Internet Protocol (IP)

2 basic functions

- Assigning unique addresses to each attachment point (router/host interface)
- Best effort packet delivery
  - Fragmentation and re-assembly when needed

How does a router figure out where to forward each packet?

- Based on the information provided by network routing protocols

Where we are in the book

4.5 Routing algorithms (this lecture)

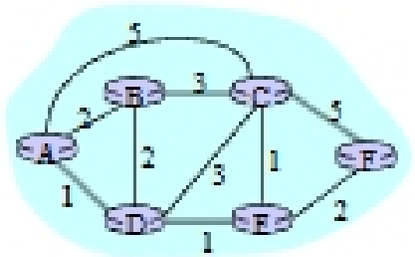
- Link state
- Distance Vector
- Hierarchical routing (FYI)

4.6 Routing in the Internet (next lecture)

- RIP: Routing Information Protocol
- OSPF: Open Shortest Path First Protocol
- BGP: Border Gateway Protocol

4.7 Broadcast and multicast routing (8th week)

Network Graph Abstraction



Graph: G = (N,E)

N = set of routers = { A, B, C, D, E, F }

E = set of links ={ (A,B), (A,C), (A,D), (B,C), (B,D), (C,D), (C,E), (C,F), (E,F) }

Cost of link (a, b) = c(a, b)
- e.g. c(A, B) = 2
- link cost could always be 1, or inversely related to bandwidth, or inversely related to congestion

Cost of path $(x_1, x_2, x_3, \ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

Routing algorithm: given a graph, find least-cost path from a given node to all the other nodes in the graph

Network Routing: algorithms vs. protocols

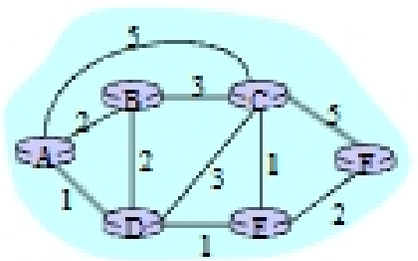
Route computation algorithms:

link-state (Dijkstra): each router

- has complete topology graph with all link costs
- computes shortest paths to all destinations

distance-vector (Bellman-Ford): a router

- knows its physically-connected neighbors and link costs to each
- computes its shortest paths based on the paths it learns from all neighbors



Routing protocols

- define the format of routing information exchanges
- define the computation upon receiving routing updates
- network topology changes over time → routing protocol must continuously update the routers with latest changes

Link-State algorithm: basic notations

- all nodes have identical info for network topology graph and link cost
- Each node computes least cost paths from itself to all other nodes
  - builds forwarding table for next hop
- iterative: after *k* iterations, a node know the best path to *k* destinations

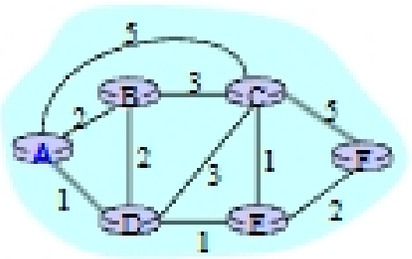
Notation:

- c(x,y): link cost from node x to y
  - Initializes to ∞ if (x, y) are not direct neighbors
- D(v): current value of cost of path from source to destination v
- p(v): predecessor node along path from source to v
- N': set of nodes whose least-cost paths are known

Link-State algorithm

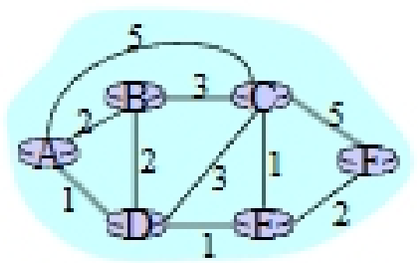
(consider the computation done at node A)

```
1  Initialization:
2  N' = {A}
3  for all nodes v
4    if v adjacent to A
5      then D(v) = c(A,v)
6      else D(v) = ∞
7
8  Loop
9   find w not in N' such that D(w) is minimum
10    add w to N'
11    update D(v) for all v adjacent to w and not in N':
12       D(v) = min( D(v), D(w) + c(w,v) ), p(v) = w
13  /* new cost to v is either the old cost, or the
14    shortest path cost to w plus the cost from w to v */
15 until all nodes in N'
```



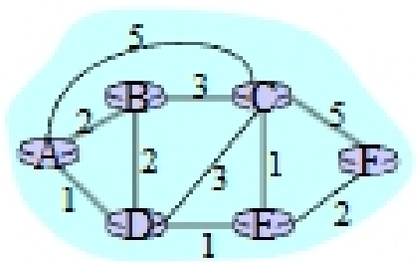
Link-State algorithm: example

Step	start N'	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
→ 0	A	2,A	5,A	1,A	∞	∞



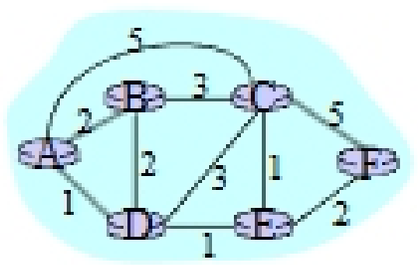
Link-State algorithm: example

Step	start N'	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
→ 1	AD	2,A	4,D		2,D	



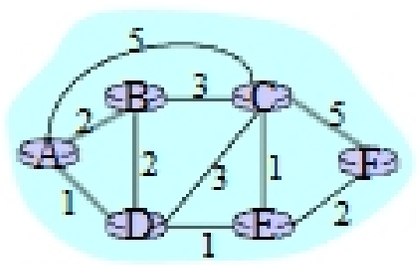
Link-State algorithm: example

Step	start N'	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD	2,A	4,D		2,D	
2	ADE		3,E			4,E



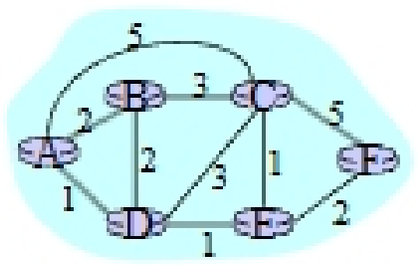
Link-State algorithm: example

Step	start N'	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD	2,A	4,D		2,D	
2	ADE		3,E			4,E
3	ADEB		3,E			



Link-State algorithm: example

Step	start N'	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD	2,A	4,D		2,D	
2	ADE		3,E			4,E
3	ADEB		3,E			
4	ADEBC					4,E



Link-State algorithm: example

Step	start N'	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD	2,A	4,D		2,D	
2	ADE		3,E			4,E
3	ADEB		3,E			
4	ADEBC					4,E
5	ADEBCF