Formal Methods for Wireless Mesh Networks

Peter Höfner

Rome
IFIP 2.1 Working Group Meeting
February 6-10, 2012

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

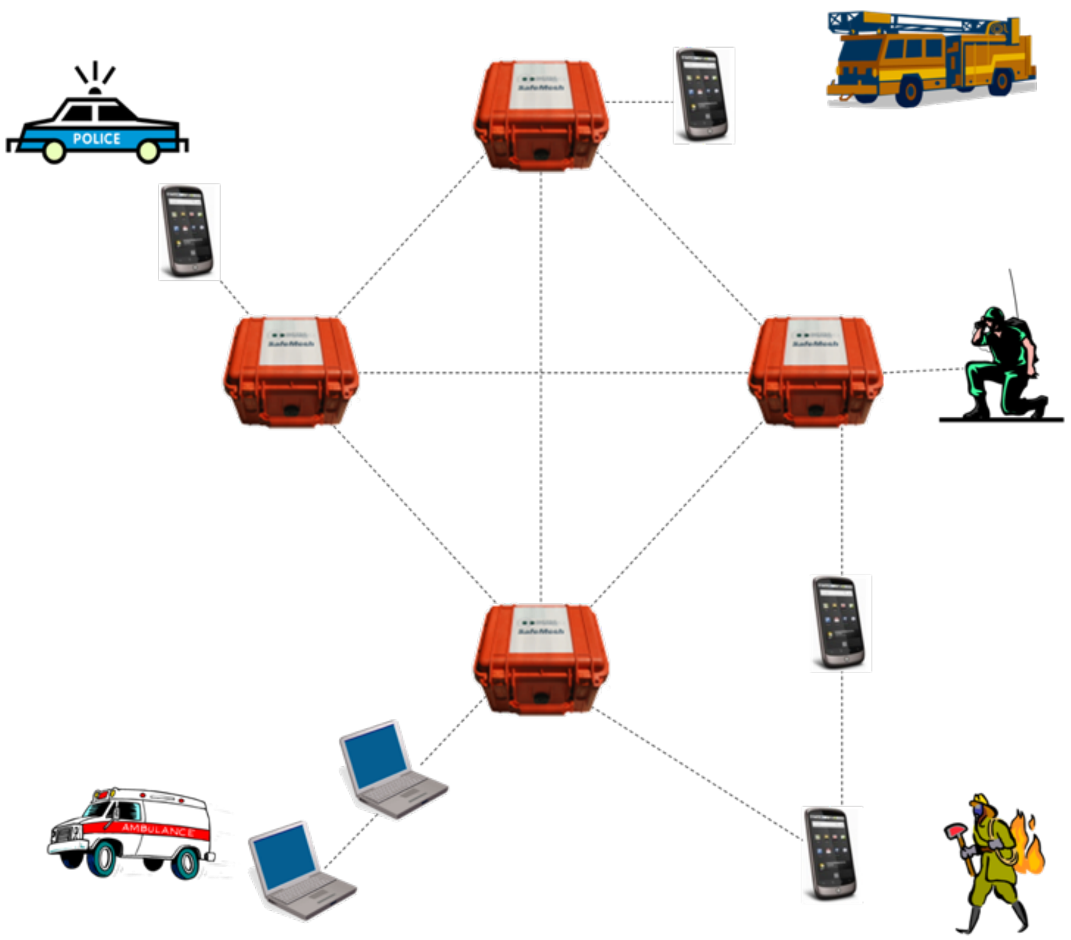
NICTA Members

NICTA Partners

What is the Problem?

NICTA

- 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



Formal Methods for Mesh Networks

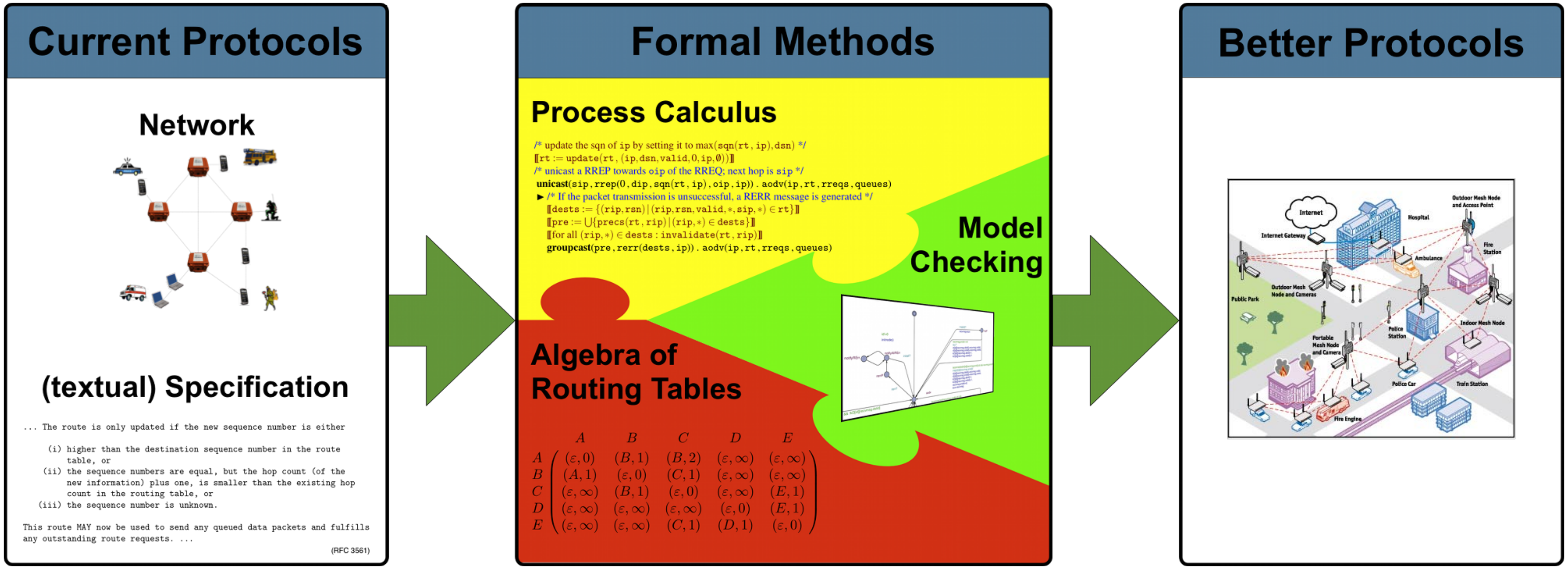
- Goal
  - model, analyse, verify and increase the performance of routing protocols for wireless mesh networks
  - develop suitable formal methods languages and techniques
- Benefits
  - more reliable protocols
  - finding and fixing bugs
  - better performance
  - proving correctness
  - reduce “time-to-market”
- Team (Formal Methods)
  - Ansgar Fehnker, Rob van Glabbeek, Peter Höfner, Annabelle McIver, Marius Portmann, Wee Lum Tan

Main Ingredients for WMNs

- Network with mobile nodes and dynamic topology
- Messages, which are sent through the network
  - route request (RREQ)
  - route reply (RREP)
  - route error (RERR)
  - ...
- Communication (message sending)
  - broadcast
  - unicast
  - groupcast (multicast/iterative unicast)
- Data
  - routing tables
  - node names

Formal Methods for Mesh Networks

- Two Approaches
  - Avoiding the discussion of it
  - LAoP



NICTA

Avoiding the discussion of it

Process Algebra

NICTA

```
+ [ (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)]]
```

Process Algebra

- Desired Properties
  (implies the creation of new process algebra)
  - guaranteed broadcast
  - conditional unicast
  - data structure

- Inspired by
  - π - Calculus (no creation of nodes)
  - ω - Calculus
  - (LOTOS)

Structure of WMNs

- 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)

NICTA

- Syntax of sequential process expressions

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

Structural Operational Semantics I

- Internal state determined by expression and valuation

$$\xi, \mathbf{broadcast}(ms).p \xrightarrow{\mathbf{broadcast}(\xi(ms))} \xi, p$$

$$\xi, \mathbf{groupcast}(dests, ms).p \xrightarrow{\mathbf{groupcast}(\xi(dests),\xi(ms))} \xi, p$$

$$\xi, \mathbf{unicast}(dest, ms).p \blacktriangleright q \xrightarrow{\mathbf{unicast}(\xi(dest),\xi(ms))} \xi, p$$

$$\xi, \mathbf{unicast}(dest, ms).p \blacktriangleright q \xrightarrow{\neg\mathbf{unicast}(\xi(dest))} \xi, q$$

$$\xi, \mathbf{send}(ms).p \xrightarrow{\mathbf{send}(\xi(ms))} \xi, p$$

$$\xi, \mathbf{deliver}(data).p \xrightarrow{\mathbf{deliver}(\xi(data))} \xi, p$$

$$\xi, \mathbf{receive}(\mathtt{msg}).p \xrightarrow{\mathbf{receive}(m)} \xi[\mathtt{msg} := m], p \qquad (\forall m \in \mathtt{MSG})$$

Network

- Node expressions: $M ::= \quad ip : P : R \quad | \quad M \| M$
- Operational Semantics (snippet)

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

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

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

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

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
- Resulting algebra is in *de Simone* format (by this strong bisimulation are congruences)
- Both parallel operators are associative (follows by a meta result of Cranen, Mousavi, Reniers)

Case Study

- Ad hoc On-Demand Distance Vector (AODV) Routing Protocol
- Achievements
  - full concise specification of AODV (RFC 3561) (no time)
  - verified/disproved properties
    - route discovery
    - packet delivery
    - loop freedom
      - first (correct) proof
      - disproved loop freedom for variants of AODV (as implemented in at least open source implementation)
  - found several ambiguities, mistakes, shortcomings
  - found solutions for some limitations

Avoiding the discussion of it

- "The destination sequence number of this routing entry, if it exists and is valid, is incremented [...]"

- "The route is only updated if the new sequence number is either (i) [...], or (iii) the sequence number is unknown."

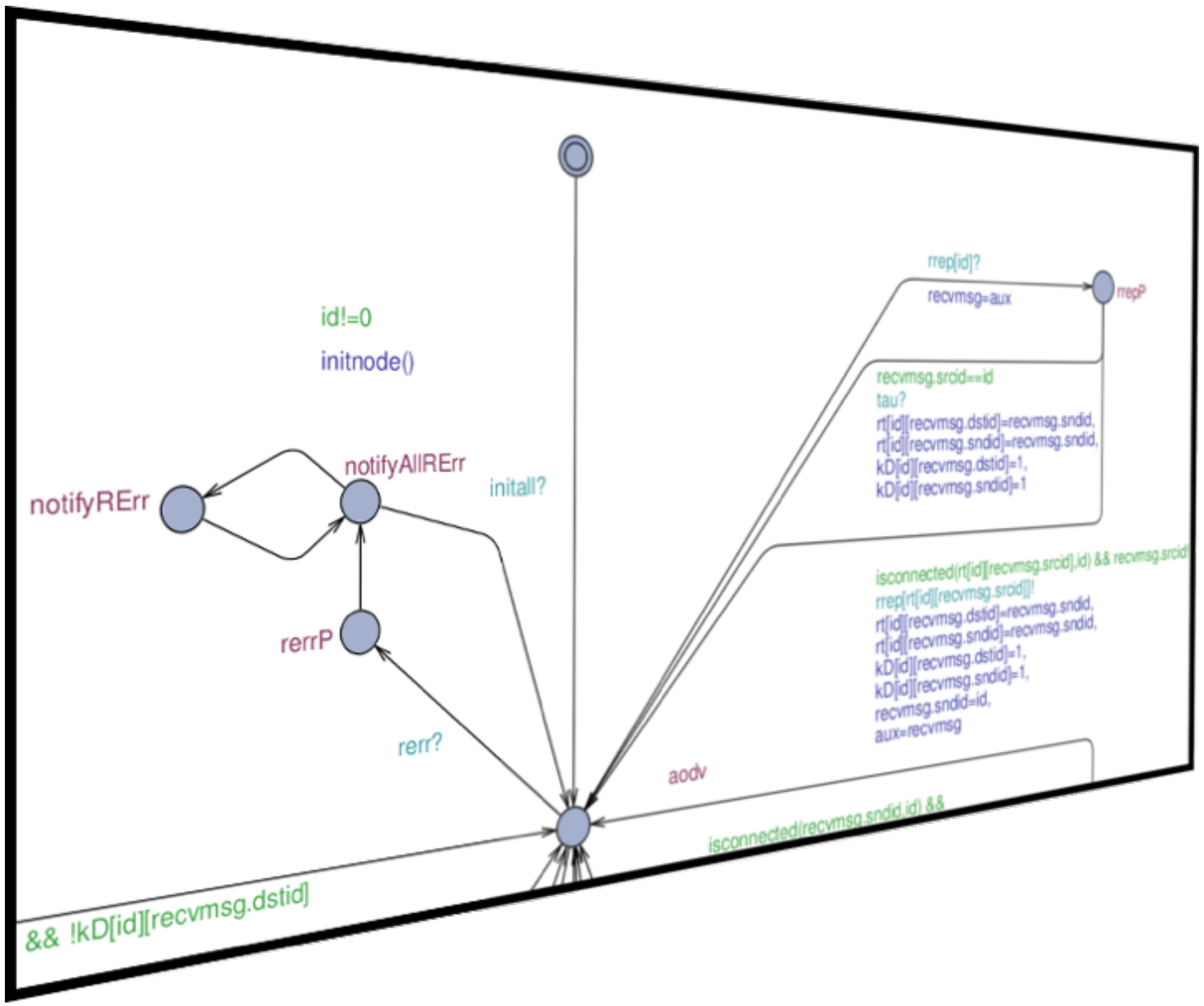
[RFC3561- AODV Specification]

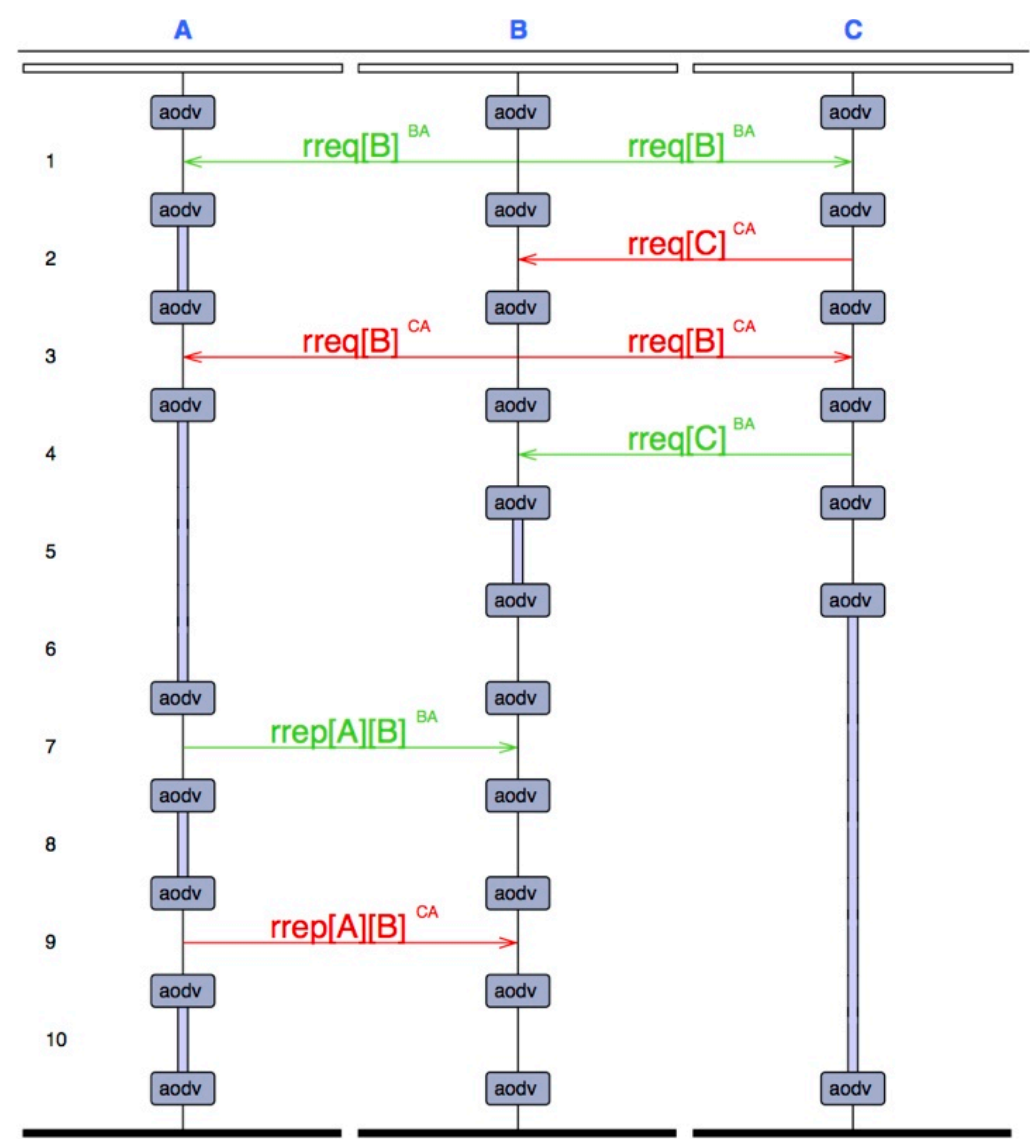
Process Algebra

```
[[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
```

- Formal language
- Readable for software/network engineers

Model Checking



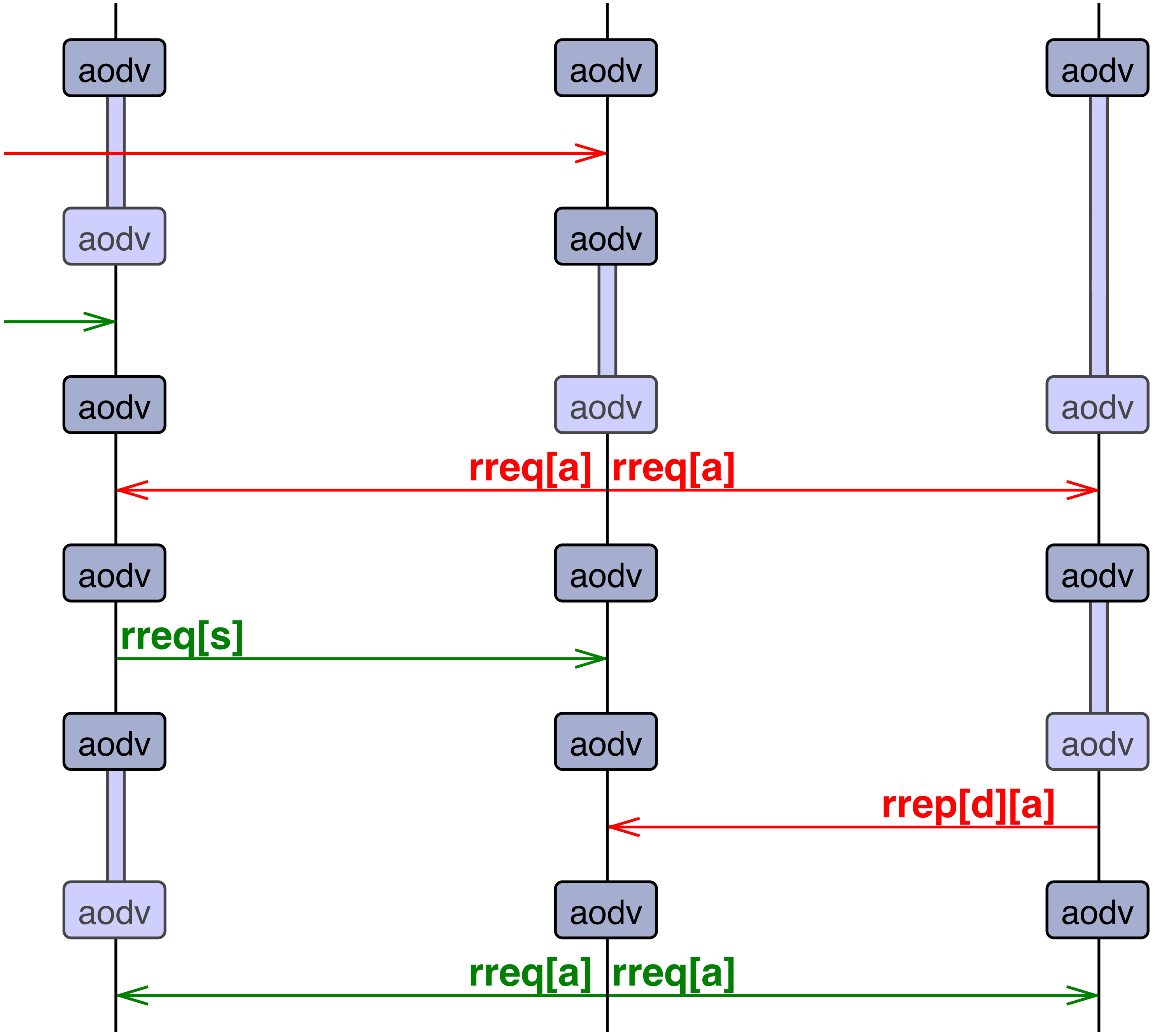


Model Checking

- 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

Results: Route Discovery (2004)

- Route discovery fails in a linear 3-node topology



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

LAoP

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

- Matrices over routing table entries

$$
\begin{array}{c}
 \\ A \\ B \\ C \\ D \\ \vdots
\end{array}
\begin{array}{c}
\begin{array}{ccccc} A & B & C & D & \ldots \end{array} \\
\left(\begin{array}{ccccc}
(\_, 0) & (B, 1) & (B, 2) & (\_, \infty) & \\
(A, 1) & (\_, 0) & (C, 1) & (\_, \infty) & \ldots \\
(\_, \infty) & (B, 1) & (\_, 0) & (\_, \infty) & \\
(\_, \infty) & (\_, \infty) & (\_, \infty) & (\_, 0) & \\
 & \vdots & & & \ddots
\end{array}\right)
\end{array}
\text{routing table of } A
$$

"routes" to B

- Standard matrix operations

Routing Algebra – Elements, Operators

- Routing table entries $(\mathtt{nhip}\,,\,\mathtt{hops})$
  (next hop, distance)

- 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

Further Abstraction

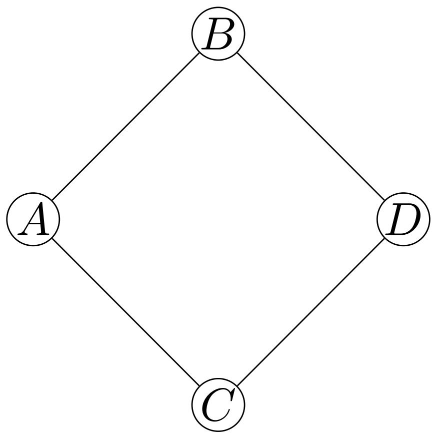
- Interpret matrix as a semiring element
- Kleene algebra allows iteration,
- (Co)Domain and tests model projections

Routing Algebra

- Achievements
  - model main aspects of routing protocols (message sending, routing table update)
  - full abstraction to algebraic structures
  - enables use of automated theorem provers

Example

- A route request is broadcast



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

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

NICTA

- Sending messages

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

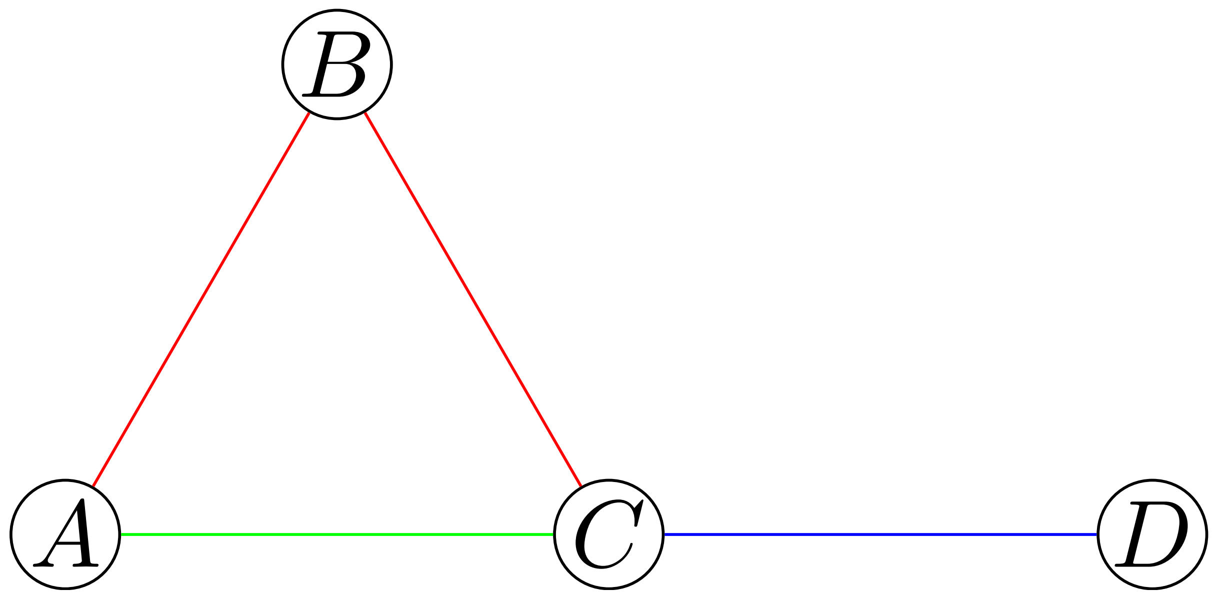
- By distributivity

$$a \quad + \quad p \cdot b \cdot q \quad + \quad p \cdot b \cdot q \cdot c$$

  snapshot, 1-hop connection learnt, content sent
- Broadcast, unicast, groupcast are the same (modelled by different topologies)
- Kleene star models flooding the network (modal operators terminate flooding)

Lost and Found

- Adding sequence numbers



$$
\begin{aligned}
r \cdot b &= (B, 2, 5) \cdot (D, 1, 10) &= (B \cdot D, 2 + 1, \max(5, 10)) &= (B, 3, 10) \\
g \cdot b &= (C, 1, 3) \cdot (D, 1, 10) &= (C \cdot D, 1 + 1, \max(3, 10)) &= (C, 2, 10)
\end{aligned}
$$

$$
r \cdot b + g \cdot b \neq (r + g) \cdot b
$$

Lost and Found

- Restrict multiplication
  - partial defined operation
  - only topologies allowed on the left-hand side
  - Kleene star has to be adapted

- Module-like structure
  (scalars are subalgebra)

Conclusion/Future Work

- So far
  - concentrated on basic language
    (process algebra / routing algebra)
  - considered only AODV (IETF-standard)

- Future
  - add additional necessary concepts
    (time probability)
  - include other protocols
    OSLR, DYMO, DSR, ...
  - define notions for protocol quality to compare protocols

NICTA

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

Appendix

Ad Hoc On-Demand Distance Vector Protocol

- 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

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

- 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
    SEND RERR

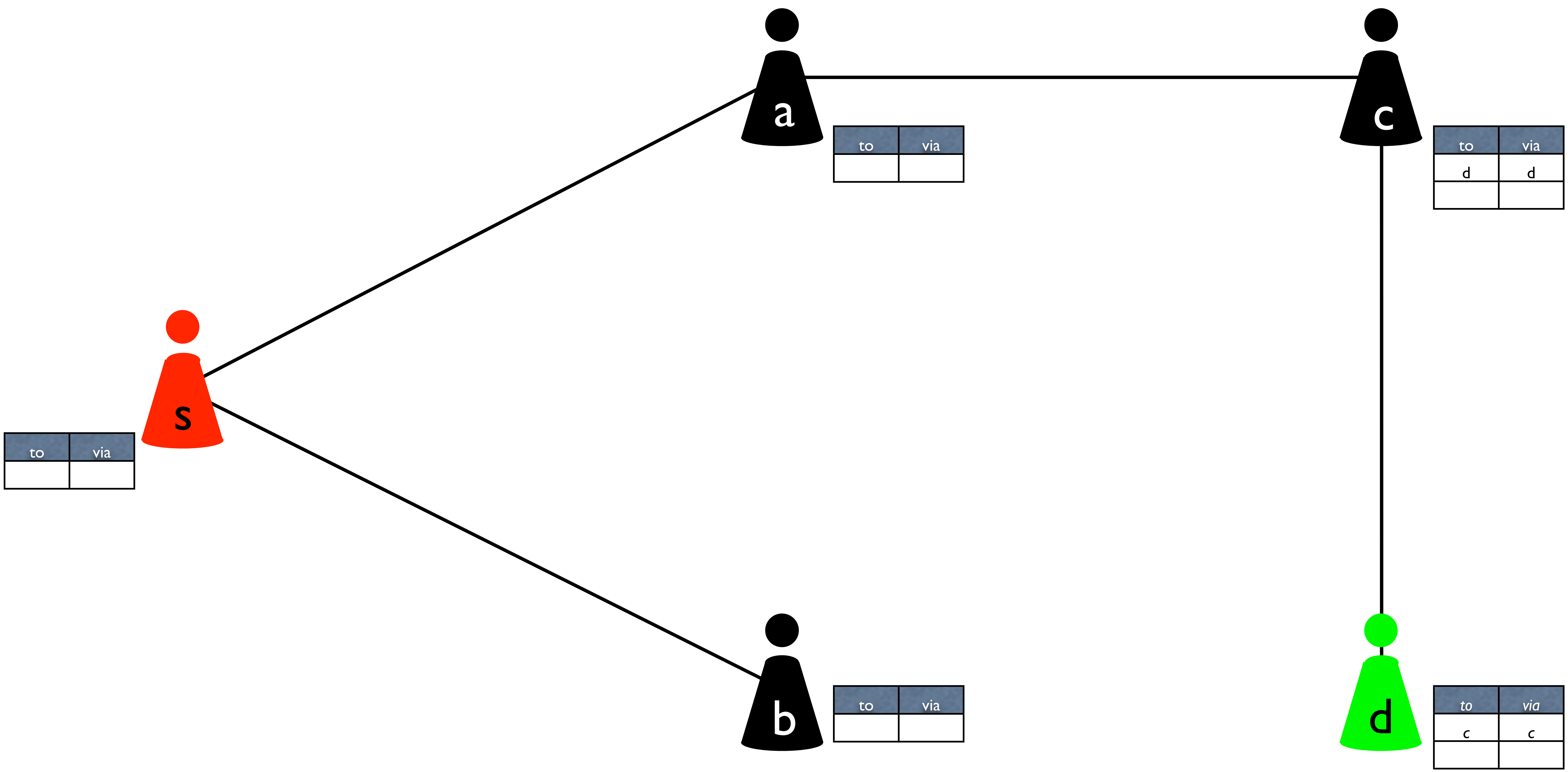
RFC 3561

- Request for Comments (de facto standard)

```
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.
```

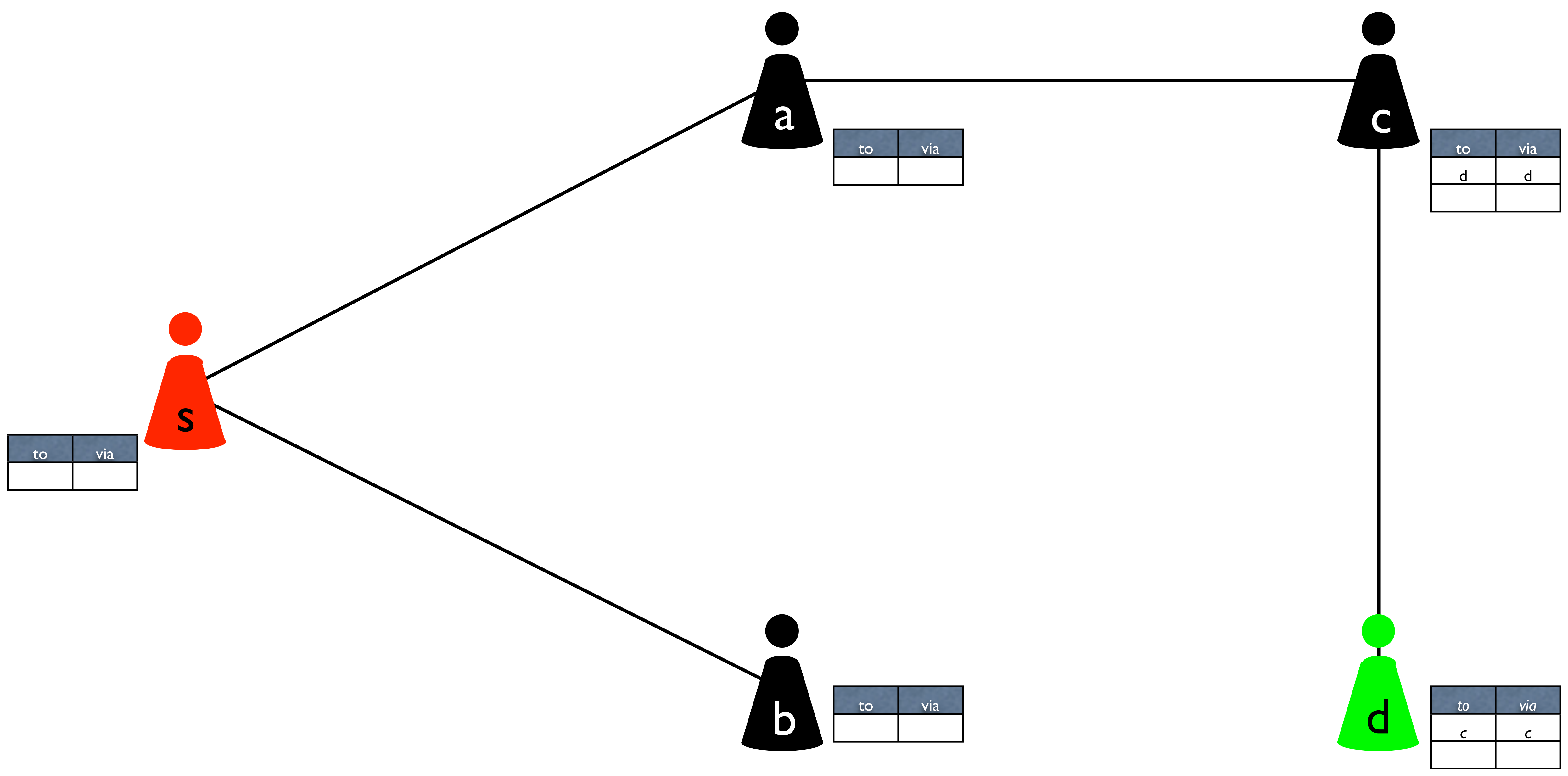
AODV – An Example

NICTA

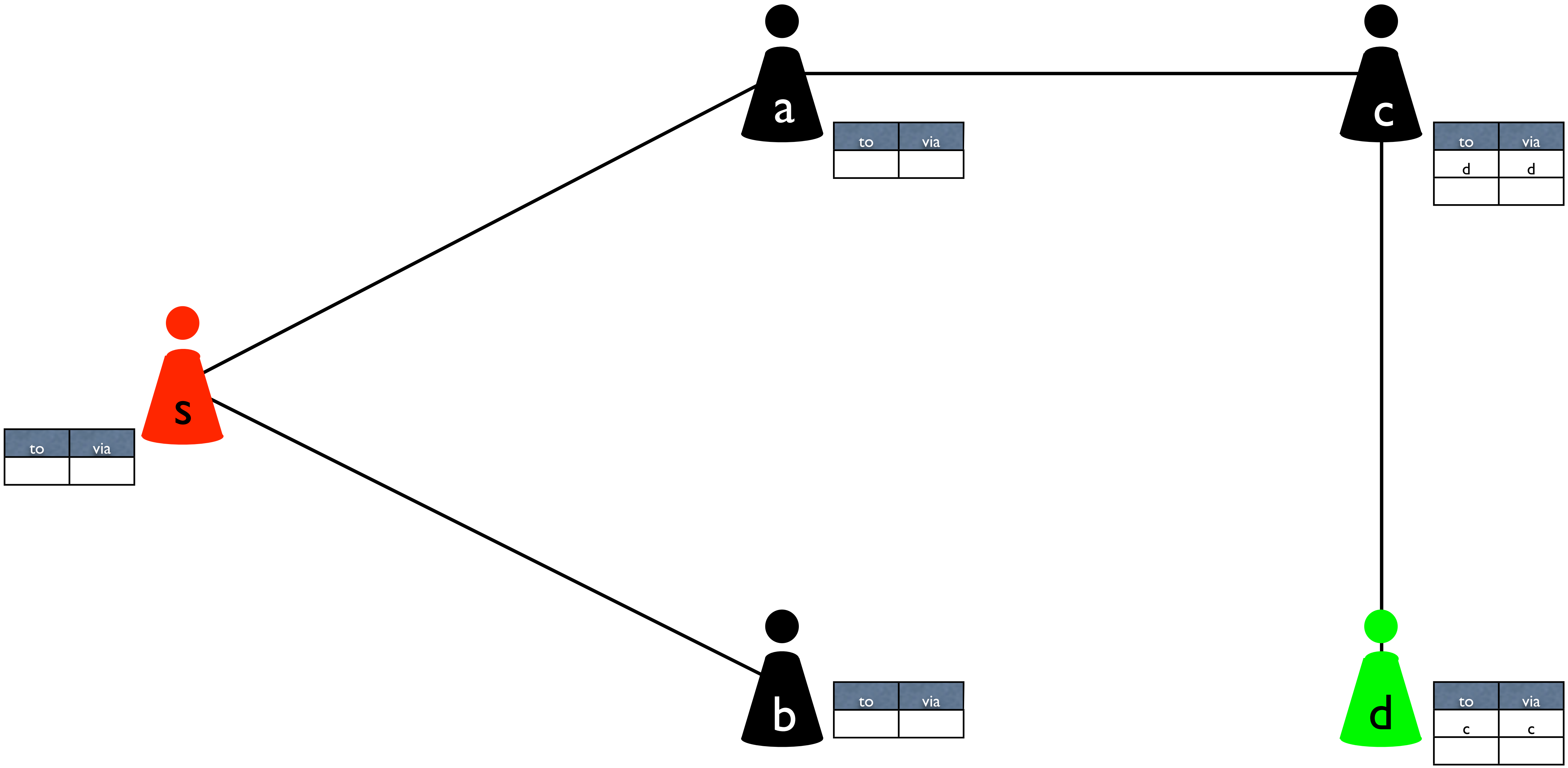


s is looking for a route to d

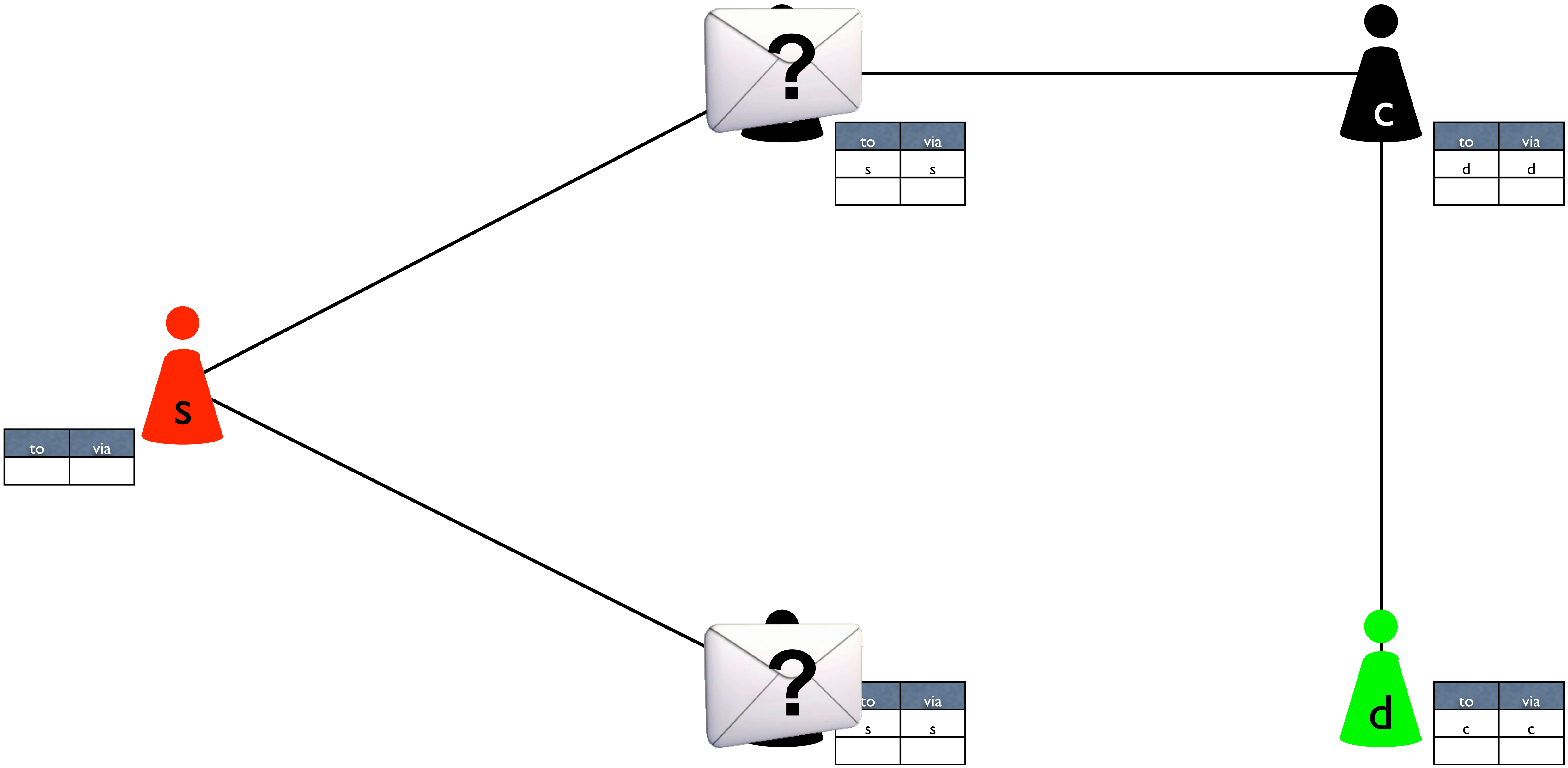
AODV – An Example



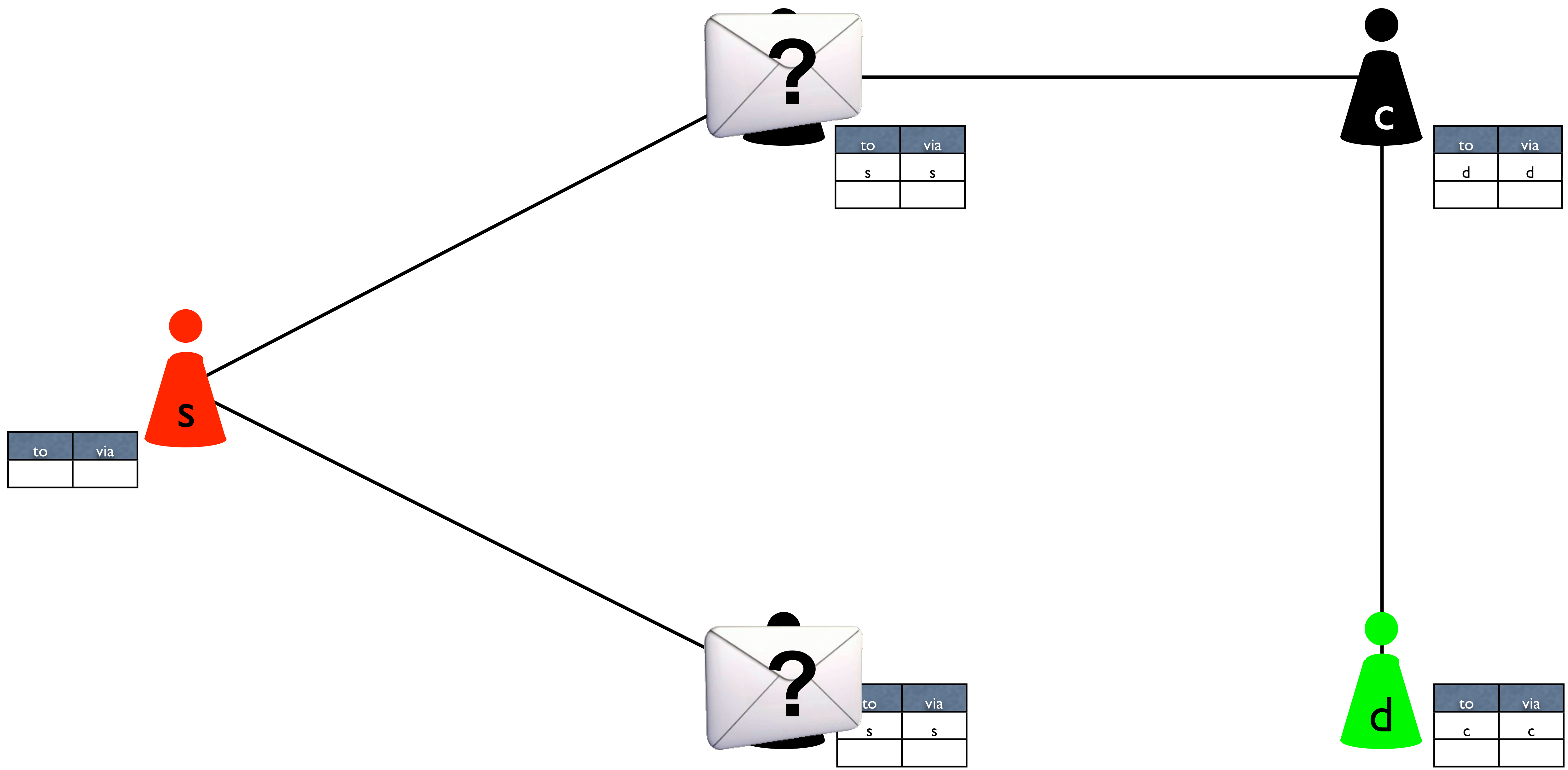
AODV – An Example



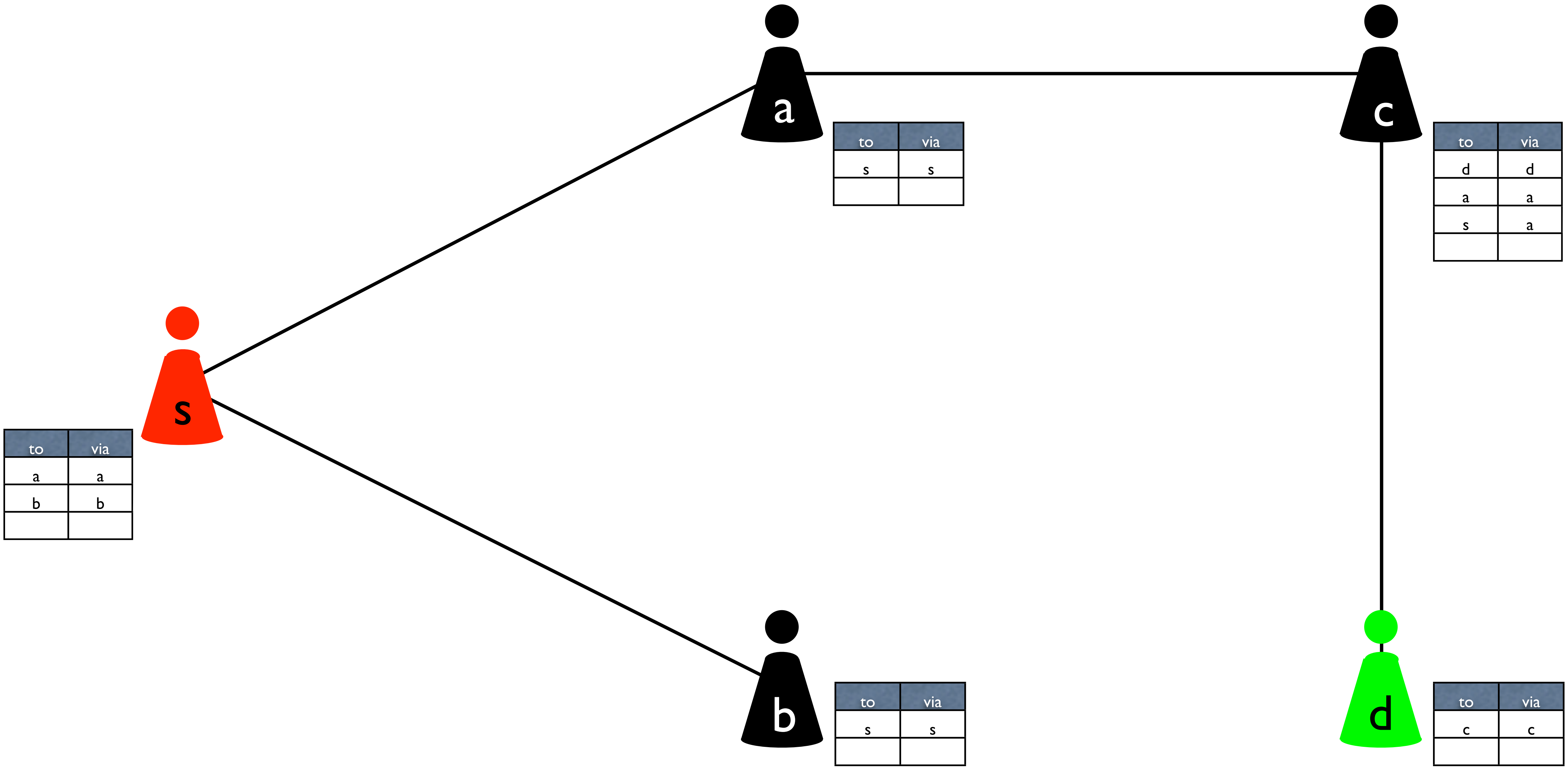
AODV – An Example



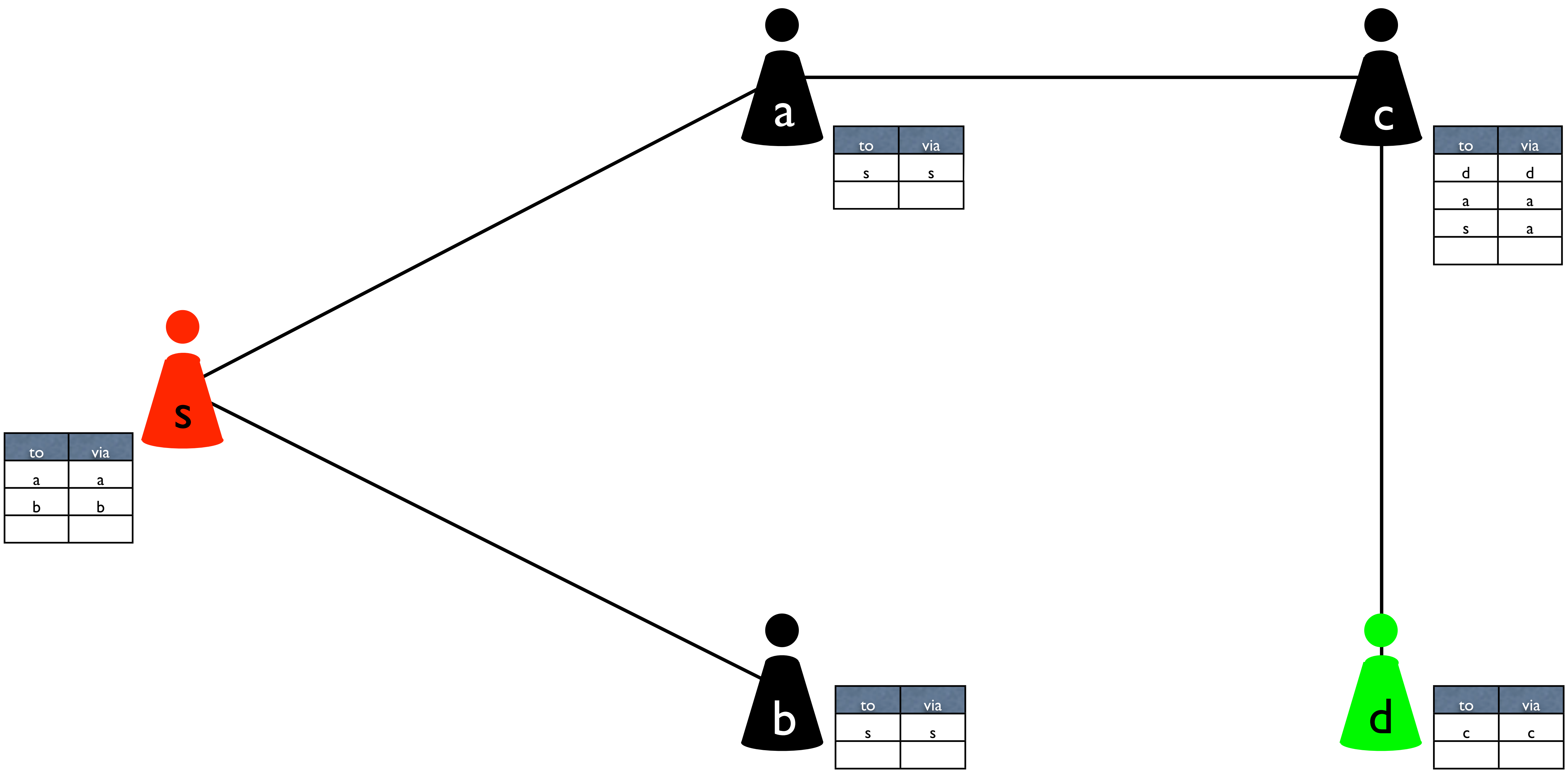
AODV – An Example



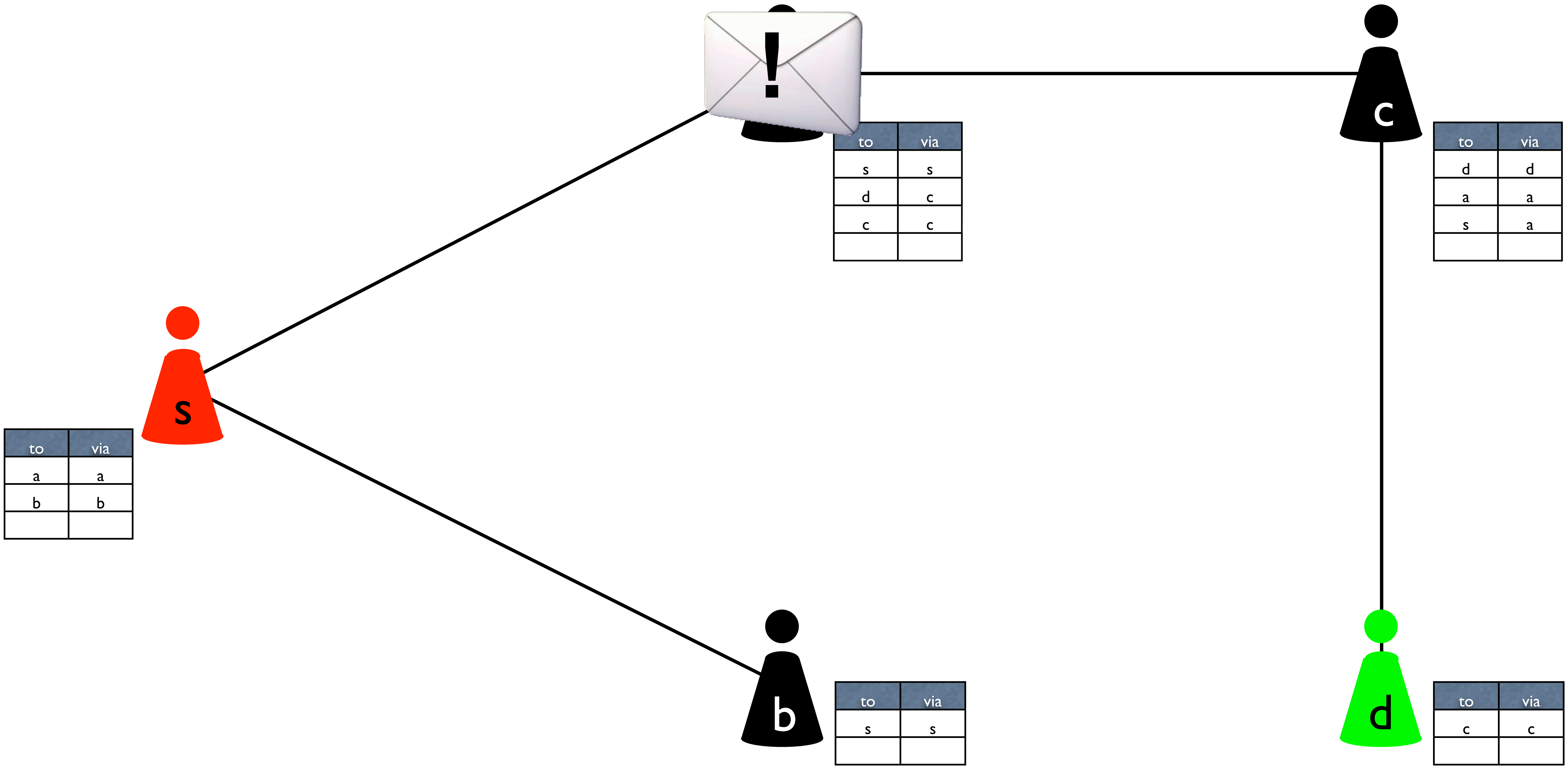
AODV – An Example



AODV – An Example

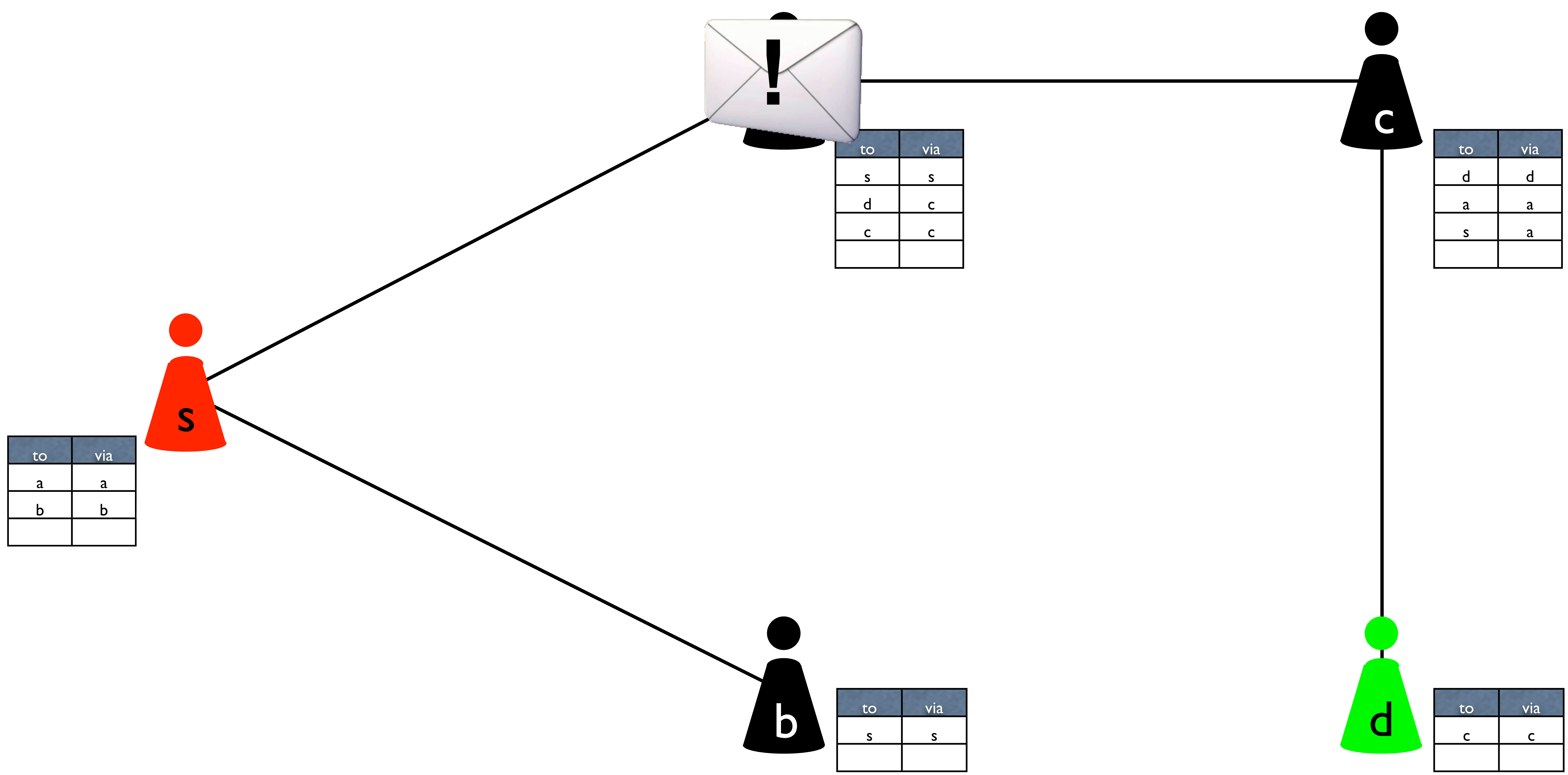


AODV – An Example

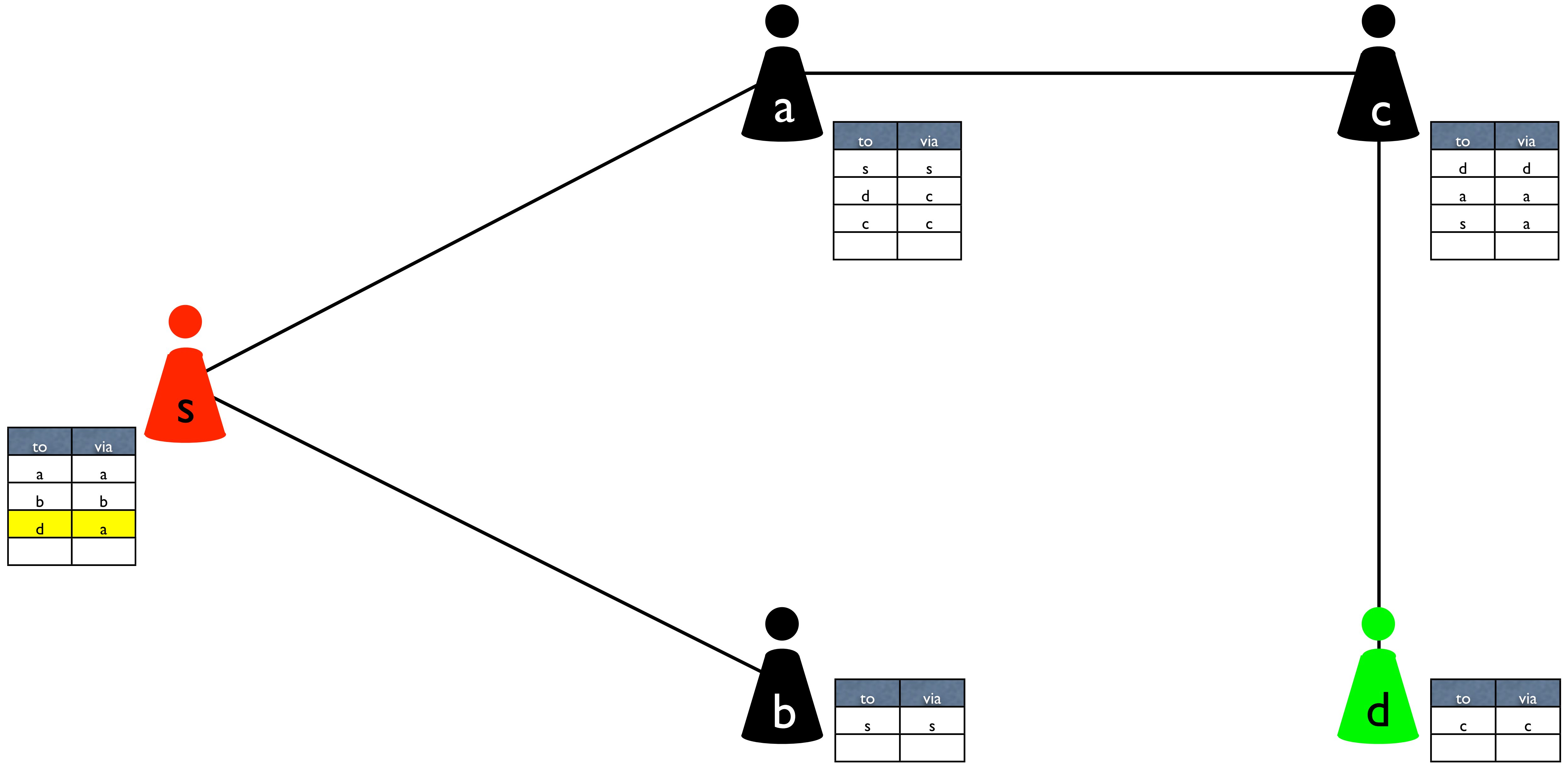


AODV – An Example

NICTA



AODV – An Example



to	via
s	s
d	c
c	c

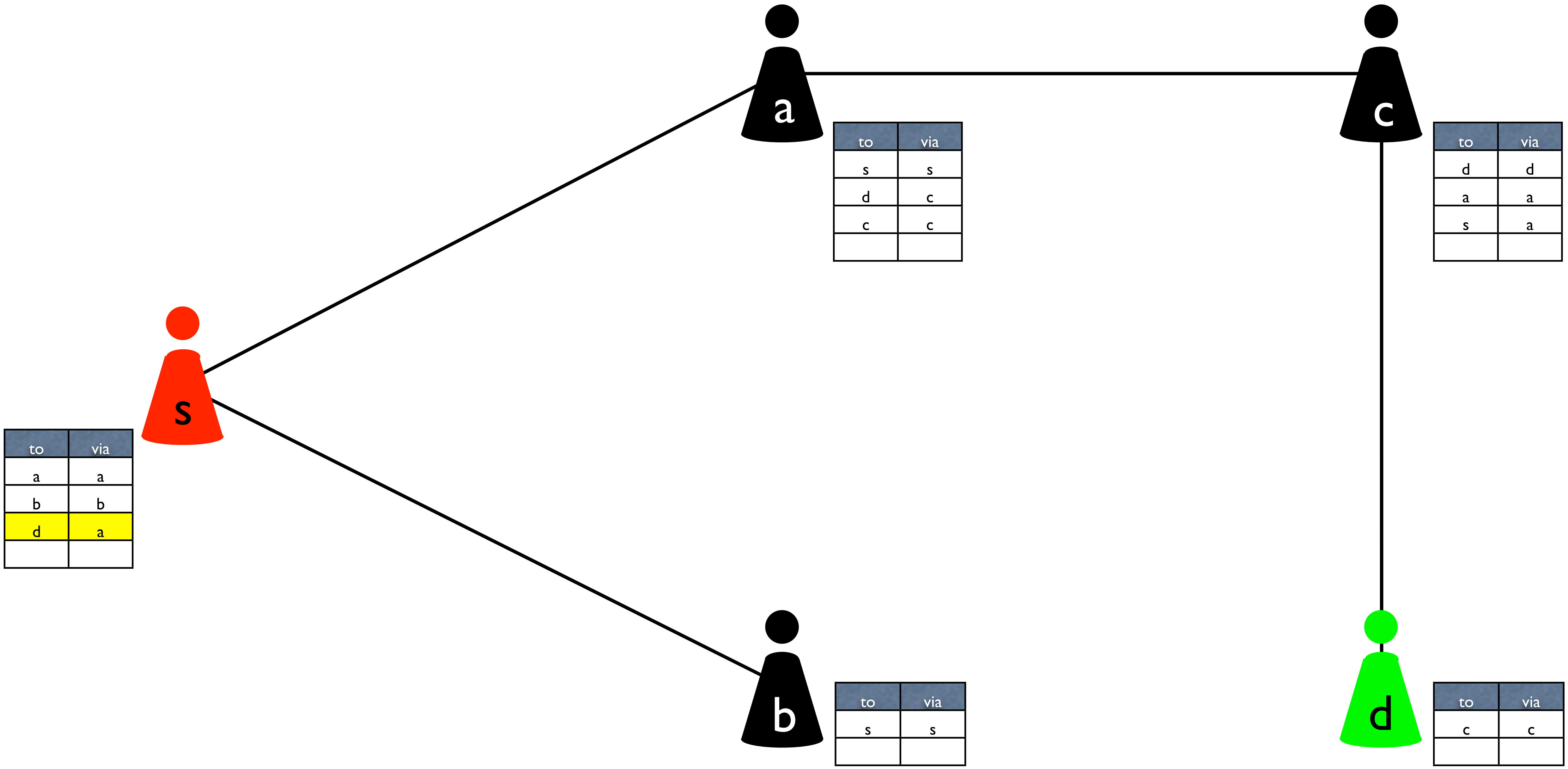
to	via
d	d
a	a
s	a

to	via
a	a
b	b
d	a

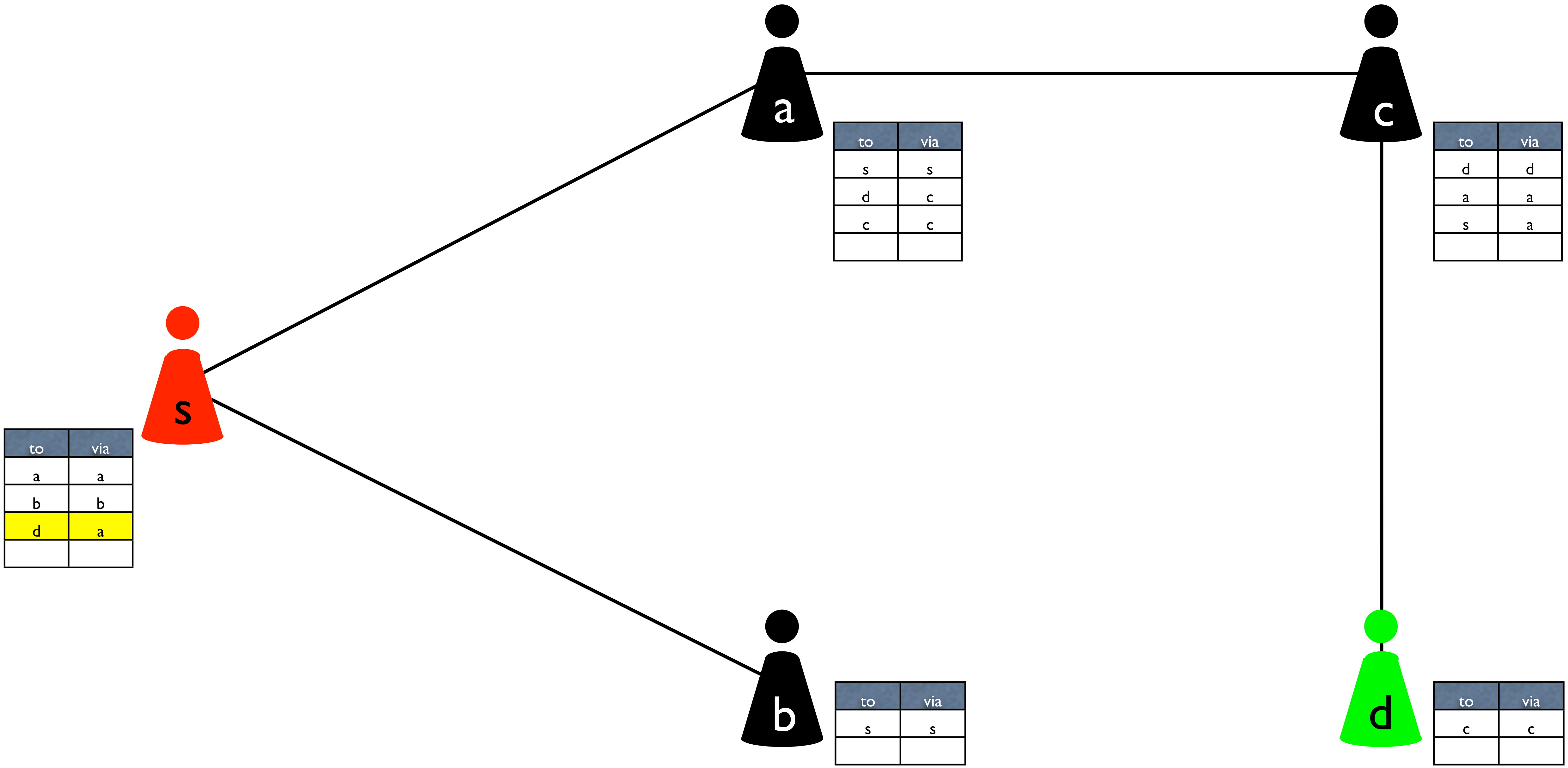
to	via
s	s

to	via
c	c

AODV – An Example



AODV – An Example



s has found a route to d

Ad Hoc On-Demand Distance Vector Protocol

- *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* ✓ (at least for some interpretations)
  - *route found* ✗
  - *packet delivery* ✗

Ad Hoc On-Demand Distance Vector Protocol

- 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

Ad Hoc On-Demand Distance Vector Protocol

- Properties of AODV
  - route correctness ✓
  - loop freedom ✓ (at least for some interpretations)
  - route found ✗
  - packet delivery ✗
- so far only simulation and test-bed evaluations
  - important, valid methods
  - limitations
    - resource intensive, time-consuming, no generality