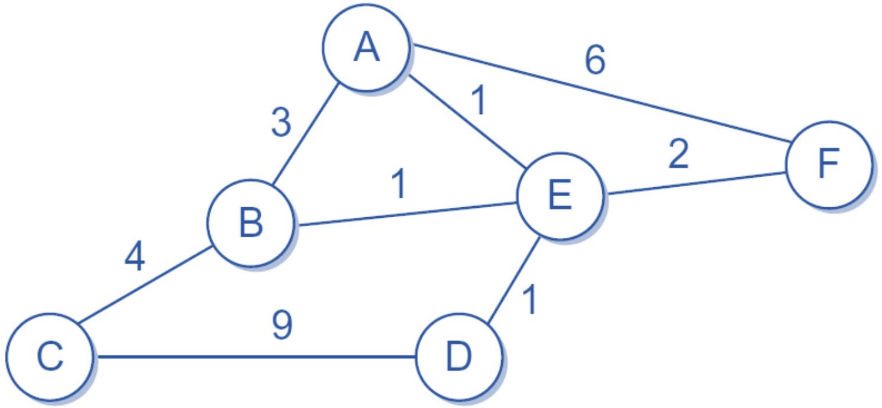
CS120: Computer Networks

Lecture 10. Routing 1

Zhice Yang

Network as a Graph

- The basic problem of routing is to find the lowest-cost path between any two nodes
  - Static approach has several shortcomings
    - Can not handle node or link failures
    - Can not handle addition of new nodes or links
    - Edge costs cannot change
  - Centralized solution does not scale
  - ➢ Distributed and dynamic protocol



Routing Protocols

- Routing Information Protocol (RIP)
  - Algorithm: Distance Vector
- Open Shortest Path First (OSPF)
  - Algorithm: Link State
- Border Gateway Protocol (BGP)

Intradomain Routing Protocol

Interdomain Routing Protocol

Distance Vector Algorithm

- Bellman-Ford equation

let

$d_x(y)$ =cost of lowest-cost path from x to y

then

$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

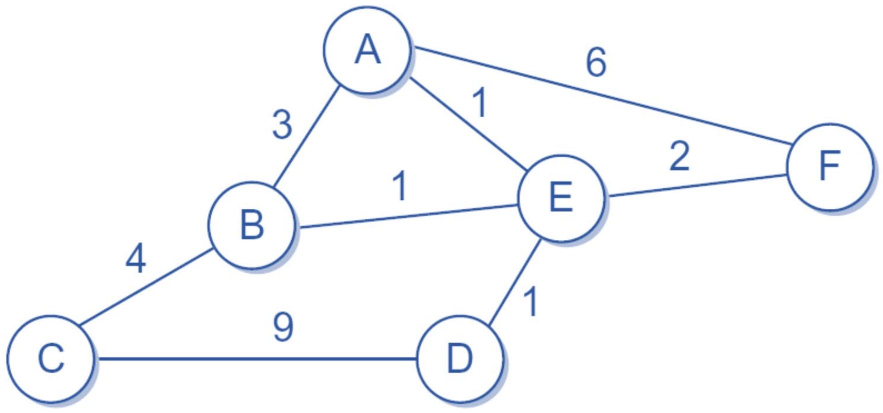
lowest-cost from neighbor v to destination y

cost to neighbor v

min taken over all neighbors v of x

Example

- $d_B(A) = 2$
- $d_D(A) = 2$
- $d_C(A) = \min(d_B(A) + 4, d_D(A) + 9) = 6$



Distance Vector Algorithm

- x maintains its distance vector estimate $\boldsymbol{D_x(y)} = \{D_x(y): y \in N\}$
- x knows:
  - cost to each neighbor v: $c(x, v)$
  - neighbors' distance vectors estimate: $\boldsymbol{D_v(y)} = \{D_v(y): y \in N\}$
- Algorithm idea:
  - From time-to-time, each node sends its own distance vector estimate to neighbors
  - When x receives new distance vector estimate from neighbor, it updates its own distance vector estimate using Bellman-Ford equation
  - Under minor, natural conditions, the estimate $D_x(y)$ will converge to the actual lowest cost $d_x(y)$

Distance Vector Algorithm

y	$D_A(y)$
A	0
B	inf
C	inf
D	inf
E	inf
F	inf
G	inf

y	$D_B(y)$
A	inf
B	0
C	inf
D	inf
E	inf
F	inf
G	inf

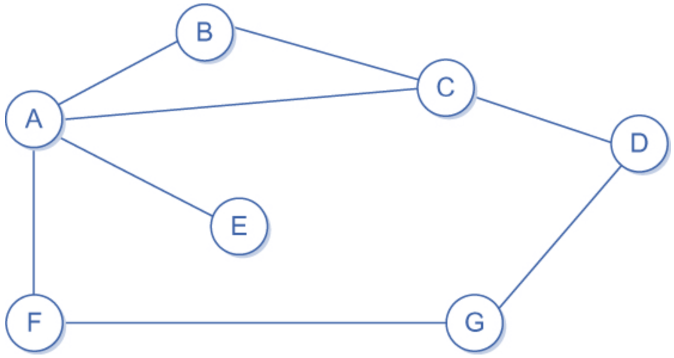
y	$D_C(y)$
A	inf
B	inf
C	0
D	inf
E	inf
F	inf
G	inf

y	$D_D(y)$
A	inf
B	inf
C	inf
D	0
E	inf
F	inf
G	inf

y	$D_E(y)$
A	inf
B	inf
C	inf
D	inf
E	0
F	inf
G	inf

y	$D_F(y)$
A	inf
B	inf
C	inf
D	inf
E	inf
F	0
G	inf

y	$D_G(y)$
A	inf
B	inf
C	inf
D	inf
E	inf
F	inf
G	0



Distance Vector Algorithm

y	$D_A(y)$
A	0
B	1
C	1
D	inf
E	1
F	1
G	inf

y	$D_B(y)$
A	1
B	0
C	1
D	inf
E	inf
F	inf
G	inf

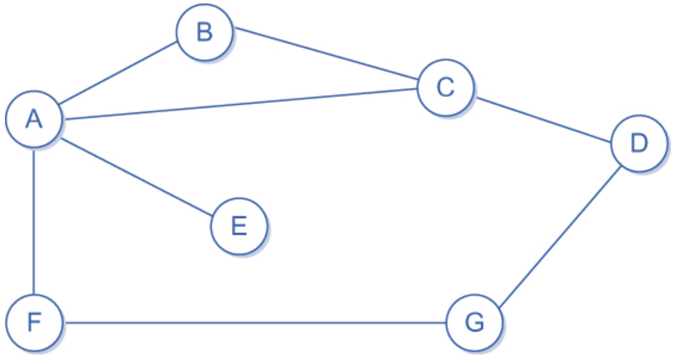
y	$D_C(y)$
A	1
B	1
C	0
D	1
E	inf
F	inf
G	inf

y	$D_D(y)$
A	inf
B	inf
C	1
D	0
E	inf
F	inf
G	1

y	$D_E(y)$
A	1
B	inf
C	inf
D	inf
E	0
F	inf
G	inf

y	$D_F(y)$
A	1
B	inf
C	inf
D	inf
E	inf
F	0
G	1

y	$D_G(y)$
A	inf
B	inf
C	inf
D	1
E	inf
F	1
G	0



Distance Vector Algorithm

- Every T seconds each router sends its table to its neighbor
- Each router then updates its table based on the new information

Distance Vector Algorithm

y	$D_A(y)$
A	0
B	1
C	1
D	inf
E	1
F	1
G	inf

y	$D_B(y)$
A	1
B	0
C	1
D	inf
E	inf
F	inf
G	inf

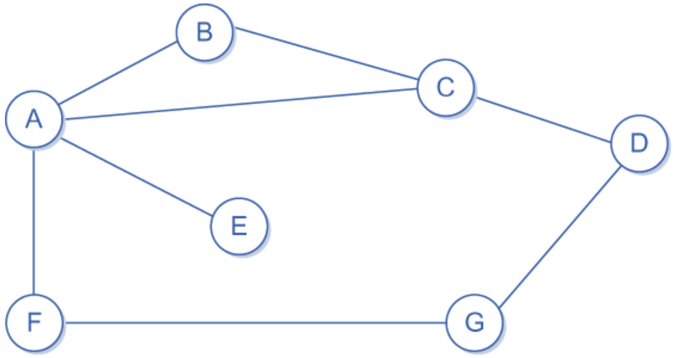
y	$D_C(y)$
A	1
B	1
C	0
D	1
E	inf
F	inf
G	inf

y	$D_D(y)$
A	inf
B	inf
C	1
D	0
E	inf
F	inf
G	1

y	$D_E(y)$
A	1
B	inf
C	inf
D	inf
E	0
F	inf
G	inf

y	$D_F(y)$
A	1
B	inf
C	inf
D	inf
E	inf
F	0
G	1

y	$D_G(y)$
A	inf
B	inf
C	inf
D	1
E	inf
F	1
G	0



Distance Vector Algorithm

y	$D_A(y)$
A	0
B	1
C	1
D	inf
E	1
F	1
G	inf

y	$D_B(y)$
A	1
B	0
C	1
D	inf
E	inf
F	inf
G	inf

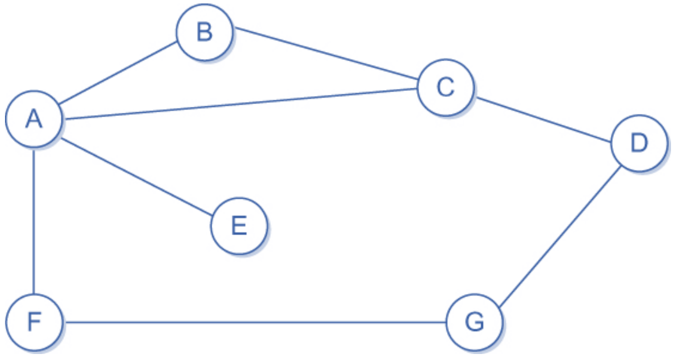
y	$D_C(y)$
A	1
B	1
C	0
D	1
E	inf
F	inf
G	inf

y	$D_D(y)$
A	inf
B	inf
C	1
D	0
E	inf
F	inf
G	1

y	$D_E(y)$
A	1
B	inf
C	inf
D	inf
E	0
F	inf
G	inf

y	$D_F(y)$
A	1
B	inf
C	inf
D	inf
E	inf
F	0
G	1

y	$D_G(y)$
A	inf
B	inf
C	inf
D	1
E	inf
F	1
G	0



Distance Vector Algorithm

y	$D_A(y)$
A	0
B	1
C	1
D	2
E	1
F	1
G	inf

y	$D_B(y)$
A	1
B	0
C	1
D	inf
E	inf
F	inf
G	inf

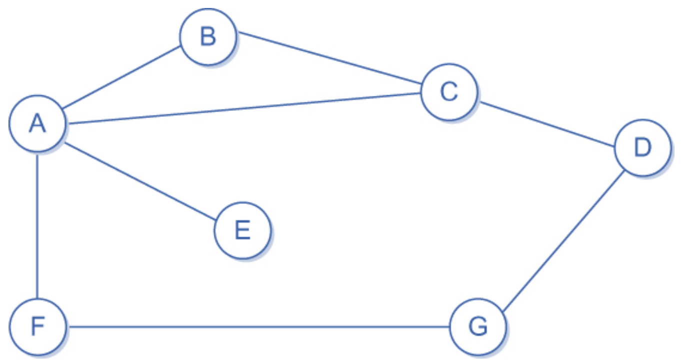
y	$D_C(y)$
A	1
B	1
C	0
D	1
E	inf
F	inf
G	inf

y	$D_D(y)$
A	inf
B	inf
C	1
D	0
E	inf
F	inf
G	1

y	$D_E(y)$
A	1
B	inf
C	inf
D	inf
E	0
F	inf
G	inf

y	$D_F(y)$
A	1
B	inf
C	inf
D	inf
E	inf
F	0
G	1

y	$D_G(y)$
A	inf
B	inf
C	inf
D	1
E	inf
F	1
G	0



Distance Vector Algorithm

y	$D_A(y)$
A	0
B	1
C	1
D	2
E	1
F	1
G	inf

y	$D_B(y)$
A	1
B	0
C	1
D	inf
E	inf
F	inf
G	inf

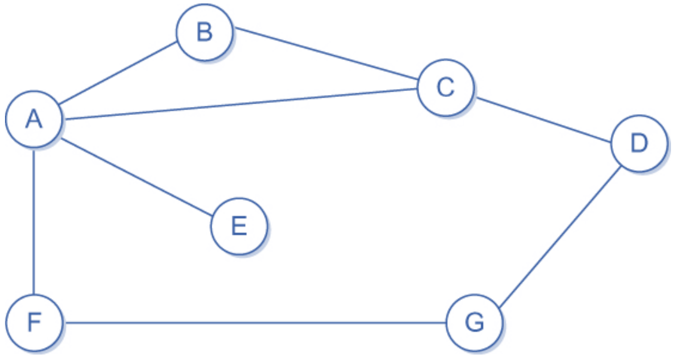
y	$D_C(y)$
A	1
B	1
C	0
D	1
E	inf
F	inf
G	inf

y	$D_D(y)$
A	inf
B	inf
C	1
D	0
E	inf
F	inf
G	1

y	$D_E(y)$
A	1
B	inf
C	inf
D	inf
E	0
F	inf
G	inf

y	$D_F(y)$
A	1
B	inf
C	inf
D	inf
E	inf
F	0
G	1

y	$D_G(y)$
A	inf
B	inf
C	inf
D	1
E	inf
F	1
G	0



Distance Vector Algorithm

y	$D_A(y)$
A	0
B	1
C	1
D	2
E	1
F	1
G	2

y	$D_B(y)$
A	1
B	0
C	1
D	inf
E	inf
F	inf
G	inf

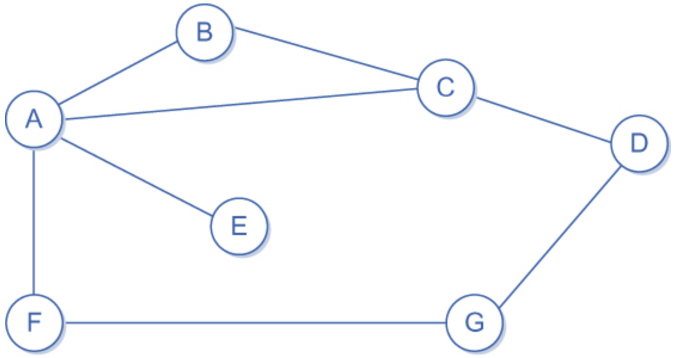
y	$D_C(y)$
A	1
B	1
C	0
D	1
E	inf
F	inf
G	inf

y	$D_D(y)$
A	inf
B	inf
C	1
D	0
E	inf
F	inf
G	1

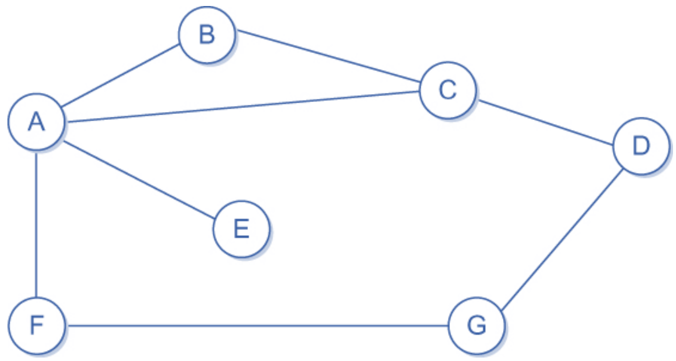
y	$D_E(y)$
A	1
B	inf
C	inf
D	inf
E	0
F	inf
G	inf

y	$D_F(y)$
A	1
B	inf
C	inf
D	inf
E	inf
F	0
G	1

y	$D_G(y)$
A	inf
B	inf
C	inf
D	1
E	inf
F	1
G	0



Distance Vector Algorithm



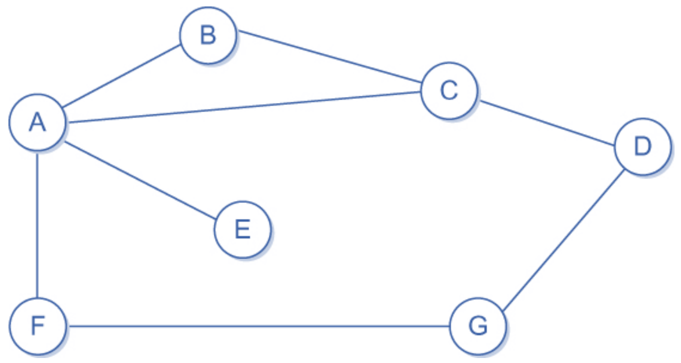
Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0

y	$D_A(y)$	via
A	0	A
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	2	F

Distance Vector Algorithm

- Good news travels fast

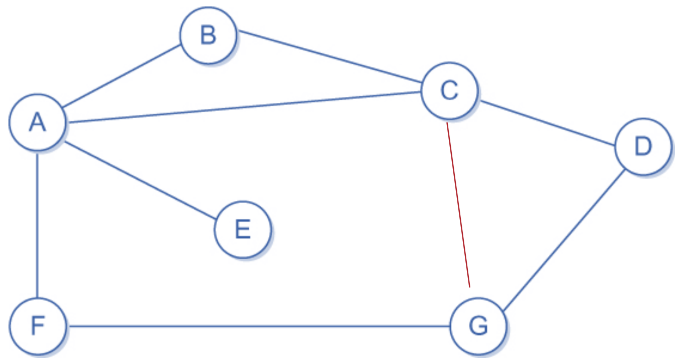
Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0



Distance Vector Algorithm

- Good news travels fast

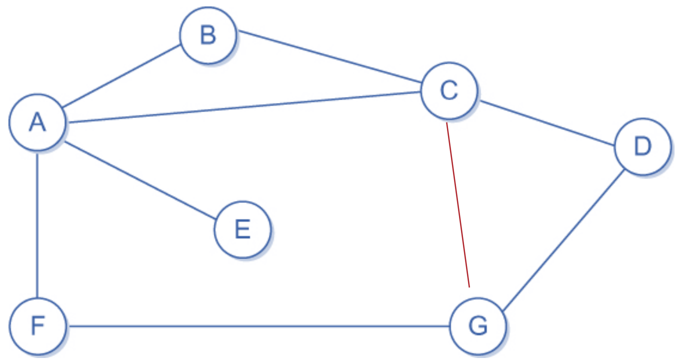
Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	$\boxed{1}$
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	$\boxed{1}$	1	3	1	0



Distance Vector Algorithm

- Good News Travels Fast

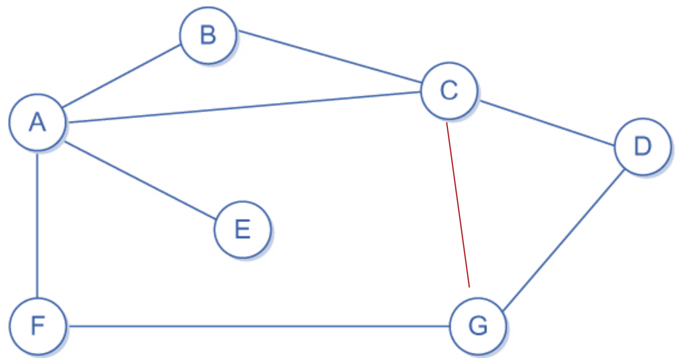
Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	2
C	1	1	0	1	2	2	1
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	2	1	1	3	1	0



Distance Vector Algorithm

- Good News Travels Fast

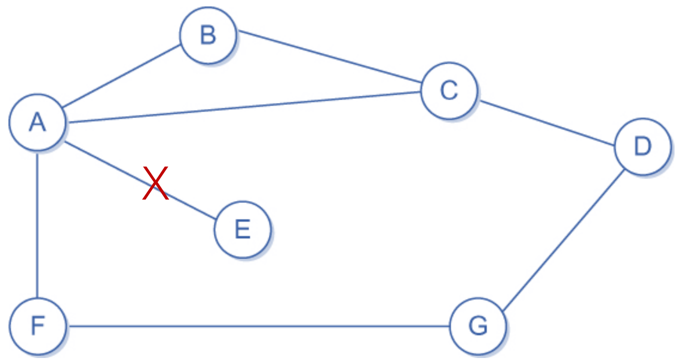
Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	$\boxed{2}$
C	1	1	0	1	2	2	$\boxed{1}$
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	$\boxed{2}$	$\boxed{1}$	1	3	1	0



Distance Vector Algorithm

- Bad News Travels Slow

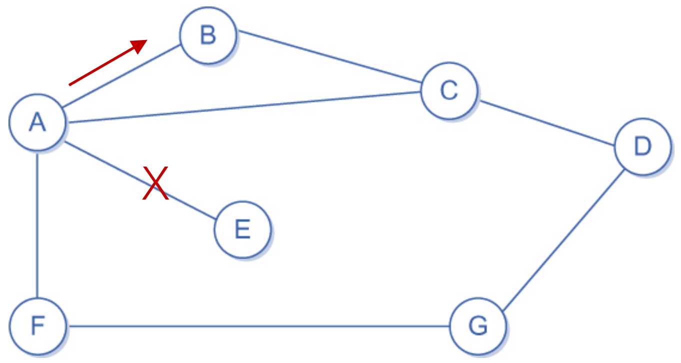
Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	inf	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0



Distance Vector Algorithm

- Bad News Travels Slow

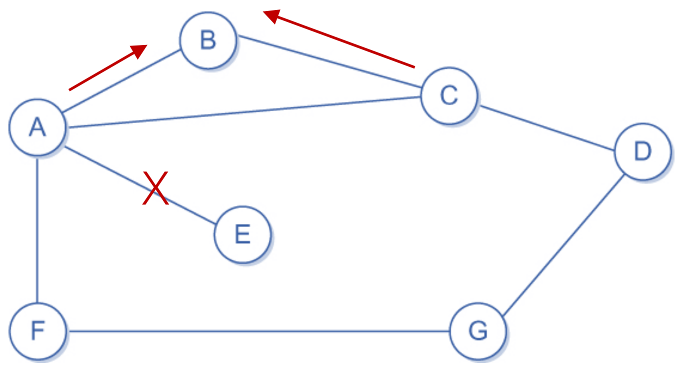
Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	inf	inf	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0



Distance Vector Algorithm

- Bad News Travels Slow

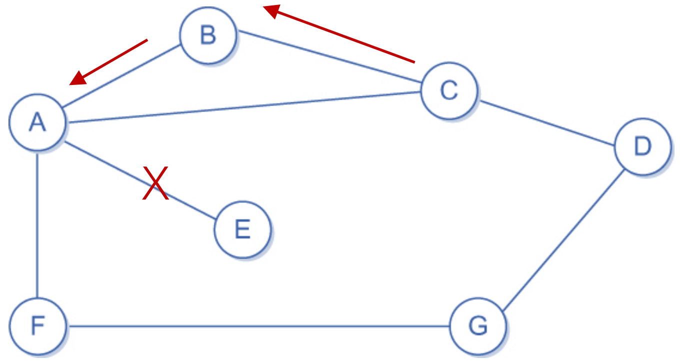
Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	inf	3	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0



Distance Vector Algorithm

- Bad News Travels Slow

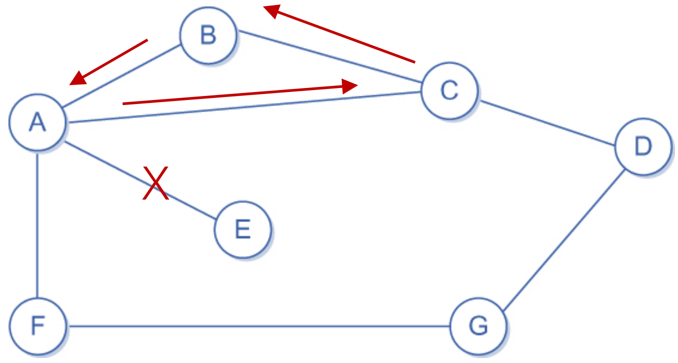
Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	4	3	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0



Distance Vector Algorithm

- Bad News Travels Slow

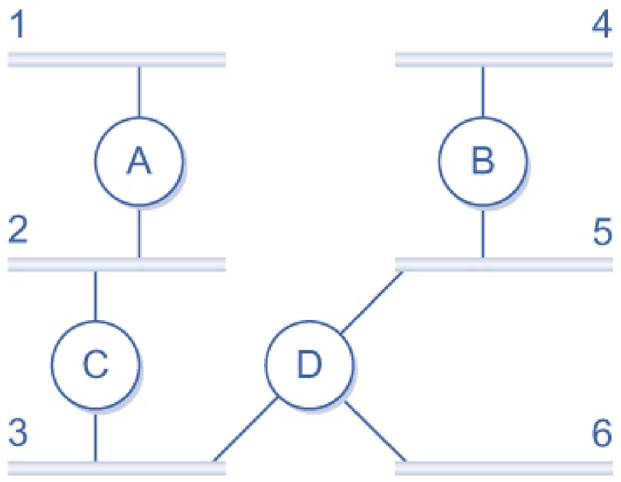
Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	4	3	5	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0



Count-to-infinity Problem

Routing Information Protocol (RIP)

- Included in BSD-UNIX distribution in 1982
- Use distance vector algorithm
  - Distance metric: # hops (max = 15 hops), each link has cost 1
  - Distance Vectors exchanged with neighbors every 30 sec in response message
    - Each message: list of up to 25 destination subnets



Routing Table A

SubnetNum	Distance	NextHop
1	0	Net1
2	0	Net2
3	1	C
4	3	C
5	2	C
6	2	C

Forwarding Table vs. Routing Table

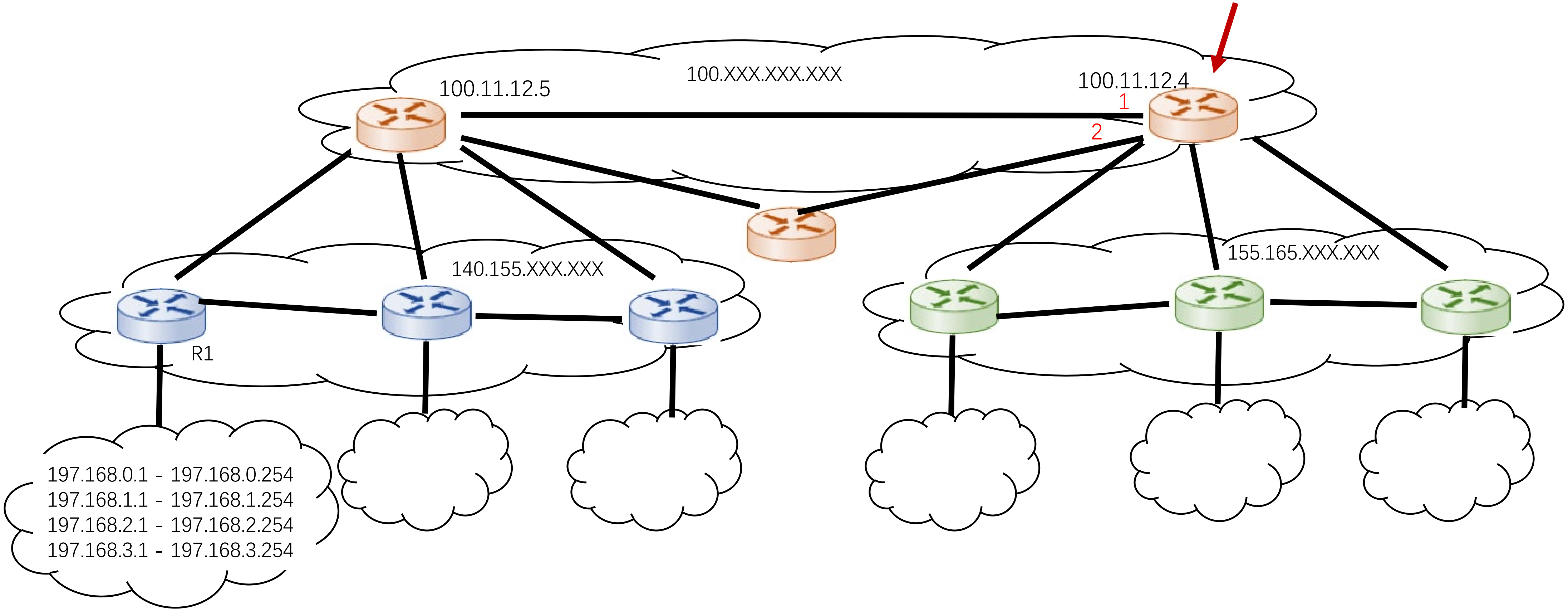
- Forwarding table
  - Determines local forwarding
    - Optimized for looking up an address when forwarding a packet
  - Normally in hardware
  - Contains the mapping from network numbers to outgoing interfaces and some MAC addresses
- Routing table
  - Built by the routing algorithm as a precursor to build the forwarding table
    - Optimized for calculating changes in network topology
  - Normally in software
  - Contains mapping from network numbers to next hops

Routing Table

SubnetNum	NextHop
197.168.0.0/22	100.11.12.5

Forwarding Table

destaddress	Interface	MAC
197.168.0.0/22	1	AB.CD.EF.12.34.56



Reference

- Textbook 3.3