# Modelling and Analysis of AODV in UPPAAL

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

NICTA

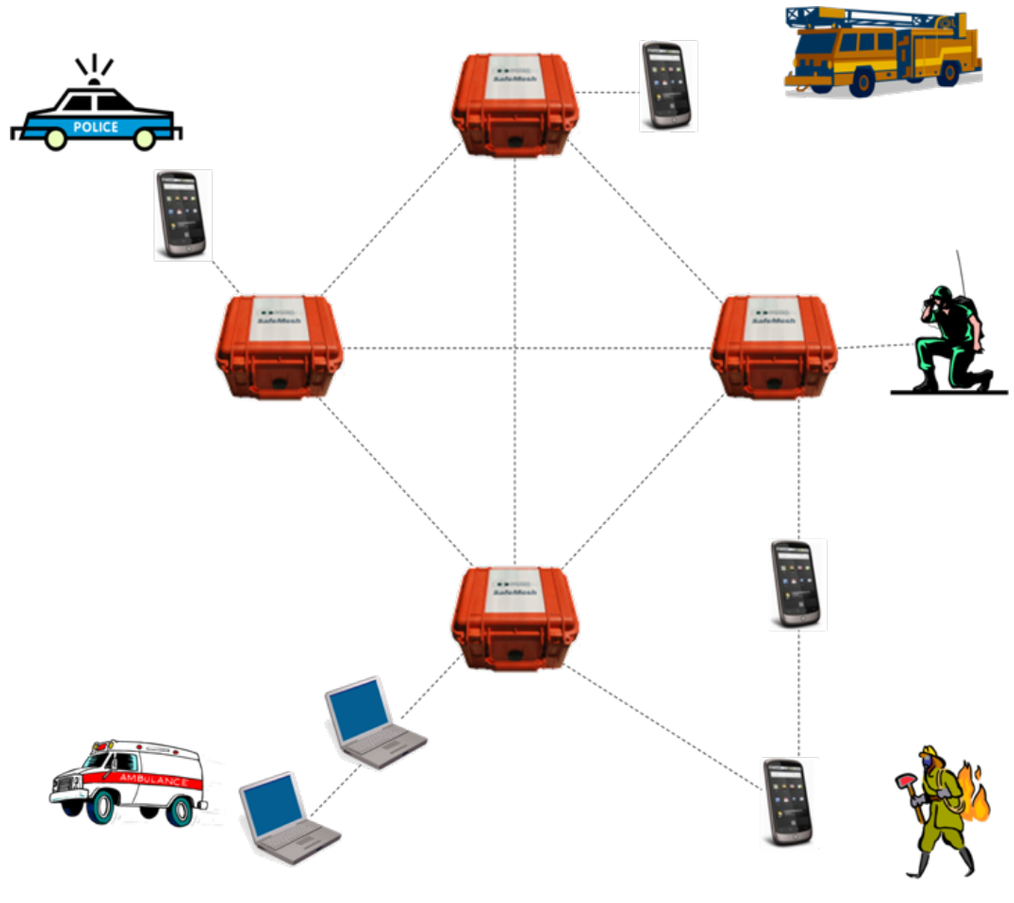
Australian Government
Department of Broadband, Communications and the Digital Economy
Australian Research Council

NICTA Members
ANU
UNSW
Department of State and Regional Development
Victoria The Place To Be
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
  - ...




© NICTA 2011

# Ultimate Goal

- Formal Methods for Mesh Networks
  - process algebra
  - model checking
  - routing algebra / meta routing
  - ...

## 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. ...

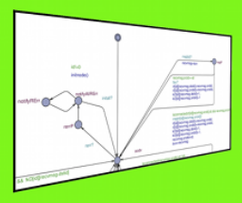
(RFC 3561)

## 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)

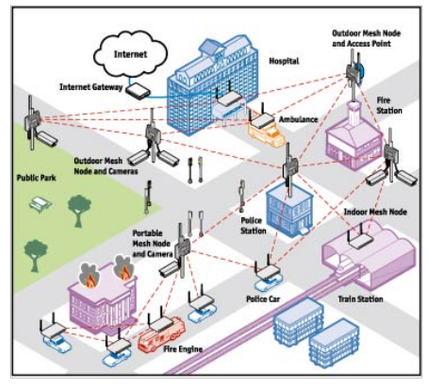
### Model Checking



### 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




© NICTA 2011

# 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)
- RFC by the IETF MANET working group (1 of 4)
- basis of upcoming IEEE 802.11s

© NICTA 2011

# 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: $(\texttt{dip}, \texttt{dsn}, \texttt{val}, \texttt{hops}, \texttt{nhip}, \texttt{pre})$

© NICTA 2011

# UPPAAL Model Checker

NICTA

- Well established model checker
- Uses networks of timed automata
- Has been used for protocol verification
- Synchronisation mechanisms
  - binary handshake synchronisation (unicast communication)
  - broadcast synchronisation (broadcast communication)
- Common data structures
  - arrays, structs, ...
  - C-like programming language
- Provides mechanisms for time and probability

© NICTA 2011

# Modelling AODV in UPPAAL

- derived from process-algebraic model
- allows interplay
- increases trust

- process algebra AWN
  - developed specifically for WMN routing protocols
  - easily readable
  - three necessary features:
    data structures, local broadcast, conditional unicast

© NICTA 2011

# Process Algebra AWN

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) */
⟦$\mathtt{r} := \mathtt{addpre}(\sigma_{route}(\mathtt{rt}, \mathtt{dip}), \{\mathtt{sip}\})$; $\mathtt{rt} := \mathtt{update}(\mathtt{rt}, \mathtt{r})$⟧

© NICTA 2011

# Analysis of AODV

- evaluation of WMN routing protocols

- confirm problematic and undesirable behaviours
- find new problems
- exhaustive search
- easily adapted to variants

- no test-bed or simulation-based experiments
  - important and valid methods for protocol evaluation
  - but resource intensive and time-consuming
- complements proofs in AWN
  - based on same spec is important

© NICTA 2011

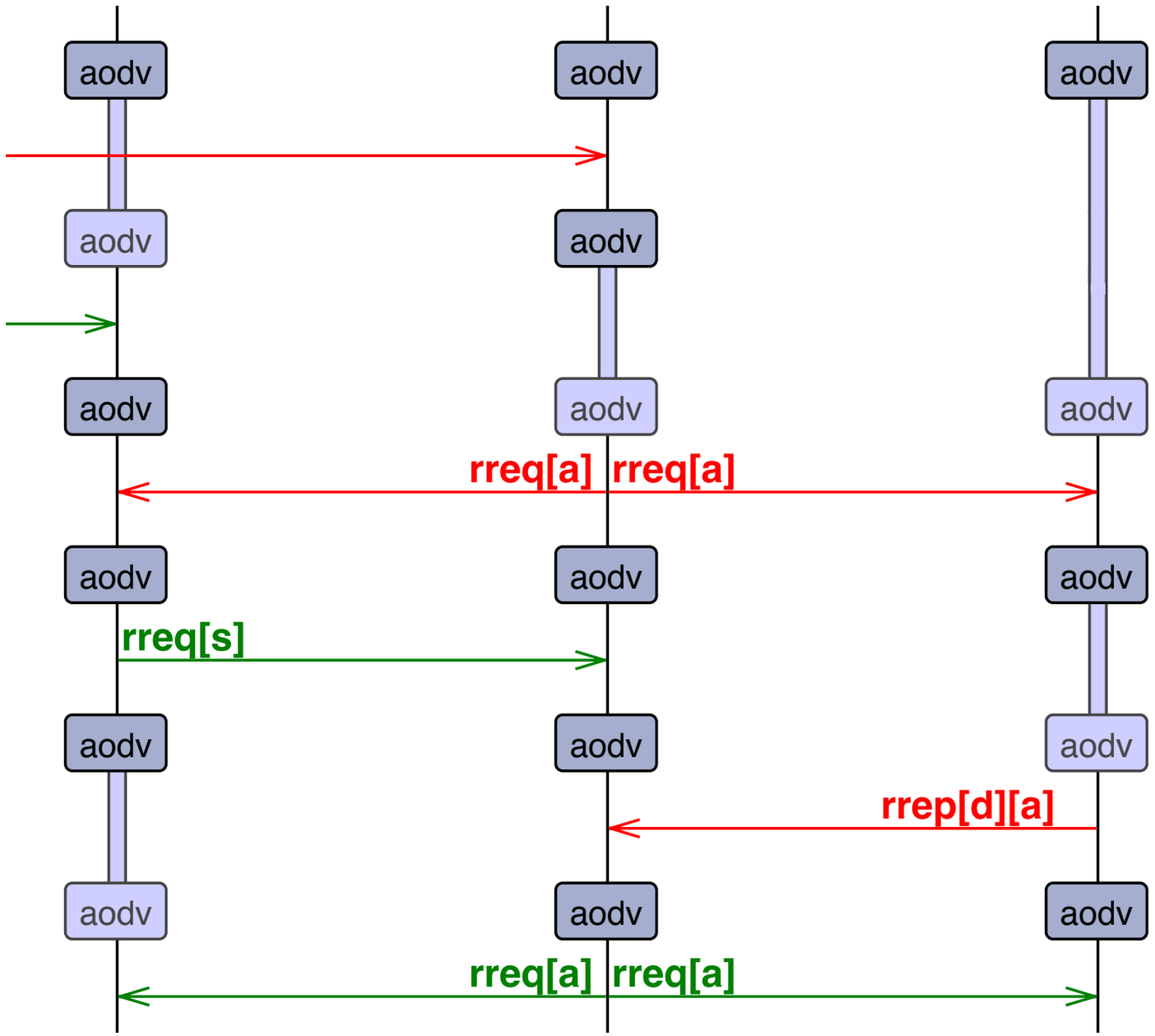
# Experiments Set-Up

- in the paper: only confirmation and replay (test of UPPAAL)
- meanwhile: exhaustive search
  - different properties
  - all topologies up to 5 nodes (one topology change)
  - 2 route discovery processes
  - 17400 scenarios
  - variants of AODV (4 models)

© NICTA 2011

# Results: Route Discovery (2004)

- Route discovery fails in a linear 3-node topology




© NICTA 2011

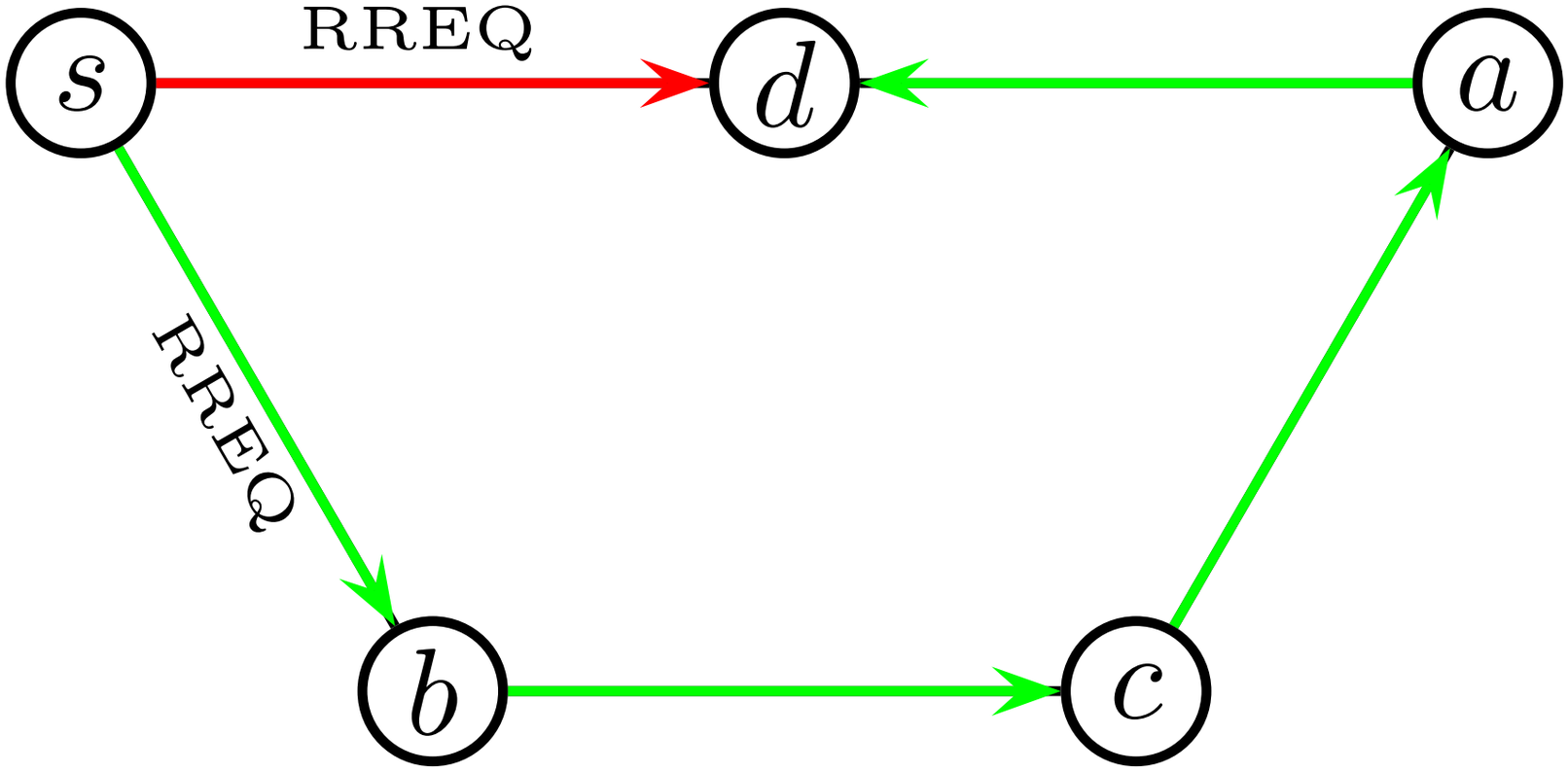
# Results: Route Discovery

- exhaustive search
  (potential failure in route discovery)
  - static topology: 47.3%
  - dynamic topology (add link): 42.5%
  - dynamic topology (remove link): 73.7%
- AODV repeats route request
- Other solution: forward route reply
- Future work: define quality of protocols

© NICTA 2011

# Results: Route Optimality (2010)

- Route optimality fails due to different reasons




- Is shortest path what we want?

© NICTA 2011

# Results: Loop Freedom

- Proven before
- But: really depends on the interpretation of the RFC
  - ambiguities
  - unspecified cases
  - ...
- Experiments with process algebra and modell checking helps in analysing modifications/variants
  - small changes
  - easy set up

© NICTA 2011

# Conclusion

- Formal Methods can help in analysis and understanding of real protocols
- Combination of process algebra and modell checking

- Process algebra
  - easy to read
  - specifically design for WMNs routing protocols
  - proof environment (loop freedom, route discovery, ...)
- Model checking
  - complements process algebra
  - exhaustive search
  - (automatic) correct transformation from process-algebraic model
  - so far only UPPAAL

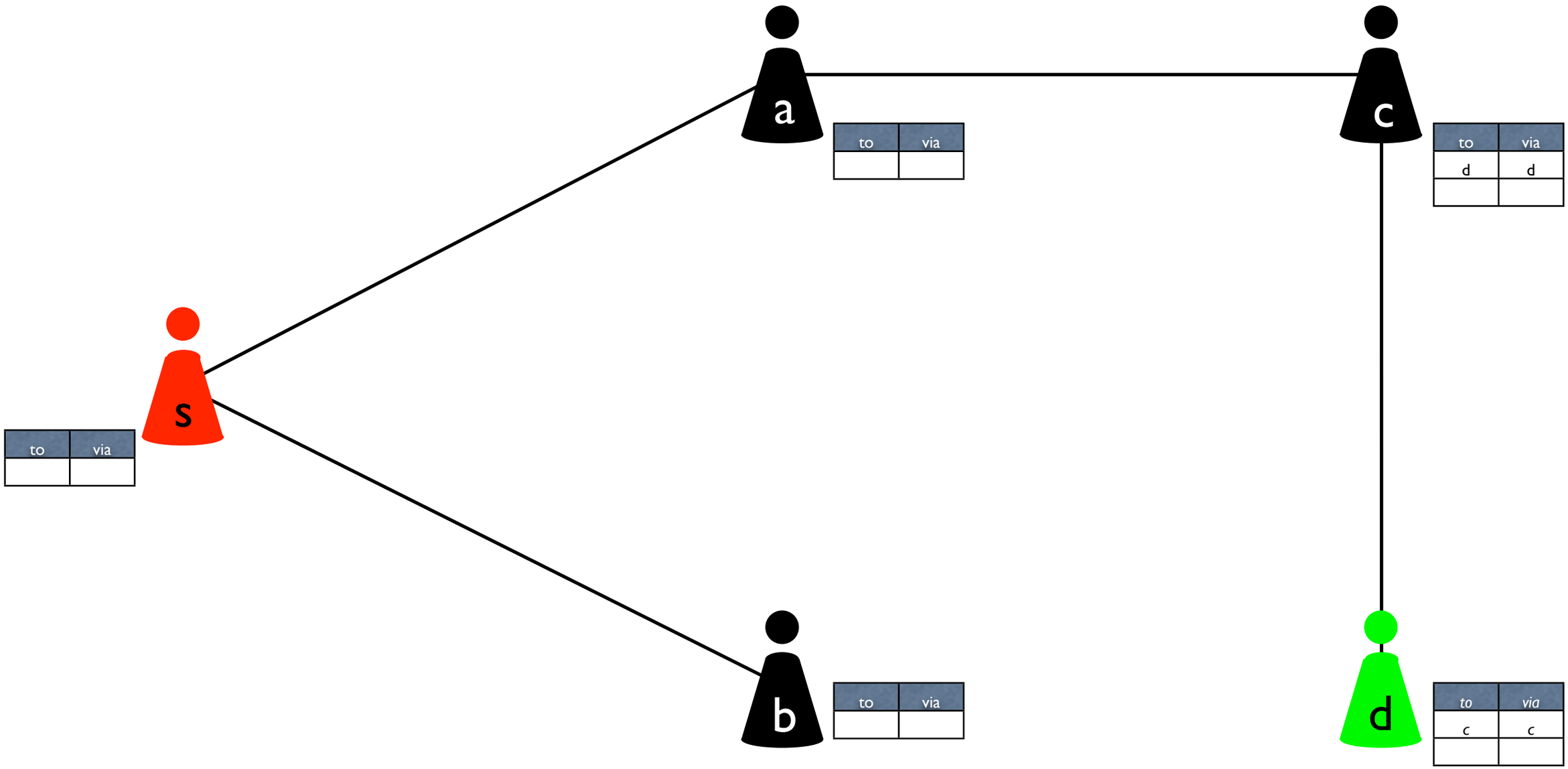
© NICTA 2011

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

# NICTA

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

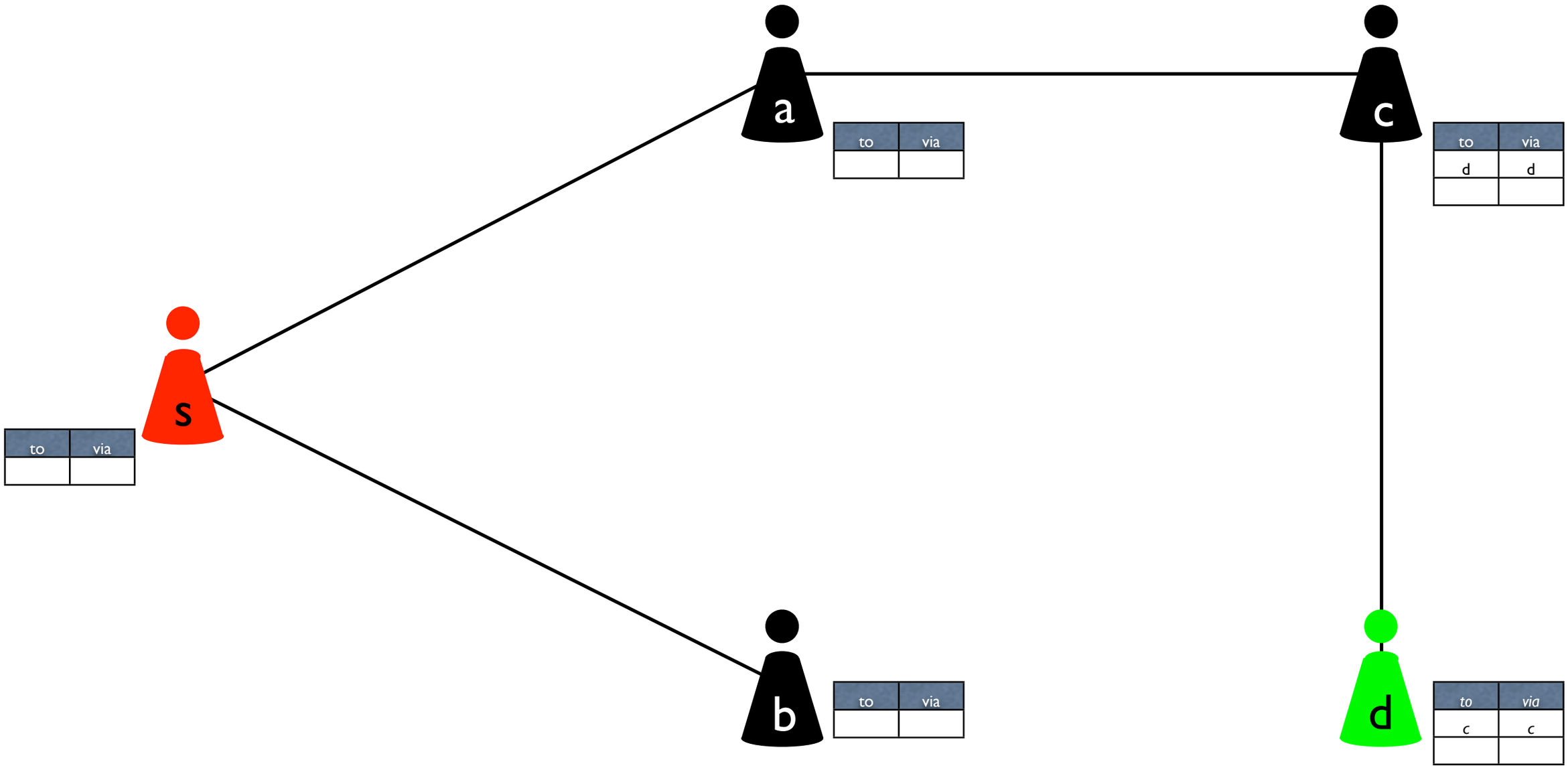
# AODV – An Example



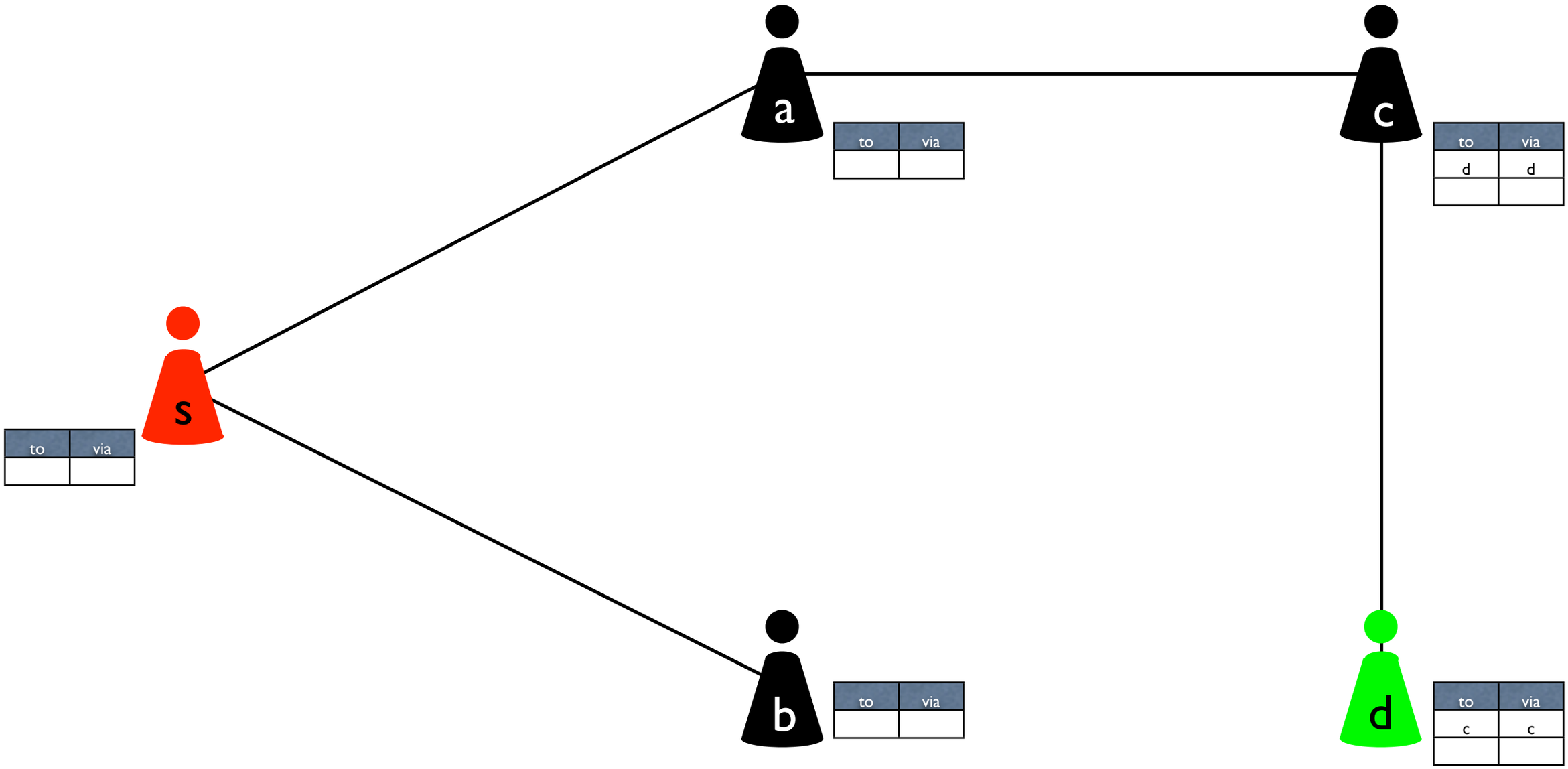

s is looking for a route to d

# AODV – An Example

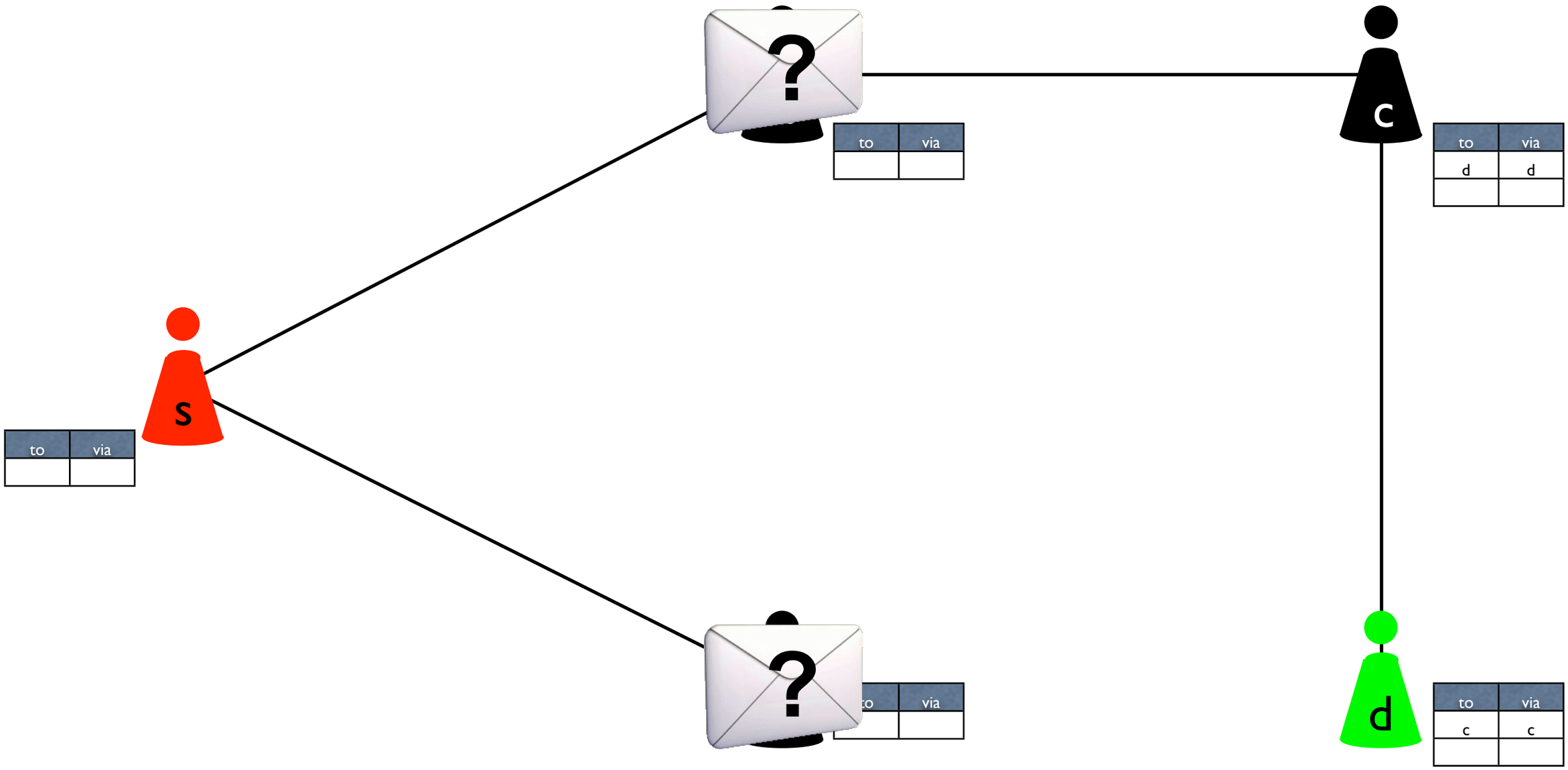
NICTA



# AODV – An Example

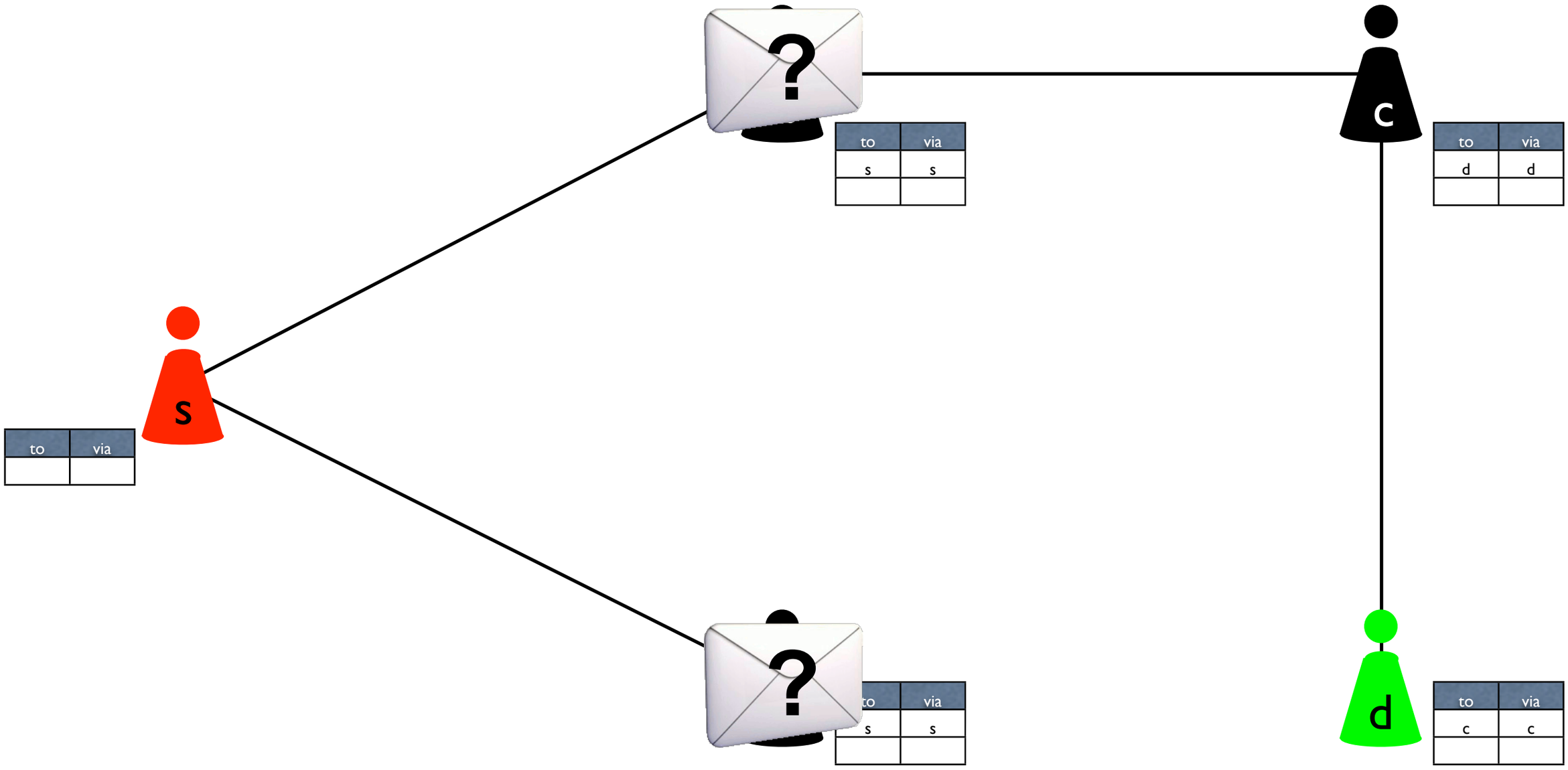


# AODV – An Example




s broadcasts a route request

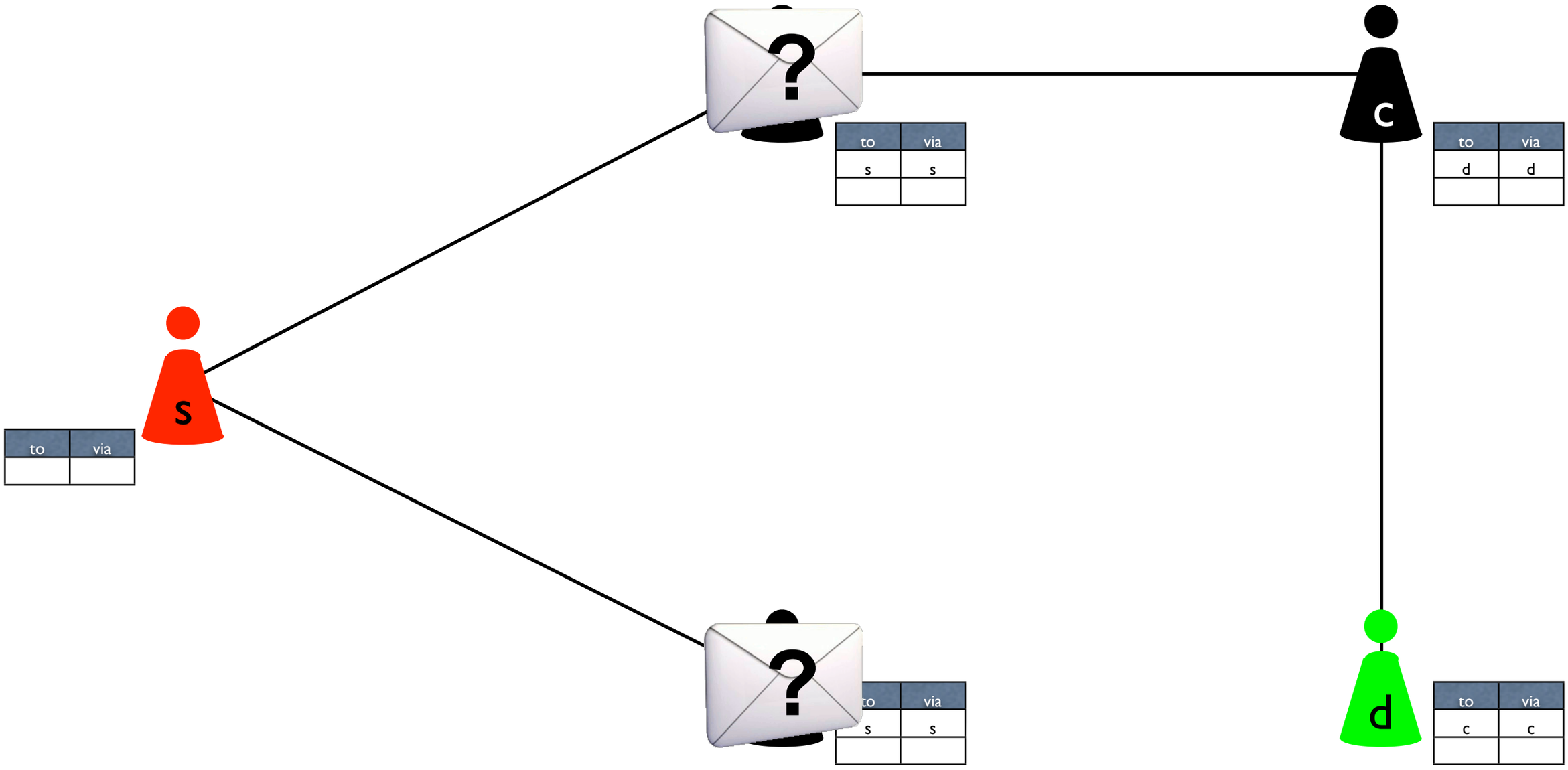
# AODV – An Example




s broadcasts a route request

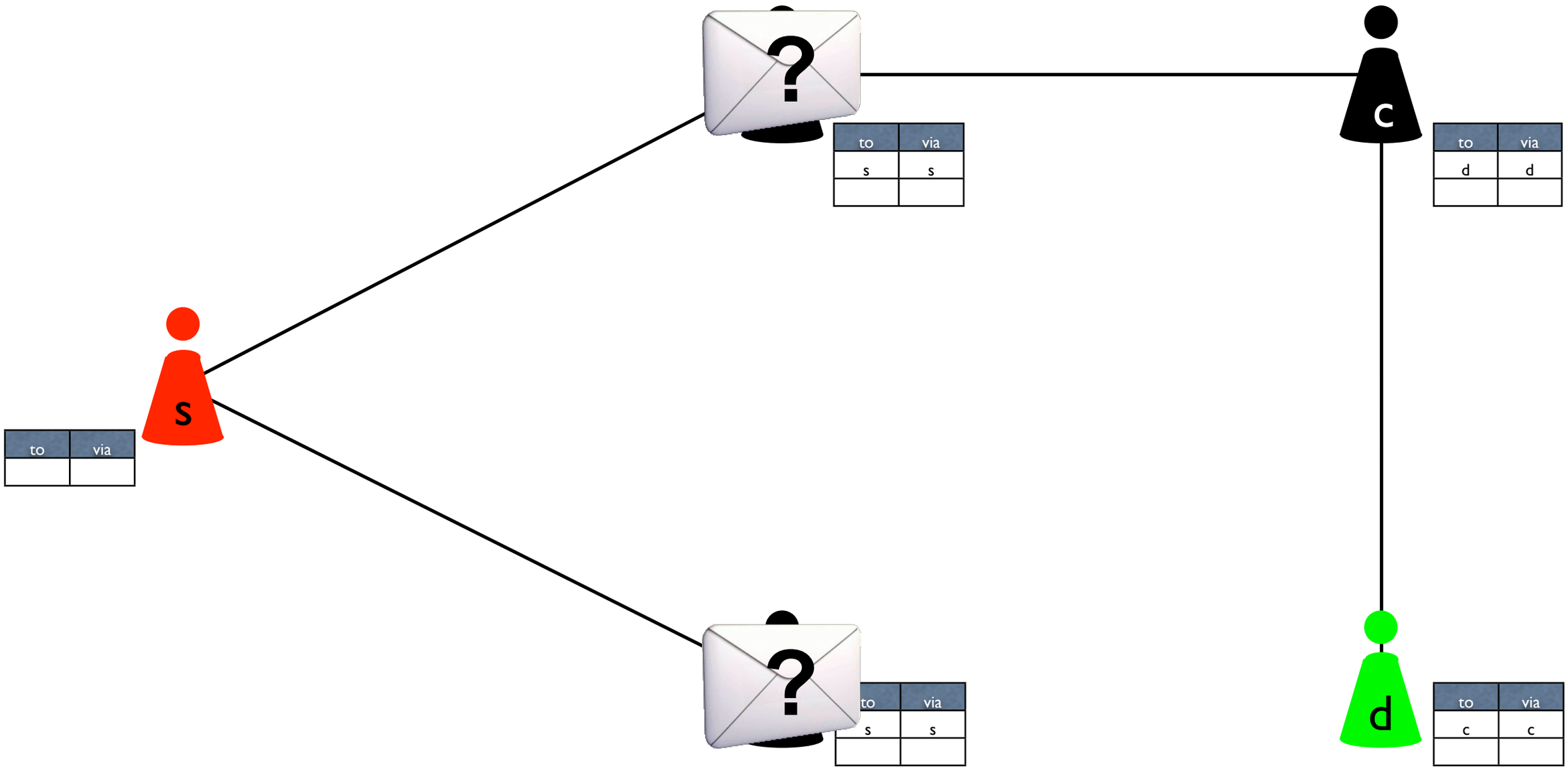
# AODV – An Example

NICTA

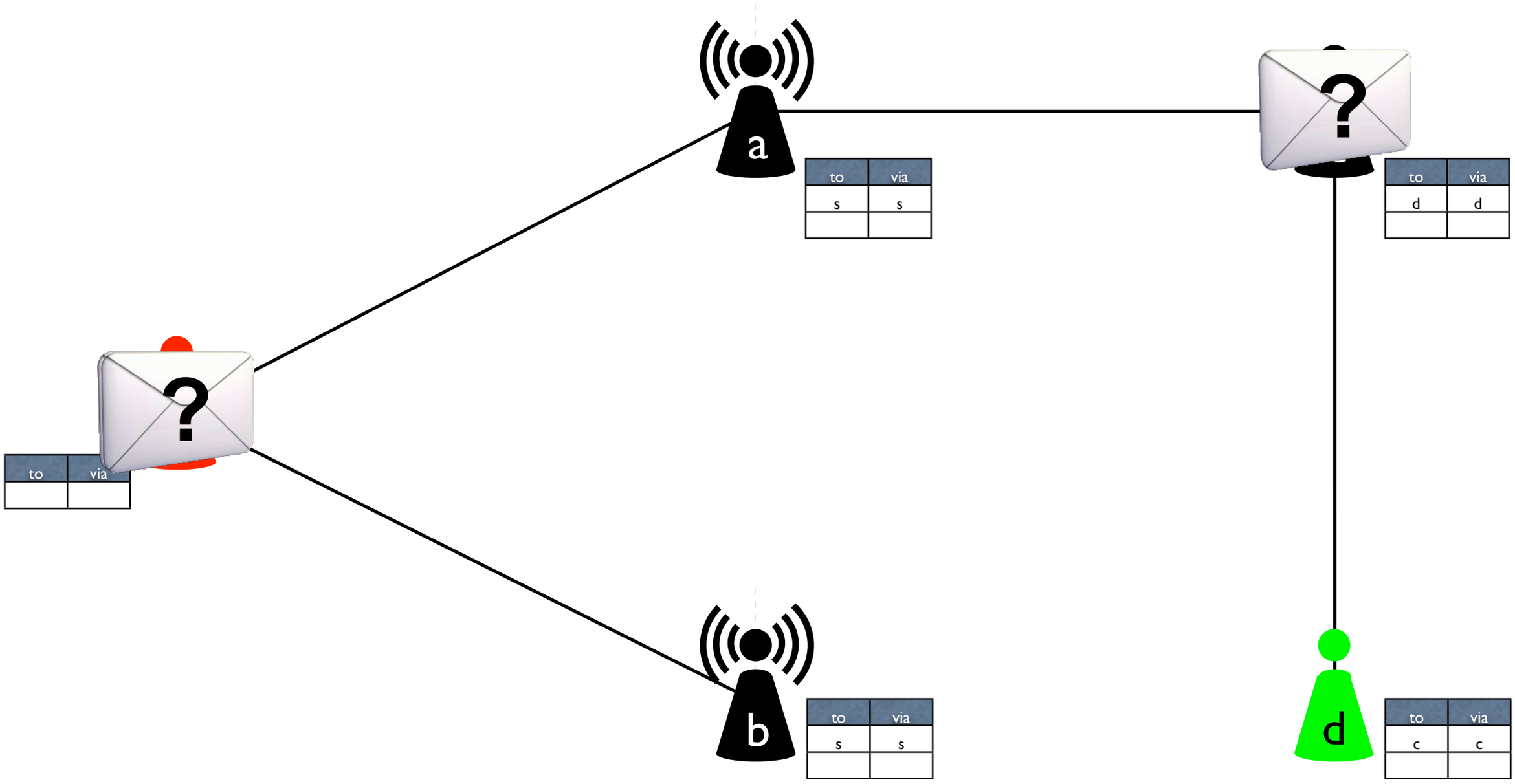


# AODV – An Example

NICTA

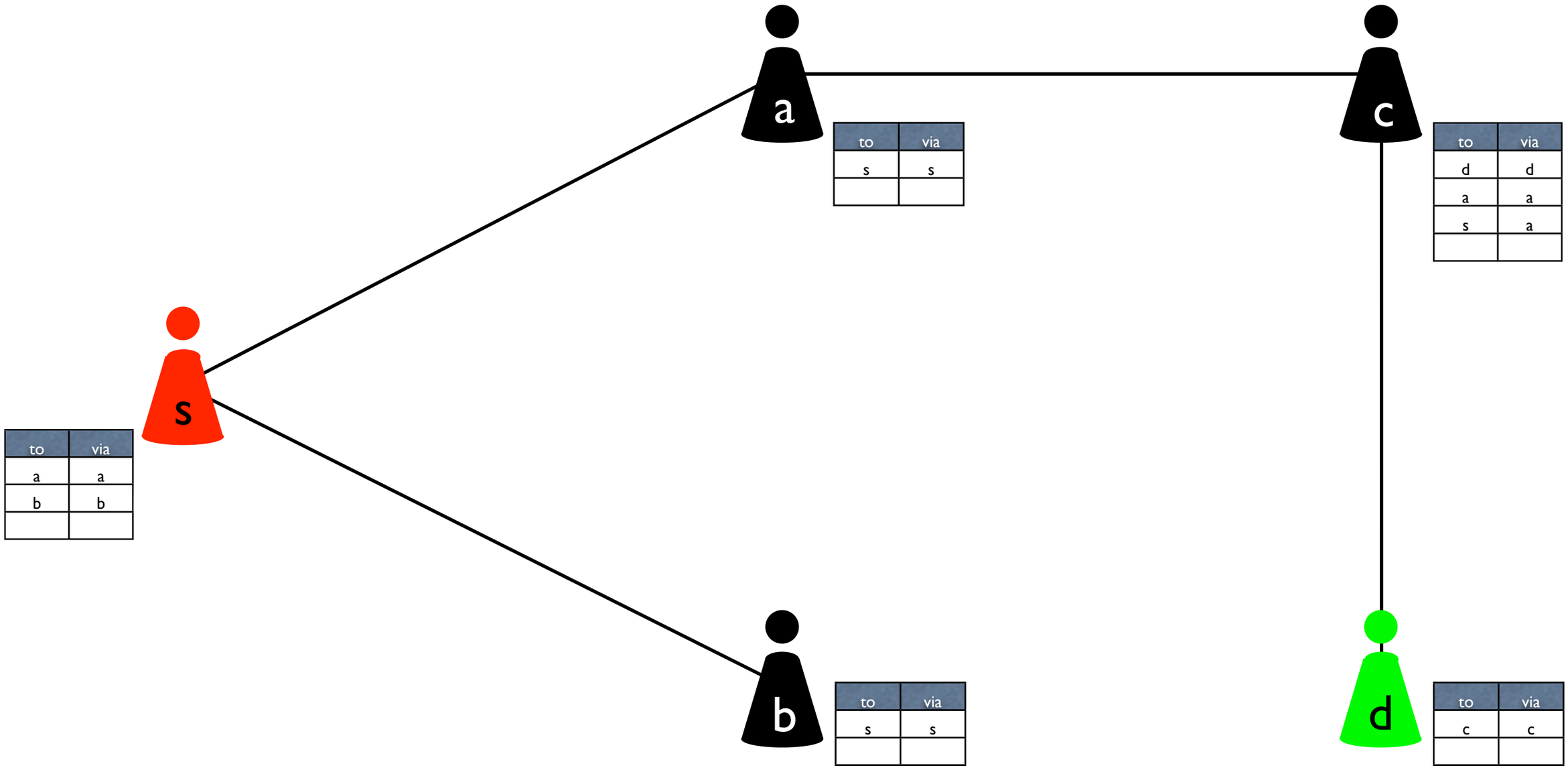


# AODV – An Example



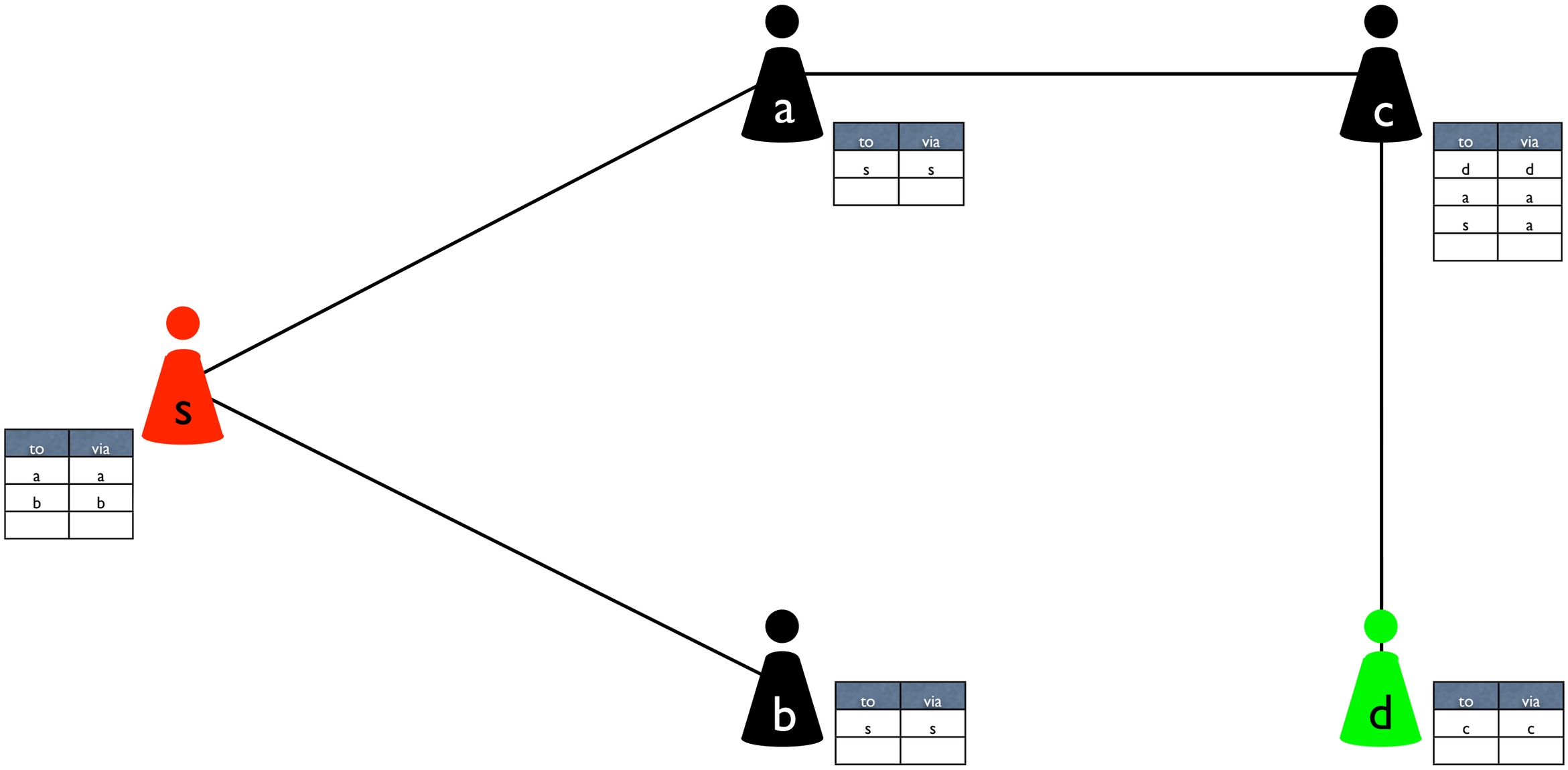

a,b forward the route request

# AODV – An Example



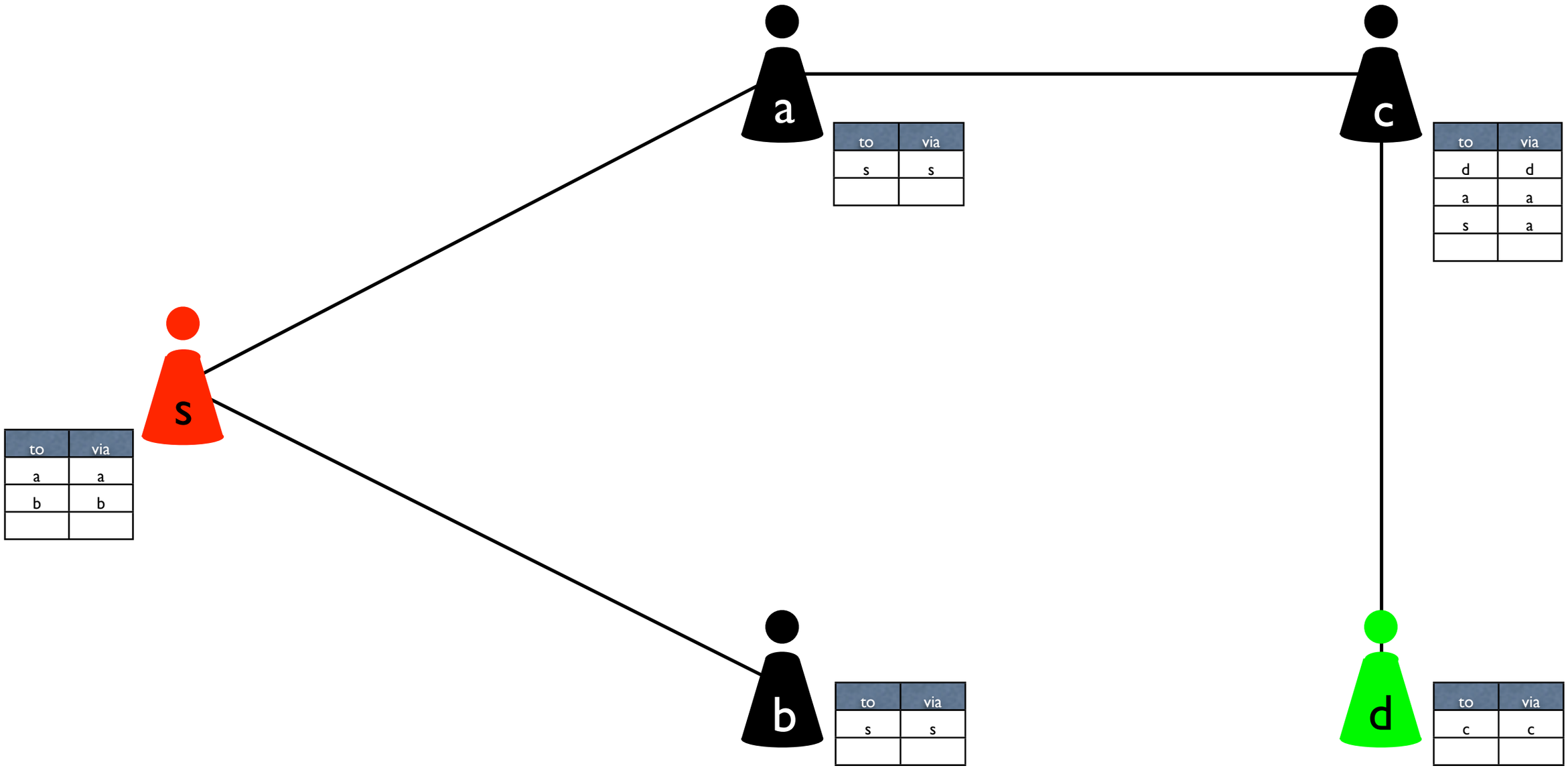

a,b forward the route request

# AODV – An Example

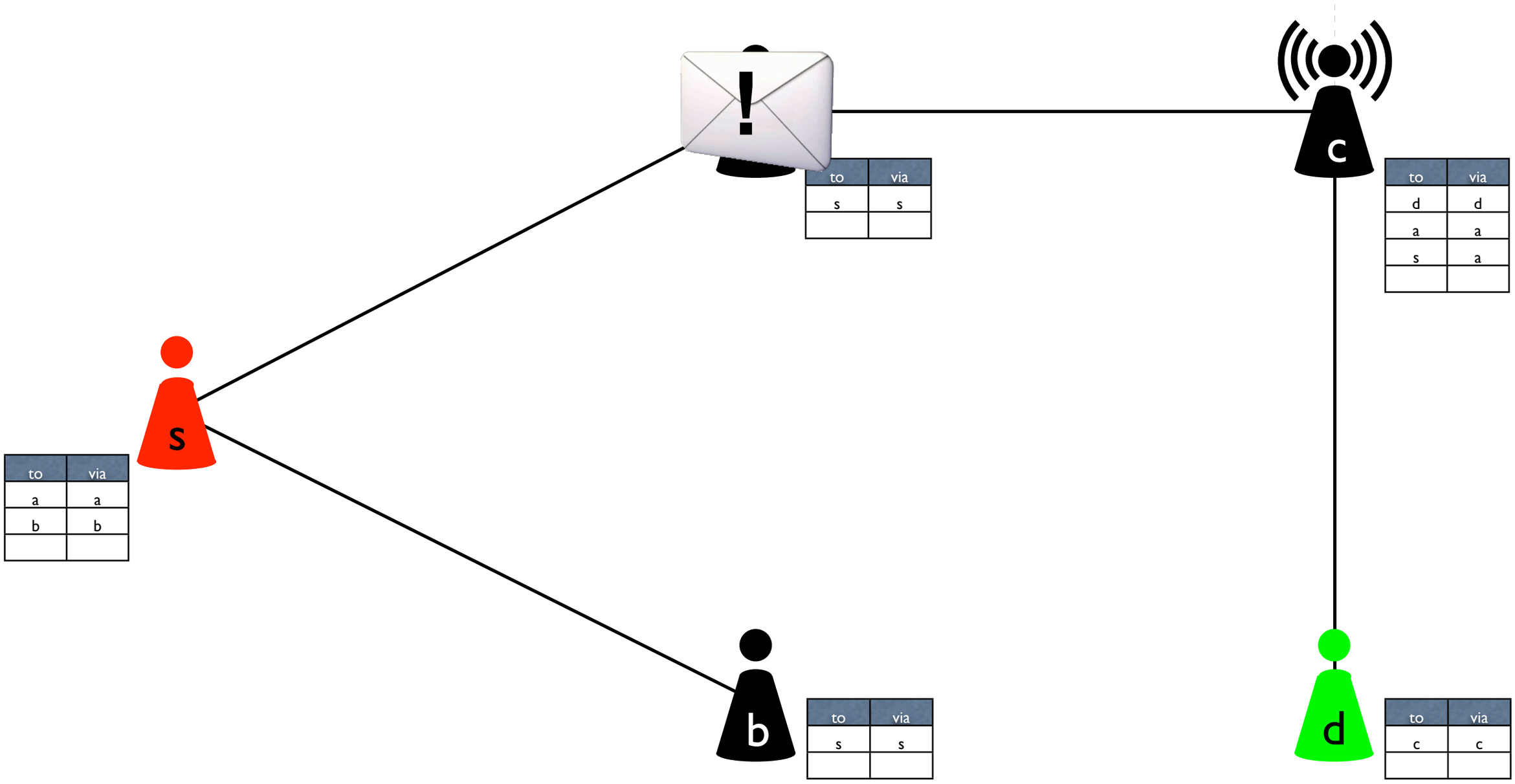


# AODV – An Example

NICTA



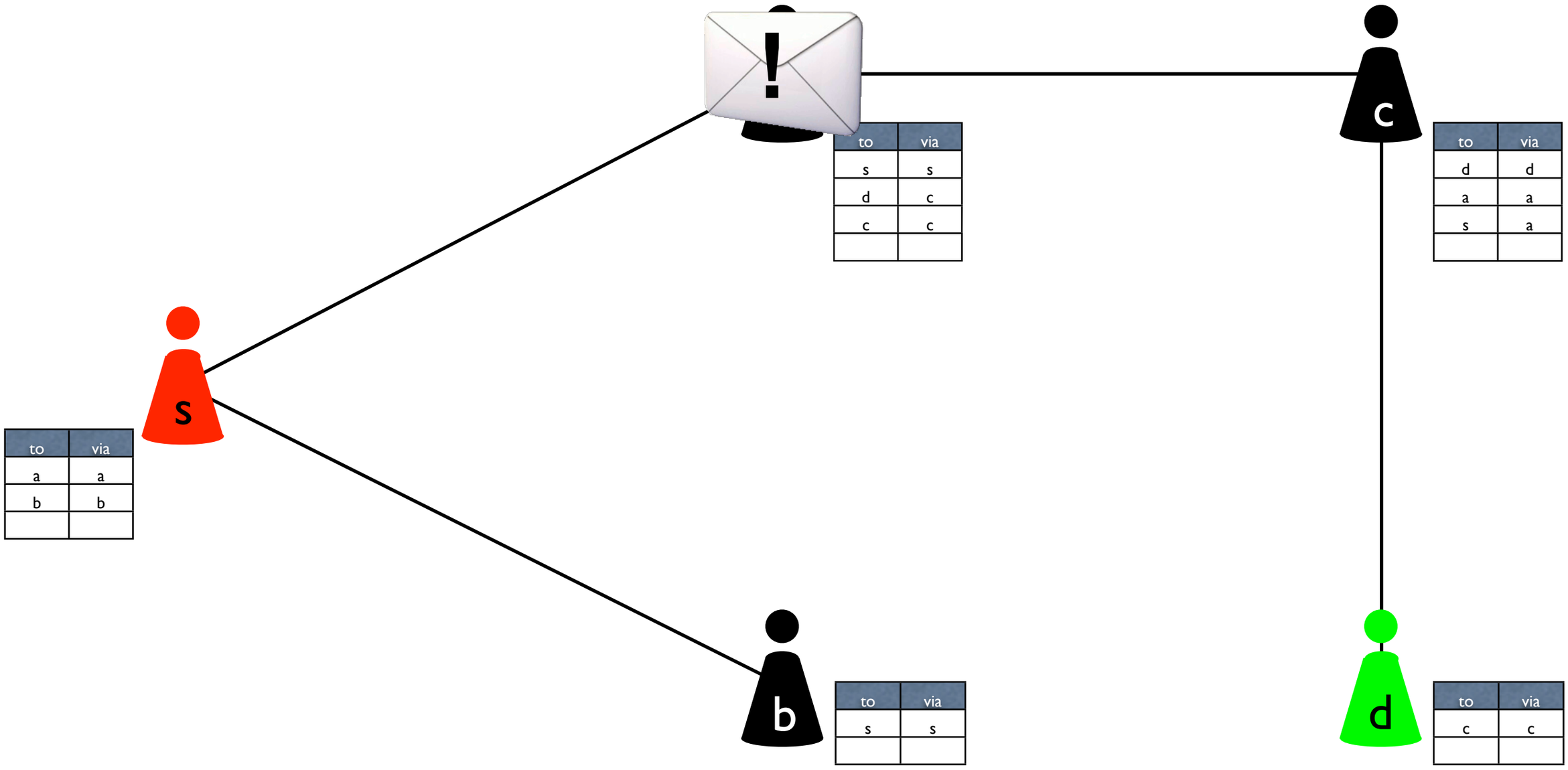
# AODV – An Example




c has information about d

c answers route request and sends reply

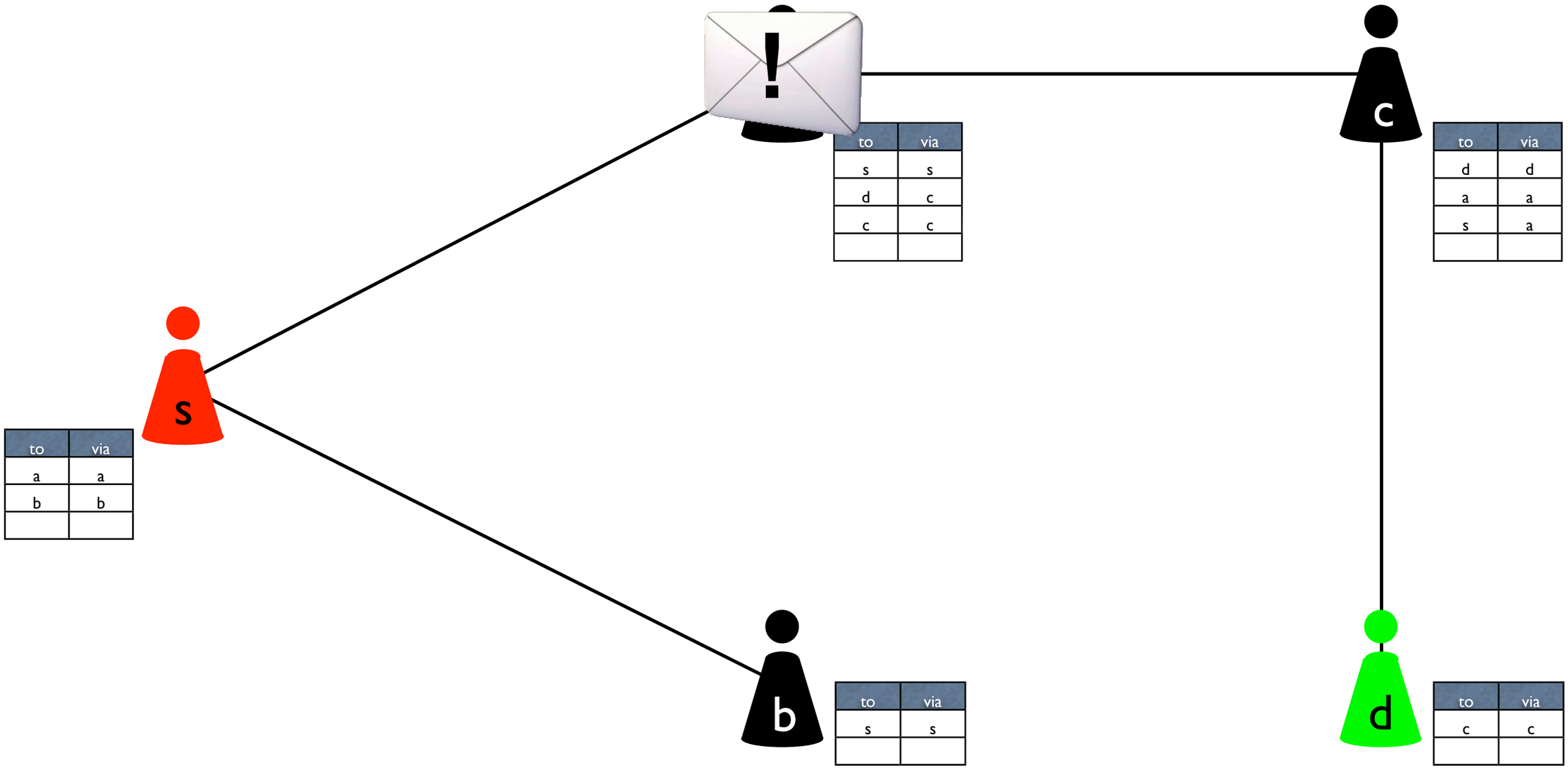
# AODV – An Example



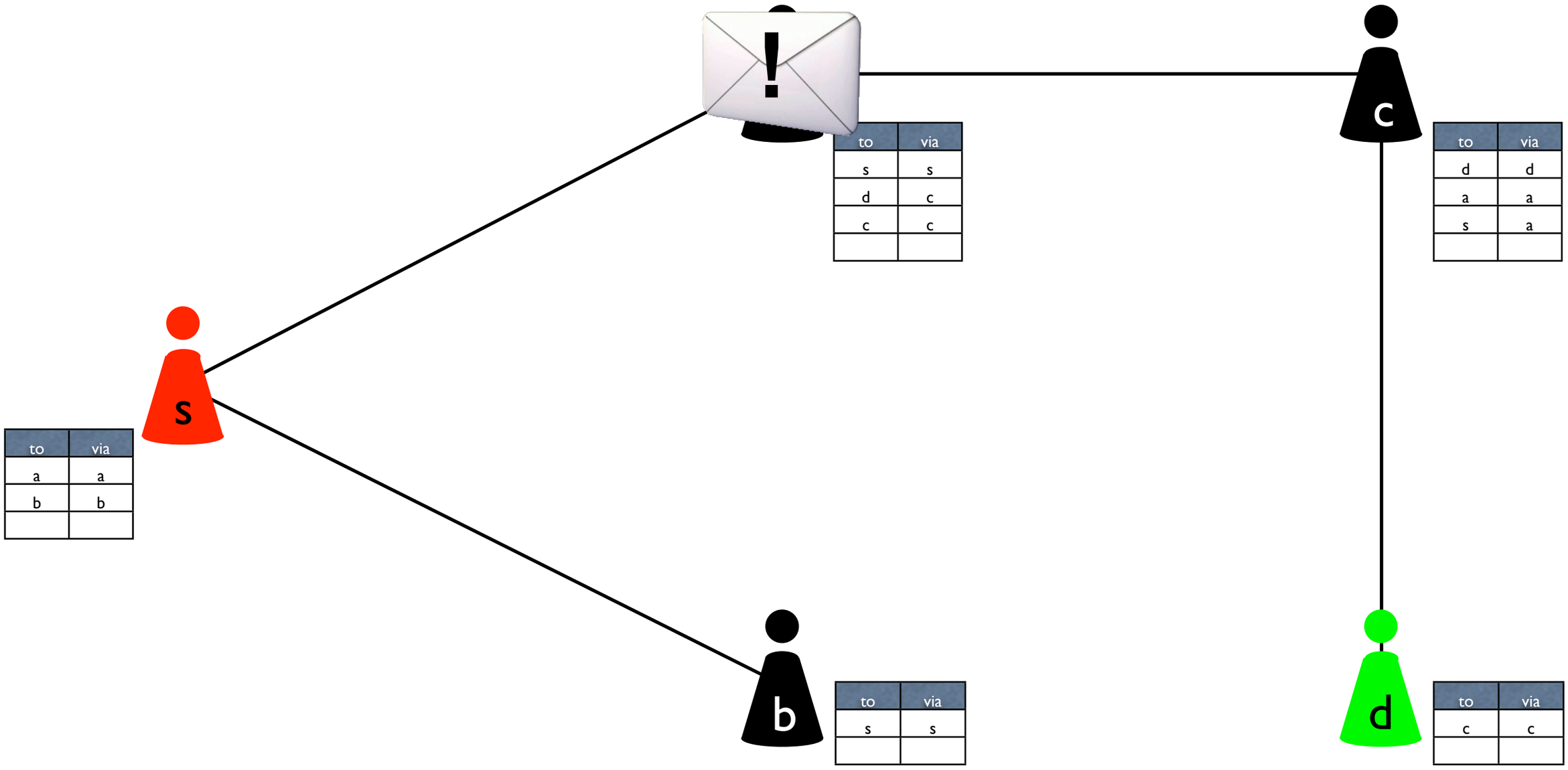

c has information about d

c answers route request and sends reply

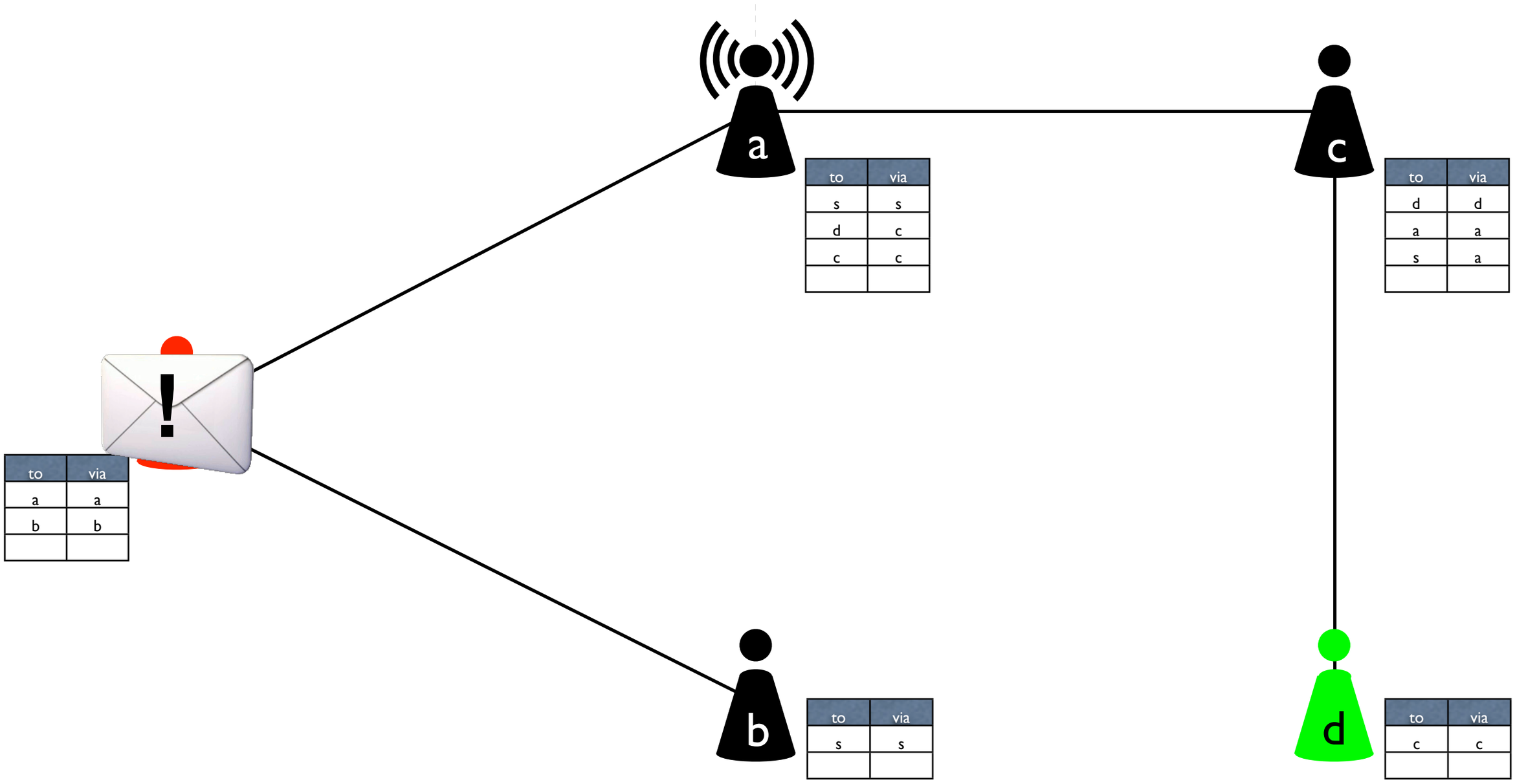
# AODV – An Example



# AODV – An Example

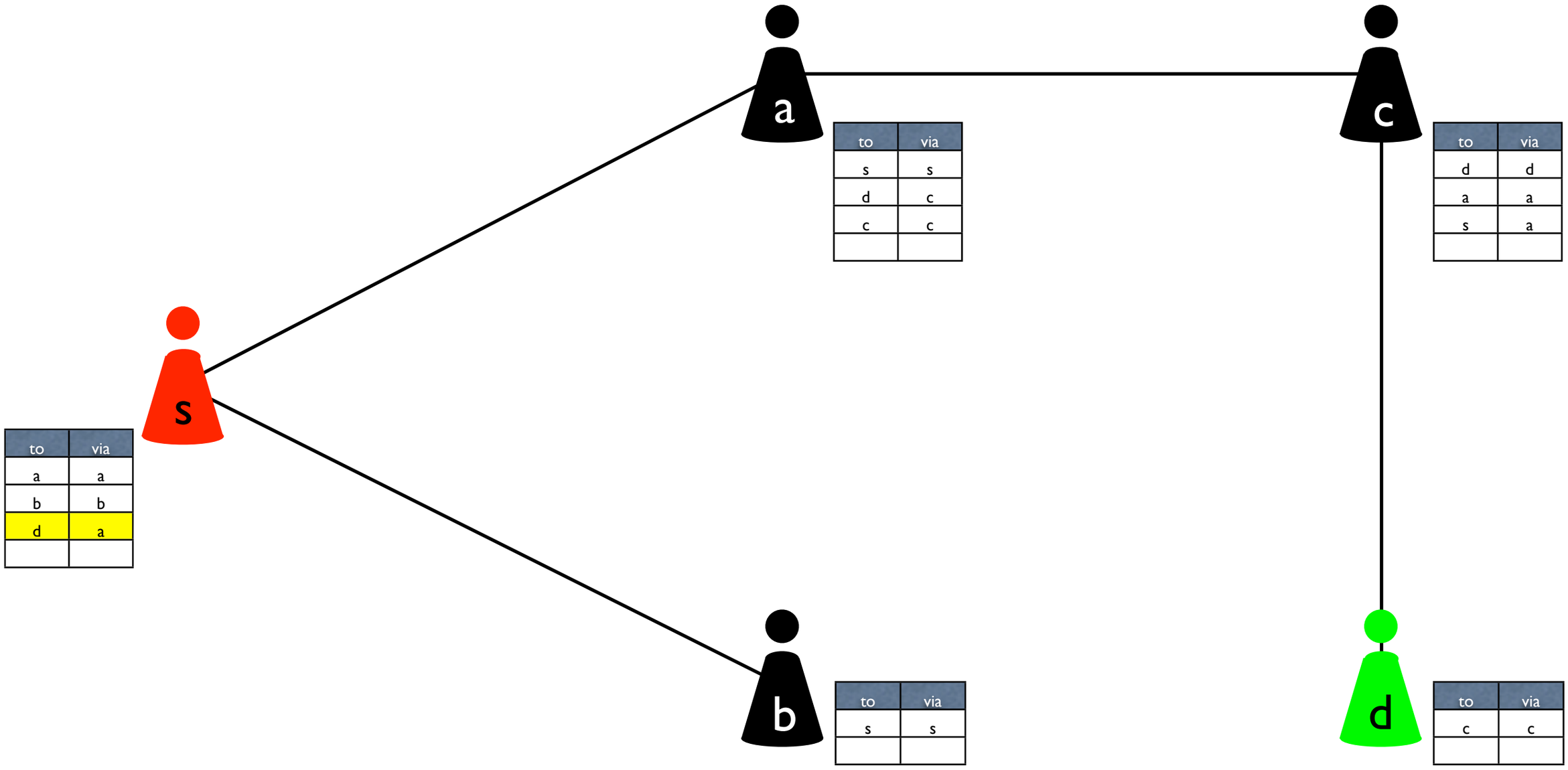


# AODV – An Example



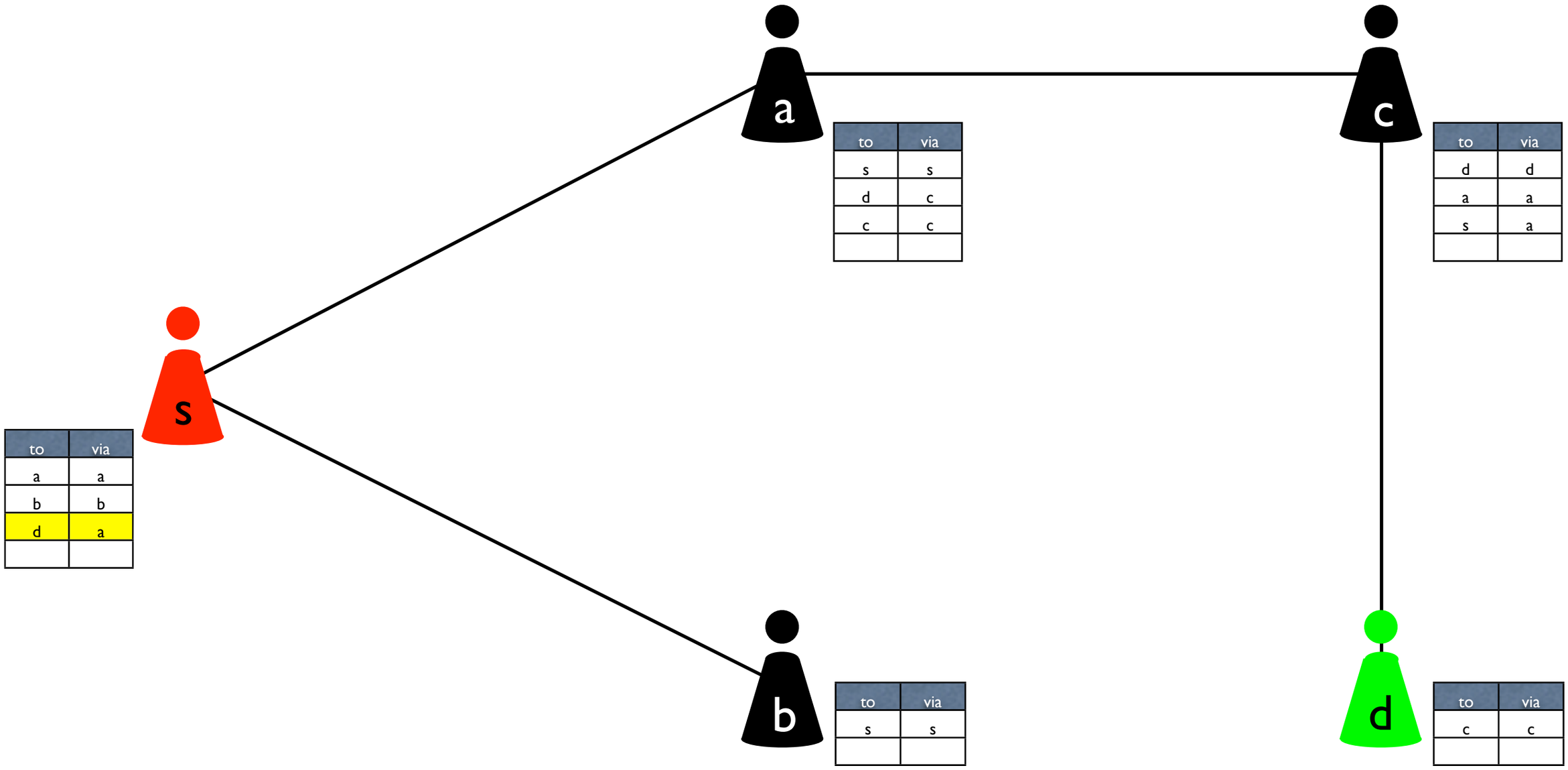

a forwards route reply

# AODV – An Example



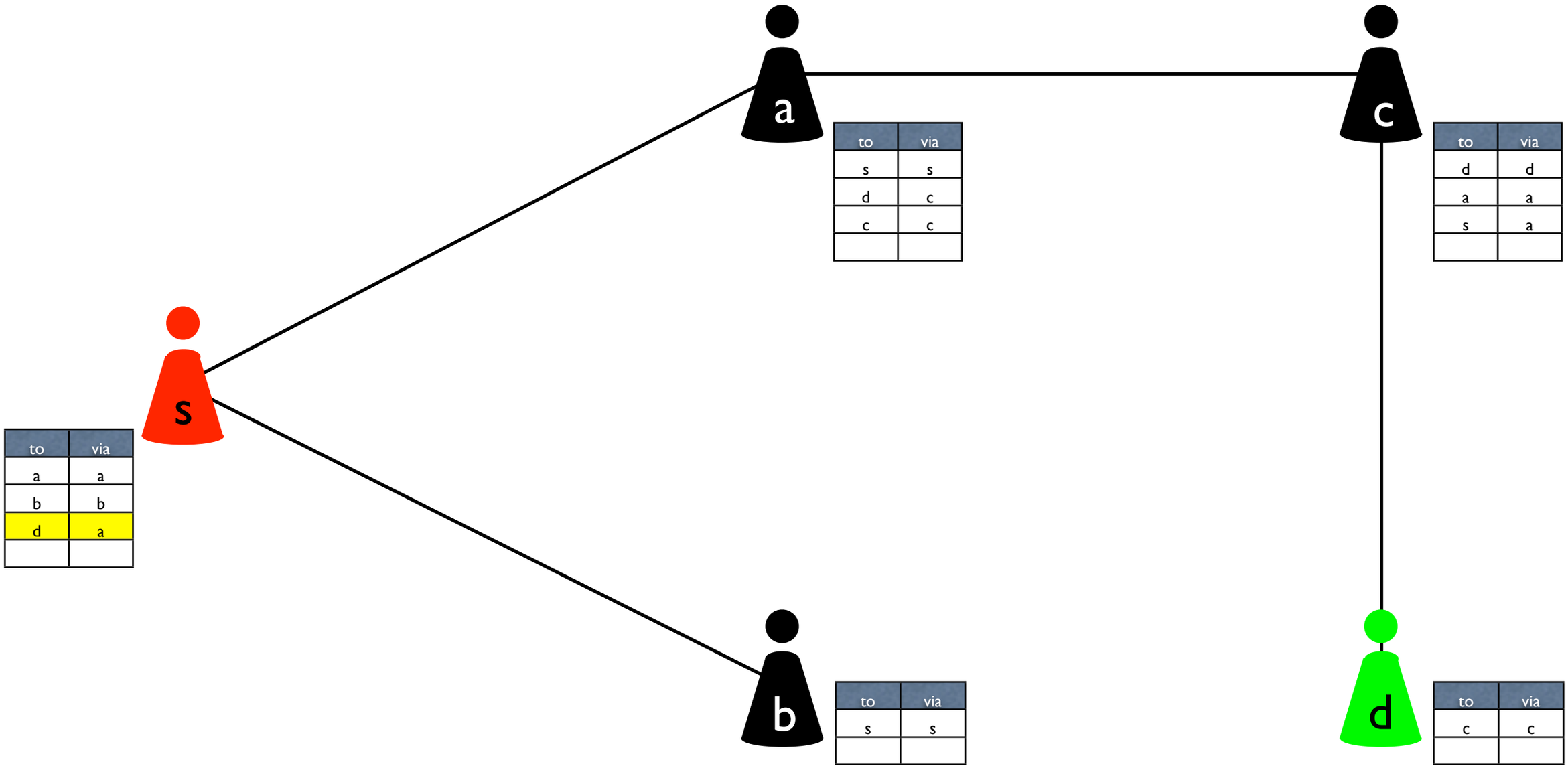

a forwards route reply

# AODV – An Example

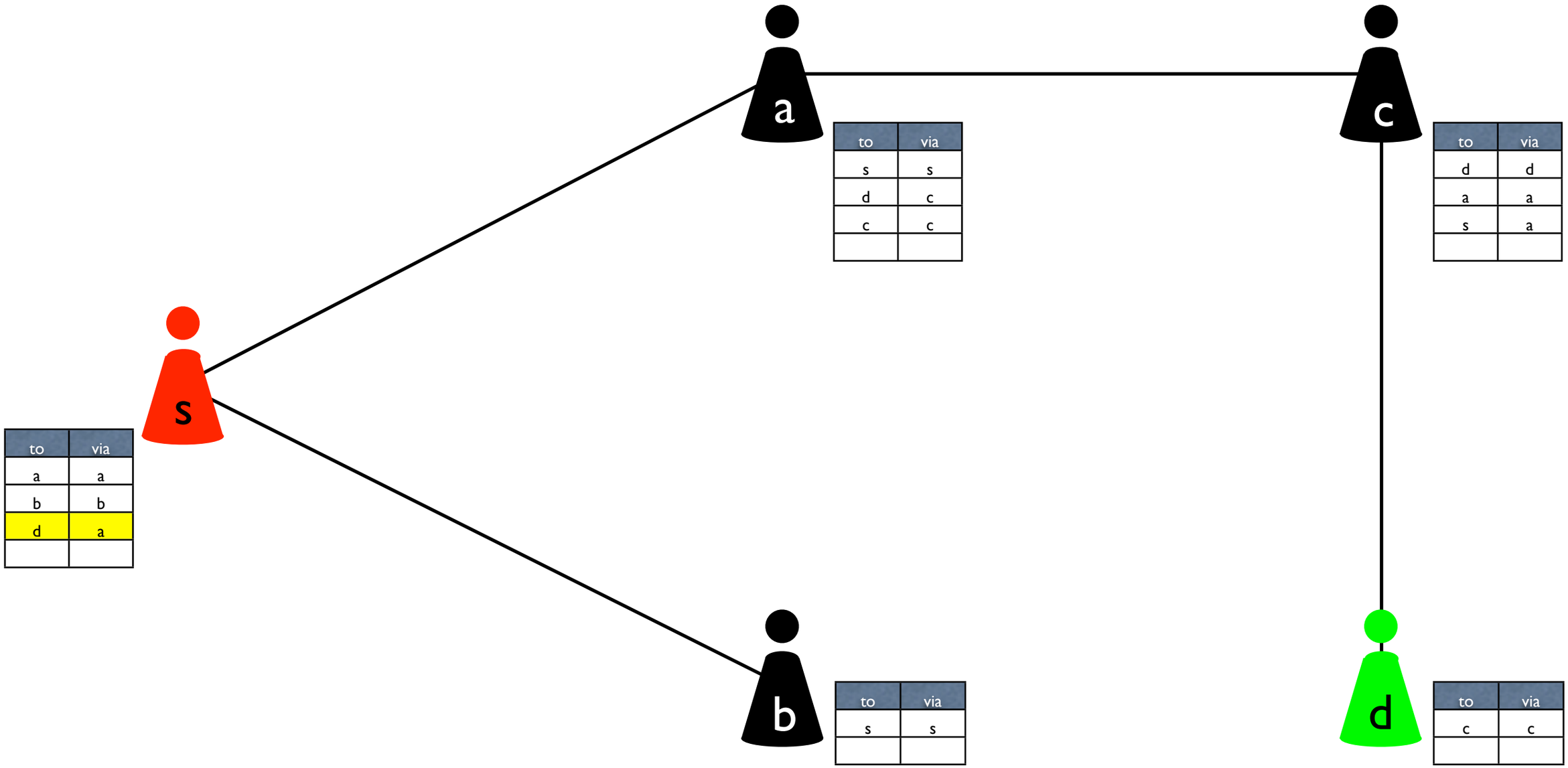


# AODV – An Example

NICTA

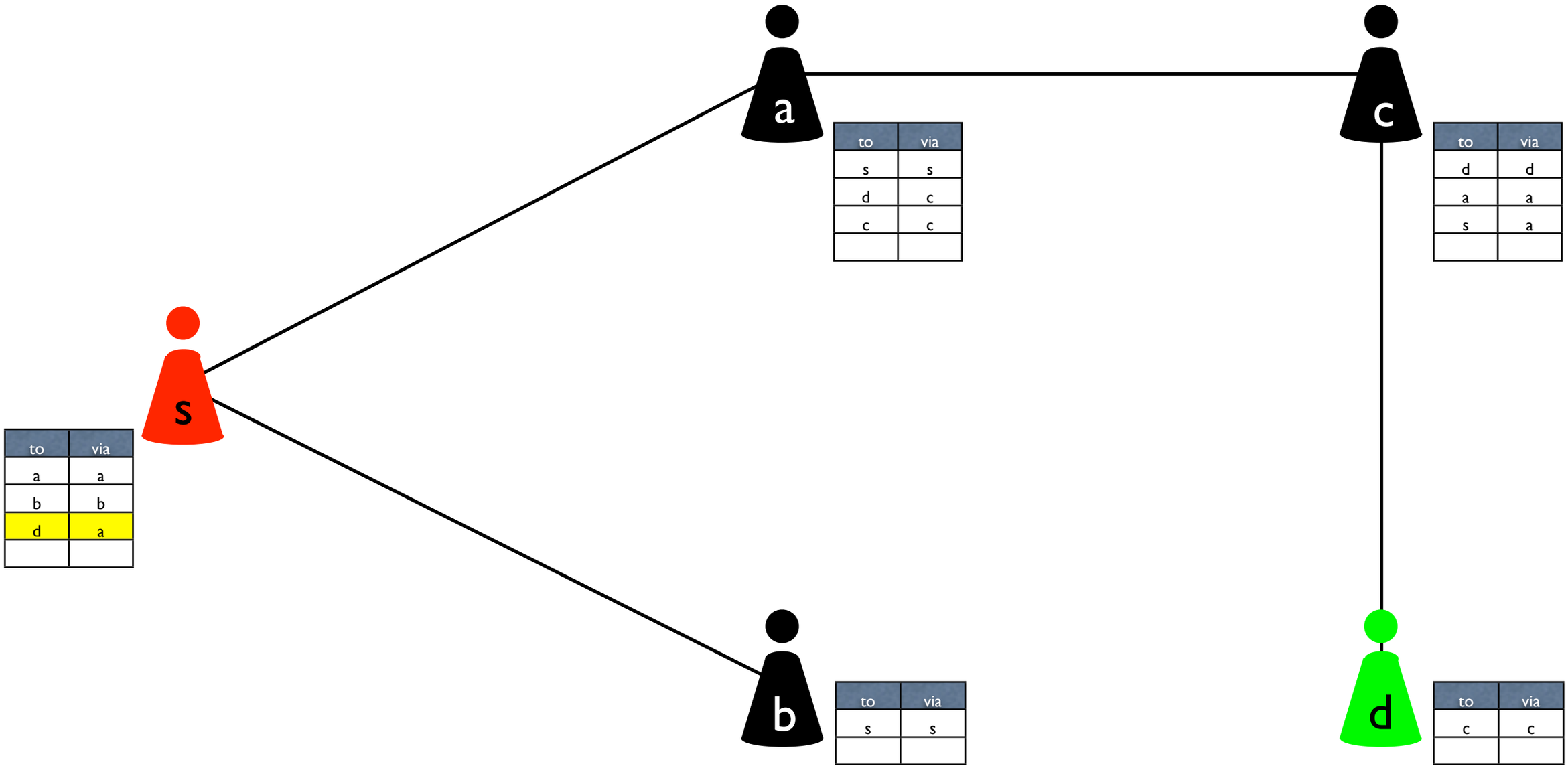


# AODV – An Example



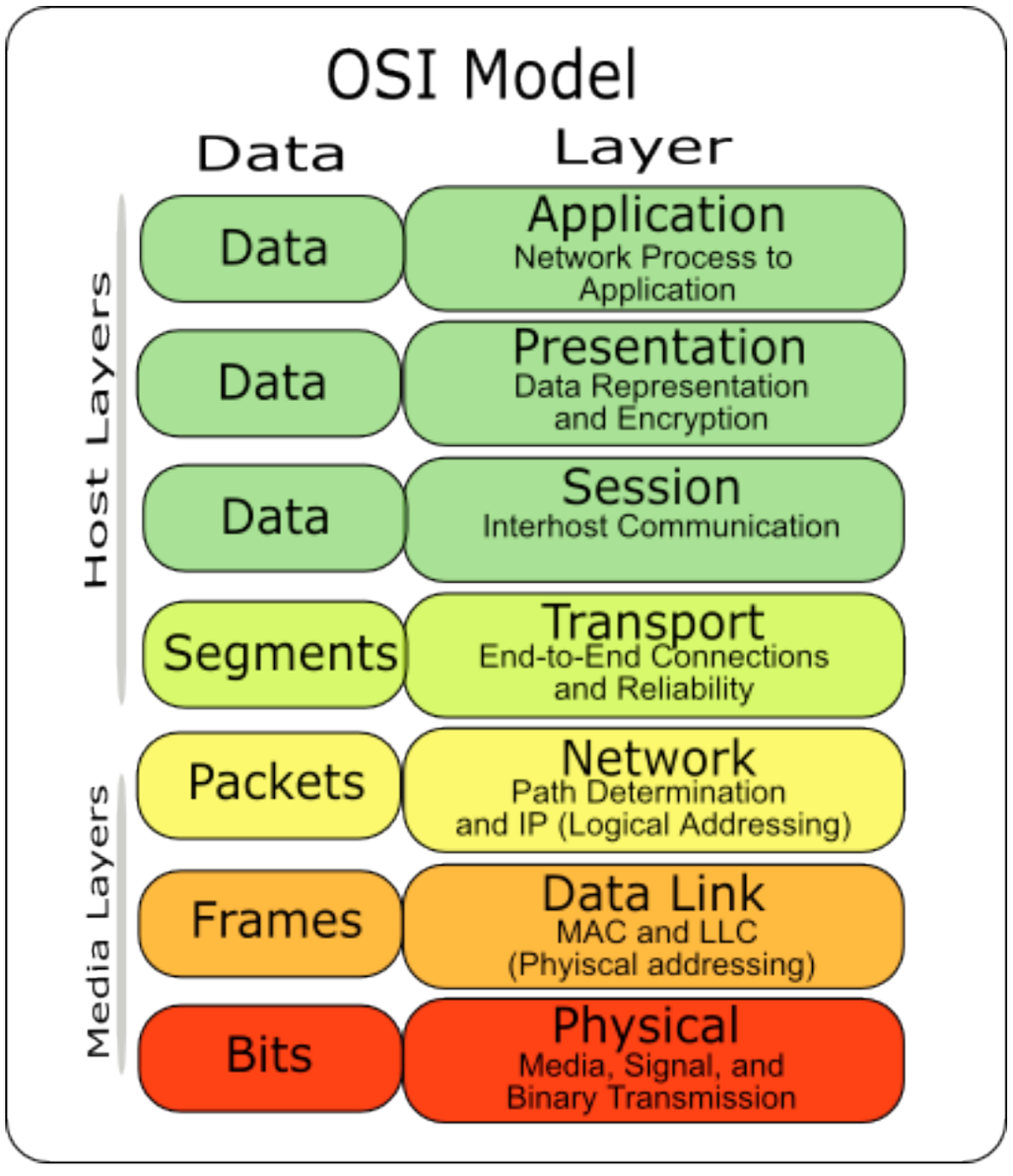

s has found a route to d

# AODV – An Example




s has found a route to d

# Different Network Layers




© NICTA 2011

# 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

© NICTA 2011