Formal Methods for Wireless Mesh Networks

Peter Höfner

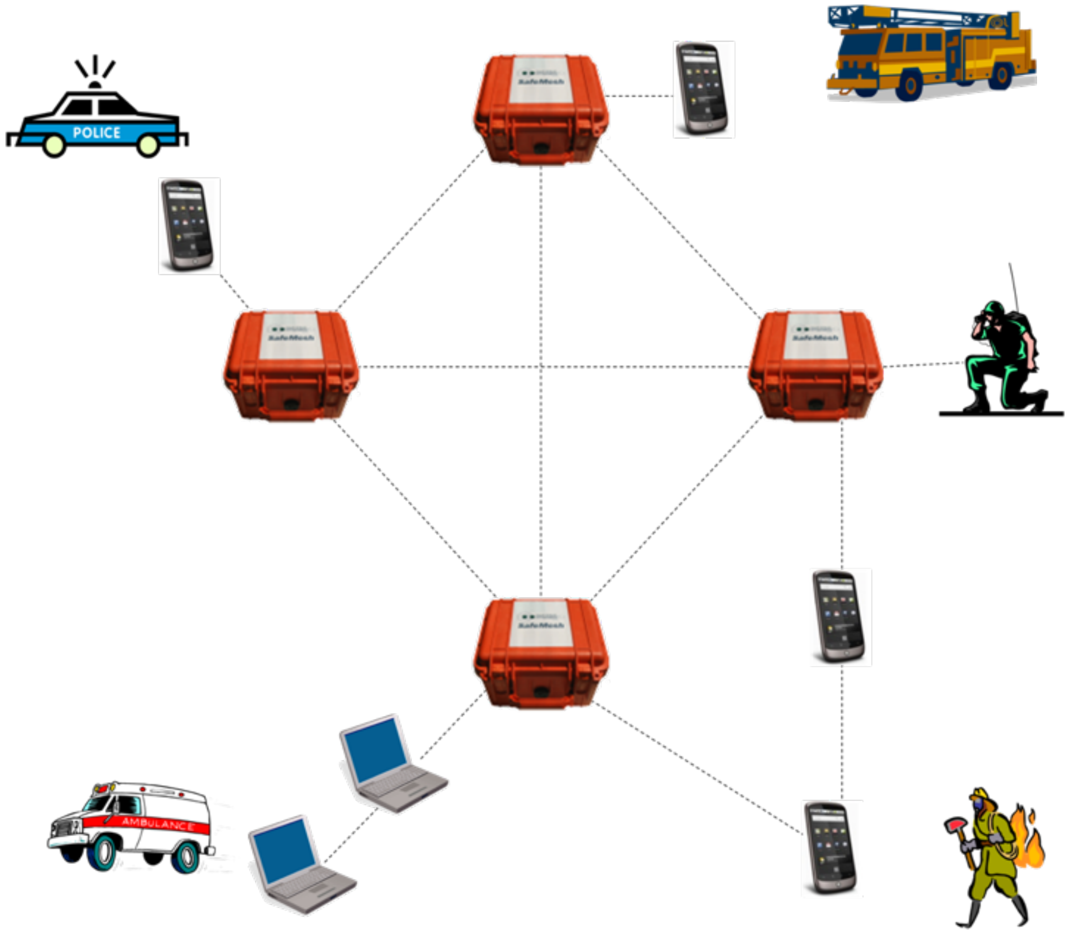
Australian Government
Department of Broadband, Communications and the Digital Economy
Australian Research Council

NICTA Members

NICTA Partners

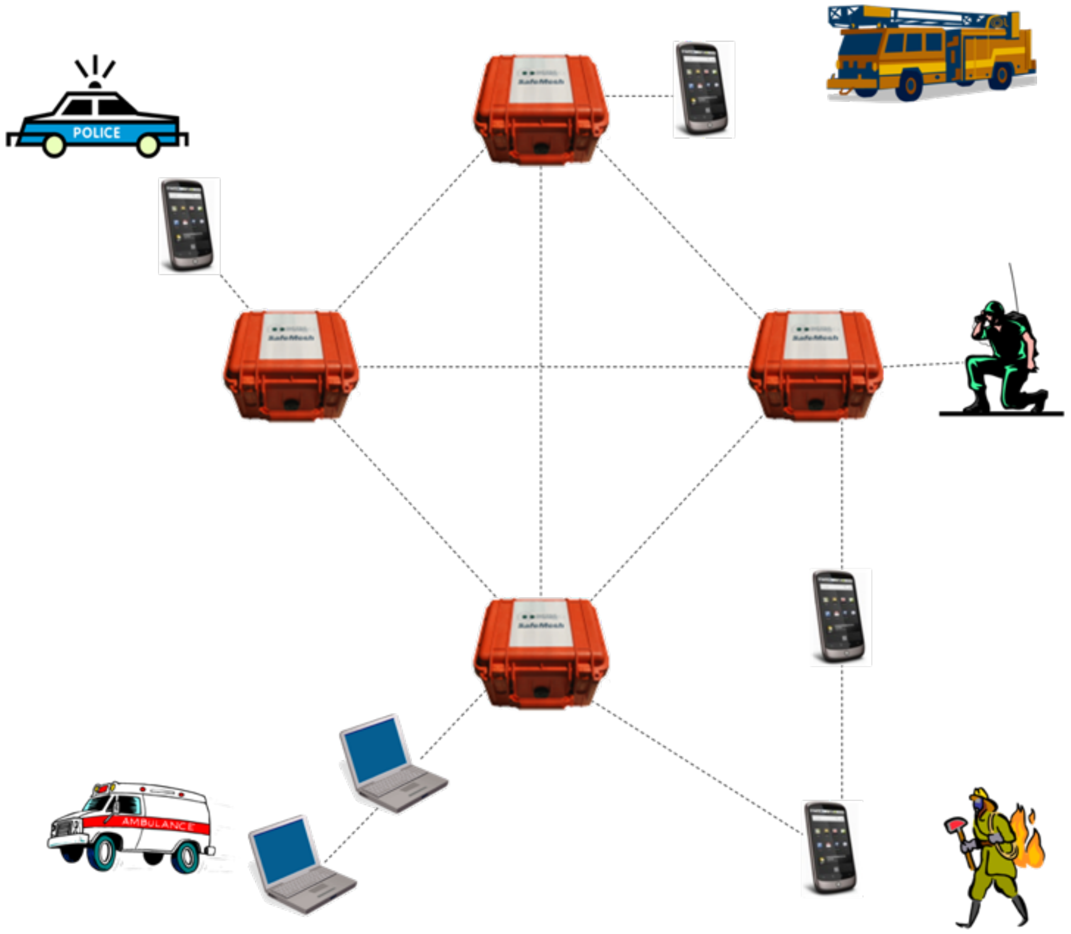
What is the Problem?

- 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
  - ...



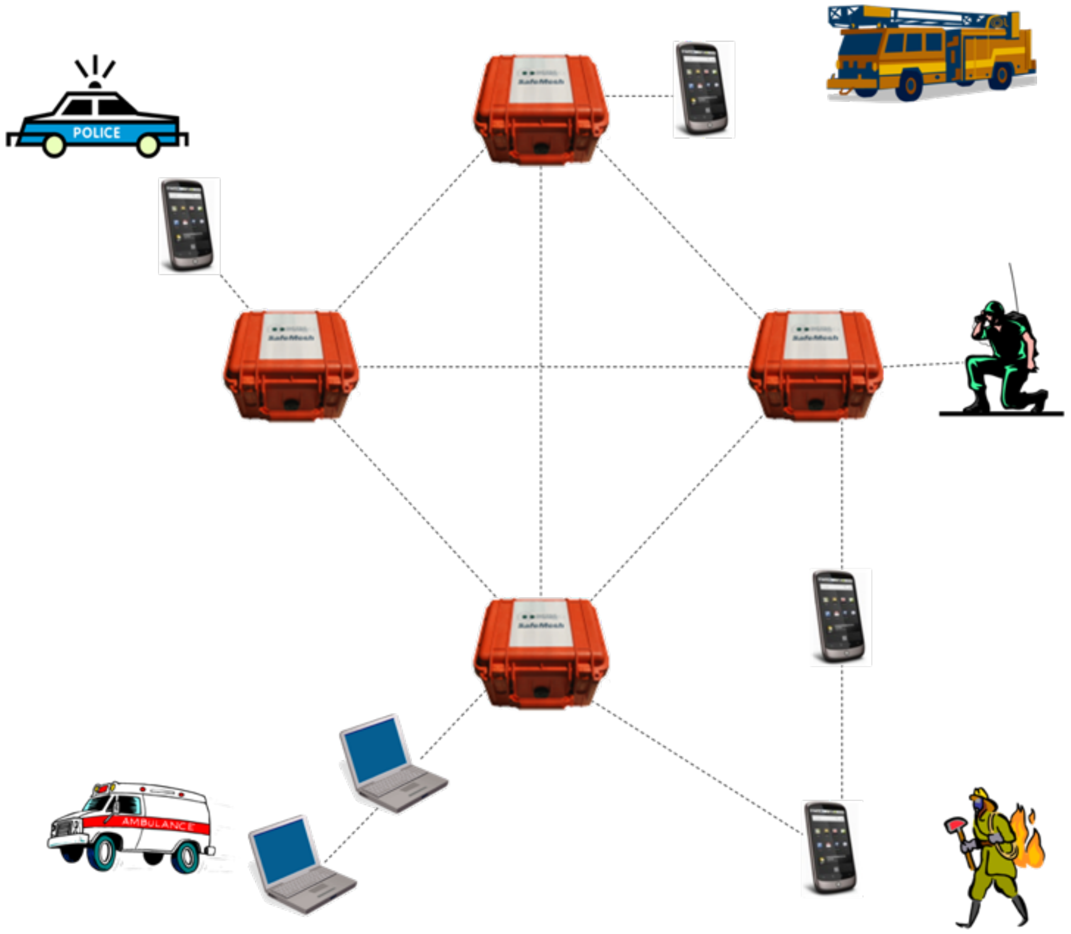
What is the Problem?

- 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
- **DOES NOT WORK** (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 [...]"

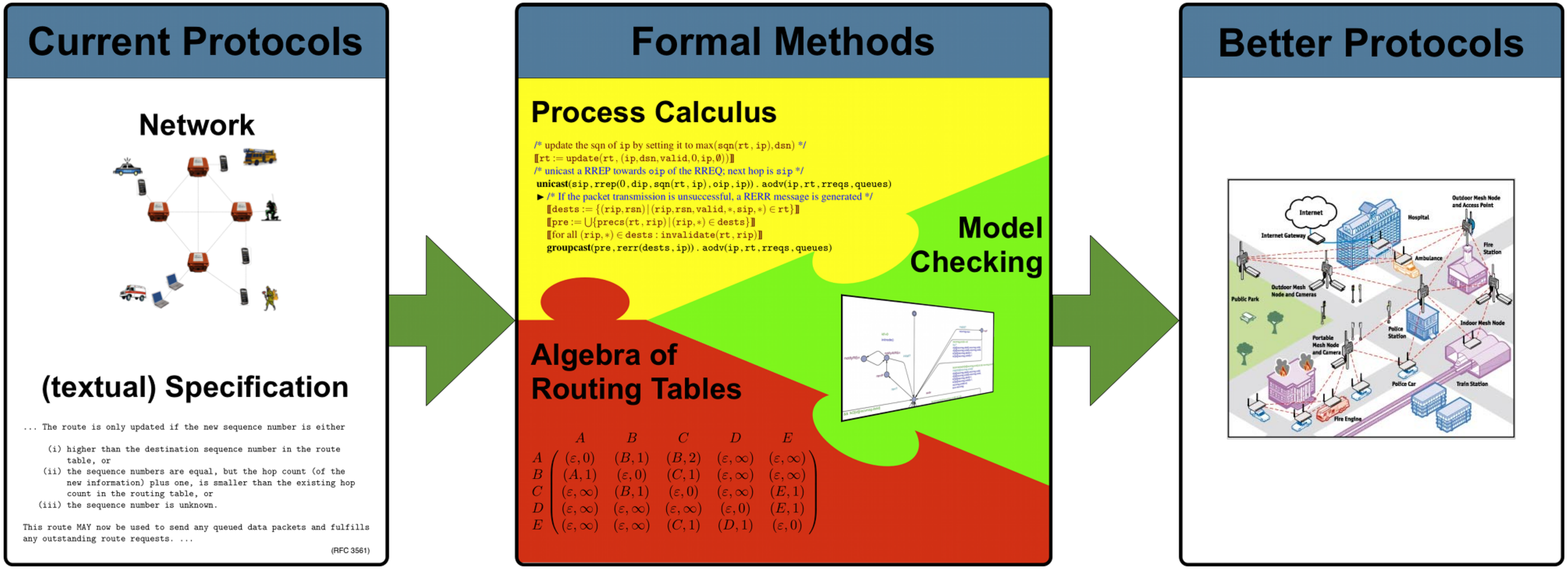
Rick Loebler, Applied Technology Manager,
NSW Police Force

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

Formal Methods for Mesh Networks

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



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)

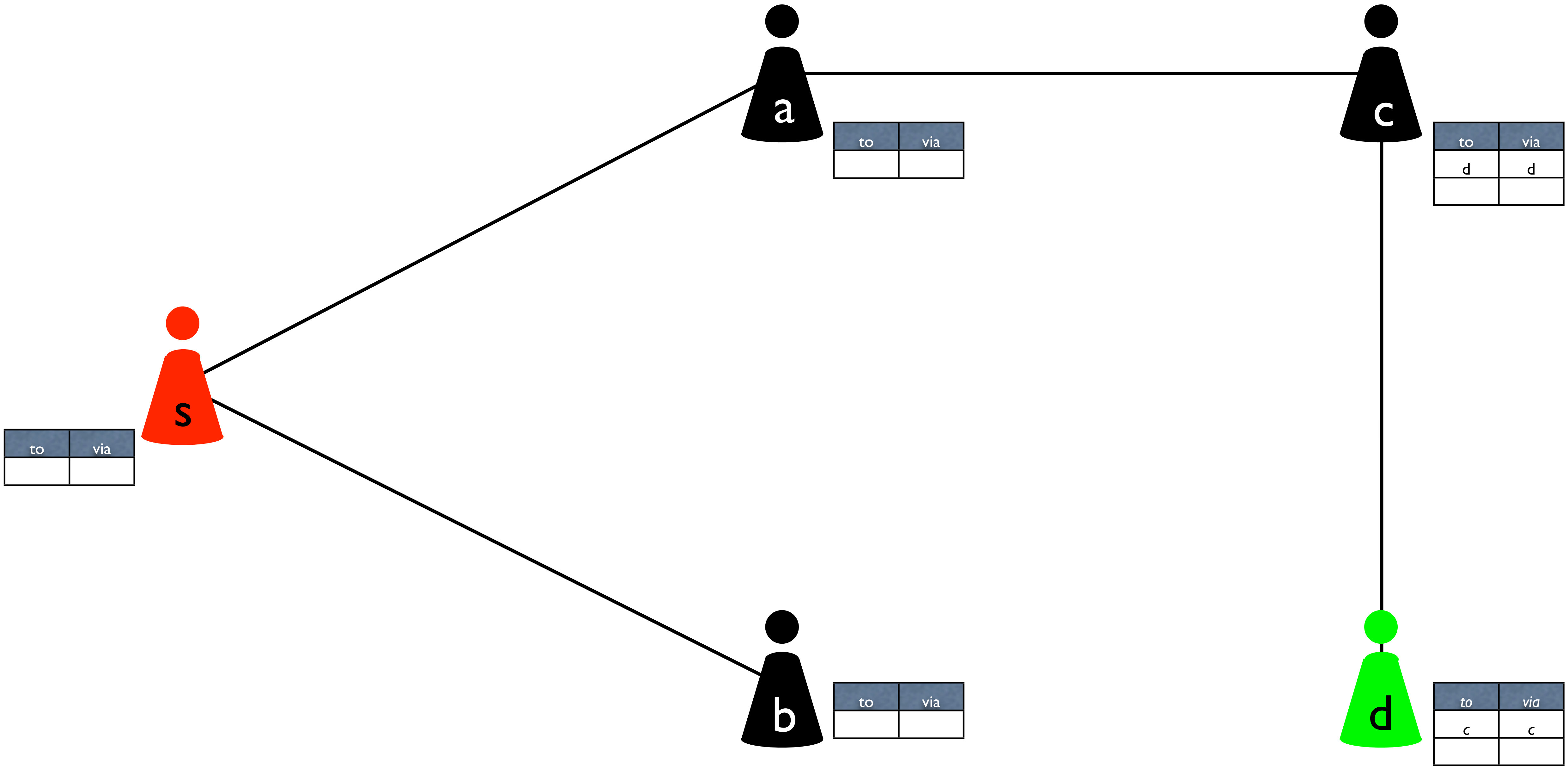
- Developed 1997-2001 by Perkins, Beldig-Royer and Das (University of Cincinnati)

Ad Hoc On-Demand Distance Vector Protocol

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

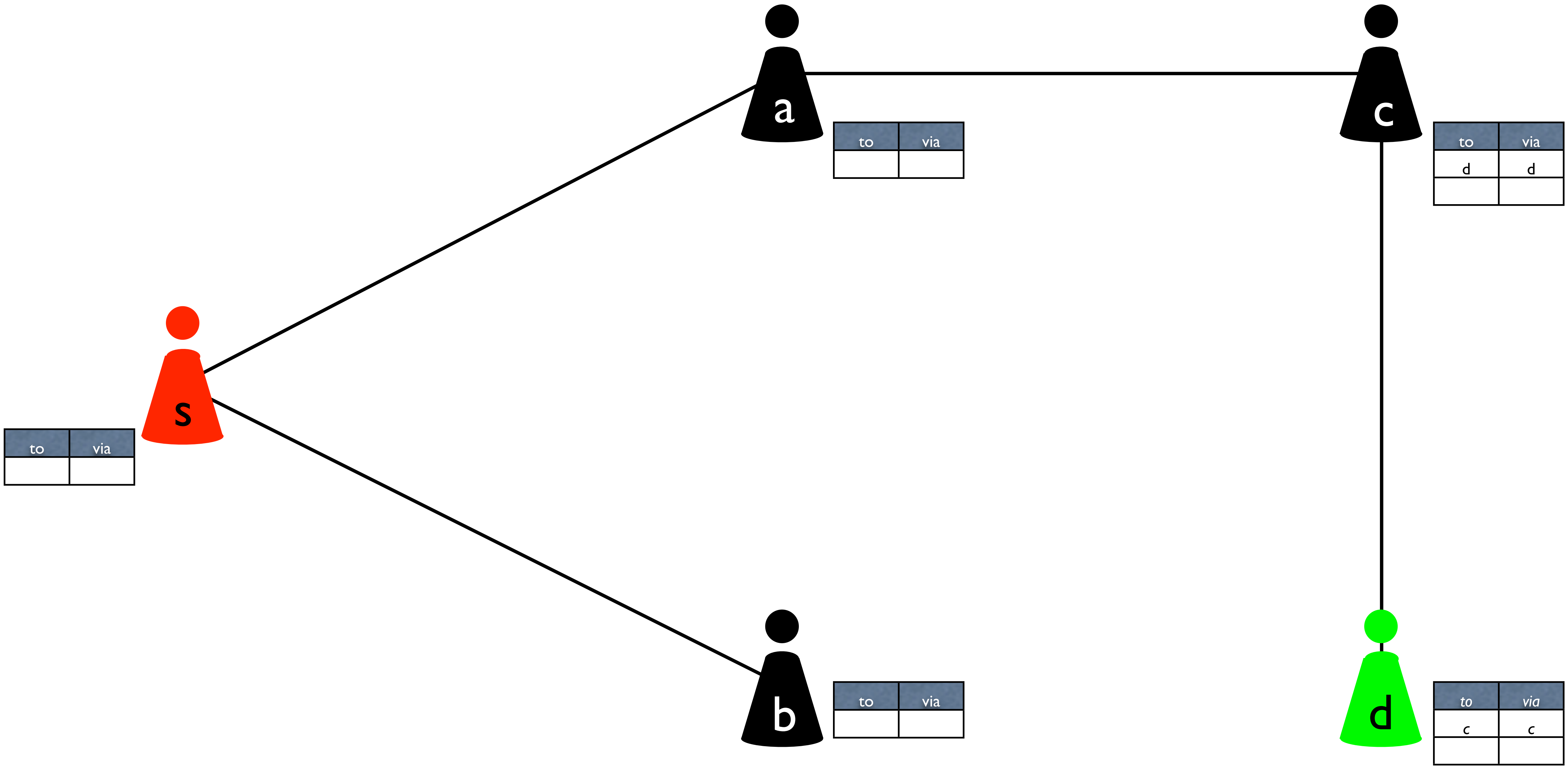
- Information at nodes
  - own IP address
  - a local sequence number (freshness/timer)
  - a routing table
    - local knowledge
    - entries: $(\mathtt{dip}, \mathtt{dsn}, \mathtt{val}, \mathtt{hops}, \mathtt{nhip}, \mathtt{pre})$

AODV - An Example

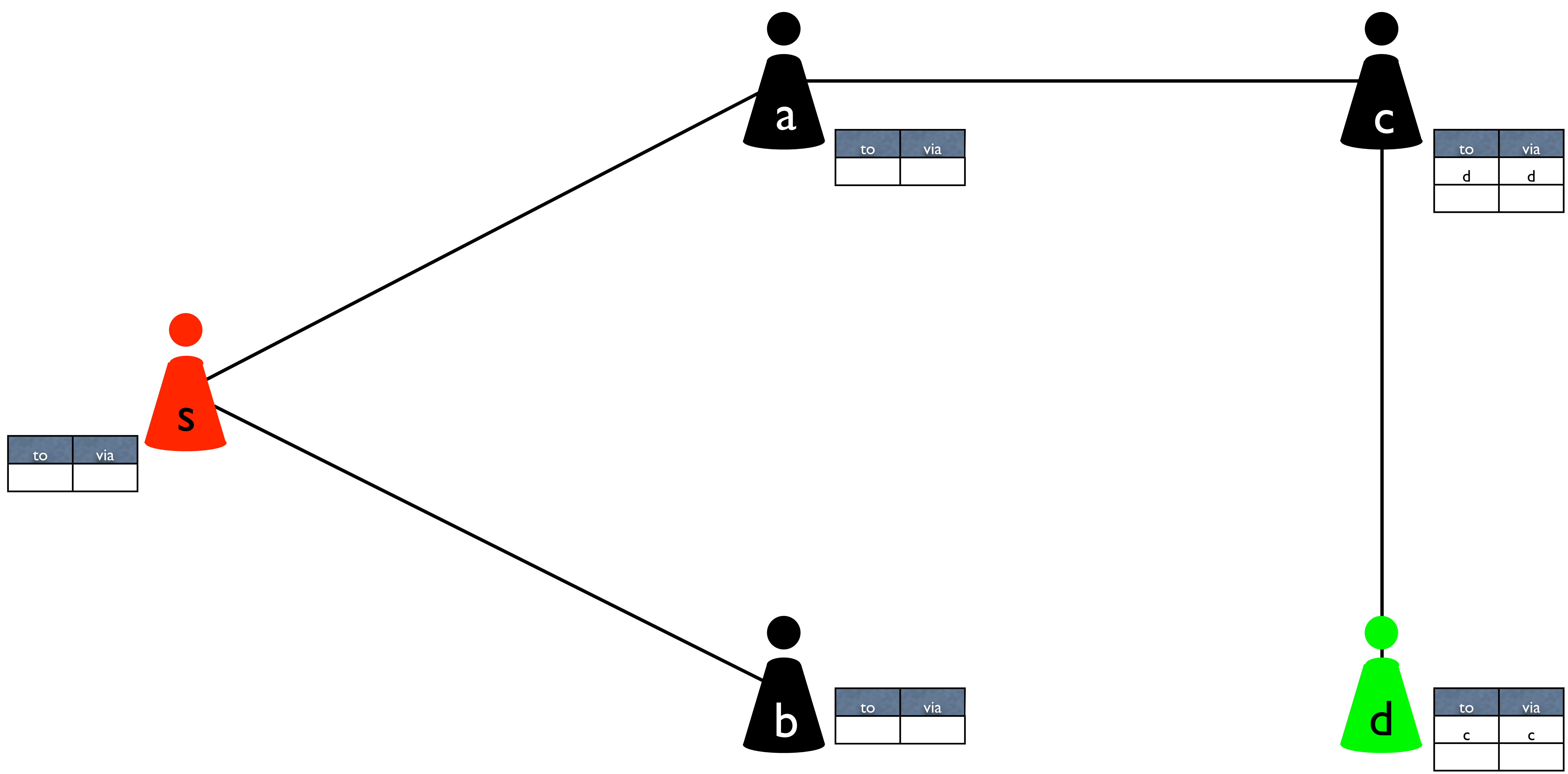


s is looking for a route to d

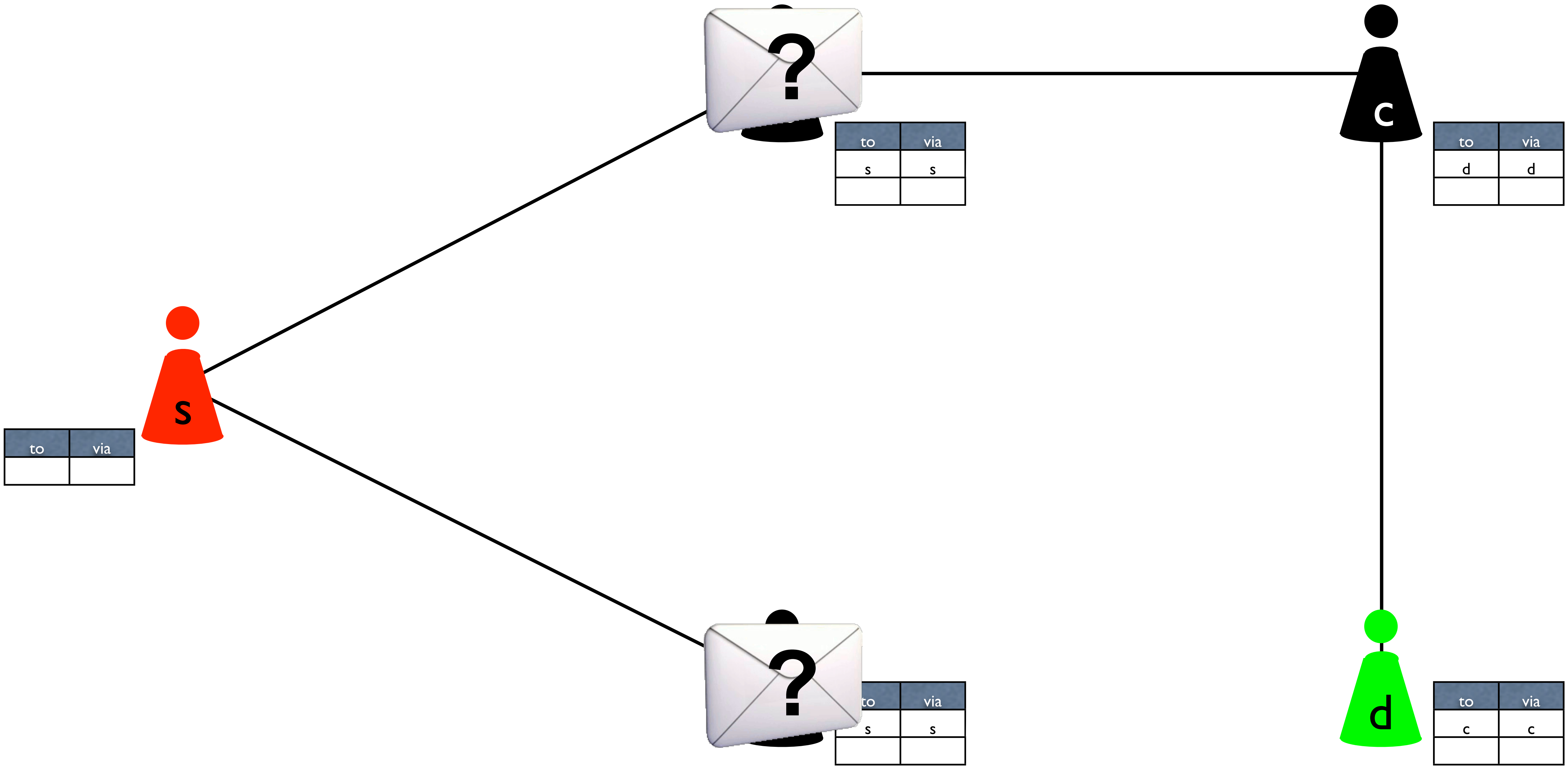
AODV – An Example



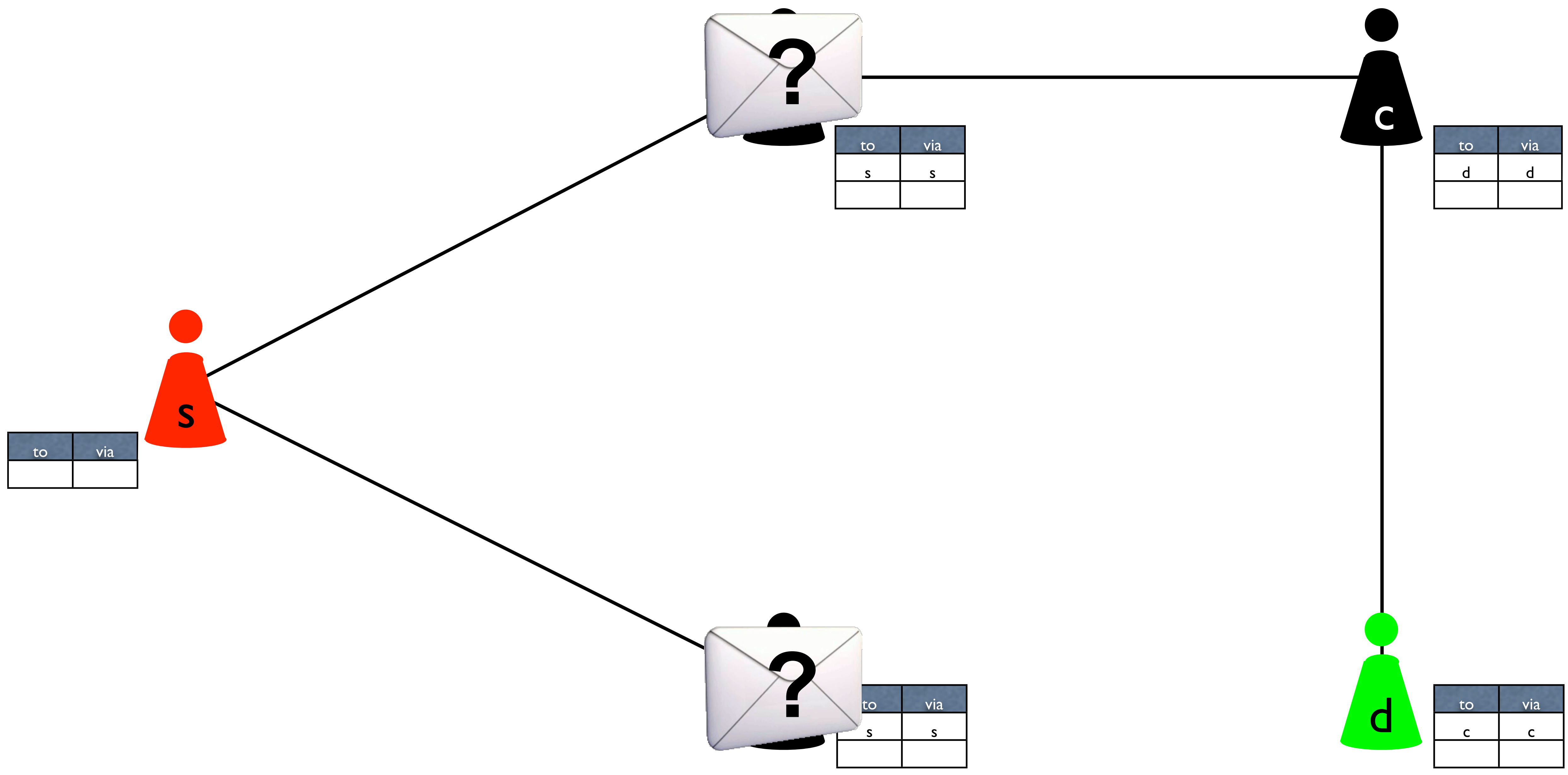
AODV – An Example



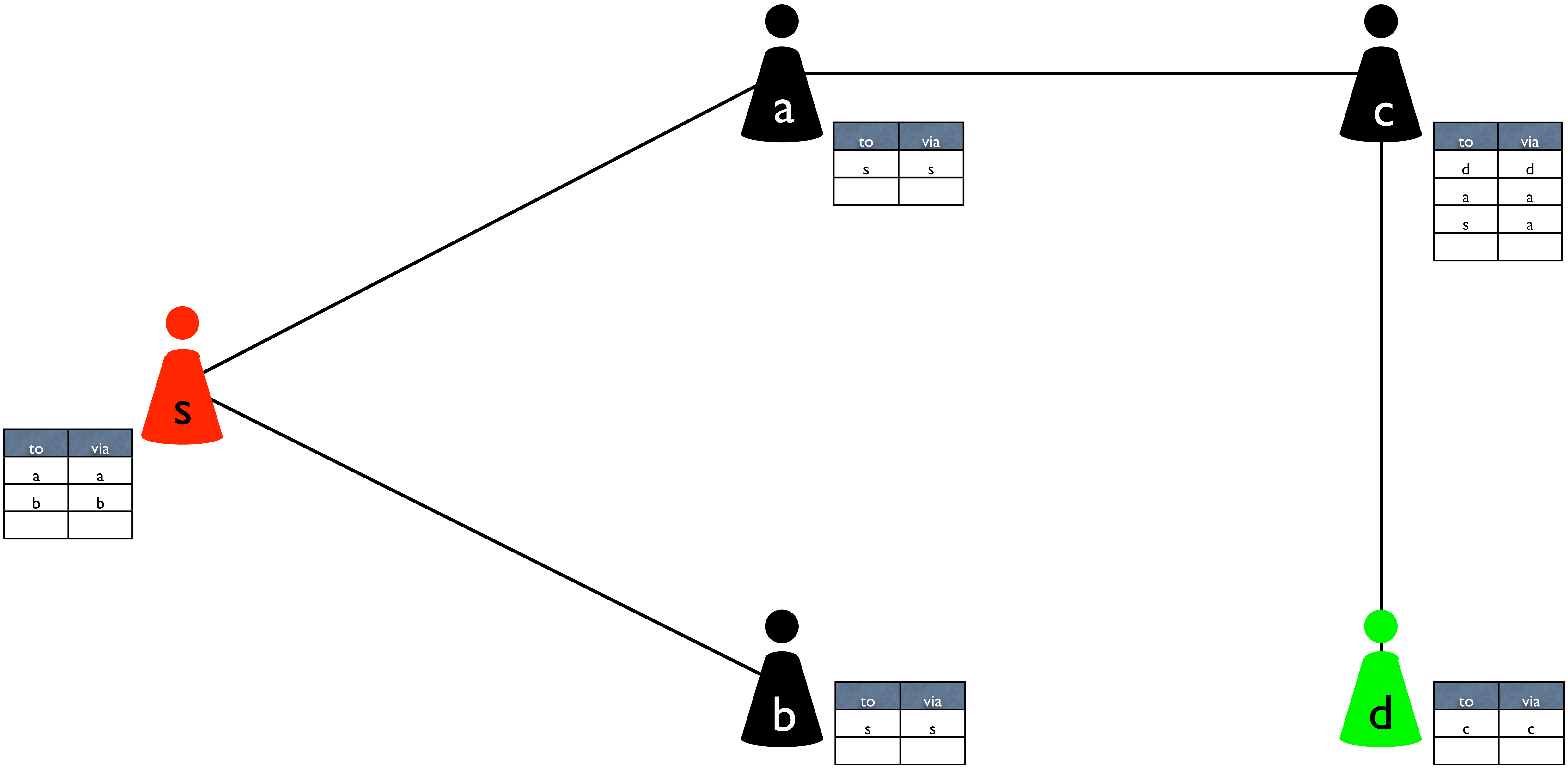
AODV – An Example



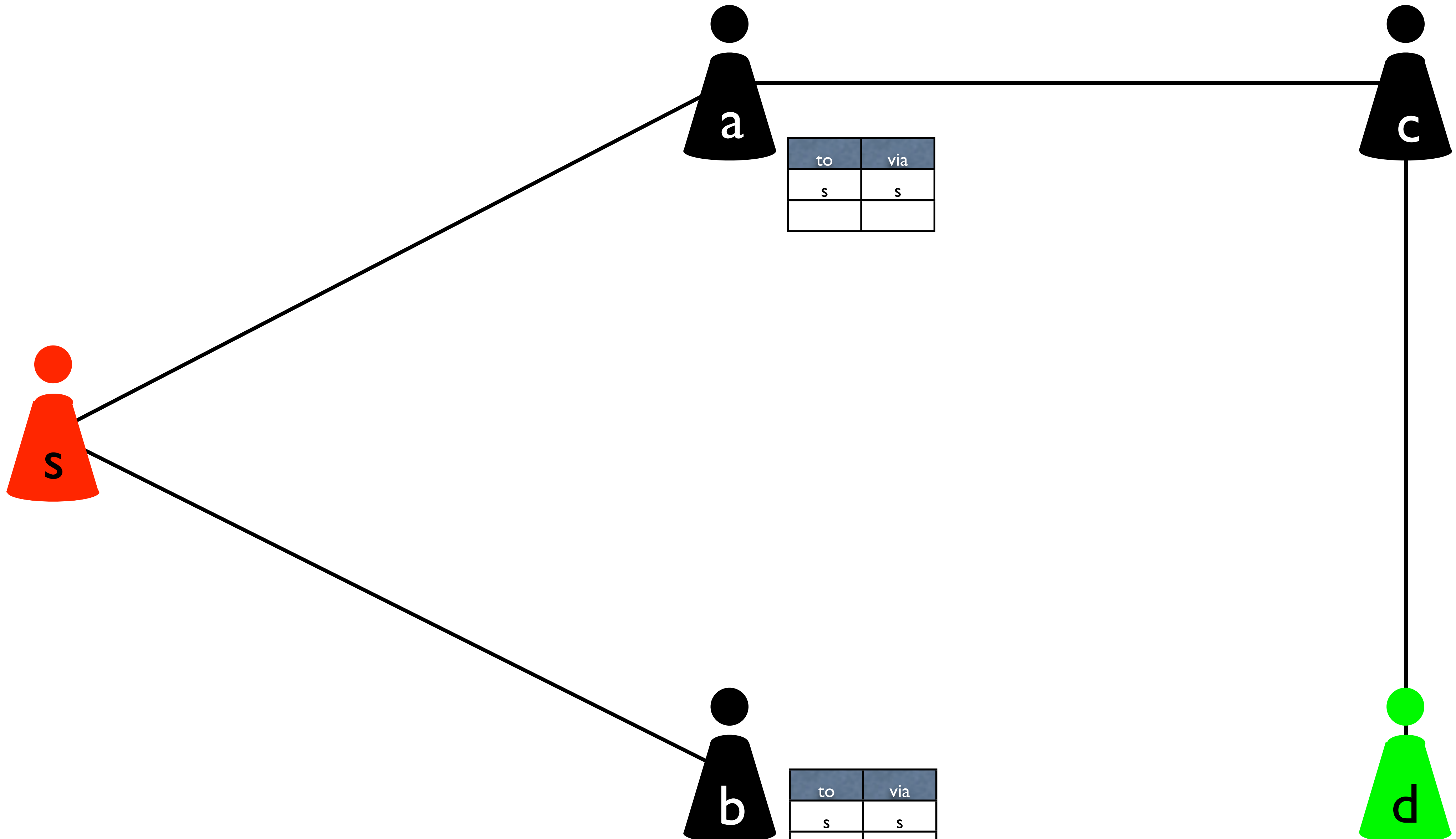
AODV – An Example



AODV – An Example



AODV – An Example



to	via
a	a
b	b

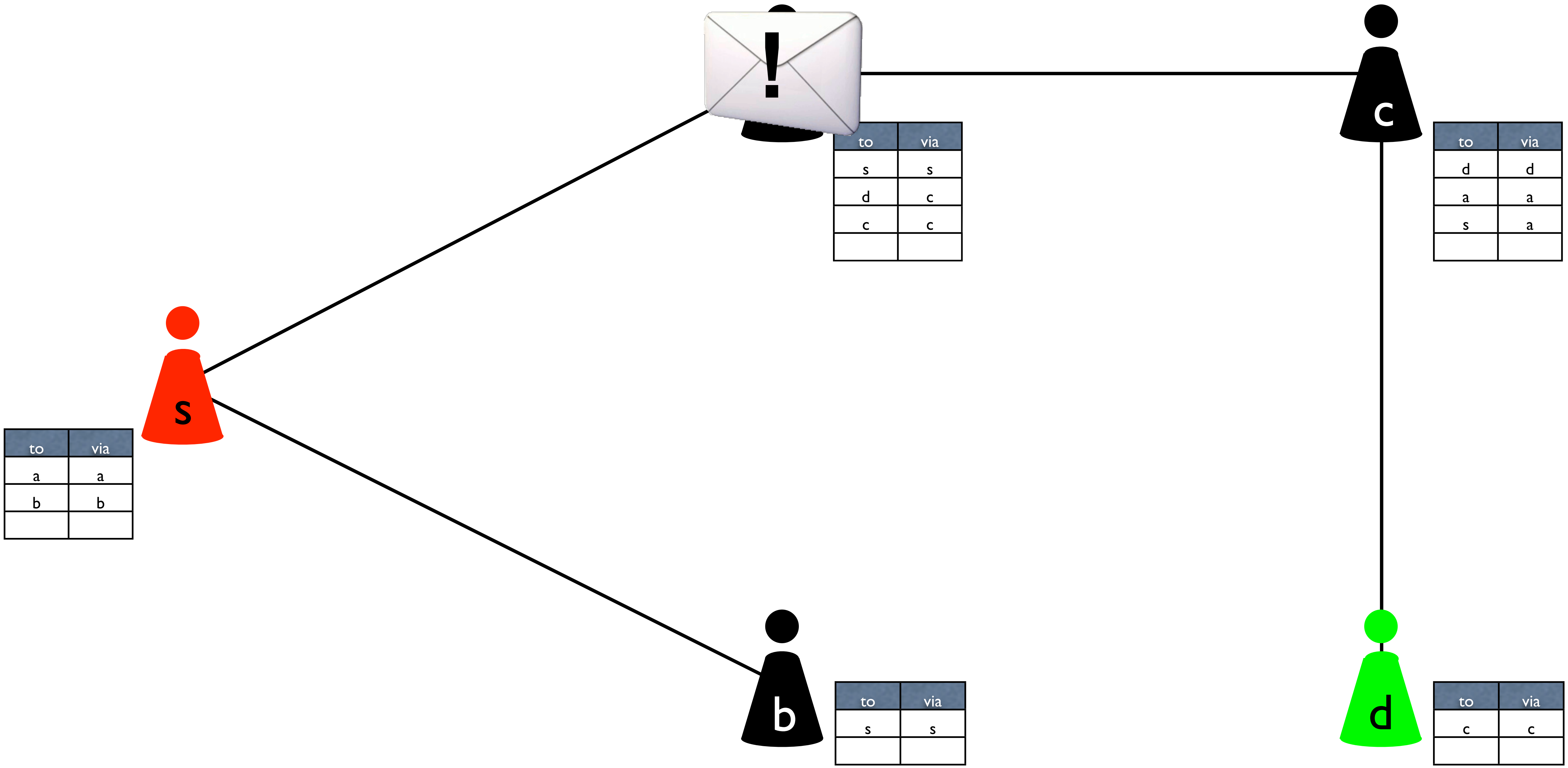
to	via
s	s

to	via
d	d
a	a
s	a

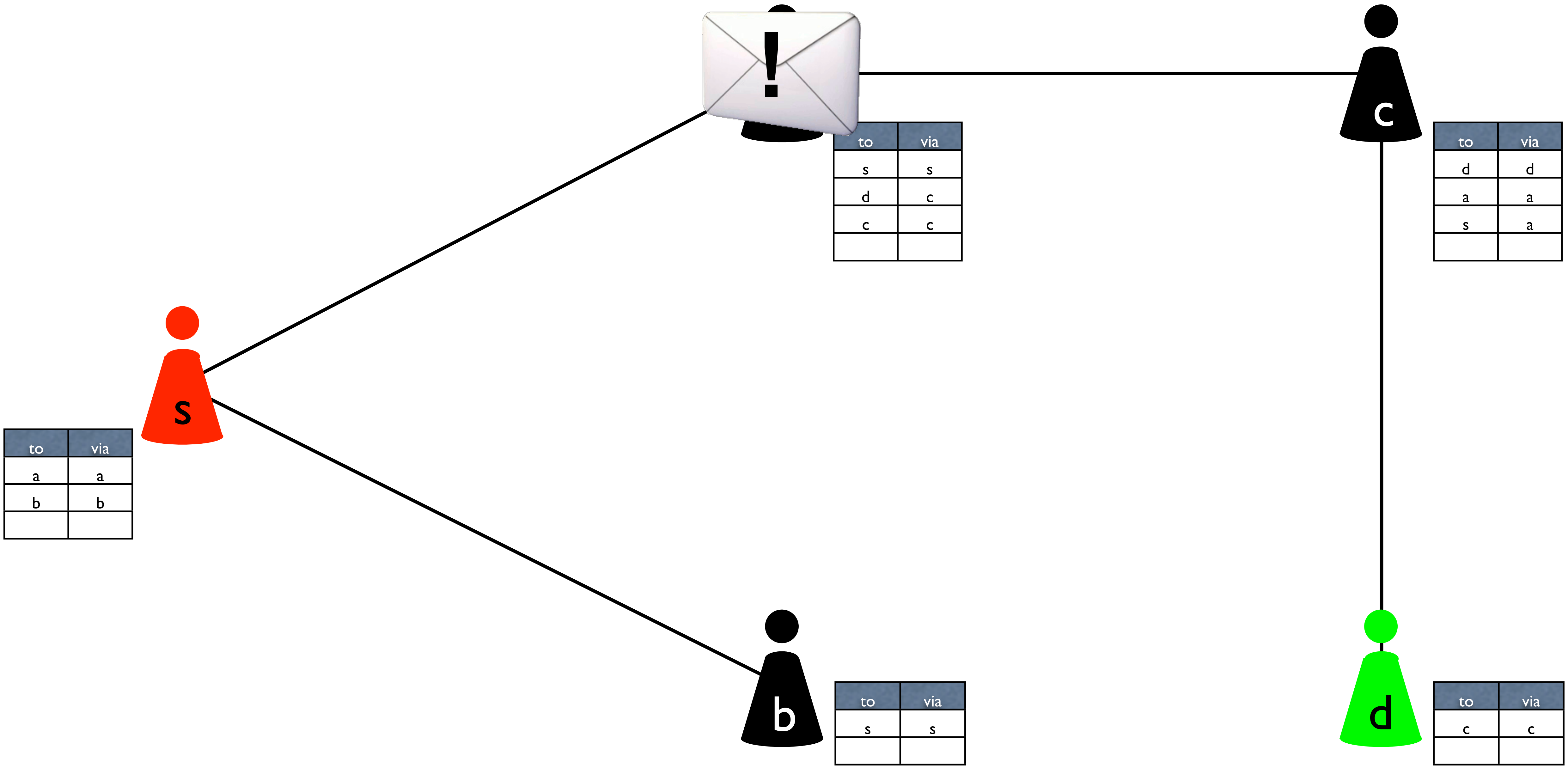
to	via
s	s

to	via
c	c

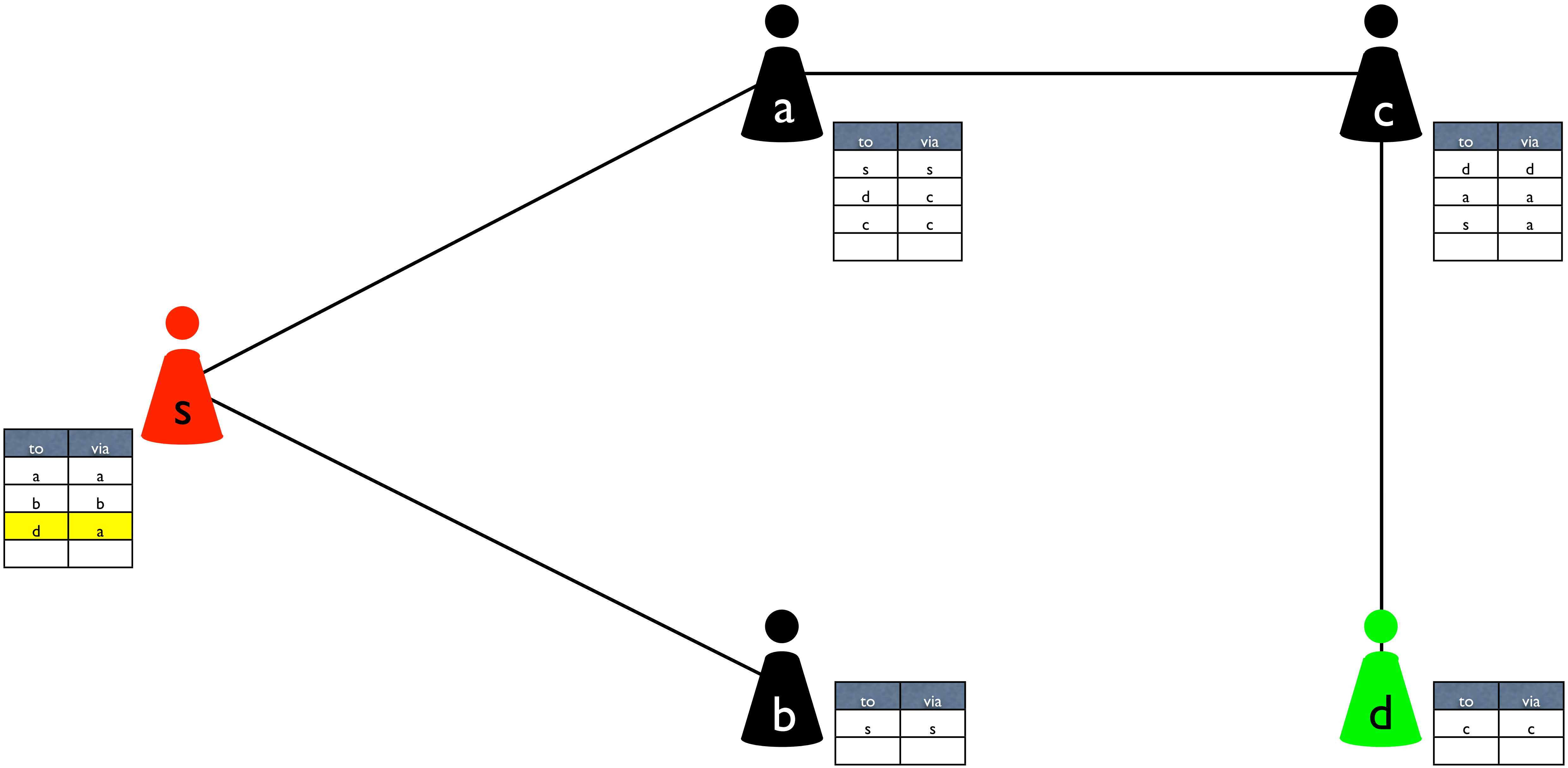
AODV – An Example



AODV – An Example



AODV – An Example



to	via
a	a
b	b
d	a

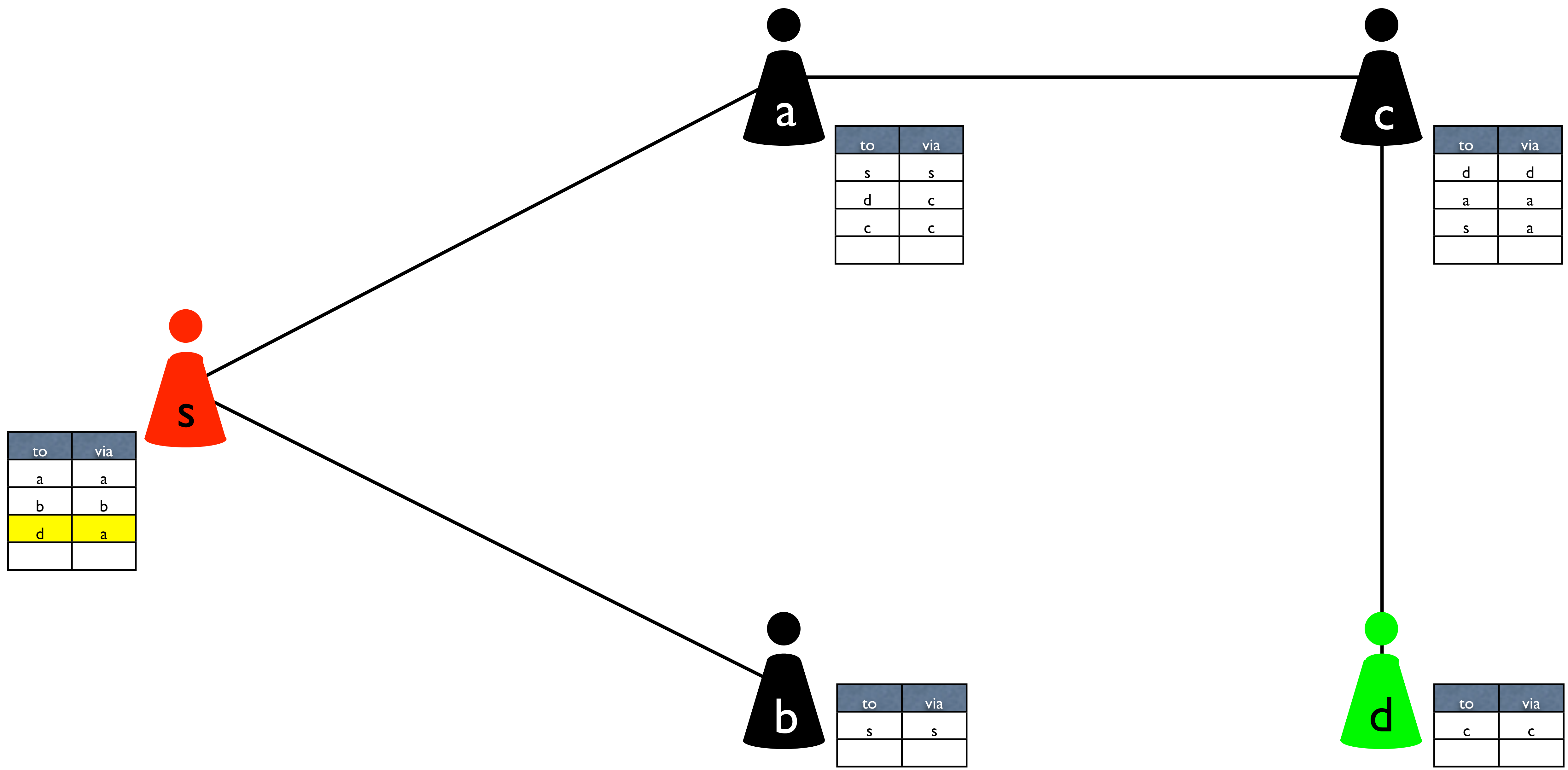
to	via
s	s
d	c
c	c

to	via
d	d
a	a
s	a

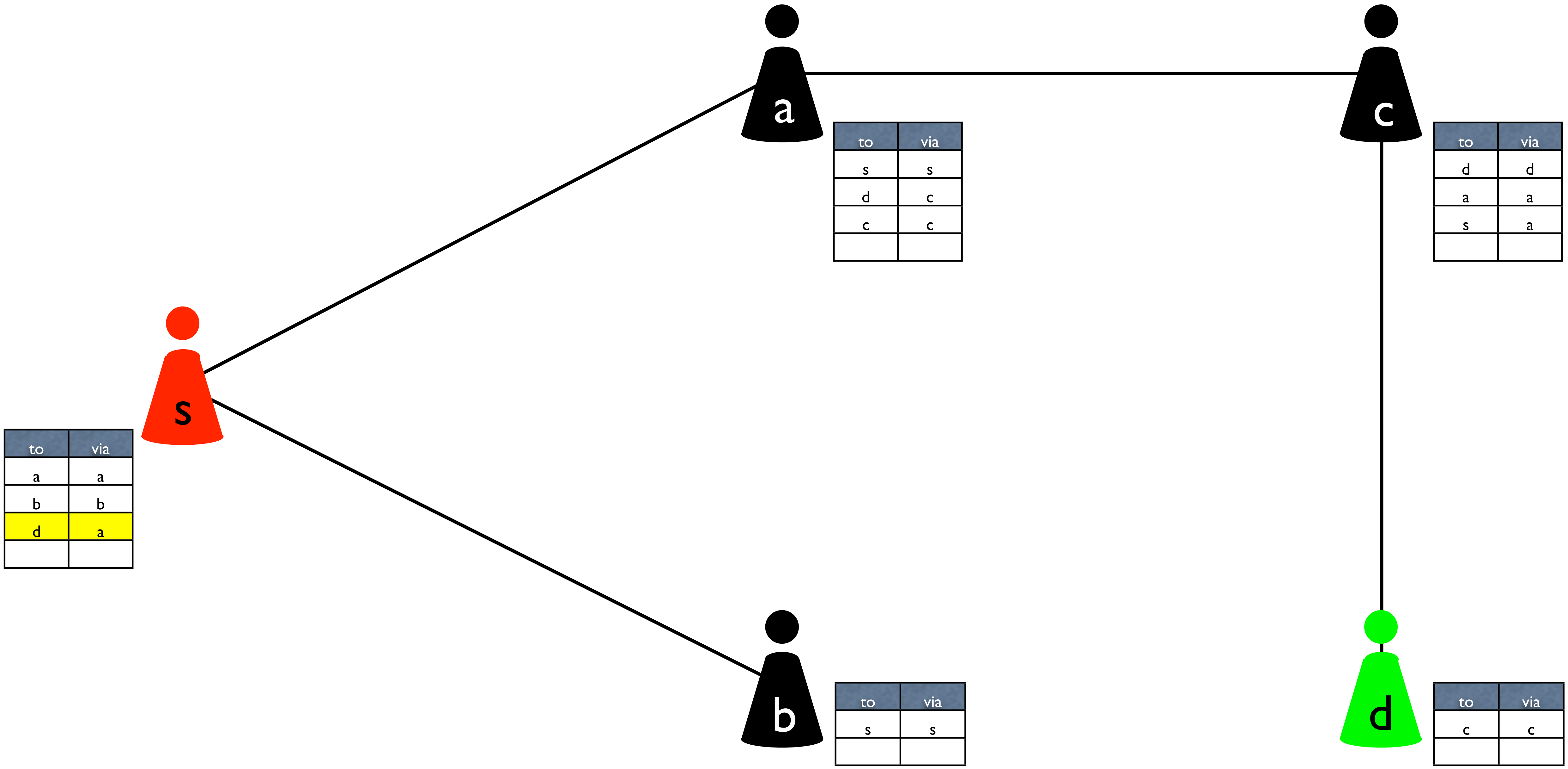
to	via
s	s

to	via
c	c

AODV – An Example



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

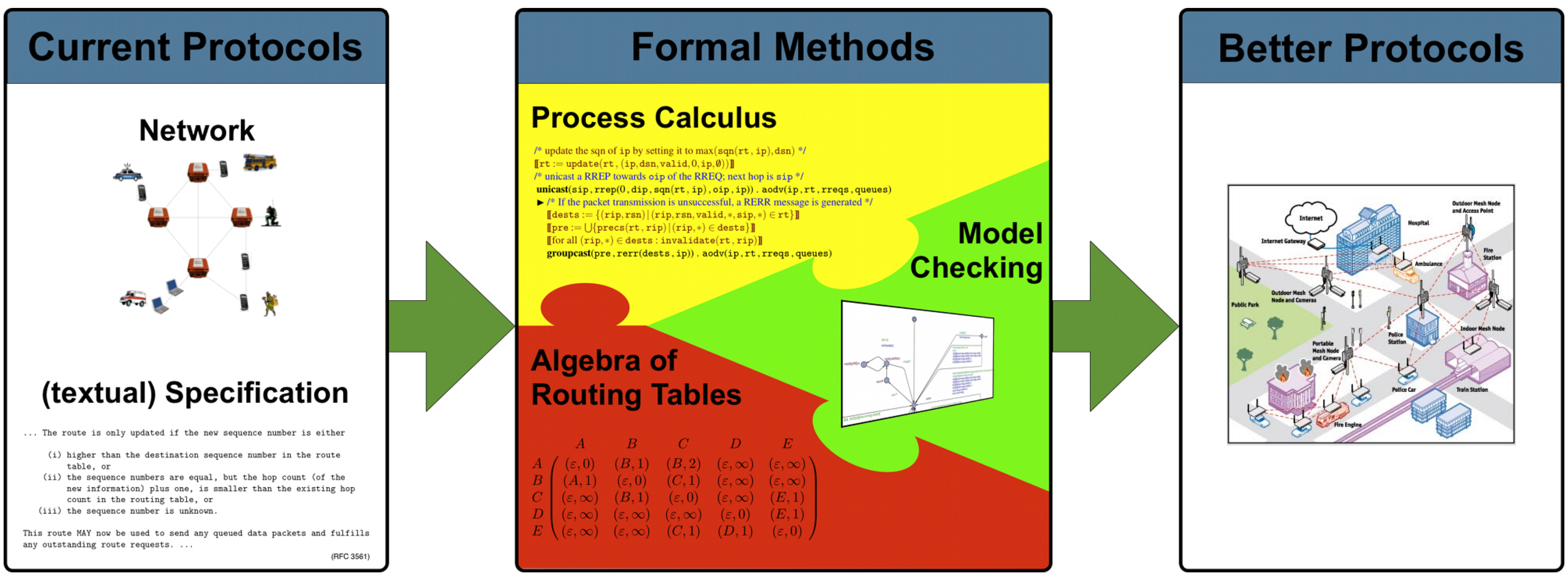
- Properties of AODV
  - route correctness ✓
  - loop freedom ✓
  - route found ✗
  - packet delivery ✗

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

Formal Methods for Mesh Networks

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



Process Algebra

$+ [\,(\mathtt{oip}, \mathtt{rreqid}) \notin \mathtt{rreqs}\,]$ /* 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 $\mathtt{sip}$ */

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

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

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

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

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

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

$+ [\,\mathtt{dip} \neq \mathtt{ip}\,]$ /* 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

- 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

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

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

UPPAAL Model Checker

- 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

Analysis of AODV

- evaluation of WMN routing protocols

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

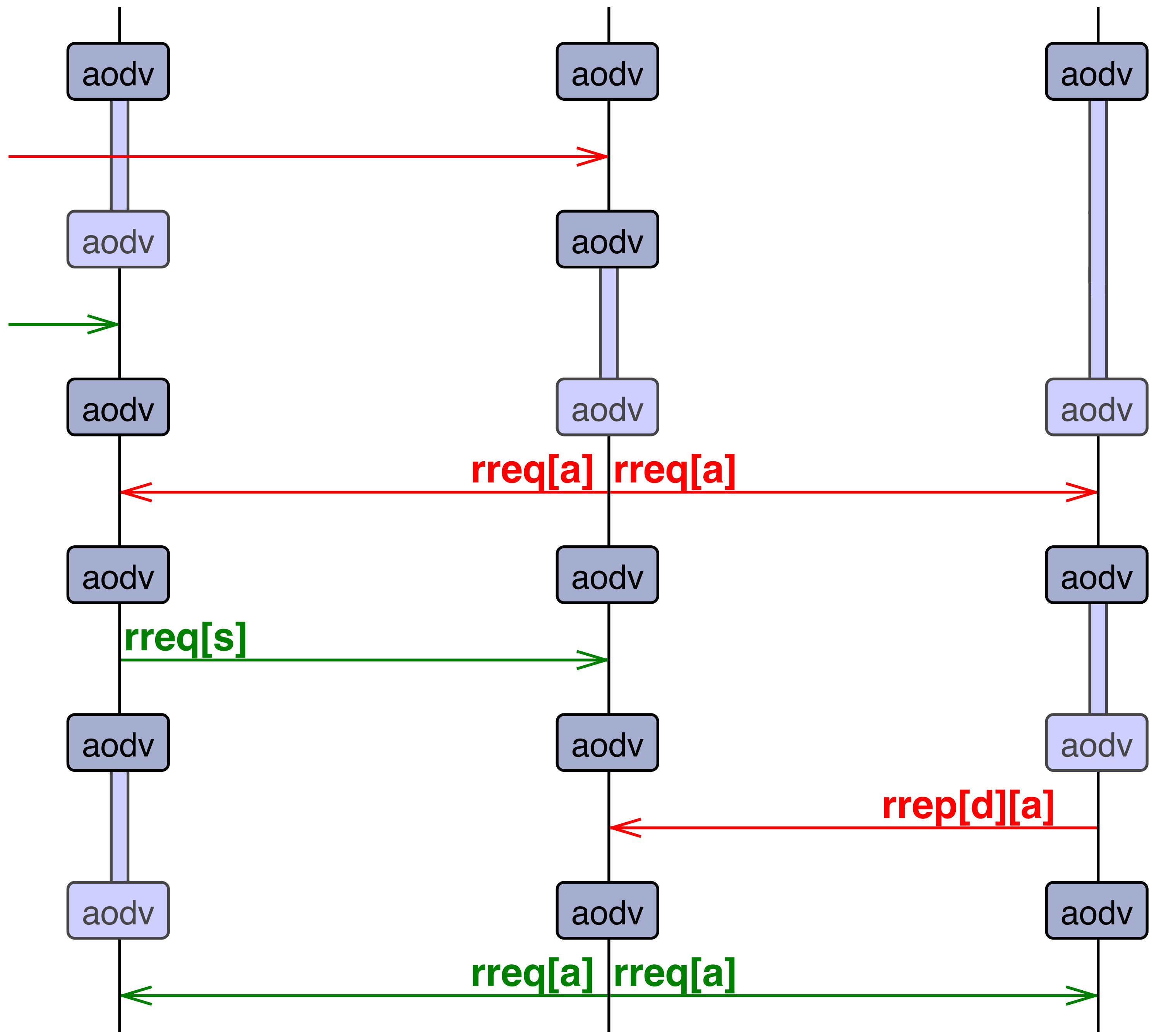
- no test-bed or simulation-based experiments
  - important and valid methods for protocol evaluation
  - but resource intensive and time-consuming
- complements proofs in AWN
  - based on same spec is important

Experiments Set-Up

- Exhaustive search
  - different properties
  - all topologies up to 5 nodes (one topology change)
  - 2 route discovery processes
  - 17400 scenarios
  - variants of AODV (4 models)

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
-

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

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

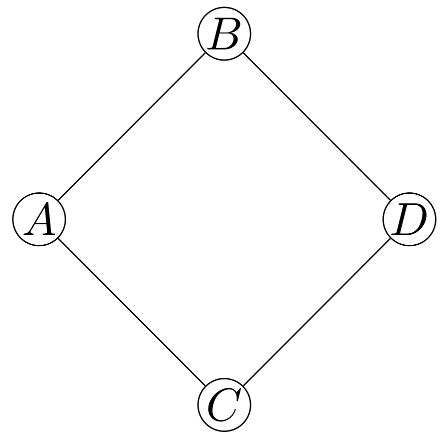
routing table of A

“routes” to B

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

Example

- A route request is broadcast



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

topology | sender | routing table

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

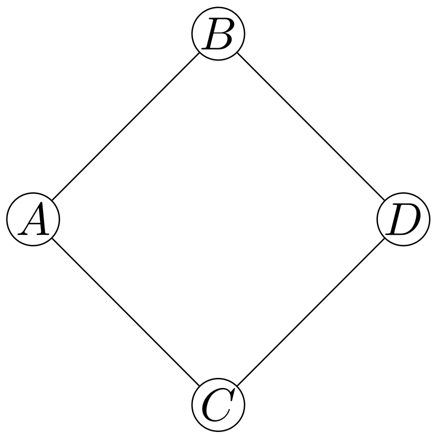
updated routing table

Further Abstraction

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

Example

- A route request is broadcast



$$\underbrace{\begin{pmatrix} (\_,0) & (B,1) & (C,1) & (\_,\infty) \\ (A,1) & (\_,0) & (\_,\infty) & (D,1) \\ (A,1) & (\_,\infty) & (\_,0) & (D,1) \\ (\_,\infty) & (B,1) & (C,1) & (\_,0) \end{pmatrix}}_{\text{topology}} \bullet \underbrace{\begin{pmatrix} (\_,0) & (\_,\infty) & (\_,\infty) & (\_,\infty) \\ (\_,\infty) & (\_,\infty) & (\_,\infty) & (\_,\infty) \\ (\_,\infty) & (\_,\infty) & (\_,\infty) & (\_,\infty) \\ (\_,\infty) & (\_,\infty) & (\_,\infty) & (\_,\infty) \end{pmatrix}}_{\text{sender}} \bullet \underbrace{\begin{pmatrix} (\_,0) & (B,1) & (\_,\infty) & (\_,\infty) \\ (\mathbf{D},\mathbf{3}) & (\_,0) & (\_,\infty) & (\_,\infty) \\ (A,1) & (\_,\infty) & (\_,0) & (D,1) \\ (C,2) & (\_,\infty) & (C,1) & (\_,0) \end{pmatrix}}_{\text{routing table}}$$

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

Sent Messages

- sending messages

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

- by distributivity

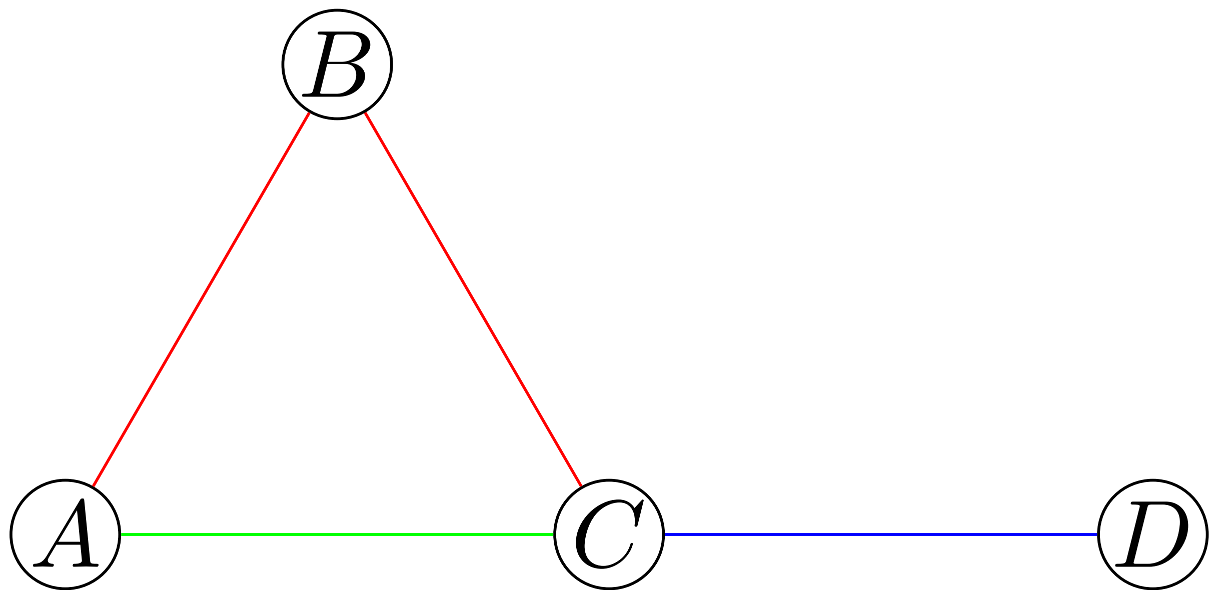
$$a \;\; + \;\; p \cdot b \cdot q \;\; + \;\; p \cdot b \cdot q \cdot c$$

  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)

- QUESTION: Can unicast modelled purely algebraically?

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 AODV
  - well known
  - IETF standard
- Extend formal methods to other protocols
  - OSLR, DYMO, ...
- Add further necessary concepts
  - time
  - probability (links, messurements)
  - define quality of protocols

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

NICTA

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

Different Network Layers