Lecture 9: Distance Vector Routing

Kyle Jamieson

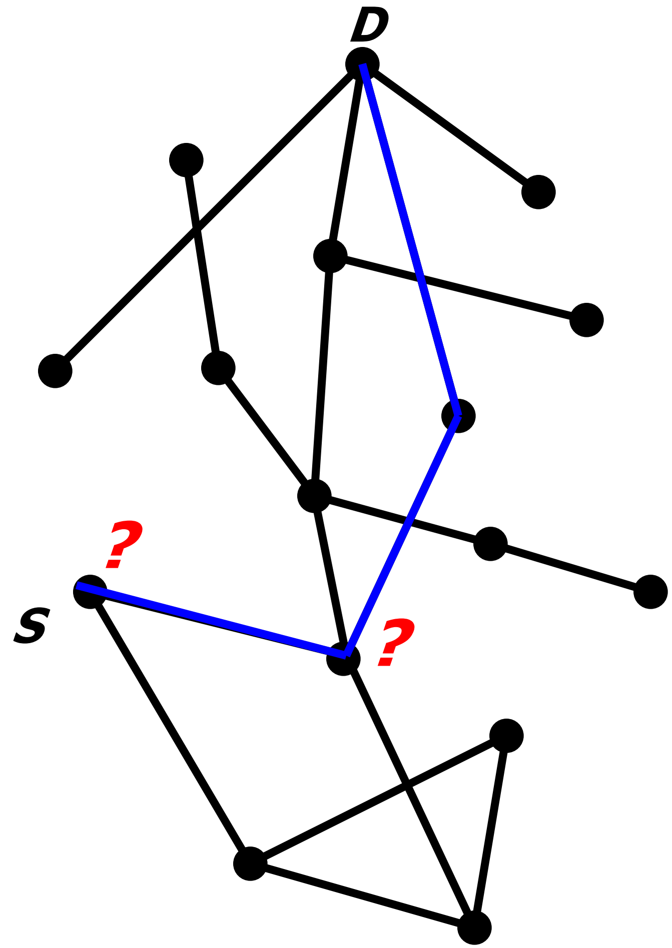
COS 461: Computer Networks

Outline

- Routing Problem Definition
- 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



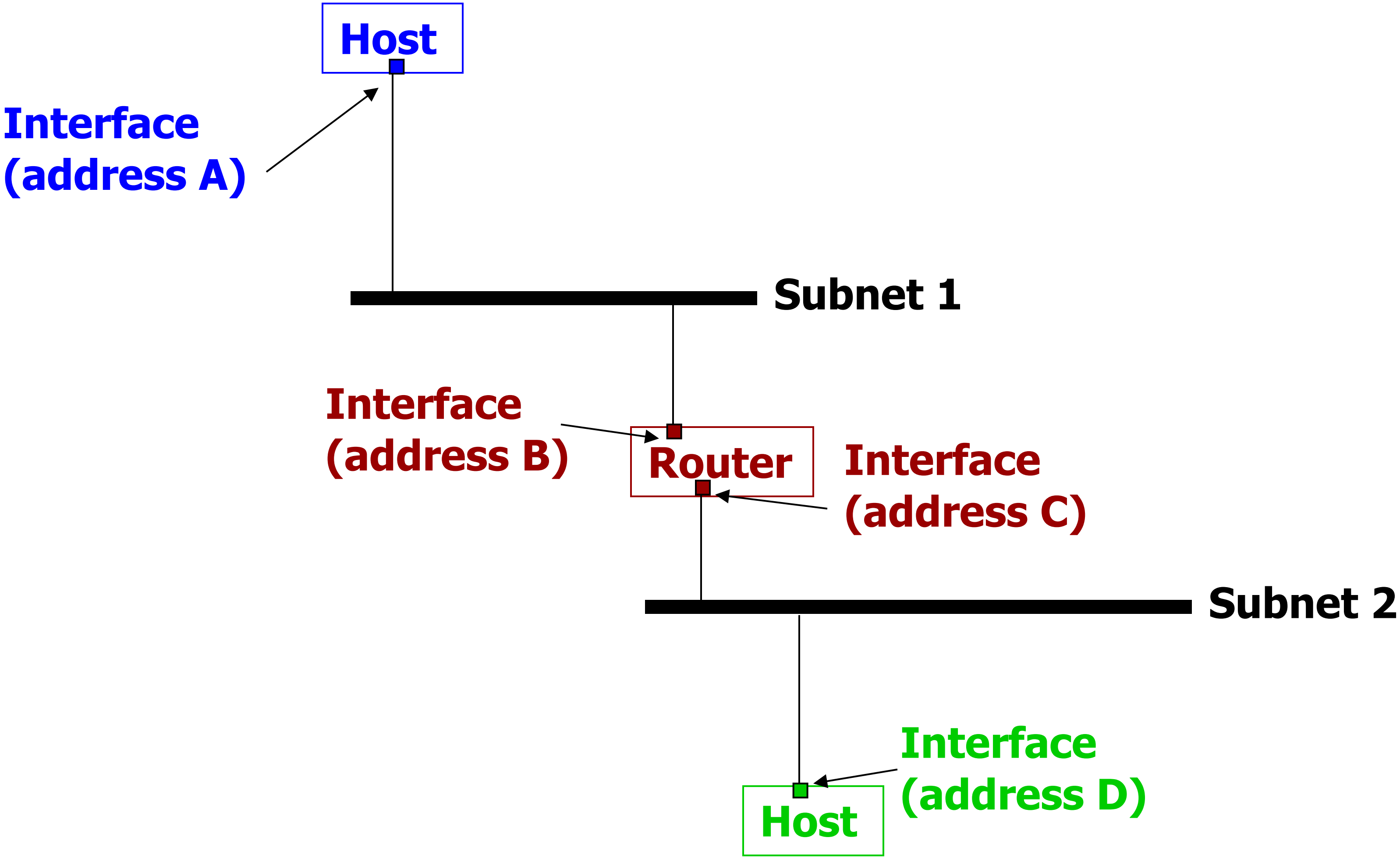
Routing on Changing Networks

- 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

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
- 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)

- 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
```

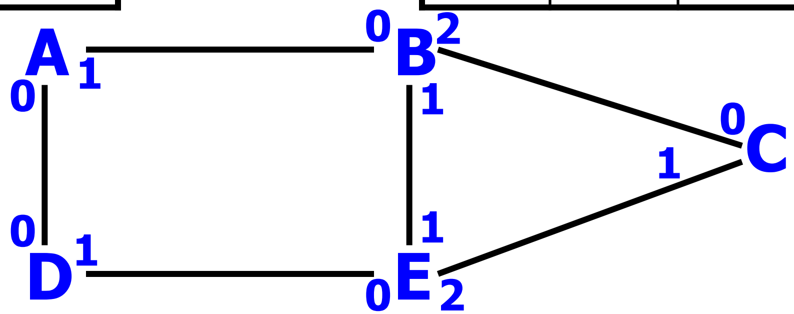
Distance Vector: Example

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

Dst	I/f	Metric
A	local	0

Dst	Metric
A	0

Dst	I/f	Metric
B	local	0



Dst	I/f	Metric
C	local	0

Dst	I/f	Metric
D	local	0

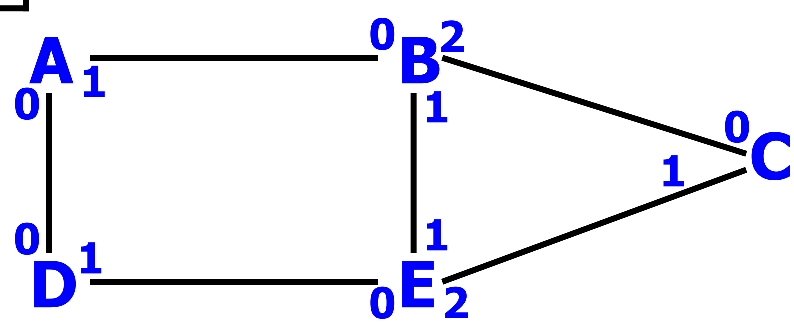
Dst	I/f	Metric
E	local	0

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

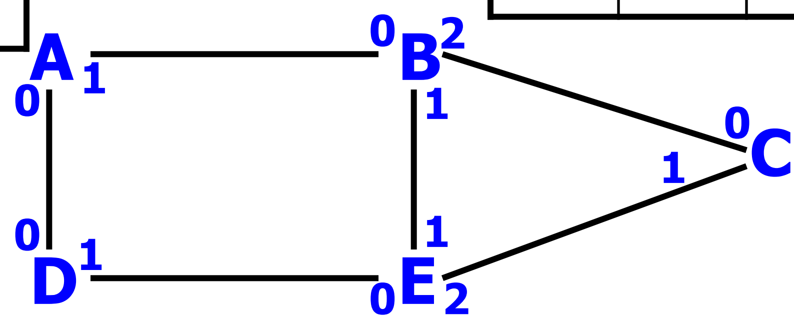
Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1

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
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

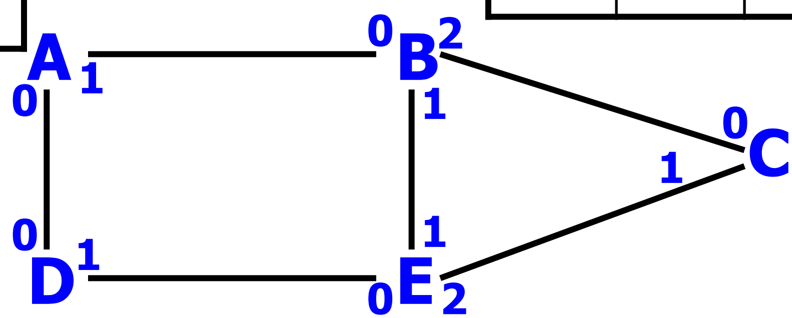
Distance Vector: Iteration 2

- 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
C	local	0
B	0	1
E	1	1
A	0	2
D	1	2



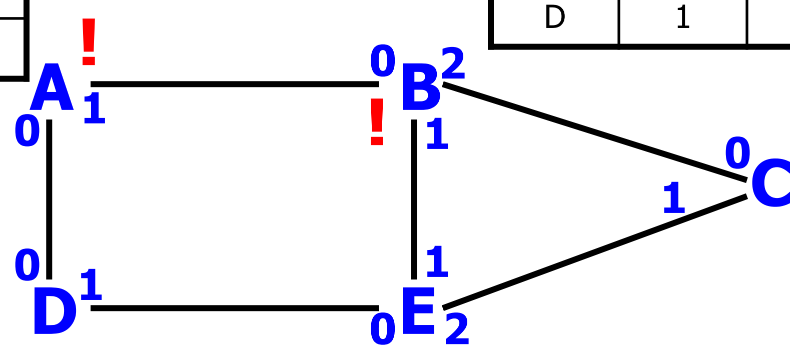
Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	1	2
C	1	2

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

Link Failure (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

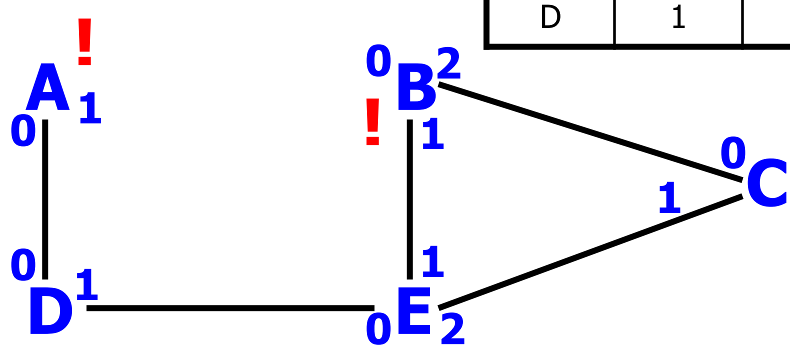
Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	1	2
C	1	2

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	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
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

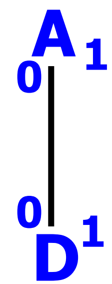
Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	1	2

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

→ (no change)

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

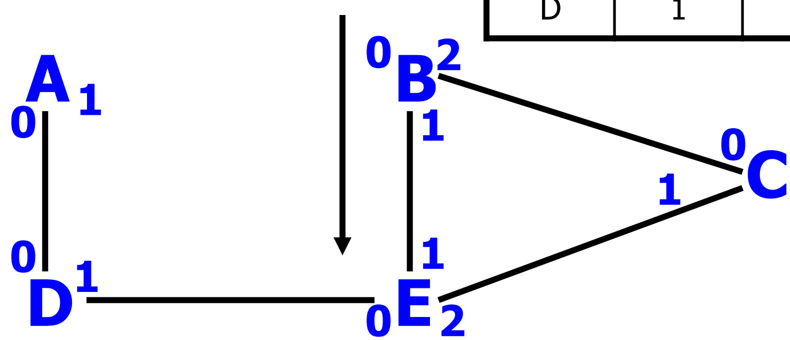
Link Failure (IV)

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

Dst	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

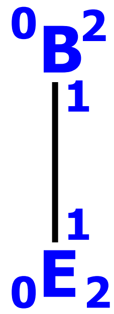
Dst	I/f	Metric
C	local	0
B	0	1
E	1	1
A	0	2
D	1	2



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

→

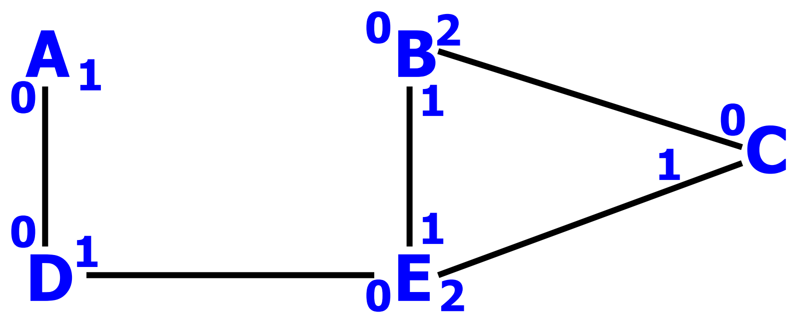
Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	1	**Inf**

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
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

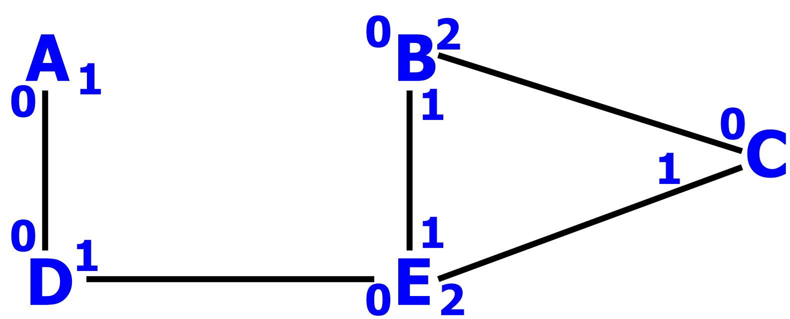
Dst	I/f	Metric
E	local	0
D	0	1
B	1	1
C	2	1
A	1	**Inf**

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



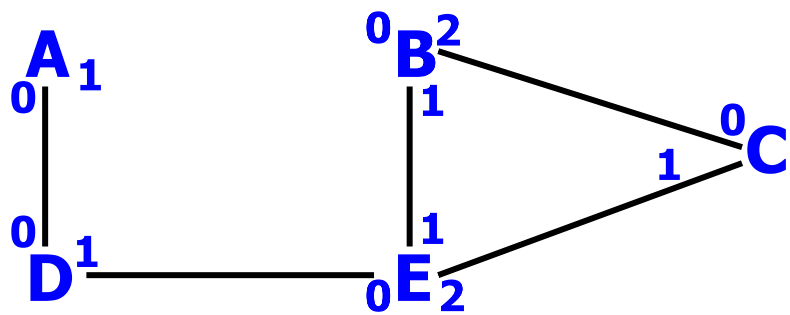
Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	1	2
C	1	2

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
C	local	0
B	0	1
E	1	1
A	**1**	**3**
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**

Outline

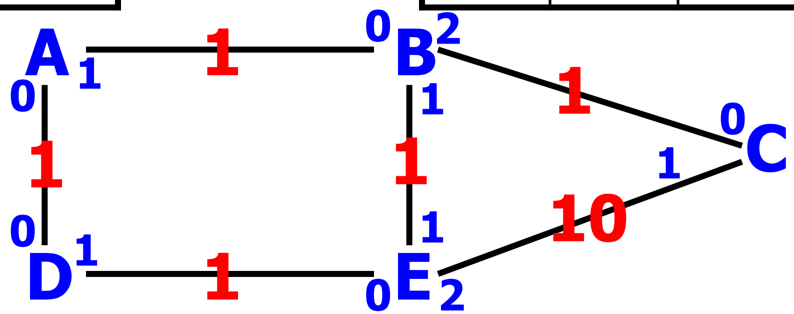
- Routing Problem Definition
- Definitions: Hosts, Routers, Interfaces, Subnets
- Shortest-Path Routing
- Routing Tables
- Distance Vector Algorithm
- Pathologies: Bouncing and Counting to Infinity
- Optimizations: Split Horizon and Poison Reverse
- War Story: Synchronization of Routing Messages

Bouncing (I)

- Consider same network, (C, E) has metric 10; all others 1

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

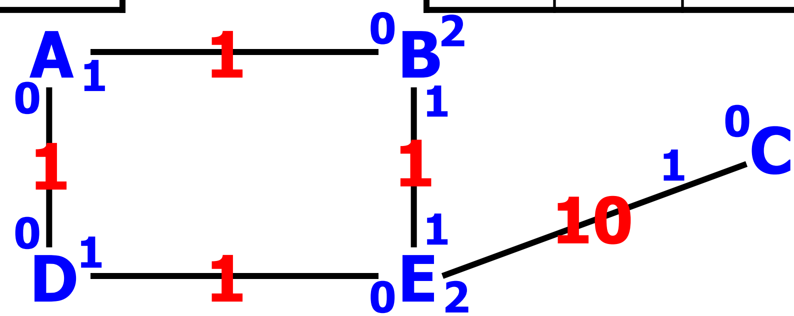
Dst	I/f	Metric
C	1	2

Bouncing (II)

- Suppose A advertises its table first…

Dst	I/f	Metric
C	1	2

Dst	I/f	Metric
C	2	**Inf**



Dst	I/f	Metric
C	local	0

Dst	I/f	Metric
C	1	3

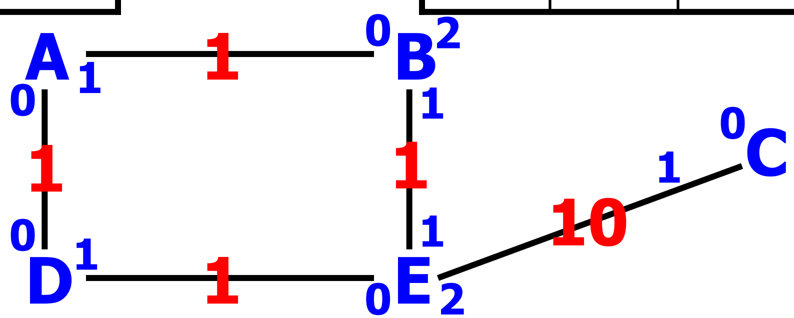
Dst	I/f	Metric
C	1	2

Bouncing (III)

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

Dst	I/f	Metric
C	1	2

Dst	I/f	Metric
C	**0**	**3**



Dst	I/f	Metric
C	local	0

Dst	I/f	Metric
C	1	3

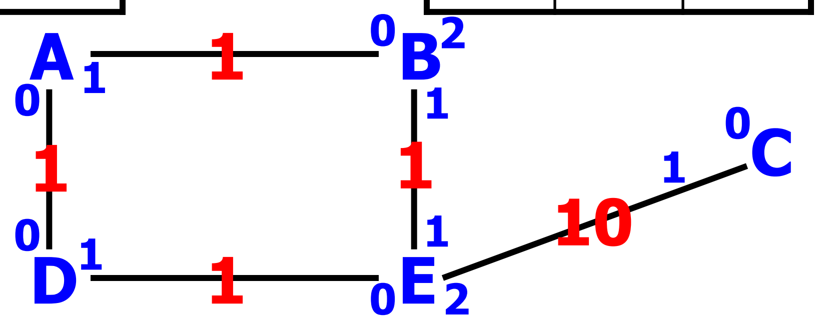
Dst	I/f	Metric
C	1	2

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!

Dst	I/f	Metric
C	1	4

Dst	I/f	Metric
C	0	3



Dst	I/f	Metric
C	local	0

Dst	I/f	Metric
C	1	3

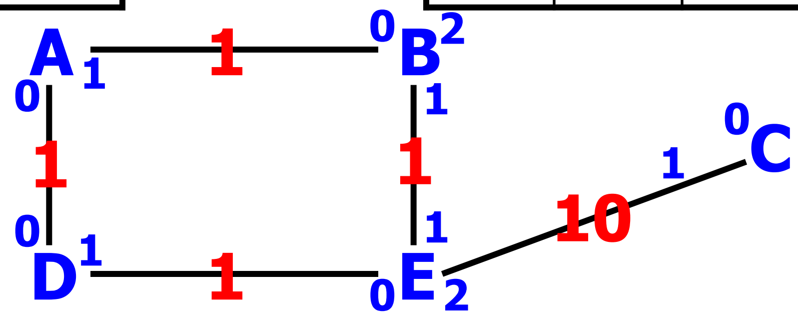
Dst	I/f	Metric
C	1	4

Bouncing (V)

- Suppose A and E advertise next…

Dst	I/f	Metric
C	1	4

Dst	I/f	Metric
C	0	5



Dst	I/f	Metric
C	local	0

Dst	I/f	Metric
C	1	5

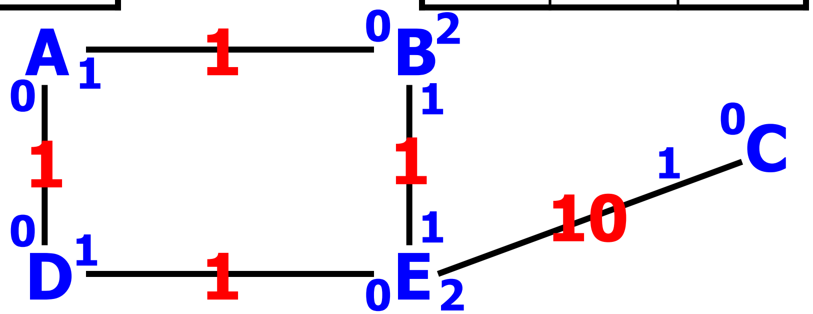
Dst	I/f	Metric
C	1	4

Bouncing (VI)

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

Dst	I/f	Metric
C	1	**6**

Dst	I/f	Metric
C	**0**	**5**



Dst	I/f	Metric
C	local	0

Dst	I/f	Metric
C	1	**5**

Dst	I/f	Metric
C	1	**6**

Bouncing (VII)

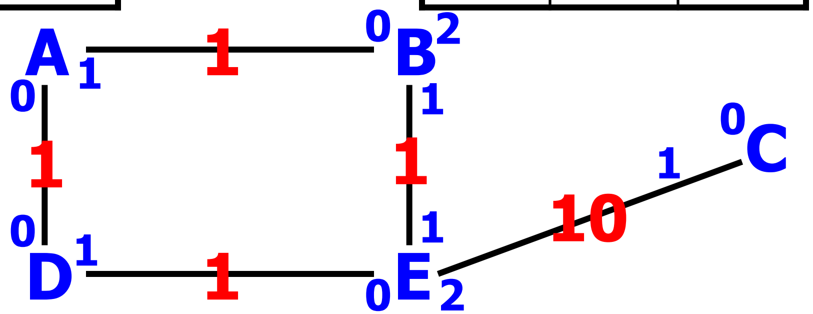
- Suppose A and E advertise next…
- …and B advertises next…
- …and A advertises next…

Router A:

Dst	I/f	Metric
C	1	6

Router B:

Dst	I/f	Metric
C	0	7



Router C:

Dst	I/f	Metric
C	local	0

Router D:

Dst	I/f	Metric
C	1	5

Router E:

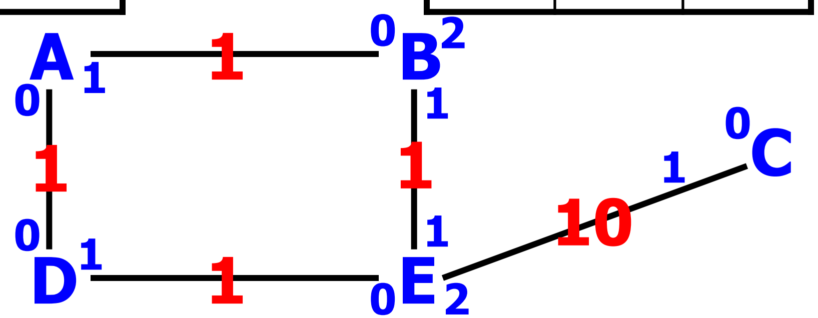
Dst	I/f	Metric
C	1	6

Bouncing (VIII)

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

Dst	I/f	Metric
C	1	12

Dst	I/f	Metric
C	1	11



Dst	I/f	Metric
C	local	0

Dst	I/f	Metric
C	1	11

Dst	I/f	Metric
C	2	10

Outline

- Routing Problem Definition
- 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

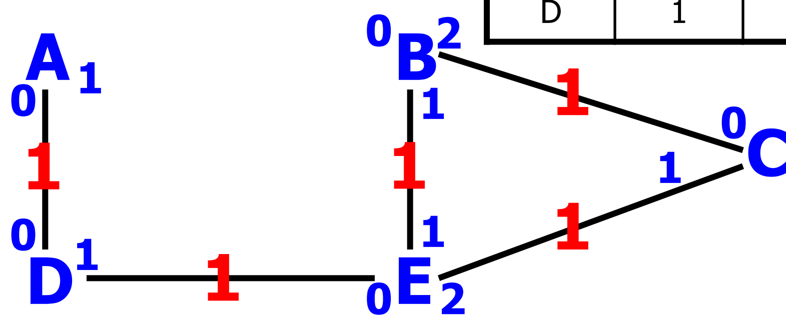
Counting to Infinity (I)

- 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
C	local	0
B	0	1
E	1	1
A	1	3
D	1	2



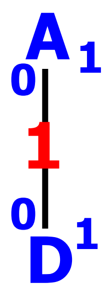
Dst	I/f	Metric
D	local	0
A	0	1
E	1	1
B	1	2
C	1	2

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

Counting to Infinity (II)

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



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**

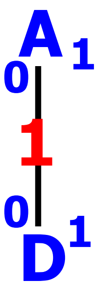
Counting to Infinity (III)

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

Dst	I/f	Metric
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
- 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

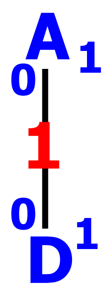
- Again, consider link (A, B), destination D
- A's next hop toward D is B

- 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

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

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



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

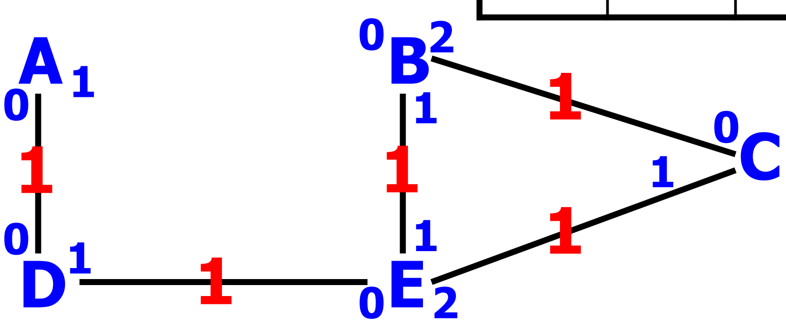
→

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



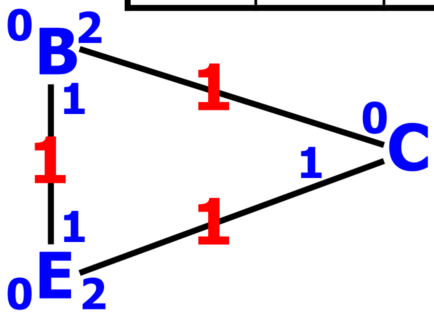
Dst	I/f	Metric
D	1	2

Dst	I/f	Metric
D	0	1

Limitations (II): Split Horizon and Poison Reverse

- E notices failed link, updates local table

Dst	I/f	Metric
D	1	2



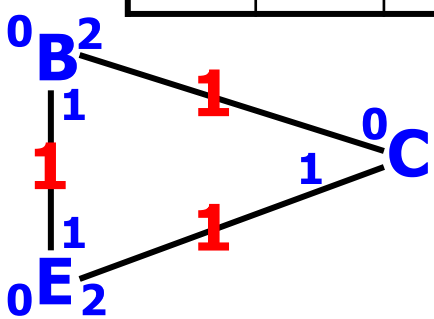
Dst	I/f	Metric
D	1	2

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

Limitations (III): Split Horizon and Poison Reverse

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

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

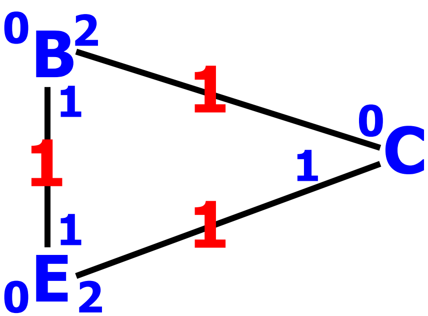
Dst	I/f	Metric
D	1	Inf

+

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

+

Dst	Metric
D	Inf

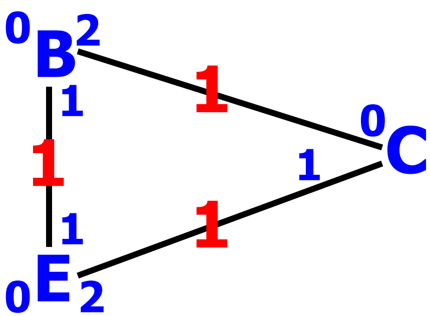
→

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
- 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
- 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
- If update packets arrive from neighbor while preparing own update packets, process them before sending own update packets
- Send own update packets
- Reset timer interval [P – r, P + r]
  - P: desired update interval
  - r: uniform random jitter component

Emergent Behavior: Synchronization of Route Updates

- Initially, routers all send updates at random times
- "Collision": update from B arrives at A while A is preparing its own update
  - Timer not reset until A finishes sending update
  - Result: longer period between updates by A
  - So higher probability update arrives from some other router C before timer reset

- If triggered update arrives from some other router before timer expires, A immediately prepares and sends update, without waiting for timer to expire

- Result: routers eventually all synchronize to send all updates at same time!

Avoiding Routing Update Synchronization

- Floyd and Jacobson: 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