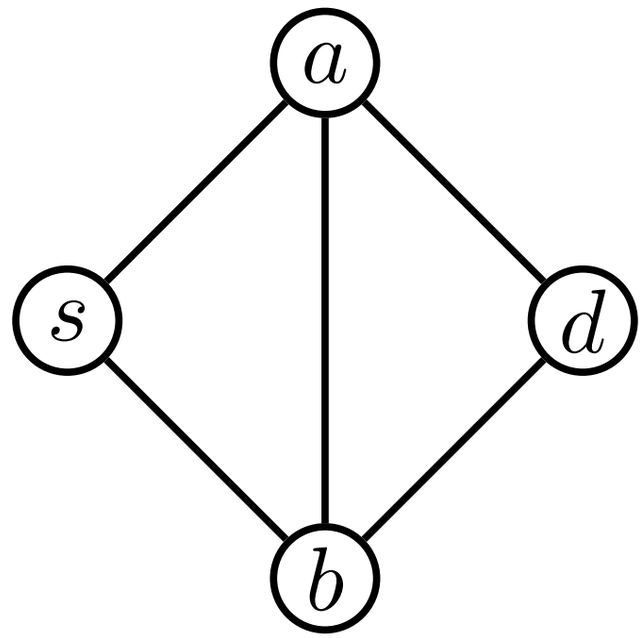
- Goal
  - model, analyse, verify and increase the performance of wireless mesh routing protocols
  - develop suitable formal methods techniques
- Benefits
  - more reliable protocols
  - finding and fixing bugs
  - better performance
  - proving correctness
  - reduce “time-to-market”

AODV

- Ad Hoc On-Demand Distance Vector Protocol
  - Routing protocol for WMNs and MANETs
  - 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 currently standardised by the IETF MANET working group (IEEE 802.11s)

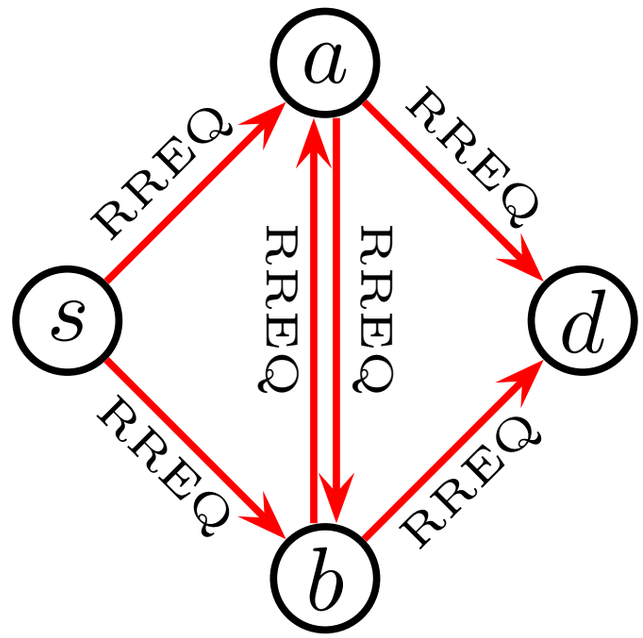
Case Study

- 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



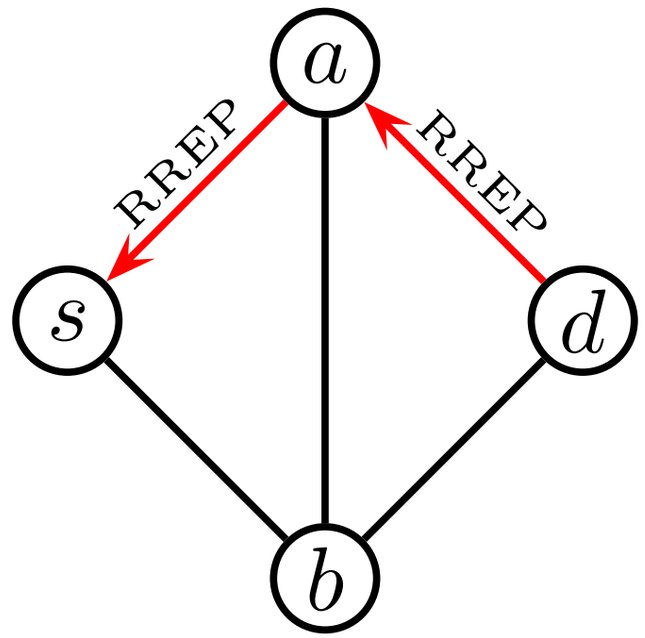
Case Study

- 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



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



Problems

- Standards (IETF RFCs) are not precise
  - written in English
  - ambiguous (sometimes incomplete)
  - no formal specification
- Compliant implementations
  - have different behaviours
  - are not compatible
  - have serious flaws
- Traditional evaluation techniques: simulation and test-bed
  - expensive
  - limited to (a small number of) specific scenarios
  - errors found after years of evaluation
  - barely offer any guarantee for properties such as route discovery

Why Formal Specification?

Why Formal Specification?

Complete and Accurate Formalisation of 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)) .

Proposed Formal Method

- Developed Process Algebra
  - inspired by π-calculus and LOTOS; based on ω-calculus
  - main process expressions

$X(exp_1, \ldots, exp_n)$	process calls
$P + Q$	nondeterministic choice
$[\varphi]P$	if-construct
$[\![\texttt{var} := exp]\!]P$	assignment followed by P
$\mathbf{broadcast}(ms).P$	broadcast message followed by P
$\mathbf{unicast}(dest, ms).P \blacktriangleright Q$	unicast ms to $dest$; if successful proceed with P; otherwise with Q
$\mathbf{receive}(\texttt{msg}).P$	receive message

Requirements for Formal Methods

- "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 Requirements (for formal methods)
  - relatively easy to extract code
  - complete specification
  - implementation independent

- Easy to use
  - only a few (well-known) programming constructs

Rigorous Analysis of AODV

NICTA

- Achievements
  - full concise specification of AODV (RFC 3561)
    - 6 processes (~140 lines; instead of 40 pages English prose)
    - without time
  - verified/disproved properties
    - route discovery
    - packet delivery
    - loop freedom
      - first (correct) proof
      - disproved loop freedom for variants of AODV (implemented in at least 3 open source implementations, e.g. AODV-ns2)
  - found several ambiguities, contradictions, shortcomings
  - found solutions for some limitations

Rigorous Analysis

- Proofs
  - independent of topology
    - no need for specific scenarios (e.g., Random Way Point Model)
    - without having to simulate each and every network topology
  - modularity / reusability
    - line-by-line analysis
    - mainly based on invariants
    - loop-freedom proof based on sequence of smaller results
  - analysed 432 “reasonable” interpretations (112 are loop free and “correct”)
  - even some interpretations we never thought about

- simulation and test-bed experiment would have to be repeated for each interpretation
  - only a few topologies can be checked

Ambiguity

- One of the Ambiguities (RFC3561)

```
sequence number field is set to false.  The route is only updated if
the new sequence number is either

(i)       higher than the destination sequence number in the route
          table, or

(ii)      the sequence numbers are equal, but the hop count (of the
          new information) plus one, is smaller than the existing hop
          count in the routing table, or

(iii)     the sequence number is unknown.
```

Analysing Variants of AODV

NICTA

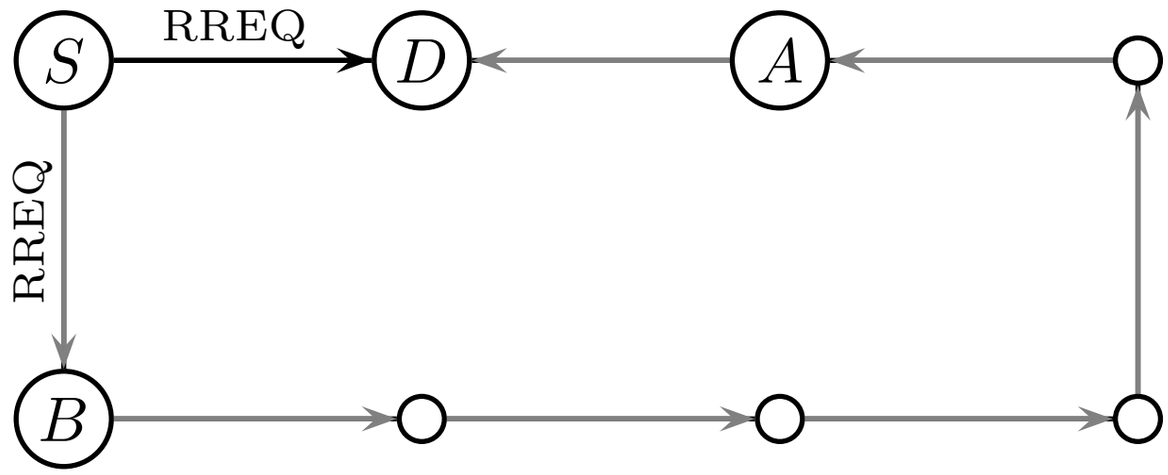
$+$ [$(\texttt{oip},\texttt{rreqid}) \notin \texttt{rreqs}$] /* the RREQ is new to this node */
⟦$\texttt{rt} := \texttt{update}(\texttt{rt},(\texttt{oip},\texttt{osn},\texttt{kno},\texttt{val},\texttt{hops}+1,\texttt{sip},\emptyset))$⟧ /* update the route to oip in rt */
⟦$\texttt{rreqs} := \texttt{rreqs} \cup \{(\texttt{oip},\texttt{rreqid})\}$⟧ /* 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)) .

Analysing Variants of 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)) .
          unicast(nhop(rt,dip),rrep(hops + 1,oip,osn,dip,ip)).

Non-Optimal Route Selection

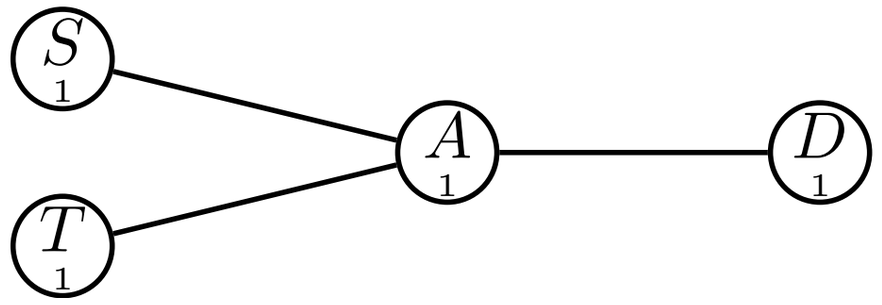


- during route discovery *only* nodes lying on route from source to destination find optimal routes

[MiskovicKnightly10]

- modification: forward route request
  - only a few more messages on average
  - specification changes in **5 lines** only
  - invariants to be checked only for these lines

Failure of Route Discovery Process

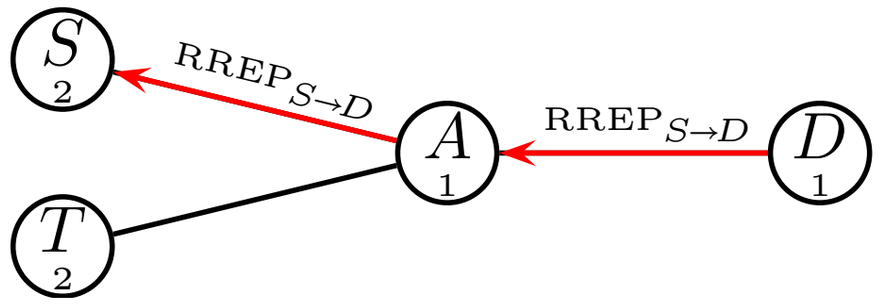


- route replies are dropped if they do not carry new information; this might yield route discovery failure

[IETF Mailing List]

- modification: forward route request
  - only a few more messages in average
  - specification changes in **1 line** only (2 lines can be dropped)
  - invariants to be checked only for this line (it is easy to adapt the proof)

Failure of Route Discovery Process

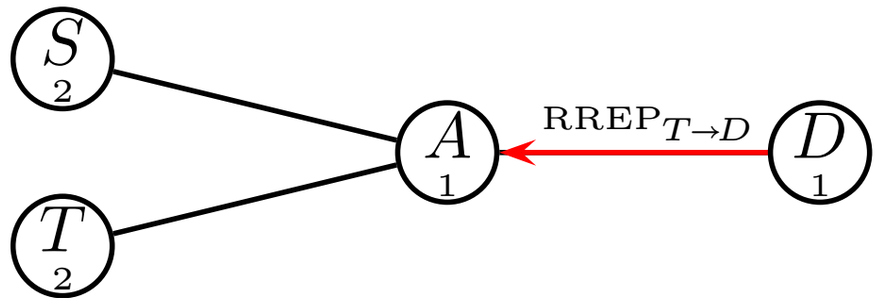


- route replies are dropped if they do not carry new information; this might yield route discovery failure

[IETF Mailing List]

- modification: forward route request
  - only a few more messages in average
  - specification changes in **1 line** only (2 lines can be dropped)
  - invariants to be checked only for this line (it is easy to adapt the proof)

Failure of Route Discovery Process



- route replies are dropped if they do not carry new information; this might yield route discovery failure

[IETF Mailing List]

- modification: forward route request
  - only a few more messages in average
  - specification changes in **1 line** only (2 lines can be dropped)
  - invariants to be checked only for this line
    (it is easy to adapt the proof)

Conclusion

- So far concentrated on AODV
  - well known
  - IETF standard
- Full concise specification of AODV (RFC 3561)
  - verified/disproved properties (independent of topologies)
  - modular proofs
  - found several ambiguities, contradictions, shortcomings
  - adapted proofs to verify interpretations and variants of AODV

Future Work

- Extend formal methods to other protocols
  - OSLR, B.A.T.M.A.N., ...
- Add further necessary concepts
  - time
  - probability (links, (quantitative) measurements)
  - quality

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

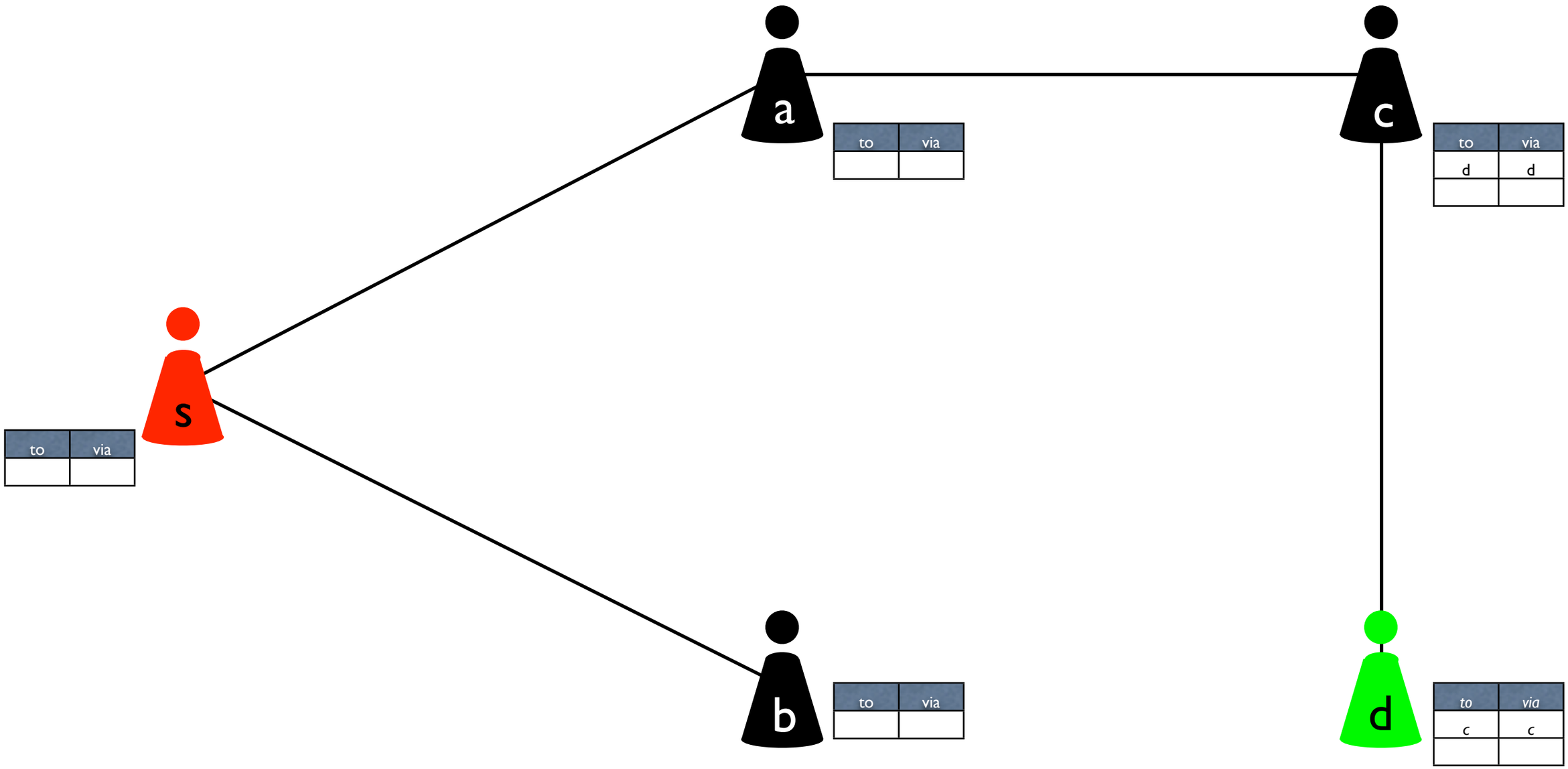
NICTA

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

NICTA

DEMO

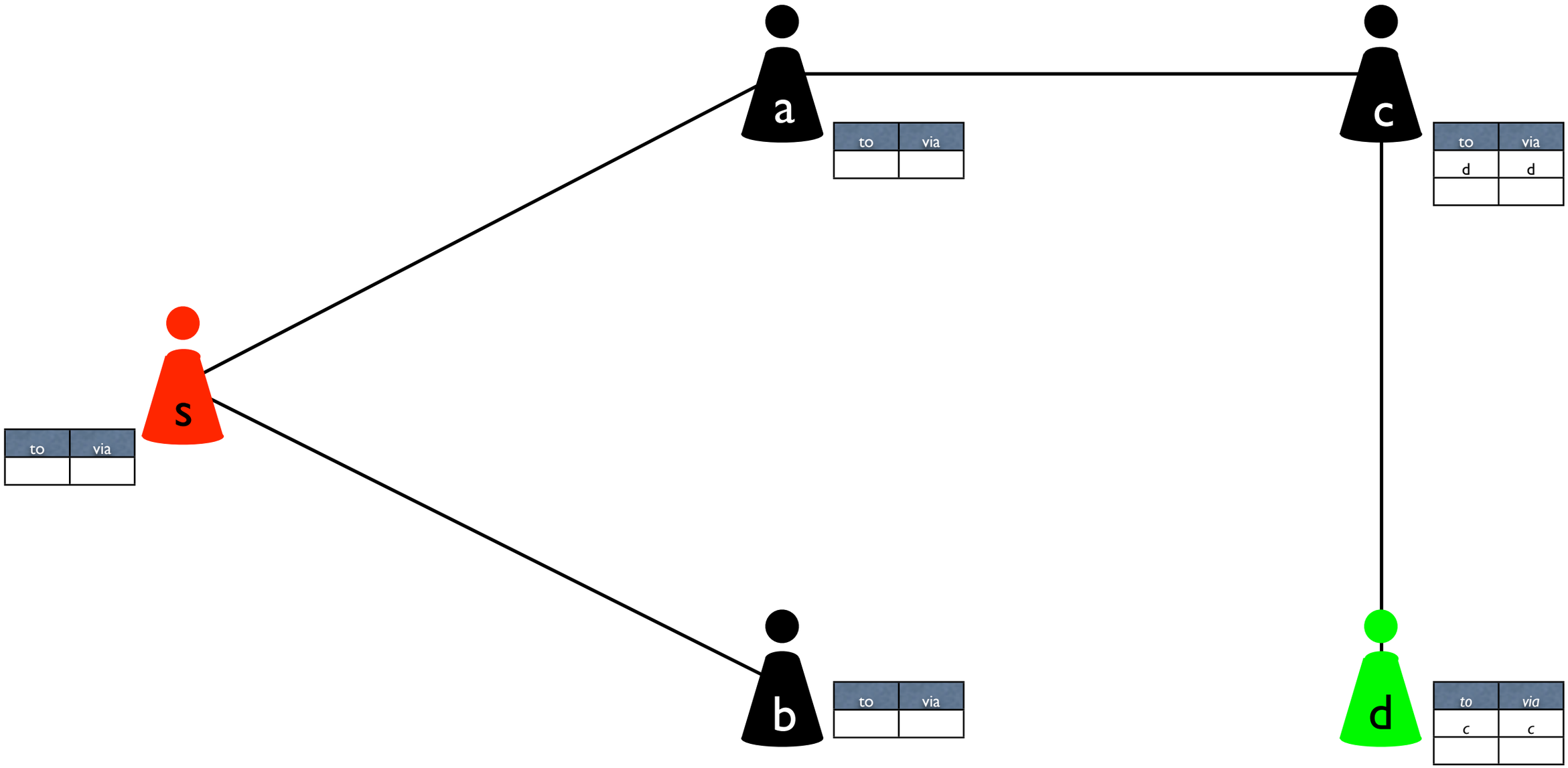
AODV – An Example



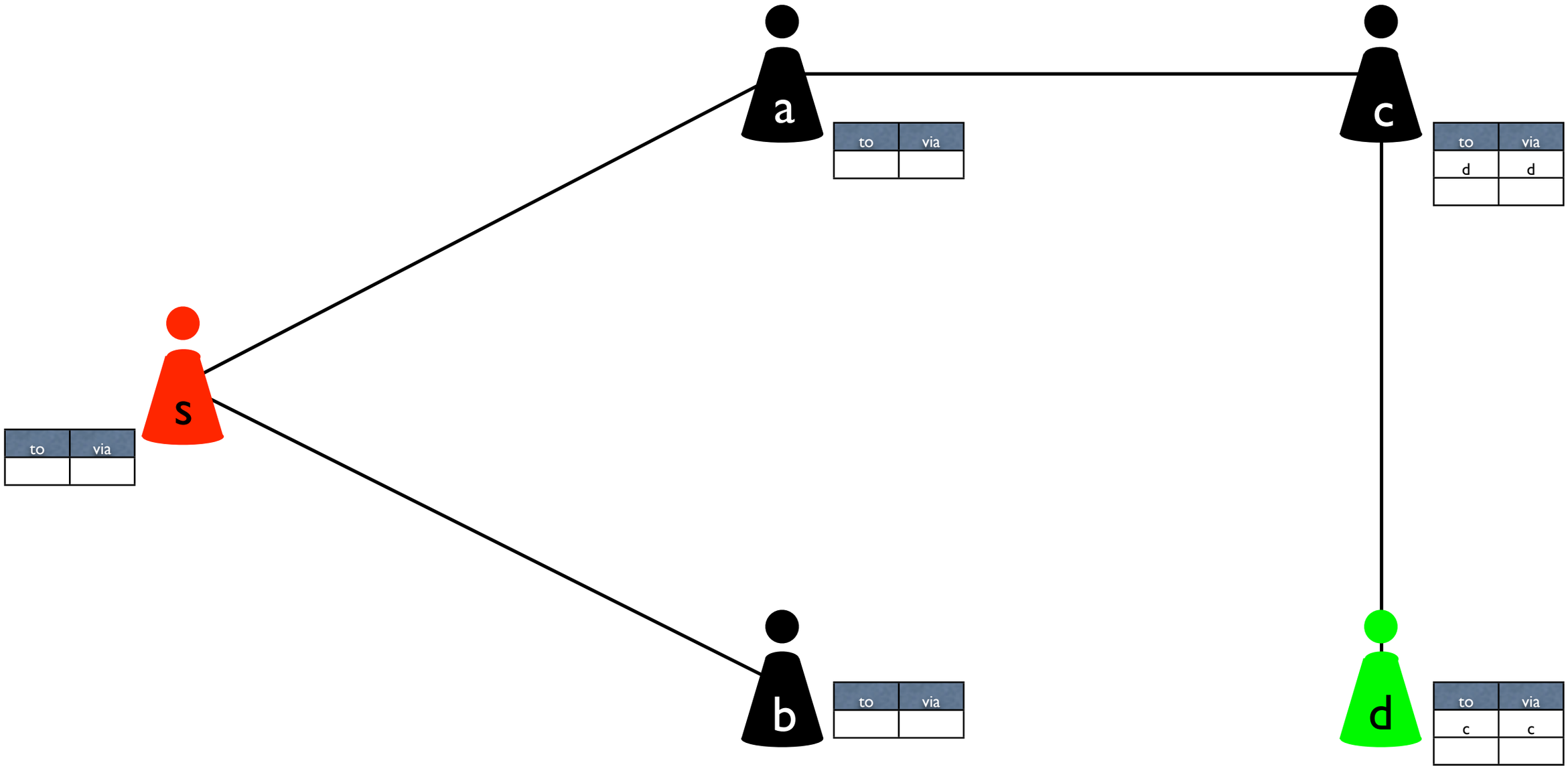
s is looking for a route to d

AODV – An Example

NICTA

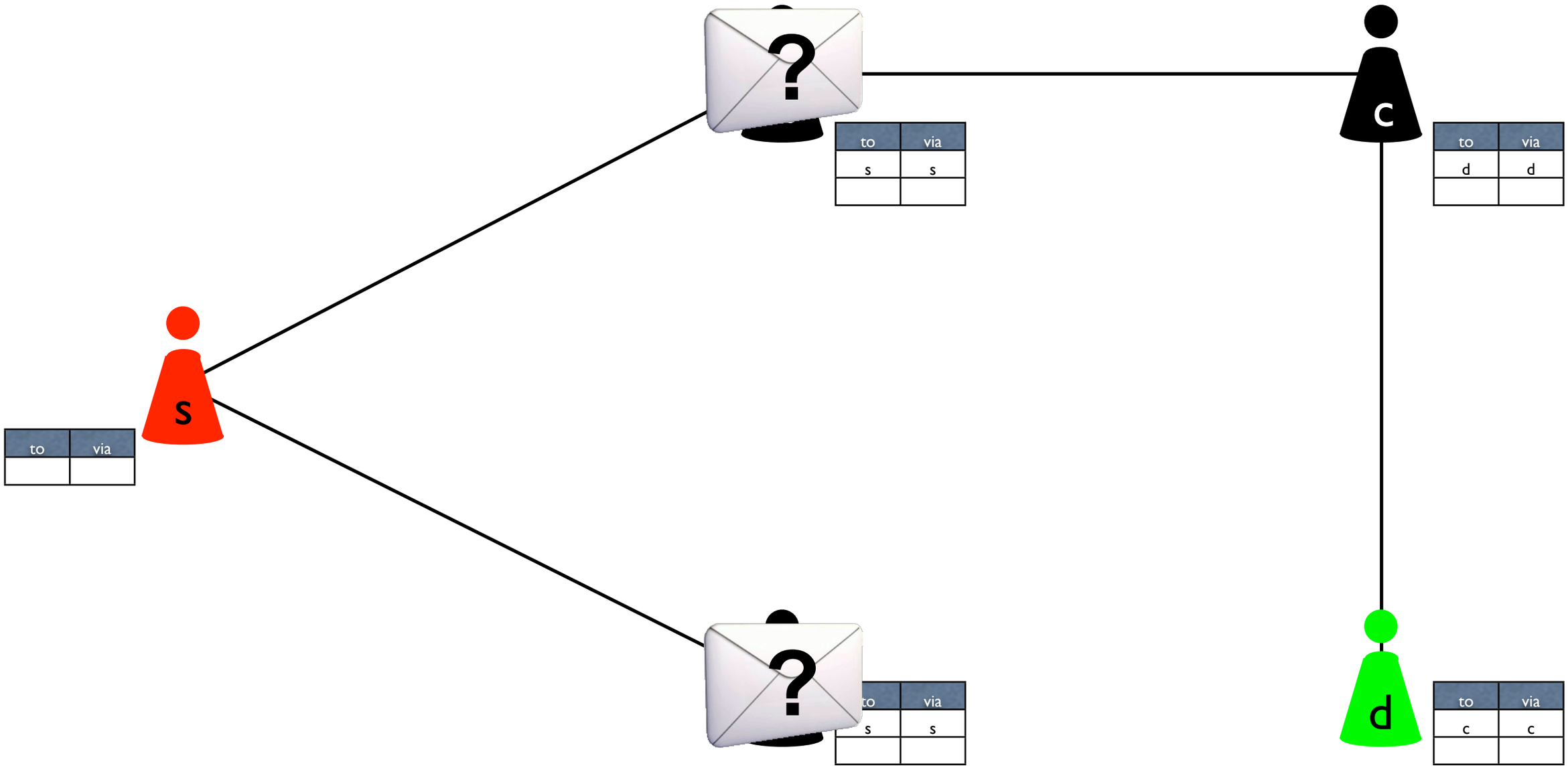


AODV – An Example



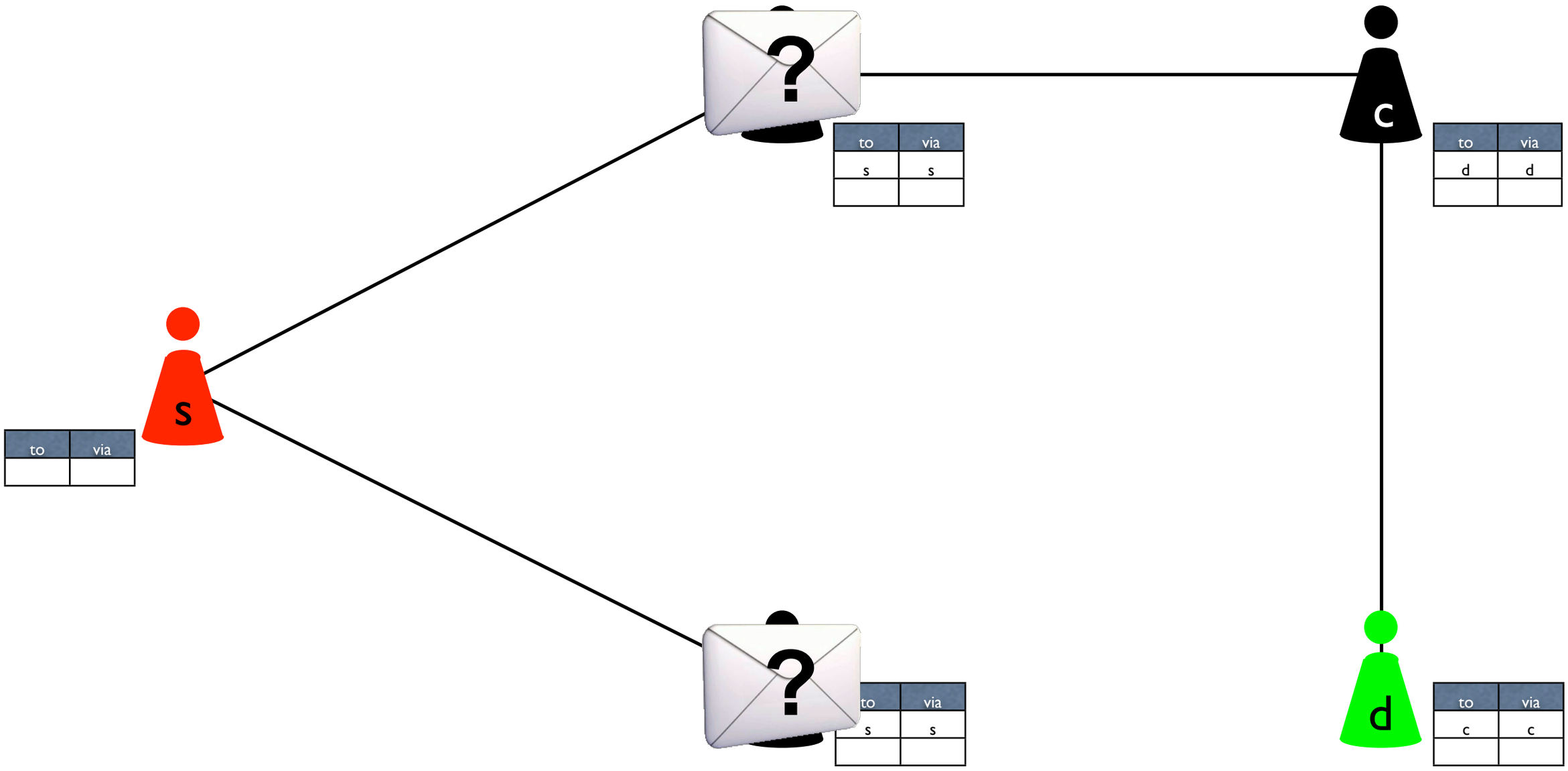
AODV – An Example

NICTA



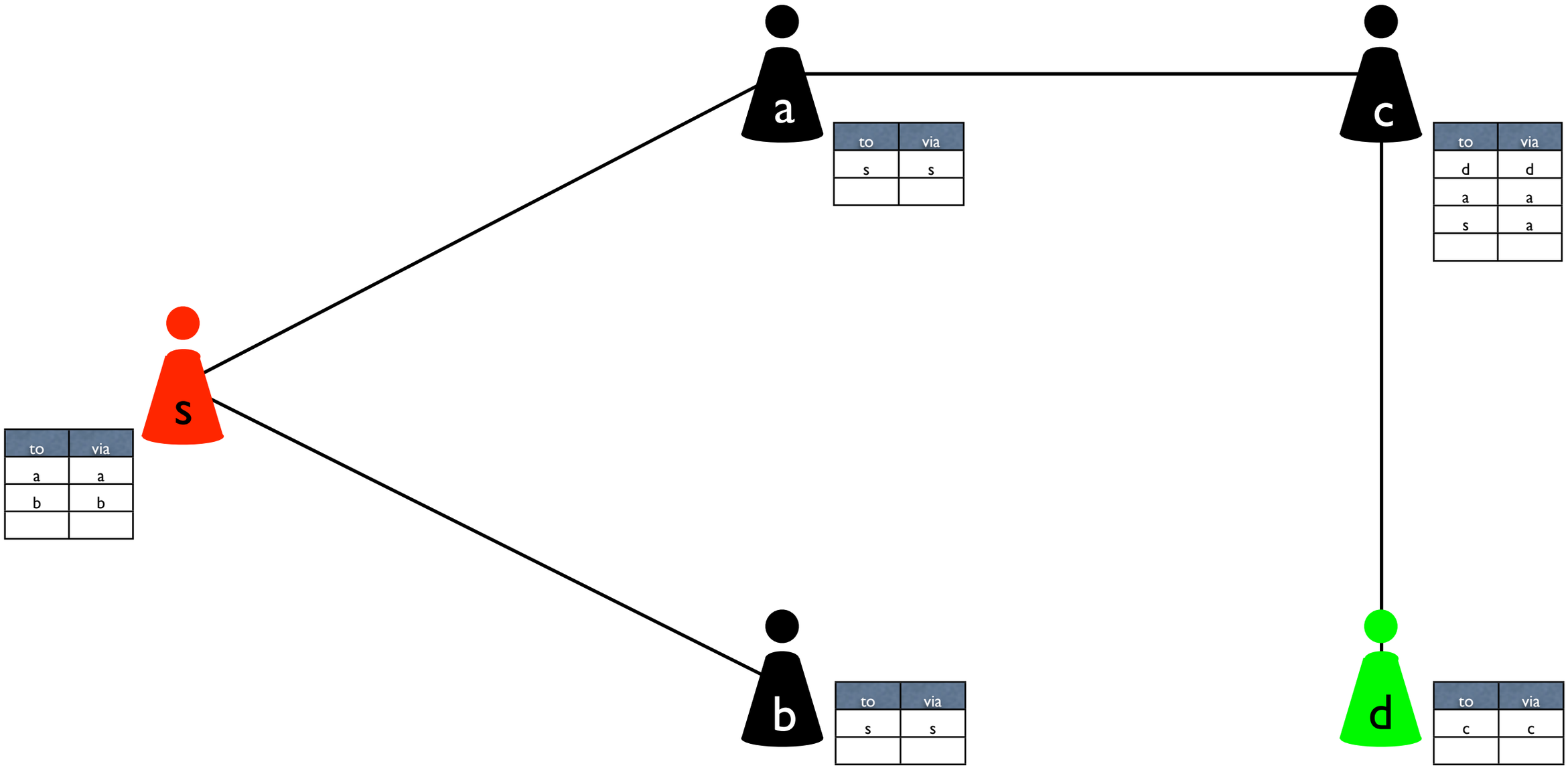
AODV – An Example

NICTA



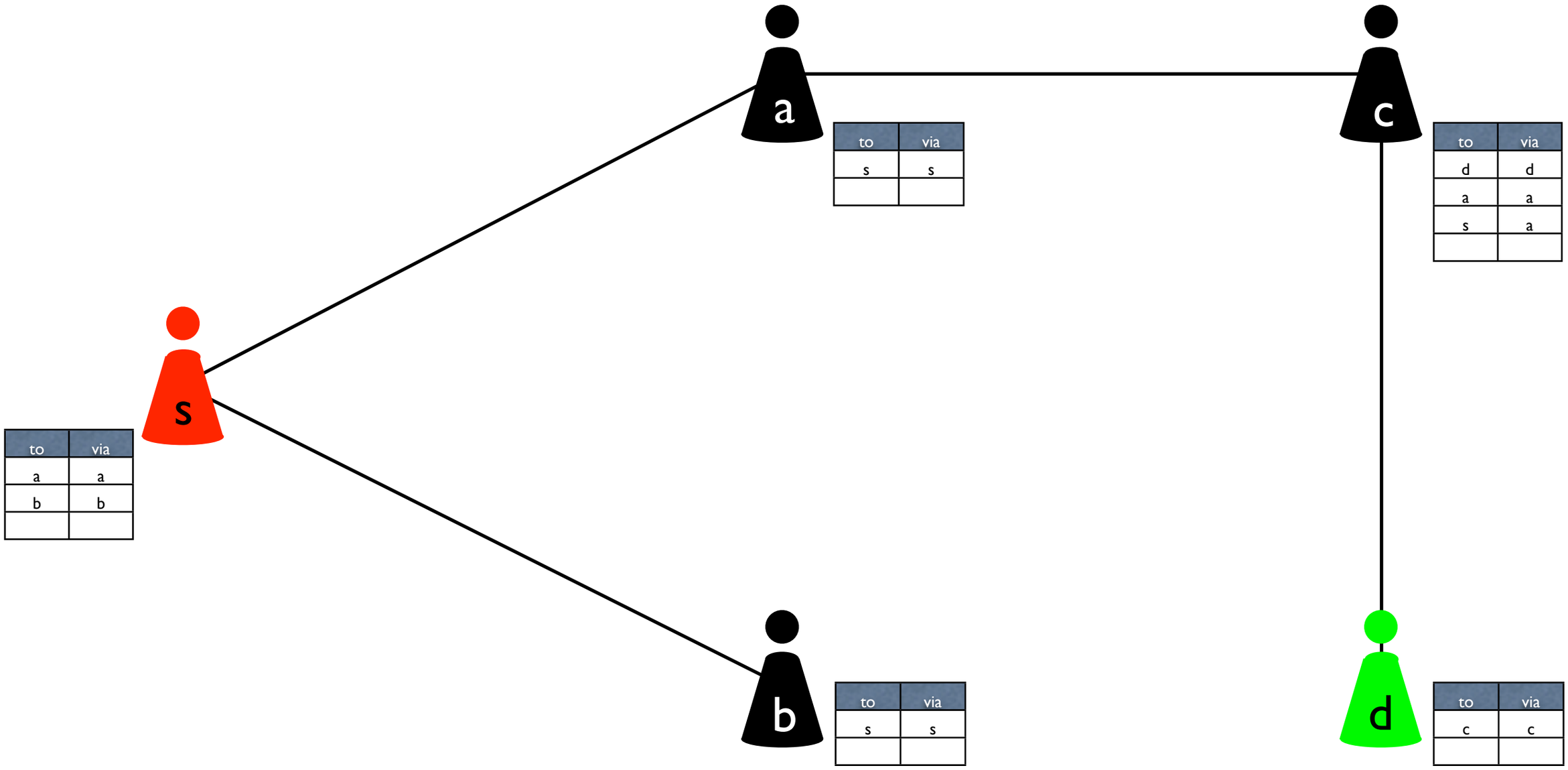
AODV – An Example

NICTA



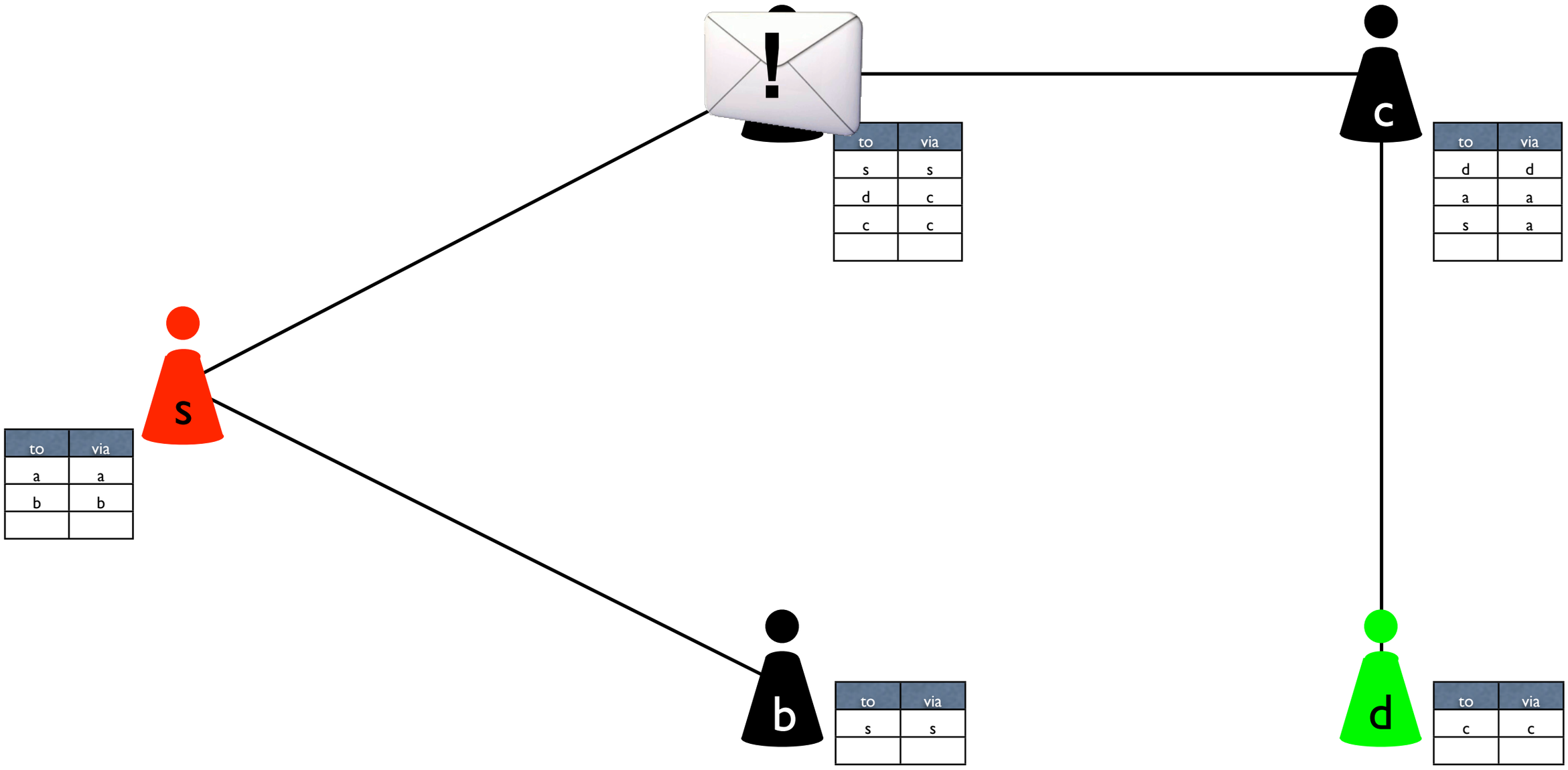
AODV – An Example

NICTA

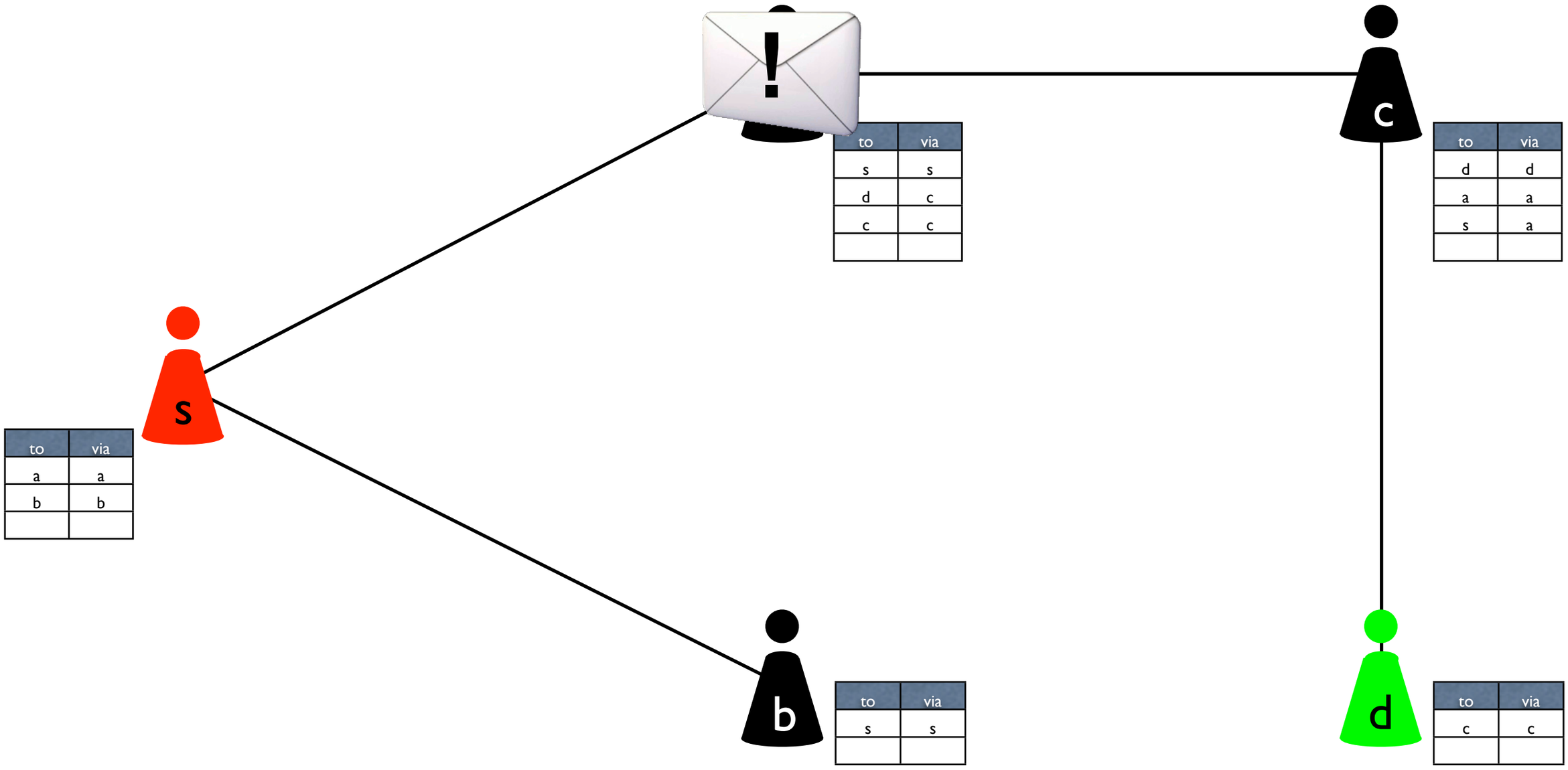


AODV – An Example

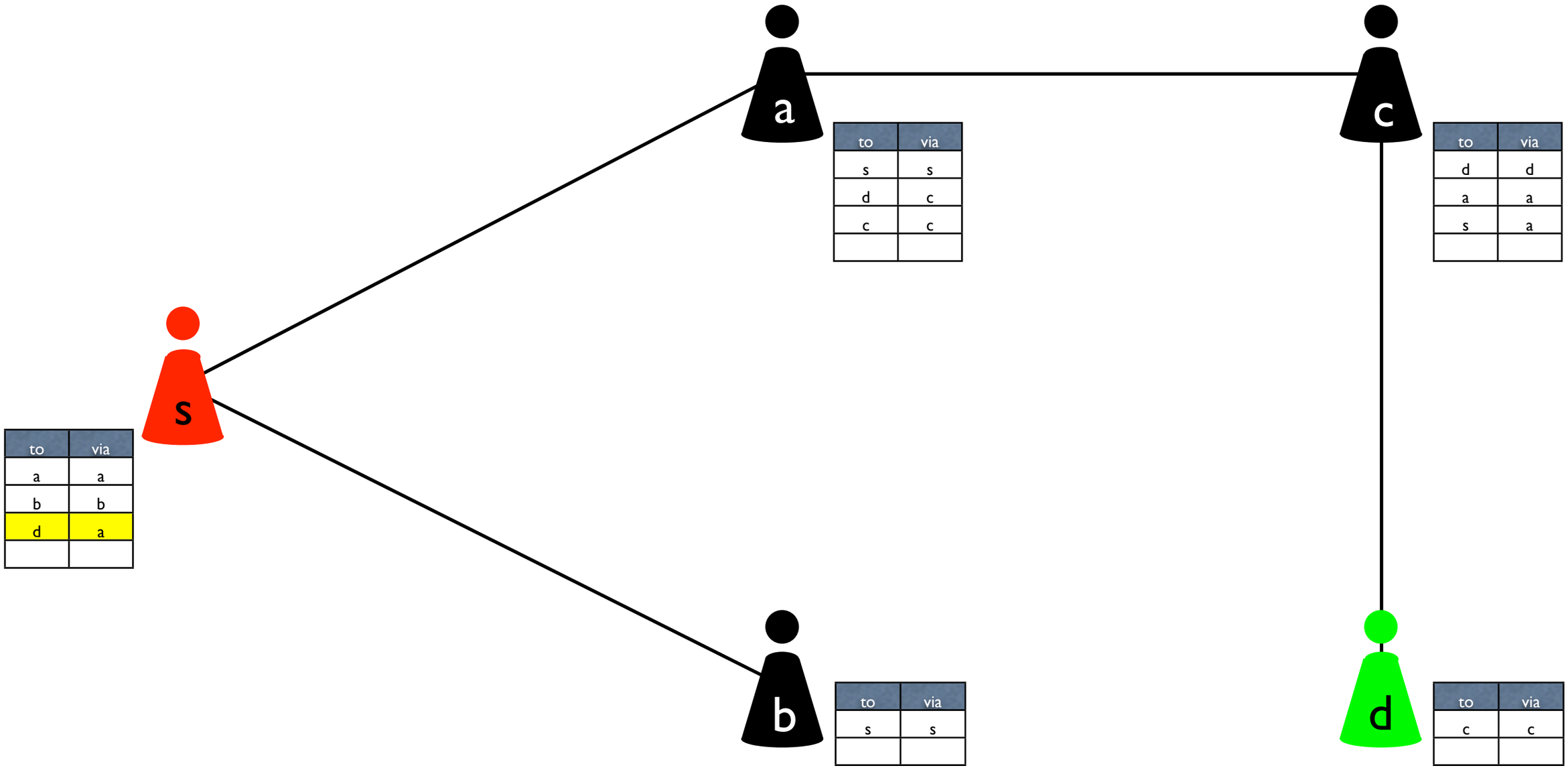
NICTA



AODV – An Example

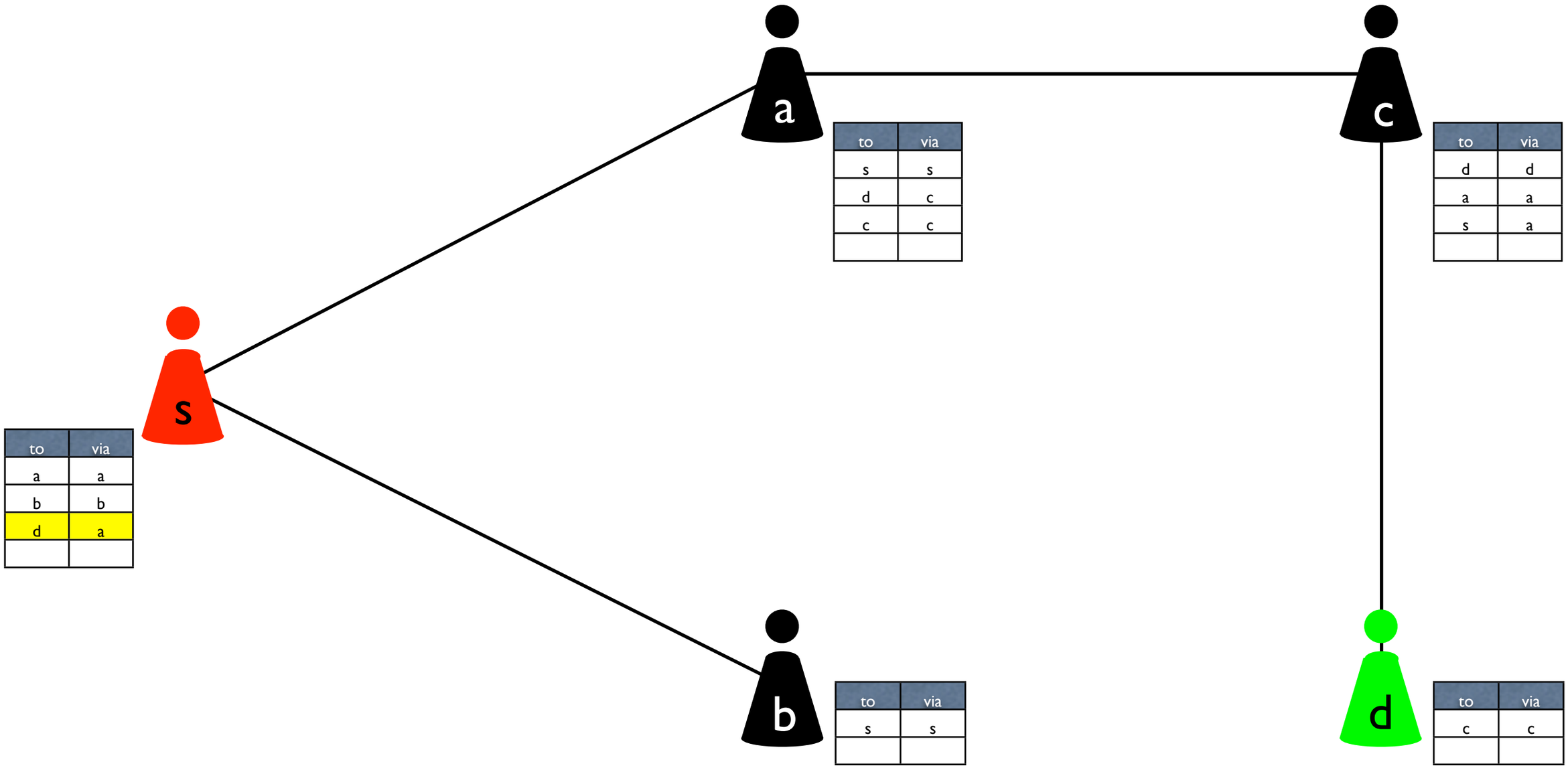


AODV – An Example

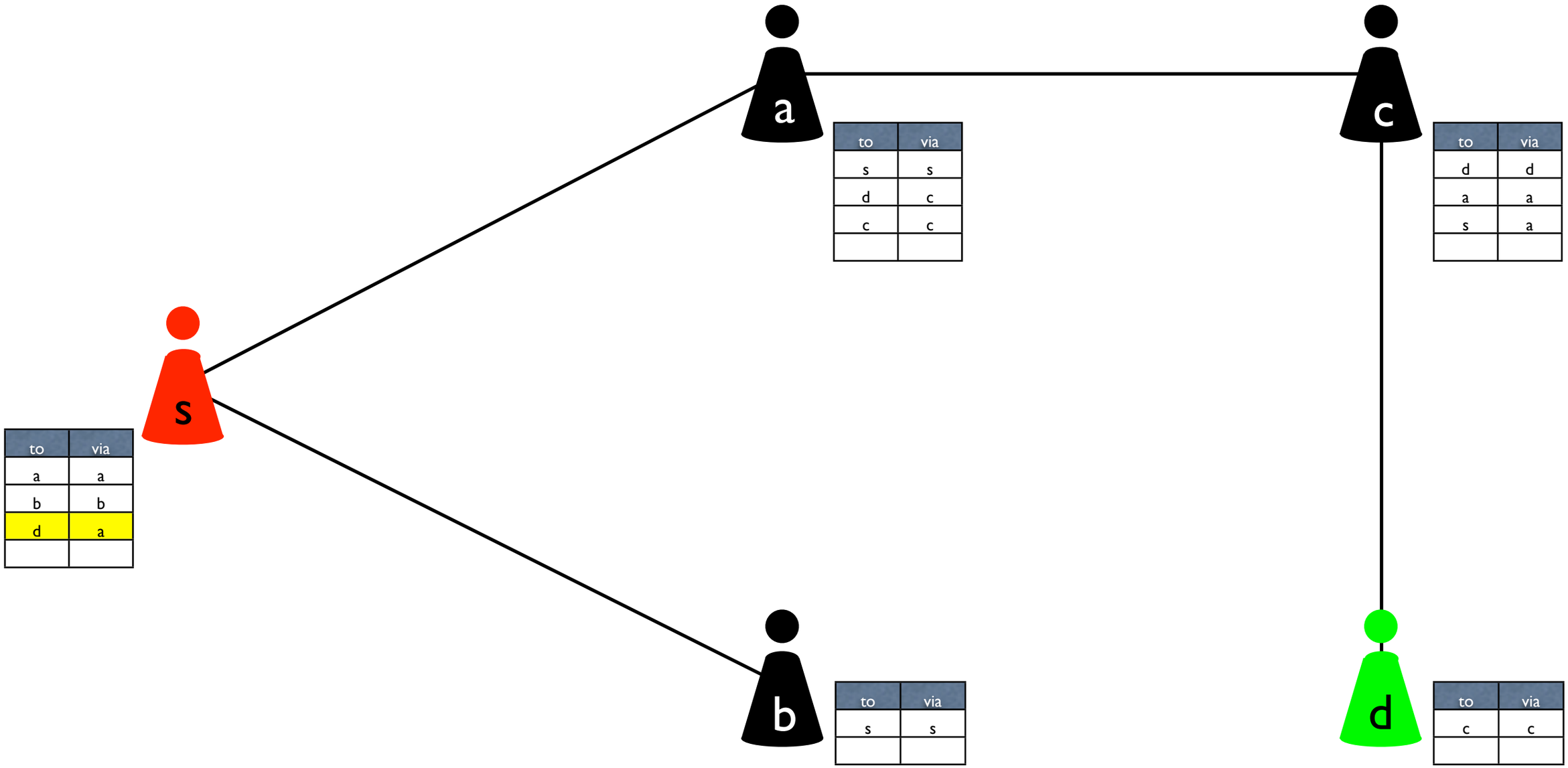


AODV – An Example

NICTA



AODV – An Example



s has found a route to d

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)

Update Function

$$\texttt{update}(rt,r) := \begin{cases} rt \cup \{r\} & \text{if } \pi_1(r) \notin \texttt{kD}(rt) \quad //r \text{ is new} \\ nrt \cup \{nr\} & \text{if } \texttt{sqn}(rt,\pi_1(r)) < \pi_2(r) \quad //\text{fresher} \\ nrt \cup \{nr\} & \text{if } \texttt{sqn}(rt,\pi_1(r)) = \pi_2(r) \\ & \quad \wedge\, \texttt{dhops}(rt,\pi_1(r)) > \pi_4(r) \quad //\text{shorter} \\ nrt \cup \{nr\} & \text{if } \texttt{sqn}(rt,\pi_1(r)) = \pi_2(r) \quad //\text{replaces} \\ & \quad \wedge\, \texttt{flag}(rt,\pi_1(r)) = \texttt{inv} \quad \text{invalid} \\ nrt \cup \{nr'\} & \text{if } \pi_2(r) = 0 \quad //\text{unk. sqn} \\ nrt \cup \{ns\} & \text{otherwise ,} \end{cases}$$

NICTA

Pro. 1, Lines 10, 14, 18: The entry $(\texttt{sip}, 0, \texttt{val}, 1, \texttt{sip}, \emptyset)$ is used for the update; its destination is $dip := \texttt{sip}$. We assume this entry is actually inserted in the routing table of ip. Since $dip = \texttt{sip} = \texttt{nhop}_N^{ip}(dip) = nhip$, the antecedent of the invariant to be proven is not satisfied.

Pro. 2, Line 5: The entry $(\texttt{oip}, \texttt{osn}, \texttt{val}, \texttt{hops}+1, \texttt{sip}, *)$ is used for the update; again we assume it is inserted into the routing table of node ip. So $dip := \texttt{oip}$, $nhip := \texttt{sip}$, $\texttt{nsqn}_N^{ip}(dip) := \texttt{osn}$ and $\texttt{dhops}_N^{ip}(dip) := \texttt{hops}+1$. This information is distilled from a received route request message (cf. Lines 1 and 8 of Pro. 1). By Proposition 7.1 of [8], this message was sent before, say in state N'; by Proposition 7.8 of [8], the sender of this message has identified itself correctly, and is $\texttt{sip}$.