Distance Vector Routing

Brad Karp
UCL Computer Science

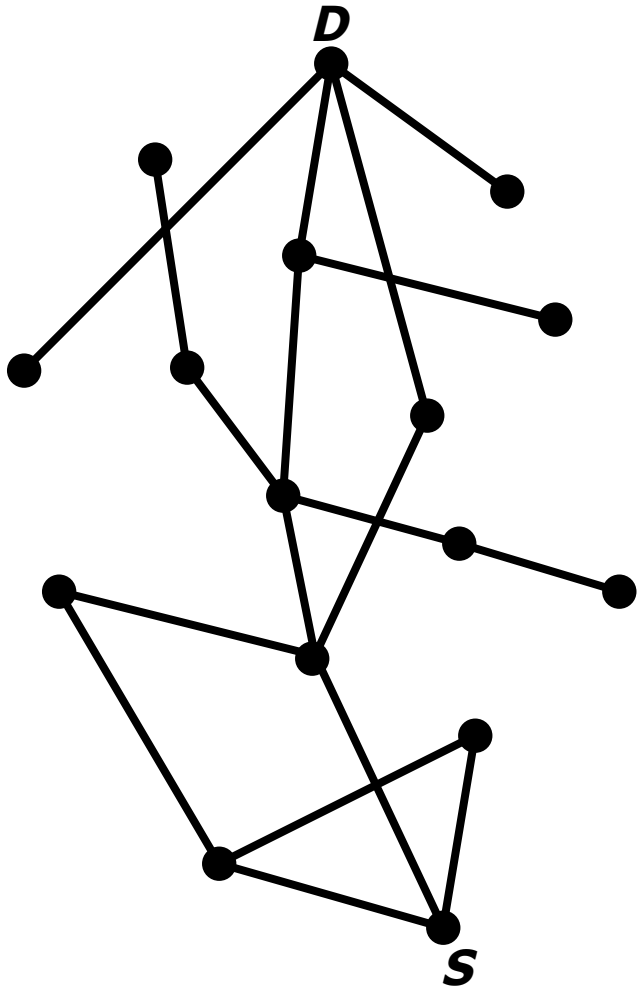
CS 3035/GZ01
19th November 2013

Outline

- Routing Problem Definition
- Routing in Practice: traceroute Examples
- Definitions: Hosts, Routers, Interfaces, Subnets
- Shortest-Path Routing
- Routing Tables
- Distance Vector Algorithm
- Pathologies: Bouncing and Counting to Infinity
- Optimizations: Split Horizon and Poison Reverse
- War Story: Synchronization of Routing Messages

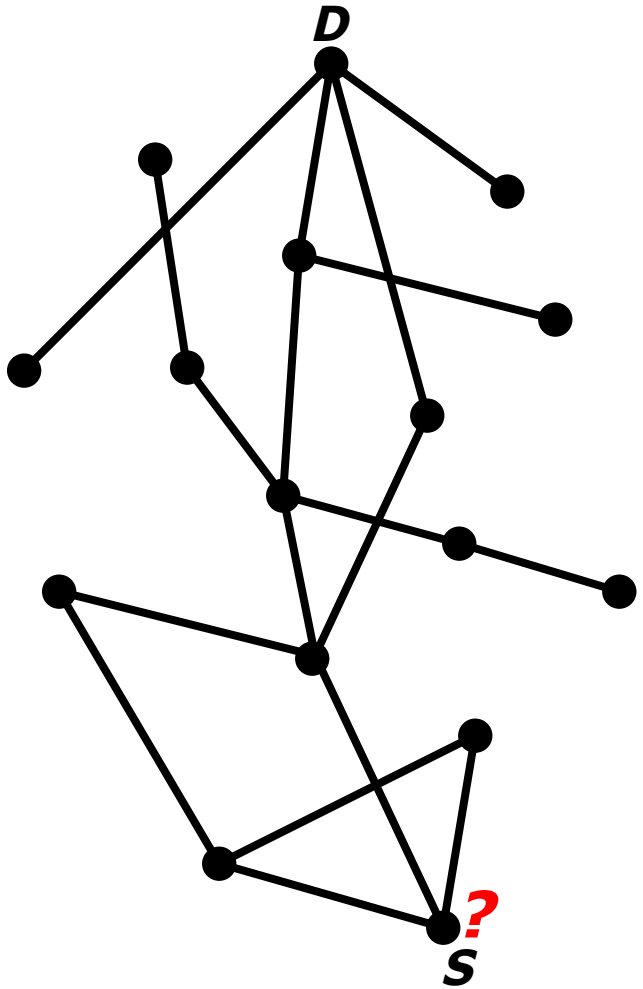
The Routing Problem

- Each router has several interfaces to links
- Each router has unique node ID
- Packets stamped with destination node ID
- Router must choose next hop for received packet
- Routing protocol: communication to accumulate state for use in forwarding decisions
- Routes change with topology



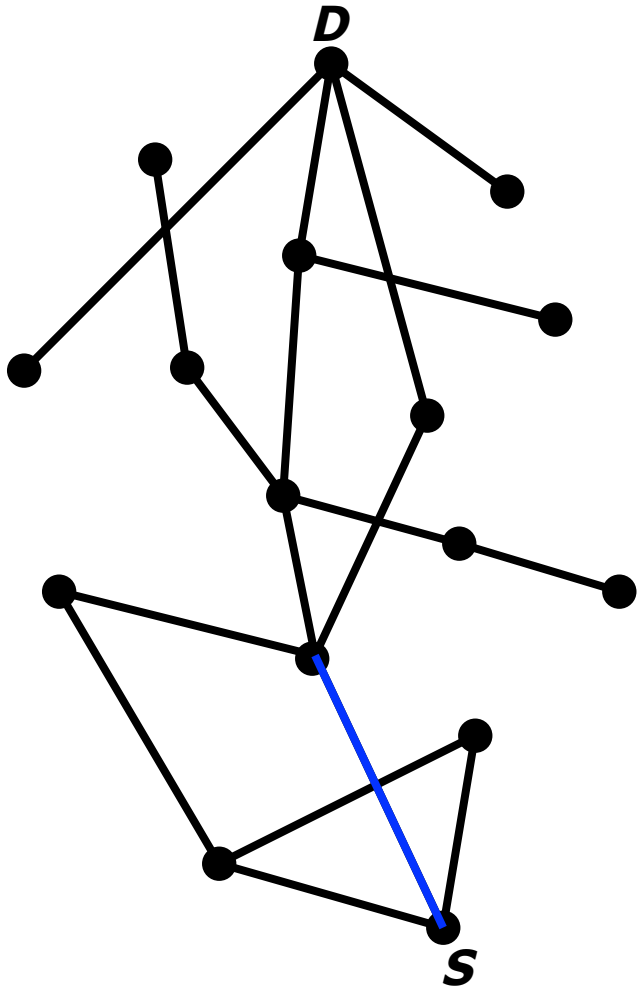
The Routing Problem

- Each router has several interfaces to links
- Each router has unique node ID
- Packets stamped with destination node ID
- Router must choose next hop for received packet
- Routing protocol: communication to accumulate state for use in forwarding decisions
- Routes change with topology



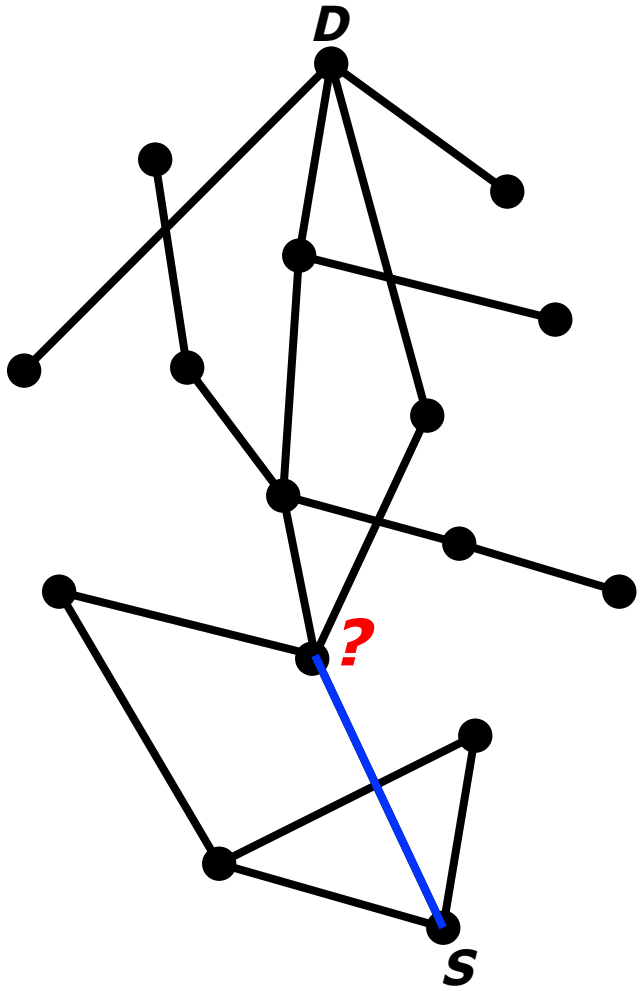
The Routing Problem

- Each router has several interfaces to links
- Each router has unique node ID
- Packets stamped with destination node ID
- Router must choose next hop for received packet
- Routing protocol: communication to accumulate state for use in forwarding decisions
- Routes change with topology



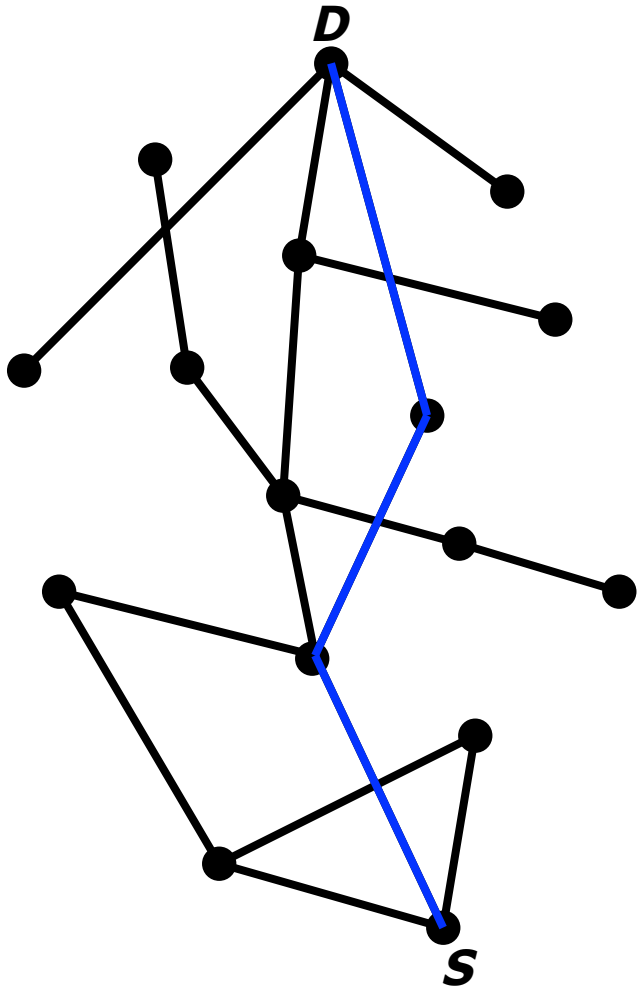
The Routing Problem

- Each router has several interfaces to links
- Each router has unique node ID
- Packets stamped with destination node ID
- Router must choose next hop for received packet
- Routing protocol: communication to accumulate state for use in forwarding decisions
- Routes change with topology



The Routing Problem

- Each router has several interfaces to links
- Each router has unique node ID
- Packets stamped with destination node ID
- Router must choose next hop for received packet
- Routing protocol: communication to accumulate state for use in forwarding decisions
- Routes change with topology



Routing on Changing Networks

- Links may be cut
- Routers or their interfaces may fail
- Hazard: traffic loops
  - Amplify traffic; severely congest links
  - TTL will eventually drop packets, but typically only after congestion
- Hazard: disconnection
  - Any routing algorithm will take time to converge to correct routes after link(s) break

traceroute: Internet Routes Exposed

- UNIX: traceroute <destination>
- Windows: tracert <destination>
- Displays all hops on route between host where invoked and <destination>
- How:
  - sends sequence of carefully constructed packets that "expire" after 1 hop, 2 hops, …
  - each elicits ICMP error from router that many hops from sender

Traceroute: boffin.cs.ucl.ac.uk to www.cl.cam.ac.uk (Cambridge, UK)

traceroute to www.cl.cam.ac.uk (128.232.0.20), 64 hops max, 40 byte packets

1 cisco (128.16.64.1) 0.370 ms 0.322 ms 0.361 ms

2 128.40.255.29 (128.40.255.29) 0.483 ms 0.348 ms 0.487 ms

3 128.40.20.1 (128.40.20.1) 0.486 ms 0.342 ms 0.362 ms

4 128.40.20.62 (128.40.20.62) 0.486 ms 0.474 ms 0.363 ms

5 ulcc-gsr.lmn.net.uk (194.83.101.5) 0.485 ms 0.346 ms 0.362 ms

6 london-bar1.ja.net (146.97.40.33) 0.485 ms 0.470 ms 0.488 ms

7 po10-0.lond-scr.ja.net (146.97.35.5) 0.735 ms 0.722 ms 0.610 ms

8 po0-0.cambridge-bar.ja.net (146.97.35.10) 5.232 ms 4.964 ms 4.734 ms

9 route-enet-3.cam.ac.uk (146.97.40.50) 4.982 ms 4.841 ms 4.860 ms

10 route-cent-3.cam.ac.uk (192.153.213.194) 4.984 ms 4.964 ms 4.861 ms

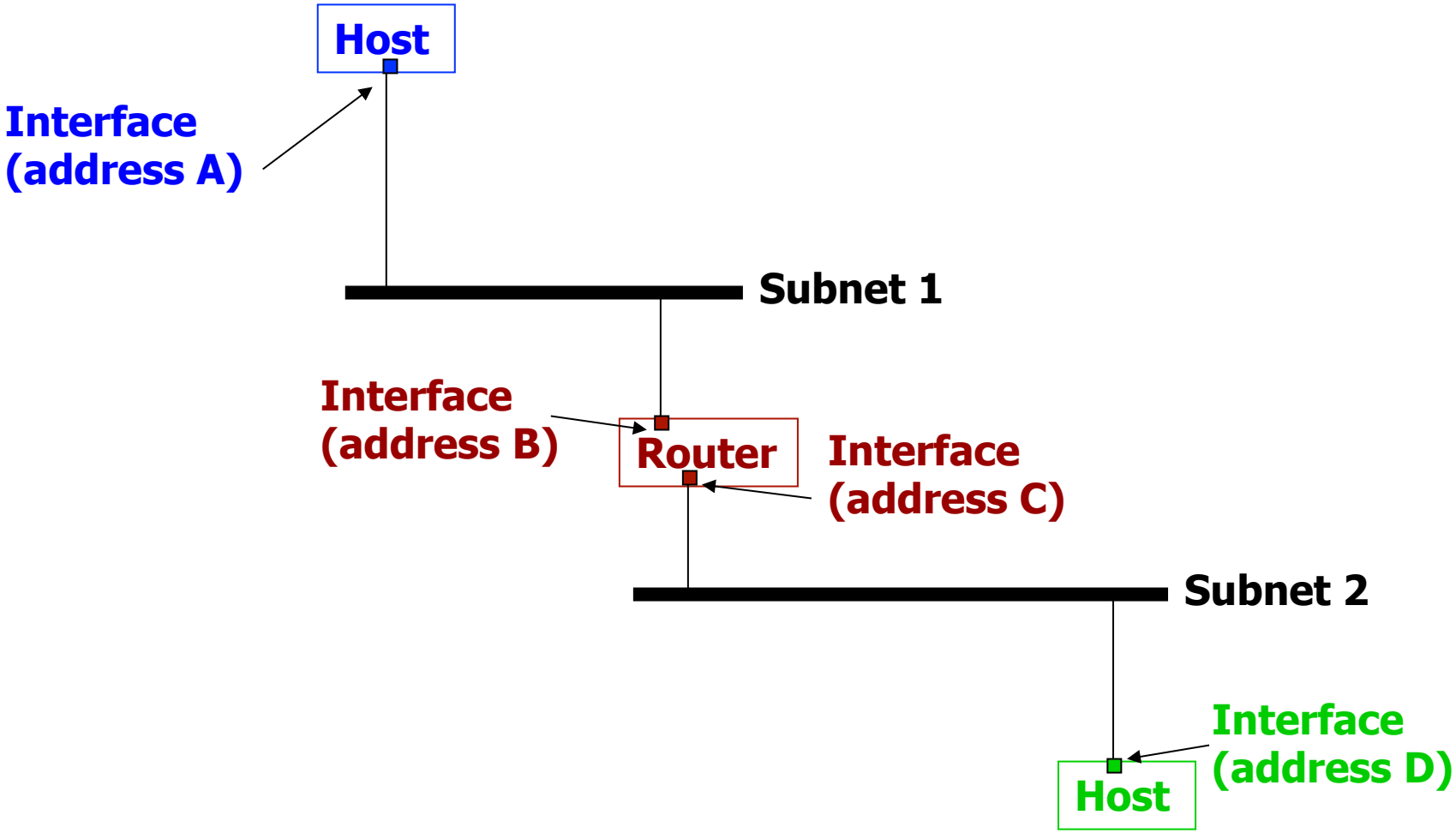
traceroute: boffin.cs.ucl.ac.uk to www.icir.org (Berkeley, CA, USA)

```
traceroute to www.icir.org (192.150.187.11), 64 hops max, 40 byte packets
 1  cisco (128.16.64.1)  0.258 ms  0.310 ms  0.239 ms
 2  128.40.255.29 (128.40.255.29)  0.481 ms  0.472 ms  0.368 ms
 3  128.40.20.129 (128.40.20.129)  0.479 ms  0.350 ms  0.363 ms
 4  128.40.20.190 (128.40.20.190)  0.486 ms  0.474 ms  0.363 ms
 5  ulcc-gsr.lmn.net.uk (194.83.101.5)  0.360 ms  0.471 ms  0.362 ms
 6  london-bar1.ja.net (146.97.40.33)  0.486 ms  0.471 ms  0.363 ms
 7  po10-0.lond-scr.ja.net (146.97.35.5)  0.610 ms  0.595 ms  0.614 ms
 8  po6-0.lond-scr3.ja.net (146.97.33.30)  1.110 ms  1.094 ms  0.989 ms
 9  po1-0.gn2-gw1.ja.net (146.97.35.98)  0.983 ms  0.846 ms  0.862 ms
10  janet.rt1.lon.uk.geant2.net (62.40.124.197)  1.110 ms  1.092 ms  1.109 ms
11  uk.ny1.ny.geant.net (62.40.96.169)  69.695 ms  97.916 ms  69.688 ms
12  198.32.11.50 (198.32.11.50)  80.680 ms  70.045 ms  83.318 ms
13  chinng-nycmng.abilene.ucaid.edu (198.32.8.82)  95.302 ms  101.900 ms  89.927 ms
14  iplsng-chinng.abilene.ucaid.edu (198.32.8.77)  93.712 ms  94.003 ms  93.680 ms
15  kscyng-iplsng.abilene.ucaid.edu (198.32.8.81)  106.290 ms  105.278 ms  102.921 ms
16  dnvrng-kscyng.abilene.ucaid.edu (198.32.8.13)  113.542 ms  113.530 ms  115.905 ms
17  snvang-dnvrng.abilene.ucaid.edu (198.32.8.1)  138.648 ms  138.256 ms  138.294 ms
18  losang-snvang.abilene.ucaid.edu (198.32.8.94)  145.736 ms  145.625 ms  145.780 ms
19  hpr-lax-gsr1--abilene-LA-10ge.cenic.net (137.164.25.2)  145.881 ms  146.131 ms  146.770 ms
20  svl-hpr--lax-hpr-10ge.cenic.net (137.164.25.13)  153.514 ms  153.487 ms  153.521 ms
21  hpr-ucb-ge--svl-hpr.cenic.net (137.164.27.134)  196.734 ms  154.875 ms  154.764 ms
22  g3-17.inr-202-reccev.Berkeley.EDU (128.32.0.35)  154.639 ms  154.746 ms  154.643 ms
23  fast4-1-0.inr-667-eva.Berkeley.EDU (128.32.0.90)  154.893 ms  154.749 ms  154.896 ms
24  router3-fast1-0-0.ICSI.Berkeley.EDU (169.229.0.138)  155.133 ms  155.249 ms  154.884 ms
25  router1-vlan5.icsi.berkeley.edu (192.150.187.249)  155.397 ms  155.245 ms  156.017 ms
```

Hosts, Routers, Interfaces, Subnets

- Host: at least one interface, sometimes multiple ones
- Host: runs applications
- Router: typically doesn't run applications
- Router: has multiple interfaces, routes packets among them
- Each interface has unique IP address (true both for hosts and routers)
- Subnet: typically a single Ethernet broadcast domain, shared by hosts and routers

Hosts, Routers, Interfaces, Subnets



Address Aggregation

- Each Internet host (interface) has unique 32-bit IP address
- Must every router in entire Internet know about every other router?
- No; interfaces on same subnet share address prefix
  - e.g., 128.16.64.30, 128.16.64.92 on same subnet
- IP routing destination is subnet's prefix; not single 32-bit IP address

Shortest-Path Routing

- View network as graph
  - Routers are vertices, links are edges
  - Link metrics are edge weights

Shortest paths problem:

  - What path between two vertices offers minimal sum of edge weights?

- Classic algorithms find single-source shortest paths when entire graph known centrally
  - Dijkstra's Algorithm, Bellman-Ford Algorithm
- In Internet, each router only knows its own interfaces' addresses; no central knowledge of entire graph

Outline

- Routing Problem Definition
- Routing in Practice: traceroute Examples
- Definitions: Hosts, Routers, Interfaces, Subnets
- Shortest-Path Routing
- Routing Tables
- Distance Vector Algorithm
- Pathologies: Bouncing and Counting to Infinity
- Optimizations: Split Horizon and Poison Reverse
- War Story: Synchronization of Routing Messages

Routing Tables

- Destination field: subnet ID (address prefix)
- Interface field: which interface of router on which to forward to reach destination
- Metric field: total cost to reach that destination
- Administrator assigns metrics to interfaces
- Startup: initialize table to contain one entry for each interface's subnet

Destination	Interface	Metric
A	0	0
B	1	0

Routing Tables: Forwarding

- Packet arrives for destination D
- Search for D in destination field of routing table
  - if found, forward on interface number in table entry
  - if not found, drop packet; no route known

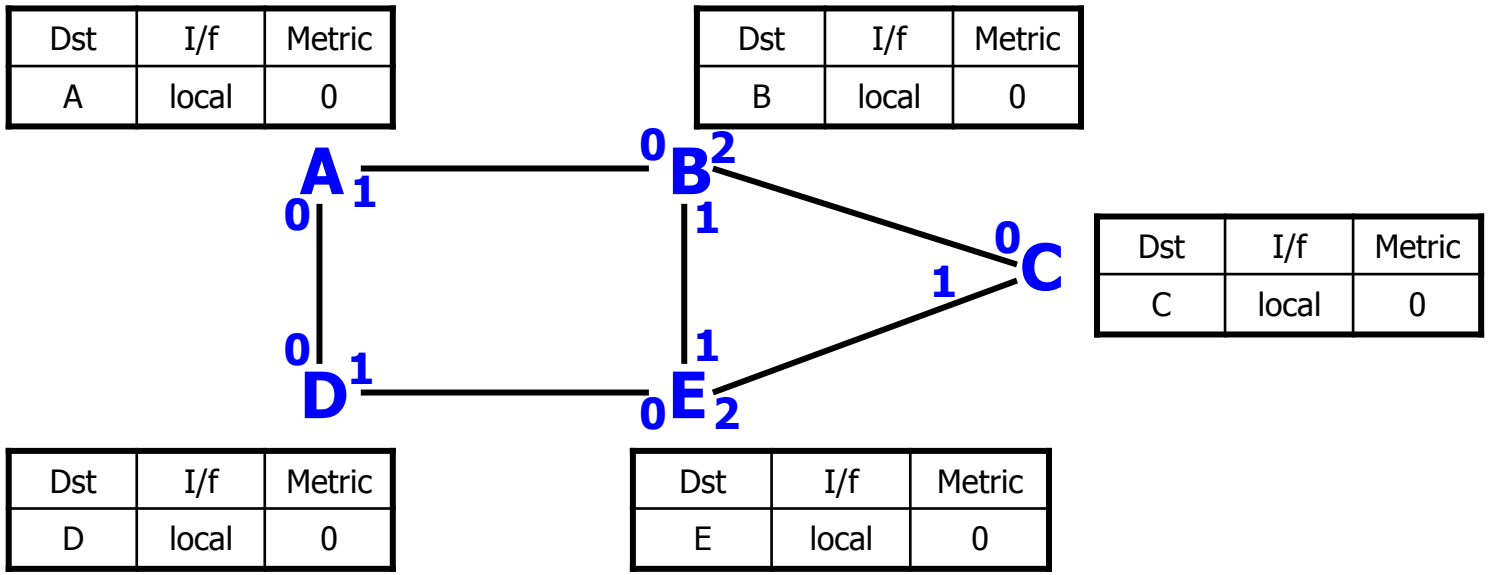
Basic Distance Vector Algorithm
(Failures Not Yet Considered)

- Distributed Bellman-Ford (DBF)
- Periodically, send all routing table entries (destination and metric fields) to all immediate neighbor routers
- Upon receipt of routing table entry for destination D with metric m on interface i:

```
m += metric for interface i
r = lookup(D) in routing table
if (r = "not found") then
    newr = new routing table entry
    newr.D = D; newr.m = m; newr.i = i
    add newr to table
else if (m < r.m) then
    r.m = m; r.i = i
```

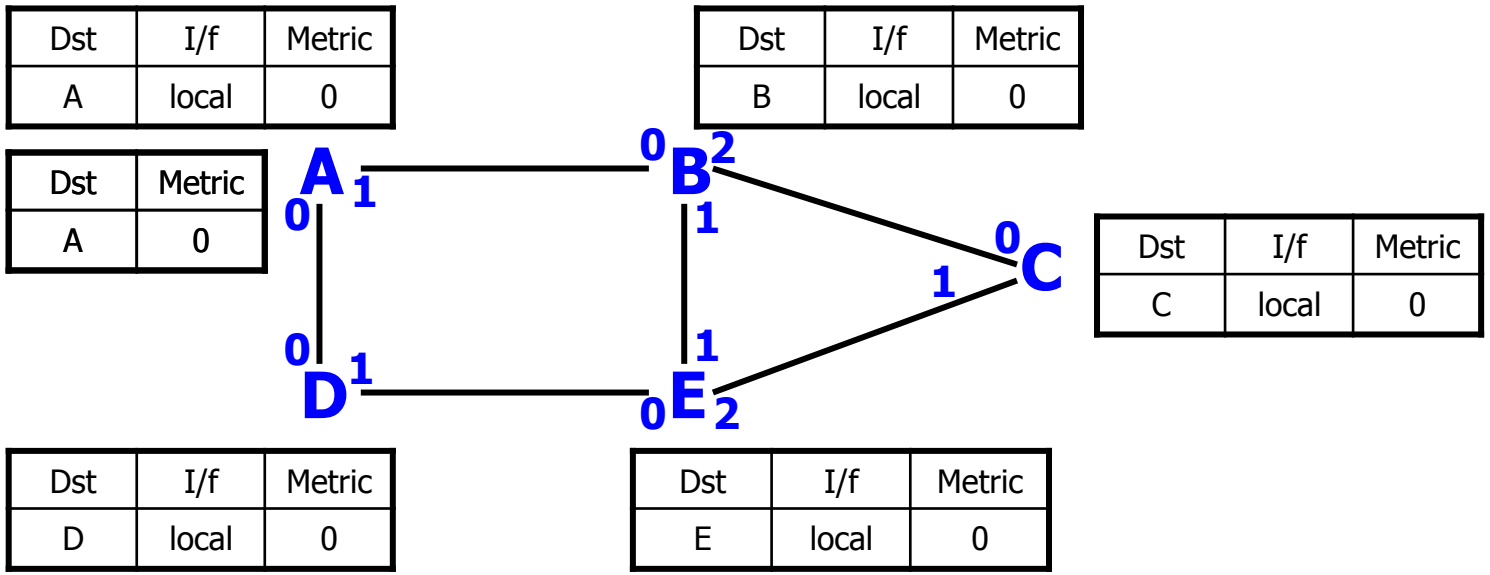
Distance Vector: Example

- Consider simple network where all nodes are routers, addresses are simply single letters
- Initial routing tables when routers first start:



Distance Vector: Example

- Consider simple network where all nodes are routers, addresses are simply single letters
- Initial routing tables when routers first start:

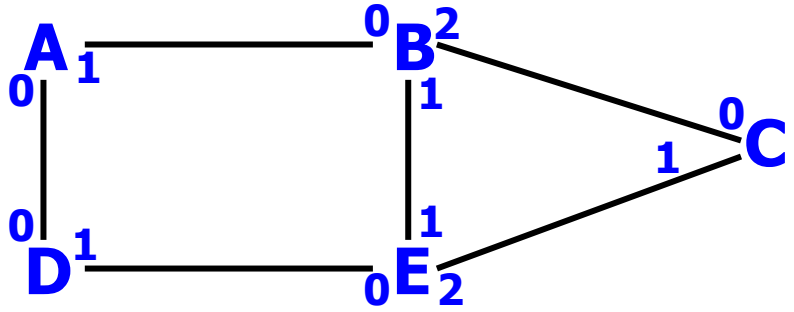


Distance Vector: Iteration 1

- Routers incorporate received announcements:

Dst	I/f	Metric
A	local	0
B	1	1
D	0	1

Dst	I/f	Metric
B	local	0
A	0	1
C	2	1
E	1	1



Dst	I/f	Metric
C	local	0
B	0	1
E	1	1

Dst	I/f	Metric
D	local	0
A	0	1
E	1	1

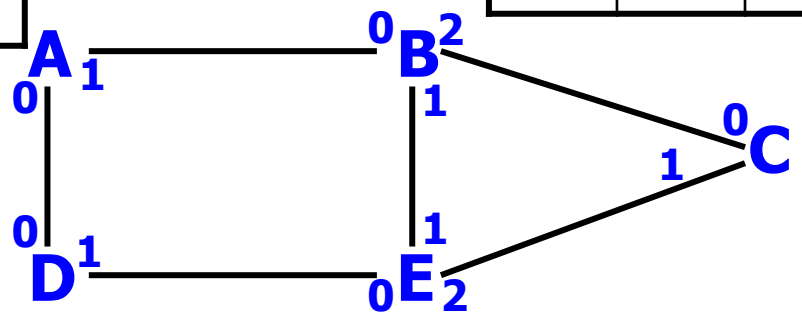
Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1

Distance Vector: Iteration 2

- Routers incorporate received announcements:

Dst	I/f	Metric
A	local	0
B	1	1
D	0	1
E	0	2
C	1	2

Dst	I/f	Metric
B	local	0
A	0	1
C	2	1
E	1	1
D	1	2



Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	1	2
C	1	2

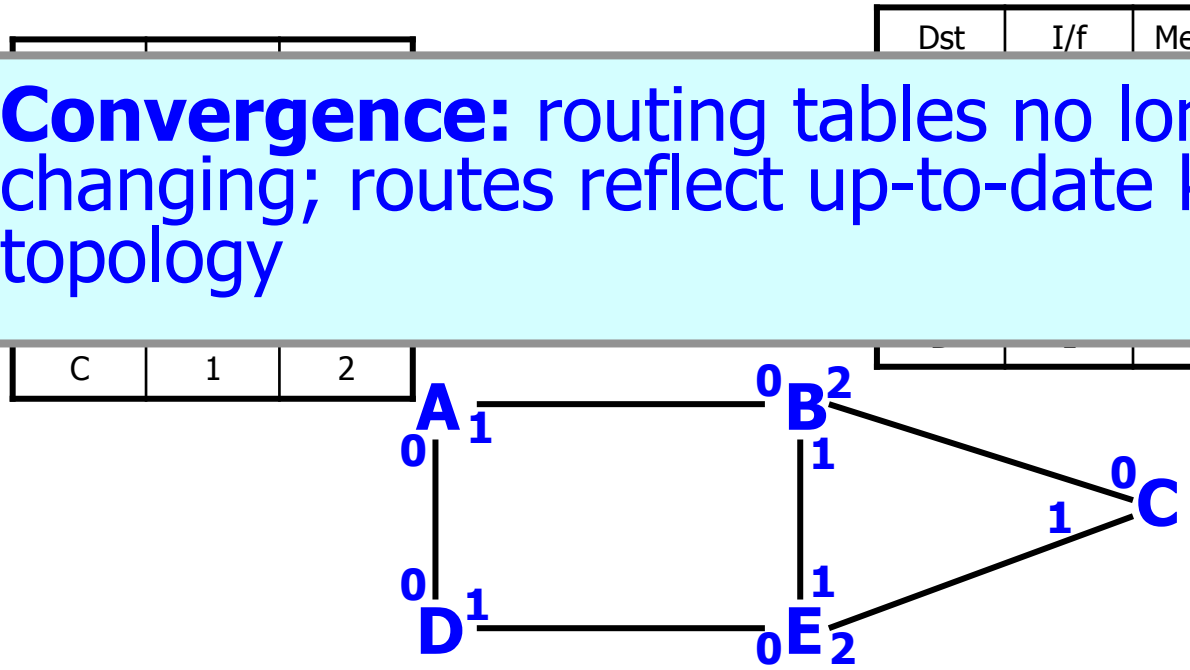
Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	1	2

Dst	I/f	Metric
C	local	0
B	0	1
E	1	1
A	0	2
D	1	2

Distance Vector: Iteration 2

- Routers incorporate received announcements:

Convergence: routing tables no longer changing; routes reflect up-to-date knowledge of topology



Dst	I/f	Metric
C	1	2

Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	1	2
C	1	2

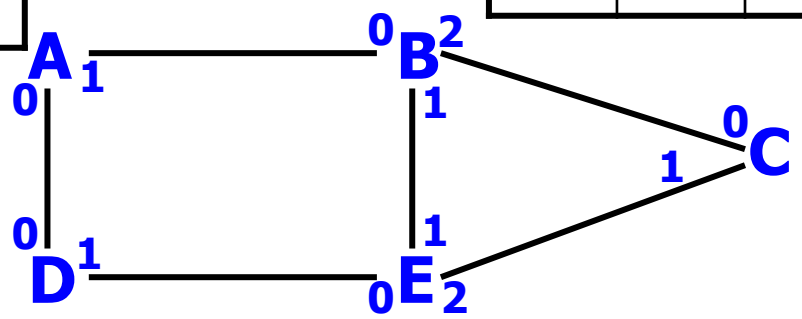
Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	1	2

Dst	I/f	Metric
C	local	0
B	0	1
E	1	1
A	0	2
D	1	2

Link Failure (I)

Dst	I/f	Metric
A	local	0
B	1	1
D	0	1
E	0	2
C	1	2

Dst	I/f	Metric
B	local	0
A	0	1
C	2	1
E	1	1
D	1	2



Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	1	2
C	1	2

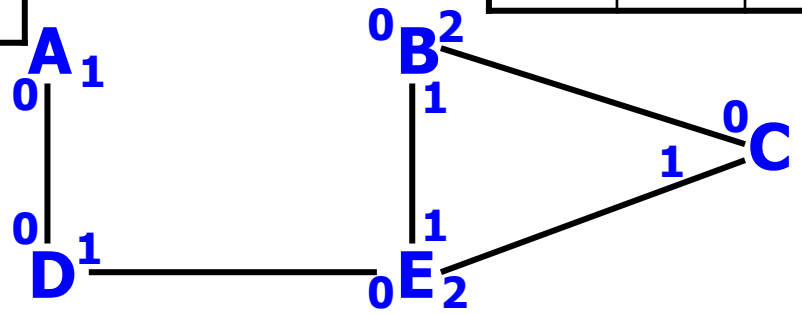
Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	1	2

Dst	I/f	Metric
C	local	0
B	0	1
E	1	1
A	0	2
D	1	2

Link Failure (I)

Dst	I/f	Metric
A	local	0
B	1	1
D	0	1
E	0	2
C	1	2

Dst	I/f	Metric
B	local	0
A	0	1
C	2	1
E	1	1
D	1	2



Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	1	2
C	1	2

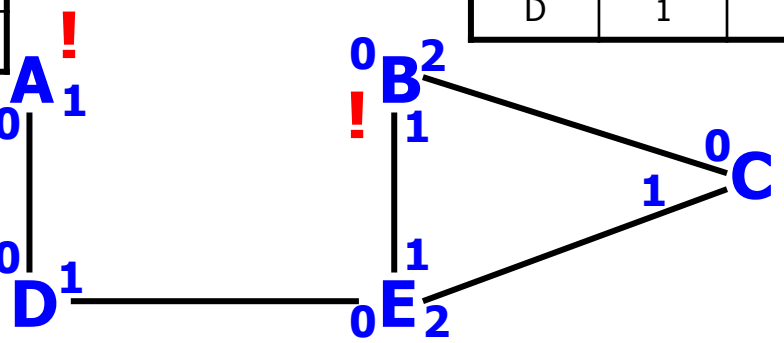
Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	1	2

Dst	I/f	Metric
C	local	0
B	0	1
E	1	1
A	0	2
D	1	2

Link Failure (I)

Dst	I/f	Metric
A	local	0
B	1	1
D	0	1
E	0	2
C	1	2

Dst	I/f	Metric
B	local	0
A	0	1
C	2	1
E	1	1
D	1	2



Dst	I/f	Metric
C	local	0
B	0	1
E	1	1
A	0	2
D	1	2

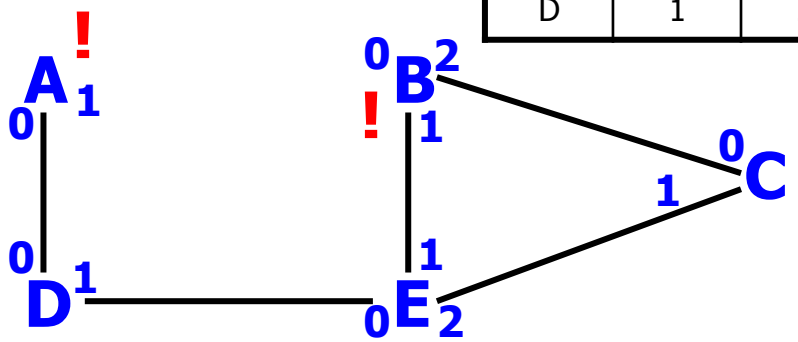
Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	1	2
C	1	2

Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	1	2

Link Failure (II)

Dst	I/f	Metric
A	local	0
B	1	**Inf**
D	0	1
E	0	2
C	1	**Inf**

Dst	I/f	Metric
B	local	0
A	0	**Inf**
C	2	1
E	1	1
D	1	2



Dst	I/f	Metric
C	local	0
B	0	1
E	1	1
A	0	2
D	1	2

Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	1	2
C	1	2

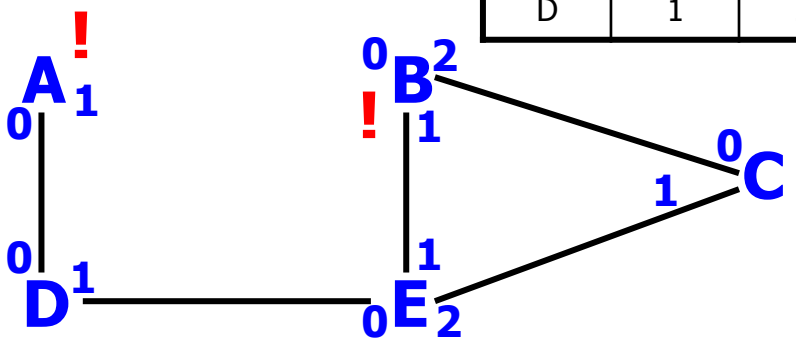
Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	1	2

Link Failure (II)

Dst	I/f	Metric
A	local	0
B	1	**Inf**
D	0	1
E	0	2
C	1	**Inf**

Dst	Metric
A	0
B	**Inf**
D	1
E	2
C	**Inf**

Dst	I/f	Metric
B	local	0
A	0	**Inf**
C	2	1
E	1	1
D	1	2



Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	1	2
C	1	2

Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	1	2

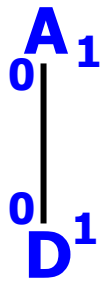
Dst	I/f	Metric
C	local	0
B	0	1
E	1	1
A	0	2
D	1	2

DV Algorithm, Revised

- Upon receipt of routing table entry for destination D with metric m on interface i:

```
m += metric for interface i
r = lookup(D) in routing table
if (r = "not found") then
    newr = new routing table entry
    newr.D = D; newr.m = m; newr.i = i
    add newr to table
else if (i == r.i) then
    r.m = m
else if (m < r.m) then
    r.m = m; r.i = i
```

Link Failure (III)

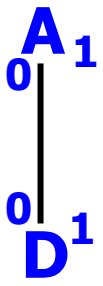


Dst	Metric
A	1
B	**Inf**
D	2
E	3
C	**Inf**

+

Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	1	2
C	1	2

Link Failure (III)



Dst	Metric
A	1
B	**Inf**
D	2
E	3
C	**Inf**

+

Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	1	2
C	1	2

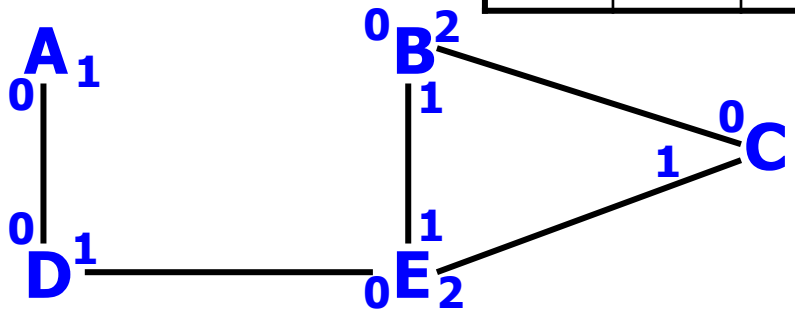
(no change)

Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	1	2
C	1	2

Link Failure (IV)

Dst	I/f	Metric
A	local	0
B	1	**Inf**
D	0	1
E	0	2
C	1	**Inf**

Dst	I/f	Metric
B	local	0
A	0	**Inf**
C	2	1
E	1	1
D	1	2



Dst	I/f	Metric
C	local	0
B	0	1
E	1	1
A	0	2
D	1	2

Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	1	2
C	1	2

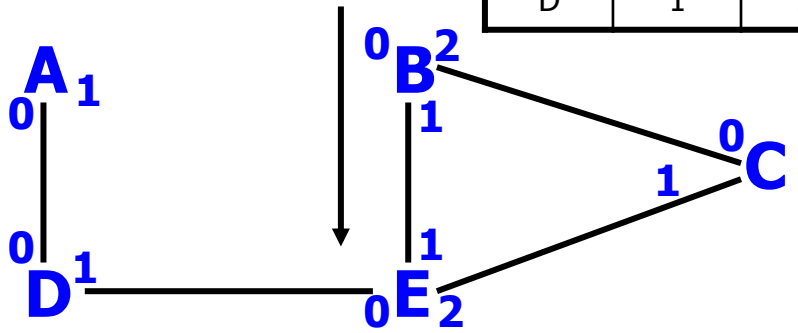
Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	1	2

Link Failure (IV)

Dst	I/f	Metric
A	local	0
B	1	**Inf**
D	0	1
E	0	2
C	1	**Inf**

Dst	Metric
B	0
A	**Inf**
C	1
E	1
D	2

Dst	I/f	Metric
B	local	0
A	0	**Inf**
C	2	1
E	1	1
D	1	2

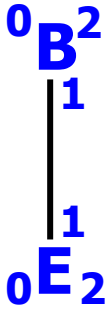


Dst	I/f	Metric
C	local	0
B	0	1
E	1	1
A	0	2
D	1	2

Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	1	2
C	1	2

Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	1	2

Link Failure (V)



Dst	Metric
B	1
A	**Inf**
C	2
E	2
D	3

+

Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	1	2

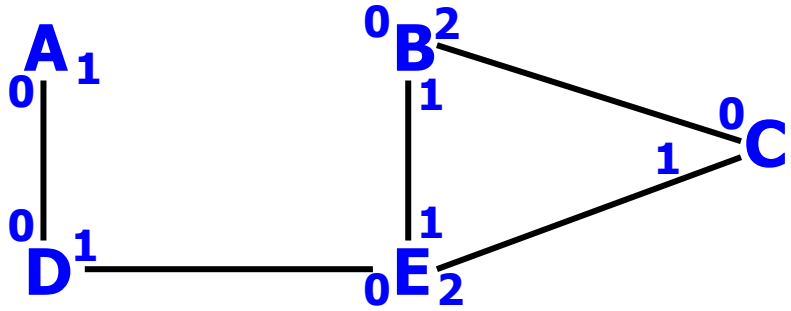
→

Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	1	**Inf**

Link Failure (VI)

Dst	I/f	Metric
A	local	0
B	1	**Inf**
D	0	1
E	0	2
C	1	**Inf**

Dst	I/f	Metric
B	local	0
A	0	**Inf**
C	2	1
E	1	1
D	1	2



Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	1	2
C	1	2

Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	1	**Inf**

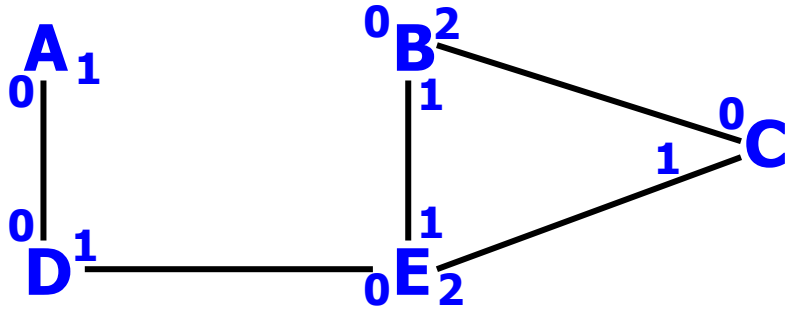
Dst	I/f	Metric
C	local	0
B	0	1
E	1	1
A	0	**Inf**
D	1	2

Link Failure (VI)

Dst	I/f	Metric
A	local	0

Dst	I/f	Metric
B	local	0

Next round: all routers broadcast their tables to neighbors…



Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	1	2
C	1	2

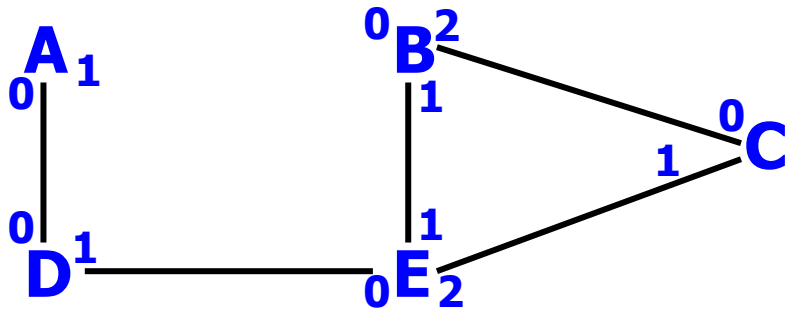
Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	1	**Inf**

Dst	I/f	Metric
C	local	0
B	0	1
E	1	1
A	0	**Inf**
D	1	2

Link Failure (VII)

Dst	I/f	Metric
A	local	0
B	**0**	**3**
D	0	1
E	0	2
C	**0**	**3**

Dst	I/f	Metric
B	local	0
A	0	**Inf**
C	2	1
E	1	1
D	1	2



Dst	I/f	Metric
C	local	0
B	0	1
E	1	1
A	0	**Inf**
D	1	2

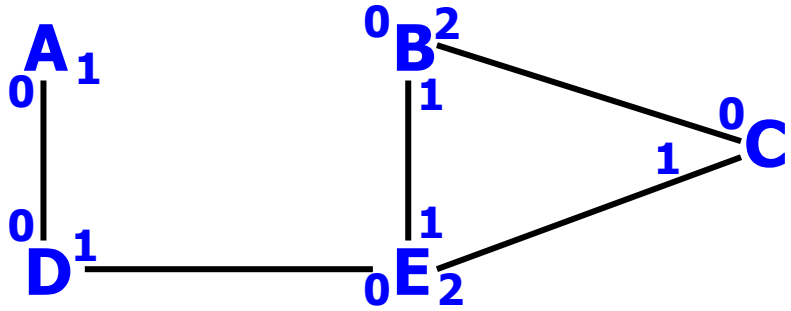
Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	1	2
C	1	2

Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	**0**	**2**

Link Failure (VIII)

Dst	I/f	Metric
A	local	0
B	**0**	**3**
D	0	1
E	0	2
C	**0**	**3**

Dst	I/f	Metric
B	local	0
A	**1**	**3**
C	2	1
E	1	1
D	1	2



Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	1	2
C	1	2

Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	**0**	**2**

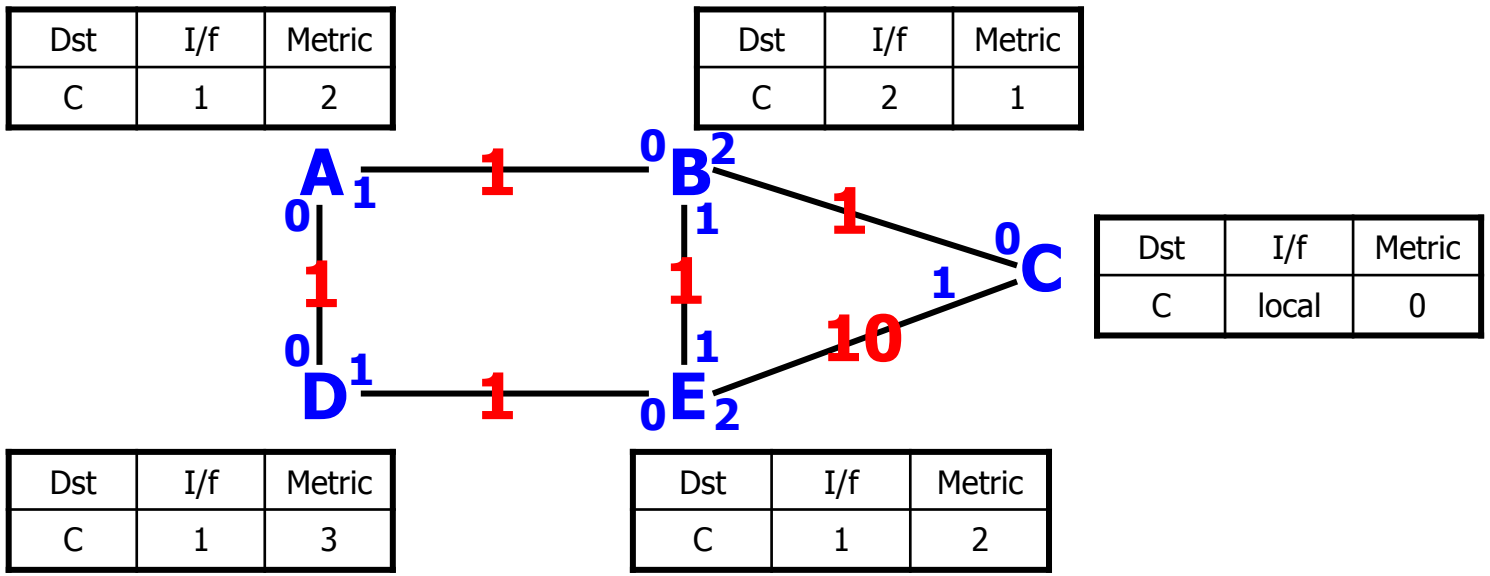
Dst	I/f	Metric
C	local	0
B	0	1
E	1	1
A	**1**	**3**
D	1	2

Outline

- Routing Problem Definition
- Routing in Practice: traceroute Examples
- Definitions: Hosts, Routers, Interfaces, Subnets
- Shortest-Path Routing
- Routing Tables
- Distance Vector Algorithm
- Pathologies: Bouncing and Counting to Infinity
- Optimizations: Split Horizon and Poison Reverse
- War Story: Synchronization of Routing Messages

Bouncing (I)

- Consider same network, where link (C, E) has metric 10; all others have metric 1
- Consider all nodes' routes to C after convergence
- Now suppose link (B, C) breaks



Dst	I/f	Metric
C	1	2

Dst	I/f	Metric
C	2	1

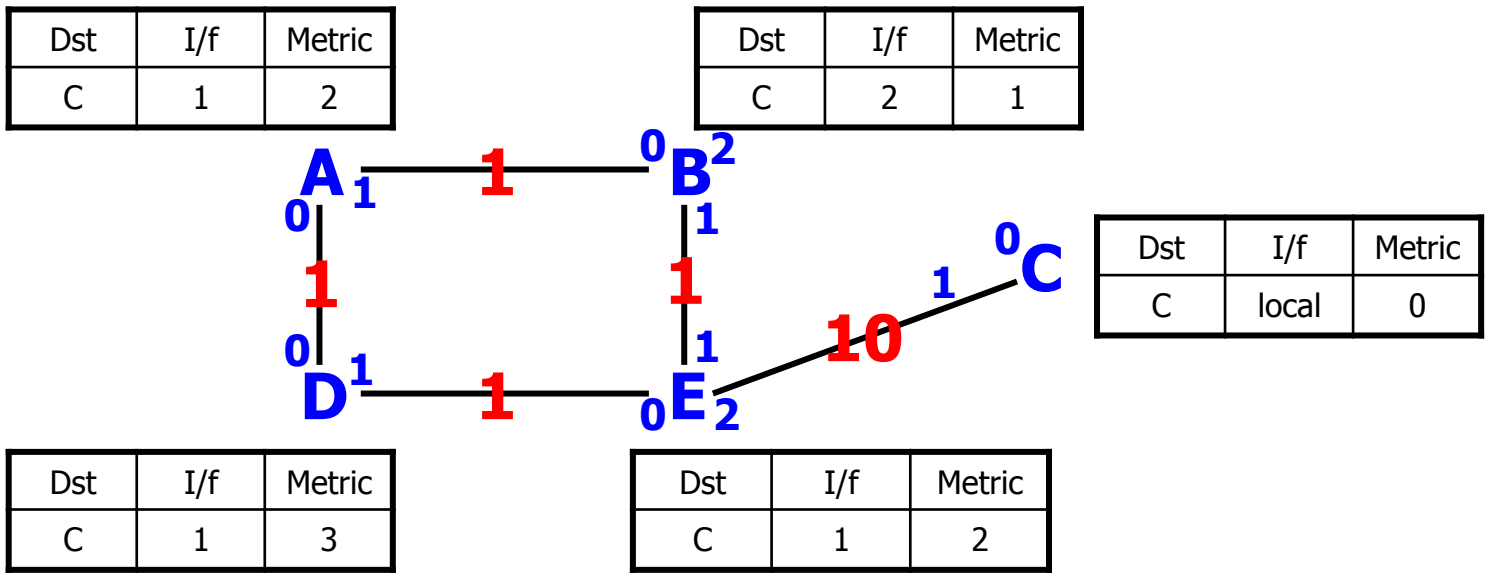
Dst	I/f	Metric
C	local	0

Dst	I/f	Metric
C	1	3

Dst	I/f	Metric
C	1	2

Bouncing (I)

- Consider same network, where link (C, E) has metric 10; all others have metric 1
- Consider all nodes' routes to C after convergence
- Now suppose link (B, C) breaks



Dst	I/f	Metric
C	1	2

Dst	I/f	Metric
C	2	1

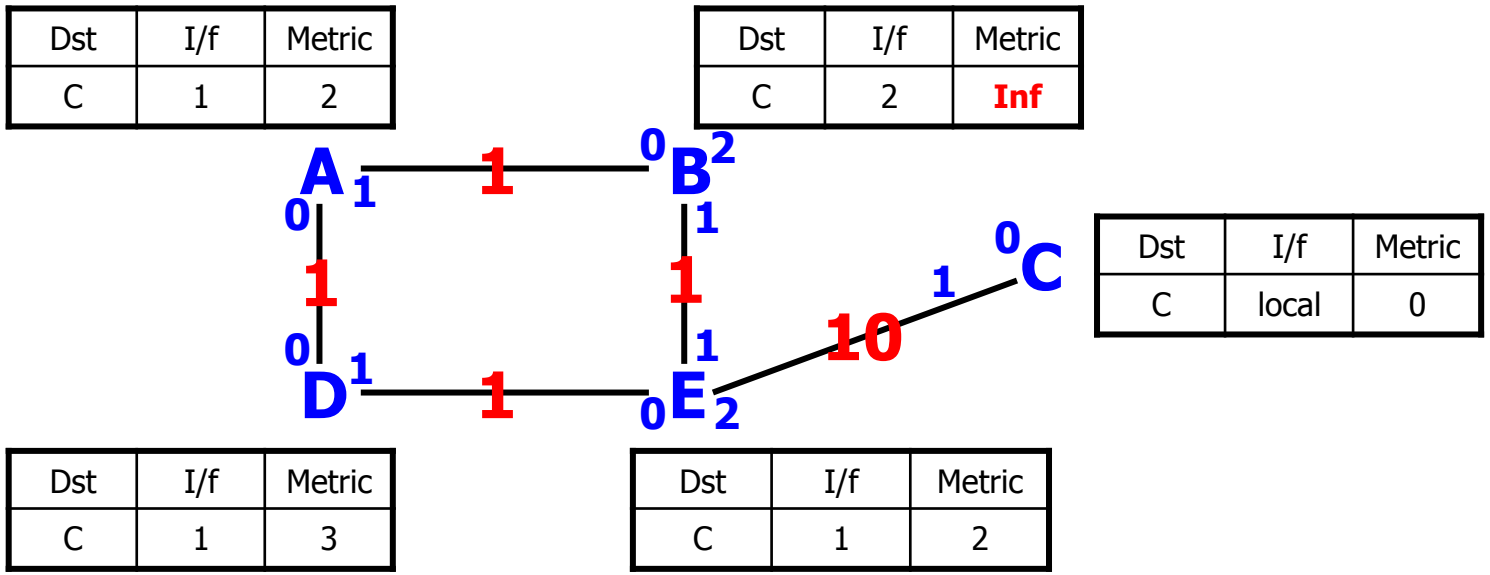
Dst	I/f	Metric
C	local	0

Dst	I/f	Metric
C	1	3

Dst	I/f	Metric
C	1	2

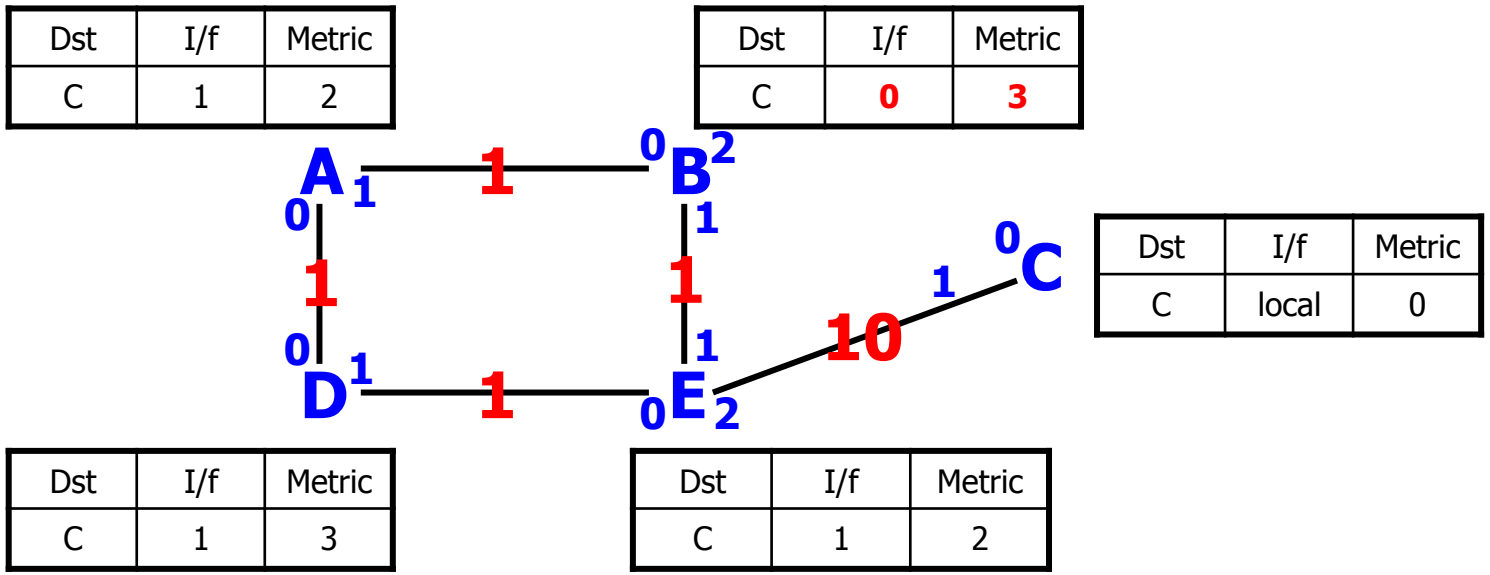
Bouncing (II)

- Suppose A advertises its table first…



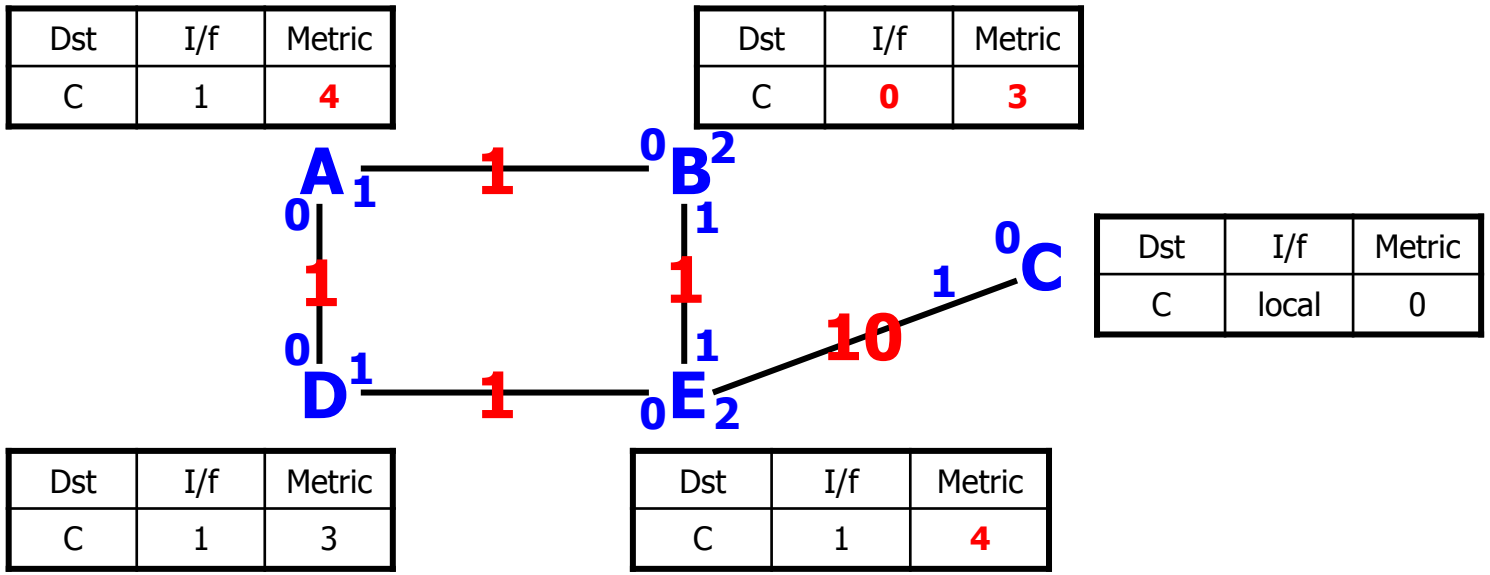
Bouncing (III)

- Suppose A advertises its table first…
- …and B advertises its table next…



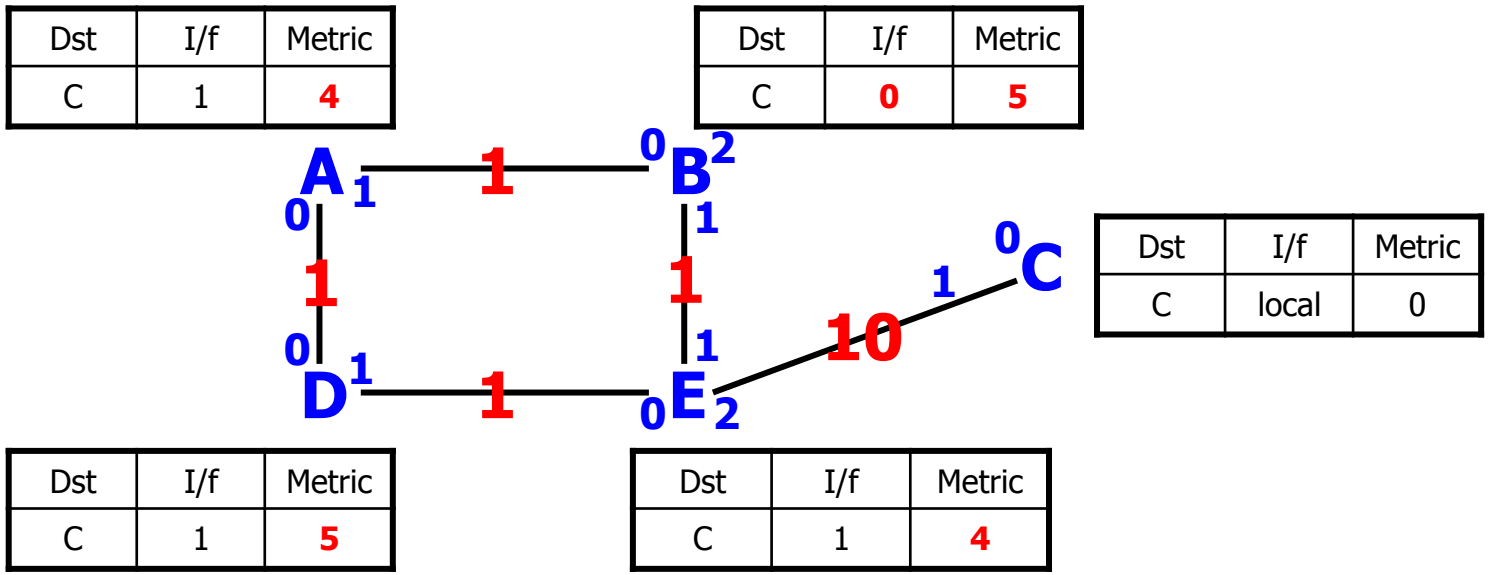
Bouncing (IV)

- Suppose A advertises its table first…
- …and B advertises its table next…
- **Loop between A and B for destination C!**
- **If C now advertises its table, E will ignore cost 10 route!**



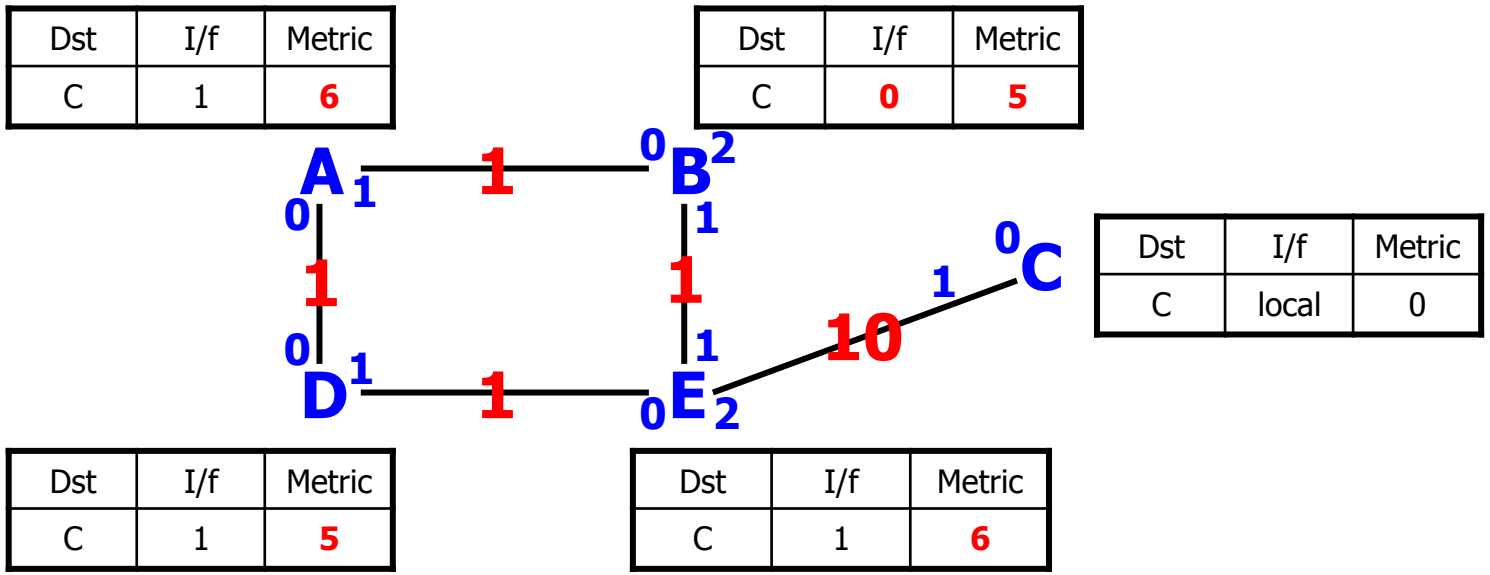
Bouncing (V)

- Suppose A and E advertise next…



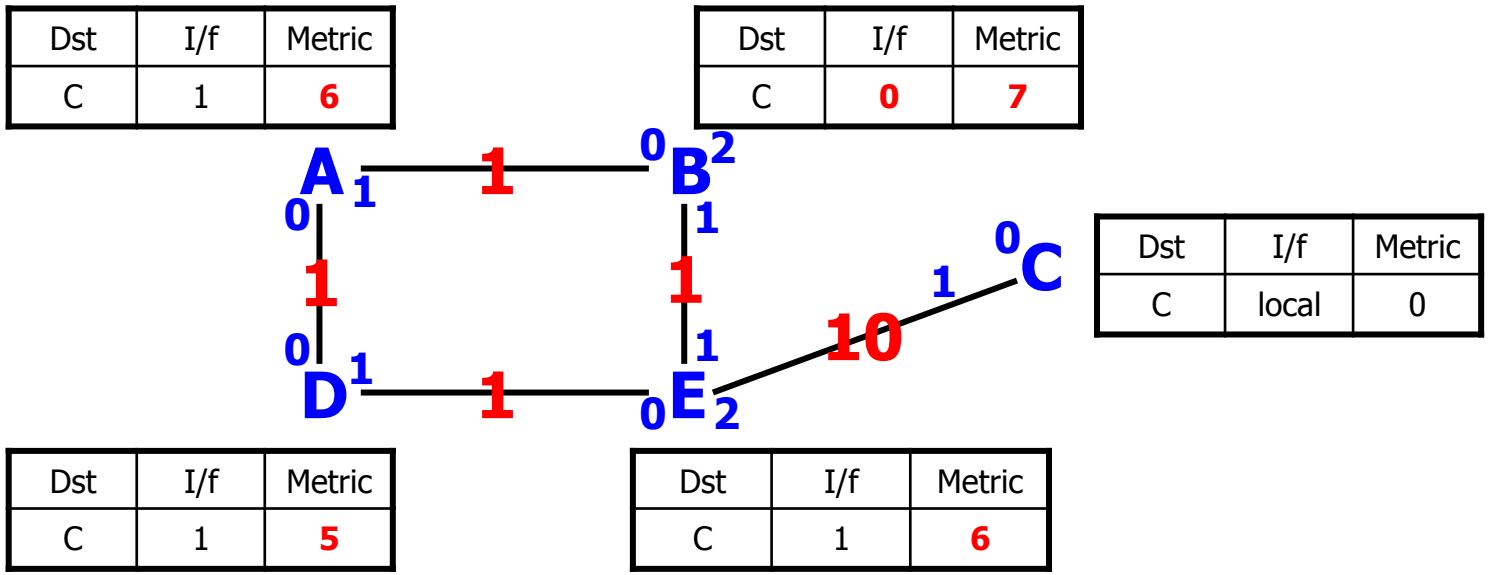
Bouncing (VI)

- Suppose A and E advertise next…
- …and B advertises next



Bouncing (VII)

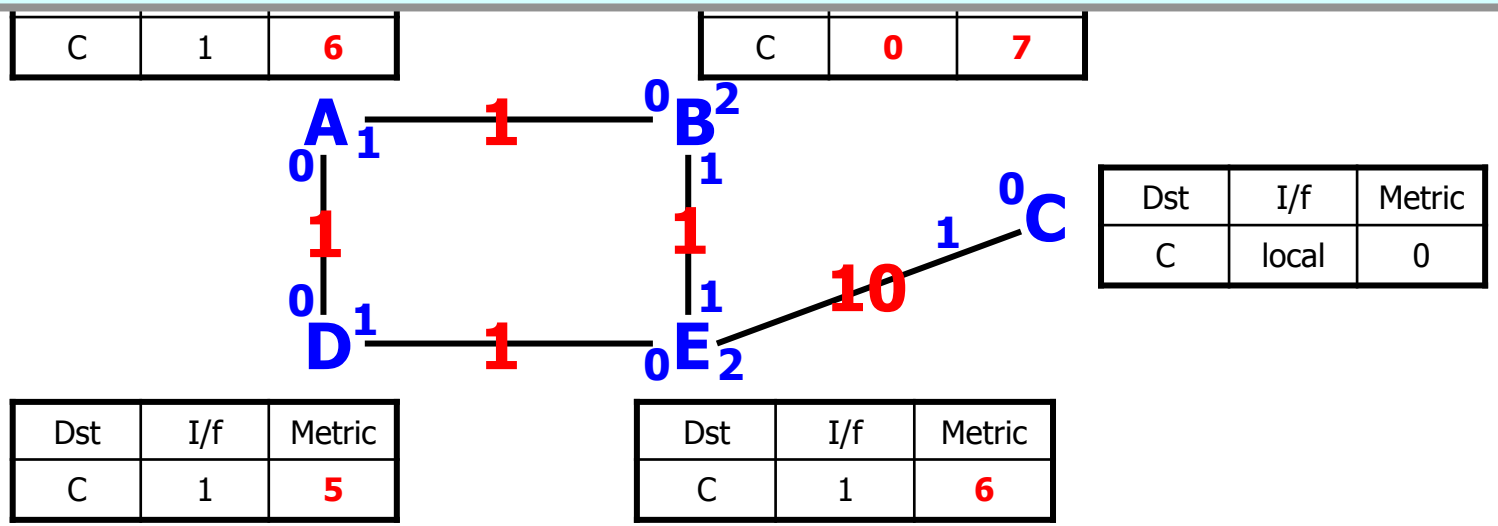
- Suppose A and E advertise next…
- …and B advertises next…
- …and A advertises next…



Bouncing (VII)

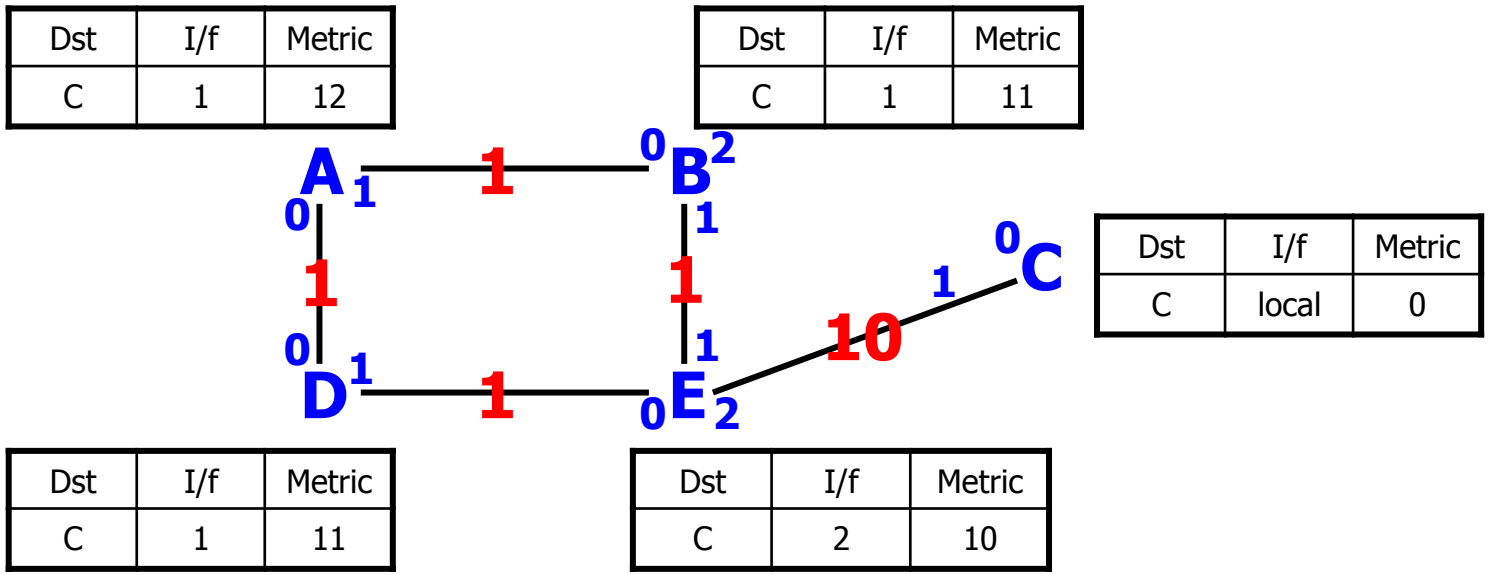
- Suppose A and E advertise next…
- …and B advertises next

And so on…



Bouncing (VIII)

- Long, painful convergence process, details dependent on message ordering
- Transient loops
- Eventually, converged state:



Outline

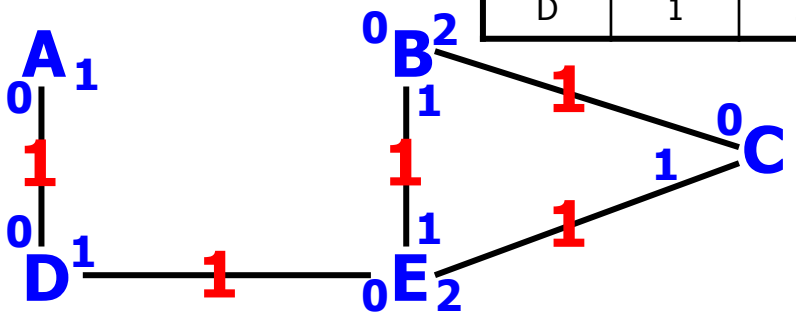
- Routing Problem Definition
- Routing in Practice: traceroute Examples
- Definitions: Hosts, Routers, Interfaces, Subnets
- Shortest-Path Routing
- Routing Tables
- Distance Vector Algorithm
- Pathologies: Bouncing and Counting to Infinity
- Optimizations: Split Horizon and Poison Reverse
- War Story: Synchronization of Routing Messages

Counting to Infinity (I)

- Converged after link (A, B) breaks
- Suppose (D, E) now breaks

Dst	I/f	Metric
A	local	0
B	0	3
D	0	1
E	0	2
C	0	3

Dst	I/f	Metric
B	local	0
A	1	3
C	2	1
E	1	1
D	1	2



Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	1	2
C	1	2

Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	0	2

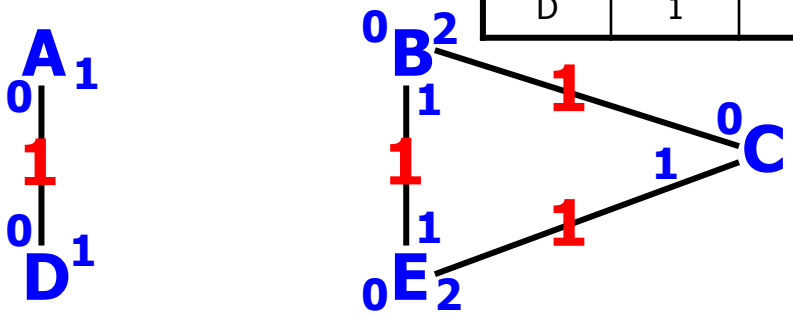
Dst	I/f	Metric
C	local	0
B	0	1
E	1	1
A	1	3
D	1	2

Counting to Infinity (I)

- Converged after link (A, B) breaks
- Suppose (D, E) now breaks

Dst	I/f	Metric
A	local	0
B	0	3
D	0	1
E	0	2
C	0	3

Dst	I/f	Metric
B	local	0
A	1	3
C	2	1
E	1	1
D	1	2



Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	1	2
C	1	2

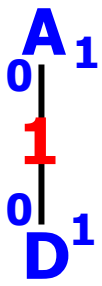
Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	0	2

Dst	I/f	Metric
C	local	0
B	0	1
E	1	1
A	1	3
D	1	2

Counting to Infinity (II)

- Network partitioned
- Focus on {A, D} partition

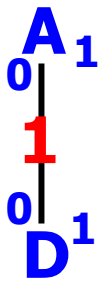
Dst	I/f	Metric
A	local	0
B	0	3
D	0	1
E	0	2
C	0	3



Dst	I/f	Metric
D	local	0
A	0	1
E	1	**Inf**
B	1	**Inf**
C	1	**Inf**

Counting to Infinity (II)

Dst	I/f	Metric
A	local	0
B	0	3
D	0	1
E	0	2
C	0	3



- Network partitioned
- Focus on {A, D} partition
- Suppose sequence of events:
  - D notices link failure
  - A advertises its routing table
- Loop for {B, C, E} between A and D!
- How long will loop persist?

Dst	Metric
A	1
B	4
D	2
E	3
C	4

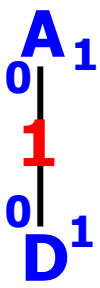
+

Dst	I/f	Metric
D	local	0
A	0	1
E	1	**Inf**
B	1	**Inf**
C	1	**Inf**

→

Dst	I/f	Metric
D	local	0
A	0	1
E	**0**	**3**
B	**0**	**4**
C	**0**	**4**

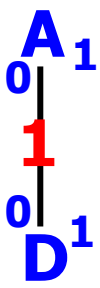
Counting to Infinity (III)



Dst	I/f	Metric
A	local	0
B	0	5
D	0	1
E	0	4
C	0	5

Dst	I/f	Metric
D	local	0
A	0	1
E	0	3
B	0	4
C	0	4

Counting to Infinity (III)



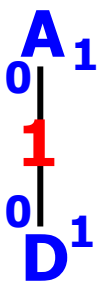
Dst	I/f	Metric
A	local	0
B	0	5
D	0	1
E	0	4
C	0	5

Dst	I/f	Metric
A	local	0
B	0	7
D	0	1
E	0	6
C	0	7

Dst	I/f	Metric
D	local	0
A	0	1
E	0	3
B	0	4
C	0	4

Dst	I/f	Metric
D	local	0
A	0	1
E	0	5
B	0	6
C	0	6

Counting to Infinity (III)



Dst	I/f	Metric
A	local	0
B	0	5
D	0	1
E	0	4
C	0	5

Dst	I/f	Metric
A	local	0
B	0	7
D	0	1
E	0	6
C	0	7

...

Dst	I/f	Metric
A	local	0
B	0	**Inf**
D	0	1
E	0	**Inf**
C	0	**Inf**

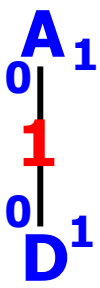
Dst	I/f	Metric
D	local	0
A	0	1
E	0	3
B	0	4
C	0	4

Dst	I/f	Metric
D	local	0
A	0	1
E	0	5
B	0	6
C	0	6

...

Dst	I/f	Metric
D	local	0
A	0	1
E	0	**Inf**
B	0	**Inf**
C	0	**Inf**

Counting to Infinity (III)



Dst	I/f	Metric
A	local	0
B	0	5
D	0	1
E	0	4
C	0	5

Dst	I/f	Metric
A	local	0
B	0	7
D	0	1
E	0	6
C	0	7

...

Dst	I/f	Metric
A	local	0
B	0	**Inf**
D	0	1
E	0	**Inf**
C	0	**Inf**

Dst	I/f	Metric
D	local	0
A	0	1
E	0	3
B	0	4
C	0	4

Dst	I/f	Metric
D	local	0
A	0	1
E	0	5
B	0	6
C	0	6

...

Dst	I/f	Metric
D	local	0
A	0	1
E	0	**Inf**
B	0	**Inf**
C	0	**Inf**

- Each advertisement increments metrics for partitioned destinations by one
- **Loop persists until count reaches infinity!**

Outline

- Routing Problem Definition
- Routing in Practice: traceroute Examples
- Definitions: Hosts, Routers, Interfaces, Subnets
- Shortest-Path Routing
- Routing Tables
- Distance Vector Algorithm
- Pathologies: Bouncing and Counting to Infinity
- Optimizations: Split Horizon and Poison Reverse
- War Story: Synchronization of Routing Messages

Split Horizon

- Bouncing and counting to infinity cause slow convergence, create loops
- Consider link (A, B), destination D
- A's next hop toward D is B
- Split Horizon: clearly, B should never choose A as next hop toward D
  - Intuition: A should never announce to B a path with short distance to D!

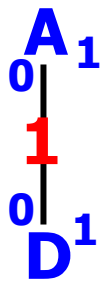
Split Horizon with Poison Reverse

- Again, consider link (A, B), destination D
- A's next hop toward D is B

- More generally: routers should announce different routing tables to different neighbors
- Split horizon: don't announce route for destination D on interface used as next hop toward D!
- Poison Reverse (optional): A announces to B its distance to D is infinity!

Example:
Split Horizon and Poison Reverse

Dst	I/f	Metric
A	local	0
B	0	3
D	0	1
E	0	2
C	0	3

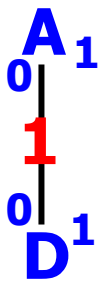


Dst	I/f	Metric
D	local	0
A	0	1
E	1	Inf
B	1	Inf
C	1	Inf

- Same example as counting to infinity: {A, D} partitioned

Example:
Split Horizon and Poison Reverse

Dst	I/f	Metric
A	local	0
B	0	3
D	0	1
E	0	2
C	0	3



- Same example as counting to infinity: {A, D} partitioned
- D detects link break, A announces first
- No loop, immediate convergence after one advertisement!

Dst	Metric
A	1
B	**Inf**
D	**Inf**
E	**Inf**
C	**Inf**

+

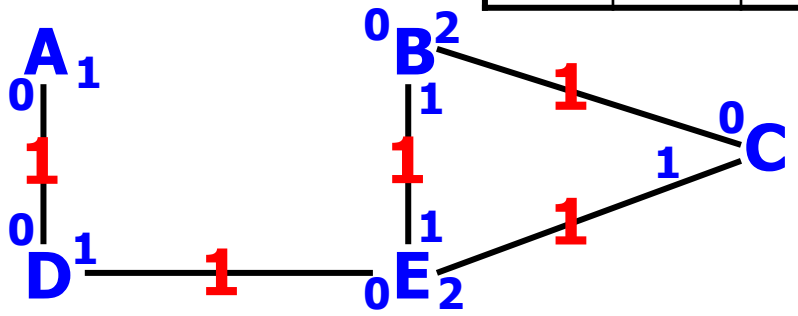
Dst	I/f	Metric
D	local	0
A	0	1
E	1	Inf
B	1	Inf
C	1	Inf

→

Dst	I/f	Metric
D	local	0
A	0	1
E	1	**Inf**
B	1	**Inf**
C	1	**Inf**

Limitations: Split Horizon and Poison Reverse

- Consider same example, but {B, C, E} partition
- Link (A, B) already failed, routing has converged
- Now link (D, E) fails
- Consider only destination D



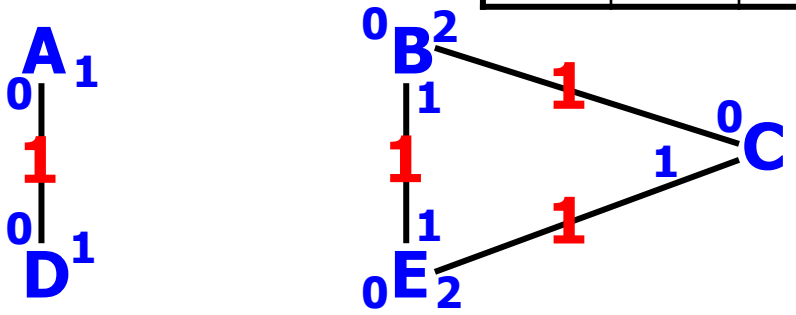
Dst	I/f	Metric
D	1	2

Dst	I/f	Metric
D	1	2

Dst	I/f	Metric
D	0	1

Limitations: Split Horizon and Poison Reverse

- Consider same example, but {B, C, E} partition
- Link (A, B) already failed, routing has converged
- Now link (D, E) fails
- Consider only destination D



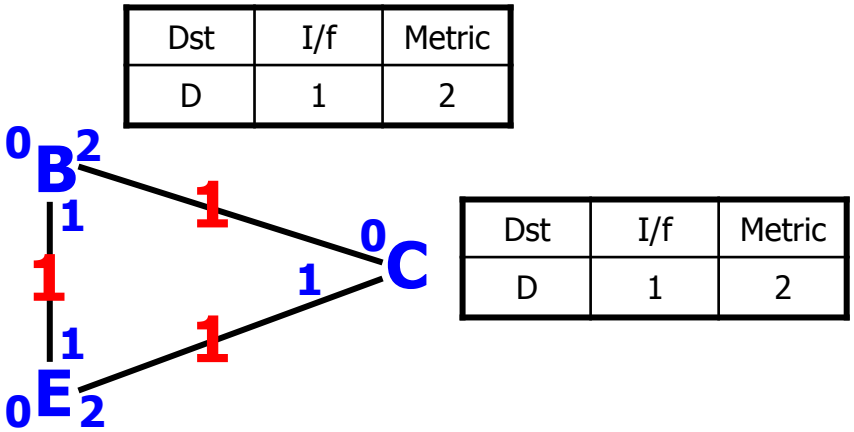
Dst	I/f	Metric
D	1	2

Dst	I/f	Metric
D	1	2

Dst	I/f	Metric
D	0	1

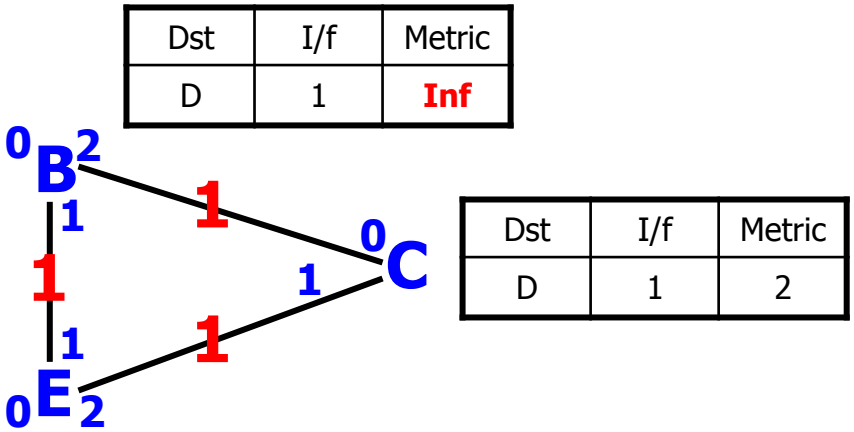
Limitations (II): Split Horizon and Poison Reverse

- E notices failed link, updates local table



Limitations (III): Split Horizon and Poison Reverse

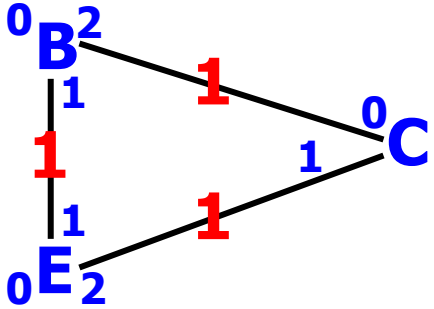
- E advertises its new table
  - Suppose advertisement reaches B, but not C



Limitations (IV): Split Horizon and Poison Reverse

- C advertises its table, with split horizon and poison reverse

Dst	I/f	Metric
D	1	Inf



Dst	I/f	Metric
D	1	2

Dst	I/f	Metric
D	0	Inf

Limitations (IV): Split Horizon and Poison Reverse

- C advertises its table, with split horizon and poison reverse

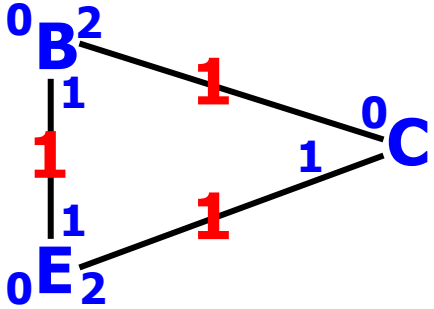
Dst	I/f	Metric
D	1	Inf

+

Dst	Metric
D	3

→

Dst	I/f	Metric
D	**2**	**3**



Dst	I/f	Metric
D	1	2

Dst	I/f	Metric
D	0	Inf

Limitations (IV): Split Horizon and Poison Reverse

- C advertises its table, with split horizon and poison reverse

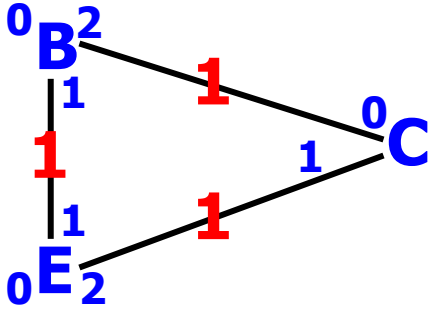
Dst	I/f	Metric
D	1	Inf

+

Dst	Metric
D	3

→

Dst	I/f	Metric
D	**2**	**3**



Dst	I/f	Metric
D	1	2

Dst	I/f	Metric
D	0	Inf

+

Dst	Metric
D	Inf

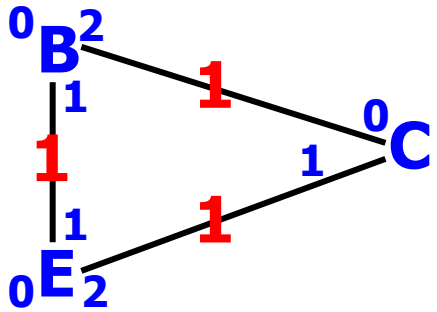
→

Dst	I/f	Metric
D	0	**Inf**

Limitations (V): Split Horizon and Poison Reverse

- B advertises its routing table, with split horizon and poison reverse

Dst	I/f	Metric
D	2	3

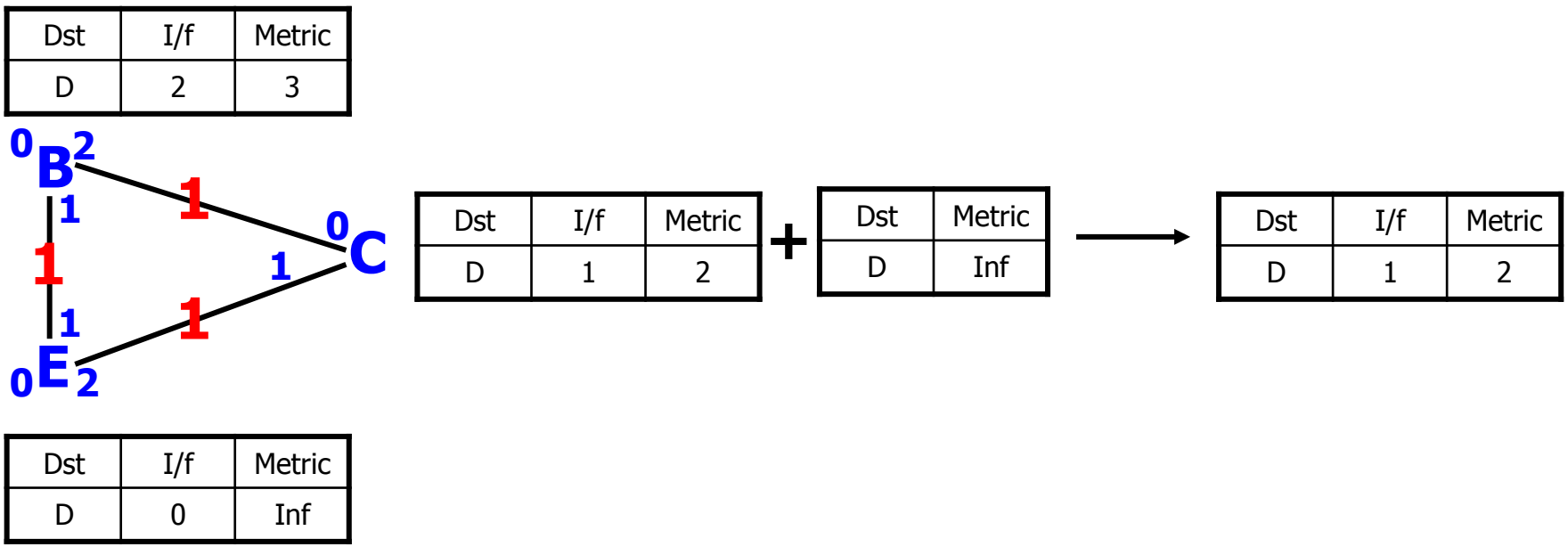


Dst	I/f	Metric
D	1	2

Dst	I/f	Metric
D	0	Inf

Limitations (V): Split Horizon and Poison Reverse

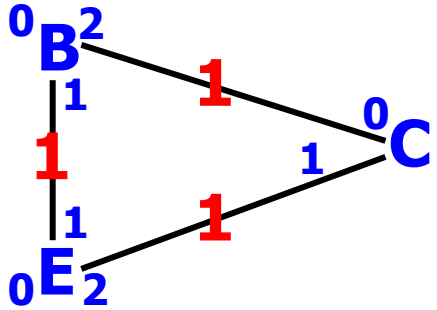
- B advertises its routing table, with split horizon and poison reverse



Limitations (V): Split Horizon and Poison Reverse

- B advertises its routing table, with split horizon and poison reverse

Dst	I/f	Metric
D	2	3



Dst	I/f	Metric
D	1	2

+

Dst	Metric
D	Inf

→

Dst	I/f	Metric
D	1	2

Dst	I/f	Metric
D	0	Inf

+

Dst	Metric
D	4

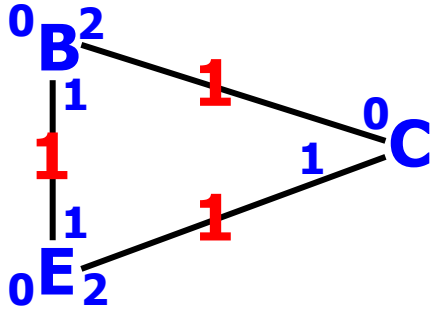
→

Dst	I/f	Metric
D	**1**	**4**

Limitations (V): Split Horizon and Poison Reverse

- B advertises its routing table, with split horizon and poison reverse
- For destination D, loop {C → E → B → C}!
  - resolved only by counting to infinity

Dst	I/f	Metric
D	2	3



Dst	I/f	Metric
D	1	2

+

Dst	Metric
D	Inf

→

Dst	I/f	Metric
D	1	2

Dst	I/f	Metric
D	0	Inf

+

Dst	Metric
D	4

→

Dst	I/f	Metric
D	**1**	**4**

Outline

- Routing Problem Definition
- Routing in Practice: traceroute Examples
- Definitions: Hosts, Routers, Interfaces, Subnets
- Shortest-Path Routing
- Routing Tables
- Distance Vector Algorithm
- Pathologies: Bouncing and Counting to Infinity
- Optimizations: Split Horizon and Poison Reverse
- War Story: Synchronization of Routing Messages

Symptom: Periodic Severe Packet Loss

- 1992: Every 30 seconds, for several seconds on end, 50 to 85% of packets passing through group of Internet routers dropped!
- RIP, a distance vector routing protocol, sends updates every 30 seconds
- Could distance vector routing be the culprit?

Timers and DV Route Updates

- When a timer expires, router prepares update packets containing current table, sends them
- If update packets arrive from neighbor while preparing own update packets, process them before resetting timer
- Reset timer, using interval [P – r, P + r]
  - P: desired update interval
  - r: uniform random jitter component
- Between timer expirations, process incoming updates immediately
- Note that timer not reset until after processing inbound messages: timing of one router's updates influenced by timing of other routers' updates!

Emergent Behavior: Synchronization of Route Updates

- Initially, routers all send multi-packet updates at random times
- "Collision"
  - Update from B arrives at A while A is preparing and sending its own update
  - A's timer not reset until A finishes sending its update **and** processing update from B
  - Similarly, update from A arrives at B while B is preparing and sending its own update…
  - A and B now reset their timers roughly concurrently!
- Collisions persist so long as A and B's update generation intervals continue to overlap significantly
- Update generation intervals lengthen each time a router "joins" a collision
- Result: routers eventually all synchronize to send all updates at same time!

Avoiding Routing Update Synchronization

- [Floyd, Jacobson 1993]: random jitter should be 50% of update interval to avoid synchronization
  - in [P – r, P + r] model, P = 30 → r = 15
  - update interval random in [15, 45] seconds

Summary: Distance Vector Routing

- DV algorithm: periodically dump routing table contents to neighbors
- Convergence: after topology change, point when routing tables stop changing
- Pro:
  - simple
  - finds correct routes after topology changes
- Con:
  - bouncing, counting to infinity cause loops
  - slow to converge after some topology changes
  - split horizon, poison reverse only partial solutions