Shortest Path Routing

Routing Basics

Goal: find the best route between two nodes in the network

min # routers, fastest route, least expensive

① determine objective function

② assign cost to each link — hop count, delay, $

③ Find the shortest path between two nodes

Path length/cost = sum of link costs

Shortest Path = Path with smallest cost (shortest length)

Routing Basics

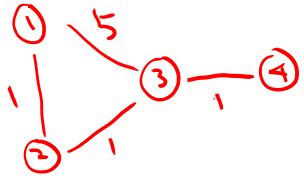
Routing scheme:

① Routing Protocol
- to distribute routing related info between nodes
- what ? How often?
- How ?

② Routing Algorithm
- how to use routing related info
- to compute shortest path
- and to update routing table

1. Distance Vector Routing

- Each node maintains
  1. A list of next hops to each destination node along the shortest path
     - routing table — H_{ij}
  2. A list of shortest distances to each destination node
     - distance vector — D_{ij}
  3. A list of link costs to each neighbor node — C_{ij}
     - ∞ link cost to each non-neighbor node



At node 1:

dest j	H_{1j}	D_{1j}	C_{1j}
1	1	0	0
2	2	1	1
3	2	2	5
4	2	3	∞

routing table; distance vector; known at the beginning

Distance Vector Routing

- Only distance vectors are exchanged between neighbor nodes

Routing Protocol

what to distribute? → global info

how ? → local

- Each node will update its lists upon receiving new distance vectors from neighbor nodes

Routing Algorithm

Bellman-Ford Algorithm

Bellman-Ford Algorithm

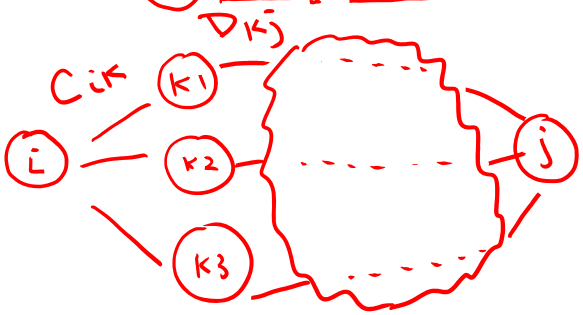
- Initialization Step

H, D, C

- Send Step
  - Each node sends its distance vector to its immediate neighbors across the local links

Bellman-Ford Algorithm

- Looping
  - If node i receives a distance vector from neighbor k or sees a link cost change to neighbor k, it re-calculates the shortest path to each destination j:
    - $D_{ij} = \min_k \{ C_{ik} + D_{kj} \}$
    - $H_{ij} = \arg\min_k \{ C_{ik} + D_{kj} \}$

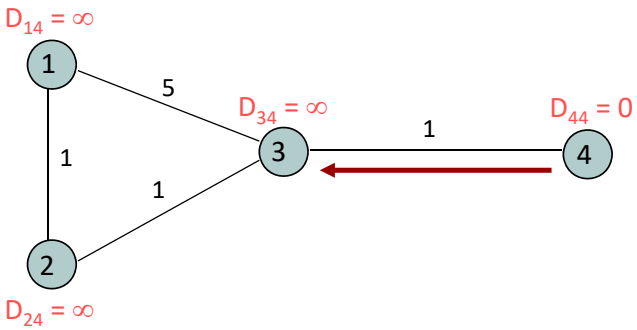


KEY STEP

Bellman-Ford Algorithm

- Looping (continued)
  - If a new D_{ij} or H_{ij} is found, go to Send Step (this is called triggered update)
  - Otherwise, periodic broadcast

Example: To Destination Node 4



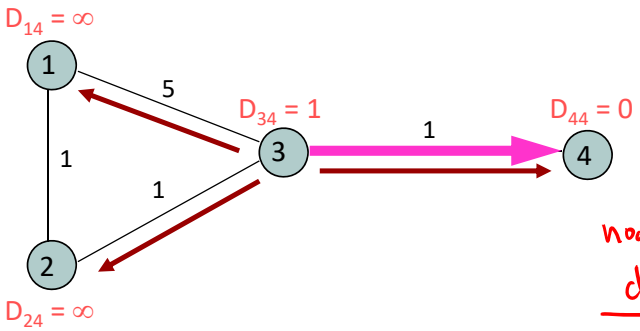
node 4's DV

dest j	D_{4j}
1	
2	
3	
4	0

	Iteration	Node 1	Node 2	Node 3
Dest 4	Initial	(-, ∞)	(-, ∞)	(-, ∞)

← (H.D)

Example: To Destination Node 4

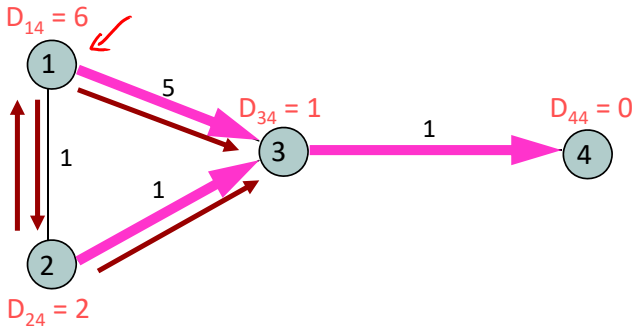


node 3's 4 DV:

dest j	D_{3j}
1	
2	
3	
4	1

	Iteration	Node 1	Node 2	Node 3
Dest 4	Initial	(-, ∞)	(-, ∞)	(-, ∞)
	1	(-, ∞)	(-, ∞)	(4, 1)

Example: To Destination Node 4



	Iteration	Node 1	Node 2	Node 3
Dest 4	Initial	(-, ∞)	(-, ∞)	(-, ∞)
	1	(-, ∞)	(-, ∞)	(4, 1)
	2	(3, 6)	(3, 2)	(4, 1)

node 1 receives:

DV from 2:

dest	D_{2j}
1	
2	
3	
4	2

DV from 3:

dest	D_{3j}
1	
2	
3	
4	1

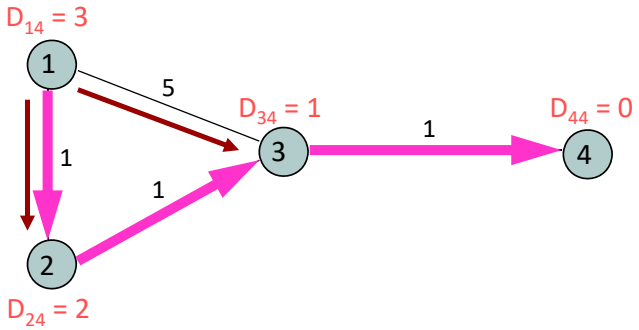
node 1 knows:

$\begin{cases} C_{12} = 1 \\ C_{13} = 5 \end{cases}$

$\begin{cases} C_{12} + D_{24} = 1 + 2 = 3 \\ C_{13} + D_{34} = 5 + 1 = 6 \end{cases}$

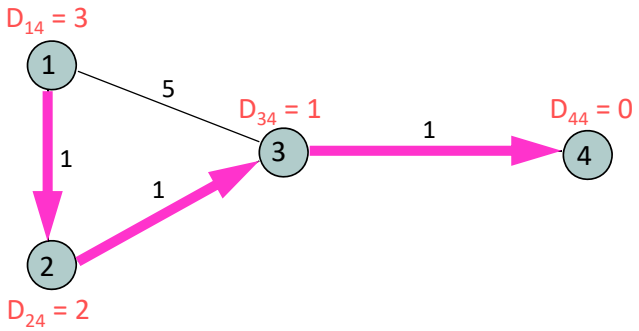
$\Rightarrow \begin{cases} D_{14} = \min_{k\{2,3\}} (C_{1k} + D_{k4}) = 3 \text{ when } k = 2 \\ \text{next-hop}_{14} = 2 \end{cases}$

Example: To Destination Node 4



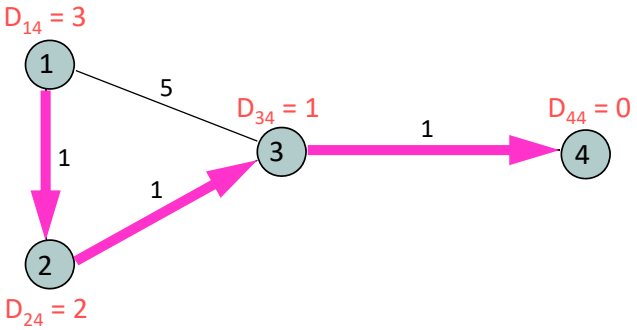
	Iteration	Node 1	Node 2	Node 3
Dest 4	**Initial**	**(-, ∞)**	**(-, ∞)**	**(-, ∞)**
	1	**(-, ∞)**	**(-, ∞)**	**(4, 1)**
	2	**(3, 6)**	**(3, 2)**	**(4, 1)**
	3	**(2, 3)**	**(3, 2)**	**(4, 1)**

Example: To Destination Node 4



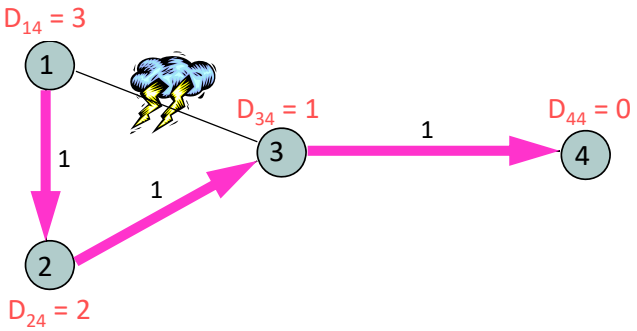
	Iteration	Node 1	Node 2	Node 3
Dest 4	**Initial**	**(-, ∞)**	**(-, ∞)**	**(-, ∞)**
	1	**(-, ∞)**	**(-, ∞)**	**(4, 1)**
	2	**(3, 6)**	**(3, 2)**	**(4, 1)**
	3	**(2, 3)**	**(3, 2)**	**(4, 1)**
	4	**(2, 3)**	**(3, 2)**	**(4, 1)**

Example: After Protocol Converges, Information at Node 1



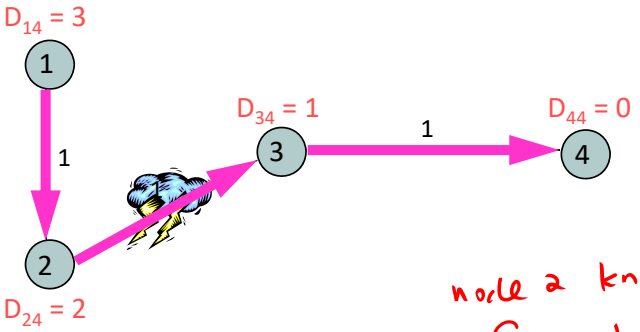
Dest J	H_{1j}	D_{1j}	C_{1j}
1	1	0	0
2	2	1	1
3	2	2	5
4	2	3	∞

What if link between 1 and 3 breaks?



Dest 4	Iteration	Node 1	Node 2	Node 3
	Initial	(2, 3)	(3, 2)	(4, 1)
	After break	(2, 3)	(3, 2)	(4, 1)

Problem Scenario: What if link between 2 and 3 breaks?



At node 2:

DV from 1:

dest	D_{1j}
1	
2	
3	
4	3

DV from 3:

dest	D_{3j}
1	
2	
3	
4	1

node 2 knows:

$\begin{cases} C_{21} = 1 \\ C_{23} = 1 \rightarrow \infty \end{cases}$

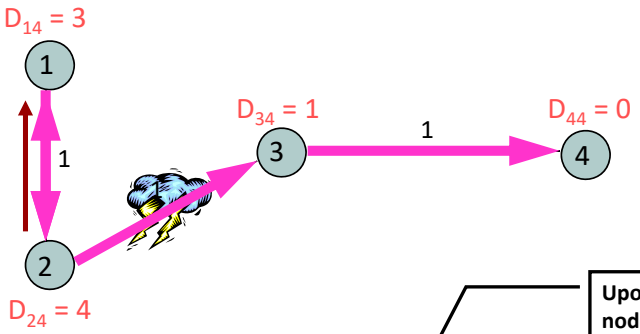
$\begin{cases} C_{21} + D_{14} = 1 + 3 = 4 \\ C_{23} + D_{34} = \infty \end{cases}$

$\Rightarrow D_{24} = \min_{k \in \{1,3\}} (C_{2k} + D_{k4})$

$= 4$ when $k = 1$

$H_{1,4} = 1$

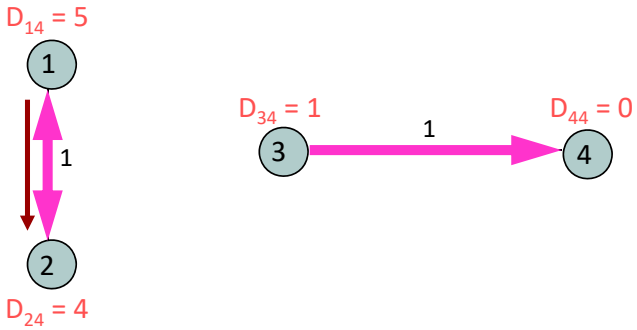
Problem Scenario: What if link between 2 and 3 breaks?



	Iteration	Node 1	Node 2	Node 3
Dest 4	**Initial**	**(2, 3)**	**(3, 2)**	**(4, 1)**
	After break	**(2, 3)**	**(1, 4)**	**(4, 1)**

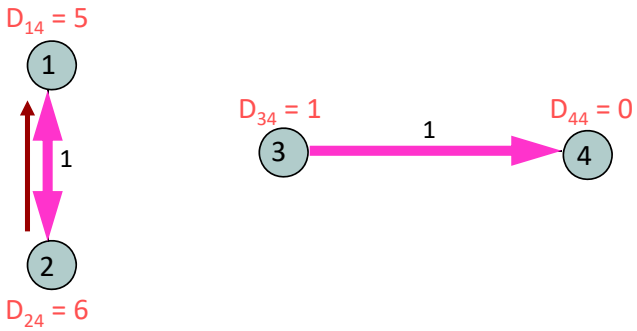
Upon link cost change seen by node 2, it recalculates the shortest path to dest 4, which is through node 1 with distance 4!

Problem Scenario: What if link between 2 and 3 breaks?



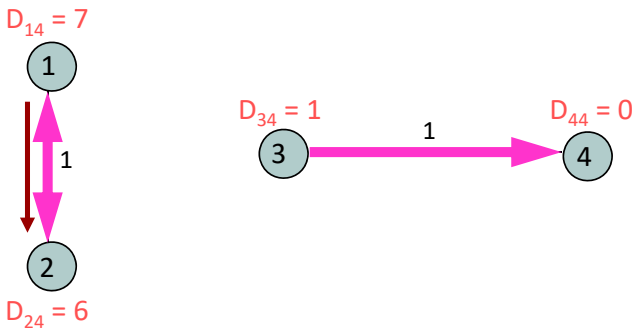
	Iteration	Node 1	Node 2	Node 3
Dest 4	**Initial**	**(2, 3)**	**(3, 2)**	**(4, 1)**
	After break	**(2, 3)**	**(1, 4)**	**(4, 1)**
	1	**(2, 5)**	**(1, 4)**	**(4, 1)**

Problem Scenario: What if link between 2 and 3 breaks?



	Iteration	Node 1	Node 2	Node 3
Dest 4	**Initial**	**(2, 3)**	**(3, 2)**	**(4, 1)**
	After break	**(2, 3)**	**(1, 4)**	**(4, 1)**
	1	**(2, 5)**	**(1, 4)**	**(4, 1)**
	2	**(2, 5)**	**(1, 6)**	**(4, 1)**

Problem Scenario: What if link between 2 and 3 breaks?



	Iteration	Node 1	Node 2	Node 3
Dest 4	Initial	(2, 3)	(3, 2)	(4, 1)
	After break	(2, 3)	(1, 4)	(4, 1)
	1	(2, 5)	(1, 4)	(4, 1)
	2	(2, 5)	(1, 6)	(4, 1)
	3	(2, 7)	(1, 6)	(4, 1)

Problem: Routing Loop → Counting to Infinity!

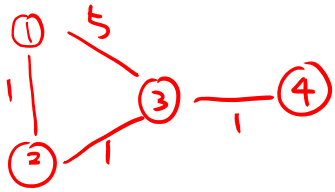
- Causes of Problem
  - Router does not know whether it is in its neighbor's path to a destination
  - Inconsistent routing tables
  - Updates do not reflect reality

Problem: Routing Loop → Counting to Infinity!

- Use heuristics to alleviate the problem
  - Split Horizon (SH)
    - For node W, its neighbor X, and destination Y, if $H_{WY} = X$, then exclude D_{WY} from node W's DV report to neighbor X

W's DV report to X:

destj	Dwj
⋮	value



node 1's DV:

dest	Dij	Hij
1	0	1
2	1	2 ✓
3	2	2 ✓
4	3	2 ✓

local

node 1's DV report to node 2:

dest	Dij
1	0

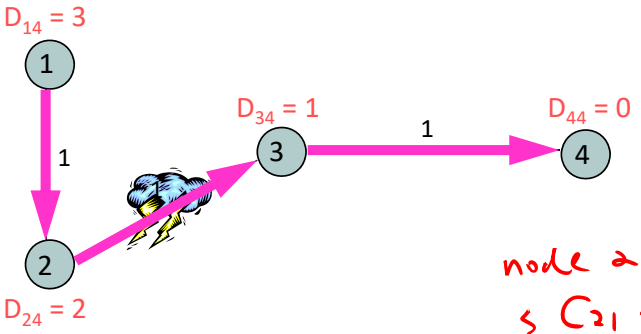
Problem: Routing Loop → Counting to Infinity!

- Use heuristics to alleviate the problem
  - Split Horizon with Poisoned Reverse (SHPR)
    - For node W, its neighbor X, and destination Y, if $H_{WY} = X$, then set $D_{WY} = \infty$ in node W's DV report to neighbor X
    - This breaks erroneous direct loops immediately

node 1's DV report to 2:

dest	Dij
1	0
2	∞
3	∞
4	∞

Example: Problem Solved with SHPR



At node 2:

DV report from 1

dest	D_{1j}
1	
2	
3	
4	~~∞~~

DV report from 3

dest	D_{3j}
1	
2	
3	
4	1

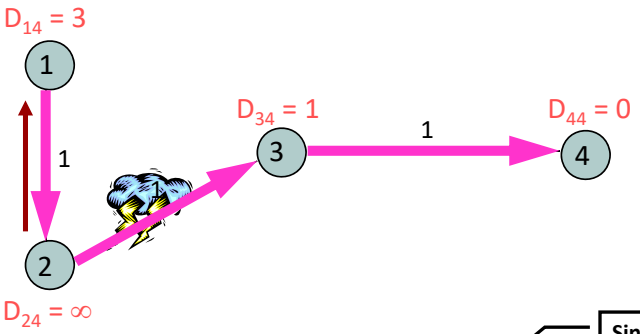
node 2 knows:

$\{ C_{21} = 1$
$C_{23} = 1 \to \infty \}$

$\{ C_{21} + D_{14} = \infty$
$C_{23} + D_{34} = \infty \}$

$\Rightarrow \{ D_{24} = \infty$
$H_{24} = N/A \}$

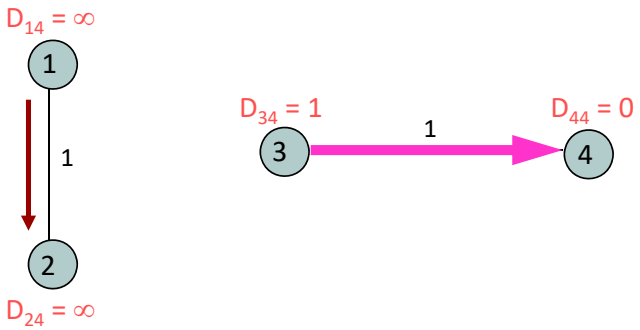
Example: Problem Solved with SHPR



	Iteration	Node 1	Node 2	Node 3
Dest 4	Initial	(2, 3)	(3, 2)	(4, 1)
	After break	(2, 3)	(-, ∞)	(4, 1)

Since node 1 advertised to node 2 that its minimum cost to dest 4 is ∞, node 2 finds no route to dest 4.

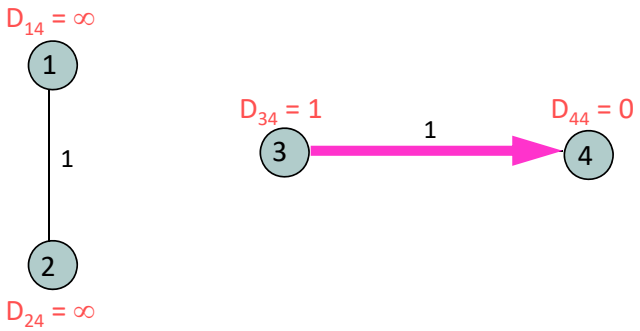
Example: Problem Solved with SHPR



	Iteration	Node 1	Node 2	Node 3
Dest 4	Initial	(2, 3)	(3, 2)	(4, 1)
	After break	(2, 3)	(-, ∞)	(4, 1)
	1	(-, ∞)	(-, ∞)	(4, 1)

Node 1 also finds no route to dest 4.

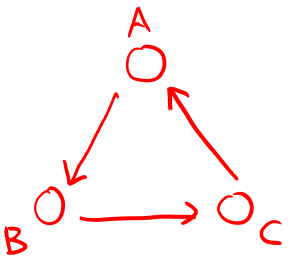
Example: Problem Solved with SHPR



	Iteration	Node 1	Node 2	Node 3
Dest 4	Initial	(2, 3)	(3, 2)	(4, 1)
	After break	(2, 3)	(-, ∞)	(4, 1)
	1	(-, ∞)	(-, ∞)	(4, 1)
	2	(-, ∞)	(-, ∞)	(4, 1)

SHPR is NOT a Loop-Free Solution!

- SHPR eliminates the routing loops that only involve 2 nodes
- SHPR does not eliminate the routing loops that involve >2 nodes



SHPR is NOT a Loop-Free Solution!

- Example Loop-free Scheme: Path Vector Routing
  - Each node sends to its neighbors the entire path information to every destination
  - Each node uses a neighbor's information for a certain destination only if itself is not on this neighbor's path to the destination
  - Each node prepends itself to paths before further propagation
  - Example:
    - Node 1's path vector

Dest	1	2	3	4
Distance	0	1	2	3
Path	<1>	<1, 2>	<1, 2, 3>	<1, 2, 3, 4>