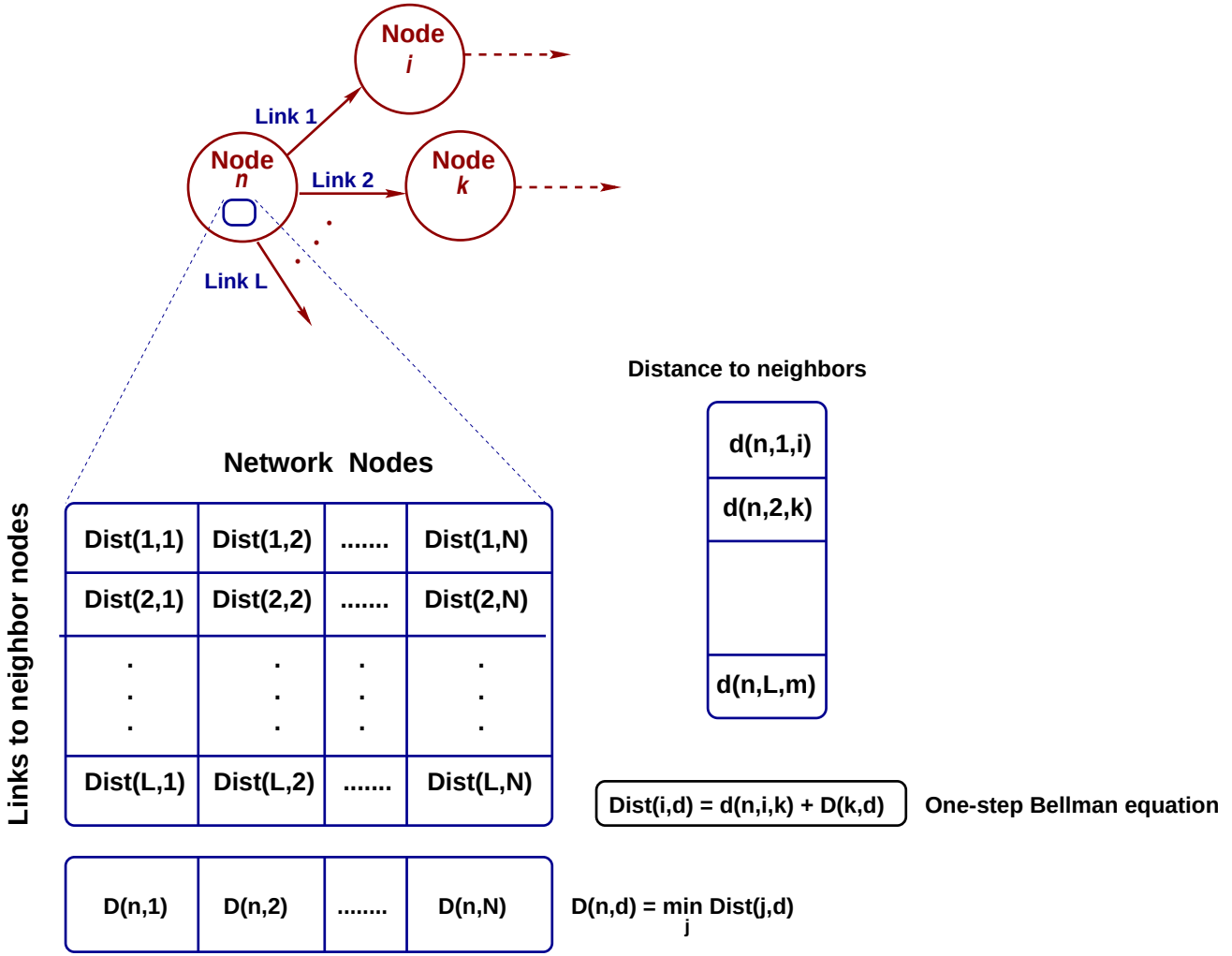
Shortest-path: Distance-vector algorithms

- Each node n maintains a vector $Dist(i,d)$ of *Distance* estimates for each destination d in the network and for each one of its outcoming links i.

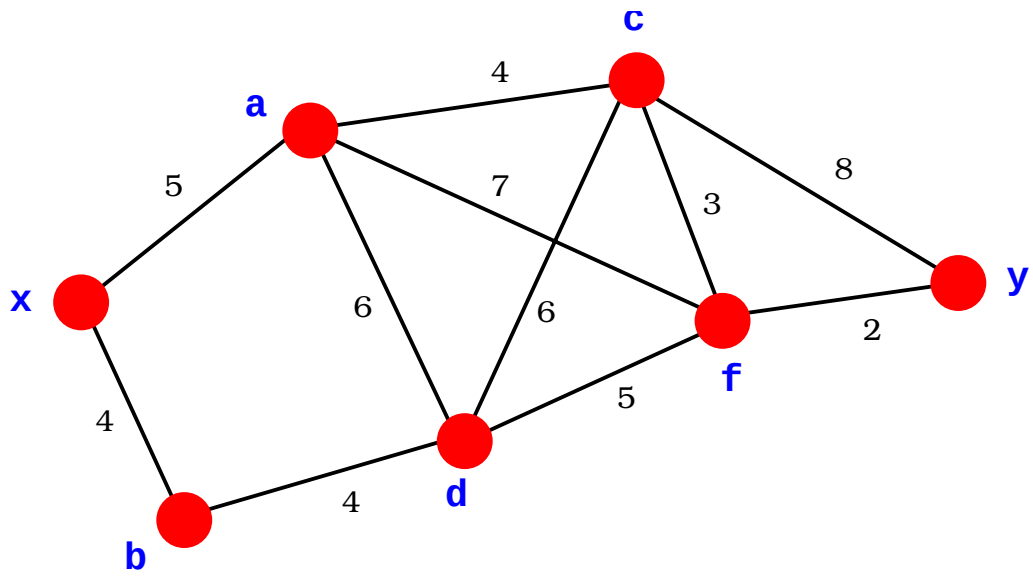


Shortest-path: Distance-vector algorithms

- Optimal way for distance computation in the stationary case: *Distributed (asynchronous) dynamic-programming* (Bellman-Ford, 1958)
- Each node is a *state* of the DP algorithm, its *value* is the *distance* to each possible destination
- The vector $D(n, d)$ of the best Distance estimates are periodically sent to each neighbor node and here used, in turn, to update their Distance estimates
- Used on the Internet (RIP)
- How do we set the frequency?
- Couuting to infinity problem

Shortest-path: Distance-vector algorithms

- At each node x define the *distance* to reach any other node y:
  $d_x(y)$ = cost of least-cost path from x to y
- Update distances based on neighbors (*one-step Bellman eq.*):
  $$d_x(y) = \min\{c(x,v) + d_v(y)\}, \quad \forall v \in \mathcal{N}(x)$$



$$d_x(y) = \min\{c(x,a) + d_a(y),\ c(x,b) + d_b(y)\}$$

Distance-vector algorithms

- $c(x, v)$ = *cost for direct link from x to v*

  - Node x maintains costs-to-go to the neighbors: $c(x, v), \forall v \in \mathcal{N}(x)$

- $D_x(y)$ = *estimate of least cost from x to y*

  - Node x maintains a distance vector $D_x = [D_x(y),\ \forall y \in N]$

- Node x maintains the distance vectors of all its neighbors

  - Node x maintains $D_v = [D_v(y),\ \forall v \in \mathcal{N}(x)\ \wedge \forall y \in N]$

- Each node v periodically sends D_v to its neighbors in $\mathcal{N}(v)$

  - Each neighbor x updates its own distance vectors:

$$D_x(y) \leftarrow \min_v \{c(x, v) + D_v(y)\}, \quad \forall y \in N$$

- Over time, the distance vector D_x converges to the least-cost

Distance-vector at work

Optimum 1-hop paths

Table for A		
Dst	Cst	Hop
A	0	A
B	4	B
C	∞	–
D	∞	–
E	2	E
F	6	F

Table for B		
Dst	Cst	Hop
A	4	A
B	0	B
C	∞	–
D	3	D
E	∞	–
F	1	F

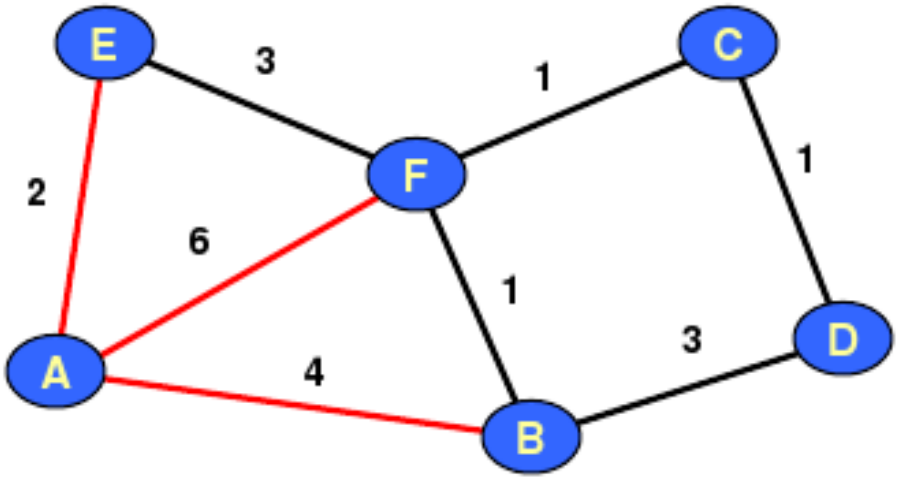


Table for C		
Dst	Cst	Hop
A	∞	–
B	∞	–
C	0	C
D	1	D
E	∞	–
F	1	F

Table for D		
Dst	Cst	Hop
A	∞	–
B	3	B
C	1	C
D	0	D
E	∞	–
F	∞	–

Table for E		
Dst	Cst	Hop
A	2	A
B	∞	–
C	∞	–
D	∞	–
E	0	E
F	3	F

Table for F		
Dst	Cst	Hop
A	6	A
B	1	B
C	1	C
D	∞	–
E	3	E
F	0	F

Distance-vector at work

Optimum 2-hop paths

Table for A			Table for B		
Dst	Cst	Hop	Dst	Cst	Hop
A	0	A	A	4	A
B	4	B	B	0	B
C	7	F	C	2	F
D	7	B	D	3	D
E	2	E	E	4	F
F	5	E	F	1	F

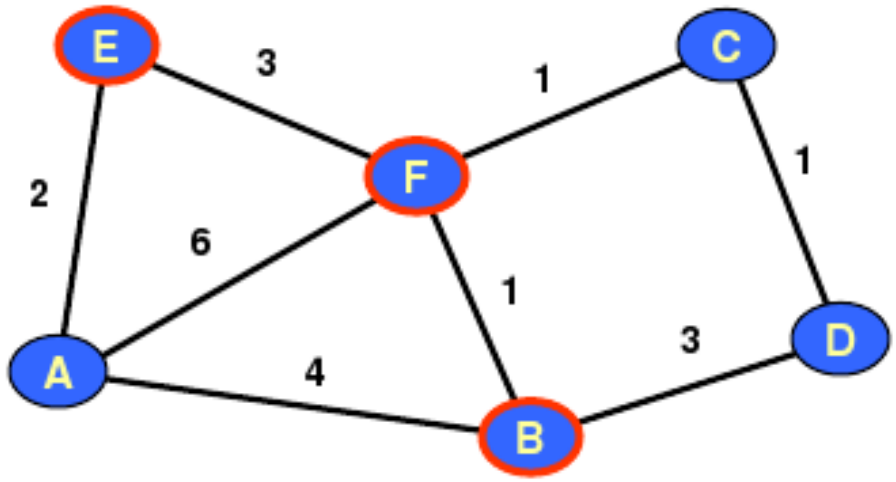


Table for C			Table for D			Table for E			Table for F		
Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop
A	7	F	A	7	B	A	2	A	A	5	B
B	2	F	B	3	B	B	4	F	B	1	B
C	0	C	C	1	C	C	4	F	C	1	C
D	1	D	D	0	D	D	∞	–	D	2	C
E	4	F	E	∞	–	E	0	E	E	3	E
F	1	F	F	2	C	F	3	F	F	0	F

Distance-vector at work

Optimum 3-hop paths

Table for A		
Dst	Cst	Hop
A	0	A
B	4	B
C	6	E
D	7	B
E	2	E
F	5	E

Table for B		
Dst	Cst	Hop
A	4	A
B	0	B
C	2	F
D	3	D
E	4	F
F	1	F

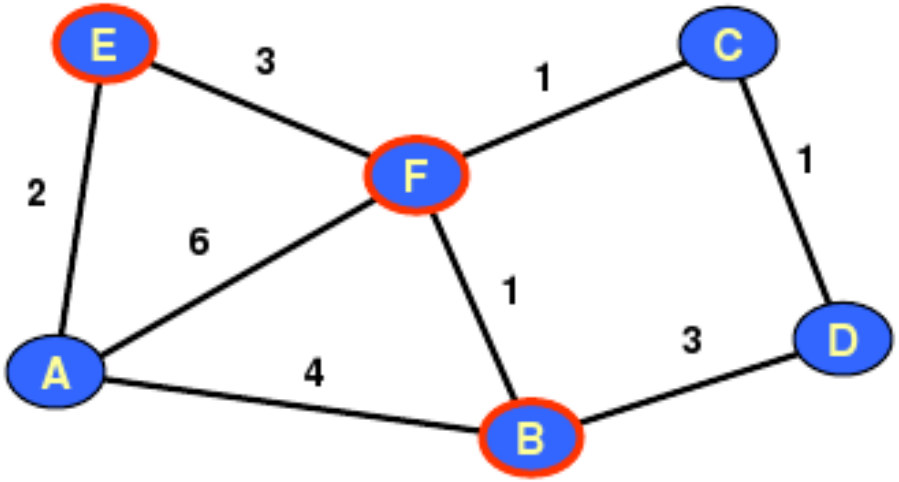


Table for C		
Dst	Cst	Hop
A	6	F
B	2	F
C	0	C
D	1	D
E	4	F
F	1	F

Table for D		
Dst	Cst	Hop
A	7	B
B	3	B
C	1	C
D	0	D
E	5	C
F	2	C

Table for E		
Dst	Cst	Hop
A	2	A
B	4	F
C	4	F
D	5	F
E	0	E
F	3	F

Table for F		
Dst	Cst	Hop
A	5	B
B	1	B
C	1	C
D	2	C
E	3	E
F	0	F

Distance-vector: node behavior

1. Monitor changes in local link costs
2. Wait for distance vector update messages from neighbors
3. Recompute distance estimates and notify the neighbors if any change has occured, or. . .
4. . . . Periodically send distance vectors (e.g., 30 seconds)