UNIVERSITY OF BIRMINGHAM

Computer Systems
Week 10b – Network Layer Routing Algorithms

Based on material and slides from
Computer Networking: A Top Down Approach, 7th Edition – Chapter 5
Jim Kurose, Keith Ross
Pearson/Addison Wesley

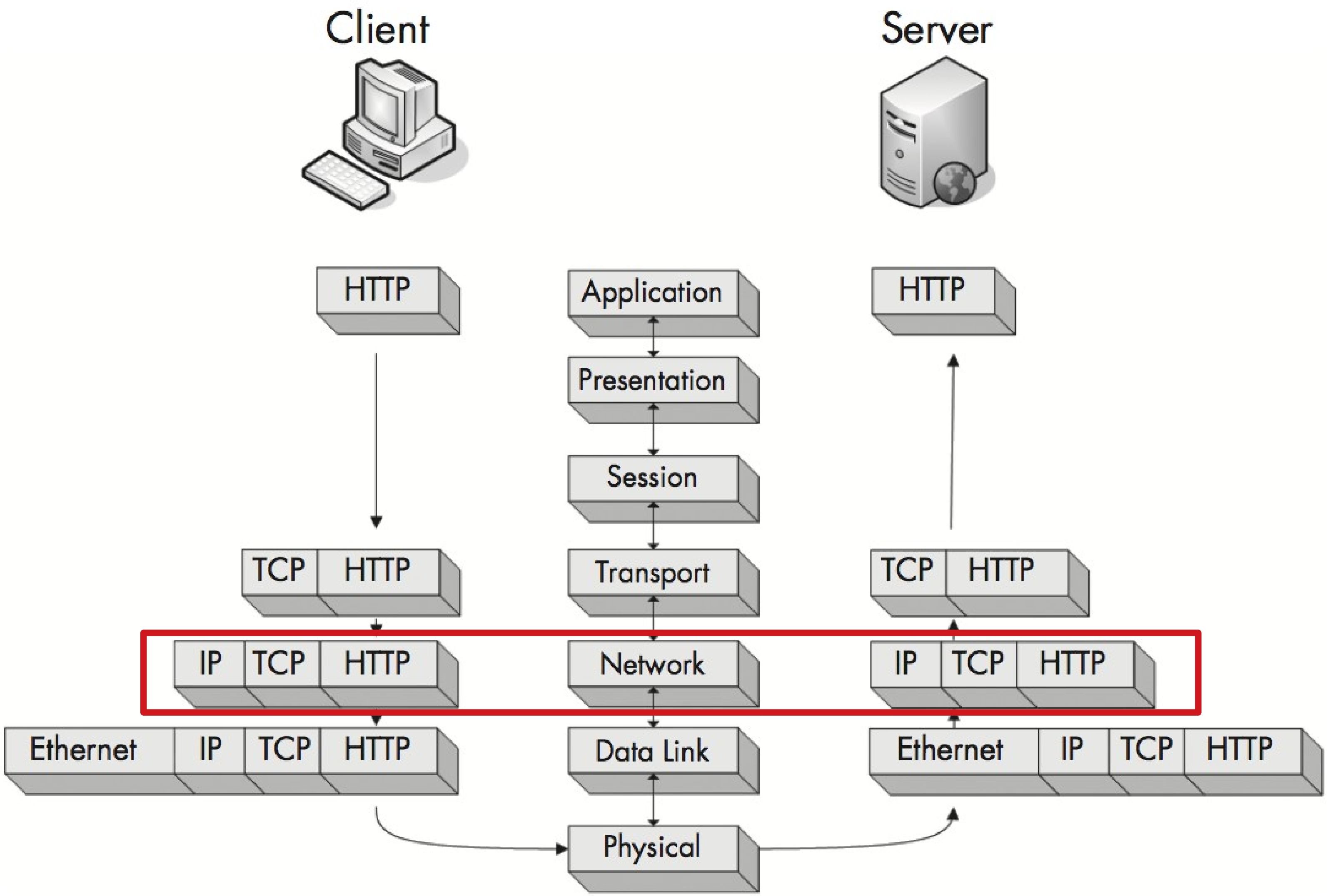
Lecture Objective

The objective of this lecture is to understand the conceptual aspects of network layer routing algorithms

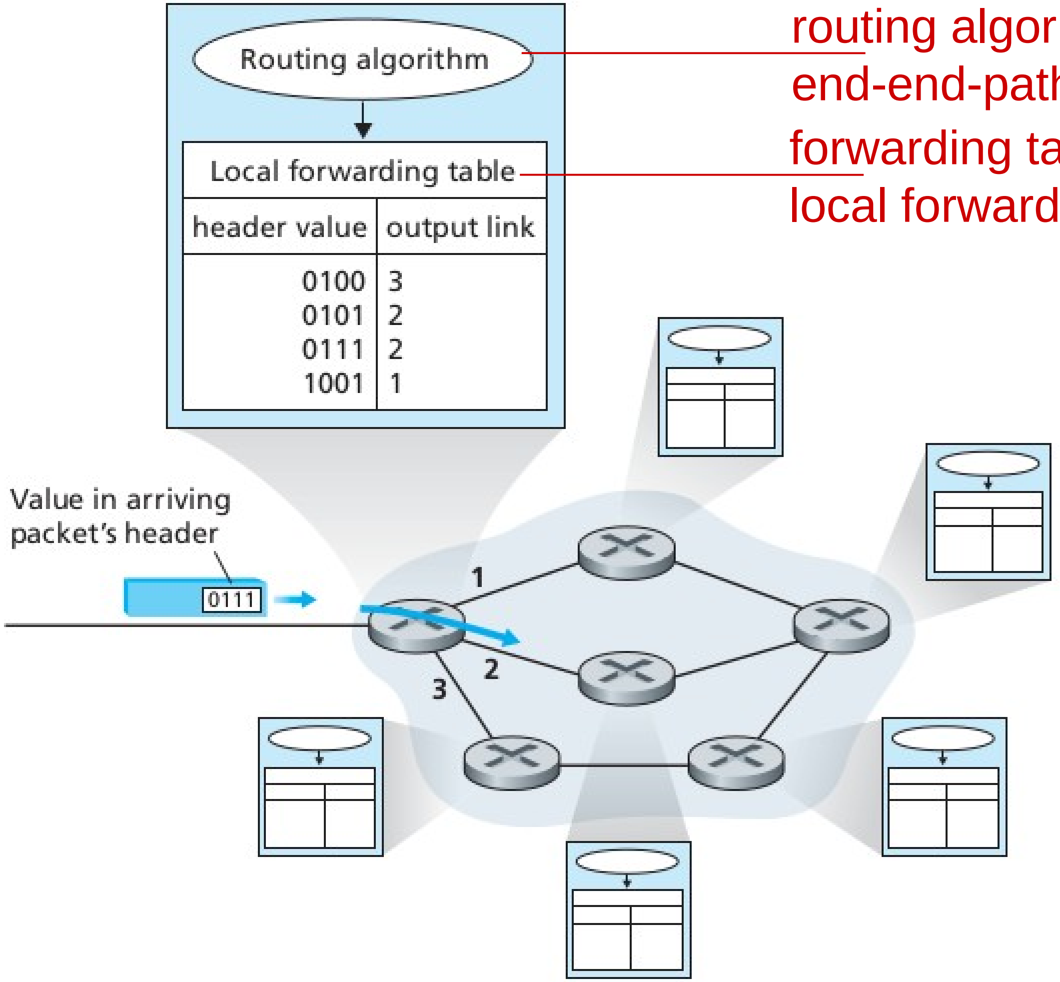
Lecture Outline

- Routing vs. Forwarding – Recap
- Graph Abstraction
- Routing Algorithms – Classification
- Link State Routing – Dijkstra’s Algorithm
- Distance Vector Routing – Bellman-Ford Algorithm
- Hierarchical Routing
- Summary

Recap – Network Layers



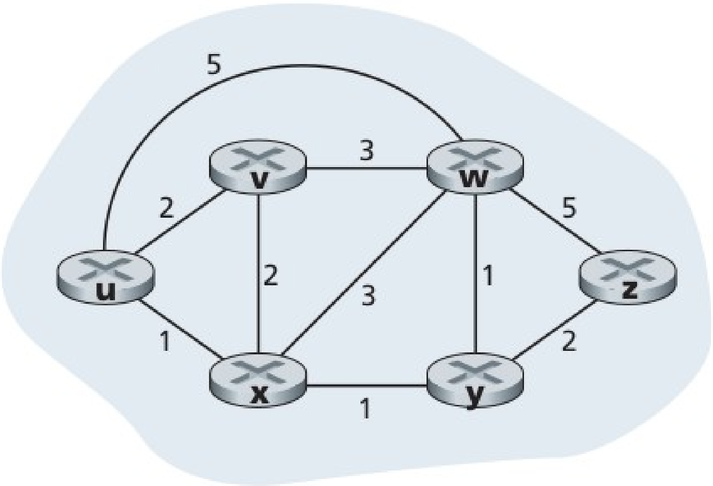
Interplay between Routing & Forwarding



routing algorithm determines end-end-path through network

forwarding table determines local forwarding at this router

Graph Abstraction



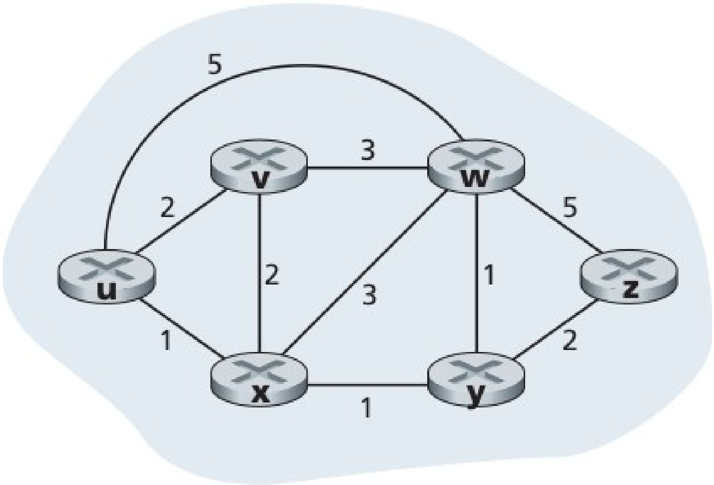
graph: G = (N,E)

N = set of routers = { u, v, w, x, y, z }

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

Aside: graph abstraction is useful in other network contexts, e.g., P2P, where *N* is set of peers and *E* is set of TCP connections

Graph Abstraction



$c(x,x')$ = cost of link (x,x')

e.g., $c(w,z) = 5$

cost could always be 1, or inversely related to bandwidth, or inversely related to congestion

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

Key Question: What is the least-cost path between u and z ?
Routing Algorithm: Algorithm that finds that least cost path

Routing Algorithm Classification

Q: Global or decentralized information?

Global:

- All routers have complete topology, link cost info
- "Link State" algorithms

Decentralized:

- Router knows physically-connected neighbors, link costs to neighbors
- Iterative process of computation, exchange of info with neighbors
- "Distance Vector" algorithms

Q: static or dynamic?

Static:

- Routes change slowly over time

Dynamic:

- Routes change more quickly
  - periodic update
  - in response to link cost changes

A Link-State Algorithm

Dijkstra's Algorithm

- Net 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 node x to y; = ∞ if not direct neighbors
- **D(v)**: current value of cost of path from source to dest. v
- **p(v)**: predecessor node along path from source to v
- **N'**: set of nodes whose least cost path definitively known

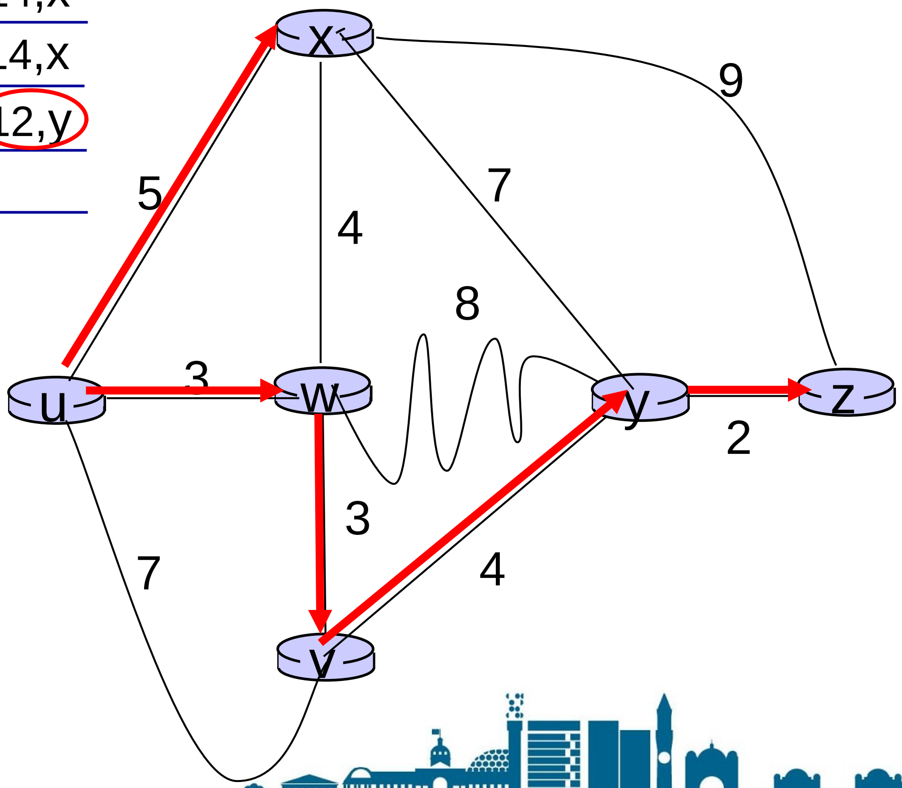
Dijkstra's Algorithm

```
Initialization:
  N' = {u}
  for all nodes v
    if v adjacent to u
        then D(v) = c(u,v)
    else D(v) = ∞

 Loop
   find w not in N' such that D(w) is a minimum
   add w to N'
   update D(v) for all v adjacent to w and not in N' :
      D(v) = min( D(v), D(w) + c(w,v) )
   /* new cost to v is either old cost to v or known
    shortest path cost to w plus cost from w to v */
  until 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	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					

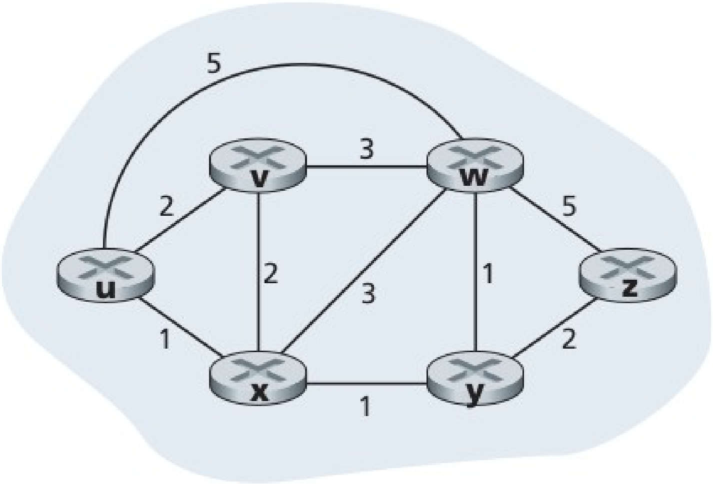


Notes:

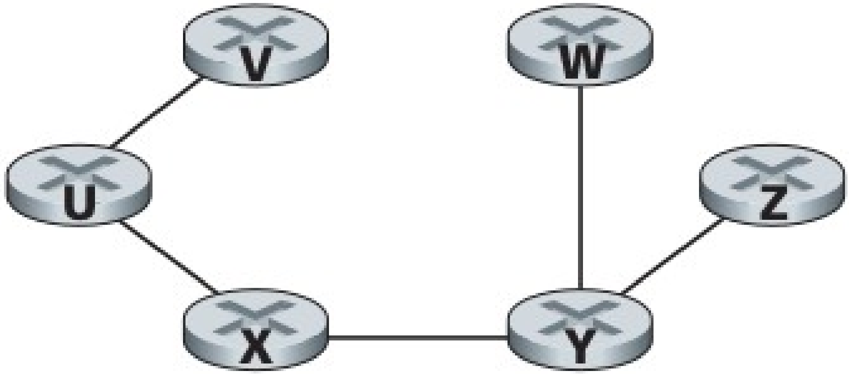
- construct shortest path tree by tracing predecessor nodes
- ties can exist (can be broken arbitrarily)

Dijkstra's Algorithm – Another 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	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



Dijkstra's Algorithm – Least Cost Path and Forwarding Table for Node u



Destination	Link
v	(u, v)
w	(u, x)
x	(u, x)
y	(u, x)
z	(u, x)

Resulting shortest-path tree from u:

Resulting forwarding table in u:

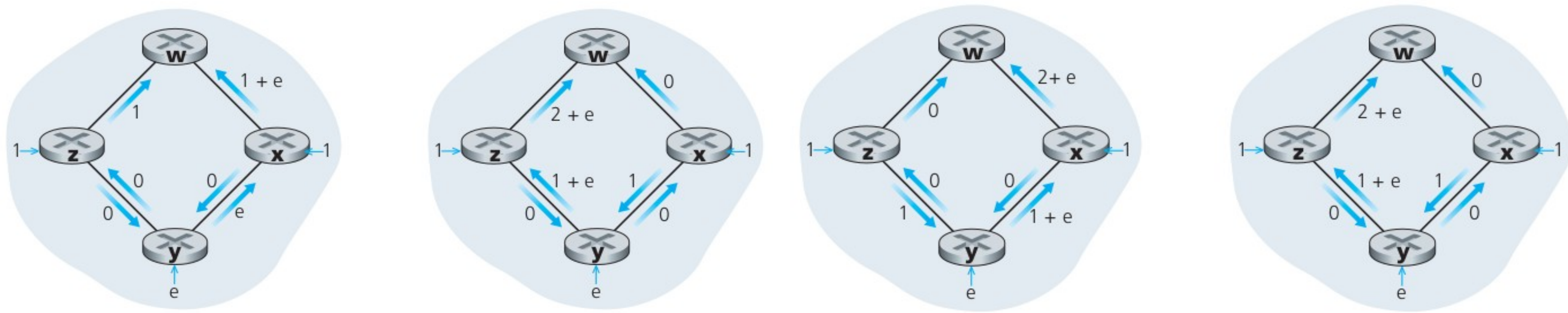
Dijkstra’s Algorithm – Complexity

Algorithm Complexity*:* n nodes

- Each iteration: need to check all nodes, w, not in N’
- n(n+1)/2 comparisons: $O(n^2)$
- More efficient implementations possible: O(n logn)

Dijkstra's Algorithm – Oscillations

Support link cost equals amount of carried traffic.



a. Initial routing

b. *x*, *y* detect better path to *w*, clockwise

c. *x*, *y*, *z* detect better path to *w*, counterclockwise

d. *x*, *y*, *z*, detect better path to *w*, clockwise

Given these costs, find new routing….
resulting in new costs

Distance-Vector (DV) Routing Algorithm

Bellman-Ford equation:

let

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

then

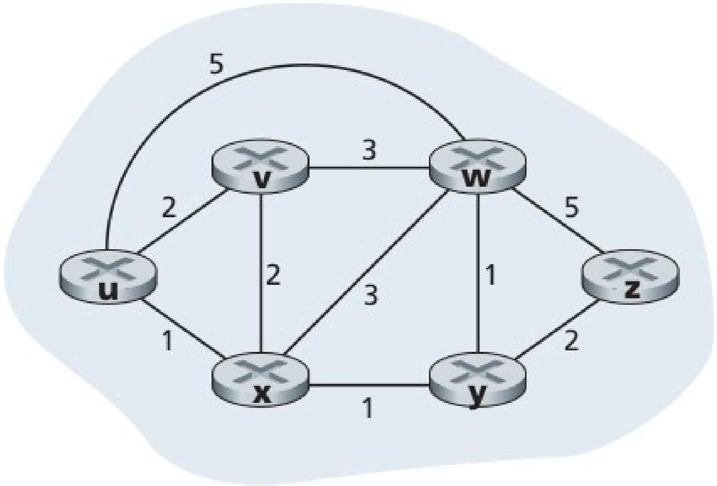
$$d_x(y) = min_v \{c(x,v) + d_v(y)\}$$

cost from neighbor v to destination y

cost to neighbor v

min taken over all neighbors v of x

Bellman-Ford – Example



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

B-F equation gives:

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

Node achieving minimum is next-hop in shortest path, used in forwarding table.

Distance Vector Algorithm

- $D_x(y)$ = estimate of least cost from x to y
  - x maintains distance vector $\mathbf{D}_x = [D_x(y): y \in N]$
- Node x:
  - knows cost to each neighbor v: $c(x,v)$
  - maintains its neighbors' distance vectors. For each neighbor v, x maintains:
    $\mathbf{D}_v = [D_v(y): y \in N]$

Distance Vector Algorithm

Key Idea:

- From time-to-time, each node sends its own distance vector estimate to neighbors
- When x receives new DV estimate from neighbor, it updates its own DV using B-F equation:

$$D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\} \quad \textit{for each node } y \in N$$

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

Distance Vector Algorithm

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:

wait for (change in local link cost or msg from neighbor)

↓

recompute estimates

↓

if DV to any dest has changed, *notify* neighbors

$$D_x(y) = \min\{c(x,y) + D_y(y),\ c(x,z) + D_z(y)\}$$
$$= \min\{2+0\ ,\ 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y) + D_y(z),\ c(x,z) + D_z(z)\}$$
$$= \min\{2+1\ ,\ 7+0\} = 3$$

node x table

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

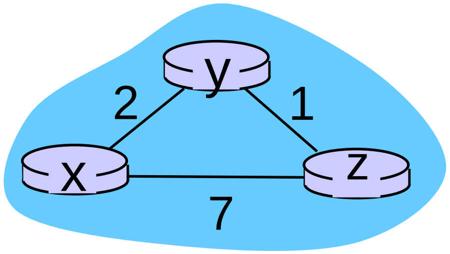
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

node y table

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

node z table

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



time

$$D_x(y) = \min\{c(x,y) + D_y(y),\ c(x,z) + D_z(y)\} = \min\{2+0\ ,\ 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y) + D_y(z),\ c(x,z) + D_z(z)\} = \min\{2+1\ ,\ 7+0\} = 3$$

node x table

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

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

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

node y table

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

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

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

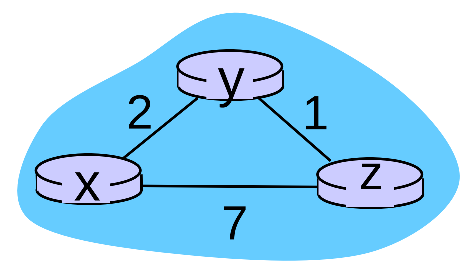
node z table

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

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

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

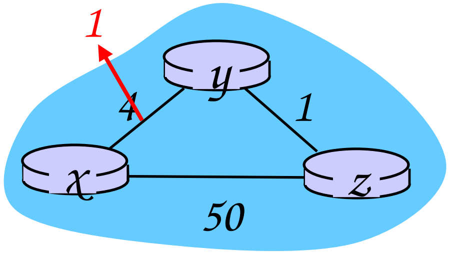
time →



Distance Vector – Link Cost Changes

Link Cost Changes:

- Node detects local link cost change
- Updates routing info, recalculates distance vector
- If DV changes, notify neighbors



t_0 : *y* detects link-cost change, updates its DV, informs its neighbors.

t_1 : *z* receives update from *y*, updates its table, computes new least cost to *x* , sends its neighbors its DV.

t_2 : *y* receives *z*'s update, updates its distance table. *y*'s least costs do *not* change, so *y* does *not* send a message to *z*.

"Good news travels fast!"

What would happen if the link cost increases?

Comparison of LS and DV Algorithms

Message Complexity

- **LS**: with N nodes, E links, O(NE) messages sent
- **DV**: exchange between neighbors only
  - convergence time varies

Speed of Convergence

- **LS**: $O(N^2)$ algorithm requires O(NE) messages
  - may have oscillations
- **DV**: convergence time varies
  - may be routing loops
  - count-to-infinity problem

Robustness: what happens if router malfunctions?

LS:

- node can advertise incorrect ***link*** cost
- each node computes only its *own* table

DV:

- DV node can advertise incorrect ***path*** cost
  - each node’s table used by others
  - error propagate through the network

Hierarchical Routing

Our routing study thus far - idealization

- All routers identical
- Network is "flat" … *not* true in practice

Scale: with 600 million destinations:

- Can't store all dest's in routing tables!
- Routing table exchange would swamp links!

淹没

Administrative Autonomy

- Internet = network of networks
- Each network admin may want to control routing in its own network

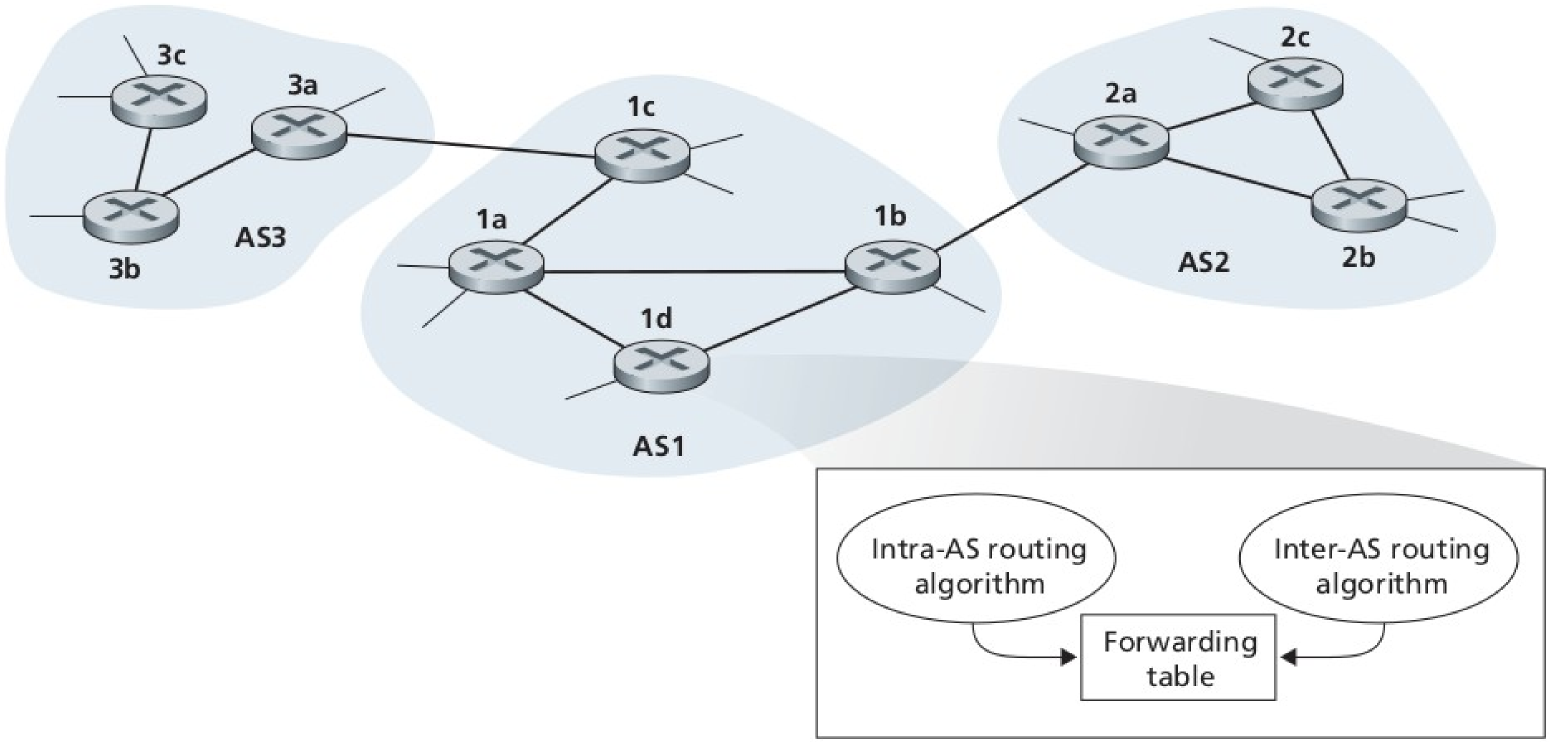
Hierarchical Routing

- Aggregate routers into regions, "Autonomous Systems" (AS)
- Routers in same AS run same routing protocol
  - "intra-AS" routing protocol
  - routers in different AS can run different intra-AS routing protocol

Gateway Router:

- At "edge" of its own AS
- Has link to router in another AS

Interconnected ASes



- Forwarding table configured by both intra- and inter-AS routing algorithm
  - intra-AS sets entries for internal destinations
  - inter-AS & intra-AS sets entries for external destinations

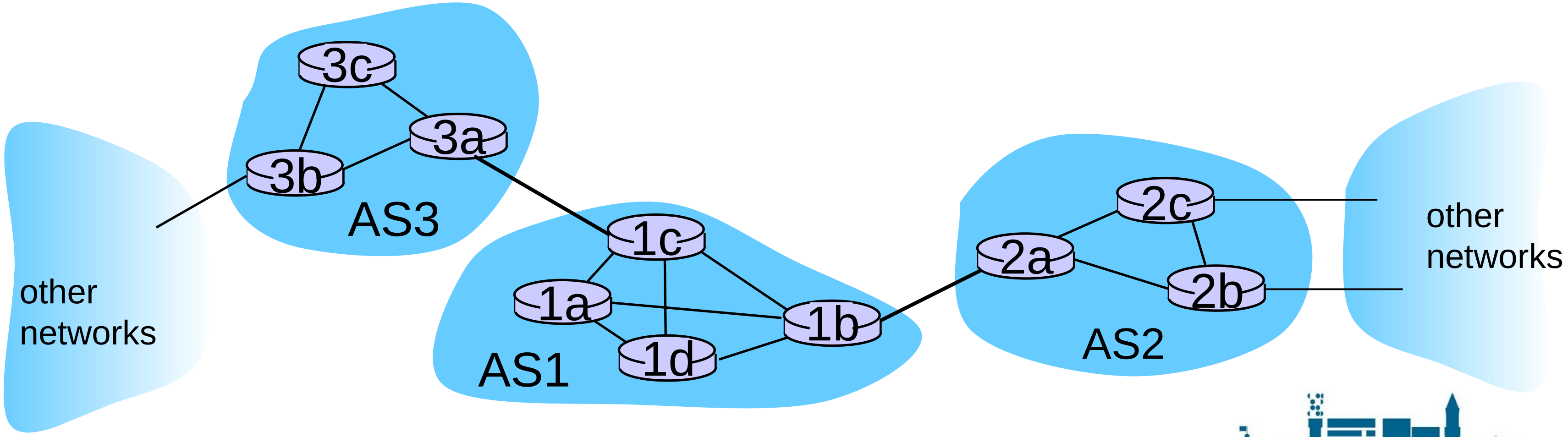
Inter-AS Tasks

- Suppose a router (1d) in AS1 receives datagram destined outside of AS1:
  - router should forward packet to gateway router, but which one?

AS1 must:

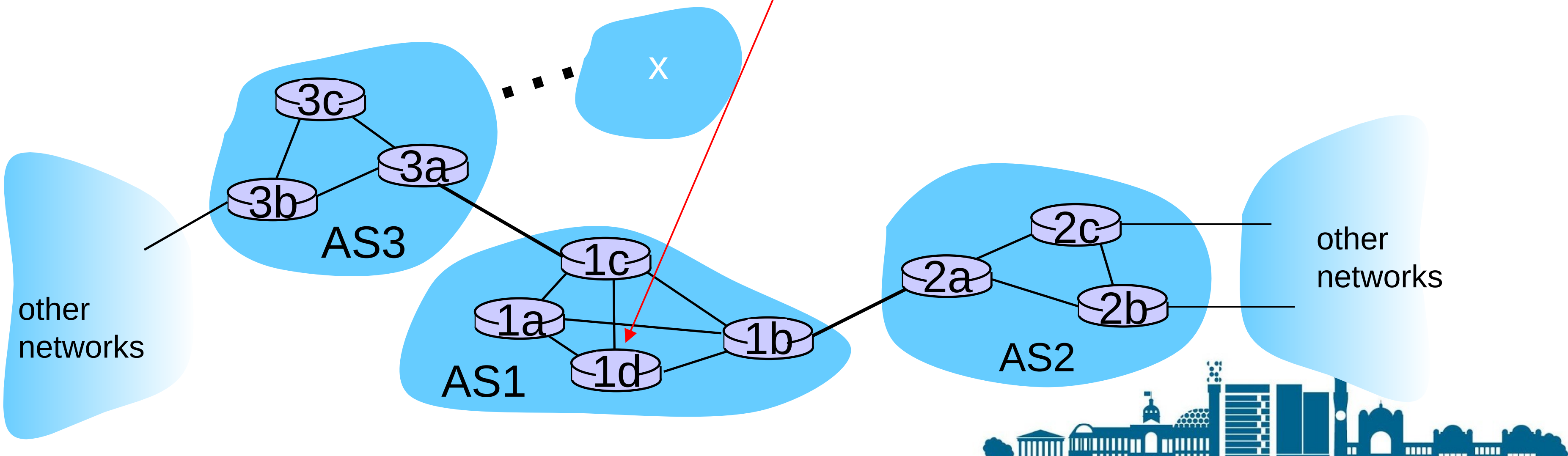
1) Learn which dests are reachable through AS2, which through AS3
2) Propagate this reachability info to all routers in AS1

Job of inter-AS routing!



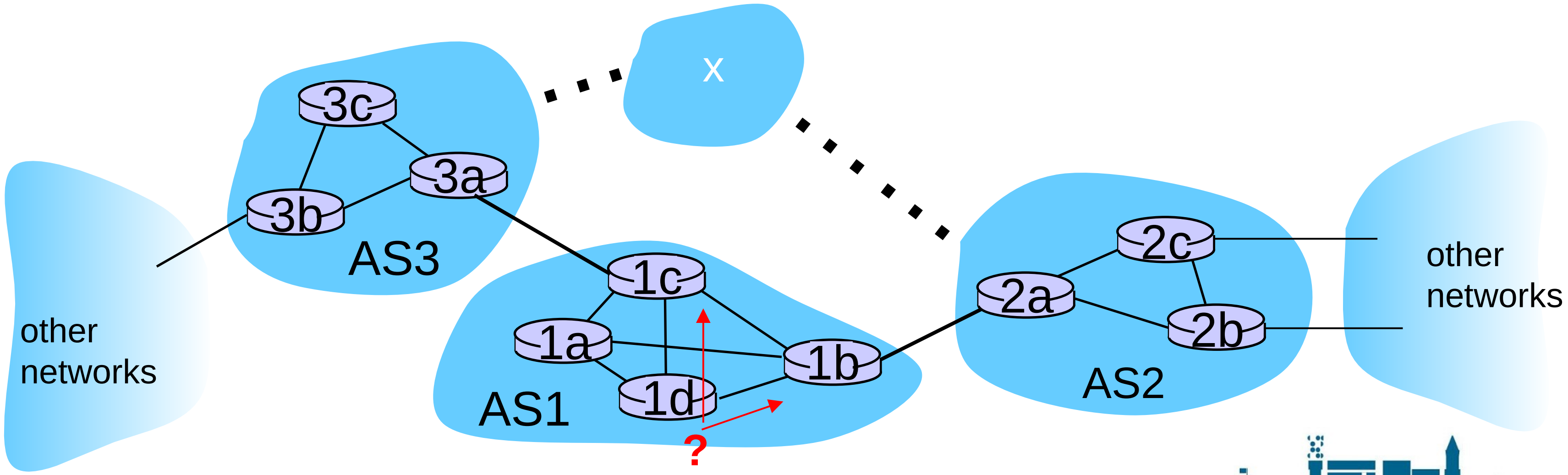
Example: Setting Forwarding Table in Router 1d

- Suppose AS1 learns (via inter-AS protocol) that the subnet x is reachable via AS3 (gateway 1c), but not via AS2
  - inter-AS protocol propagates reachability info to all internal routers
- Router 1d determines from intra-AS routing info that its interface I is on the least cost path to 1c
  - installs forwarding table entry *(x,I)*



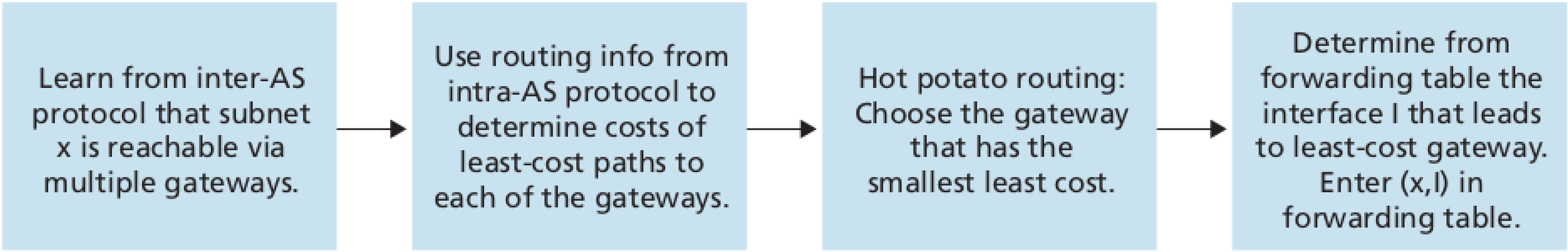
Example: Choosing Among Multiple ASes

- Now suppose AS1 learns from inter-AS protocol that subnet *x* is reachable from AS3 *and* from AS2.
- To configure forwarding table, router 1d must determine which gateway it should forward packets towards for dest x
  - This is also job of inter-AS routing protocol!



Example: Choosing Among Multiple ASes

- Now suppose AS1 learns from inter-AS protocol that subnet *x* is reachable from AS3 *and* from AS2.
- To configure forwarding table, router 1d must determine which gateway it should forward packets towards for dest x
  - This is also job of inter-AS routing protocol!
- *Hot Potato Routing:* send packet towards closest of two routers.



Summary

In this lecture, we have seen:

- The two main classes of routing algorithms i.e. Link State and Distance Vector routing.
- Dijkstra’s Algorithm is a Link State algorithm.
- Bellman-Ford as a Distance Vector algorithm.
- The importance of Hierarchical Routing.

References / Links

- Chapter #5: **The Network Layer: Control Plane,**

  Computer Networking: A Top-Down Approach (7^{th} edition) by Kurose & Ross