A Process Algebra for Wireless Mesh Networks

Peter Höfner

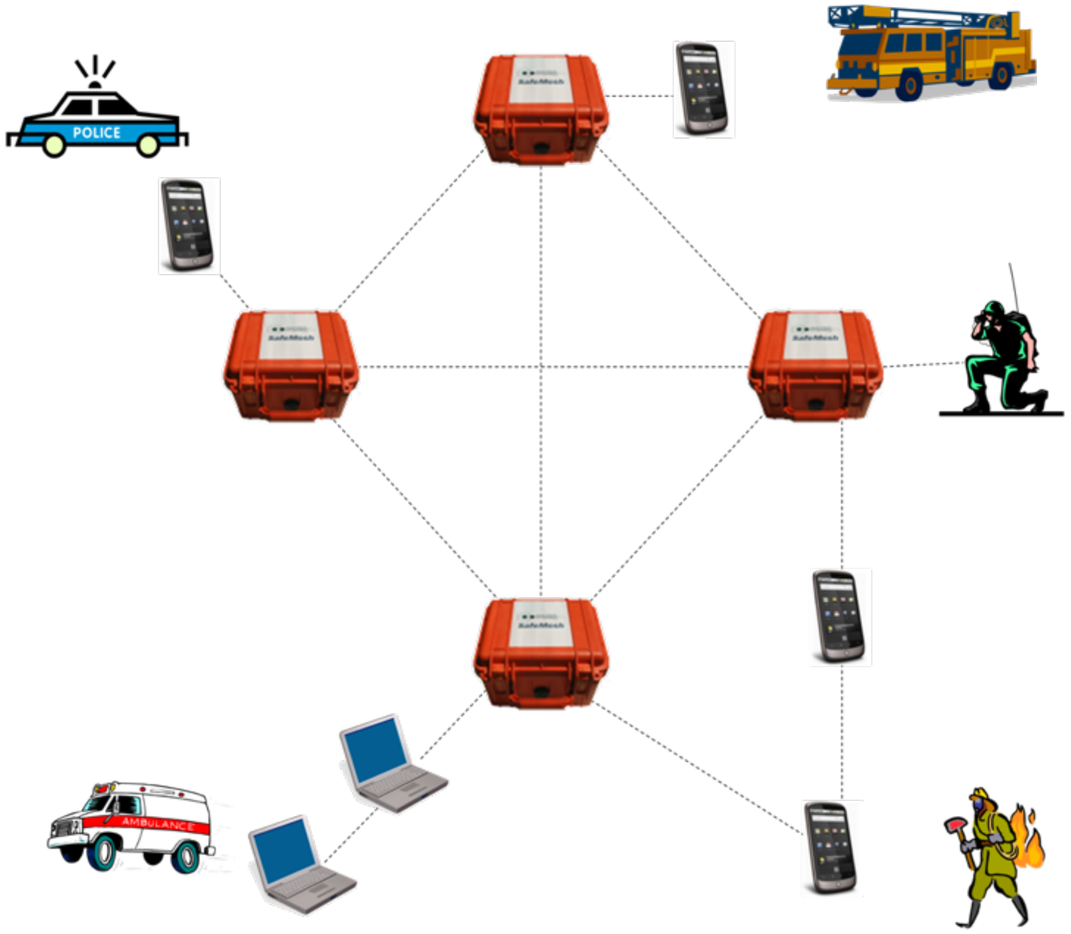
Australian Government
Department of Broadband, Communications and the Digital Economy
Australian Research Council

NICTA Members

NICTA Partners

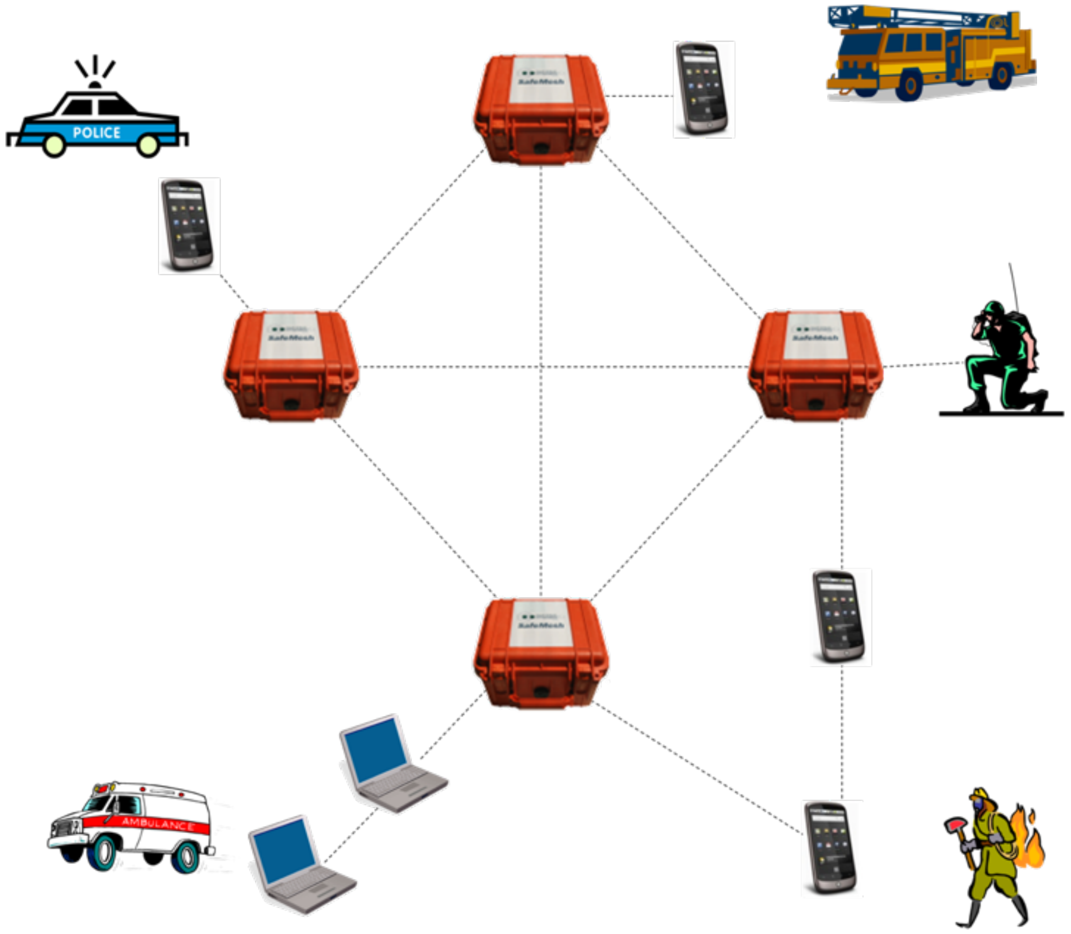
What is the Problem?

- 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
  - ...



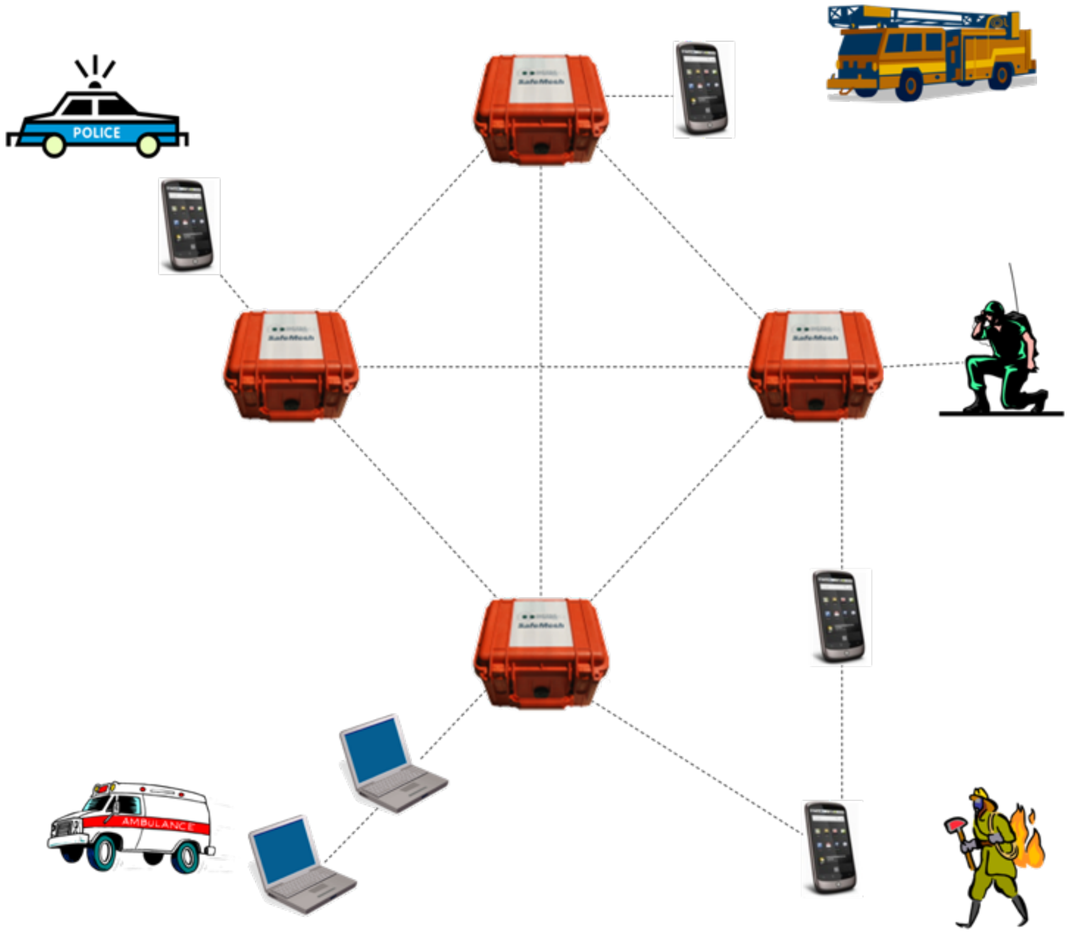
What is the Problem?

- 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

- 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

Process Algebra

NICTA

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

/* update the route to $\mathtt{oip}$ in $\mathtt{rt}$ */

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

/* update $\mathtt{rreqs}$ by adding $(\mathtt{oip}, \mathtt{rreqid})$ */

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

(

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

/* update the sqn of $\mathtt{ip}$ by setting it to $\max(\mathtt{sqn}(\mathtt{rt}, \mathtt{ip}), \mathtt{dsn})$ */

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

/* unicast a RREP towards $\mathtt{oip}$ of the RREQ; next hop is $\mathtt{sip}$ */

$\mathbf{unicast}(\mathtt{sip}, \mathtt{rrep}(0, \mathtt{dip}, \mathtt{sqn}(\mathtt{rt}, \mathtt{ip}), \mathtt{oip}, \mathtt{ip}))\ .\ \mathtt{AODV}(\mathtt{ip}, \mathtt{rt}, \mathtt{rreqs}, \mathtt{queues})$

$\blacktriangleright$ /* If the packet transmission is unsuccessful, a RERR message is generated */

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

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

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

$\mathbf{groupcast}(\mathtt{pre}, \mathtt{rerr}(\mathtt{dests}, \mathtt{ip}))\ .\ \mathtt{AODV}(\mathtt{ip}, \mathtt{rt}, \mathtt{rreqs}, \mathtt{queues})$

$+ [\ \mathtt{dip} \neq \mathtt{ip}\ ]$ /* 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 $\mathtt{dip}$ that is fresh enough */

/* update $\mathtt{rt}$ by adding $\mathtt{sip}$ to $\mathtt{precs}(\mathtt{rt}, \mathtt{dip})$ */

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

Process Algebra

- Desired Properties
  - guaranteed broadcast
  - prioritised unicast
  - data structure

- Inspired by
  - π - Calculus
  - ω - Calculus
  - (LOTOS)

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

$$
\begin{array}{rcl}
SP & ::= & X(\mathit{exp}_1, \ldots, \mathit{exp}_n) \mid [\varphi]SP \mid [\![\mathtt{var} := \mathit{exp}]\!]SP \mid SP + SP \mid \\
& & \alpha.SP \mid \textbf{unicast}(\mathit{dest}, \mathit{ms}).SP \blacktriangleright SP \\
\alpha & ::= & \textbf{broadcast}(\mathit{ms}) \mid \textbf{groupcast}(\mathit{dests}, \mathit{ms}) \mid \textbf{send}(\mathit{ms}) \mid \\
& & \textbf{deliver}(\mathit{data}) \mid \textbf{receive}(\mathtt{msg})
\end{array}
$$

Structual Operational Semantics I

- internal state determined by expression and valuation

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

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

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

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

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

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

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

Structural Operational Semantics II

- internal state determined by expression and valuation

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

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

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

Nodes (Parallel Processes)

- Syntax

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

- 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)

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

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

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

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

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

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

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

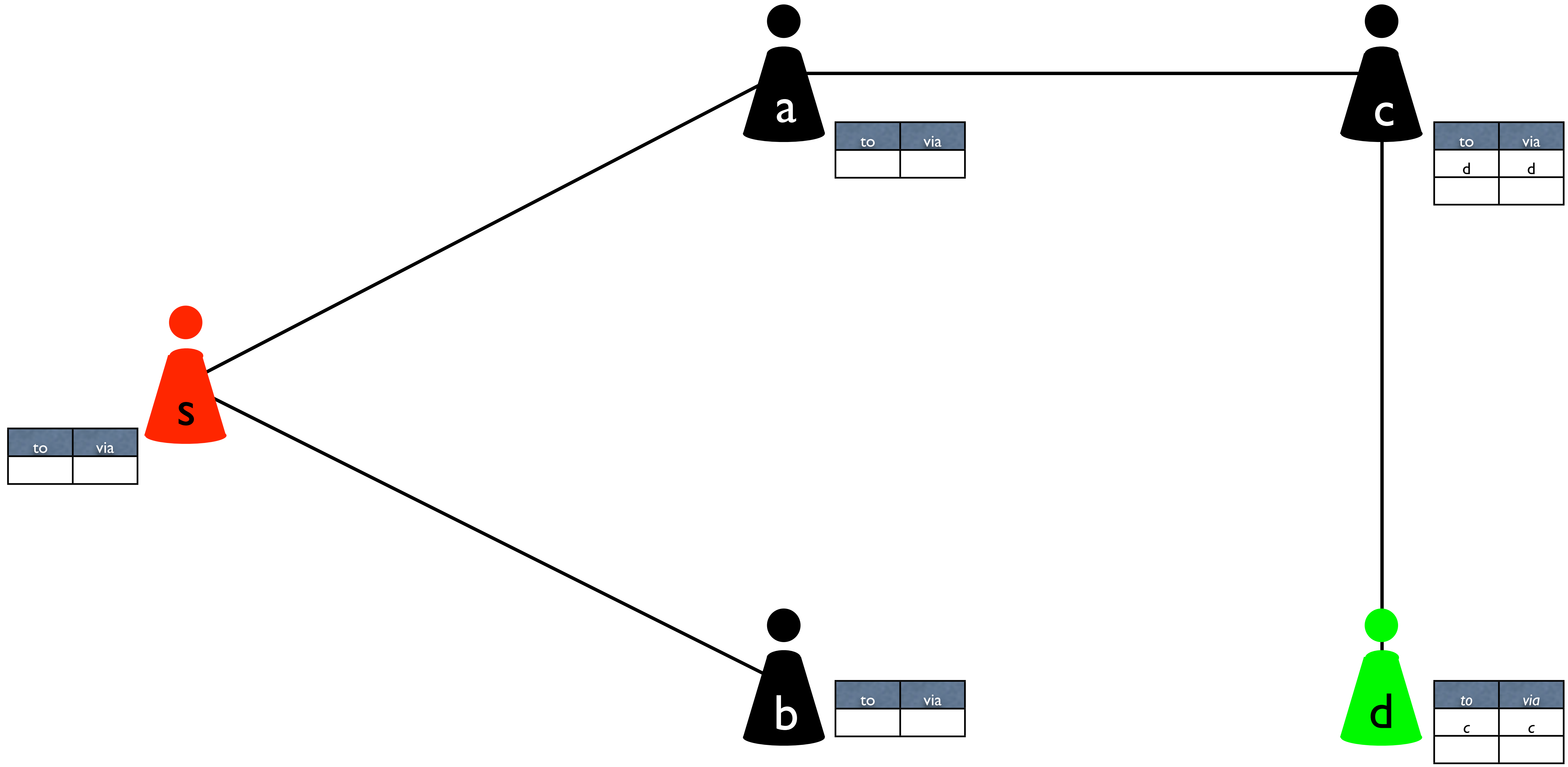
A Bit of Theoretical Results

- 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

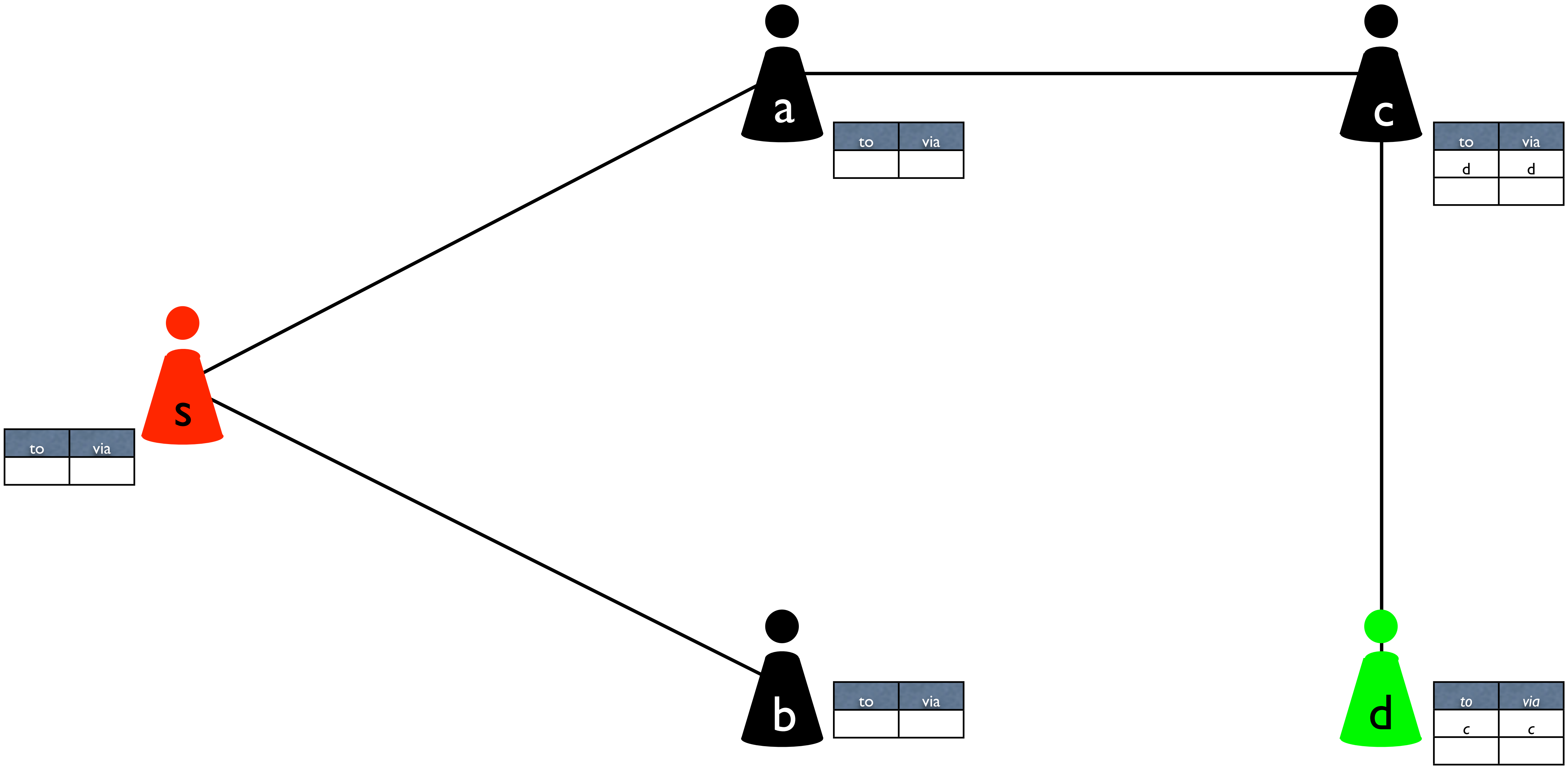
- AODV: Ad-hoc On-Demand Distance Vector Routing Protocol
  - Ad hoc (network is not static)
  - On-Demand (routes are established when needed)
  - Distance (metric is hop count)
  - Vector (routing table has the form of a vector)
  - Developed 1997-2001 by Perkins, Beldig-Royer and Das (University of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

AODV – An Example

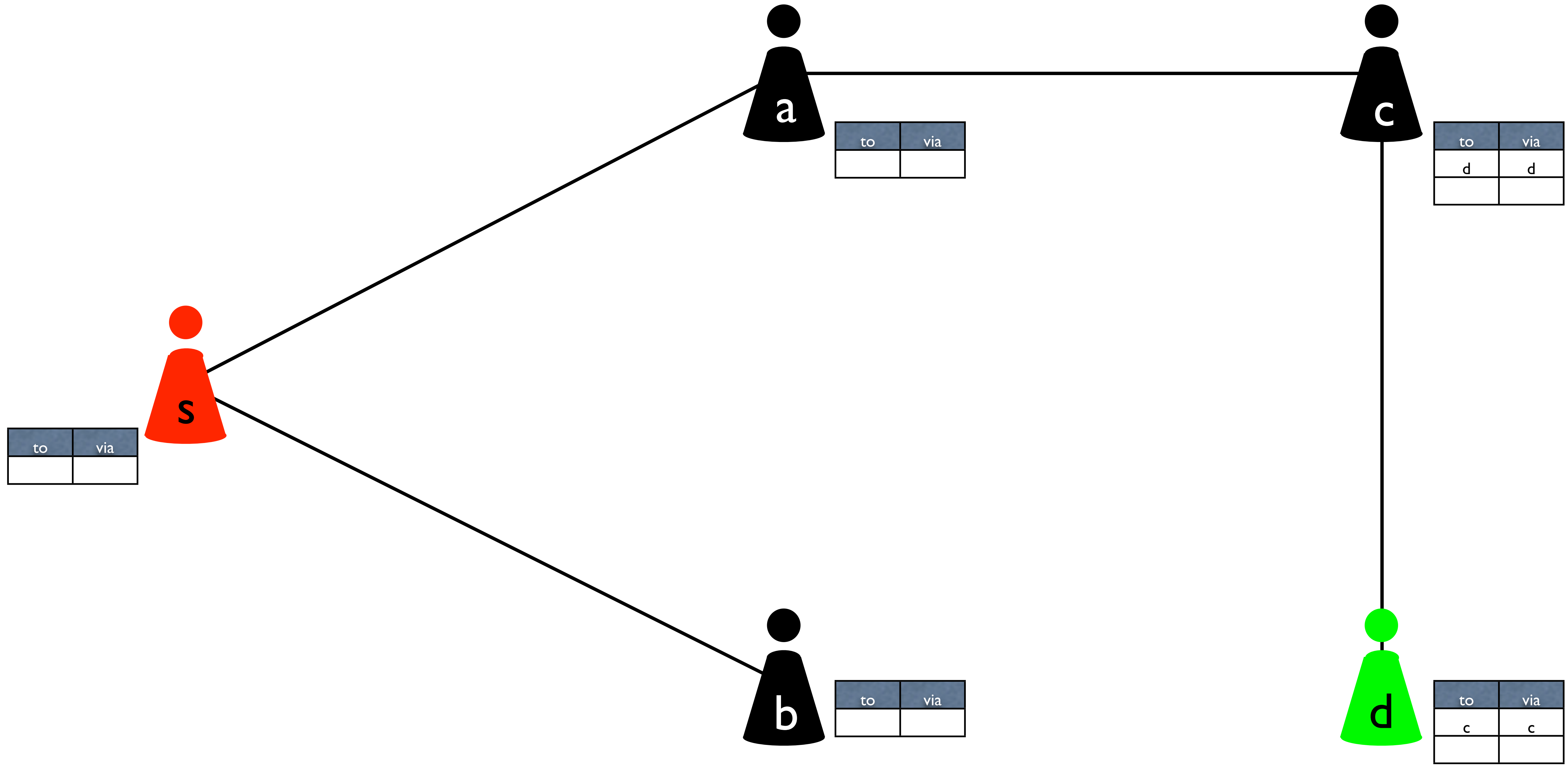


s is looking for a route to d

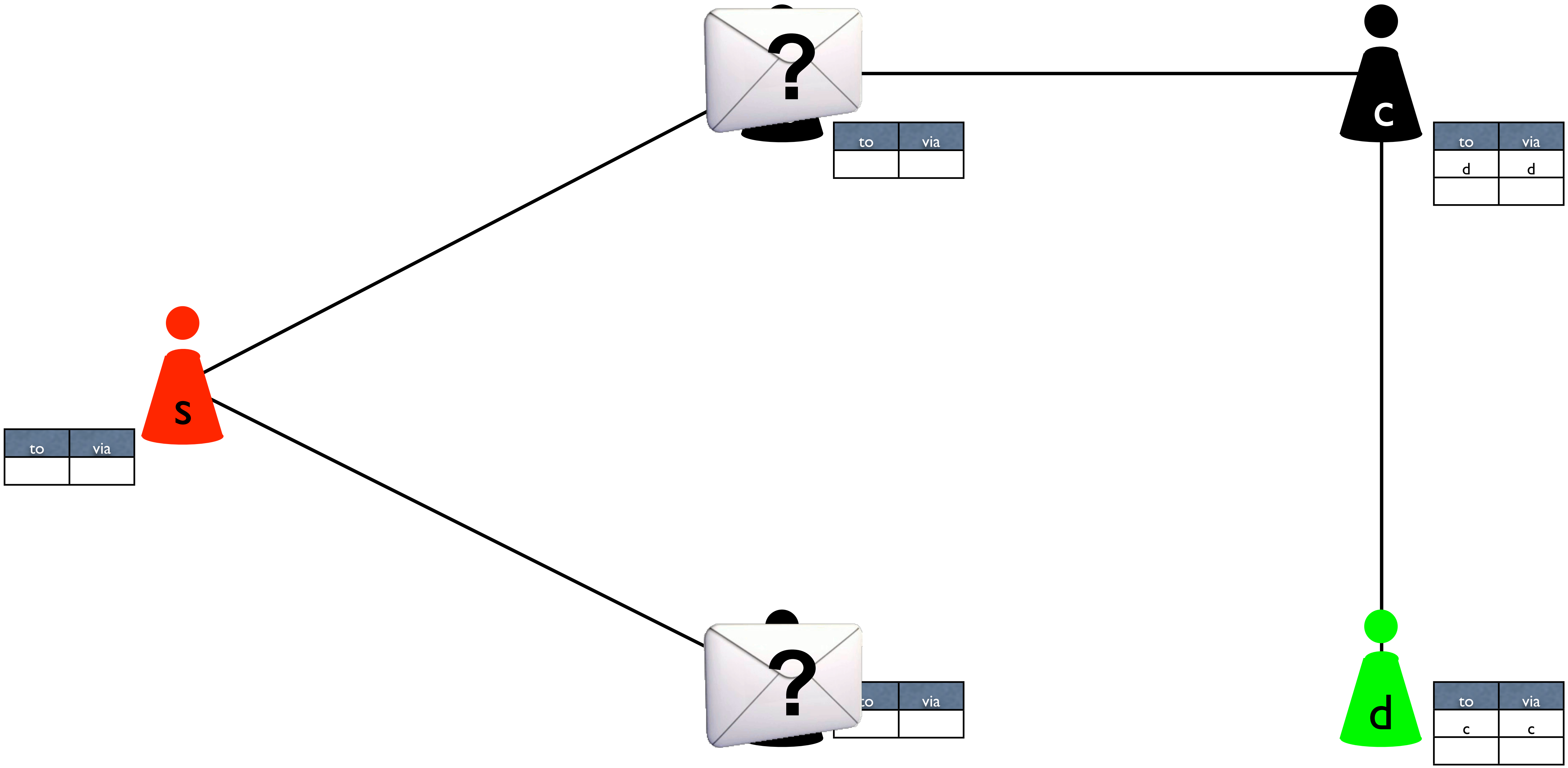
AODV – An Example



AODV – An Example

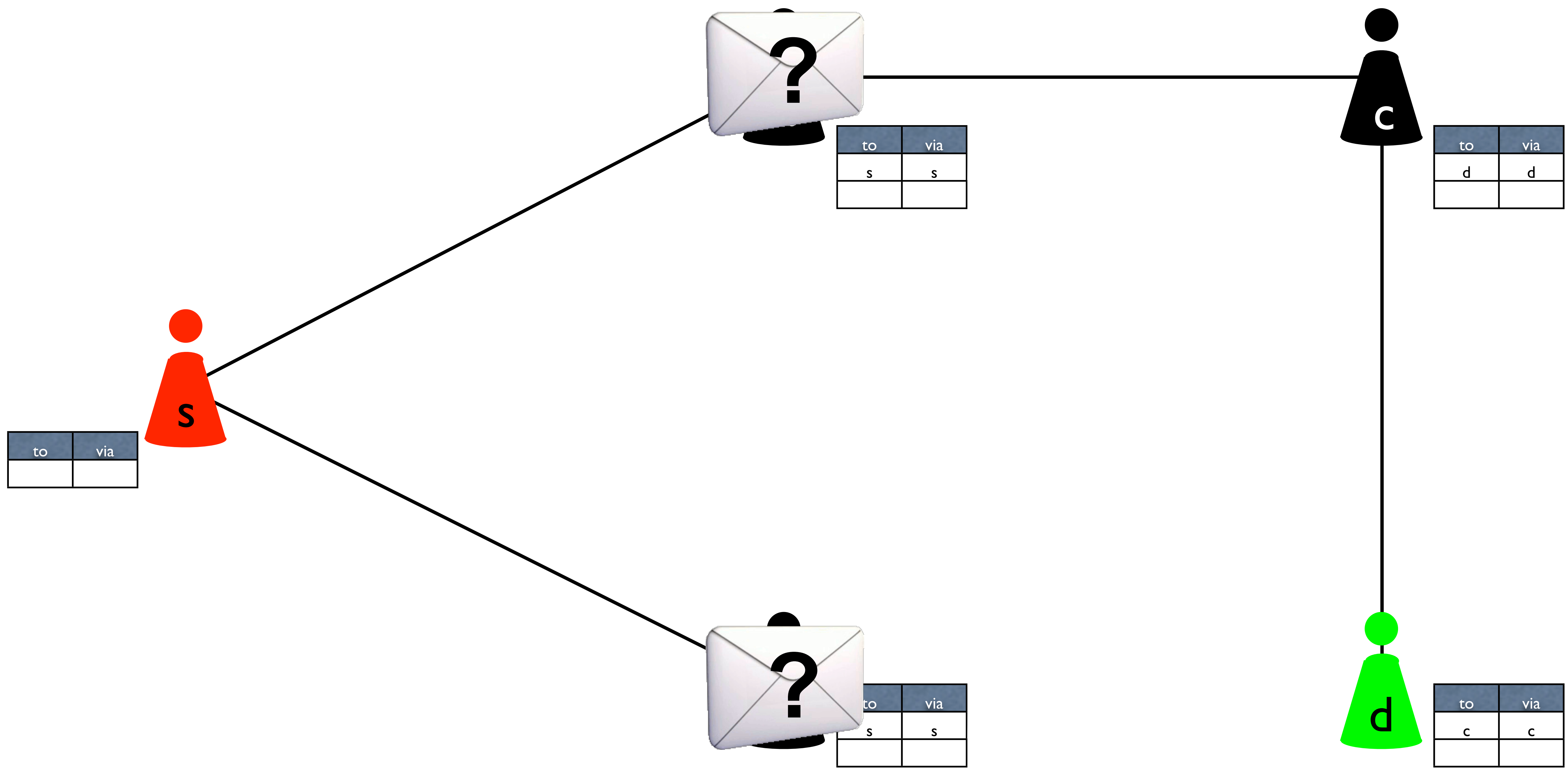


AODV – An Example



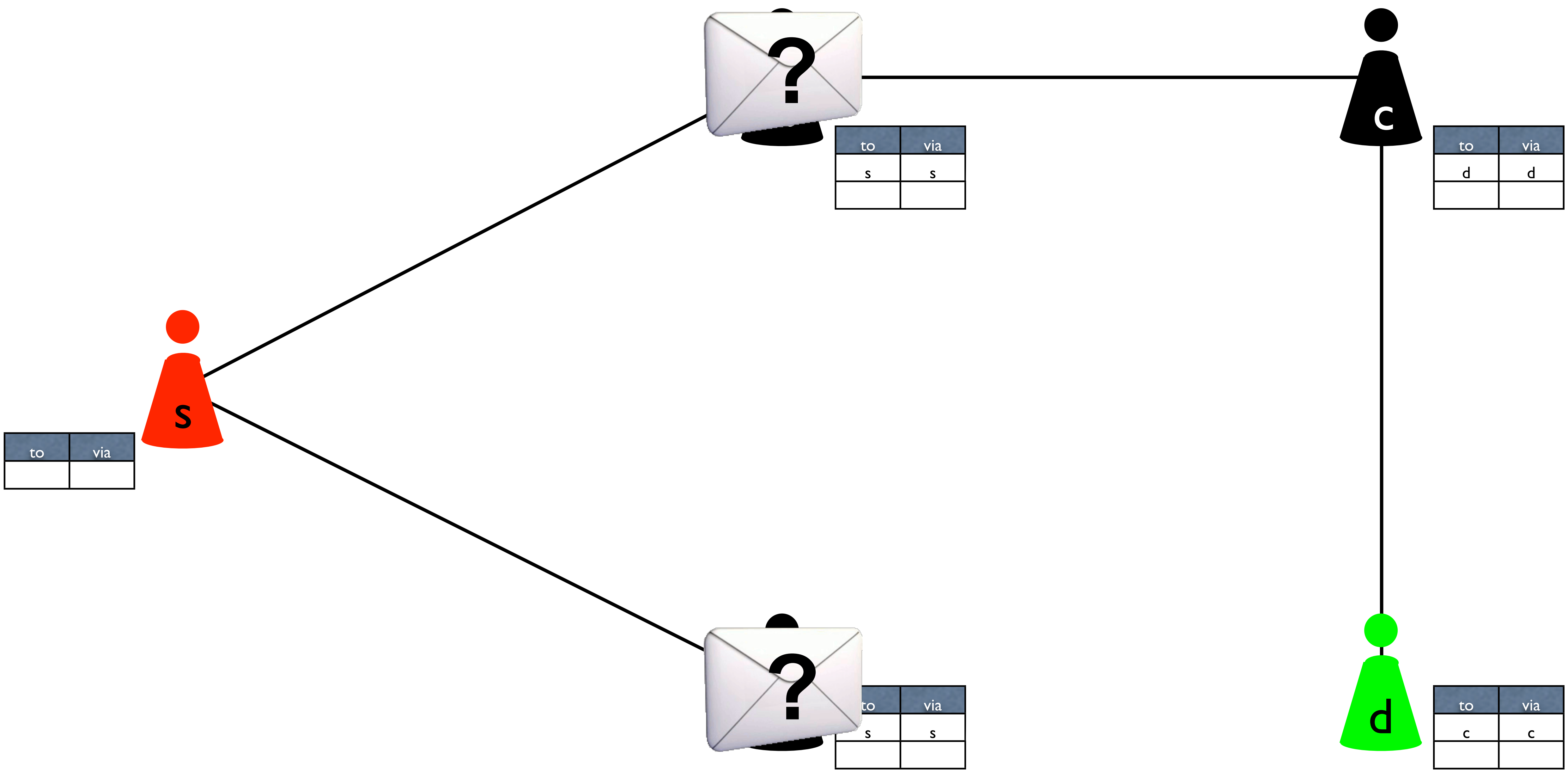
s broadcasts a route request

AODV – An Example

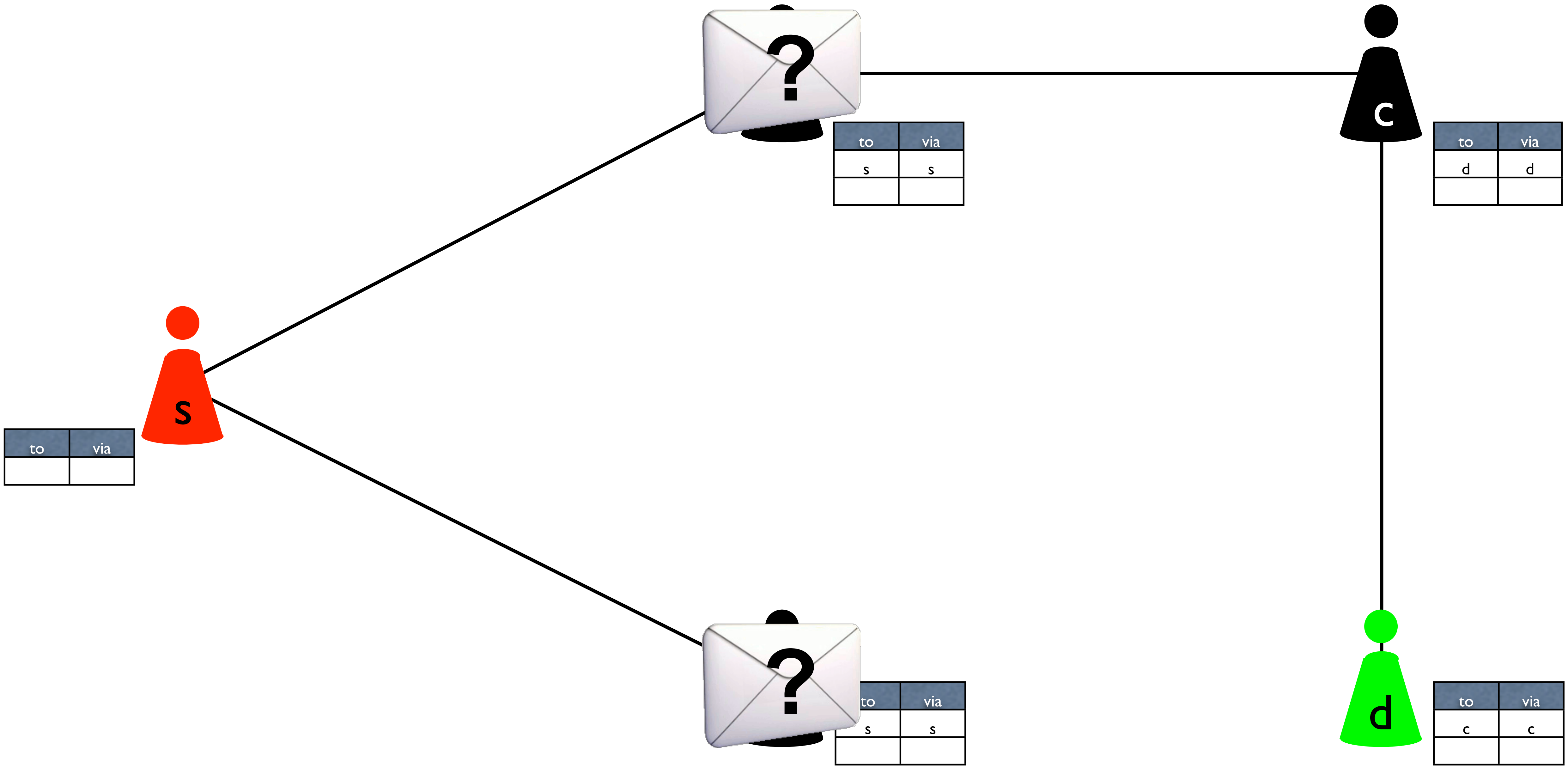


s broadcasts a route request

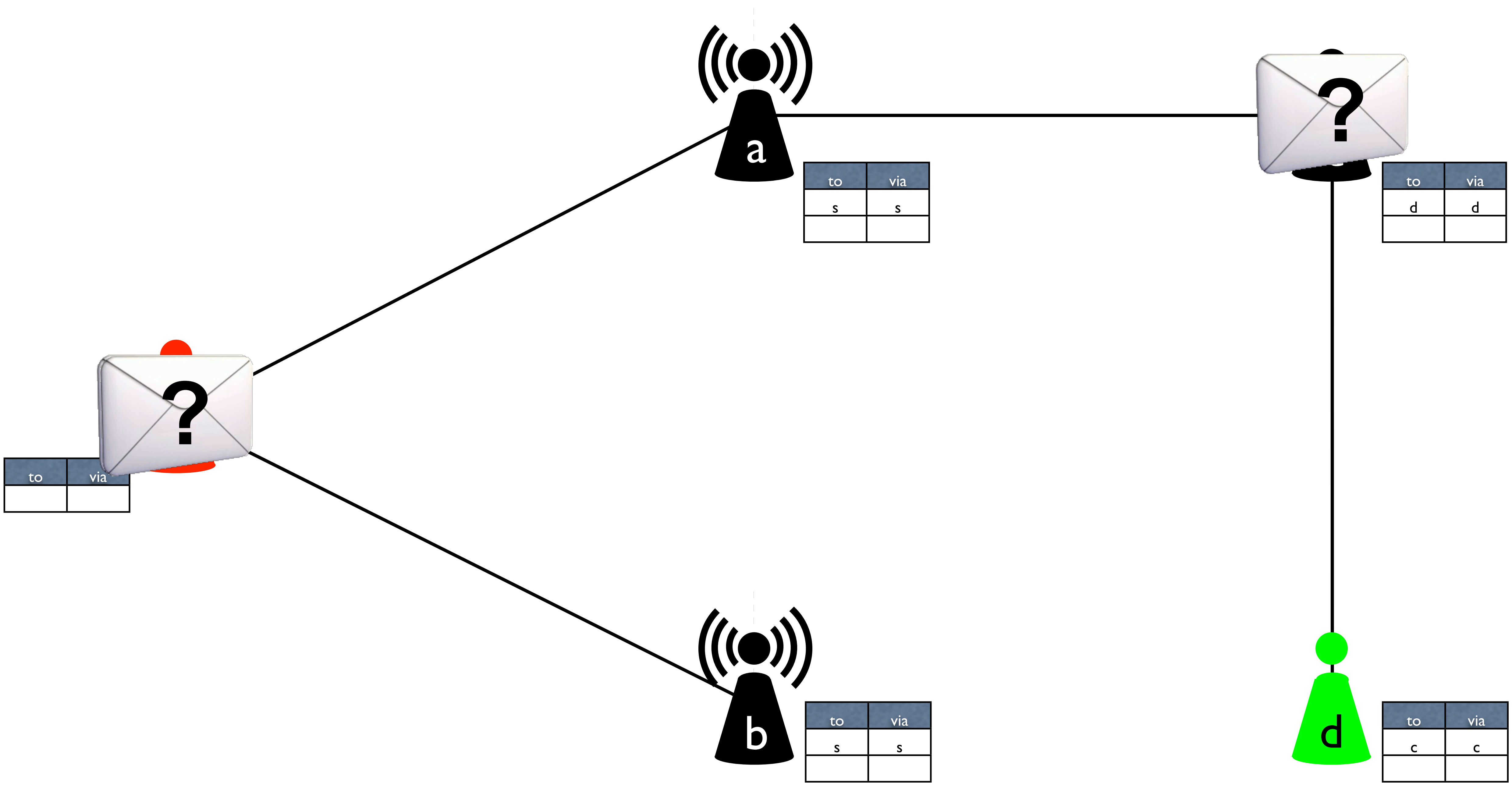
AODV – An Example



AODV – An Example

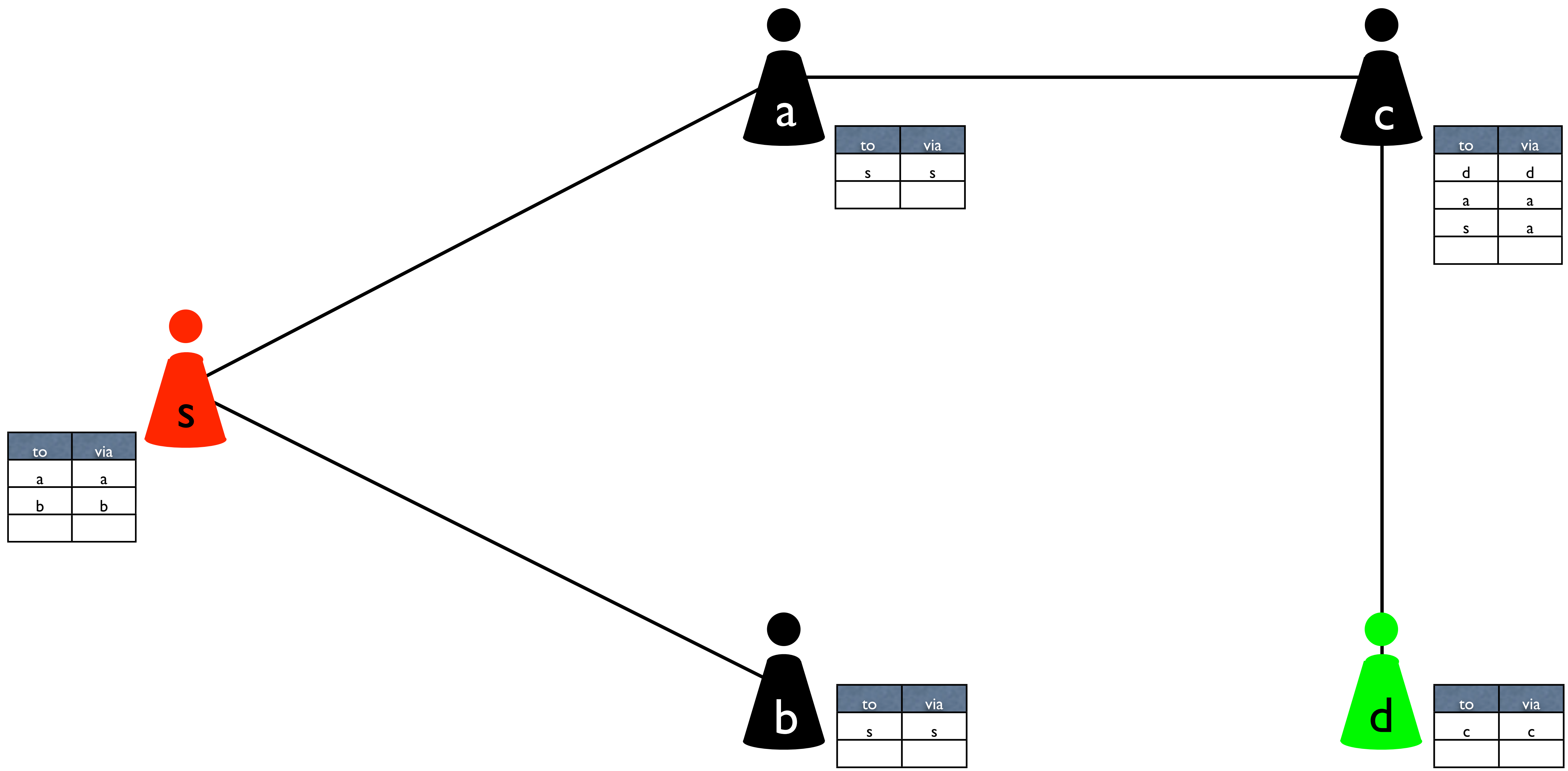


AODV – An Example



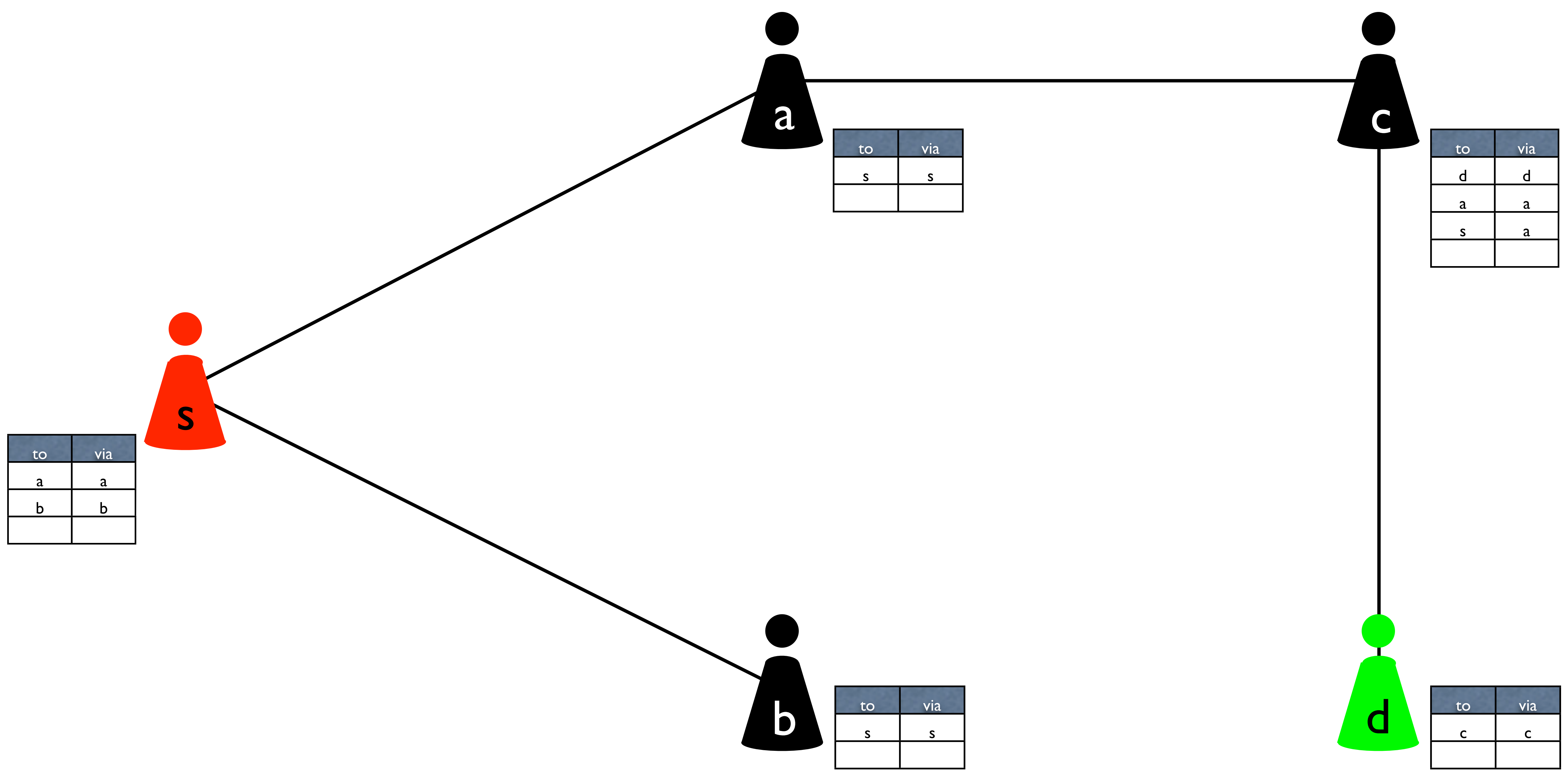
a,b forward the route request

AODV – An Example

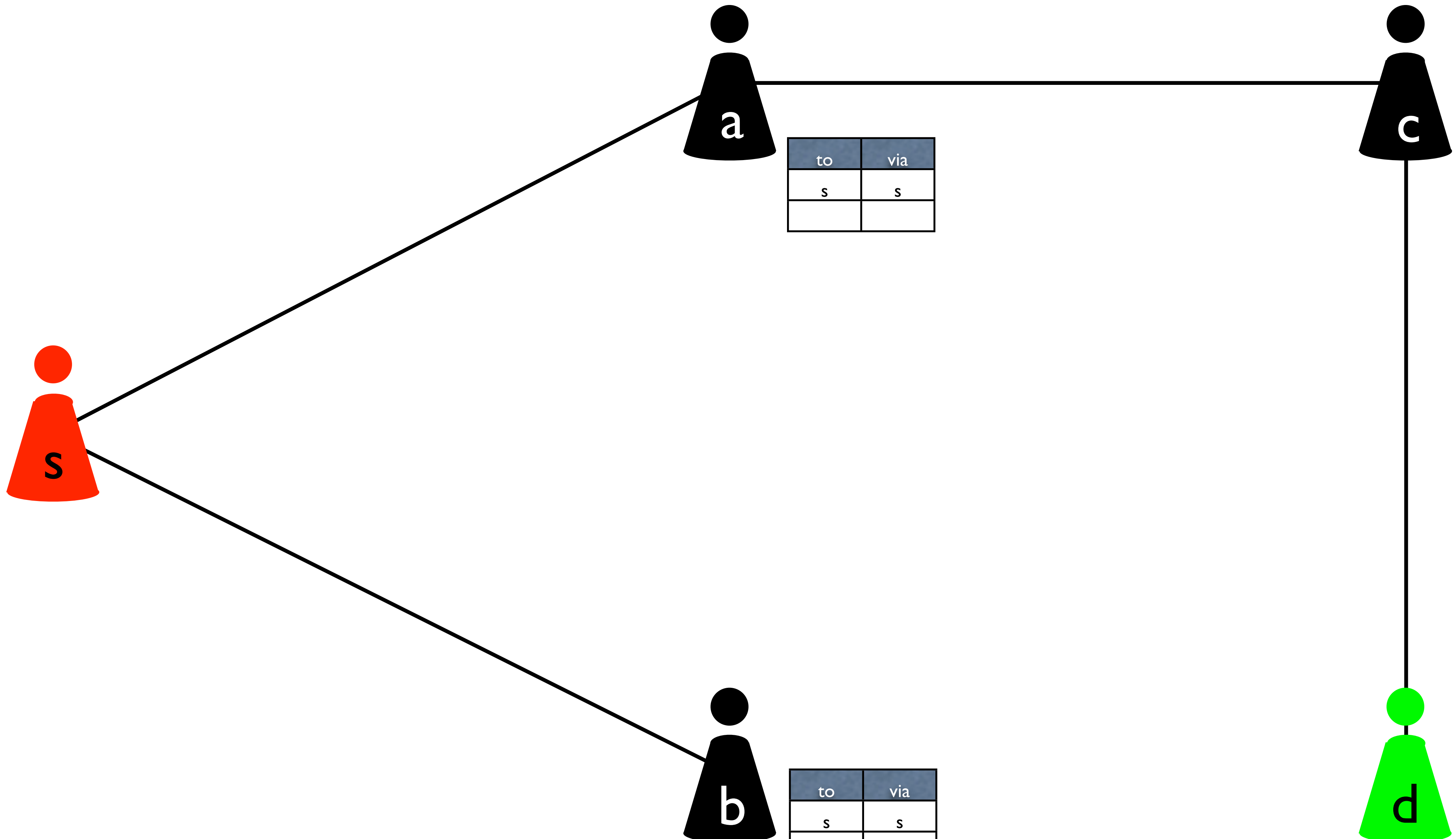


a,b forward the route request

AODV – An Example



AODV – An Example



to	via
s	s

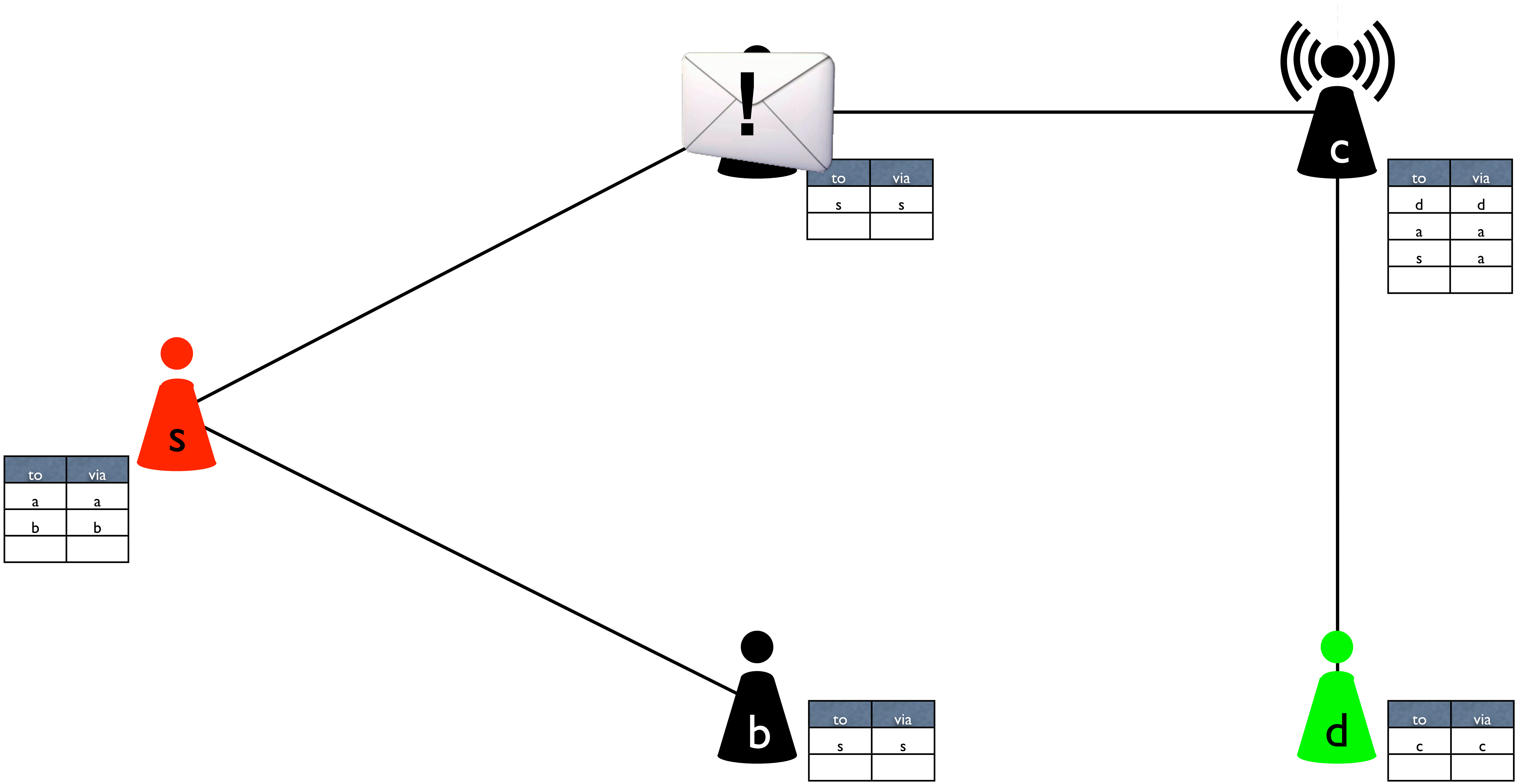
to	via
d	d
a	a
s	a

to	via
a	a
b	b

to	via
s	s

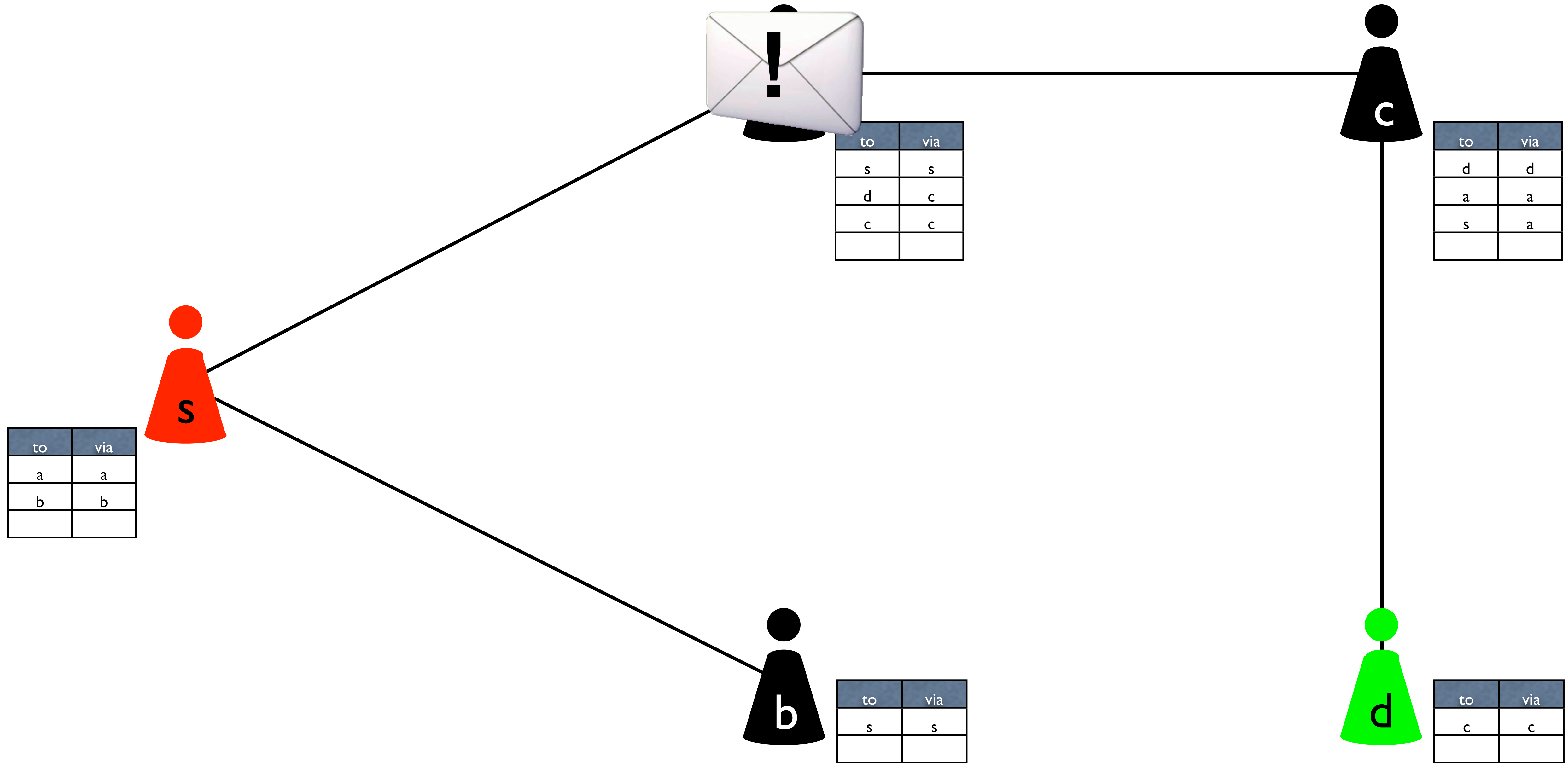
to	via
c	c

AODV – An Example



c has information about d
c answers route request and sends reply

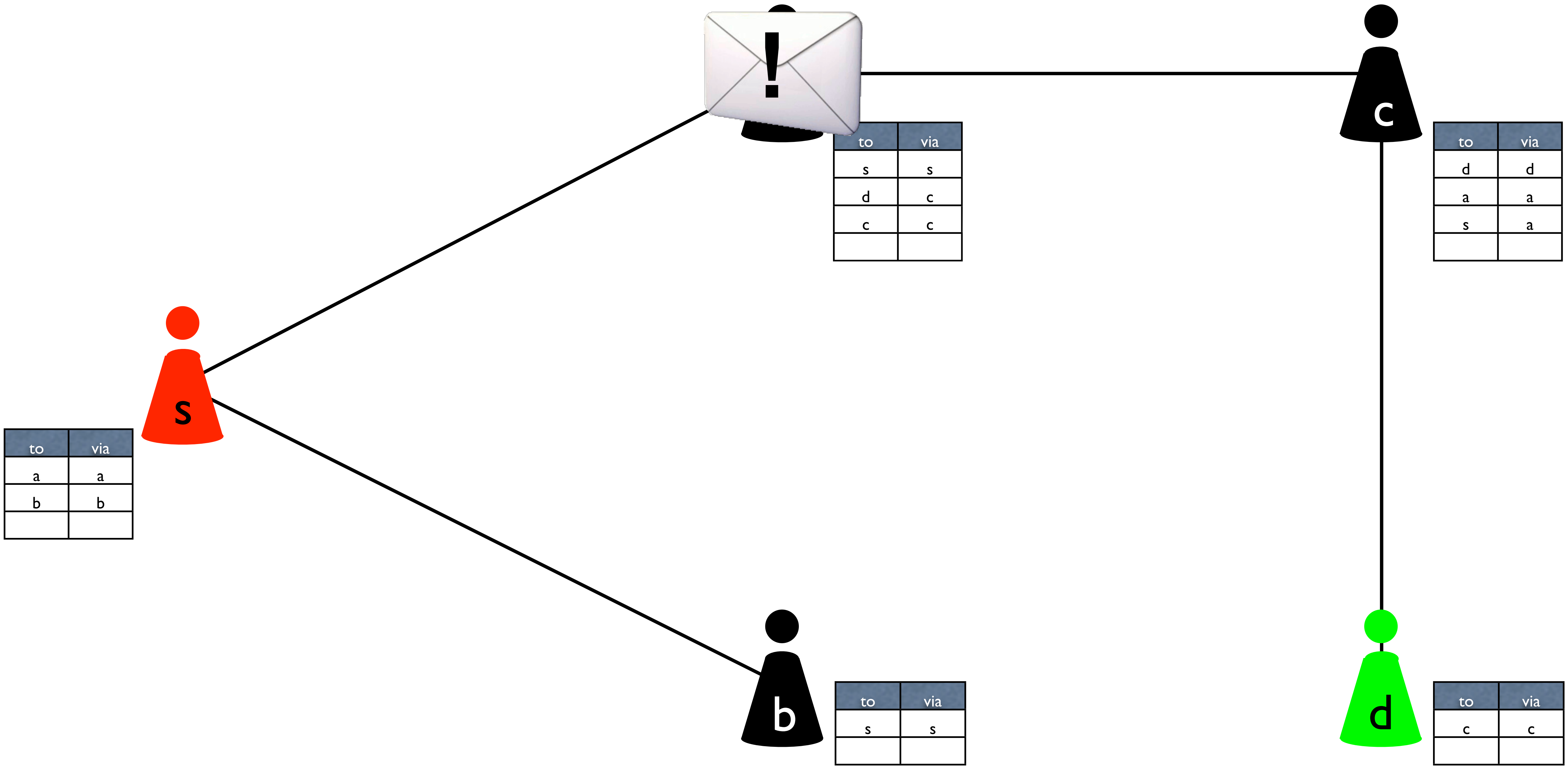
AODV – An Example



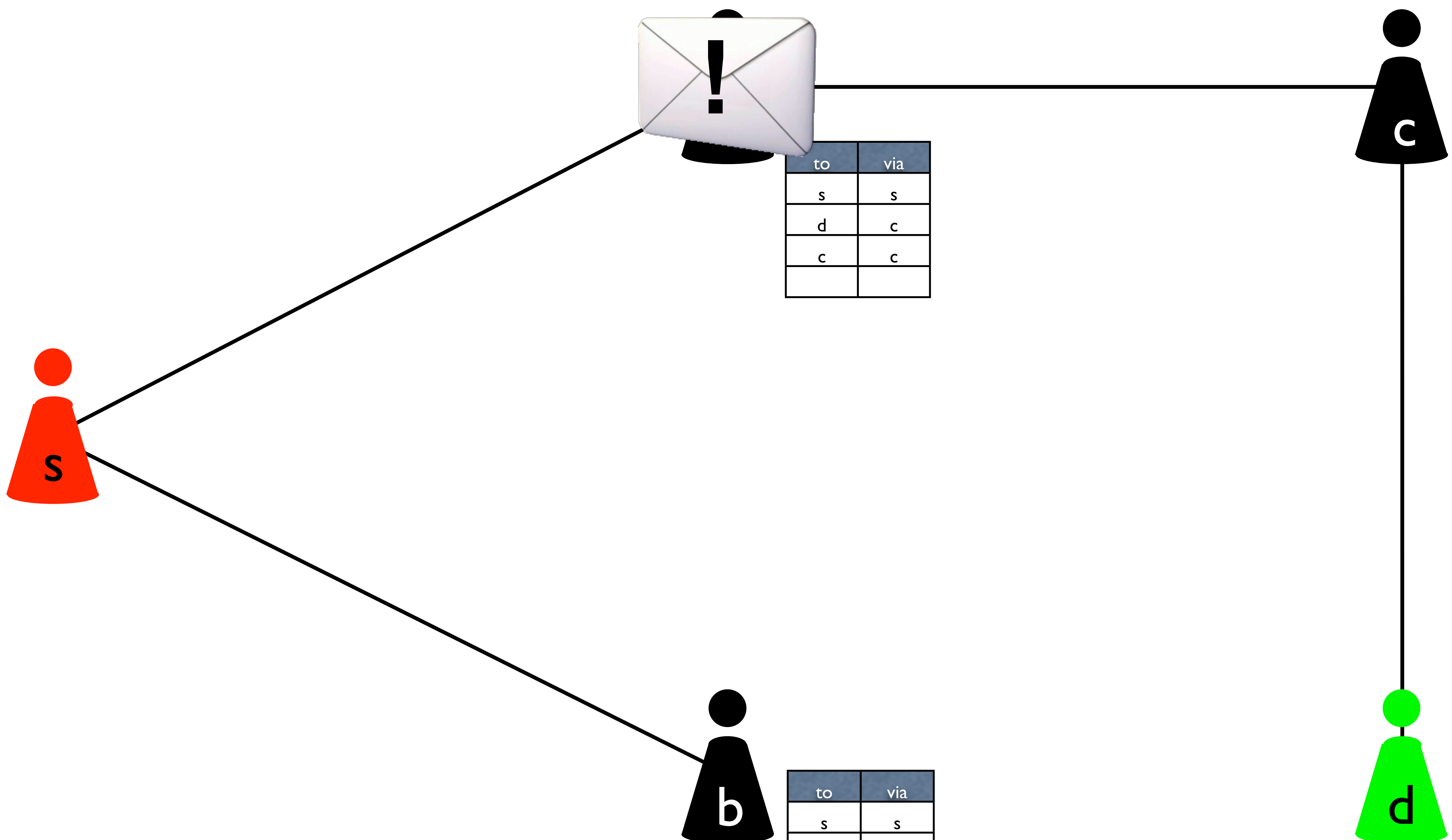
c has information about d

c answers route request and sends reply

AODV – An Example



AODV - An Example



to	via
a	a
b	b

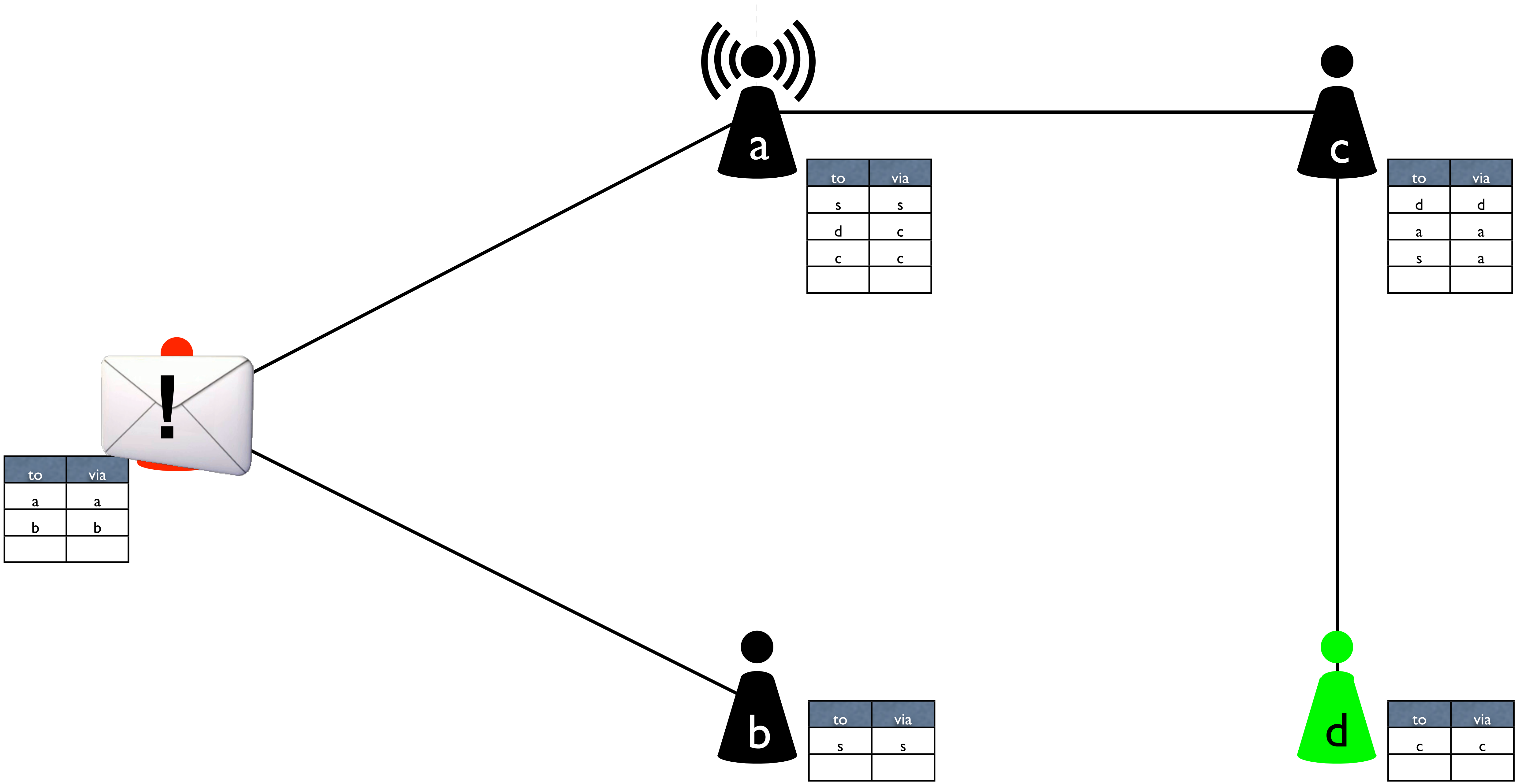
to	via
s	s
d	c
c	c

to	via
d	d
a	a
s	a

to	via
s	s

to	via
c	c

AODV – An Example



to	via
a	a
b	b

to	via
s	s
d	c
c	c

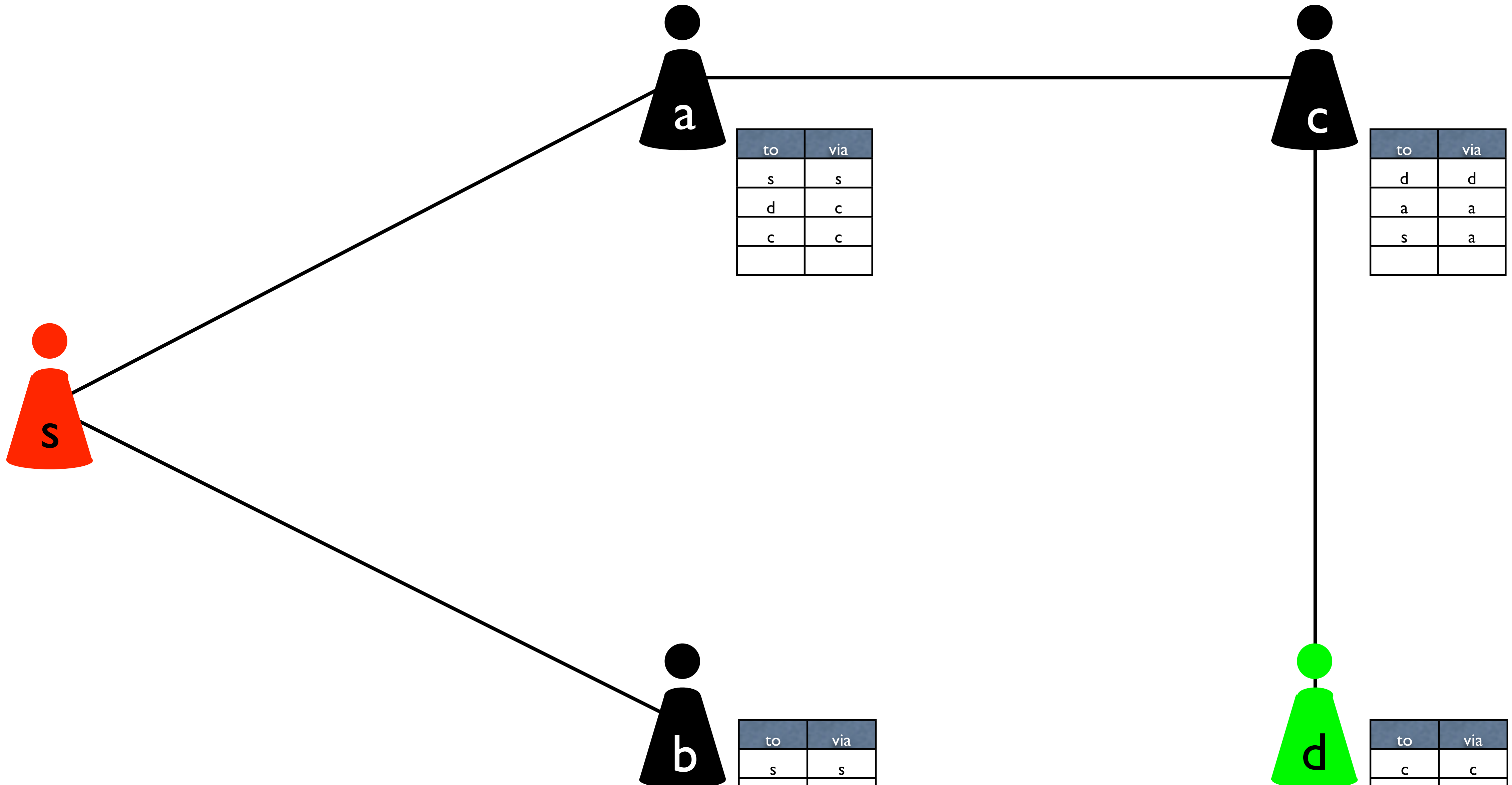
to	via
d	d
a	a
s	a

to	via
s	s

to	via
c	c

a forwards route reply

AODV – An Example



s:

to	via
a	a
b	b
d	a

a:

to	via
s	s
d	c
c	c

c:

to	via
d	d
a	a
s	a

b:

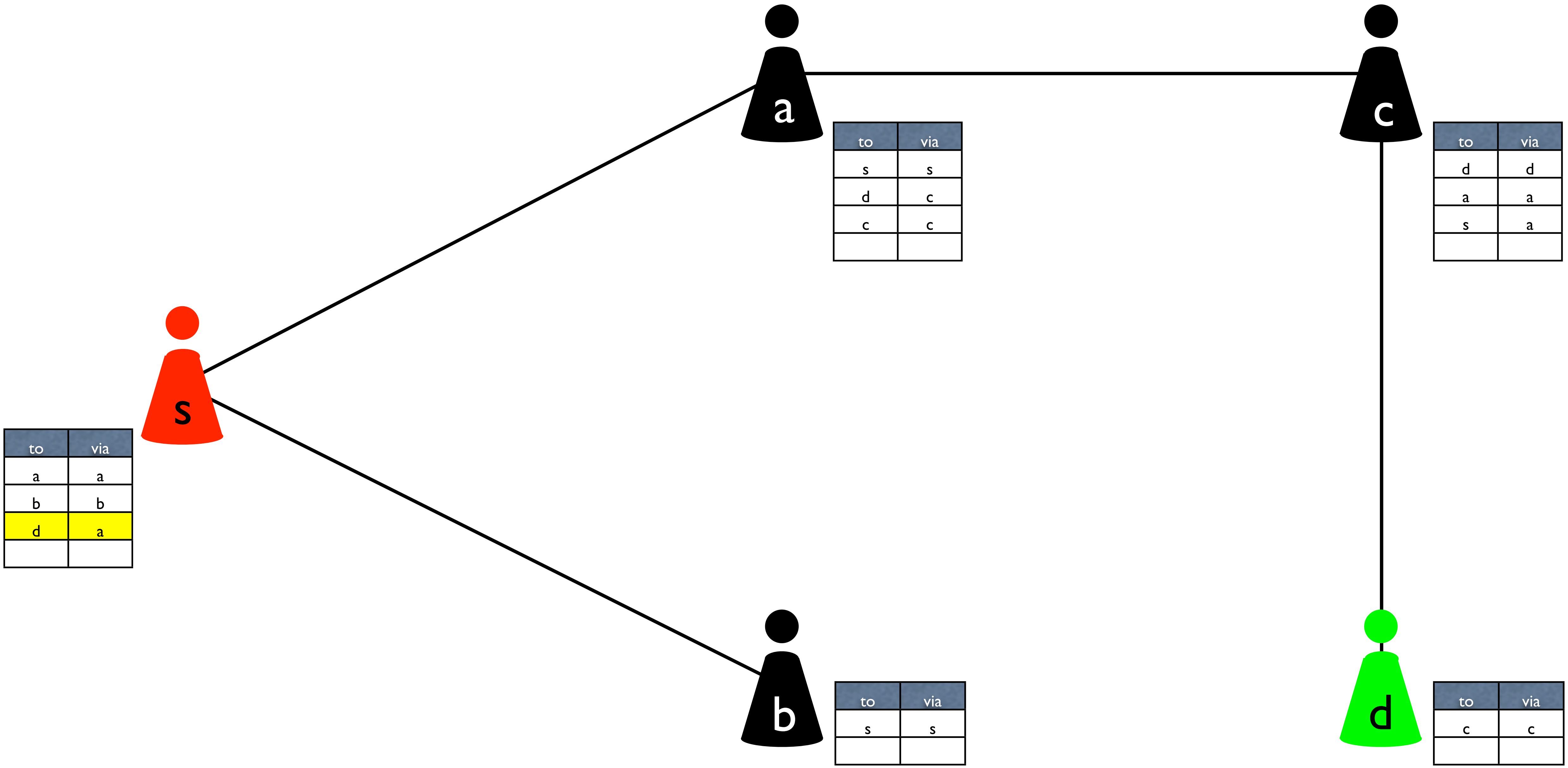
to	via
s	s

d:

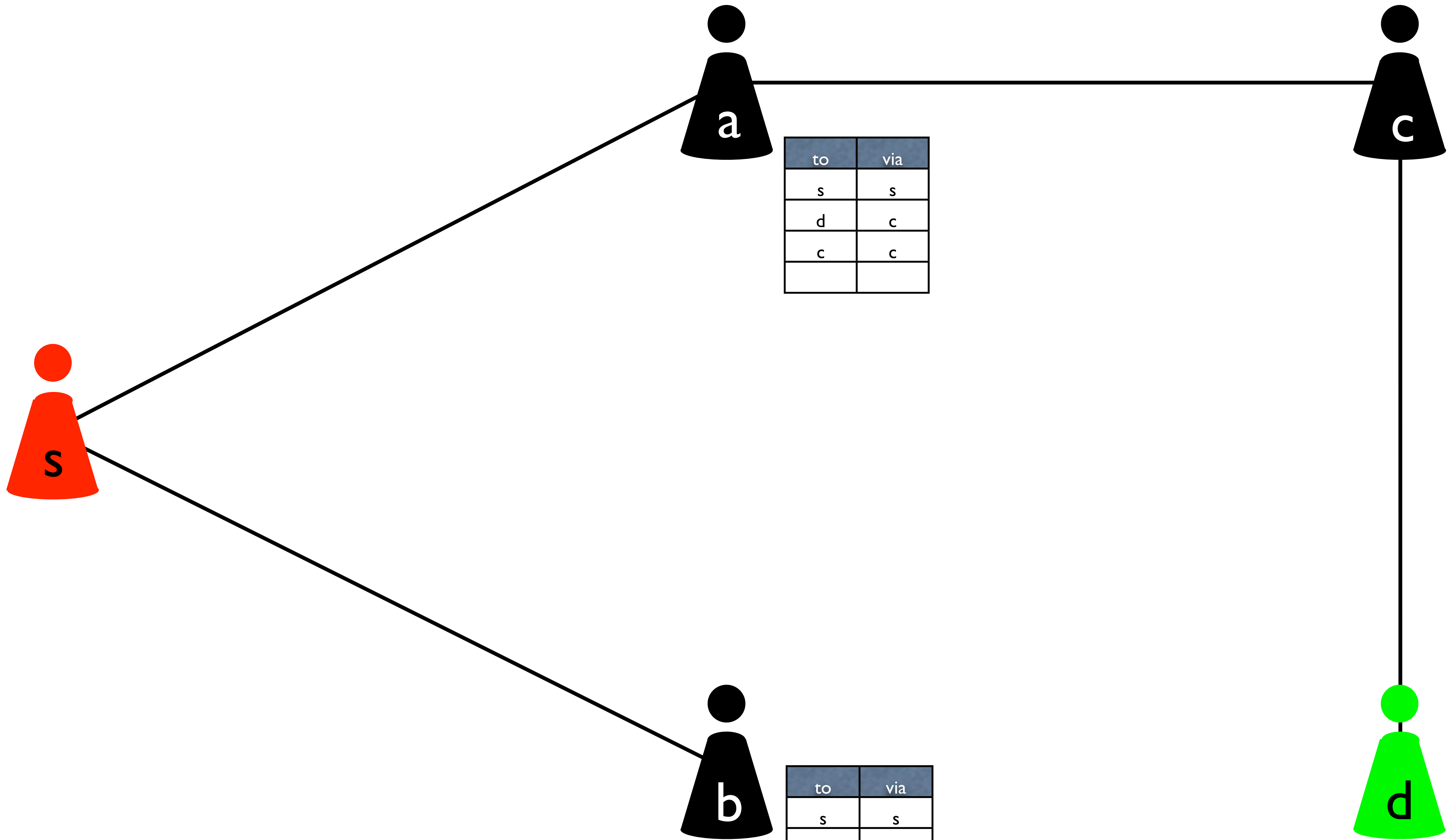
to	via
c	c

a forwards route reply

AODV – An Example



AODV – An Example



to	via
a	a
b	b
d	a

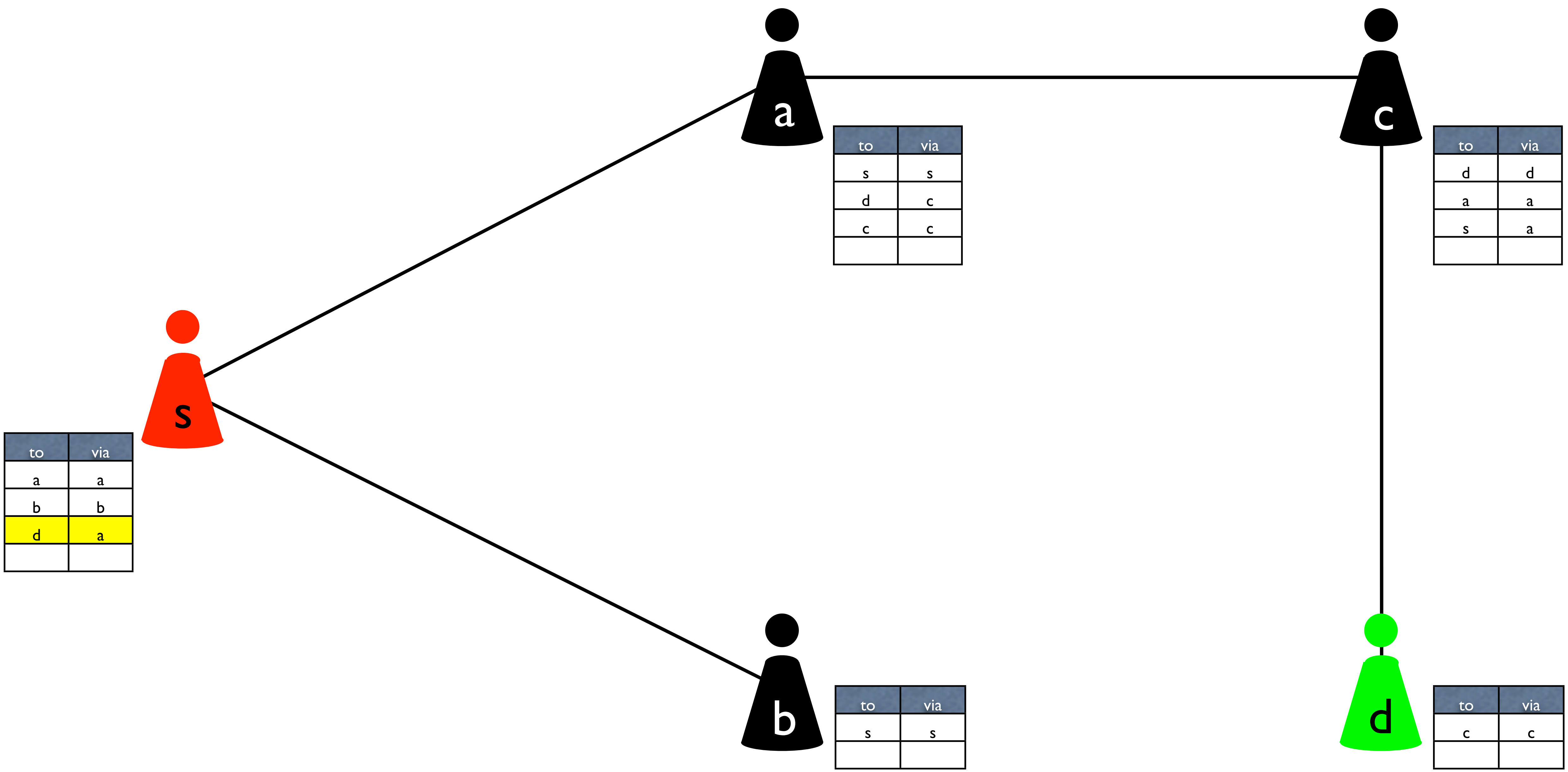
to	via
s	s
d	c
c	c

to	via
d	d
a	a
s	a

to	via
s	s

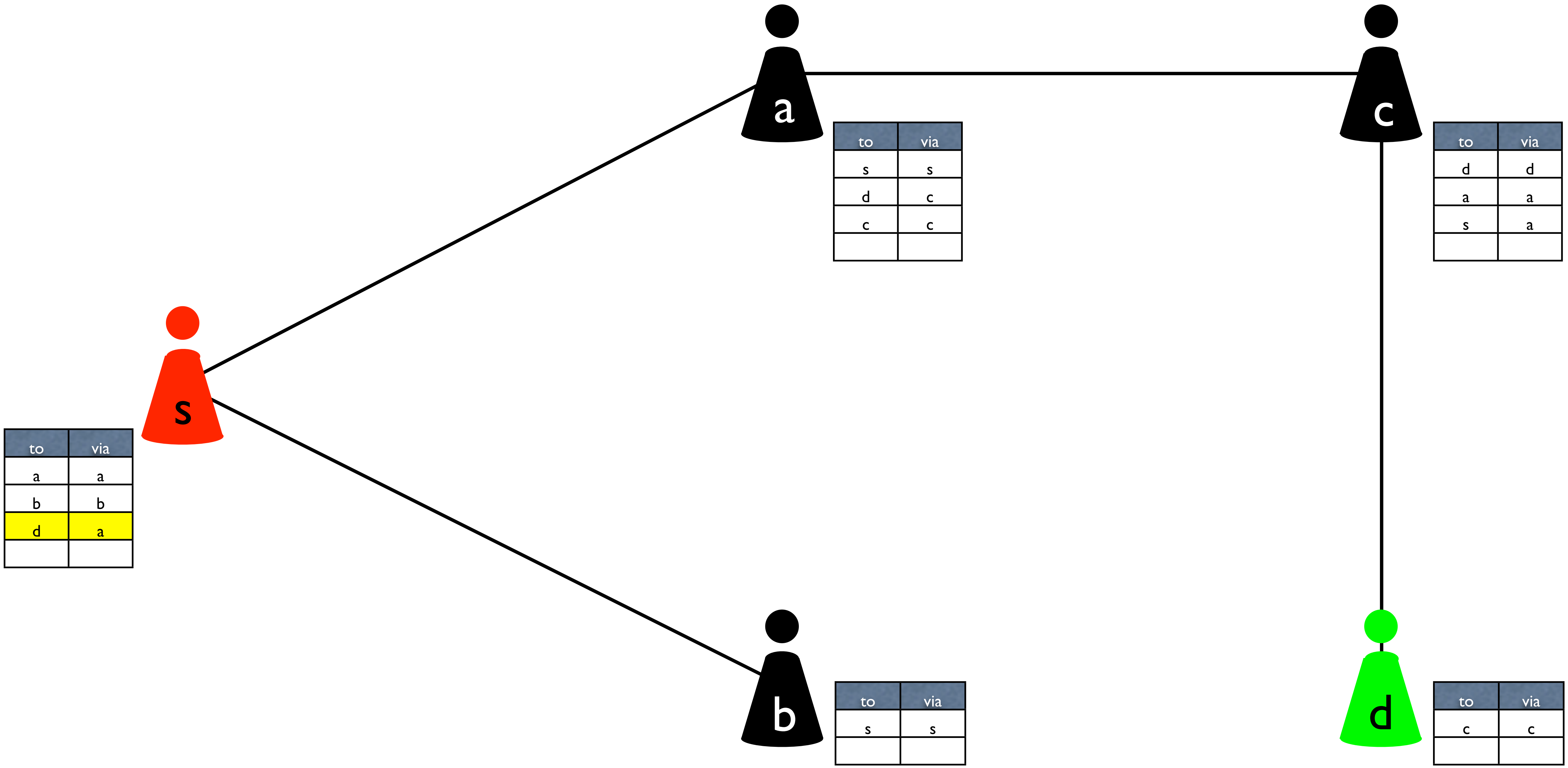
to	via
c	c

AODV – An Example



s has found a route to d

AODV – An Example



s has found a route to d

Process Algebra – Snippet

NICTA

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

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

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

NICTA

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