

CSCI 491-01

Topics: Internet Programming

Fall 2008

Network Layer

Derek Leonard

Hendrix College

November 10, 2008

Original slides copyright © 1996-2007 J.F Kurose and K.W. Ross

Chapter 4: Roadmap

4.1 Introduction

4.2 Virtual circuit and datagram networks

4.3 What's inside a router

4.4 IP: Internet Protocol

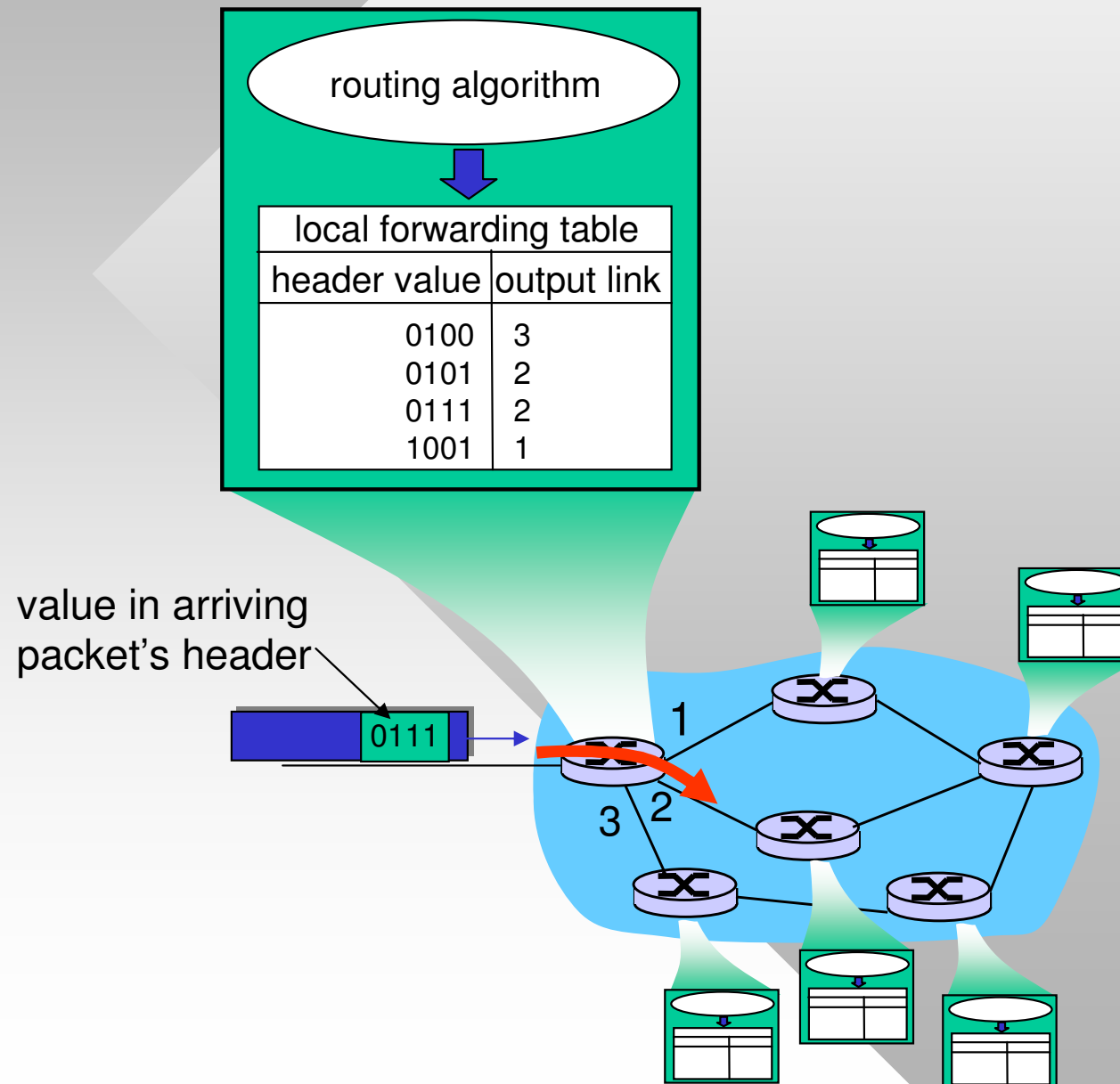
4.5 Routing algorithms

- Link state
- Distance Vector
- Hierarchical routing

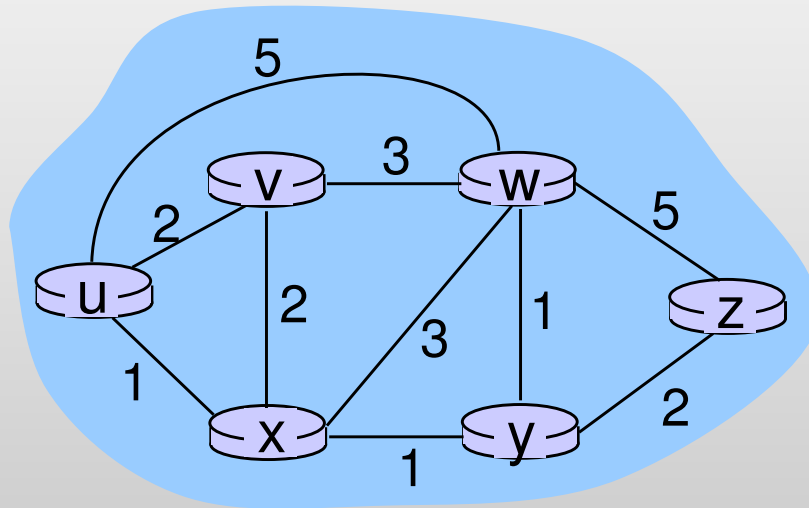
4.6 Routing in the Internet

4.7 Broadcast and multicast routing

Interplay Between Routing and Forwarding



Graph Abstraction



Graph: $G = (V, E)$

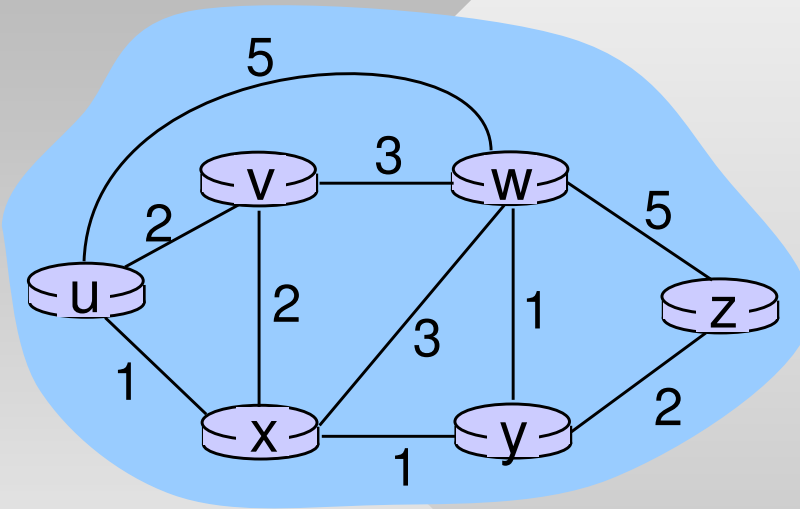
$V =$ set of routers $= \{u, v, w, x, y, z\}$

$E =$ set of links $= \{ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) \}$

Remark: Graph abstraction is useful in other network contexts

Example: P2P, where V is set of peers and E is set of TCP connections

Graph Abstraction: Costs



- $c(x,y)$ = cost of link (x,y)
 - e.g., $c(w,z) = 5$
- Cost options:
 - Could always be 1
 - Could be inversely related to bandwidth or be proportional to congestion
 - Physical distance

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

Question: What's the least-cost path between u and z ?

Routing algorithm: algorithm that finds least-cost path

Routing Algorithm Classification

Global or local information?

Global:

- All routers have complete topology, link cost info
- “Link state” algorithms

Local (decentralized):

- Router knows physically-connected neighbors, link costs to neighbors
- Iterative process of computation, exchange of info with neighbors
- “Distance vector” algorithms

Static or dynamic?

Static:

- Useful when routes change slowly over time
- Manual route change

Dynamic:

- Routes change more quickly
 - Periodic update
 - In response to link cost changes

Chapter 4: Roadmap

4.1 Introduction

4.2 Virtual circuit and datagram networks

4.3 What's inside a router

4.4 IP: Internet Protocol

4.5 Routing algorithms

- Link state
- Distance Vector
- Hierarchical routing

4.6 Routing in the Internet

4.7 Broadcast and multicast routing

Simple Link-State Routing Algorithm

Dijkstra's algorithm

- Network topology, link costs known to all nodes
 - Accomplished via “link state broadcast”
 - All nodes have same info
- Computes least cost paths from one node (“source”) to all other nodes
 - Gives **forwarding table** for that node
- Iterative: after k iterations, know least cost path to k destinations

Notation:

- $c(x,y)$: link cost from x to y
 - Cost is ∞ if not direct neighbors
- $D(v)$: current estimate of the cost from source to destination v
- $p(v)$: predecessor of v along the shortest path back to source
- N : set of nodes whose least cost path is definitively known

Dijkstra's Algorithm

Initialization:

$N = \{u\}, D(u) = 0$

for all nodes $v \neq u$

if v is adjacent to u

$$D(v) = c(u, v)$$

else

$$D(v) = \infty$$

do {

find node a not in N such that $D(a)$ is minimum

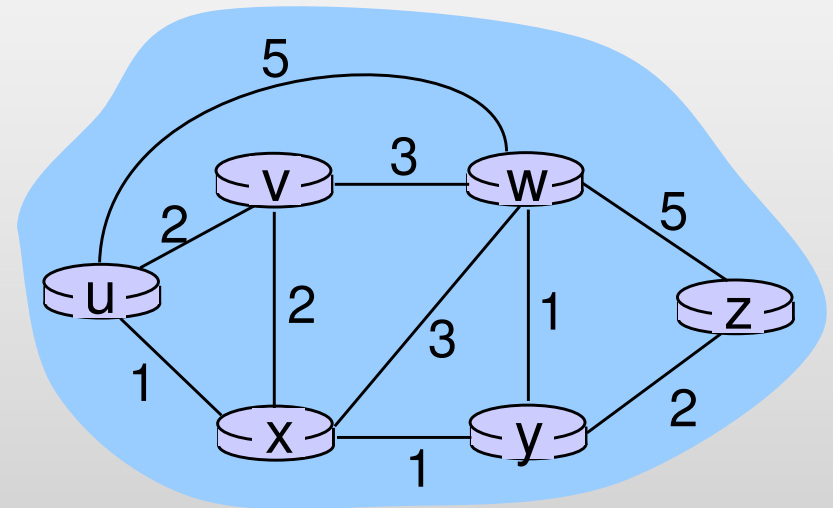
add a to N

for all b adjacent to a and not in N :

$$D(b) = \min(D(b), D(a) + c(a, b))$$

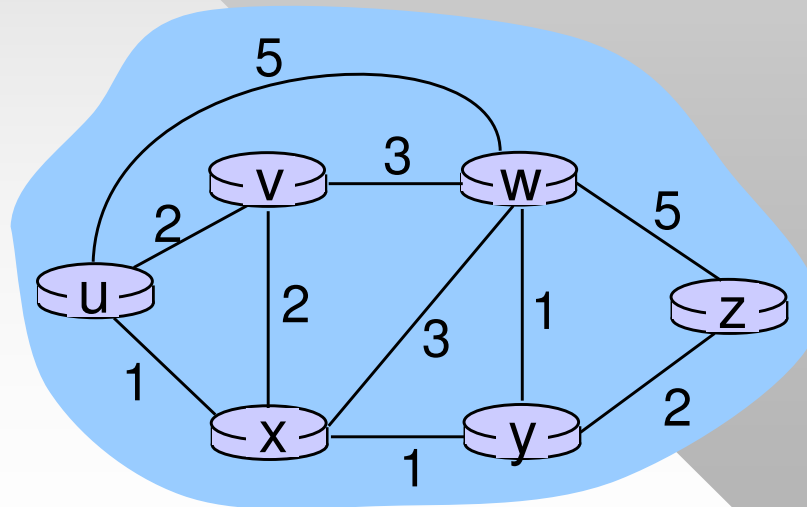
/ new cost to b is either old cost to b or known shortest path cost to a plus cost from a to b */*

} while (not all nodes in N)



Dijkstra's Algorithm: Example

Step	N	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	u	$2, u$	$5, u$	$1, u$	∞	∞
1	$ux \leftarrow$	$2, u$	$4, x$		$2, x$	∞
2	$uxy \leftarrow$	$2, u$	$3, y$			$4, y$
3	$uxyv \leftarrow$		$3, y$			$4, y$
4	$uxyvw \leftarrow$					$4, y$
5	$uxyvwz \leftarrow$					



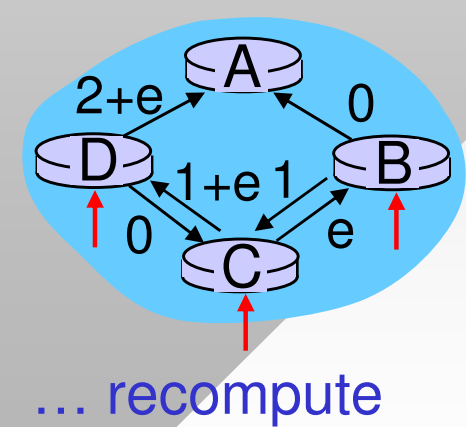
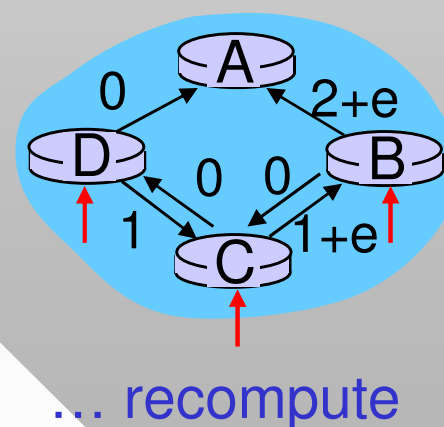
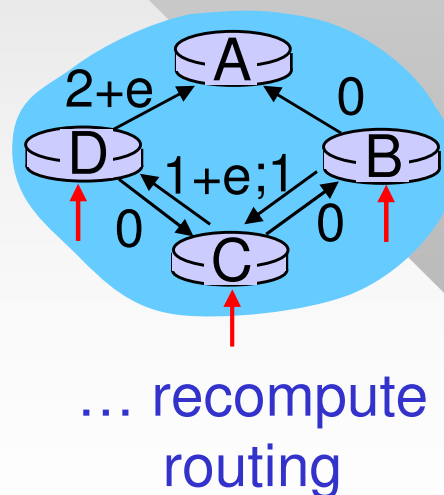
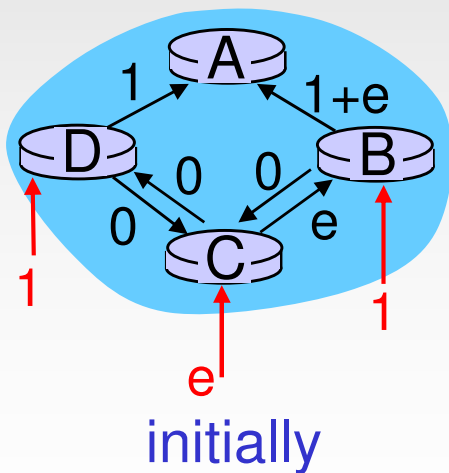
Dijkstra's Algorithm Discussion

Algorithm complexity: n nodes

- Each iteration: need to check all nodes a not in N
- Total: $n(n-1)/2$ comparisons, $O(n^2)$ complexity
- More efficient implementations possible: $O(n \log n)$

Oscillations possible:

- e.g., Link cost = amount of carried traffic



Chapter 4: Roadmap

4.1 Introduction

4.2 Virtual circuit and datagram networks

4.3 What's inside a router

4.4 IP: Internet Protocol

4.5 Routing algorithms

- Link state
- **Distance Vector**
- Hierarchical routing

4.6 Routing in the Internet

4.7 Broadcast and multicast routing

Distance Vector Algorithm

- Two metrics known to each node x
 - Estimate $D_x(y)$ of least cost from x to y
 - Cost $c(x,v)$ to each of x 's neighbors
- We can write the above vector as:

$$\vec{D}_x = \{D_x(y) : y \in V\}$$

- Node x periodically asks its neighbors to send their distance vectors
 - Thus, x has access to the following for each neighbor v

$$\vec{D}_v = \{D_v(y) : y \in V\}$$

Distance Vector Algorithm (cont'd)

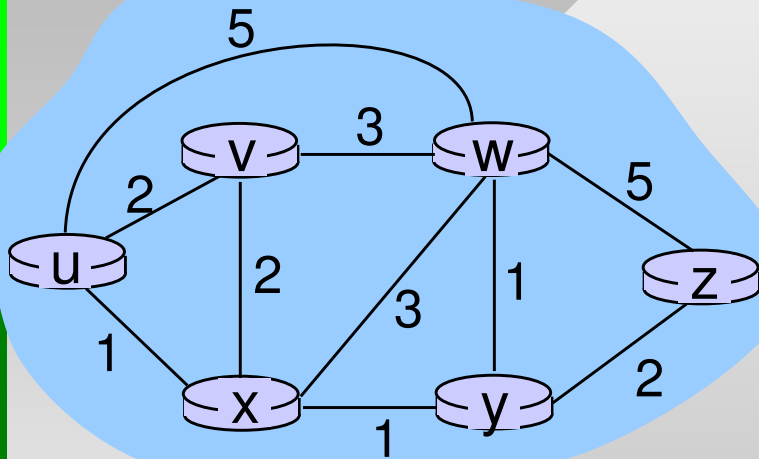
Basic idea (Bellman-Ford):

- When a node x receives new DV estimate from neighbor v , it updates its own DV using the Bellman-Ford equation:

$$D_x(y) \leftarrow \min\{D_x(y), c(x,v) + D_v(y)\}, \forall y \in V$$

- Under minor conditions, the estimate $D_x(y)$ converges to the *actual* least cost $d_x(y)$

Bellman-Ford Example



Clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

$$\begin{aligned} d_u(z) &= \min \{ c(u,v) + d_v(z), \\ &\quad c(u,x) + d_x(z), \\ &\quad c(u,w) + d_w(z) \} \\ &= \min \{ 2 + 5, \\ &\quad 1 + 3, \\ &\quad 5 + 3 \} = 4 \end{aligned}$$

Node that achieves minimum is next hop in shortest path → forwarding table

Distance Vector Algorithm (cont'd)

Iterative, asynchronous:

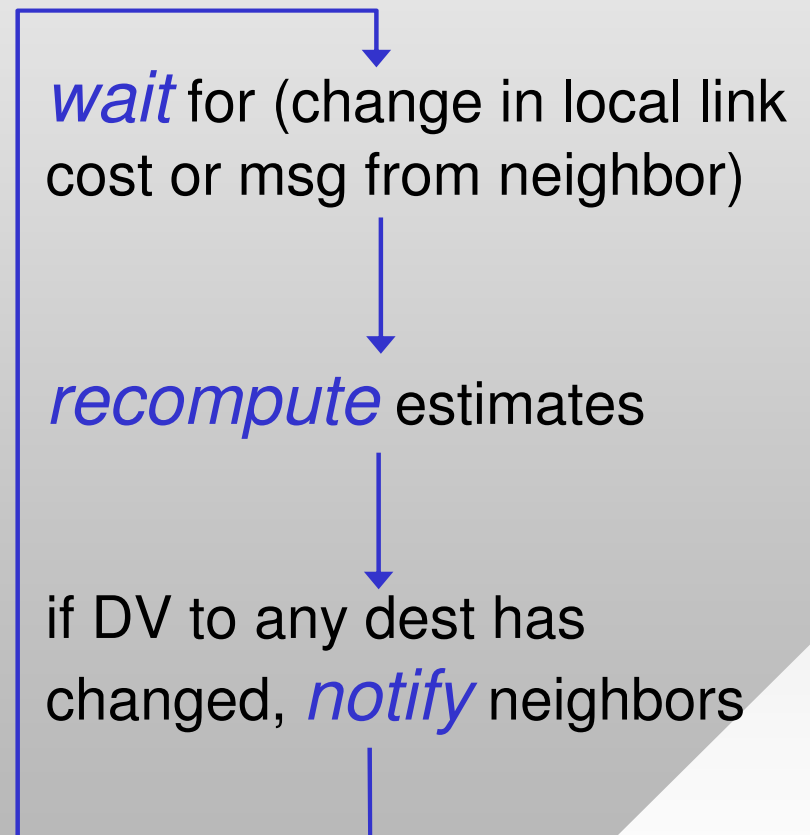
Each local iteration caused by:

- Local link cost change
- DV update message from neighbor

Distributed:

- Each node notifies neighbors *only* when its DV changes
 - Neighbors then notify their neighbors if necessary

Each node:



$$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$$

$$= \min\{2+0, 7+1\} = 2$$

$$D_x(z) = \min\{$$

$$c(x,y) + D_y(z),$$

$$c(x,z) + D_z(z)\}$$

$$= \min\{2+1, 7+0\} = 3$$

node x table

		cost to		
		x	y	z
from	x	0	2	7
	y	∞	∞	∞
	z	∞	∞	∞

node y table

		cost to		
		x	y	z
from	x	∞	∞	∞
	y	2	0	1
	z	∞	∞	∞

node z table

		cost to		
		x	y	z
from	x	∞	∞	∞
	y	∞	∞	∞
	z	7	1	0

		cost to		
		x	y	z
from	x	0	2	3
	y	2	0	1
	z	7	1	0

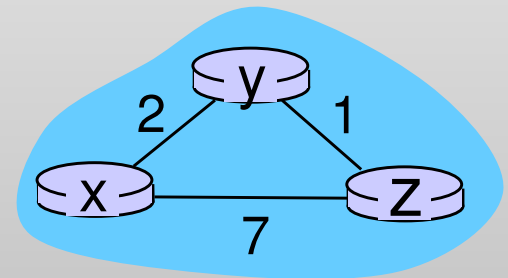
		cost to		
		x	y	z
from	x	0	2	7
	y	2	0	1
	z	7	1	0

		cost to		
		x	y	z
from	x	0	2	7
	y	2	0	1
	z	3	1	0

		cost to		
		x	y	z
from	x	0	2	3
	y	2	0	1
	z	3	1	0

		cost to		
		x	y	z
from	x	0	2	3
	y	2	0	1
	z	3	1	0

		cost to		
		x	y	z
from	x	0	2	3
	y	2	0	1
	z	3	1	0



time →