• Link-state Routing
• Distance-vector Routing
Problems to Solve

How do we route (and address) scalably, while dealing with issues of policy and economy?

How do we transport data scalably, while dealing with varying application demands?

How do we adapt new applications and technologies to an inflexible architecture?
Problems to Solve

How do we **route** (and address) **scalably**, while dealing with issues of policy and economy?

How do we **transport** data **scalably**, while dealing with varying application demands?

How do we **adapt** new applications and technologies to an inflexible architecture?
goal of a routing protocol: allow each switch to know, for every node dst in the network, a minimum-cost route to dst
**goal of a routing protocol:** build a routing table at each switch, such that `routing_table[dst]` contains a **minimum-cost route** to `dst`
Distributed Routing

1. Nodes learn about their neighbors via the HELLO protocol
Distributed Routing

1. Nodes learn about their neighbors via the HELLO protocol

2. Nodes learn about other reachable nodes via advertisements
Distributed Routing

1. Nodes learn about their neighbors via the **HELLO** protocol

2. Nodes learn about other reachable nodes via advertisements

3. Nodes determine the minimum-cost routes (of the routes they know about)
Distributed Routing

1. Nodes learn about their neighbors via the HELLO protocol

2. Nodes learn about other reachable nodes via advertisements

3. Nodes determine the minimum-cost routes (of the routes they know about)

All of these steps happen periodically, which allows the routing protocol to detect and respond to failures, and adapt to other changes in the network.
Link-state Routing
disseminate full topology information so that nodes can run a shortest-path algorithm
Link-state Routing
disseminate full topology information so that nodes can run a shortest-path algorithm
Link-state Routing
disseminate full topology information so that nodes can run a shortest-path algorithm

A node's advertisements contain a list of its neighbors and its link costs to those nodes
Link-state Routing
disseminate full topology information so that nodes can run a shortest-path algorithm

A node's advertisements contain a list of its neighbors and its link costs to those nodes

From A: [(B,19),(C,7)]
From B: [(A,19),(C,11),(D,4)]
From C: [(A,7),(B,11),(D,16),(E,5)]
From D: [(B,4),(C,16),(E,13)]
From E: [(C,5),(D,13)]
Link-state Routing
disseminate full topology information so that nodes can run a shortest-path algorithm

A node's advertisements contain a list of its neighbors and its link costs to those nodes

A node effectively sends advertisements to every other node (via flooding)

From A: [(B,19),(C,7)]
From B: [(A,19),(C,11),(D,4)]
From C: [(A,7),(B,11),(D,16),(E,5)]
From D: [(B,4),(C,16),(E,13)]
From E: [(C,5),(D,13)]
Link-state Routing
disseminate full topology information so that nodes can run a shortest-path algorithm
Link-state Routing
disseminate full topology information so that nodes can run a shortest-path algorithm
Link-state Routing
disseminate full topology information so that nodes can run a shortest-path algorithm
Link-state Routing
disseminate full topology information so that nodes can run a shortest-path algorithm

A: Self, 0

A

B

C

D

19

11

7

16

5

4

13
Link-state Routing
disseminate full topology information so that nodes can run a shortest-path algorithm

A: Self, 0
B: A->B, 19
C: A->C, 7
Link-state Routing
disseminate full topology information so that nodes can run a shortest-path algorithm

A: Self, 0
B: A->B, 19
C: A->C, 7
Link-state Routing

disseminate full topology information so that nodes can run a shortest-path algorithm

A: Self, 0
B: A->B, 19
C: A->C, 7

6.033 | spring 2020 | lacurts@mit.edu
Link-state Routing

disseminate full topology information so that nodes can run a shortest-path algorithm

A: Self, 0
B: A→B, 19
C: A→C, 7
D: A→C, 23
E: A→C, 12
Link-state Routing
disseminate full topology information so that nodes can run
a shortest-path algorithm

A: Self, 0
B: A->C, 18
C: A->C, 7
D: A->C, 23
E: A->C, 12
Link-state Routing
disseminate full topology information so that nodes can run a shortest-path algorithm

A: Self, 0
B: A→C, 18
C: A→C, 7
D: A→C, 23
E: A→C, 12
Link-state Routing

disseminate full topology information so that nodes can run a shortest-path algorithm

A: Self, 0
B: A->C, 18
C: A->C, 7
D: A->C, 23
E: A->C, 12
Link-state Routing
disseminate full topology information so that nodes can run a shortest-path algorithm
Link-state Routing
disseminate full topology information so that nodes can run
a shortest-path algorithm

A: Self, 0
B: A→C, 18
C: A→C, 7
D: A→C, 23
E: A→C, 12
Link-state Routing

disseminate full topology information so that nodes can run a shortest-path algorithm

A: Self, 0
B: A->C, 18
C: A->C, 7
D: A->C, 22
E: A->C, 12
Link-state Routing
disseminate full topology information so that nodes can run a shortest-path algorithm

A: Self, 0
B: A->C, 18
C: A->C, 7
D: A->C, 22
E: A->C, 12
**Link-state Routing**

disseminate full topology information so that nodes can run a shortest-path algorithm

A node's advertisements contain a list of its neighbors and its **link costs** to those nodes

A node effectively sends advertisements to every other node (via flooding)
Link-state Routing
disseminate full topology information so that nodes can run a shortest-path algorithm

A node’s advertisements contain a list of its neighbors and its link costs to those nodes

A node effectively sends advertisements to every other node (via flooding)

Because advertisements are flooded, link-state routing performs well when there are failures
Link-state Routing

disseminate full topology information so that nodes can run a shortest-path algorithm

A node’s advertisements contain a list of its neighbors and its link costs to those nodes

A node effectively sends advertisements to every other node (via flooding)

Because advertisements are flooded, link-state routing performs well when there are failures. However, the overhead of flooding limits scale.
Distance-vector Routing

disseminate information about the current costs to each node, rather than the actual topology
Distance-vector Routing
disseminate information about the current costs to each node, rather than the actual topology.

A node's advertisements contain a list of all the nodes it knows about and its current costs to those nodes.
Distance-vector Routing

disseminate information about the current costs to each node, rather than the actual topology.

A node’s advertisements contain a list of all the nodes it knows about and its current costs to those nodes.

A: Self, 0
B: A→B, 19
C: A→C, 7
Distance-vector Routing

disseminate information about the current costs to each node, rather than the actual topology

A node’s advertisements contain a list of all the nodes it knows about and its current costs to those nodes

A node sends advertisements only to its neighbors

A: Self, 0
B: A->B, 19
C: A->C, 7
Distance-vector Routing

disseminate information about the current costs to each node, rather than the actual topology.

A node's advertisements contain a list of all the nodes it knows about and its current costs to those nodes.

A node sends advertisements only to its neighbors.

From A: [(A,0), (B,19), (C,7)]
Distance-vector Routing

disseminate information about the current costs to each node, rather than the actual topology.
Distance-vector Routing

disseminate information about the current costs to each node, rather than the actual topology.

From A: [(A,0),(B,19),(C,7)]
From B: [(A,19),(B,0),(C,11),(D,4)]
From C: [(A,7),(B,11),(C,0),(D,16),(E,5)]
From D: [(B,4),(C,16),(D,0),(E,13)]
From E: [(C,5),(D,13),(E,0)]
Distance-vector Routing
disseminate information about the current *costs* to each node,
rather than the actual topology.
Distance-vector Routing
disseminate information about the current costs to each node, rather than the actual topology.

From A: [(A,0),(B,19),(C,7)]

From B: [(A,19),(B,0),(C,11),(D,4)]

From C: [(A,7),(B,11),(C,0),
(D,16),(E,5)]
Distance-vector Routing
disseminate information about the current costs to each node, rather than the actual topology

From A: [(A,0), (B,19), (C,7)]
From B: [(A,19), (B,0), (C,11), (D,4)]
From C: [(A,7), (B,11), (C,0), (D,16), (E,5)]
Distance-vector Routing
disseminate information about the current costs to each node,
rather than the actual topology

From A: [(A,0), (B,19), (C,7)]
From B: [(A,19), (B,0), (C,11), (D,4)]
From C: [(A,7), (B,11), (C,0), (D,16), (E,5)]
From A: [(A,0), (B,19), (C,7)]
Distance-vector Routing
disseminate information about the current costs to each node,
rather than the actual topology

From A: [(A,0), (B,19), (C,7)]
From B: [(A,19), (B,0), (C,11), (D,4)]
From C: [(A,7), (B,11), (C,0), (D,16), (E,5)]
From A: [(A,0), (B,19), (C,7)]
Distance-vector Routing
disseminate information about the current costs to each node, rather than the actual topology.

- From A: [(A,0), (B,19), (C,7)]
- From B: [(A,19), (B,0), (C,11), (D,4)]
- From C: [(A,7), (B,11), (C,0), (D,16), (E,5)]
- From D: [(A,19), (B,19), (C,11), (D,4)]
- From E: [(A,7), (B,11), (C,0), (D,13), (E,5)]

Diagram:
- A -> B: 19
- B -> C: 11
- B -> D: 4
- B -> E: 13
- C -> B: 7
- C -> D: 16
- C -> E: 5
- D -> C: 16
- E -> C: 5

From A: [(A,0), (B,19), (C,7)]
Distance-vector Routing
disseminate information about the current costs to each node, rather than the actual topology

From A: [(A,0),(B,19),(C,7)]

From B: [(A,19),(B,0),(C,11),(D,4)]

From C: [(A,7),(B,11),(C,0),(D,16),(E,5)]
Distance-vector Routing
disseminate information about the current *costs* to each node, rather than the actual topology
A sends advertisements at $t = 0, 10, 20, \ldots$; B sends advertisements at $t = 5, 15, 25, \ldots$
INFINITY

A sends advertisements at t=0, 10, 20,..; B sends advertisements at t=5, 15, 25,..

A: Self, 0
B: A->B, 1
C: A->B, 2

A: B->A, 1
B: Self, 0
C: B->C, 1
INFINITY

A sends advertisements at $t=0, 10, 20,..$; B sends advertisements at $t=5, 15, 25,..$

A: Self, 0      A: B->A, 1
B: A->B, 1      B: Self, 0
C: A->B, 2      C: B->C, 1

$t=9$: B<-C fails
A sends advertisements at $t=0, 10, 20,..$; B sends advertisements at $t=5, 15, 25,..$

A sends advertisements at $t=0, 10, 20,..$; B sends advertisements at $t=5, 15, 25,..$.

A: Self, 0  A: B→A, 1  B: Self, 0  C: None, inf
B: A→B, 1  B: Self, 0  C: None, inf
C: A→B, 2  B: Self, 0  C: None, inf

$t=9$: B→C fails
INFINITY

A sends advertisements at t=0, 10, 20,..; B sends advertisements at t=5, 15, 25,..

A: Self, 0  B: B->A, 1
B: A->B, 1  B: Self, 0
C: A->B, 2  C: None, inf

t=9: B<->C fails

t=10: B receives the following advertisement from A:
[ (A,0), (B,1), (C,2) ]
A sends advertisements at $t=0, 10, 20,..$; B sends advertisements at $t=5, 15, 25,..$

---

**A**

- **Self**, 0
- **A->B**, 1
- **A->B**, 2

**B**

- **A->B**, 1
- **Self**, 0
- **B->A**, 3

**C**

- **None**, inf
- **B->A**, 3

---

$t=9$: $B<->C$ fails

$t=10$: $B$ receives the following advertisement from $A$:

$[(A,0),(B,1),(C,2)]$
A sends advertisements at t=0, 10, 20,..; B sends advertisements at t=5, 15, 25,..

A: Self, 0  
B: A->B, 1  
C: A->B, 2

A: Self, 0  
B: A->B, 1  
C: None, inf

A: Self, 0  
B: B->A, 1  
C: B->A, 3  (2+1)

C

t=9: B<->C fails

t=10: B receives the following advertisement from A: 
[(A,0), (B,1), (C,2)]

t=15: A receives the following advertisement from B: 
[(A,1), (B,0), (C,3)]
A sends advertisements at $t=0$, 10, 20,..; B sends advertisements at $t=5$, 15, 25,..

$t=9$: $B \leftarrow C$ fails

$t=10$: $B$ receives the following advertisement from $A$: $[(A,0),(B,1),(C,2)]$

$t=15$: $A$ receives the following advertisement from $B$: $[(A,1),(B,0),(C,3)]$
A sends advertisements at \( t=0, 10, 20, \ldots \); B sends advertisements at \( t=5, 15, 25, \ldots \)

### t=9: \( B<->C \) fails

### t=10: B receives the following advertisement from A:

\[
[(A,0), (B,1), (C,2)]
\]

### t=15: A receives the following advertisement from B:

\[
[(A,1), (B,0), (C,3)]
\]

### t=20: B receives the following advertisement from A:

\[
[(A,0), (B,1), (C,4)]
\]
INFINITY

A sends advertisements at t=0, 10, 20,..; B sends advertisements at t=5, 15, 25,..

<table>
<thead>
<tr>
<th></th>
<th>A:</th>
<th>B:</th>
<th>C:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Self, 0</td>
<td>B-&gt;A, 1</td>
<td>None, inf</td>
</tr>
<tr>
<td>B</td>
<td>A-&gt;B, 1</td>
<td>Self, 0</td>
<td>C: B-&gt;A, 3 (2+1)</td>
</tr>
<tr>
<td>C</td>
<td>A-&gt;B, 2</td>
<td></td>
<td>t=9: B&lt;&gt;C fails</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A:</th>
<th>B:</th>
<th>C:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Self, 0</td>
<td>B-&gt;A, 1</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>A-&gt;B, 1</td>
<td>Self, 0</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>A-&gt;B, 1</td>
<td>B-&gt;A, 3</td>
<td>t=10: B receives the following advertisement from A: ([(A,0),(B,1),(C,2)])</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A:</th>
<th>B:</th>
<th>C:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Self, 0</td>
<td>B-&gt;A, 1</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>A-&gt;B, 1</td>
<td>Self, 0</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>A-&gt;B, 4</td>
<td>B-&gt;A, 3</td>
<td>t=15: A receives the following advertisement from B: ([(A,1),(B,0),(C,3)])</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A:</th>
<th>B:</th>
<th>C:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Self, 0</td>
<td>B-&gt;A, 1</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>A-&gt;B, 1</td>
<td>Self, 0</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>A-&gt;B, 4</td>
<td>B-&gt;A, 5</td>
<td>t=20: B receives the following advertisement from A: ([(A,0),(B,1),(C,4)])</td>
</tr>
</tbody>
</table>
A sends advertisements at t=0, 10, 20,..; B sends advertisements at t=5, 15, 25,..

continues until both costs to C are INFINITY
Split Horizon

Don’t send advertisements about a route to the node providing the route

A --- B

C

t=9: B<->C fails
Split Horizon

Don’t send advertisements about a route to the node providing the route

A: self, 0  
B: A→B, 1  
C: A→B, 2

A: B→A, 1  
B: self, 0  
C: none, inf

t=9: B→C fails
Split Horizon

Don’t send advertisements about a route to the node providing the route

A: Self, 0       A: B->A, 1
B: A->B, 1       B: Self, 0
C: A->B, 2       C: None, inf

$t=9$: B<->C fails

$t=10$: B receives the following advertisement from A:
        $[\langle A, 0 \rangle]$
Split Horizon

Don’t send advertisements about a route to the node providing the route

A:
- Self, 0
- A->B, 1
B:
- A->B, 1
- Self, 0
C:
- A->B, 2
- C: None, inf

A:
- Self, 0
- A->B, 1
B:
- A->B, 1
- Self, 0
C:
- A->B, 2
- C: None, inf

t=9: B<>C fails

t=10: B receives the following advertisement from A:

[(A,0)]
Split Horizon

Don’t send advertisements about a route to the node providing the route

A: Self, 0  A: B->A, 1
B: A->B, 1  B: Self, 0
C: A->B, 2  C: None, inf

A: Self, 0  A: B->A, 1
B: A->B, 1  B: Self, 0
C: A->B, 2  C: None, inf

A: Self, 0  A: B->A, 1
B: A->B, 1  B: Self, 0
C: A->B, 2  C: None, inf

A: Self, 0  A: B->A, 1
B: A->B, 1  B: Self, 0
C: A->B, 2  C: None, inf


t=9: B<->C fails

t=10: B receives the following advertisement from A:
[(A,0)]

t=15: A receives the following advertisement from B:
[(B,0),(C,inf)]
Split Horizon

Don’t send advertisements about a route to the node providing the route

<table>
<thead>
<tr>
<th></th>
<th>A: Self, 0</th>
<th>B: A-&gt;B, 1</th>
<th>C: A-&gt;B, 2</th>
<th>C: None, inf</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B-&gt;A, 1</td>
<td>Self, 0</td>
<td>B: Self, 0</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td>B: Self, 0</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td>C: None, inf</td>
<td></td>
</tr>
</tbody>
</table>

\[t=9: \text{B} \leftrightarrow \text{C} \text{ fails}\]

\[t=10: \text{B receives the following advertisement from A: } [(A, 0)] \]

\[t=15: \text{A receives the following advertisement from B: } [(B, 0), (C, \text{inf})] \]
Split Horizon

Don’t send advertisements about a route to the node providing the route

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>A: Self, 0</td>
<td>A: B-&gt;A, 1</td>
<td>t=9: B&lt;&gt;C fails</td>
</tr>
<tr>
<td>B: A-&gt;B, 1</td>
<td>B: Self, 0</td>
<td>t=10: B receives the following advertisement from A: [(A,0)]</td>
</tr>
<tr>
<td>C: A-&gt;B, 2</td>
<td>C: None, inf</td>
<td>t=15: A receives the following advertisement from B: [(B,0),(C,inf)]</td>
</tr>
</tbody>
</table>

split horizon takes care of this particular case
Split-horizon

Don’t send advertisements about a route to the node providing the route

C: D->B, 2    C: A->B, 2    C: B->C, 1    C: Self, 0
Split-horizon

Don’t send advertisements about a route to the node providing the route

C: D->B, 2
C: A->B, 2
C: B->C, 1
B<->C fails
Split-horizon

Don’t send advertisements about a route to the node providing the route

C: D->B, 2
C: A->B, 2
C: None, inf

B<->C fails
Split-horizon

Don’t send advertisements about a route to the node providing the route

C: D->B, 2  C: A->B, 2  C: None, inf  B<->C fails
Split-horizon

Don’t send advertisements about a route to the node providing the route

C: D→B, 2  C: A→B, 2  C: None, inf

B→C fails
B’s advertisement to A gets lost (so A makes no changes)
Split-horizon

Don’t send advertisements about a route to the node providing the route

B<->C fails

B’s advertisement to A gets lost
(so A makes no changes)
Split-horizon

Don’t send advertisements about a route to the node providing the route

C: D->B, 2  C: A->B, 2  C: None, inf

C: None, inf  C: A->B, 2  C: None, inf

B<->C fails
B’s advertisement to A gets lost (so A makes no changes)
A advertises about C to D (not to B because of split horizon)
Split-horizon

Don’t send advertisements about a route to the node providing the route

C: D->B, 2   C: A->B, 2   C: None, inf
C: None, inf   C: A->B, 2   C: None, inf
C: D->A, 3   C: A->B, 2   C: None, inf

B<->C fails
B’s advertisement to A gets lost (so A makes no changes)
A advertises about C to D (not to B because of split horizon)
Split-horizon

Don’t send advertisements about a route to the node providing the route

C: D->B, 2
C: A->B, 2
C: None, inf

C: None, inf
C: A->B, 2
C: None, inf

C: D->A, 3
C: A->B, 2
C: None, inf

B<->C fails
B’s advertisement to A gets lost (so A makes no changes)
A advertises about C to D (not to B because of split horizon)
D advertises about C to B
Split-horizon

Don’t send advertisements about a route to the node providing the route

B<->C fails

B’s advertisement to A gets lost (so A makes no changes)

A advertises about C to D (not to B because of split horizon)

D advertises about C to B
Split-horizon

Don’t send advertisements about a route to the node providing the route

<table>
<thead>
<tr>
<th>Node</th>
<th>Advertisement (Path to C)</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A-&gt;B, 2</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>None, inf</td>
<td>inf</td>
</tr>
<tr>
<td>C</td>
<td>D-&gt;B, 2</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>D-&gt;A, 3</td>
<td>3</td>
</tr>
</tbody>
</table>

B’s advertisement to A gets lost (so A makes no changes)

A advertises about C to D (not to B because of split horizon)

D advertises about C to B

B advertises about C to A
Split-horizon

Don’t send advertisements about a route to the node providing the route

C: D→B, 2  C: A→B, 2  C: None, inf

C: None, inf  C: A→B, 2  C: None, inf

C: D→A, 3  C: A→B, 2  C: None, inf

C: D→A, 3  C: A→B, 2  C: B→D, 4

C: D→A, 3  C: A→B, 5  C: B→D, 4

B<->C fails

B’s advertisement to A gets lost (so A makes no changes)

A advertises about C to D (not to B because of split horizon)

D advertises about C to B

B advertises about C to A
Split-horizon

Don’t send advertisements about a route to the node providing the route

C: D->B, 2  C: A->B, 2  C: None, inf
C: None, inf  C: A->B, 2  C: None, inf
C: D->A, 3  C: A->B, 2  C: None, inf
C: D->A, 3  C: A->B, 2  C: B->D, 4
C: D->A, 3  C: A->B, 5  C: B->D, 4

B<->C fails
B’s advertisement to A gets lost (so A makes no changes)
A advertises about C to D (not to B because of split horizon)
D advertises about C to B
B advertises about C to A

continues until all costs to C are INFINITY
**problem:** neither distance-vector nor link-state routing will scale to the size of the Internet
• **Link-state routing** works by disseminating full topology information to all nodes. It’s quite robust to failures, but the **overhead** of flooding limits its scale.

• **Distance-vector routing** works by disseminating information about the cost of the actual routes. It has less overhead, but is not as robust to failures; the way in which it handles **failures** limits its scale.

• Neither of these protocols is appropriate for routing across the entire Internet. Link-state routing works well for MIT-sized networks, but we still need a means to route outside of MIT.