Routing I: Wireless Mesh Networks

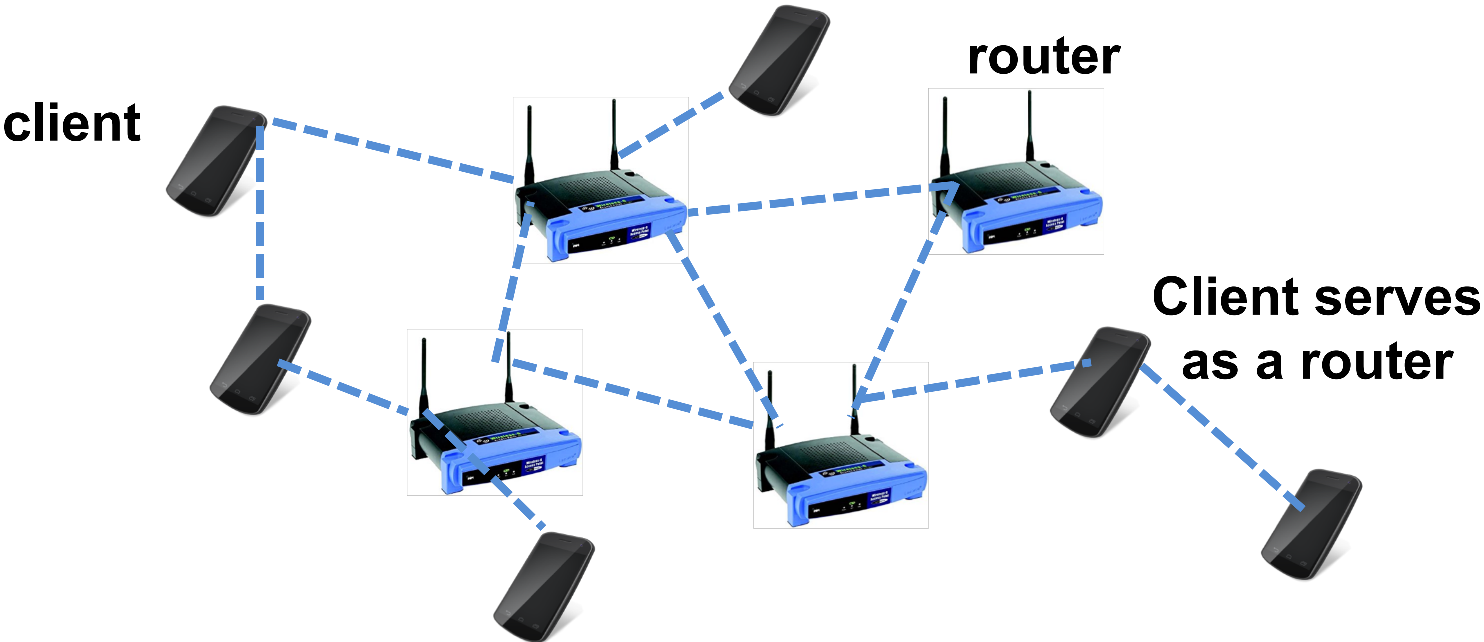
COS 463: Wireless Networks
Lecture 6

Kyle Jamieson

[Parts adapted from I. F. Akyildiz, B. Karp]

Wireless Mesh Networks: Motivation

- Most wireless network traffic goes through **APs**
- Mesh networks **remove this restriction**
  - **Multiple paths** between most pairs: **Mesh topology**



Today

1. **Distance Vector Routing**
   - **New node join**
   - Route changes
   - Broken link

2. Destination Sequenced Distance-Vector Routing (DSDV)

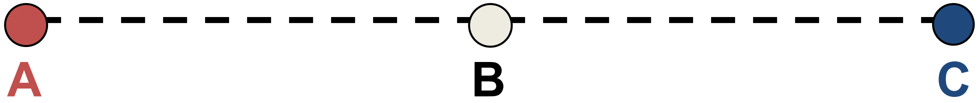
3. Dynamic Source Routing (DSR)

4. Roofnet: Quality-Aware Routing

Distance Vector Routing: Goal

- Every node maintains a ***routing table***
  - For each ***destination* node** in the mesh:
    - The **number of hops** to reach the destination (**metric**)
    - The **next** node on the path towards the destination

- All nodes **periodically, locally broadcast** routing table, **learn about every destination in network**



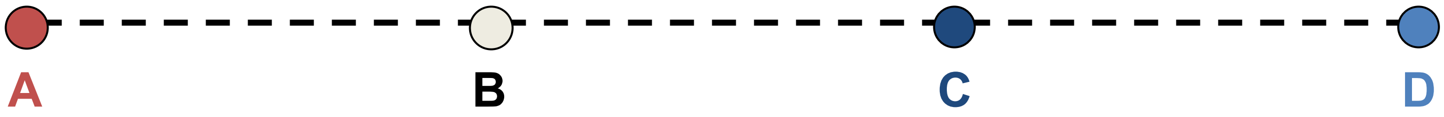
Dest.	Next	Metric
A	A	0
B	B	1
C	B	2

Dest.	Next	Metric
A	A	1
B	B	0
C	C	1

Dest.	Next	Metric
A	B	2
B	B	1
C	C	0

Distance Vector – New Node Join

- **D** joins the network



Dest.	Next	Metric
A	A	0
B	B	1
C	B	2

Dest.	Next	Metric
A	A	1
B	B	0
C	C	1

Dest.	Next	Metric
A	B	2
B	B	1
C	C	0

Distance Vector – New Node Join

- **D** joins the network
- **D's** broadcast first **updates C's table** with new entry for **D**

D's routing table

(D, D, 0)

Dest.	Next	Metric
A	A	0
B	B	1
C	B	2

Dest.	Next	Metric
A	A	1
B	B	0
C	C	1

Dest.	Next	Metric
A	B	2
B	B	1
C	C	0
D	D	1

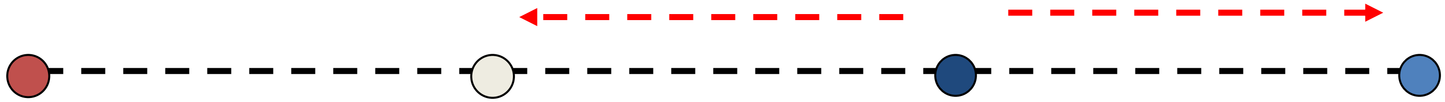
Distance Vector – New Node Join

- Now **C** broadcasts its routing table
  - – **B and D hear** and **add new entries, incrementing metric**

C's routing table

(A, B, 2)
(B, B, 1)
(C, C, 0)
(D, D, 1)

(A, B, 2)
(B, B, 1)
(C, C, 0)
(D, D, 1)



A

Dest.	Next	Metric
A	A	0
B	B	1
C	B	2

B

Dest.	Next	Metric
A	A	1
B	B	0
C	C	1
D	C	2

C

Dest.	Next	Metric
A	B	2
B	B	1
C	C	0
D	D	1

D

Dest.	Next	Metric
A	C	3
B	C	2
C	C	1
D	D	0

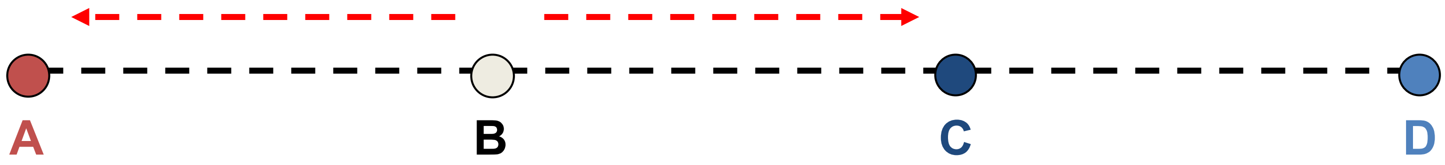
Distance Vector – New Node Join

- Now **B** broadcasts its routing table
  - **A and C hear** and **add new entries,** **if shorter route**

B's routing table

(A, A, 1)
(B, B, 0)
(C, C, 1)
(D, C, 2)

(A, A, 1)
(B, B, 0)
(C, C, 1)
(D, C, 2)



Dest.	Next	Metric
A	A	0
B	B	1
C	B	2
D	B	3

Dest.	Next	Metric
A	A	1
B	B	0
C	C	1
D	C	2

Dest.	Next	Metric
A	B	2
B	B	1
C	C	0
D	D	1

Dest.	Next	Metric
A	C	3
B	C	2
C	C	1
D	D	0

Today

1. **Distance Vector Routing**
   - New node join
   - **Route changes**
   - Broken link

2. Destination Sequenced Distance-Vector Routing (DSDV)

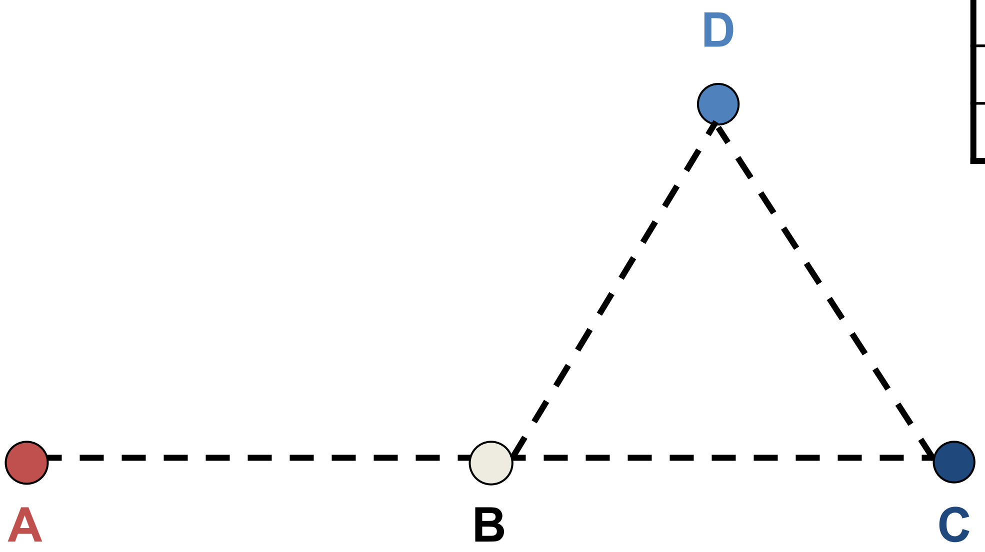
3. Dynamic Source Routing (DSR)

4. Roofnet: Quality-Aware Routing

Distance Vector – Route Change

- **D** moves to another place and broadcast its routing table

Dest.	Next	Metric
A	C	3
B	C	2
C	C	1
D	D	0



Dest.	Next	Metric
A	A	0
B	B	1
C	B	2
D	B	3

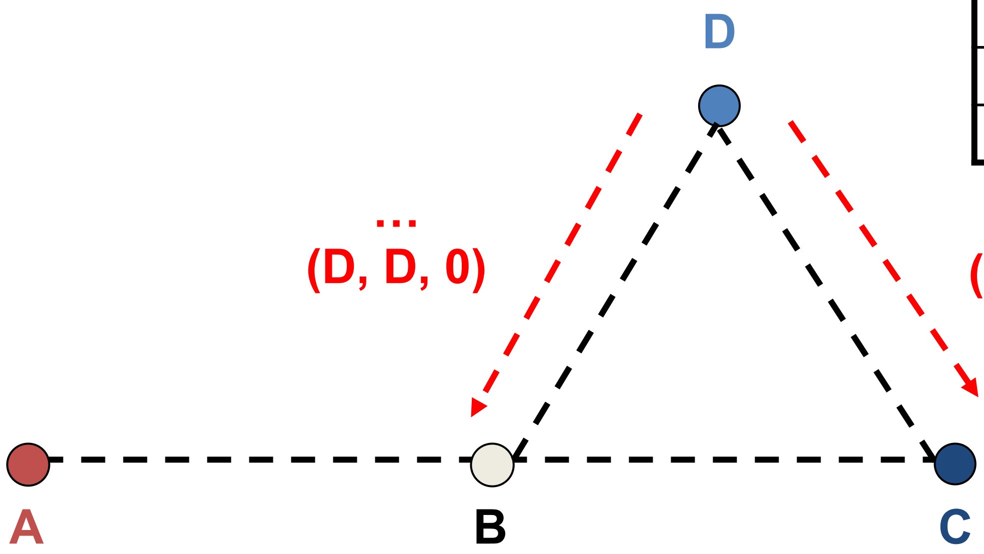
Dest.	Next	Metric
A	A	1
B	B	0
C	C	1
D	C	2

Dest.	Next	Metric
A	B	2
B	B	1
C	C	0
D	D	1

Distance Vector – Route Change

- **D** moves to another place and broadcast its routing table

Dest.	Next	Metric
A	C	3
B	C	2
C	C	1
D	D	0



Dest.	Next	Metric
A	A	0
B	B	1
C	B	2
D	B	3

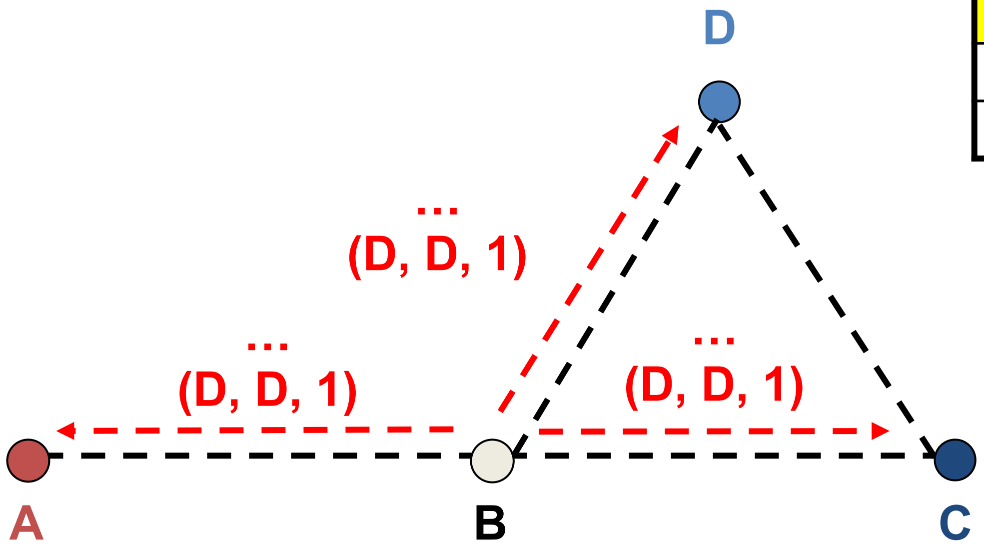
Dest.	Next	Metric
A	A	1
B	B	0
C	C	1
D	D	1

Dest.	Next	Metric
A	B	2
B	B	1
C	C	0
D	D	1

Distance Vector – Route Change

- **D** moves to another place and broadcast its routing table
- **B** broadcast its routing table

Dest.	Next	Metric
A	B	2
B	B	1
C	C	1
D	D	0



Dest.	Next	Metric
A	A	0
B	B	1
C	B	2
D	B	2

Dest.	Next	Metric
A	A	1
B	B	0
C	C	1
D	D	1

Dest.	Next	Metric
A	B	2
B	B	1
C	C	0
D	D	1

Today

1. **Distance Vector Routing**
   - New node join
   - Route changes
   - **Broken link**

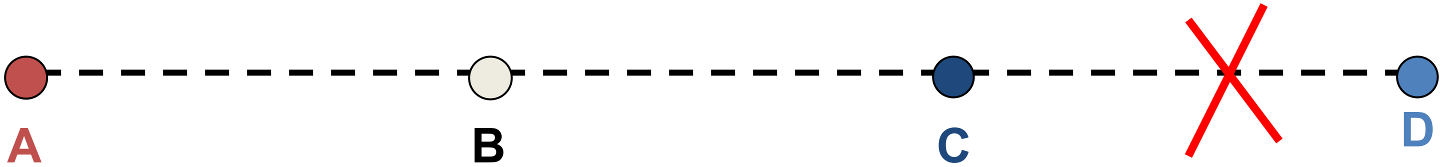
2. Destination Sequenced Distance-Vector Routing (DSDV)

3. Dynamic Source Routing (DSR)

4. Roofnet

Distance Vector – Broken Link

- **Suppose link C ←→ D breaks**



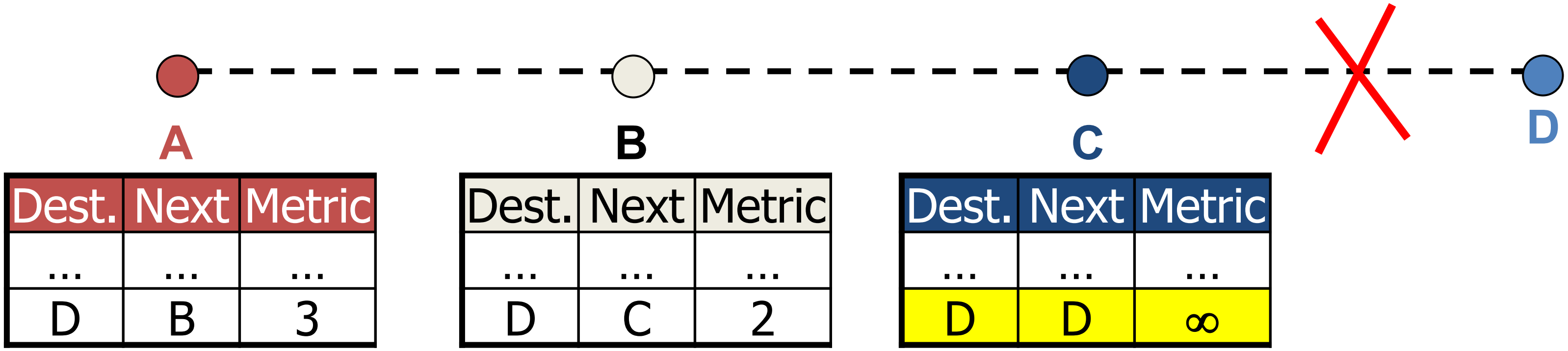
Dest.	Next	Metric
…	…	…
D	B	3

Dest.	Next	Metric
…	…	…
D	C	2

Dest.	Next	Metric
…	…	…
D	D	1

Distance Vector – Broken Link

1. **C hears no advertisement** from **D** for a ***timeout period***
   - **C** sets **D's** metric to ∞



A

Dest.	Next	Metric
…	…	…
D	B	3

B

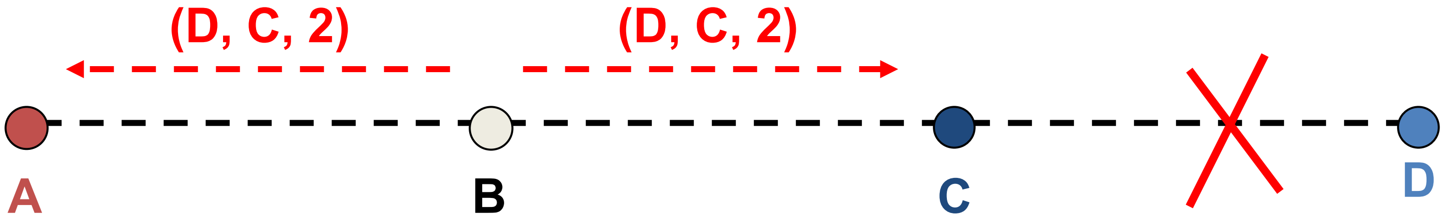
Dest.	Next	Metric
…	…	…
D	C	2

C

Dest.	Next	Metric
…	…	…
D	D	∞

Distance Vector – Broken Link

1. **C** sets **D’s** metric to ∞
2. **B** broadcasts its routing table
   - **C** **now accepts** **B’s** entry for **D** **(3 < ∞)**



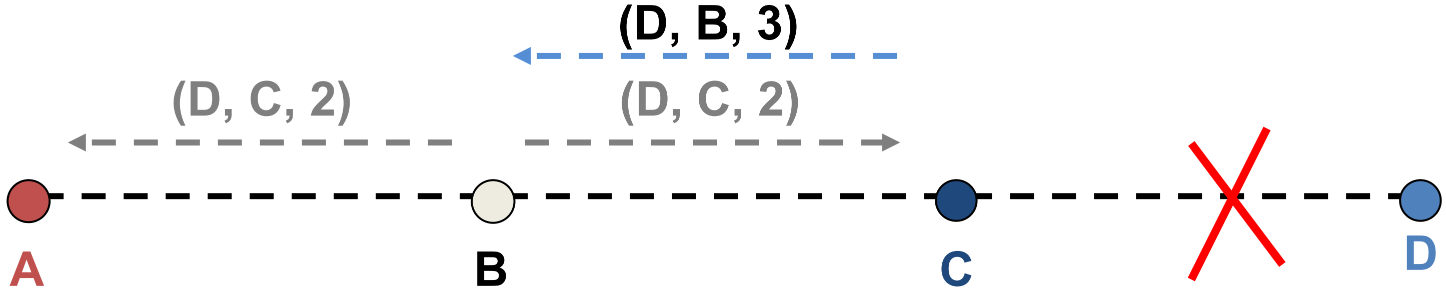
Dest.	Next	Metric
…	…	…
D	B	3

Dest.	Next	Metric
…	…	…
D	C	2

Dest.	Next	Metric
…	…	…
D	B	3

Broken Link: Counting to Infinity

1. **C** sets **D's** metric to ∞
2. **B** broadcasts its routing table
3. **C** broadcasts its routing table
   - **B accepts C's new metric** (previous next-hop: **C**)



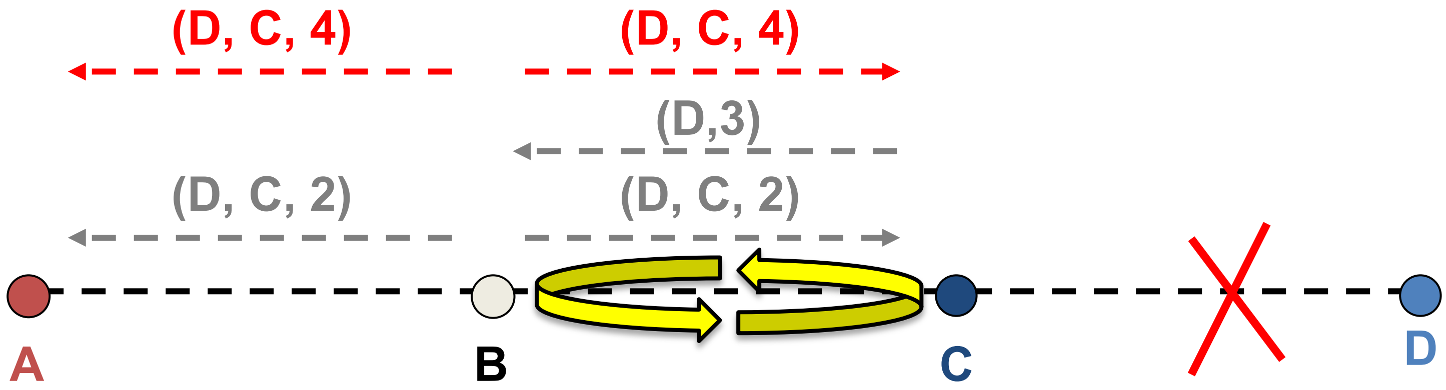
Dest.	Next	Metric
…	…	…
D	B	3

Dest.	Next	Metric
…	…	…
D	C	4

Dest.	Next	Metric
…	…	…
D	B	3

Broken Link: Counting to Infinity

1. **C** sets **D's** metric to ∞
2. **B** broadcasts its routing table
3. **C** broadcasts its routing table
4. **B** broadcasts its routing table
    - **A, C accept B's** new metric (previous next-hops: **B**)



Dest.	Next	Metric
…	…	…
D	B	5

Dest.	Next	Metric
…	…	…
D	C	4

Dest.	Next	Metric
…	…	…
D	B	5

Today

1. Distance Vector Routing

2. **Destination Sequenced Distance-Vector Routing (DSDV)**
   - **New node join**
   - Broken link
   - Route advertisement

3. Dynamic Source Routing (DSR)

4. Roofnet: Quality-Aware Routing

Destination Sequenced Distance-Vector (DSDV) Routing

- Guarantees **loop freeness**

– **New routing table information:** ***Sequence number***

1. **Per-destination** information
2. **Originated** by destination
3. **Included** in routing advertisements

Destination	Next	Metric	Seq. Nr
A	A	0	550
B	B	1	102
C	B	3	588
D	B	4	312

DSDV: Route Advertisement Rule

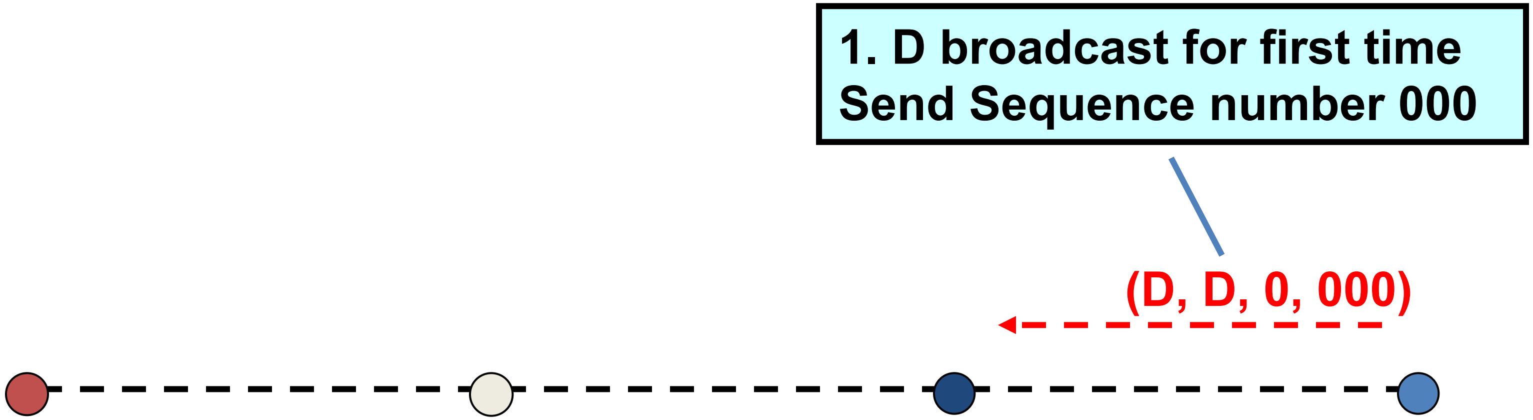
- **Rules to set sequence number:**

- Just before **node N**’s broadcast advertisement:
  - **Node N** sets:
    - **Seq(N) ← Seq(N) + 2**

- **Node N** thinks **neighbor P** is **no longer directly reachable**
  - **Node N** sets:
    - **Seq(P) ← Seq(P) + 1**
    - **Metric(P) ←** ∞

DSDV – New Node

- **D joins the network**
- **D's** broadcast first **updates C's table** with new entry **for D**



1. D broadcast for first time
Send Sequence number 000

(D, D, 0, 000)

Dest.	Next	Metric	Seq.
A	A	0	550
B	B	1	104
C	B	2	590

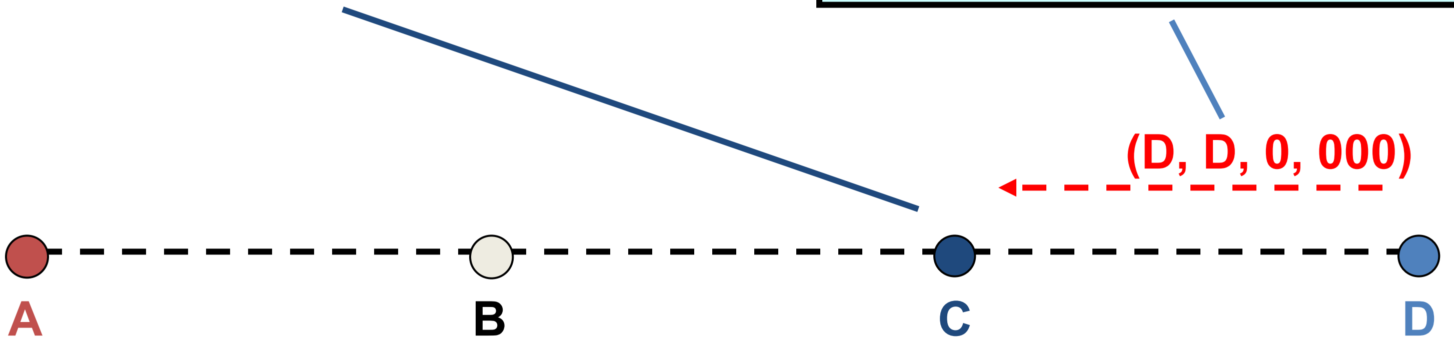
Dest.	Next	Metric	Seq.
A	A	1	550
B	B	0	104
C	C	1	590

Dest.	Next	Metric	Seq.
A	B	2	550
B	B	1	104
C	C	0	590

DSDV – New Node

2.1: Insert entry for D with same sequence number 000
2.2: Triggered broadcast of its own table

1. D broadcast for first time
Send Sequence number 000

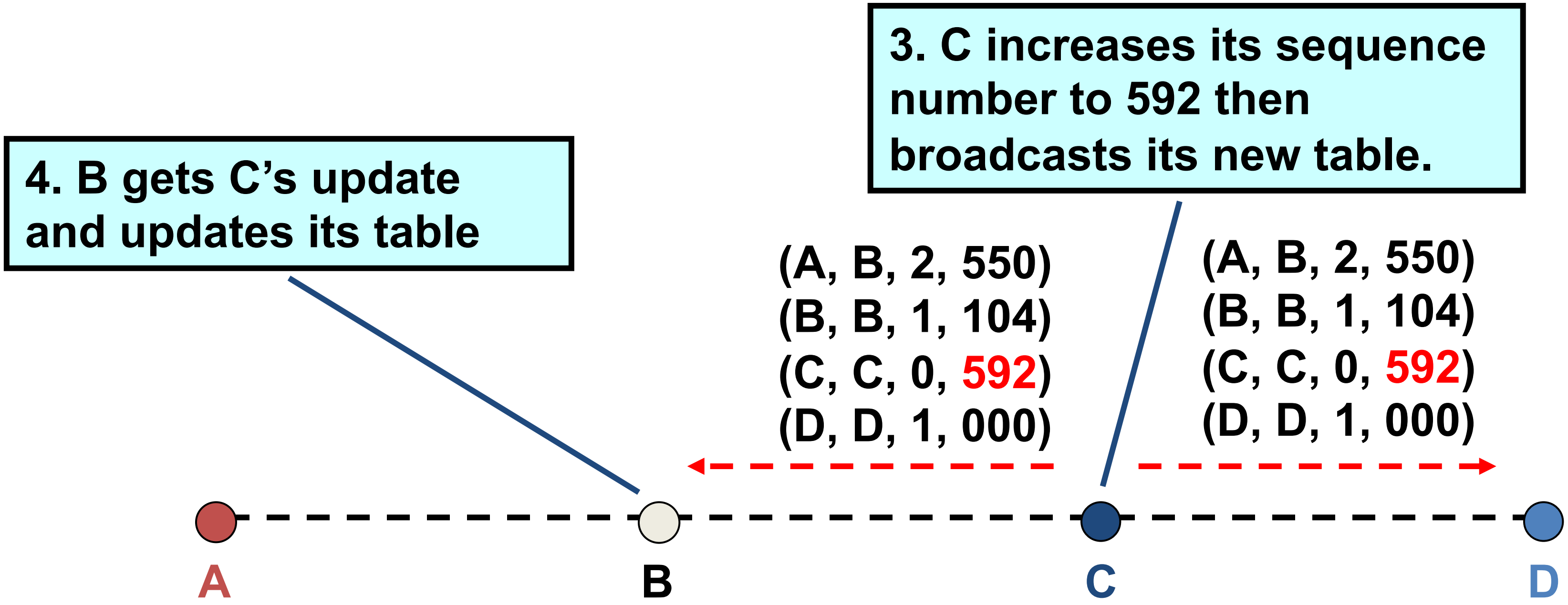


Dest.	Next	Metric	Seq.
A	A	0	550
B	B	1	104
C	B	2	590

Dest.	Next	Metric	Seq.
A	A	1	550
B	B	0	104
C	C	1	590

Dest.	Next	Metric	Seq.
A	B	2	550
B	B	1	104
C	C	0	590
D	D	1	000

DSDV – New Node



4. B gets C's update and updates its table

3. C increases its sequence number to 592 then broadcasts its new table.

(A, B, 2, 550)
(B, B, 1, 104)
(C, C, 0, 592)
(D, D, 1, 000)

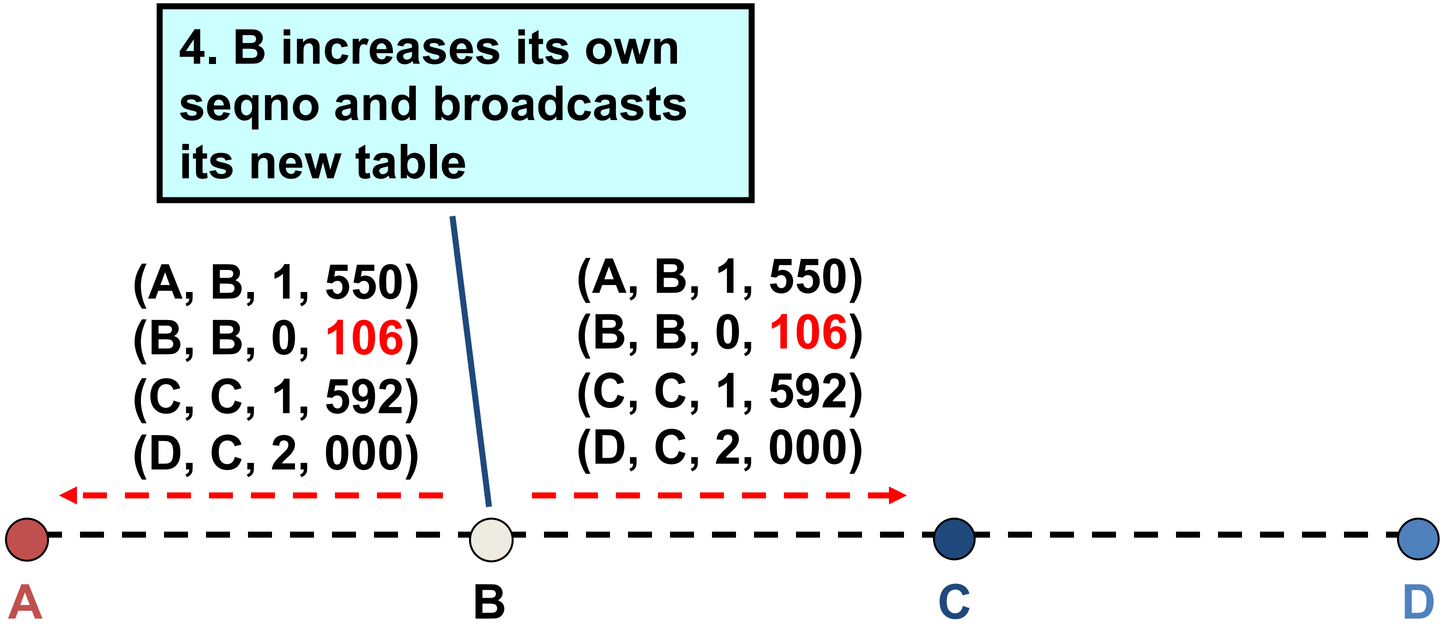
(A, B, 2, 550)
(B, B, 1, 104)
(C, C, 0, 592)
(D, D, 1, 000)

Dest.	Next	Metric	Seq.
A	A	0	550
B	B	1	104
C	B	2	590

Dest.	Next	Metric	Seq.
A	A	1	550
B	B	0	104
C	C	1	592
D	C	2	000

Dest.	Next	Metric	Seq.
A	B	2	550
B	B	1	104
C	C	0	592
D	D	1	000

DSDV – New Node



4. B increases its own seqno and broadcasts its new table

(A, B, 1, 550)
(B, B, 0, 106)
(C, C, 1, 592)
(D, C, 2, 000)

(A, B, 1, 550)
(B, B, 0, 106)
(C, C, 1, 592)
(D, C, 2, 000)

Dest.	Next	Metric	Seq.
A	A	0	550
B	B	1	106
C	B	2	592
D	B	3	000

Dest.	Next	Metric	Seq.
A	A	1	550
B	B	0	106
C	C	1	592
D	C	2	000

Dest.	Next	Metric	Seq.
A	B	2	550
B	B	1	106
C	C	0	592
D	D	1	000

Today

1. Distance Vector Routing

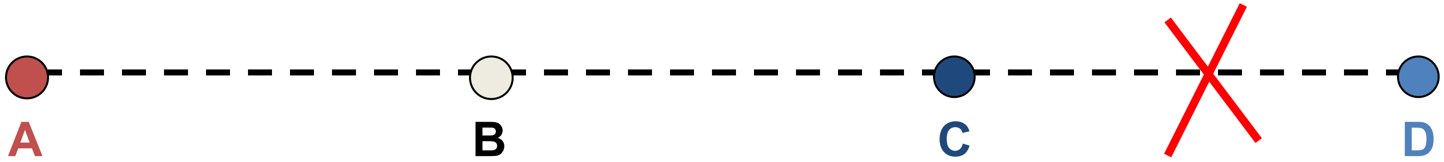
2. **Destination Sequenced Distance-Vector Routing (DSDV)**
   - New node join
   - **Broken link**
   - Route advertisement

3. Dynamic Source Routing (DSR)

4. Roofnet: Quality-Aware Routing

DSDV – Broken Link

- Suppose link **C ←→ D** **breaks**



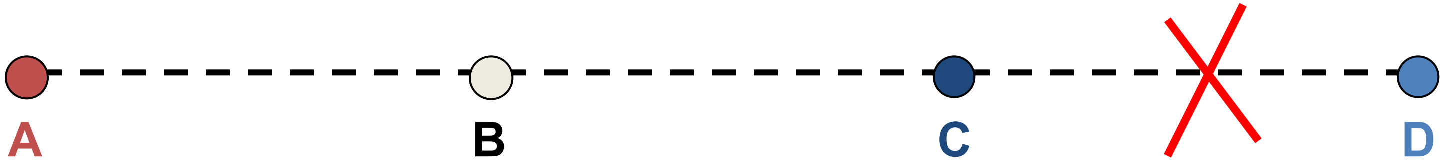
Dest.	Next	Metric	Seq.
A	A	0	550
B	B	1	106
C	B	2	592
D	B	3	000

Dest.	Next	Metric	Seq.
A	A	1	550
B	B	0	106
C	C	1	592
D	C	2	000

Dest.	Next	Metric	Seq.
A	B	2	550
B	B	1	106
C	C	0	592
D	D	1	000

DSDV – Broken Link

1. Node C detects broken Link:
→ Increase Seq. No. by 1
(only case where not the destination sets the sequence number → odd number)



Dest.	Next	Metric	Seq.
A	A	0	550
B	B	1	106
C	B	2	592
D	B	3	000

Dest.	Next	Metric	Seq.
A	A	1	550
B	B	0	106
C	C	1	592
D	C	2	000

Dest.	Next	Metric	Seq.
A	B	2	550
B	B	1	106
C	C	0	592
D	D	∞	001

DSDV: Routing Table Update Rule

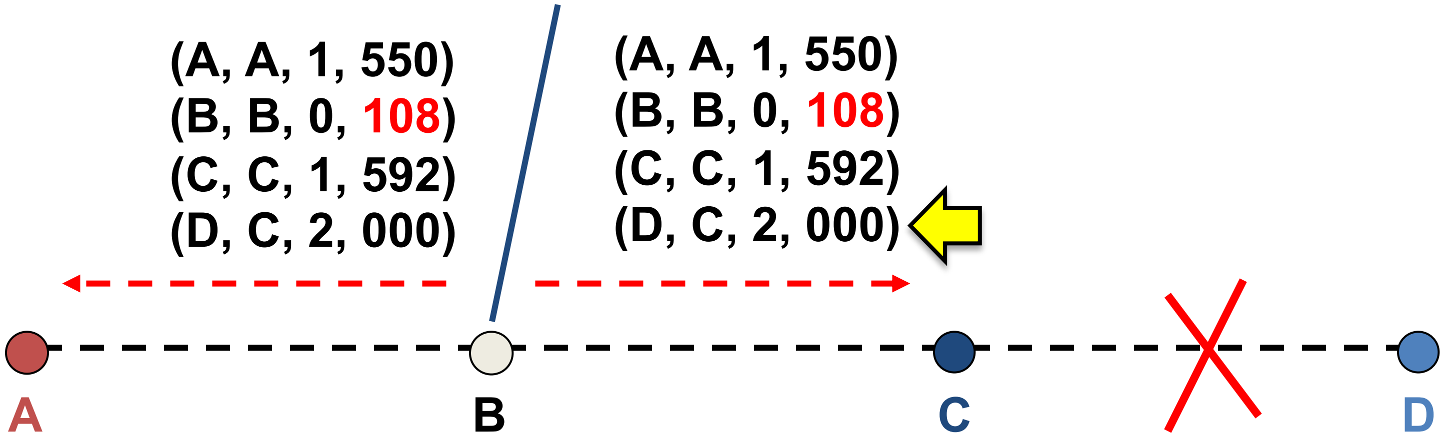
- **Rules to update routing table entry:**

- **Node N** gets routing advertisement from neighbor **Node P:**
  - Update routing table entry for **node E when:**
    - **Seq(E) in P's advertisement > Seq(E)** in **N's table**

DSDV – Broken Link

- **B** next broadcasts its routing table

> - **No effect** on **C's entry for D** **(because 001 > 000)**
> - **No loop → no count to infinity**



(A, A, 1, 550)
(B, B, 0, 108)
(C, C, 1, 592)
(D, C, 2, 000)

(A, A, 1, 550)
(B, B, 0, 108)
(C, C, 1, 592)
(D, C, 2, 000)

Dest.	Next	Metric	Seq.
A	A	0	550
B	B	1	108
C	B	2	592
D	B	3	000

Dest.	Next	Metric	Seq.
A	A	1	550
B	B	0	108
C	C	1	592
D	C	2	000

Dest.	Next	Metric	Seq.
A	B	2	550
B	B	1	108
C	C	0	592
D	**D**	∞	**001**

Today

1. Distance Vector Routing

2. **Destination Sequenced Distance-Vector Routing (DSDV)**
    - New node join
    - Broken link
    - **Route advertisement**

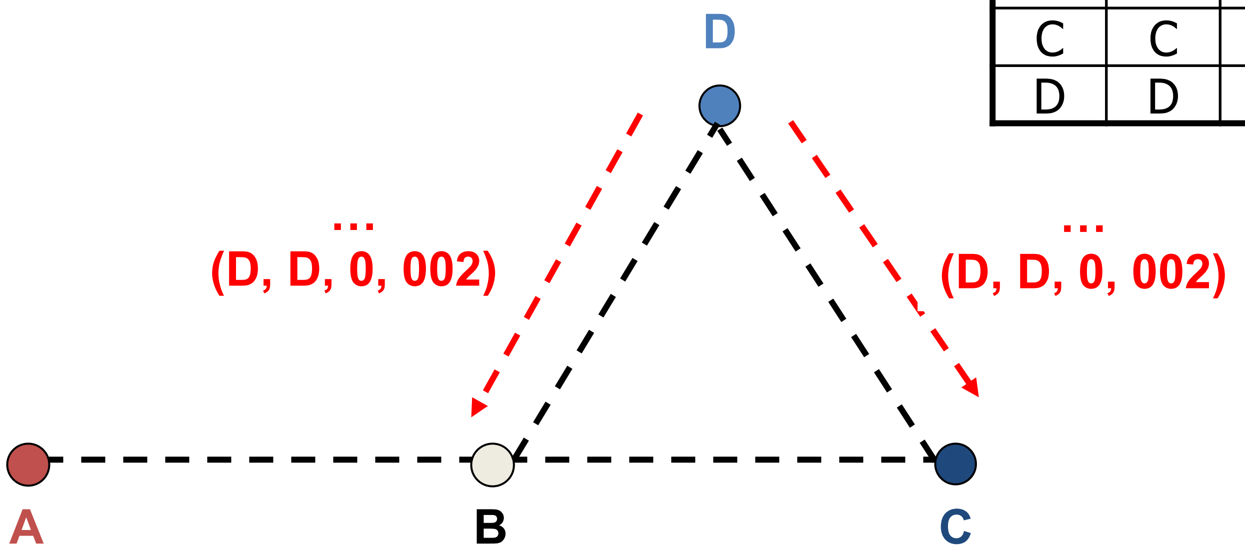
3. Dynamic Source Routing (DSR)

4. Roofnet

Distance Vector – Route Advertisement

- **D** moves to another place and broadcasts its routing table

Dest.	Next	Metric	Seq.
A	C	3	550
B	C	2	108
C	C	1	592
D	D	0	002



Dest.	Next	Metric	Seq.
A	A	0	550
B	B	1	108
C	B	2	592
D	B	3	000

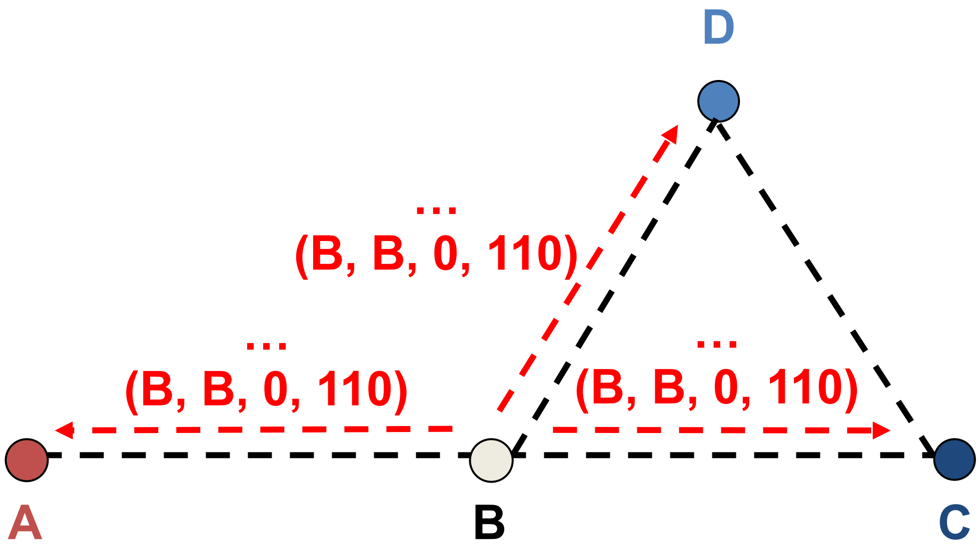
Dest.	Next	Metric	Seq.
A	A	1	550
B	B	0	108
C	C	1	592
D	D	1	002

Dest.	Next	Metric	Seq.
A	B	2	550
B	B	1	108
C	C	0	592
D	D	1	002

Distance Vector – Route Advertisement

- **D** moves to another place and broadcasts its routing table
- **B** broadcasts its routing table

Dest.	Next	Metric	Seq.
A	B	2	550
B	B	1	110
C	C	1	592
D	D	0	002



Dest.	Next	Metric	Seq.
A	A	0	550
B	B	1	110
C	B	2	592
D	B	2	002

Dest.	Next	Metric	Seq.
A	A	1	550
B	B	0	110
C	C	1	592
D	D	1	002

Dest.	Next	Metric	Seq.
A	B	2	550
B	B	1	110
C	C	0	592
D	D	1	002

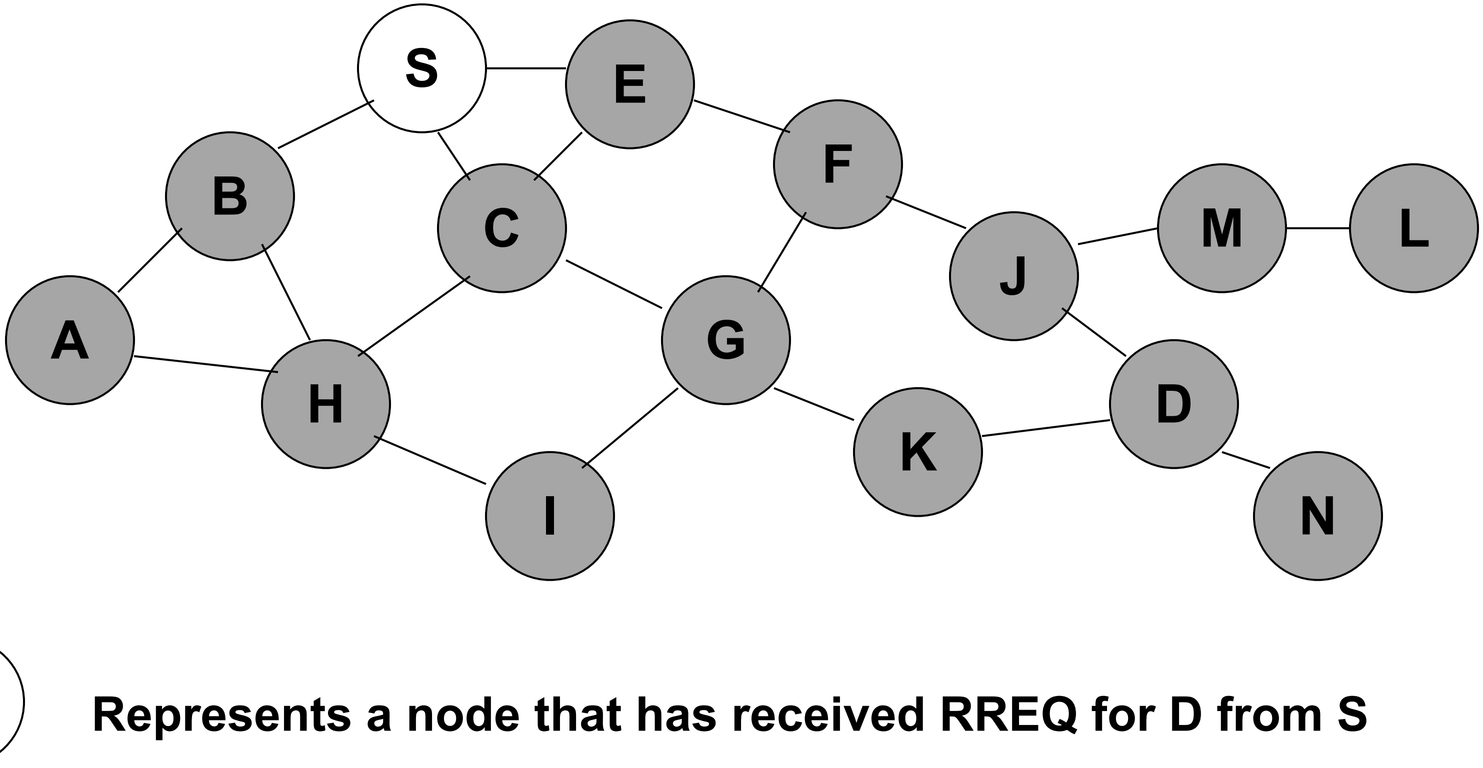
Today

1. Distance Vector Routing
2. Destination Sequenced Distance-Vector Routing (DSDV)
3. **Dynamic Source Routing (DSR)**
4. Roofnet: Quality-Aware Routing

Dynamic Source Routing (DSR)

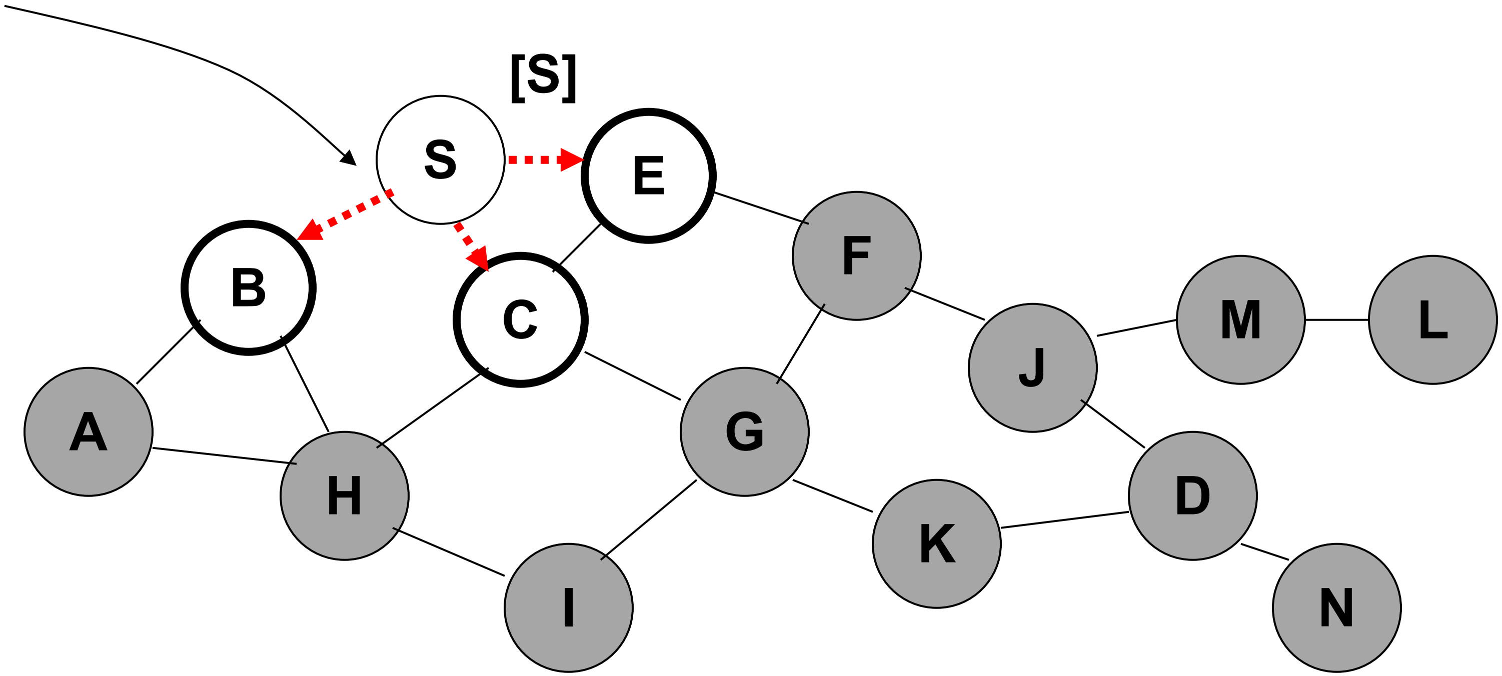
- **No periodic “beaconing”** from all nodes
- When node **S** wants to send a packet to node **D** (but doesn’t know a route to D), **S** initiates a **route discovery**
- **S** **network-floods** a ***Route Request (RREQ)***
  - Each node **appends its own id when forwarding RREQ**

Route Discovery in DSR



Represents a node that has received RREQ for D from S

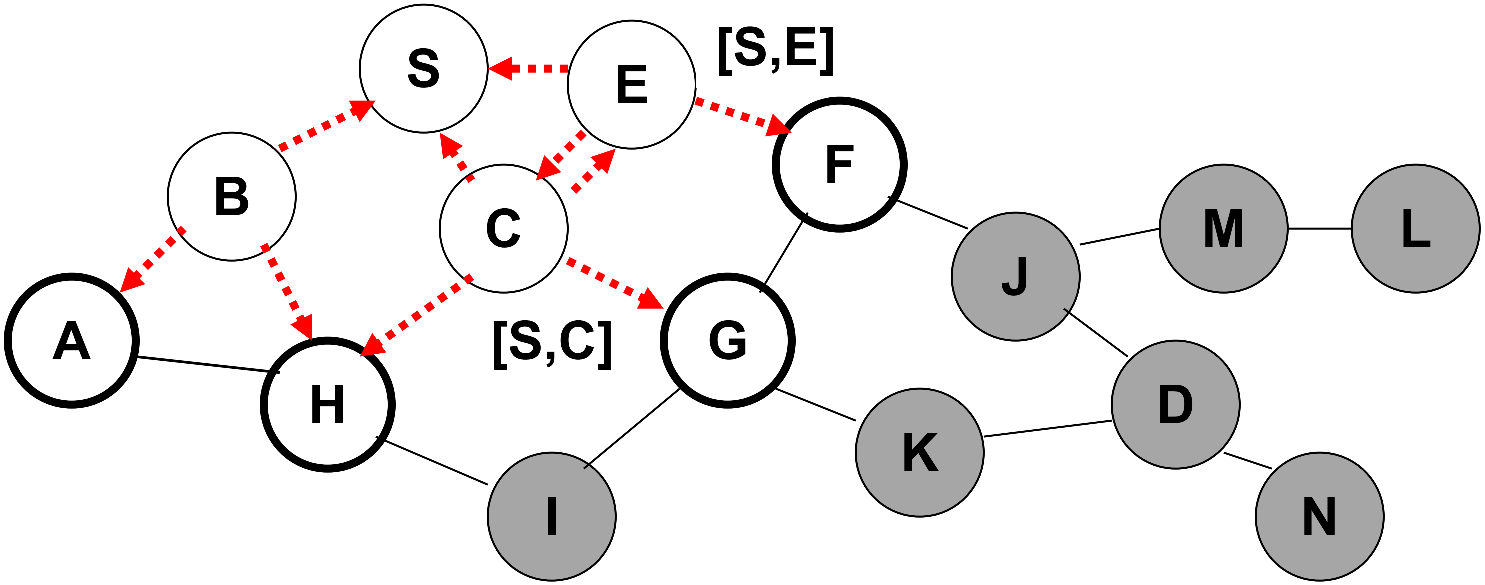
Route Discovery in DSR


Broadcast transmission

Represents transmission of RREQ

[X,Y] Represents list of identifiers appended to RREQ

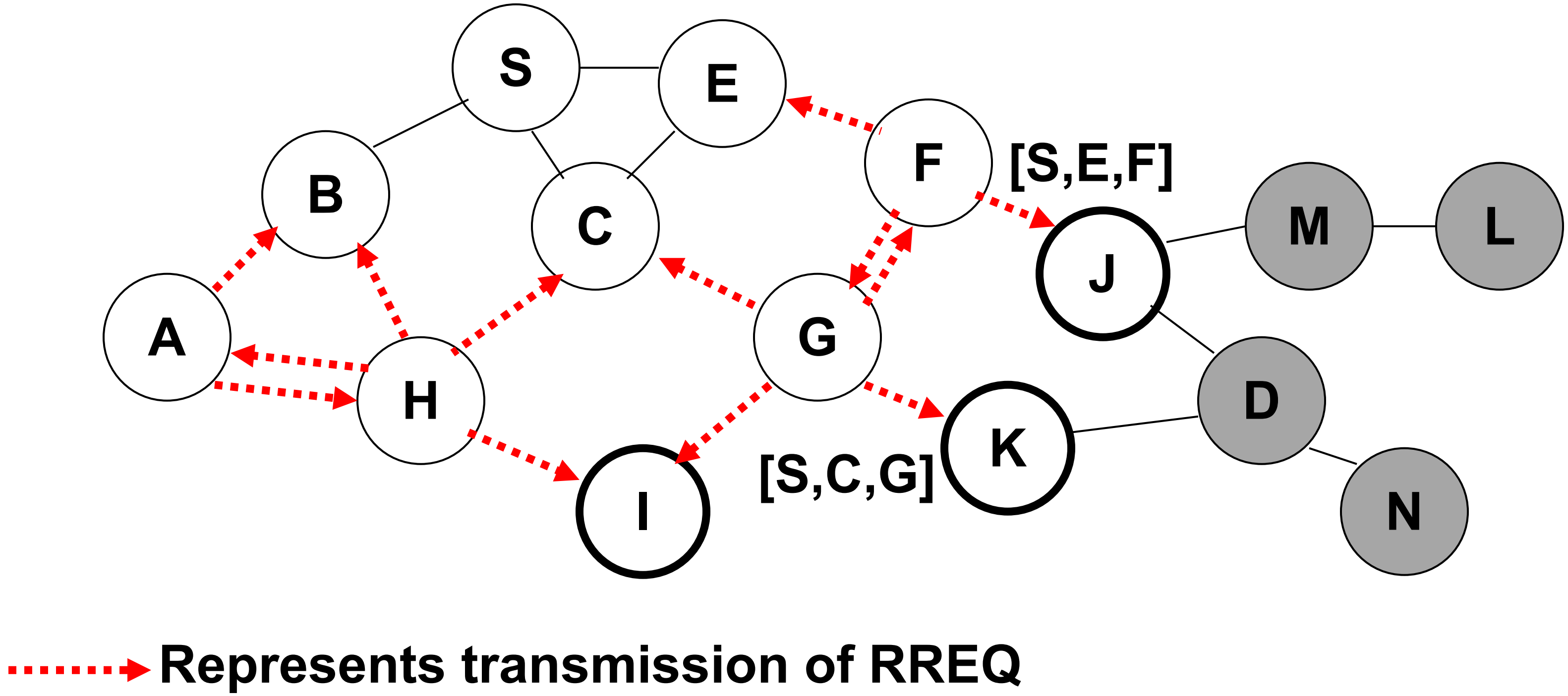
Route Discovery in DSR



Represents transmission of RREQ

[X,Y] Represents list of identifiers appended to RREQ

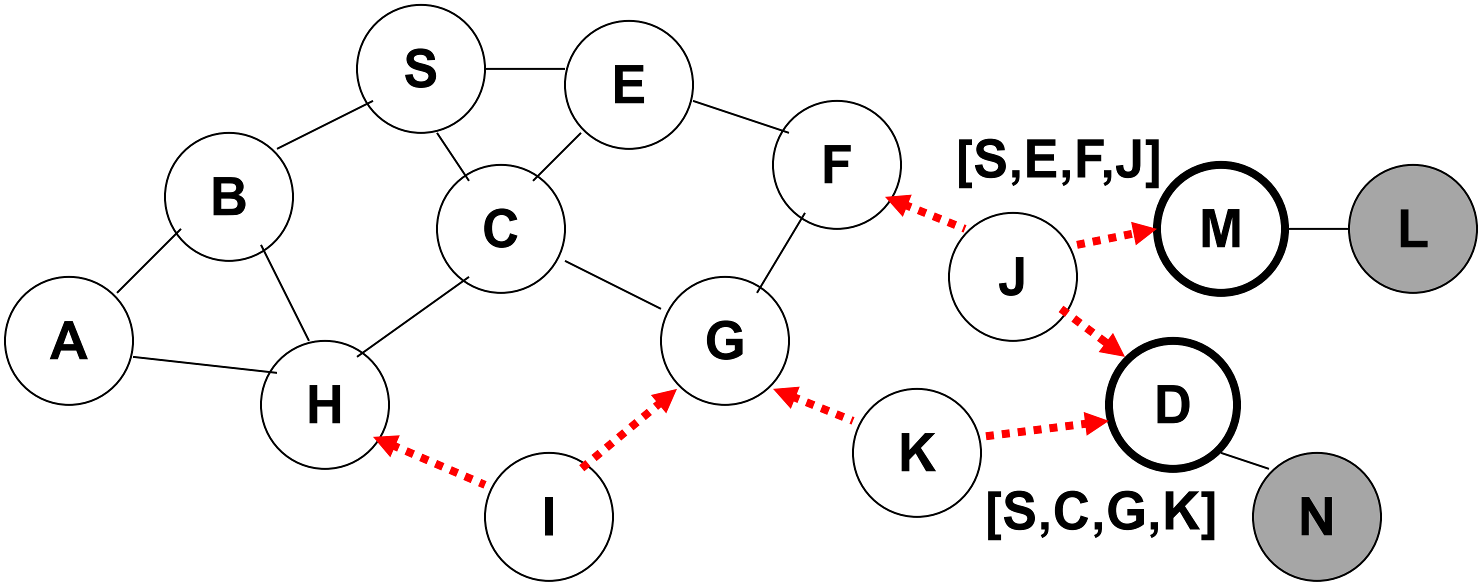
Route Discovery in DSR



Represents transmission of RREQ

- Node C receives RREQ from G and H, but does not forward it again, because node C has already forwarded RREQ once

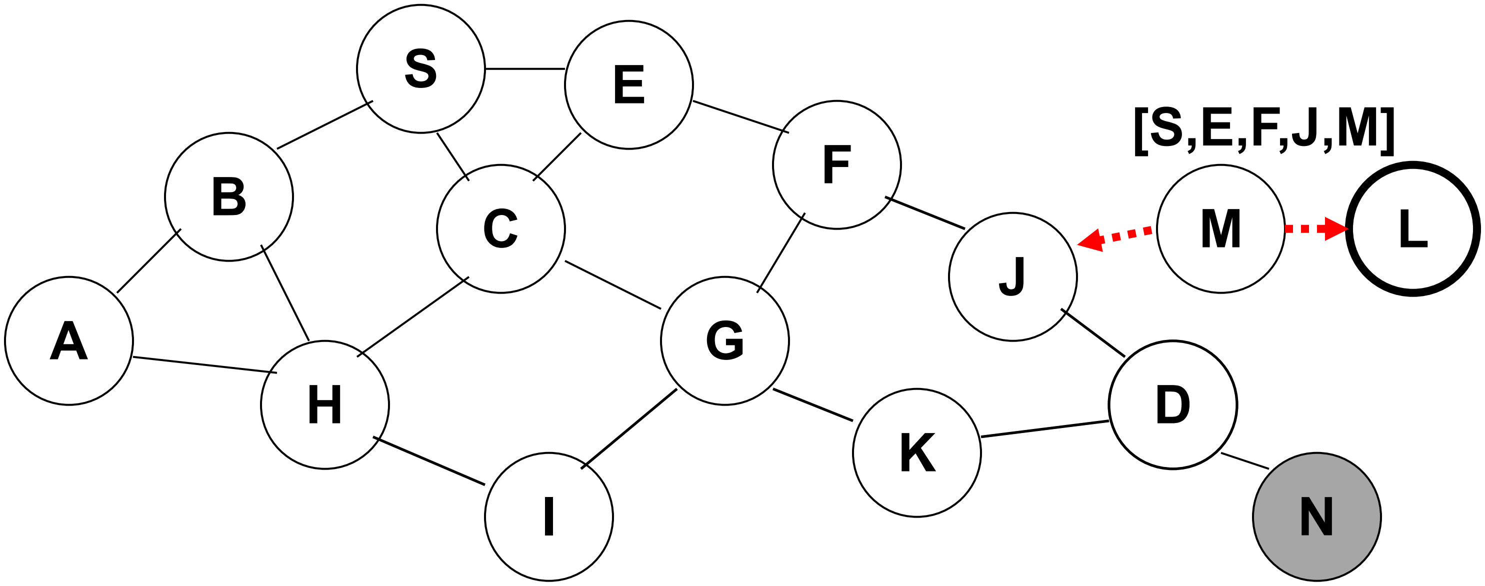
Route Discovery in DSR




Represents transmission of RREQ

- Nodes J and K both broadcast RREQ to node D
- Since nodes J and K are hidden from each other, their transmissions may collide

Route Discovery in DSR

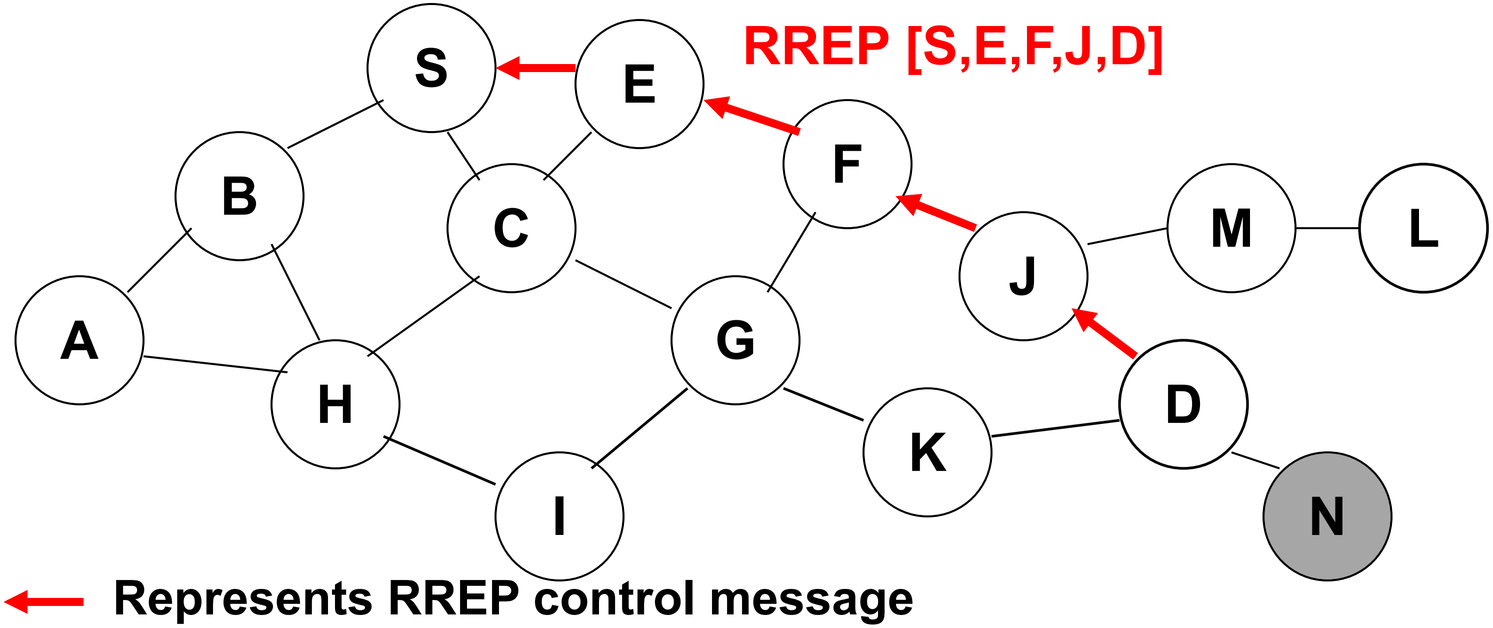


Represents transmission of RREQ

- Node D does not forward RREQ, because node D is the intended target of the route discovery

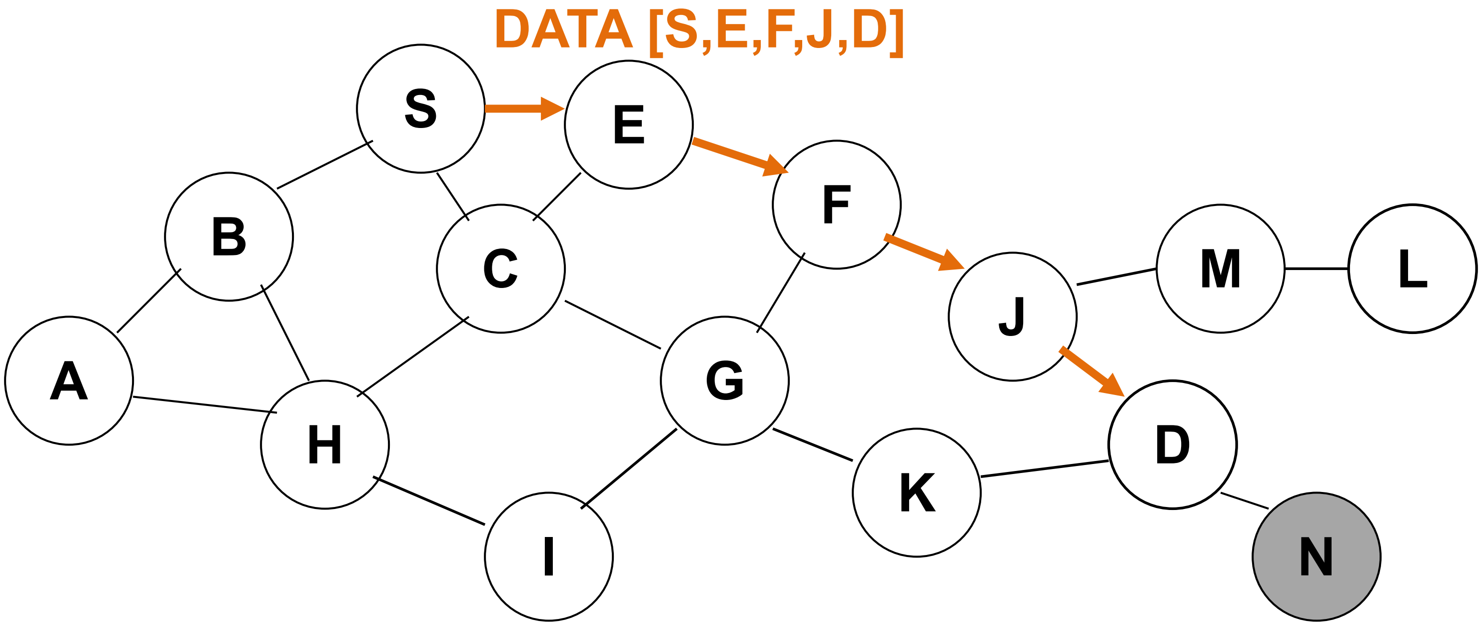
Route Reply in DSR

- On receiving **first** RREQ, **D** sends a ***Route Reply*** **(RREP)**
  - RREP sent on route obtained by **reversing** the route in the received RREQ
  - RREP **includes the route** from **S** to **D** over which D received the RREQ



Dynamic Source Routing (DSR)

- On receiving RREP, **S** caches route included therein
- When **S** sends a data packet to **D,** includes entire route in packet header
- Intermediate nodes use the **source route** included in packet to determine to whom packet should be forwarded



Today

1. Distance Vector Routing

2. Destination Sequenced Distance-Vector Routing (DSDV)

3. Dynamic Source Routing (DSR)

4. **Roofnet: Quality-Aware Routing**
   - Wireless mesh link measurements
   - Routing and bit rate selection
   - End-to-end performance evaluation

Context, *ca.* 2000-2005

- **Mobile ad hoc networking** research
  - Mobile, hence highly dynamic topologies
  - Chief metrics: routing protocol overhead, packet delivery success rate, hop count
  - Largely evaluated in simulation

- **Roofnet,** a real mesh network **deployment**
  - Fixed, PC-class nodes
  - Motivation: shared Internet access in community
  - Chief metric: TCP throughput
  - "Test of time" system, led to **Cisco Meraki**

Roofnet: Design Choices

1. Volunteer users host nodes at home
   - Open participation **without central planning**
   - No central **control** over topology

2. Omnidirectional rather than directional antennas
   - **Ease of installation**: no choice of neighbors/aiming
   - Links **interfere**, likely **low quality**

3. Multi-hop routing (not single-hop hot spots)
   - Improved **coverage** (path diversity)
   - Must build a routing protocol

4. Goal: high TCP throughput

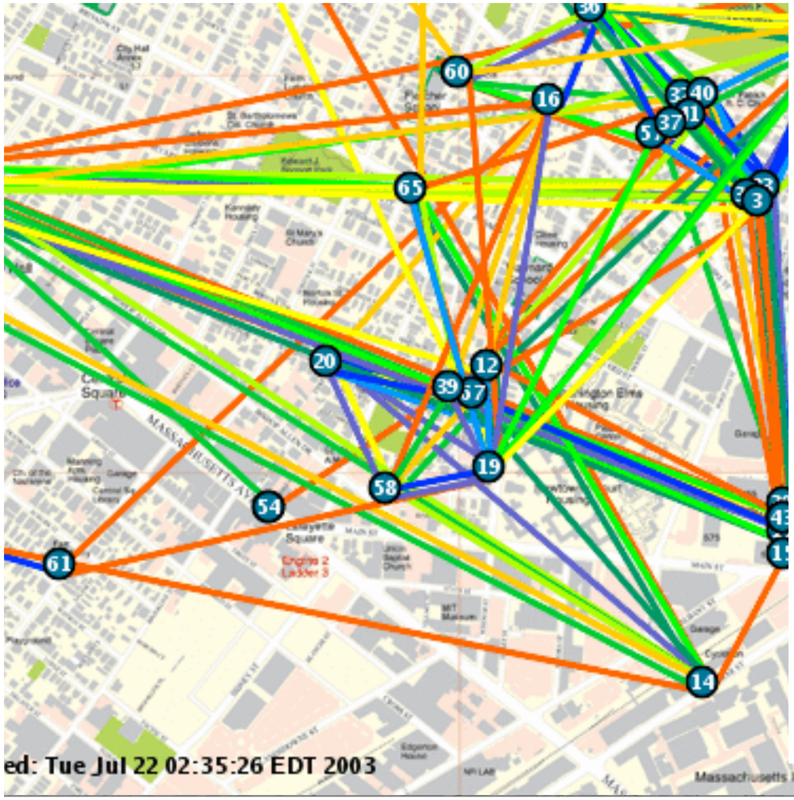
Roofnet: Goals and non-goals

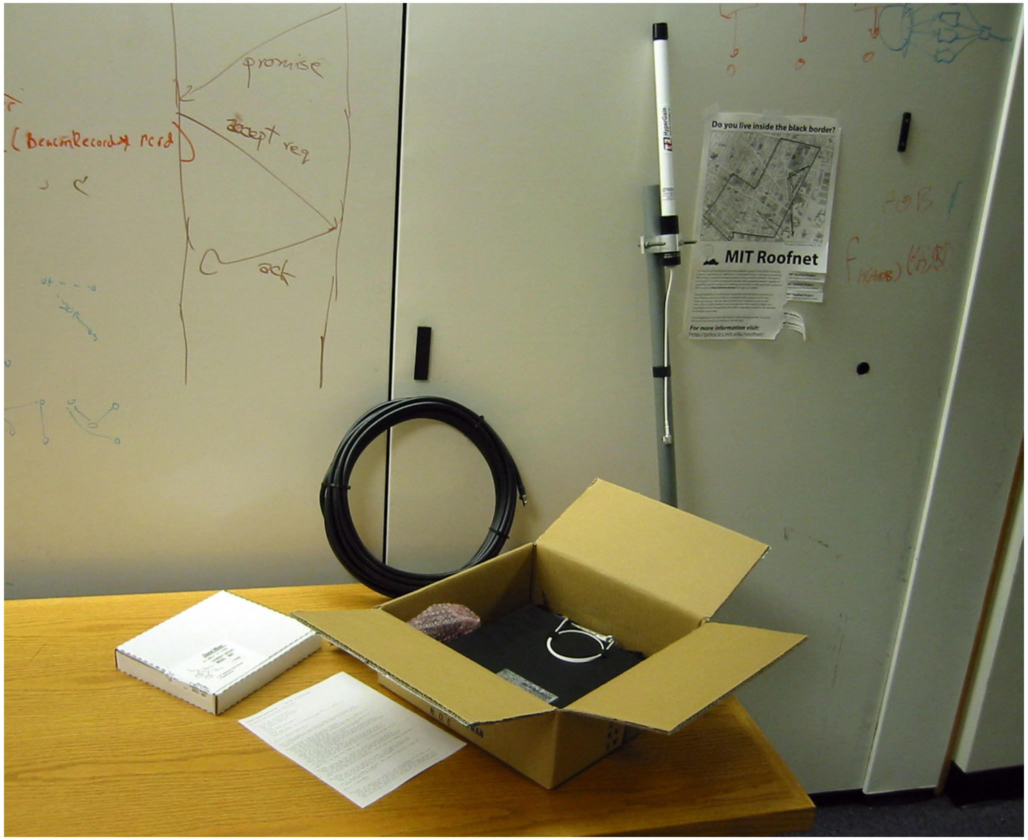
- Each part of the mesh architecture had been **previously examined** in isolation

- **Paper contribution:** A systematic evaluation of whether architecture can achieve goal of providing Internet access

- **Stated non-goals for paper:**
  - Throughput of multiple concurrent flows
  - Scalability in number of nodes
  - Design of routing protocols

Roofnet deployment





- Each node: PC, 802.11b card, roof-mounted omni antenna

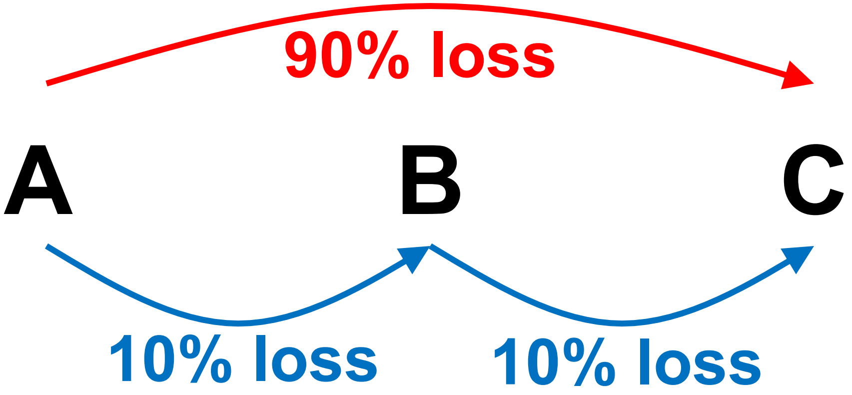
Hardware design

- PC Ethernet interface provides wired Internet for user

- Omnidirectional antenna in **azimuthal** direction
  - 3 dB **vertical** beam width of 20 degrees
    - Wide beam sacrifices gain but **removes the need** for perfect vertical **antenna orientation**

- **802.11b** radios (*Intersil Prism 2.5* chipset)
  - 200 mW transmit power
  - All share same 802.11 channel (frequency)

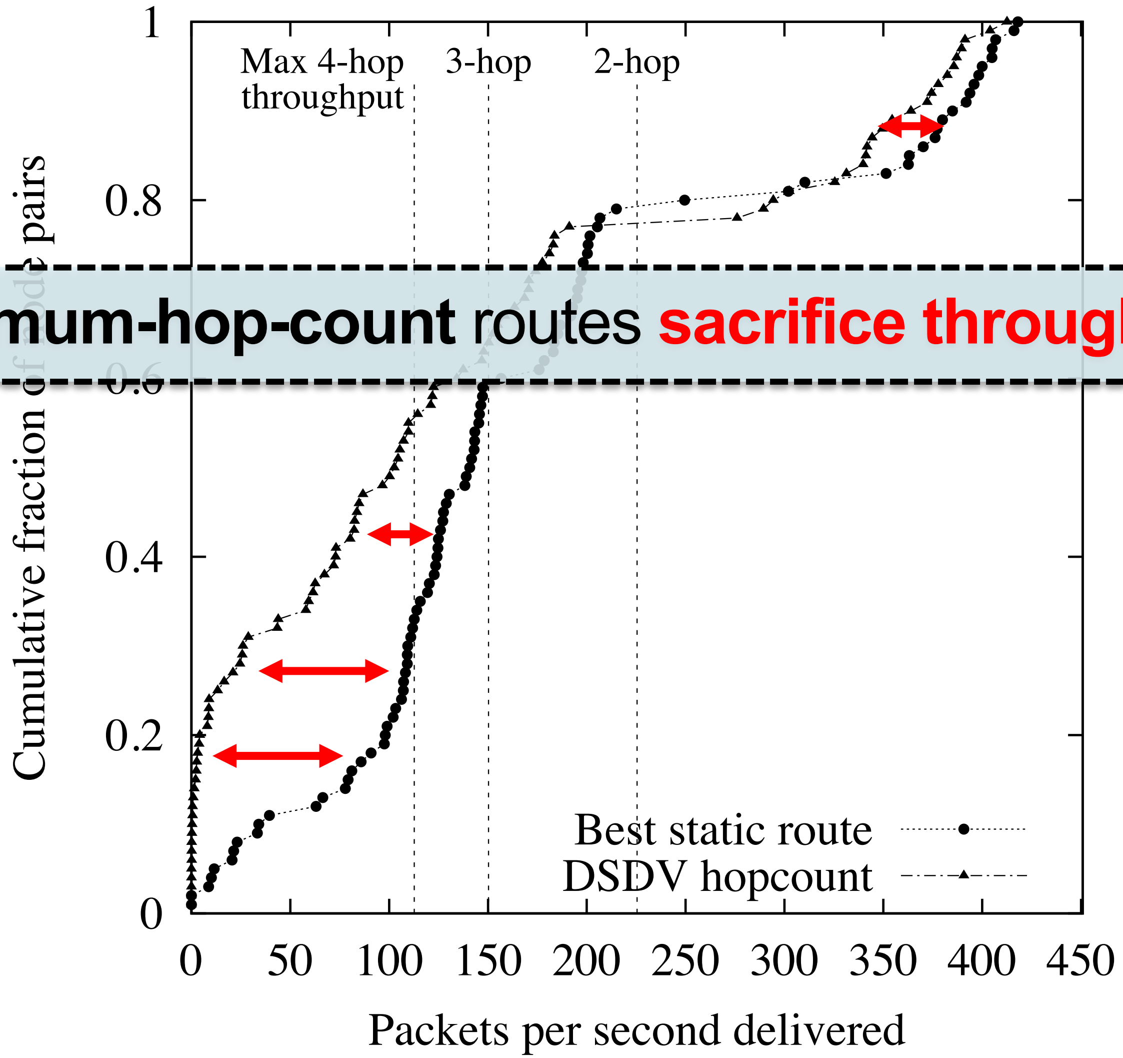
Example: Varying link loss rates



- A → C: 1 hop; **high loss**
- A → B → C: 2 hops; **lower loss**

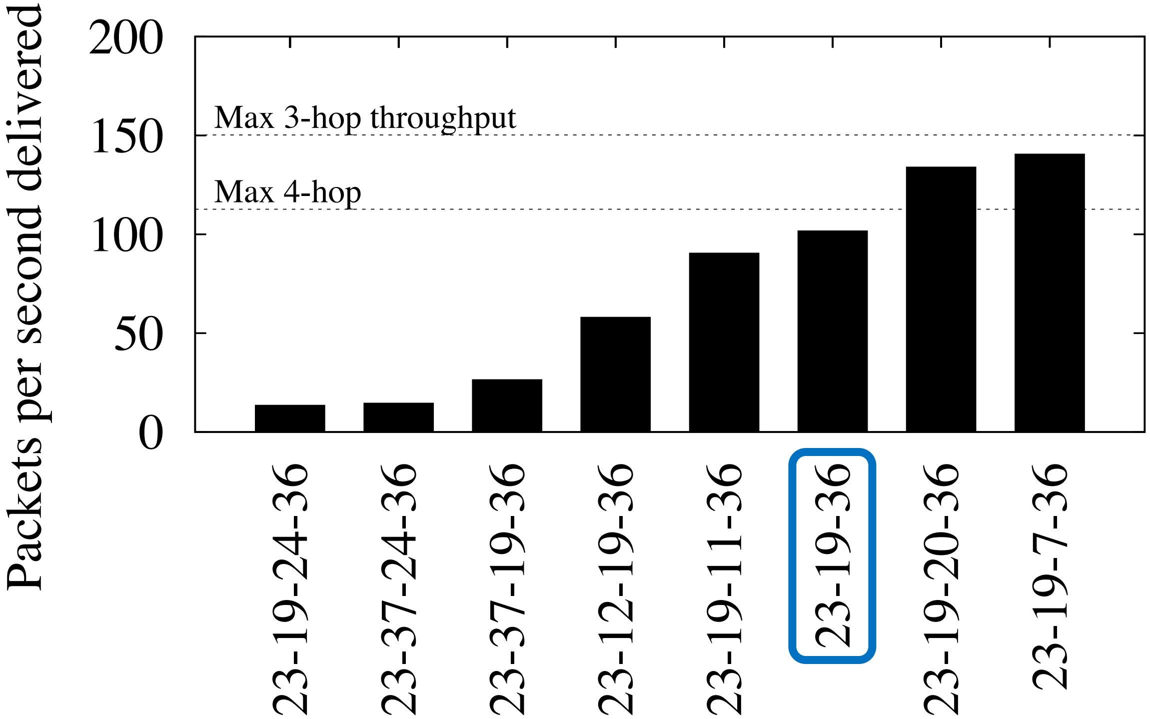
- But **does this happen** in practice?

Hop count and throughput (1)



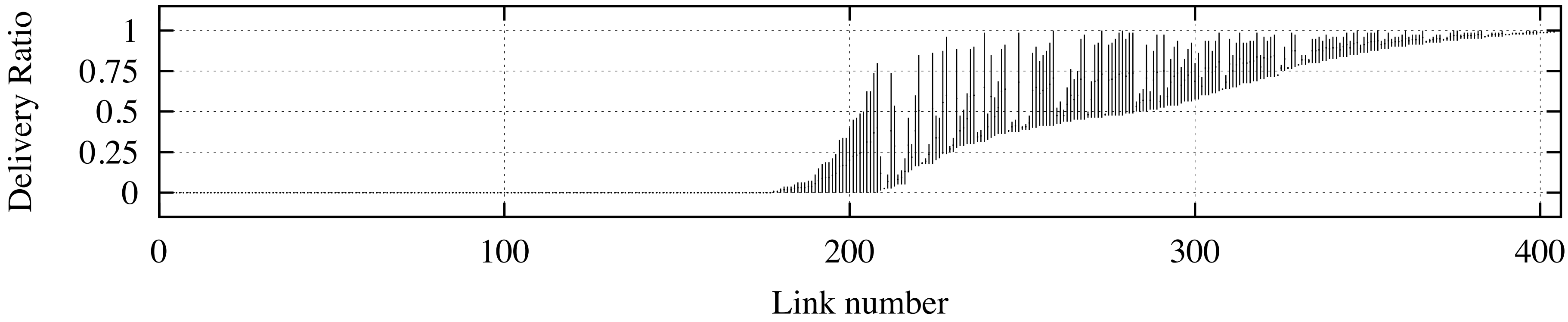
Minimum-hop-count routes **sacrifice throughput**

Hop count and throughput



- Two-hop path is **suboptimal**
- Some **3-hop paths better, some worse** than 2-hop

Link loss is high and asymmetric



- Vertical bar ends = loss rate on 1 link in each direction
- Many links **asymmetric** and **very lossy in ≥ 1 way**
- **Wide range** of loss rates

Routing protocol: Srcr

- Each link has an associated ***metric*** (not necessarily 1!)
- Data packets contain **source routes**

- Nodes keep **database of link metrics**
  - Nodes write current metric into source route of all forwarded packets
  - **DSR-like:** Nodes flood ***route queries*** when they can't find a route; queries accumulate link metrics
    - Route queries contain route from requesting node
  - Nodes cache overheard link metrics

- **Dijkstra's algorithm** computes source routes

Link metric: Strawmen

- *Discard links with loss rate above a threshold?*
  - Risks **unnecessarily disconnecting** nodes

- *Product of link delivery rates → prob. of e2e delivery?*
  - **Ignores inter-hop interference**
    - Prefers 2-hop, 0% loss route over 1-hop, 10% loss route (but latter is **double throughput**)

- *Throughput of highest-loss link on path?*
  - Also **ignores inter-hop interference**

ETX: Expected Transmission Count

- ***Link ETX:*** predicted number of transmissions
  - Calculate link ETX using forward, reverse delivery rates

  - To avoid retry, data packet **and** ACK must succeed

  - **Link ETX =** $\mathbf{1 / (d_f \times d_r)}$
    - d_f = forward link delivery ratio (data packet)
    - d_r = reverse link delivery ratio (ack packet)

- ***Path ETX:*** sum of the link ETX values on a path

Measuring link delivery ratios

- Nodes periodically send broadcast ***probe*** packets
  - All nodes know the **sending period** of probes
  - All nodes **compute loss rate** based on how many probes arrive, per measurement interval
- Nodes **enclose these loss measurements** in their transmitted probes
  - *e.g.* **B** tells node **A** the link delivery rate from **A** to **B**

Multi-bitrate radios

- ETX assumes all radios run at same bit-rate
  - But 802.11b rates: {1, 2, 5.5, 11} Mbit/s

- **Can't compare** two transmissions at 1 Mbit/s with two at 2 Mbit/s

- **Solution:** Use expected **time** spent on a packet, rather than transmission count

ETT: Expected Transmission Time

- ACKs always sent at 1 Mbps, data packets 1500 bytes

- Nodes send 1500-byte broadcast probes at every bit rate *b* to compute ***forward link delivery rates*** $\mathbf{d_f(b)}$
  - Send 60-byte (min size) probes at 1 Mbps → $\mathbf{d_r}$

- At each bit-rate *b*, $ETX_b = 1 / (d_f(b) \times d_r)$
- For packet of length *S*, $ETT_b = (S / b) \times ETX_b$
- **Link ETT** = $\min_b (ETT_b)$

ETT: Assumptions

- Path throughput estimate *t* is given by
  - t_i = throughput of hop *i*

$$t = \frac{1}{\sum_{\text{hop } i \in \text{path}} \frac{1}{t_i}}$$

- *Does ETT maximize throughput?* No!

1. Underestimates throughput for long (≥ 4-hop) paths
   - Distant nodes can send simultaneously

2. Overestimates throughput when transmissions on different hops collide and are lost

Roofnet evaluation

- TCP bulk transfers between **all node pairs** but always a **single flow at a time**
  - But background traffic present: **users always active**
- **Results:**

  1. **Wide spread** of end-to-end throughput across pairs
  2. **“Chain forwarding”** indeed creates **interference**
  3. **Lossy links** indeed **useful** in practice

Wireless Mesh Networks: Evolving Routing

- **DSDV** took DV out of wired (more static) networks
  - Better **coped with dynamism**

- **DSR** **addressed protocol overheads** of routing

- **ETX and ETT abolished hop-count** as a viable metric
  - Replaced it with **throughput as the metric**

Next Week's Precepts:

Introduction to Lab 2:
HackRF MAC Protocols

Tuesday Topic:

Geographic Routing