Formal Methods for Wireless Mesh Networks

Peter Höfner

NICTA

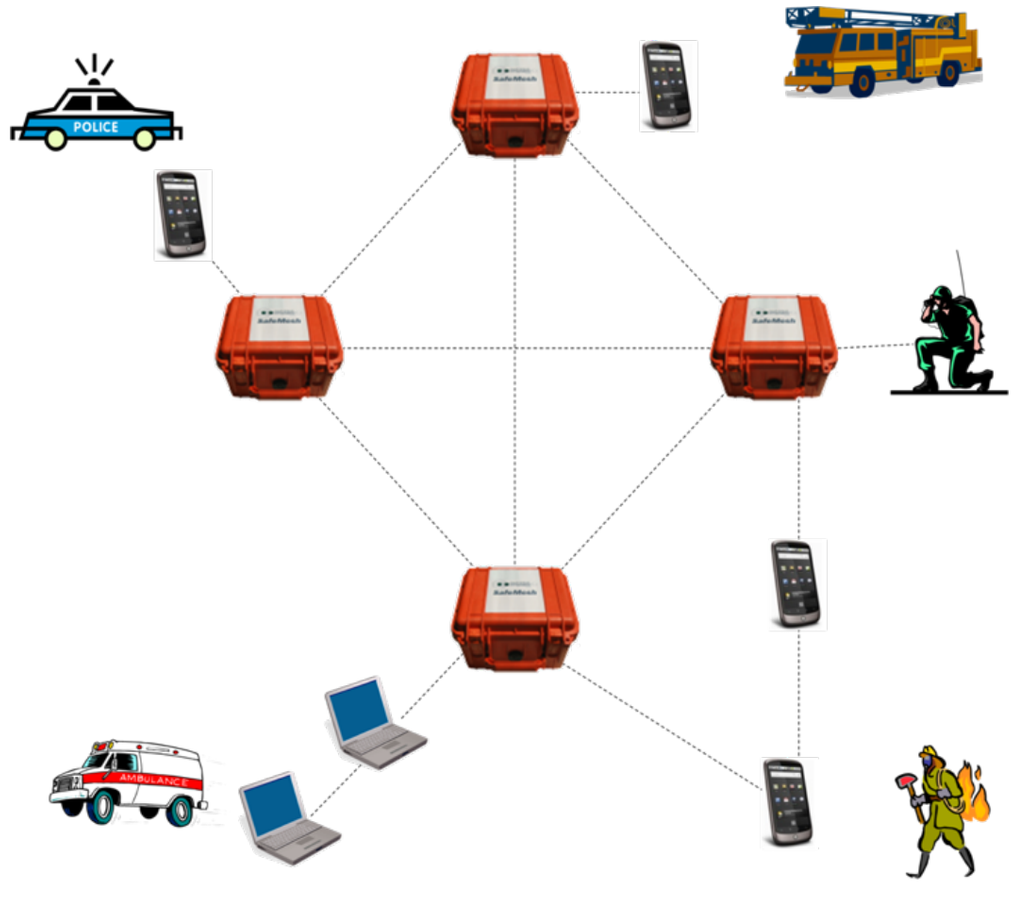
UNSW
THE UNIVERSITY OF NEW SOUTH WALES
SYDNEY · CANBERRA · AUSTRALIA

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

NICTA Members
ANU THE AUSTRALIAN NATIONAL UNIVERSITY
UNSW THE UNIVERSITY OF NEW SOUTH WALES
Department of State and Regional Development
Victoria The Place To Be
The University of Melbourne
The University of Sydney
Queensland Government
Griffith University
QUT
The University Of Queensland
NICTA Partners

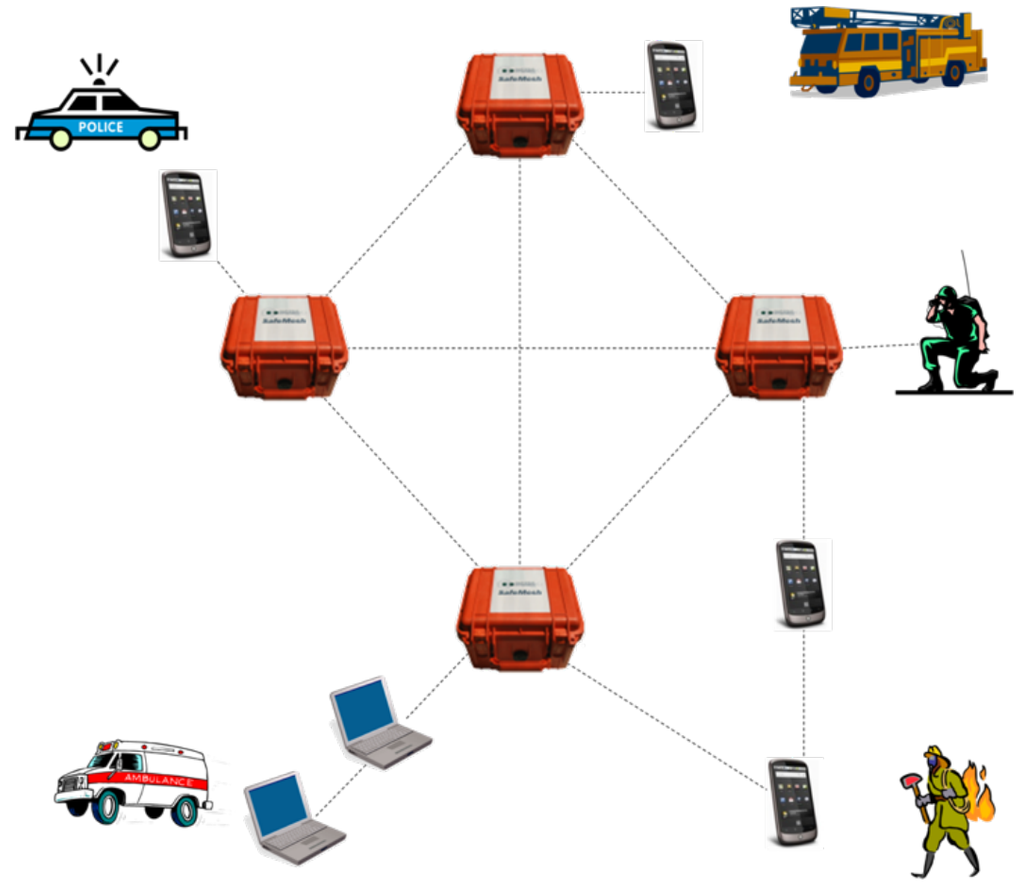
What is the Problem?

- Wireless Mesh Networks
  - key advantage: no backhaul wiring required
  - quick and low cost deployment
- Applications
  - public safety (e.g. CCTV)
  - emergencies (e.g. earthquakes)
  - mobile phone services
  - transportation
  - mining
  - military actions/counter terrorism
  - ...



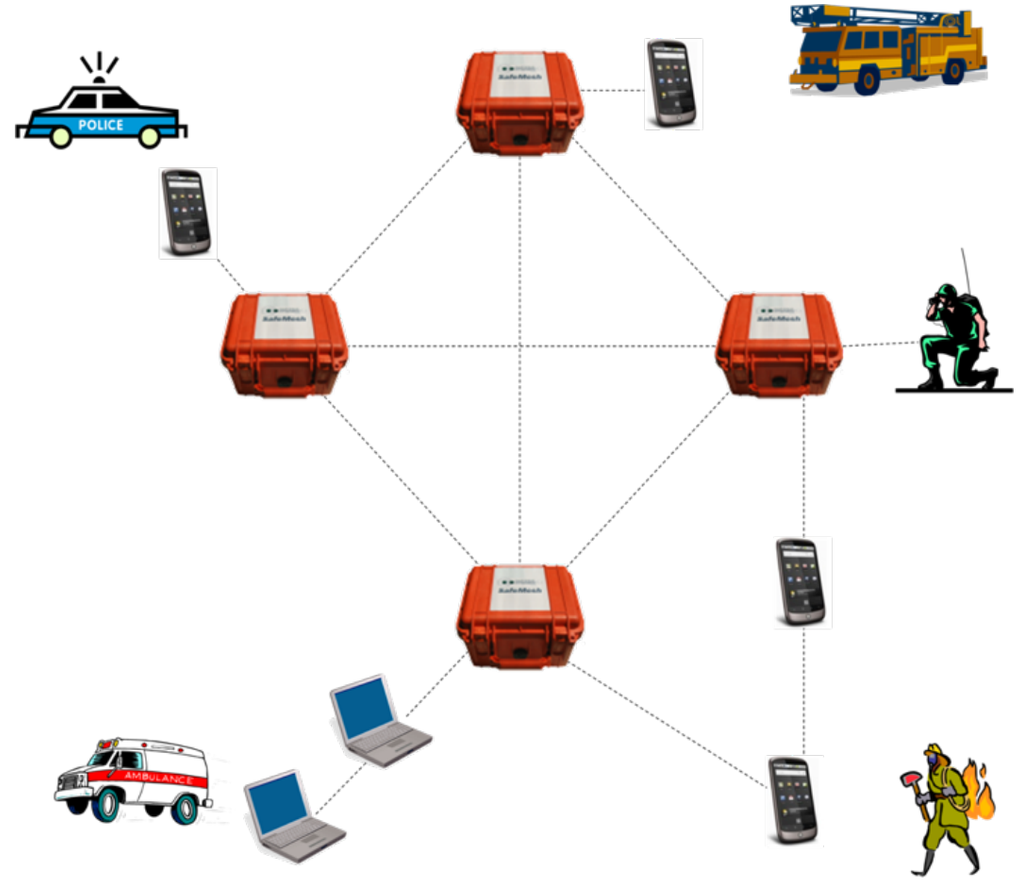
What is the Problem?

- 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
- **THIS IS NOT TRUE** (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

NICTA

- Goal
  - model, analyse, verify 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”
- Team (Formal Methods)
  - Ansgar Fehnker, Rob van Glabbeek, Peter Höfner, Annabelle McIver, Marius Portmann, Wee Lum Tan

Formal Methods for Mesh Networks

NICTA

- Main Methods used so far
  - process algebra
  - model checking
  - routing algebra

Current Protocols

Network

(textual) Specification

... 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.

This route MAY now be used to send any queued data packets and fulfills any outstanding route requests. ...

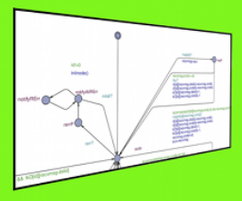
(RFC 3561)

Formal Methods

Process Calculus

/* update the sqn of ip by setting it to max(sqn(rt, ip), dsn) */
⟦rt := update(rt, (ip, dsn, valid, 0, ip, 0))⟧
/* 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)

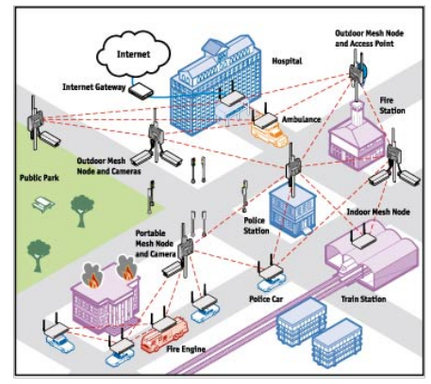
Model Checking



Algebra of Routing Tables

$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{pmatrix} (\varepsilon,0) & (B,1) & (B,2) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ (A,1) & (\varepsilon,0) & (C,1) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ (\varepsilon,\infty) & (B,1) & (\varepsilon,0) & (\varepsilon,\infty) & (E,1) \\ (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,0) & (E,1) \\ (\varepsilon,\infty) & (\varepsilon,\infty) & (C,1) & (D,1) & (\varepsilon,0) \end{pmatrix} \end{array}$$

Better Protocols



Ad Hoc On-Demand Distance Vector Protocol

NICTA

- Routing protocol for WMNs

- 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)

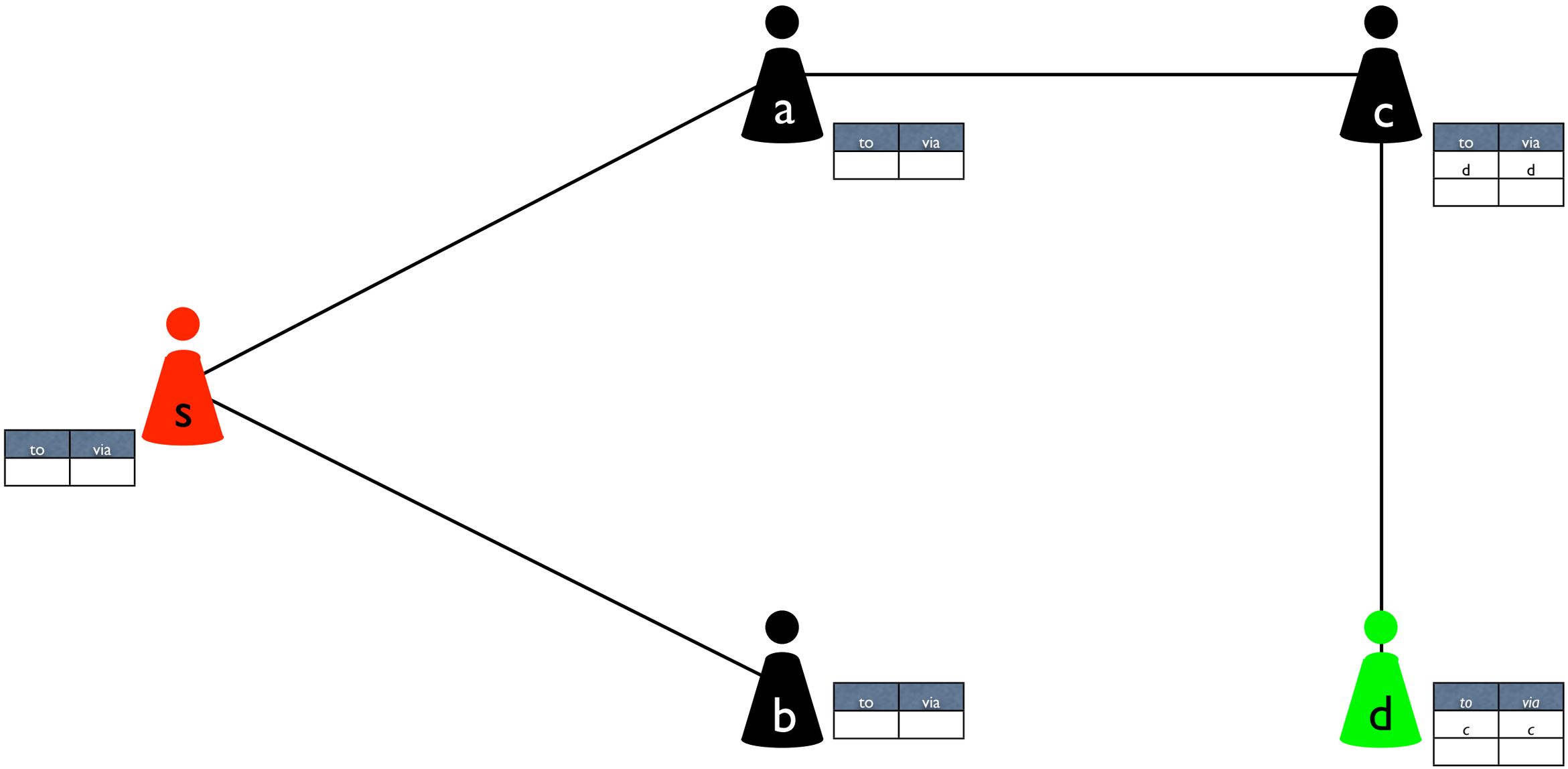
Ad Hoc On-Demand Distance Vector Protocol

NICTA

- AODV control messages
  - route request (RREQ)
  - route reply (RREP)
  - route error message (RERR)

- Information at nodes
  - own IP address
  - a local sequence number (freshness/timer)
  - a routing table
    - local knowledge
    - entries: $(\texttt{dip}, \texttt{dsn}, \texttt{val}, \texttt{hops}, \texttt{nhip}, \texttt{pre})$

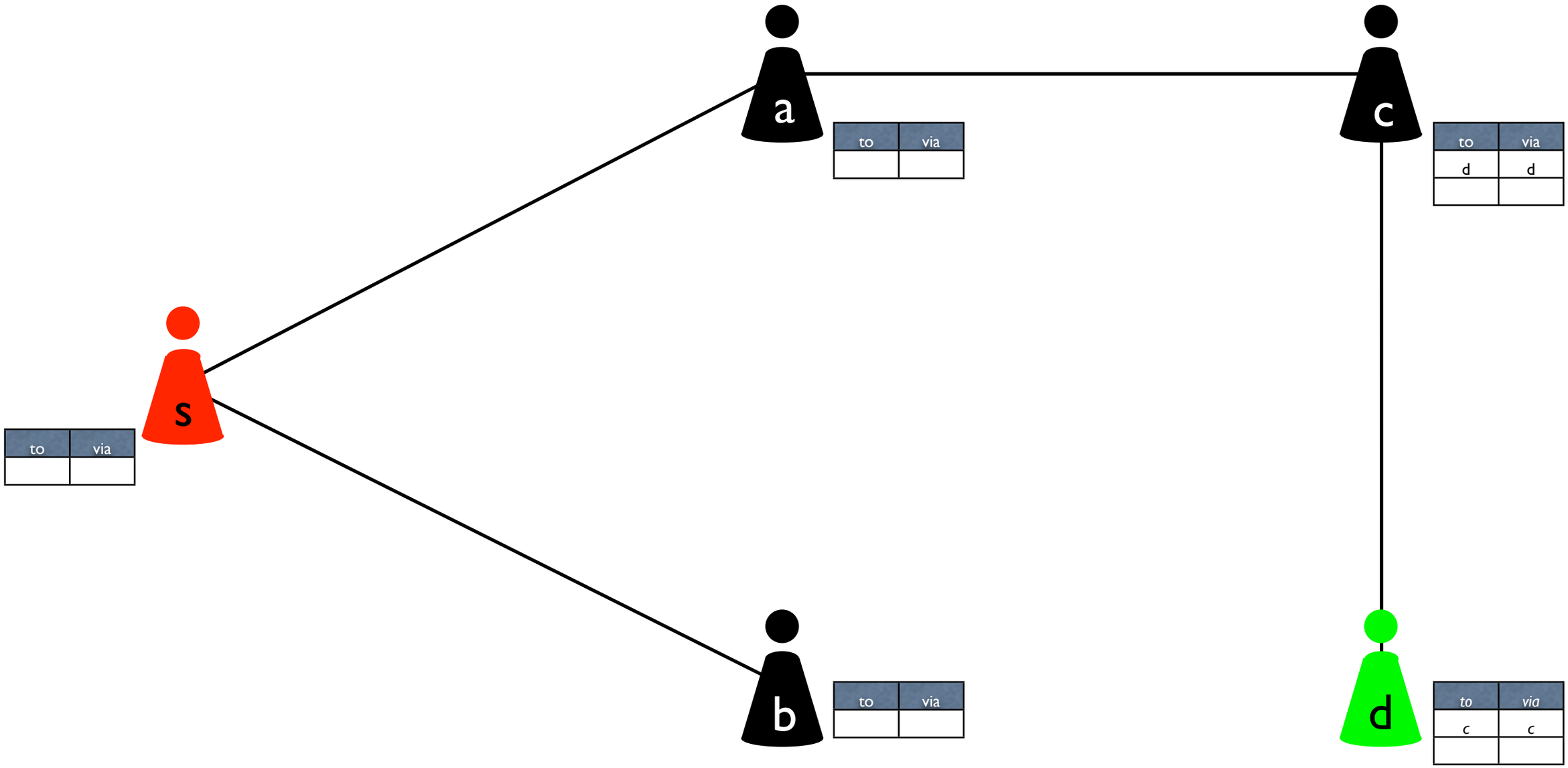
AODV – An Example



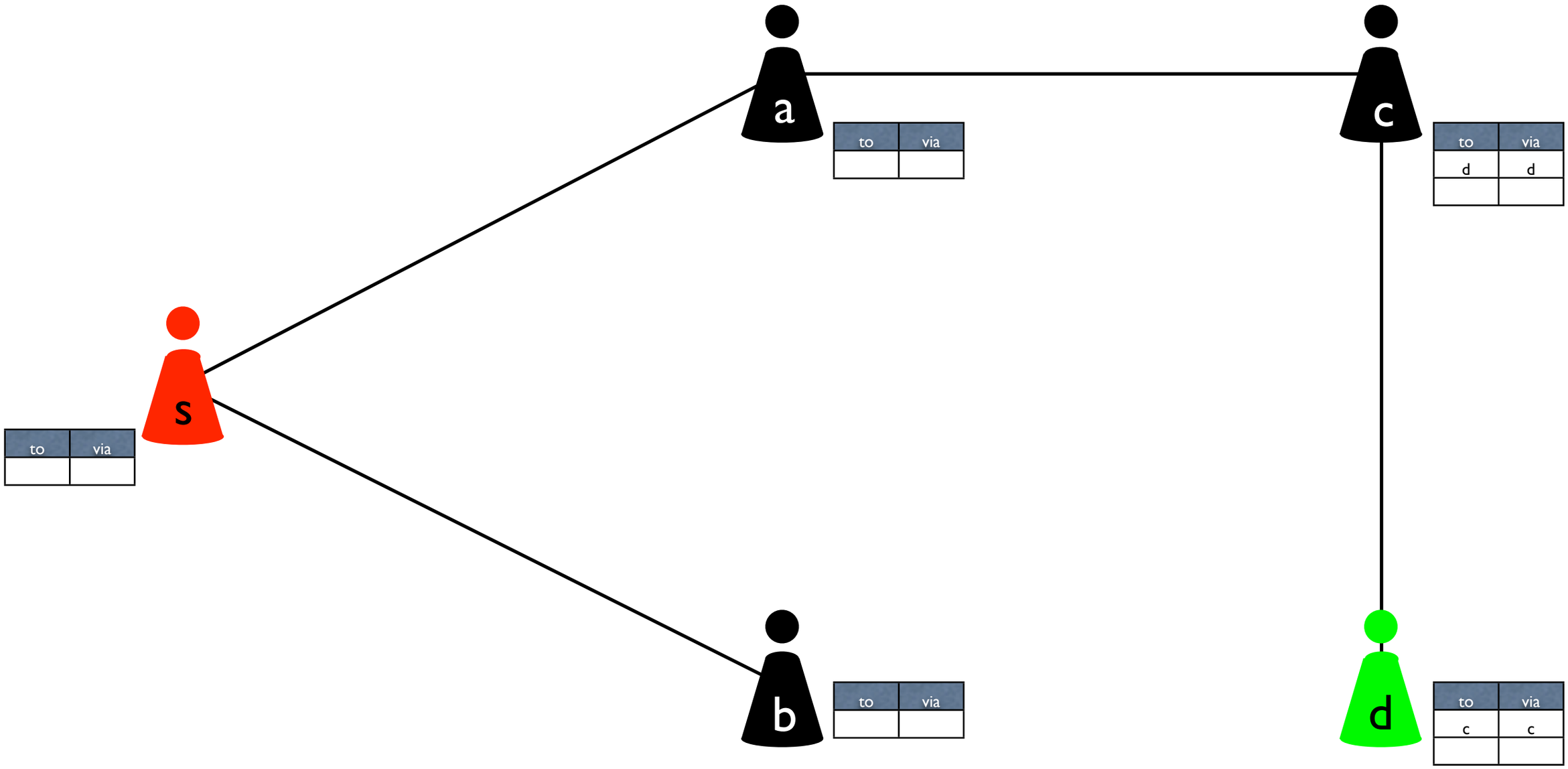
s is looking for a route to d

AODV – An Example

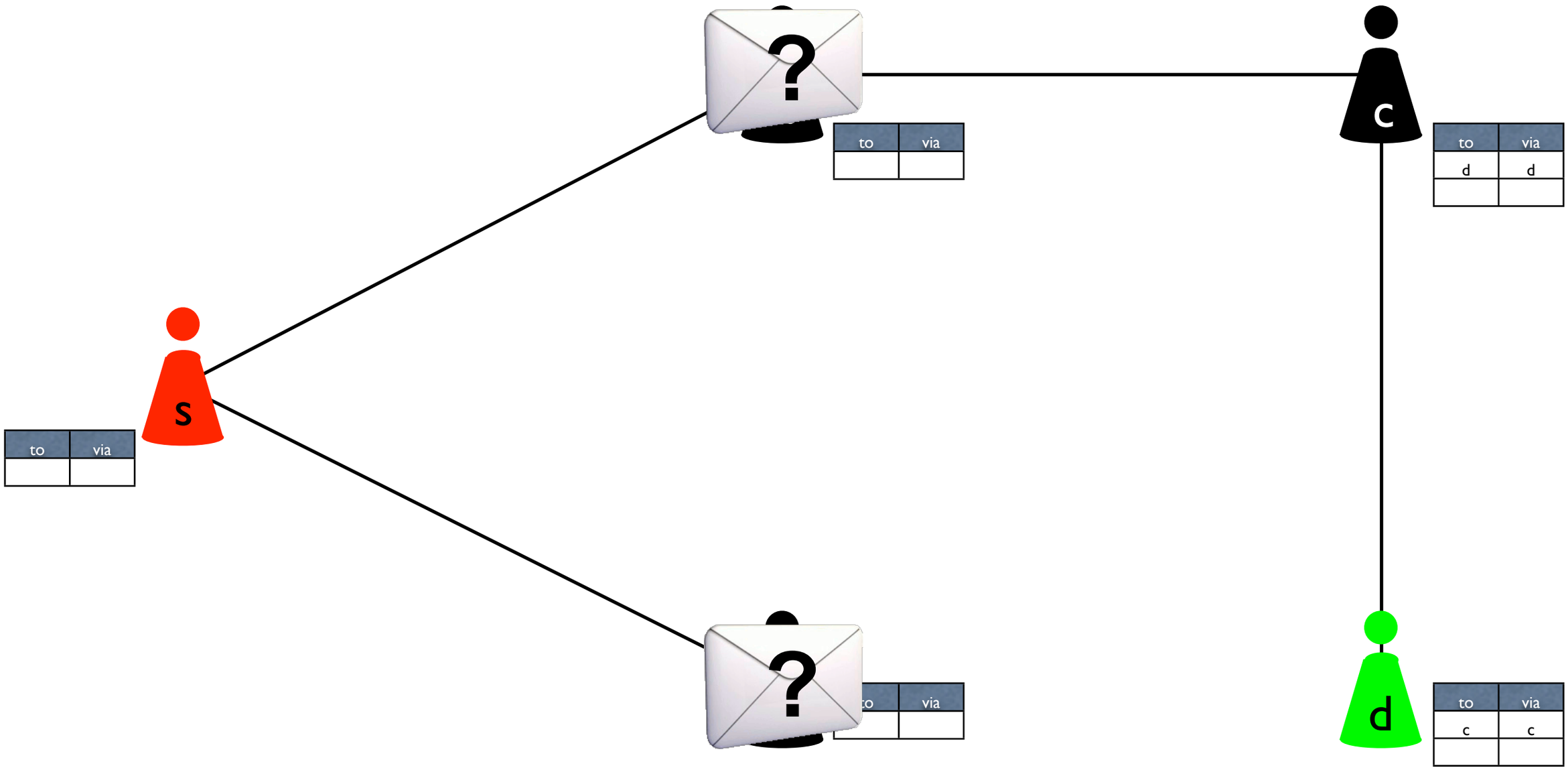
NICTA



AODV – An Example

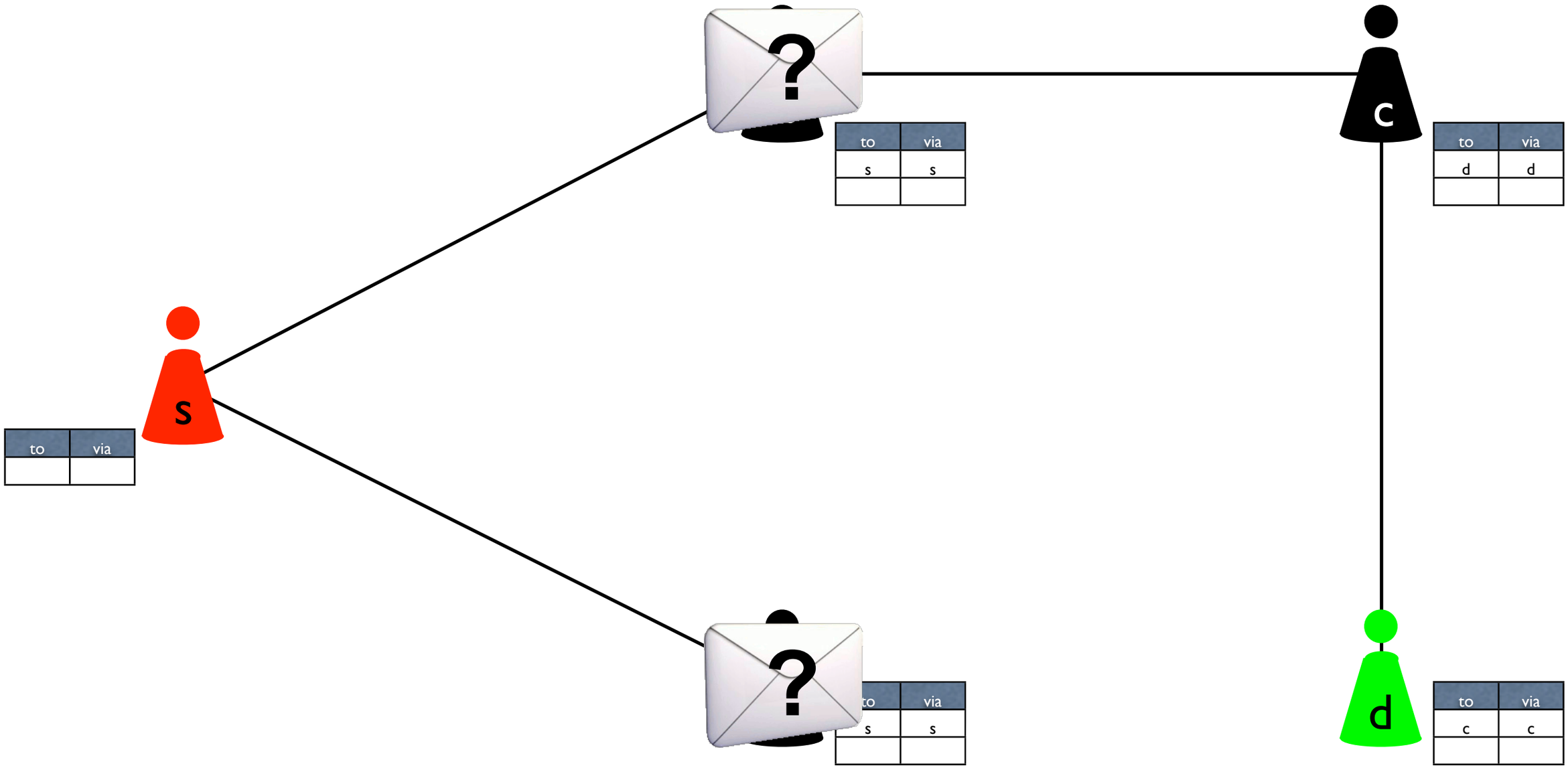


AODV – An Example



s broadcasts a route request

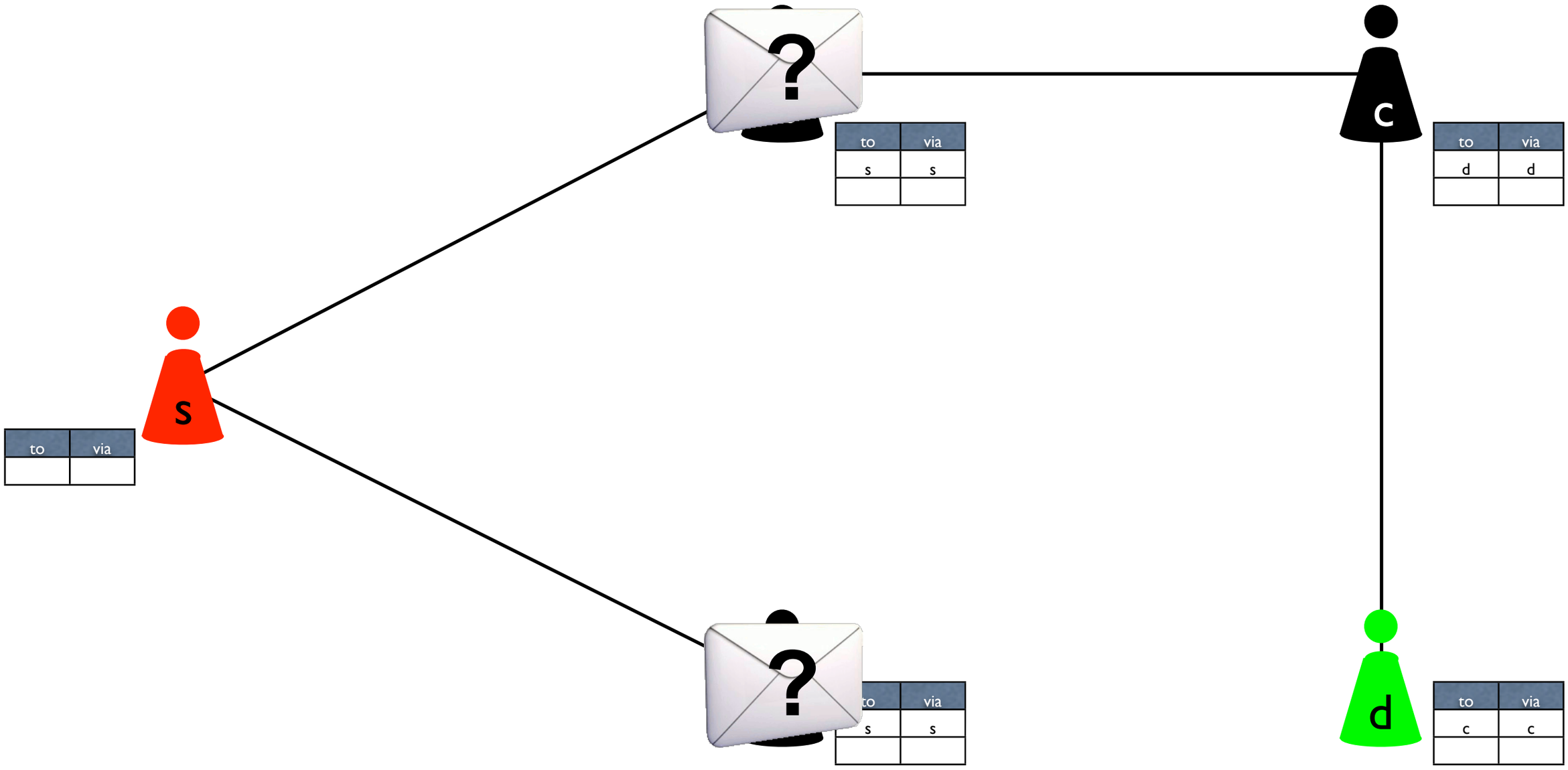
AODV – An Example



s broadcasts a route request

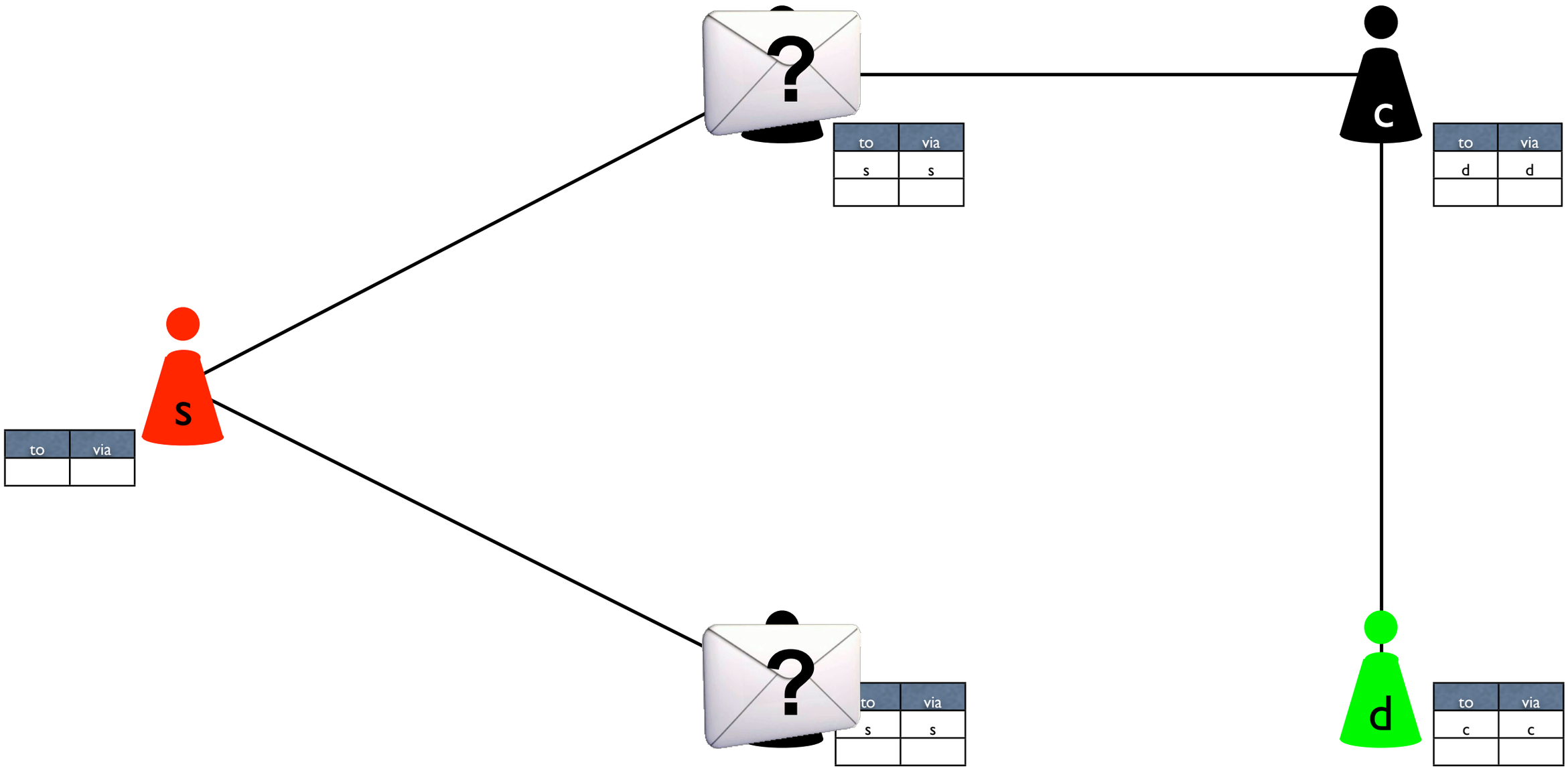
AODV – An Example

NICTA

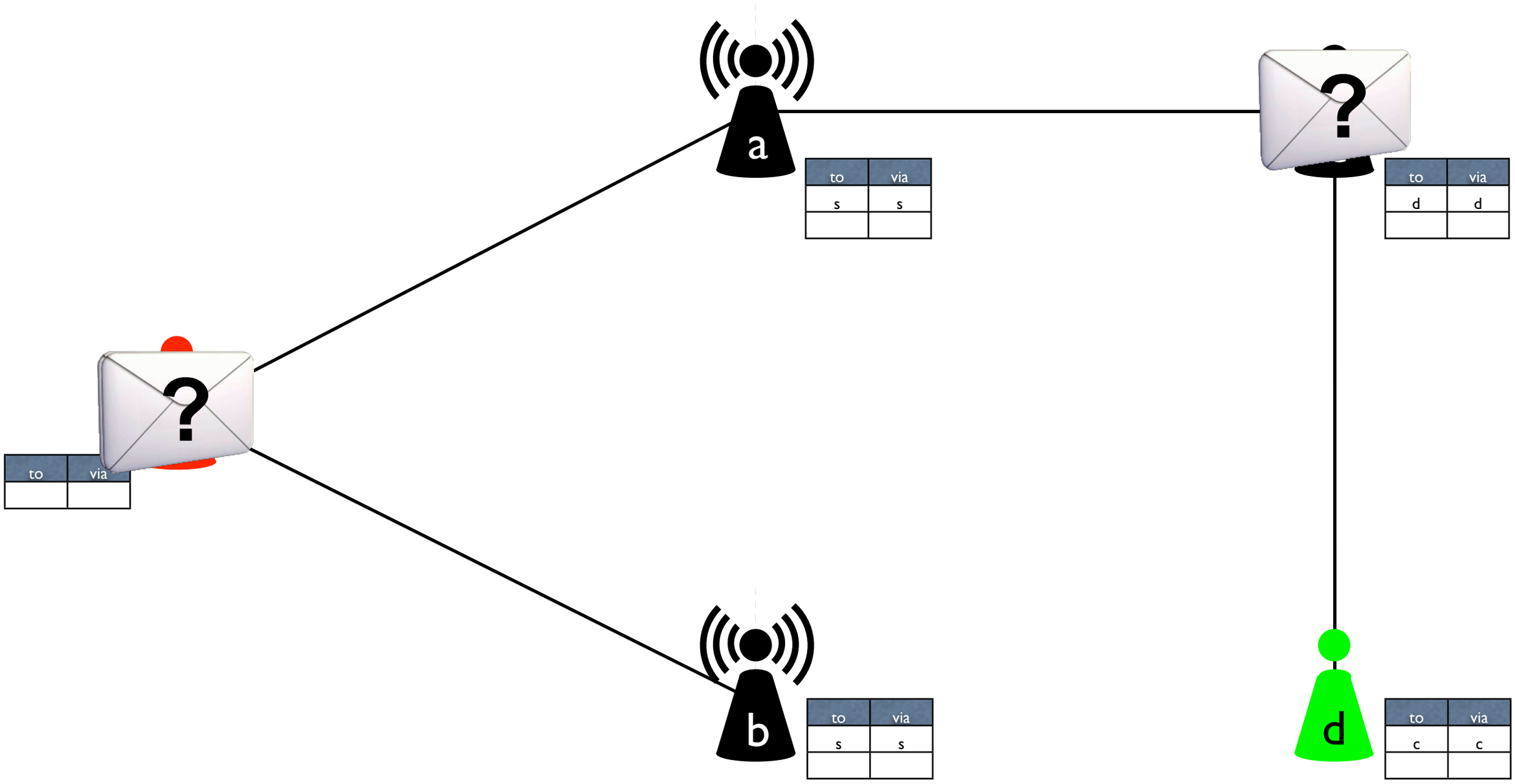


AODV – An Example

NICTA

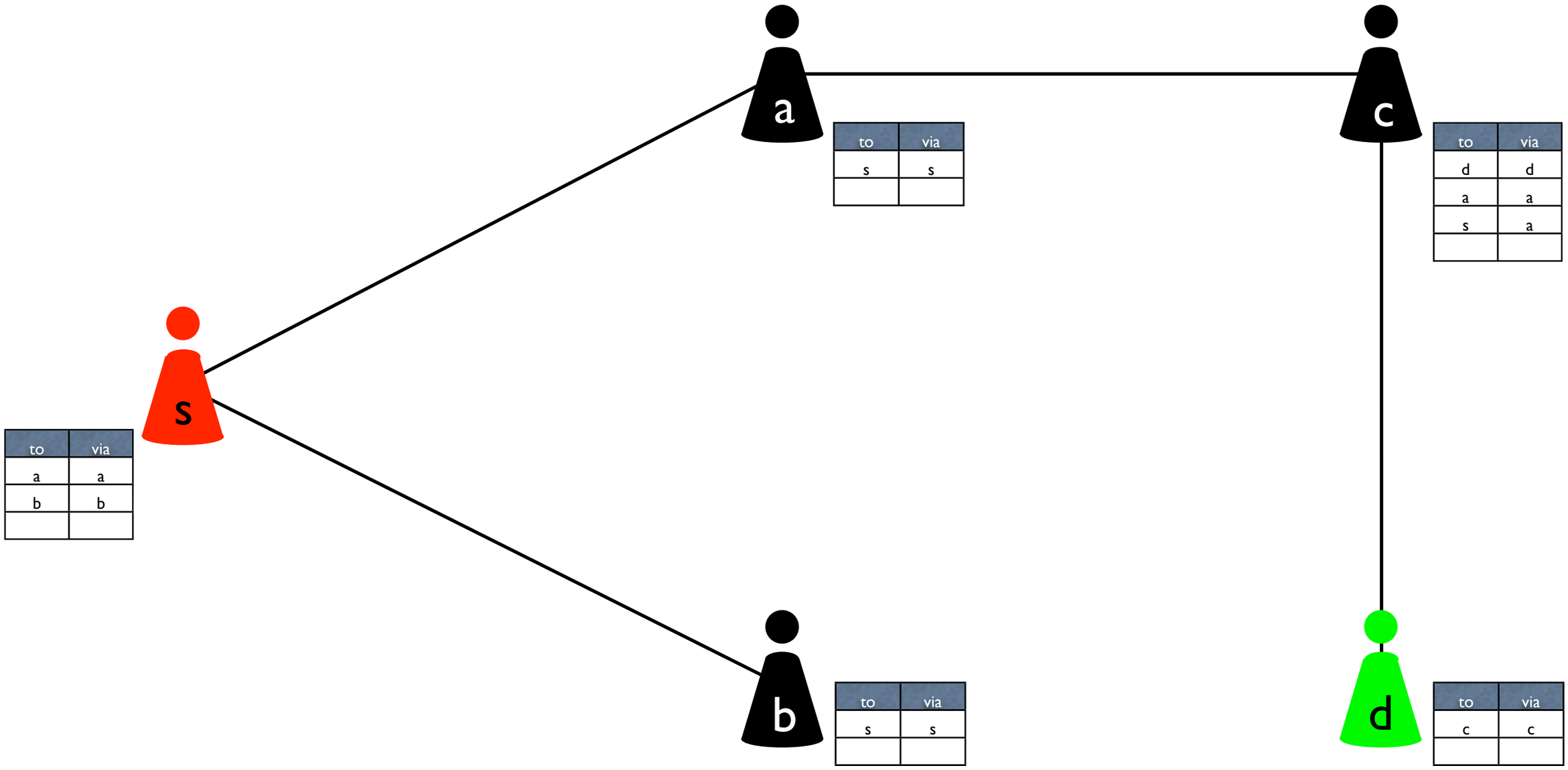


AODV – An Example



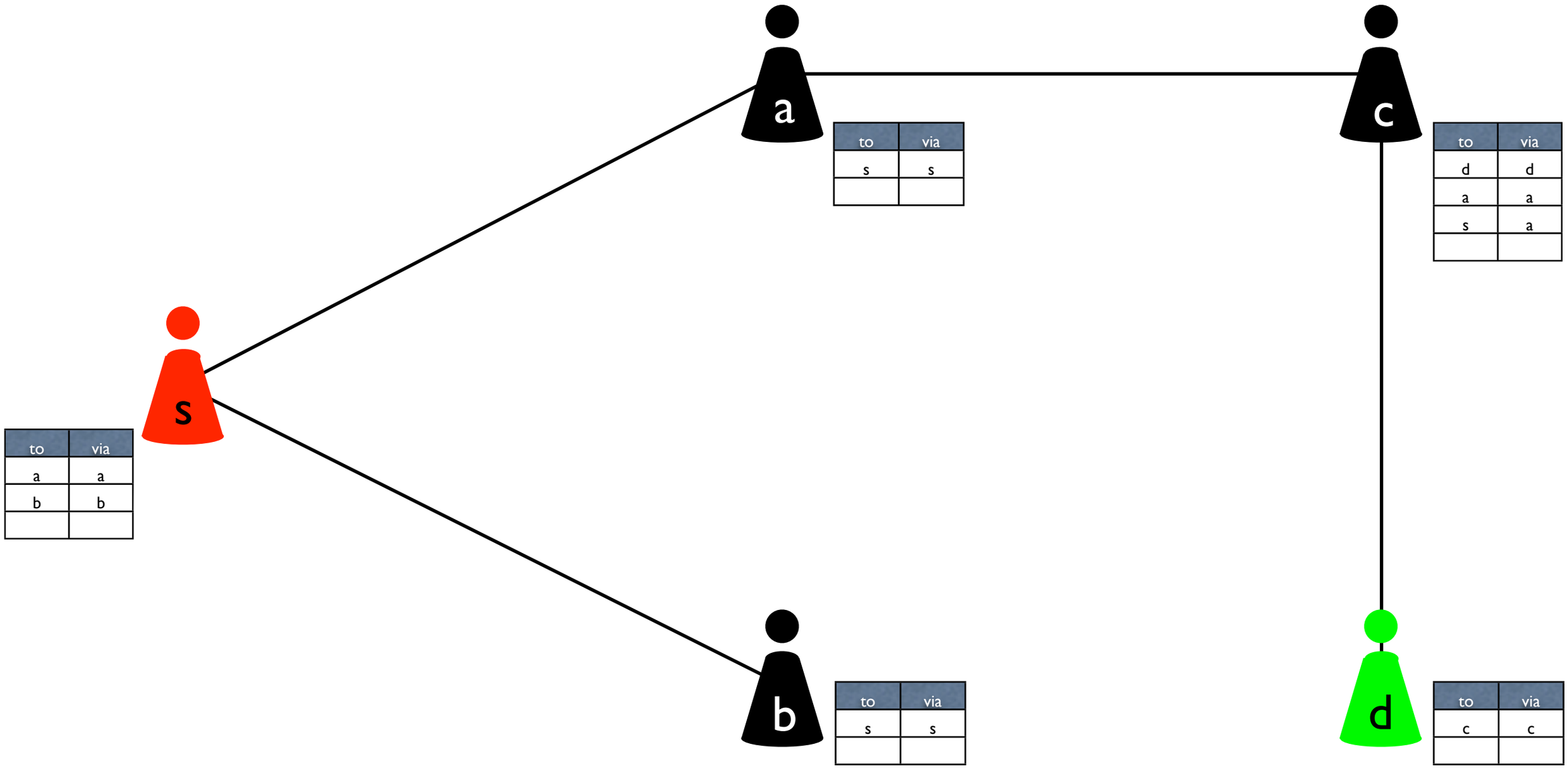
a,b forward the route request

AODV – An Example



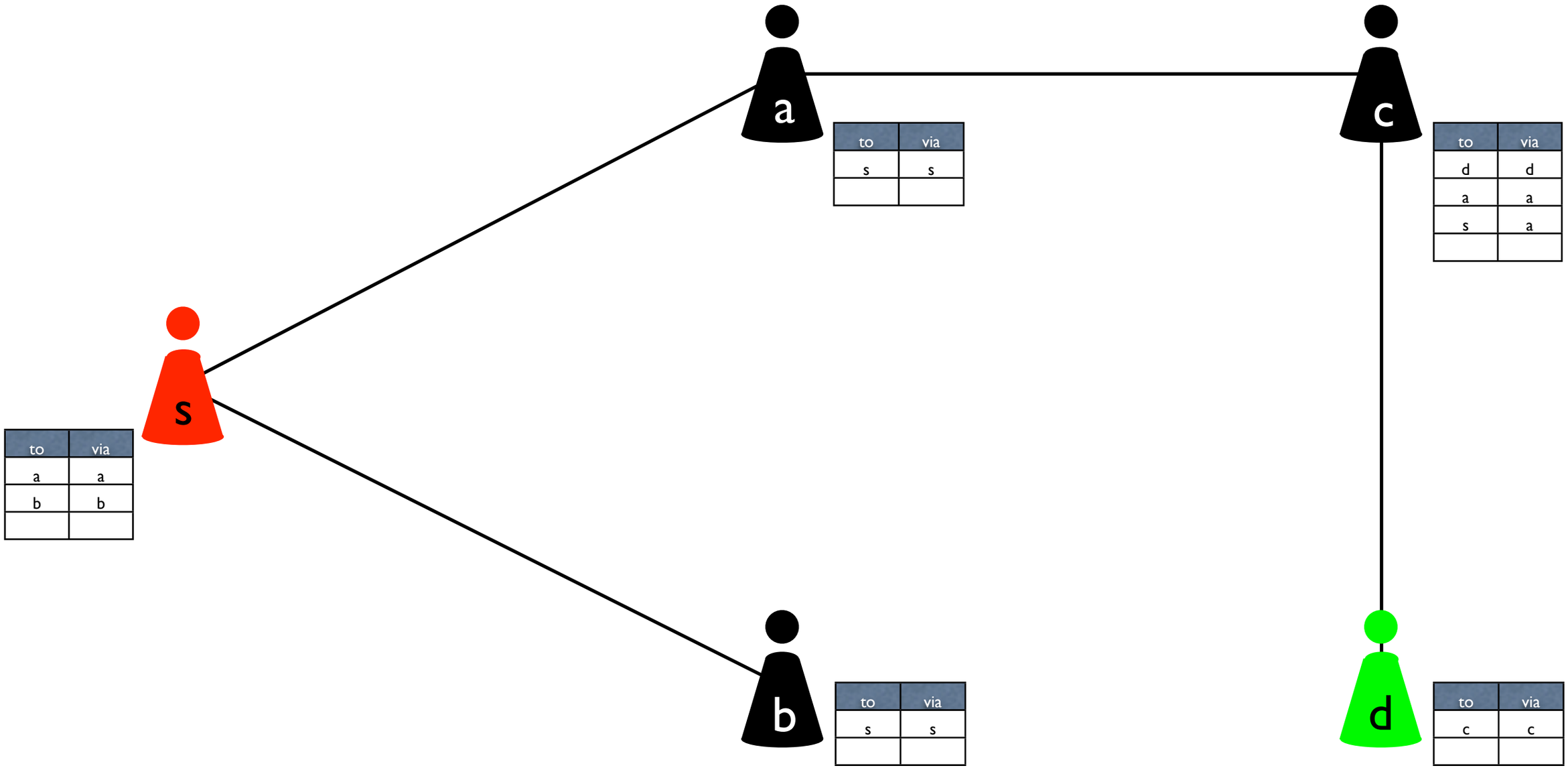
a,b forward the route request

AODV – An Example

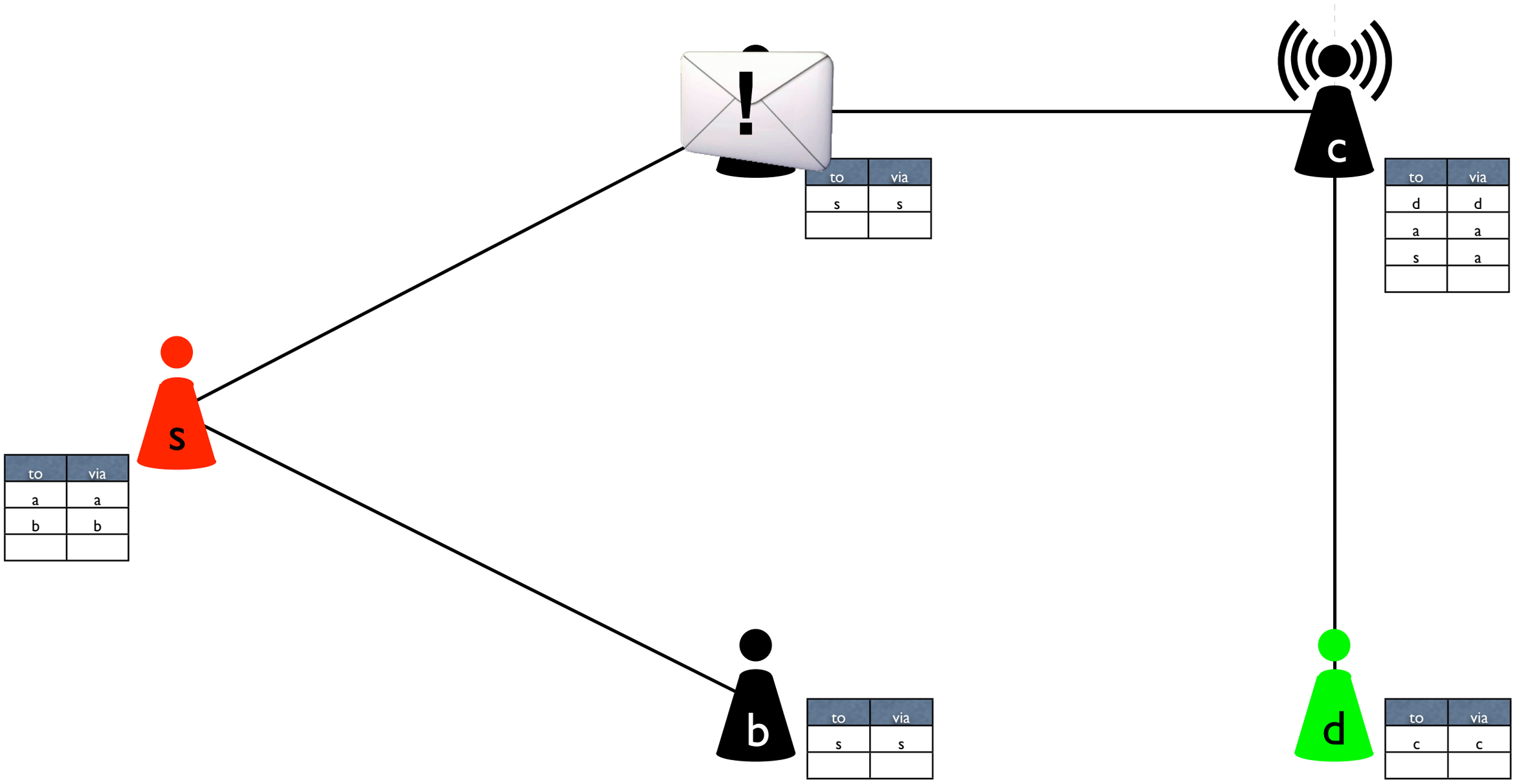


AODV – An Example

NICTA



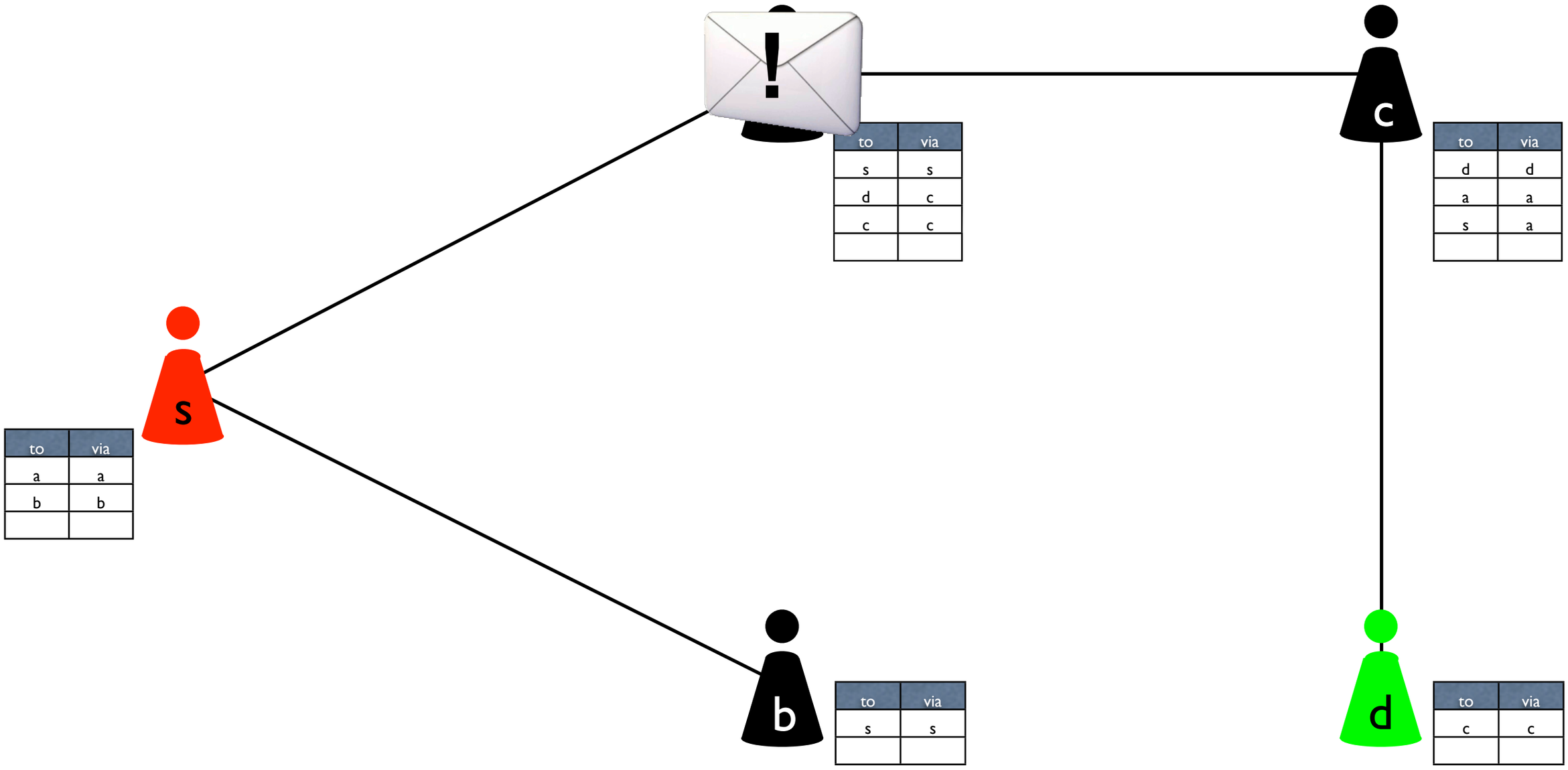
AODV – An Example



c has information about d

c answers route request and sends reply

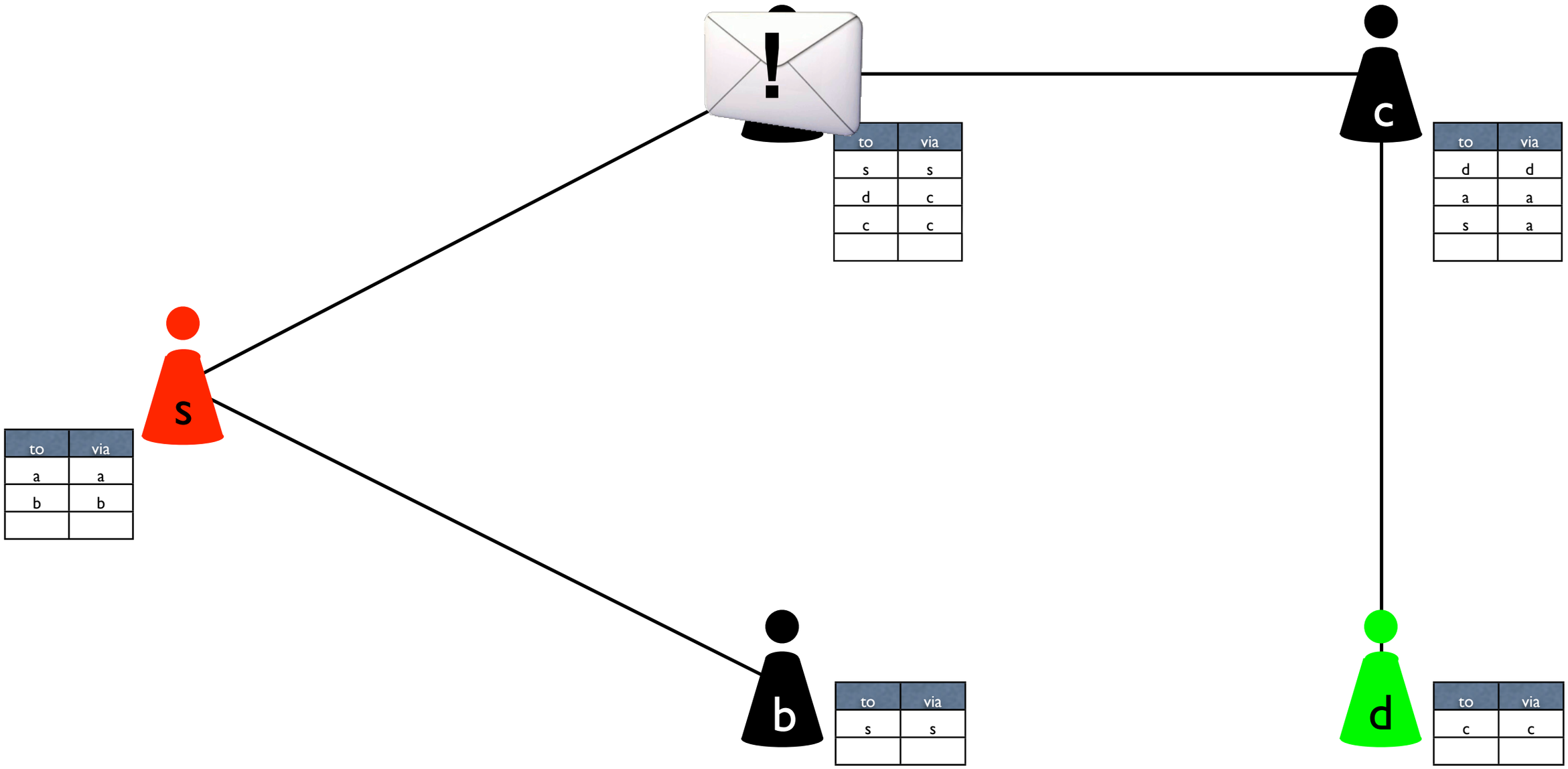
AODV – An Example



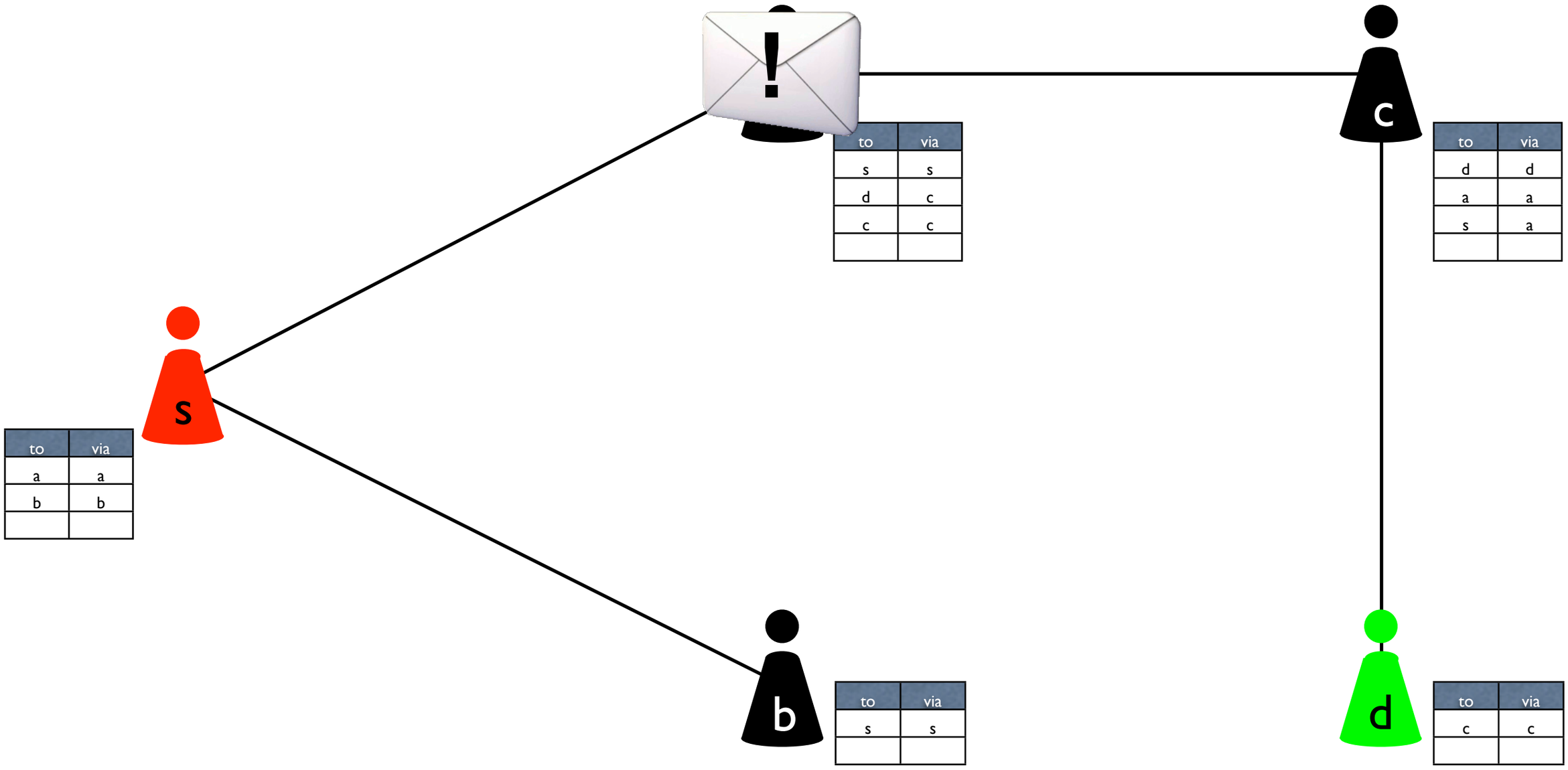
c has information about d

c answers route request and sends reply

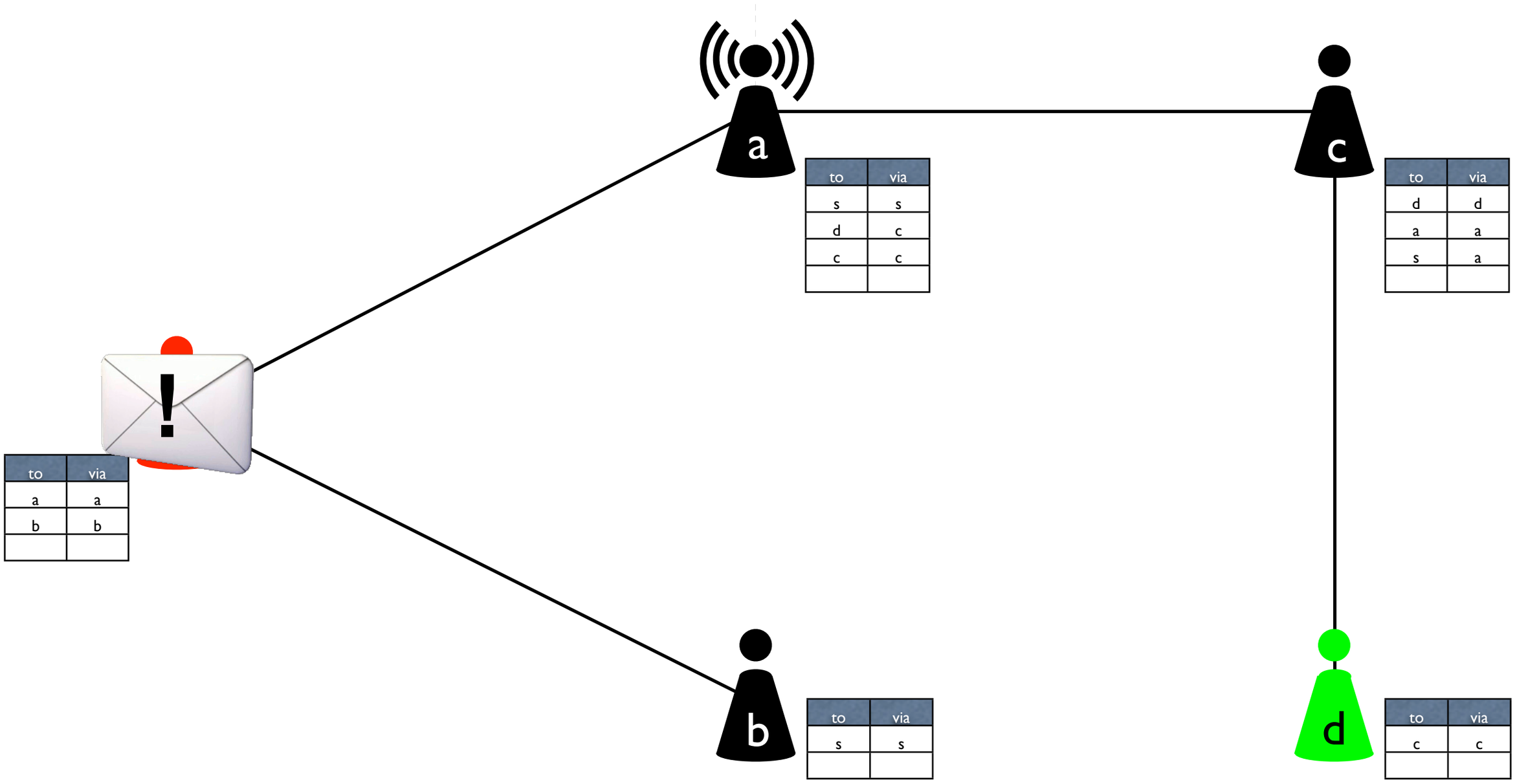
AODV – An Example



AODV – An Example

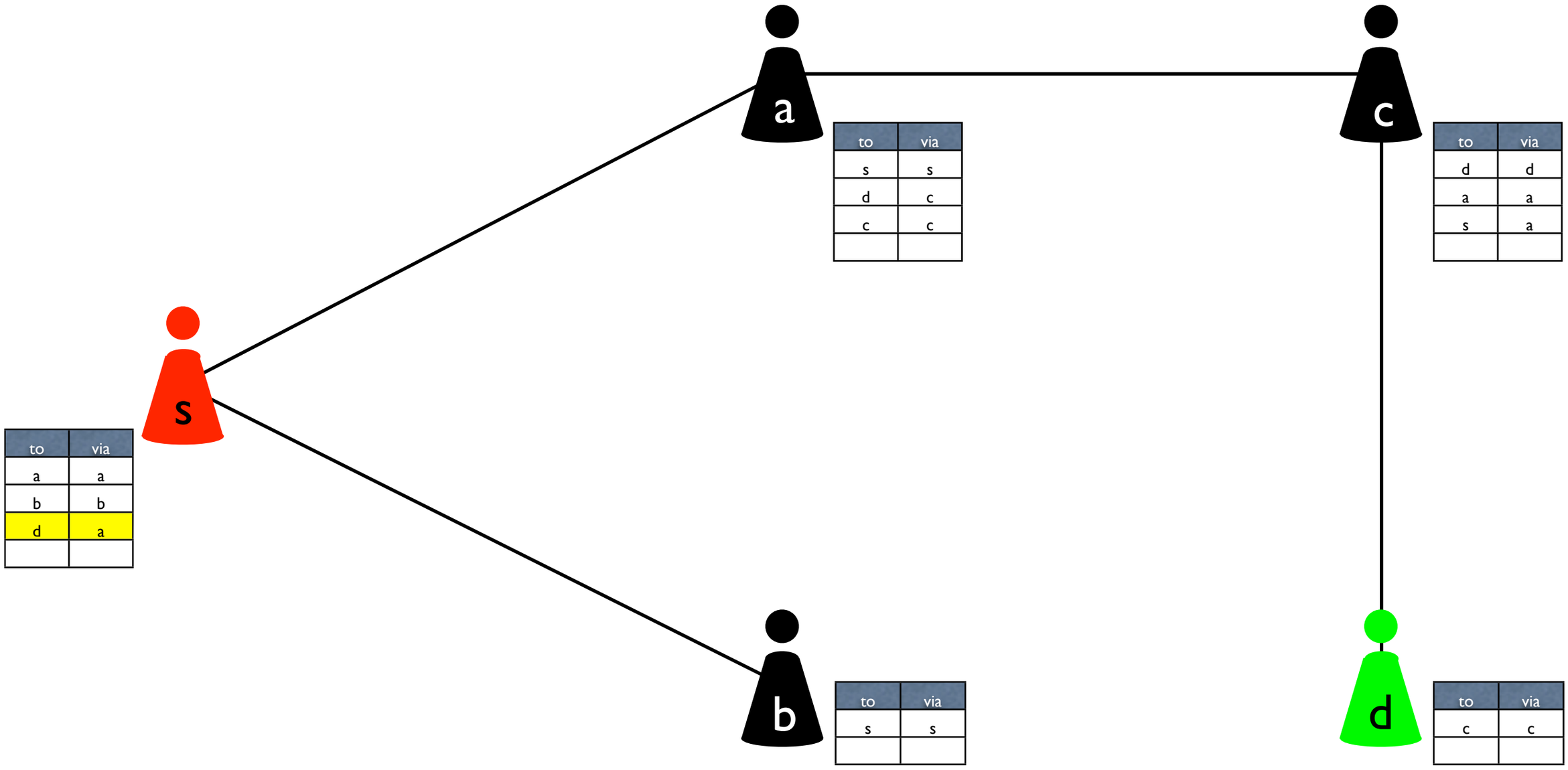


AODV – An Example



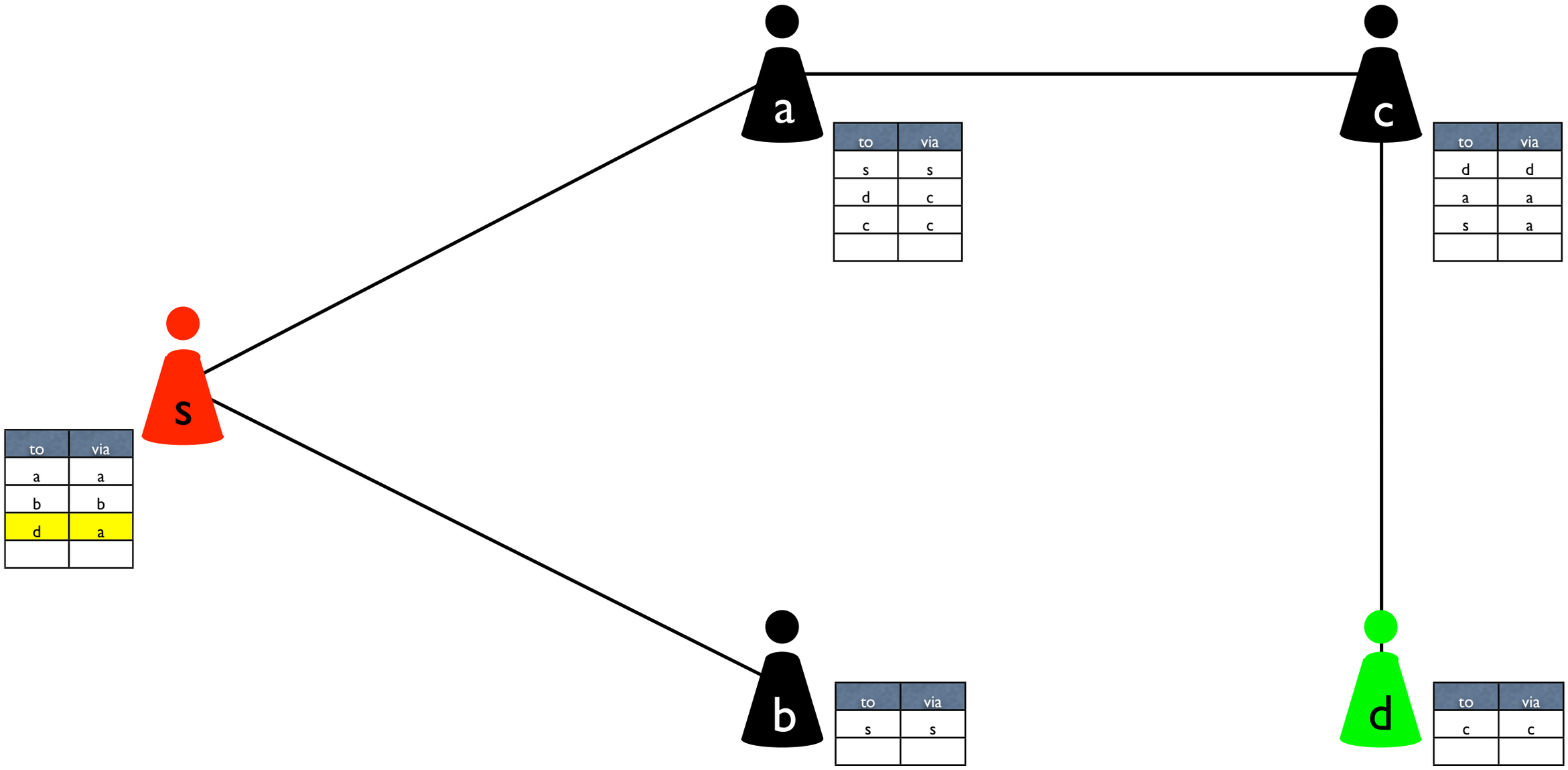
a forwards route reply

AODV – An Example



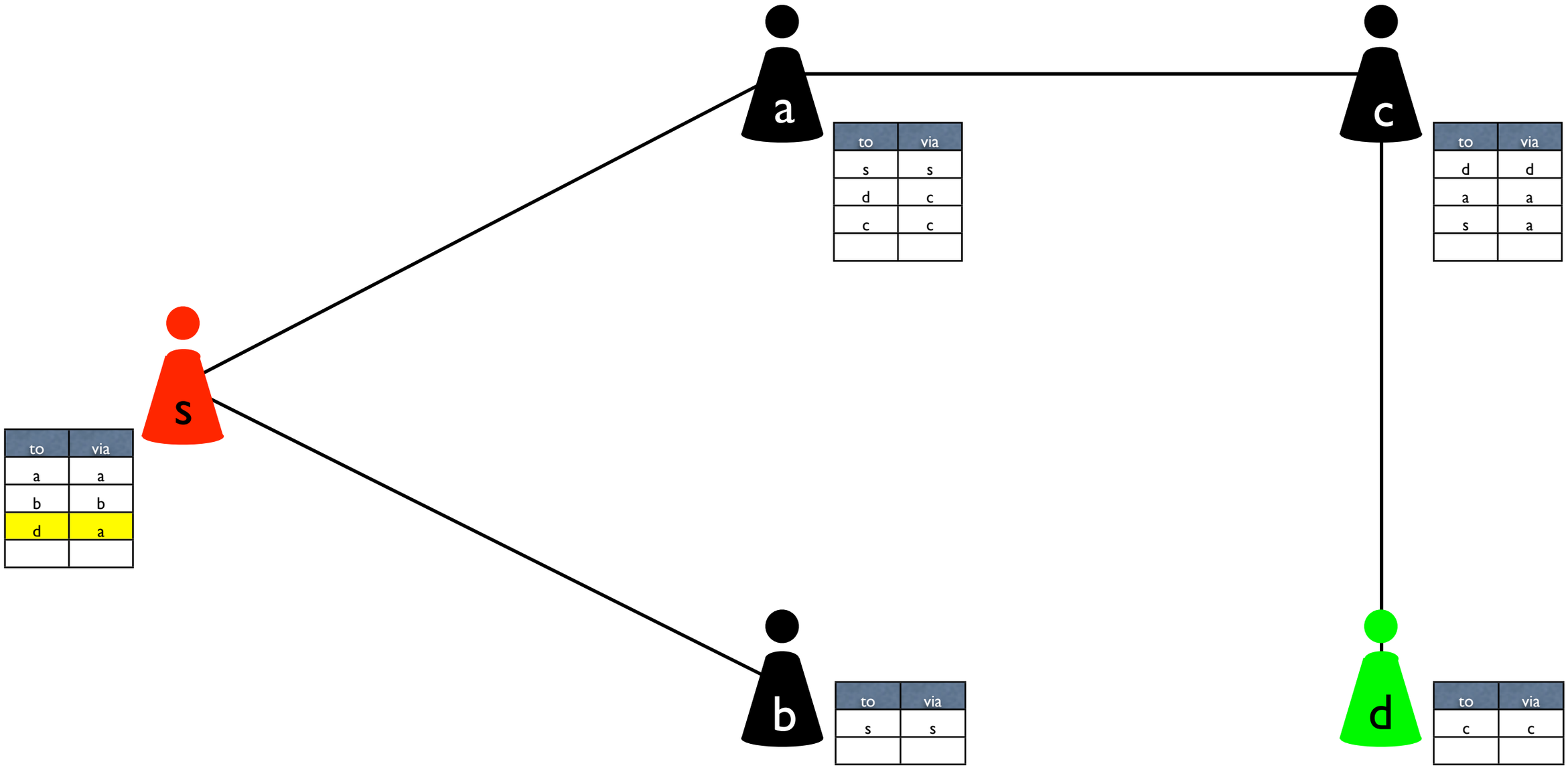
a forwards route reply

AODV – An Example

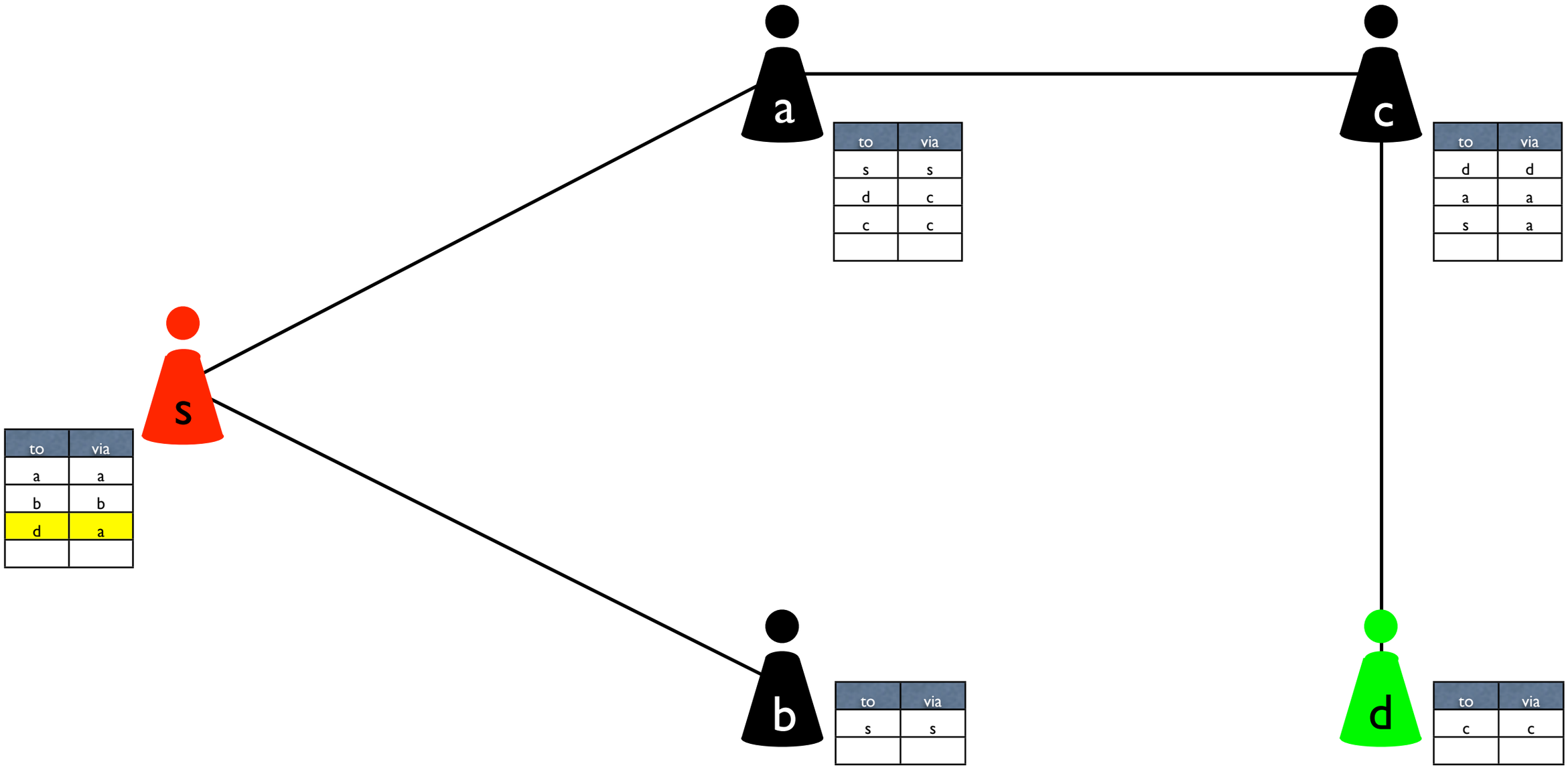


AODV – An Example

NICTA

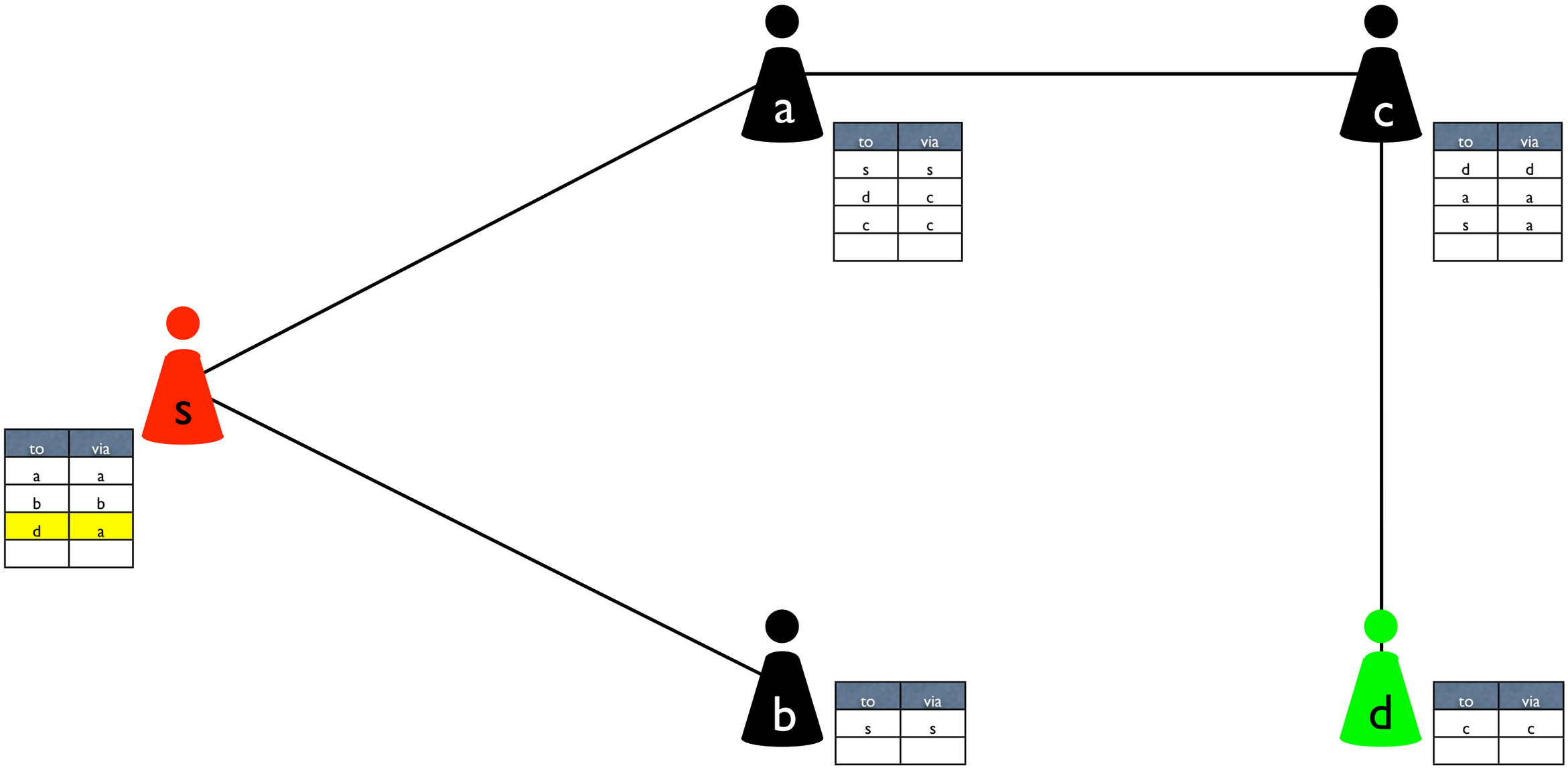


AODV – An Example



s has found a route to d

AODV – An Example



s has found a route to d

Ad Hoc On-Demand Distance Vector Protocol

NICTA

- Properties of AODV
  - route correctness
  - loop freedom
  - route found
  - packet delivery

Ad Hoc On-Demand Distance Vector Protocol

NICTA

- Properties of AODV
  - route correctness 
  - loop freedom 
  - route found 
  - packet delivery 

Ad Hoc On-Demand Distance Vector Protocol

NICTA

- Properties of AODV
  - route correctness 
  - loop freedom 
  - route found 
  - packet delivery 
- so far only simulation and test-bed evaluations
  - important, valid methods
  - limitations
    - resource intensive, time-consuming, no generality

RFC 3561

- Request for Comments (quasi-standard)
- will be standardised this year

```
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.
```

Formal Methods for Mesh Networks

NICTA

- Main Methods used so far
  - process algebra
  - model checking
  - routing algebra

Current Protocols

Network

(textual) Specification

... 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.

This route MAY now be used to send any queued data packets and fulfills any outstanding route requests. ...

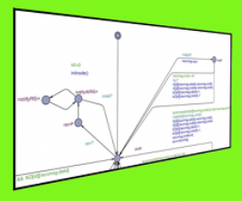
(RFC 3561)

Formal Methods

Process Calculus

/* update the sqn of ip by setting it to max(sqn(rt, ip), dsn) */
⟦rt := update(rt, (ip, dsn, valid, 0, ip, 0))⟧
/* 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)

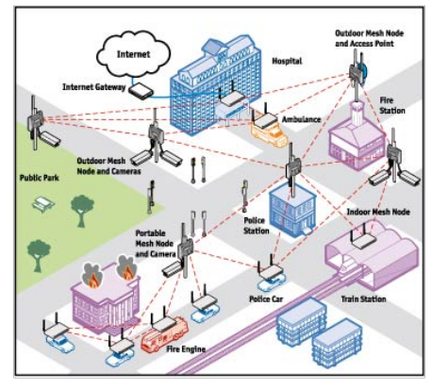
Model Checking



Algebra of Routing Tables

$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{pmatrix} (\varepsilon,0) & (B,1) & (B,2) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ (A,1) & (\varepsilon,0) & (C,1) & (\varepsilon,\infty) & (\varepsilon,\infty) \\ (\varepsilon,\infty) & (B,1) & (\varepsilon,0) & (\varepsilon,\infty) & (E,1) \\ (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,\infty) & (\varepsilon,0) & (E,1) \\ (\varepsilon,\infty) & (\varepsilon,\infty) & (C,1) & (D,1) & (\varepsilon,0) \end{pmatrix} \end{array}$$

Better Protocols



Process Algebra

NICTA

$+$ [$(\mathtt{oip}, \mathtt{rreqid}) \notin \mathtt{rreqs}$] /* the RREQ is new to this node */
/* update the route to oip in rt */
⟦$\mathtt{rt} := \mathtt{update}(\mathtt{rt}, (\mathtt{oip}, \mathtt{osn}, \mathtt{valid}, \mathtt{hops}+1, \mathtt{sip}, \emptyset))$⟧
/* update rreqs by adding (oip, rreqid) */
⟦$\mathtt{rreqs} := \mathtt{rreqs} \cup \{(\mathtt{oip}, \mathtt{rreqid})\}$⟧
(
[$\mathtt{dip} = \mathtt{ip}$] /* this node is the destination node */
/* update the sqn of ip by setting it to max(sqn(rt, ip), dsn) */
⟦$\mathtt{rt} := \mathtt{update}(\mathtt{rt}, (\mathtt{ip}, \mathtt{dsn}, \mathtt{valid}, 0, \mathtt{ip}, \emptyset))$⟧
/* unicast a RREP towards oip of the RREQ; next hop is sip */
unicast$(\mathtt{sip}, \mathtt{rrep}(0, \mathtt{dip}, \mathtt{sqn}(\mathtt{rt}, \mathtt{ip}), \mathtt{oip}, \mathtt{ip}))$. $\mathtt{AODV}(\mathtt{ip}, \mathtt{rt}, \mathtt{rreqs}, \mathtt{queues})$
▶ /* If the packet transmission is unsuccessful, a RERR message is generated */
⟦$\mathtt{dests} := \{(\mathtt{rip}, \mathtt{rsn}) \mid (\mathtt{rip}, \mathtt{rsn}, \mathtt{valid}, *, \mathtt{sip}, *) \in \mathtt{rt}\}$⟧
⟦$\mathtt{pre} := \bigcup\{\mathtt{precs}(\mathtt{rt}, \mathtt{rip}) \mid (\mathtt{rip}, *) \in \mathtt{dests}\}$⟧
⟦for all $(\mathtt{rip}, *) \in \mathtt{dests}$: $\mathtt{invalidate}(\mathtt{rt}, \mathtt{rip})$⟧
groupcast$(\mathtt{pre}, \mathtt{rerr}(\mathtt{dests}, \mathtt{ip}))$. $\mathtt{AODV}(\mathtt{ip}, \mathtt{rt}, \mathtt{rreqs}, \mathtt{queues})$
$+$ [$\mathtt{dip} \neq \mathtt{ip}$] /* this node is not the destination node */
(
[$\mathtt{dip} \in \mathtt{aD}(\mathtt{rt}) \wedge \mathtt{dsn} \leq \mathtt{sqn}(\mathtt{rt}, \mathtt{dip}) \wedge \mathtt{sqn}(\mathtt{rt}, \mathtt{dip}) \neq 0$] /* valid route to dip that is fresh enough */
/* update rt by adding sip to precs(rt, dip) */
⟦$\mathtt{r} := \mathtt{addpre}(\sigma_{route}(\mathtt{rt}, \mathtt{dip}), \{\mathtt{sip}\})$; $\mathtt{rt} := \mathtt{update}(\mathtt{rt}, \mathtt{r})$⟧

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

Model Checking

- Model checking routing algorithms
  - executable models
- Complementary to process algebra
  - find bugs and typos in model of process algebra
  - 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

DEMO

Routing Algebra

$$
\begin{array}{c}
 \\
A \\
B \\
C \\
D \\
E
\end{array}
\begin{array}{c}
\begin{array}{ccccc} A & B & C & D & E \end{array} \\
\begin{pmatrix}
(\epsilon,0) & (B,1) & (B,2) & (\epsilon,\infty) & (\epsilon,\infty) \\
(A,1) & (\epsilon,0) & (C,1) & (\epsilon,\infty) & (\epsilon,\infty) \\
(\epsilon,\infty) & (B,1) & (\epsilon,0) & (\epsilon,\infty) & (E,1) \\
(\epsilon,\infty) & (\epsilon,\infty) & (\epsilon,\infty) & (\epsilon,0) & (E,1) \\
(\epsilon,\infty) & (\epsilon,\infty) & (C,1) & (D,1) & (\epsilon,0)
\end{pmatrix}
\end{array}
$$

Routing Algebra – Elements, Operators

- Routing table entries (no sequence number so far)

$$(\texttt{nhip}, \texttt{hops})$$

- Choice: $(A, 5) + (B, 2) = (B, 2)$
- Multiplication: $(A, 5) \cdot (B, 2) = (A, 7)$
  - destination and source must coincide

- idea: back to Backhouse, Carré, Griffin, Sobrinho

Routing Algebra – Elements, Operators

- Matrices over routing table entries

$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ \vdots \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & \dots \end{array} \\ \left(\begin{array}{cccccc} (\_,0) & (B,1) & (B,2) & (\_,\infty) & \\ (A,1) & (\_,0) & (C,1) & (\_,\infty) & \dots \\ (\_,\infty) & (B,1) & (\_,0) & (\_,\infty) & \\ (\_,\infty) & (\_,\infty) & (\_,\infty) & (\_,0) & \\ & \vdots & & & \ddots \end{array} \right) \end{array}$$

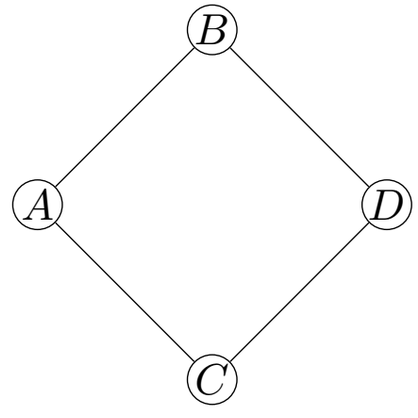
routing table of A

"routes" to B

- standard matrix operations
- further abstraction possible (semirings, test, domain, modules ...)

Example

- A route request is broadcast



$$\begin{pmatrix} (\_,0) & (B,1) & (C,1) & (\_,\infty) \\ (A,1) & (\_,0) & (\_,\infty) & (D,1) \\ (A,1) & (\_,\infty) & (\_,0) & (D,1) \\ (\_,\infty) & (B,1) & (C,1) & (\_,0) \end{pmatrix} \bullet \begin{pmatrix} (\_,0) & (\_,\infty) & (\_,\infty) & (\_,\infty) \\ (\_,\infty) & (\_,\infty) & (\_,\infty) & (\_,\infty) \\ (\_,\infty) & (\_,\infty) & (\_,\infty) & (\_,\infty) \\ (\_,\infty) & (\_,\infty) & (\_,\infty) & (\_,\infty) \end{pmatrix} + \begin{pmatrix} (\_,0) & (B,1) & (\_,\infty) & (\_,\infty) \\ (\mathbf{D},\mathbf{3}) & (\_,0) & (\_,\infty) & (\_,\infty) \\ (A,1) & (\_,\infty) & (\_,0) & (D,1) \\ (C,2) & (\_,\infty) & (C,1) & (\_,0) \end{pmatrix}$$

topology sender routing table

$$= \begin{pmatrix} (\_,0) & (B,1) & (\_,\infty) & (\_,\infty) \\ (\mathbf{A},\mathbf{1}) & (\_,0) & (\_,\infty) & (\_,\infty) \\ (A,1) & (\_,\infty) & (\_,0) & (D,1) \\ (C,2) & (\_,\infty) & (C,1) & (\_,0) \end{pmatrix}$$

updated routing table

Sent Messages

- Achievements
  - sending messages

$$a + p \cdot b \cdot q \cdot (1 + c)$$

  - broadcast, unicast, groupcast are the same (modelled by different topologies)
  - Kleene star models flooding the network (modal operators terminate flooding)
  - great potential for automation (Prover9, Isabelle, ...)

Conclusion/Future Work

- So far concentrated on AODV
  - well known
  - IETF standard
- Extend formal methods to other protocols
  - OSLR, DYMO, ...
- Add further necessary concepts
  - time
  - probability

Mastertheses

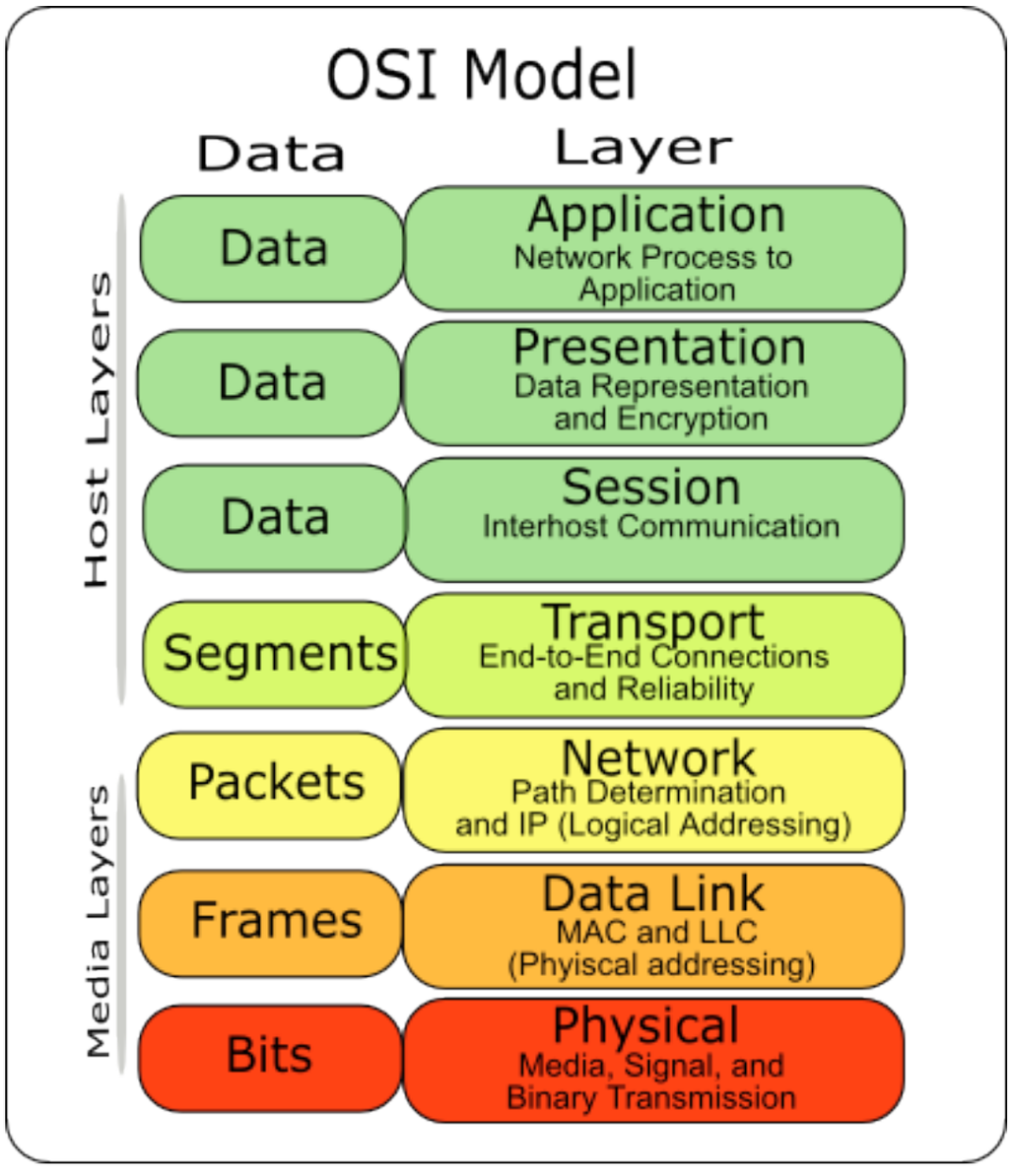
NICTA

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

NICTA

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

Different Network Layers



Routing Protocols and Routing Tables

NICTA

- Routing protocols
  - find (optimal) route
  - properties
    - loop freedom (no packet travels in loops)
    - route correctness (if a route is found, the route is valid)
    - route found (if a route exists, at least one route is found)
    - packet delivery

- Routing tables
  - data structure
  - belongs to client/router
  - lists destinations
  - sometimes metrics