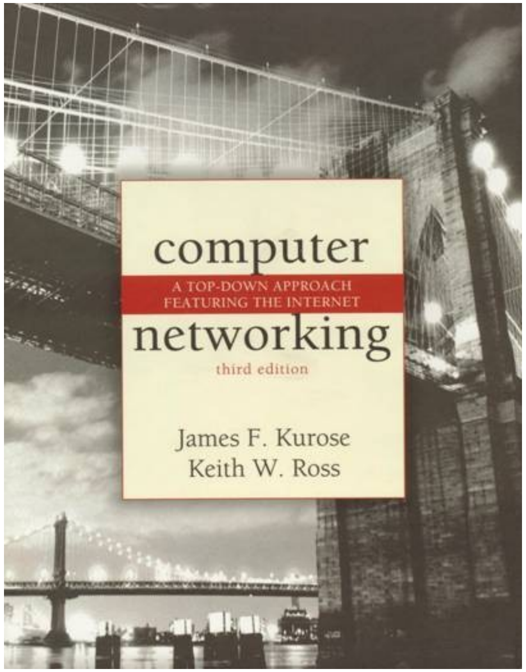
Chapter 4: Routing Algorithms
Revised by Quan Le-Trung, Dr.techn.

http://sites.google.com/site/quanletrung/



Computer Networking: A Top Down Approach Featuring the Internet, 3rd edition.
Jim Kurose, Keith Ross
Addison-Wesley, July 2004.

A note on the use of these ppt slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

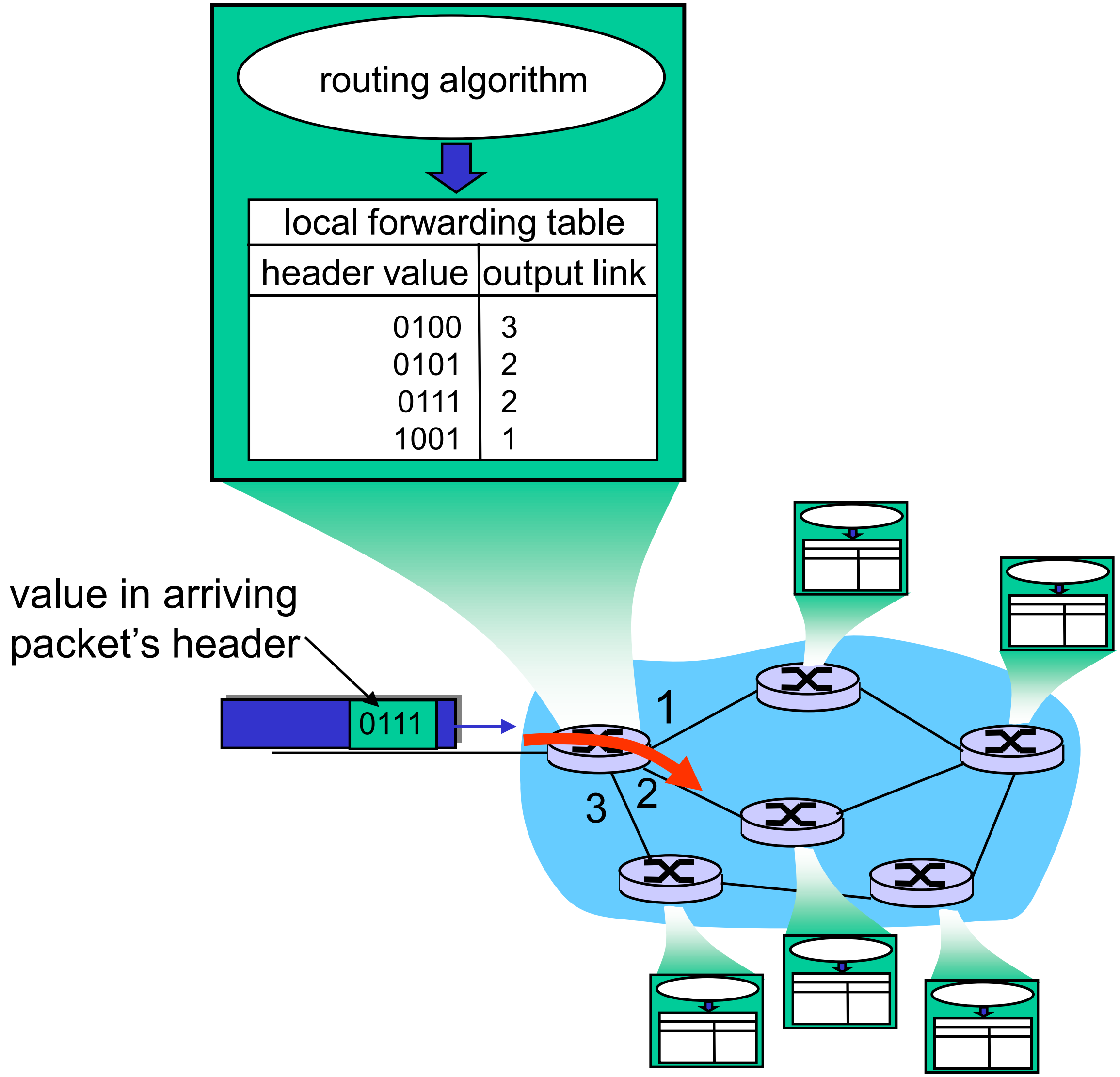
- If you use these slides (e.g., in a class) in substantially unaltered form, that you mention their source (after all, we'd like people to use our book!)
- If you post any slides in substantially unaltered form on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

Routing Algorithms

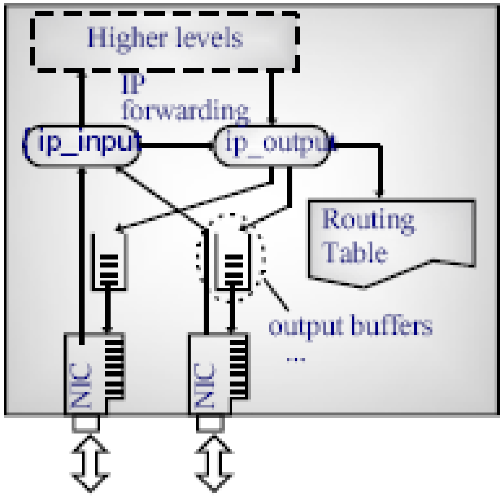
- Link-State Routing
- Distance-Vector Routing
- Hierarchical Routing

Interplay between routing and forwarding



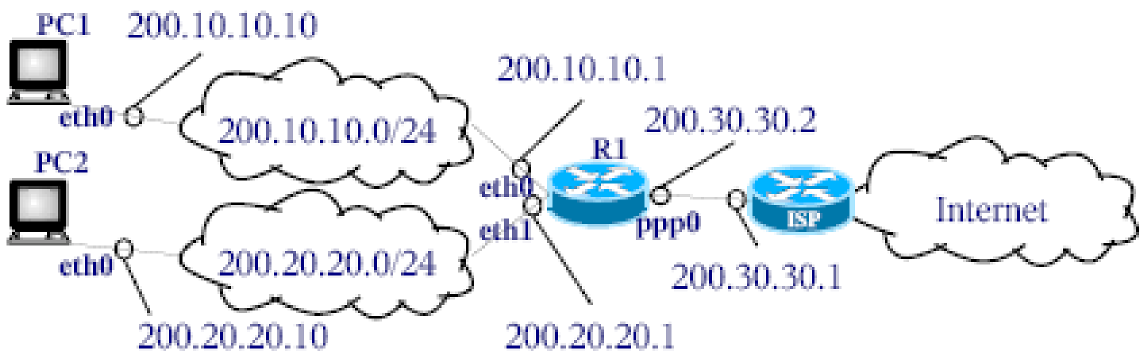
Routing Table

- ip_output() kernel function consults the routing table for each datagram.
- Routing can be:
    - Direct: The destination is directly connected to an interface.
    - Indirect: Otherwise. In this case, the datagram is sent to a router.
- Default route: Is an entry where to send all datagrams with a destination address to a network not present in the routing table. The default route address is 0.0.0.0/0.
- Hosts usually have two entries: The network where they are connected and a default route.



Basic router architecture

Routing Table – Unix Example



known destinations

how to reach the destinations

PC1 routing table:

Destination	Genmask	Gateway	Iface
200.10.10.0	255.255.255.0	0.0.0.0	eth0
0.0.0.0	0.0.0.0	200.10.10.1	eth0

PC2 routing table:

Destination	Genmask	Gateway	Iface
200.20.20.0	255.255.255.0	0.0.0.0	eth0
0.0.0.0	0.0.0.0	200.20.20.1	eth0

R1 routing table:

Destination	Genmask	Gateway	Iface
200.10.10.0	255.255.255.0	0.0.0.0	eth0
200.20.20.0	255.255.255.0	0.0.0.0	eth1
0.0.0.0	0.0.0.0	200.30.30.1	ppp0

Routing Table – Datagram Delivery Algorithm

- Check if it is the destination:

```
if(Datagram Destination == address of any of the interfaces) {
    send the datagram to the loopback (send to upper layers)
}
```

- Consult the routing table:

```
for each routing table entry ordered from longest to shortest mask
   (Longest Prefix Match) {
  if(Datagram Destination IP address & mask == Destination table
     entry) {
      return (gateway, interface) ;
  }
```

- Forward the datagram

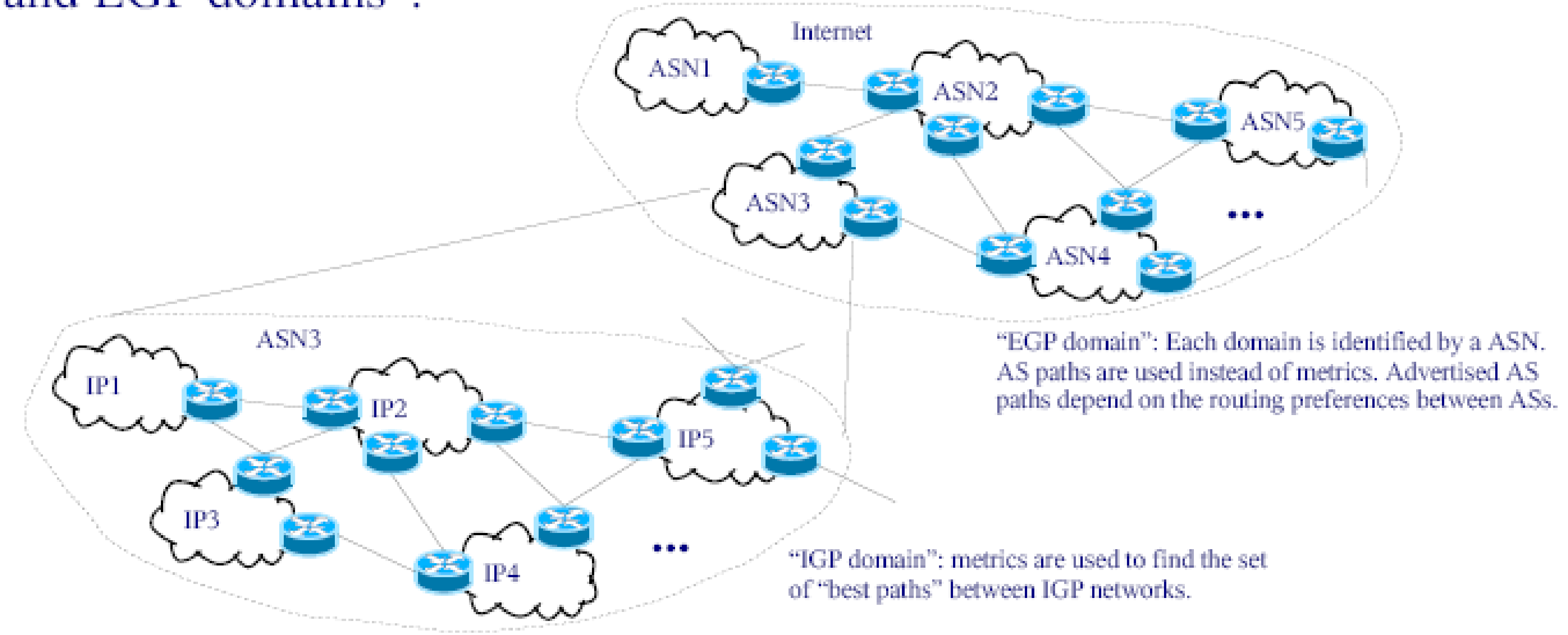
```
if(it is a direct routing) {
    send the datagram to the Datagram Destination IP address by
       the entry interface
} else { /* it is an indirect routing */
    send the datagram to the gateway IP address by the entry
       interface
}
```

Routing algorithms

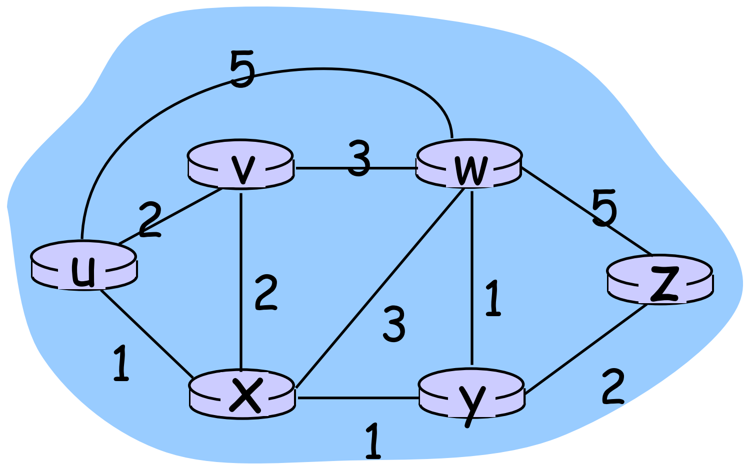
- Add entries to routing tables. Can be:
  - Static: Manual, scripts, DHCP.
  - Adaptive: Automatically update table entries, e.g. when a topology change occurs.
- Internet is organized in Autonomous Systems (AS). In terms of ASs, routing algorithms are classified as:
  - Interior Gateway Protocols (IGPs): Inside the same AS. Examples:
    - RFC standards: RIP, OSPF.
    - Proprietary: CISCO IGRP.
  - Exterior Gateway Protocols (EGPs): Between different ASs. Currently BGPv4.

Routing algorithms - Autonomous Systems (AS)

- AS definition (RFC 1930): "An AS is a connected group of one or more IP prefixes run by one or more network operators which has a SINGLE and CLEARLY DEFINED routing policy".
- Each AS is identified by a 16 bits AS Number (ASN) assigned by IANA.
- ASs facilitate Internet routing by introducing a two-level hierarchy: "IGP and EGP domains".



Graph abstraction



Graph: G = (N,E)

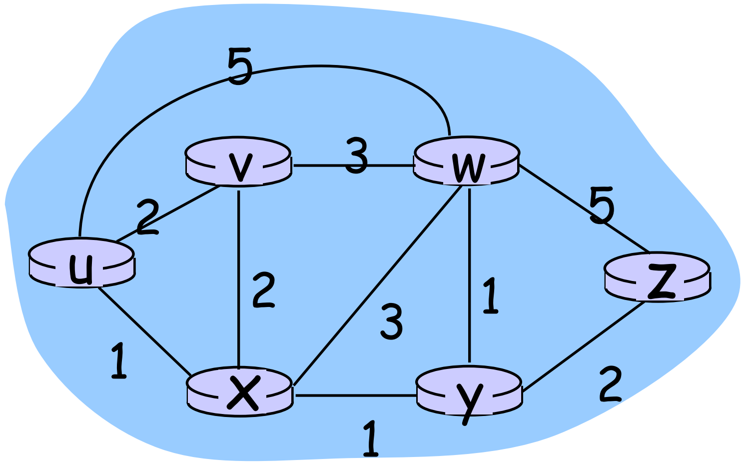
N = set of routers = { u, v, w, x, y, z }

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 N is set of peers and E is set of TCP connections

Graph abstraction: costs



- $c(x,x')$ = 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 $(x_1, x_2, x_3,\ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

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

Routing algorithm: algorithm that finds least-cost path

Routing Algorithm classification

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

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 Routing 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 dest.'s

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

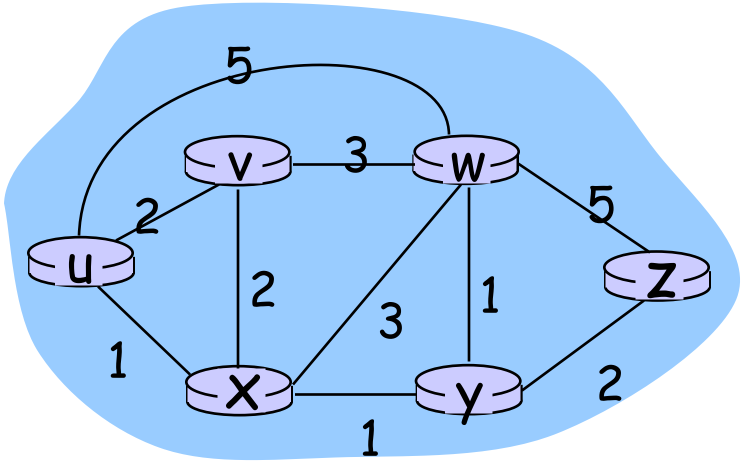
Dijsktra's Algorithm

```
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	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					



Quiz_01

- Run the Dijkstra algorithm on a specific graph, step-by-step and find all the shortest path!

Distance Vector Algorithm (1)

Bellman-Ford Equation (dynamic programming)

Define

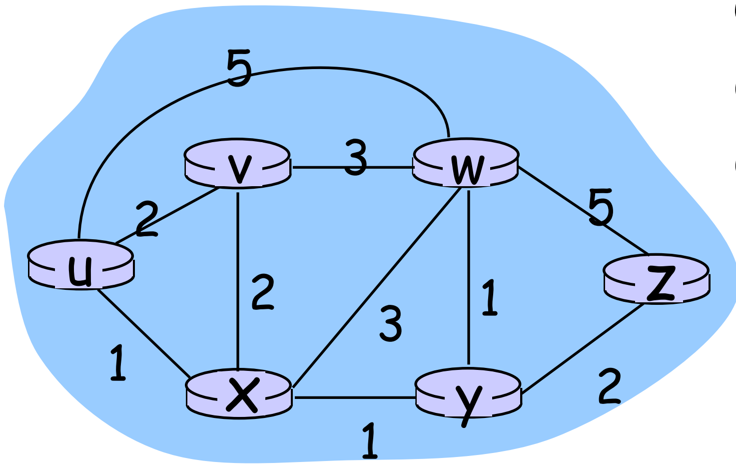
$d_x(y)$:= cost of least-cost path from x to y

Then

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

where min is taken over all neighbors of x

Bellman-Ford example (2)



Clearly, $d_v(z)/(v\text{-}x\text{-}y\text{-}z)=5$,
$d_x(z)/(x\text{-}y\text{-}z) = 3$,
$d_w(z)/(w\text{-}y\text{-}z) = 3$

B-F equation says:

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

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

Distance Vector Algorithm (3)

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

Distance vector algorithm (4)

Basic idea:

- Each node periodically sends its own distance vector estimate to neighbors
- When a node x receives new DV estimate from neighbors, it updates its own DV using B-F equation:

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

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

Distance Vector Algorithm (5)

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, or cost of msg from neighbor)

↓

recompute estimates

↓

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

Distance Vector Routing Algorithm

iterative:

- continues until no nodes exchange info.
- *self-terminating*: no "signal" to stop

asynchronous:

- nodes need *not* exchange info

distributed:

- each node communicates *only* with directly-attached neighbors

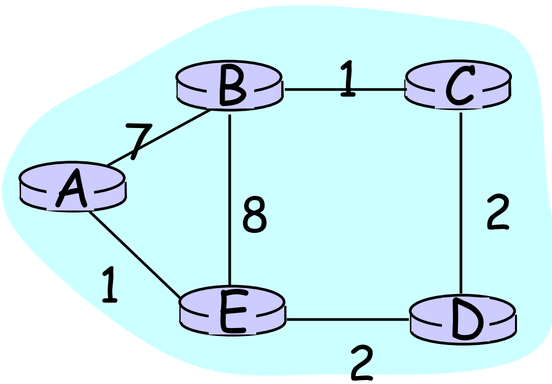
Distance Table data structure

- each node has its own row for each possible destination, and column for each directly-attached neighbor to node
- example: in node X, for dest. Y via neighbor Z:

$$D^X(Y,Z) = \text{distance } \textit{from} \text{ X } \textit{to} \text{ Y, } \textit{via} \text{ Z as next hop}$$

$$= c(X,Z) + \min_w \{D^Z(Y,w)\}$$

Distance Table example



Distance Table in E:
Finding routes from E to A,B,C,D

$D^E()$	cost to destination via A	B	D
destination A	1	14	5
B	7	8	5
C	6	9	4
D	4	11	2

$$D^E(C,D) = c(E,D) + \min_w \{D^D(C,w)\} = 2+2 = 4$$

$$D^E(A,D) = c(E,D) + \min_w \{D^D(A,w)\} = 2+3 = 5$$

loop!

$$D^E(A,B) = c(E,B) + \min_w \{D^B(A,w)\} = 8+6 = 14$$

loop!

$D^X(Y,Z)$ = distance *from* X *to* Y, *via* Z as next hop

$= c(X,Z) + \min_w \{D^Z(Y,w)\}$

Distance table gives routing table

cost to destination via

$D^E()$	A	B	D
A	1	14	5
B	7	8	5
C	6	9	4
D	4	11	2

destination

	Outgoing link to use, cost
A	A,1
B	D,5
C	D,4
D	D,4

destination

Distance table $\longrightarrow$ Routing table

Distance Vector Algorithm

At all nodes, X:

Initialization:
  for all adjacent nodes v:
    $D^X(*,v)$ = infty   /* the * operator means "for all rows" */
    $D^X(v,v) = c(X,v)$
  for all destinations, y
    send $\min_w D^X(y,w)$ to each neighbor /* w over all X's neighbors */

Distance Vector Algorithm (cont)

loop
  wait (until I see a link cost change to neighbor V
      or until I receive update from neighbor V)

  if (c(X,V) changes by d)
    /* change cost to all dest's via neighbor v by d */
    /* note: d could be positive or negative */
    for all destinations y: $D^X(y,V) = D^X(y,V) + d$

  else if (update received from V wrt destination Y)
    /* shortest path from V to some Y has changed */
    /* V has sent a new value for its $\min_w\{D^V(Y,w)\}$ */
    /* call this received new value is "newval" */
    for the single destination y: $D^X(Y,V) = c(X,V) + newval$

  if we have a new $\min_w\{D^X(Y,w)\}$ for any destination Y
    send new value of $\min_w\{D^X(Y,w)\}$ to all neighbors

forever

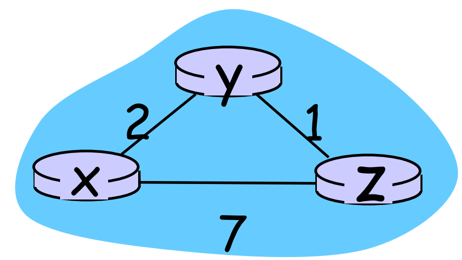
$D_x(y) = \min\{c(x,y) + D_y(y),\ c(x,z) + D_z(y)\}$
$= \min\{2+0\ ,\ 7+1\} = 2$

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

Line 23-24: only sending new value of $\min_w D^X(Y,w)$ to all neighbors

Only D^X, D^Z changed

D^Y does not changed



node x table

Initially:

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

Next:

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

NEW

Next:

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

node y table

Initially:

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

Next:

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

OLD

Next:

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

node z table

Initially:

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

Next:

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

NEW

Next:

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

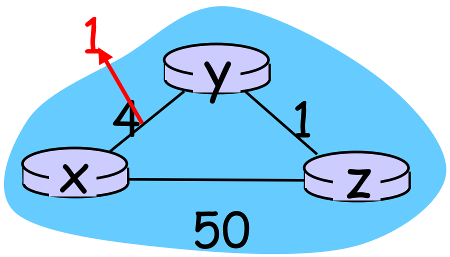
time

Initially Next Next

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



"good news travels fast"

At time t_0, y detects the link-cost change (4 to 1), updates its DV, and informs its neighbors (x, z).

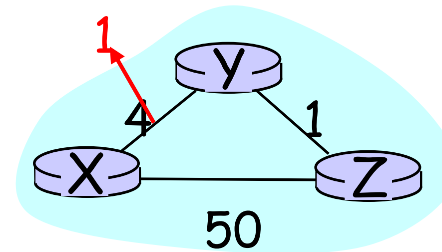
At time t_1, z receives the update from y and updates its table. It computes a new least cost to x (2 instead of 5) and sends its neighbors (x, y) its DV.

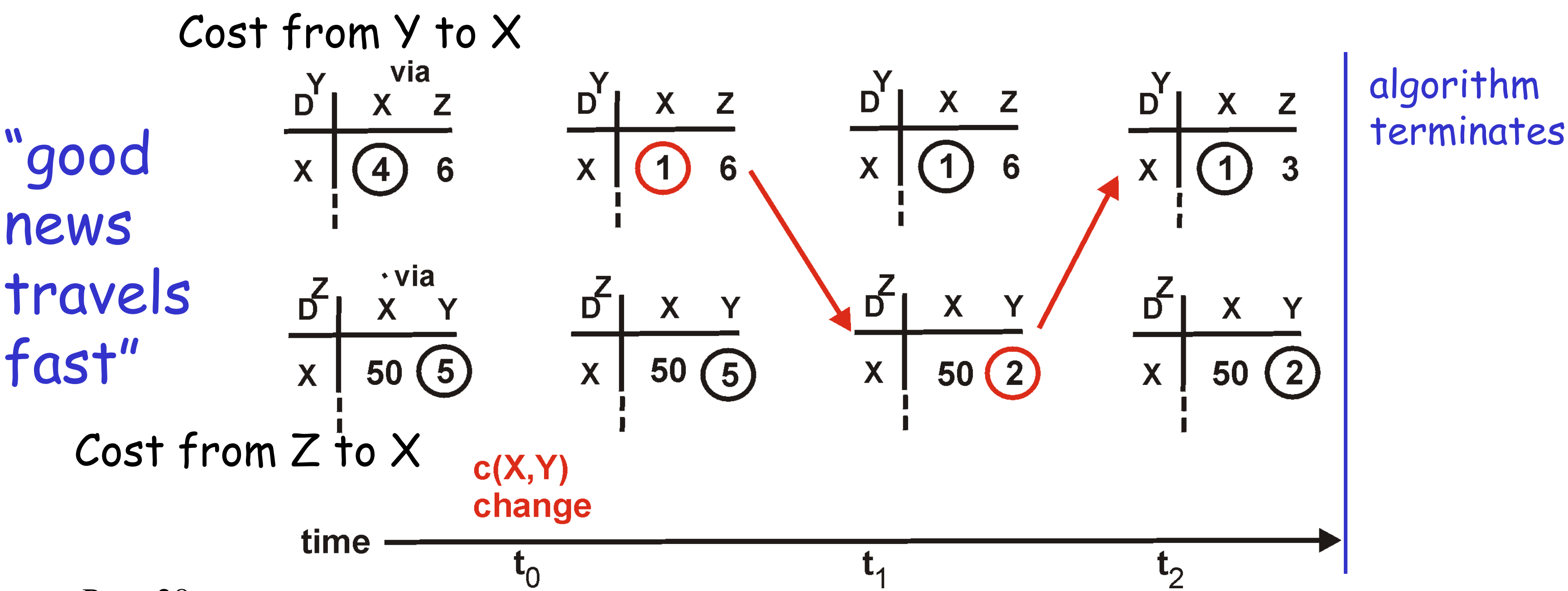
At time t_2, y receives z's update and updates its distance table. y's least costs do not change and hence y does *not* send any message to z.

Distance Vector link cost changes

Link cost changes:

- node detects local link cost change
- updates distance table (line 15)
- if cost change in least cost path, notify neighbors (lines 23,24)

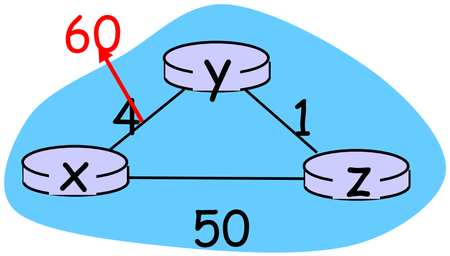




Distance Vector: link cost changes

Link cost changes:

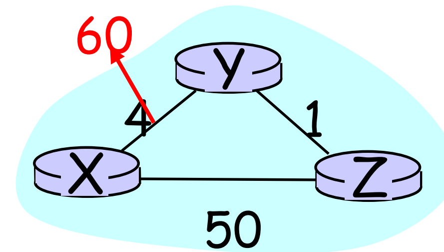
- good news travels fast
- bad news travels slow - "count to infinity" problem!
- 44 iterations before algorithm stabilizes: see next

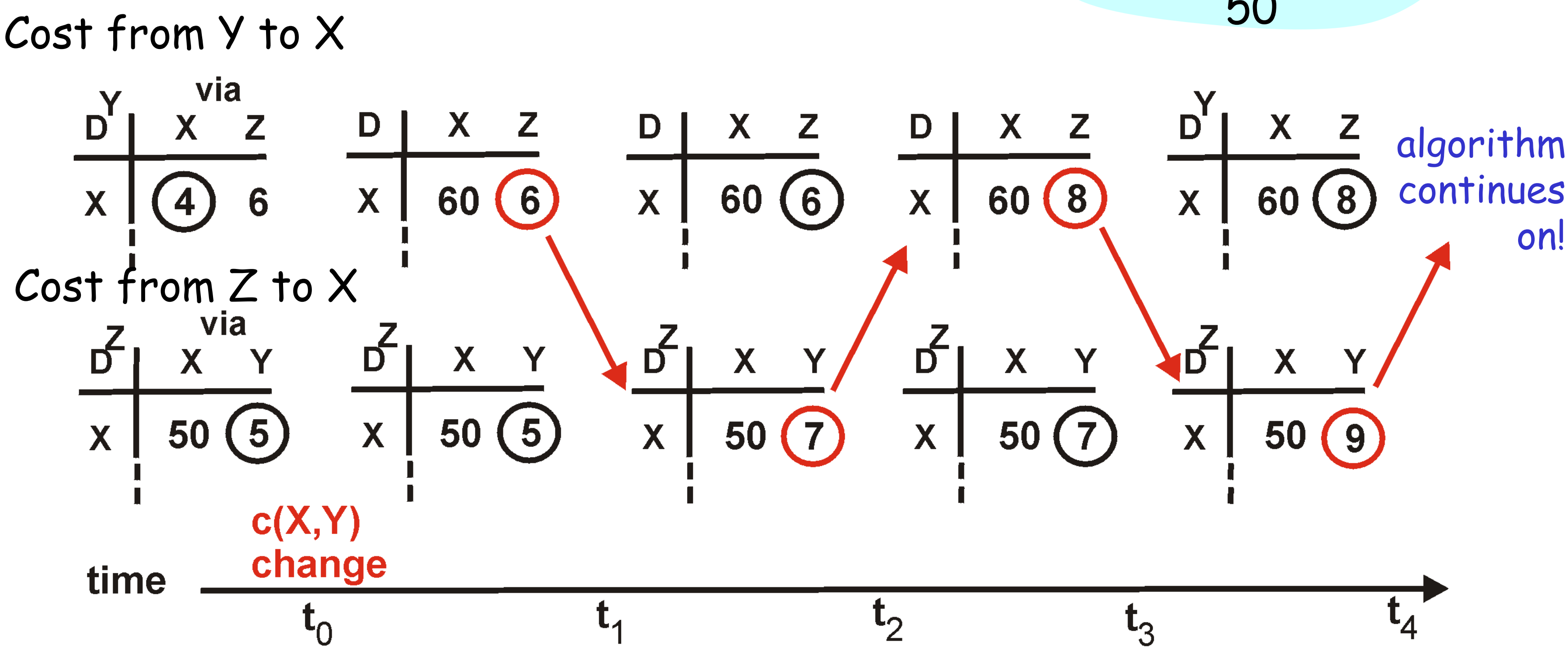


Distance Vector link cost changes

Link cost changes:

- good news travels fast
- bad news travels slow - "count to infinity" problem!

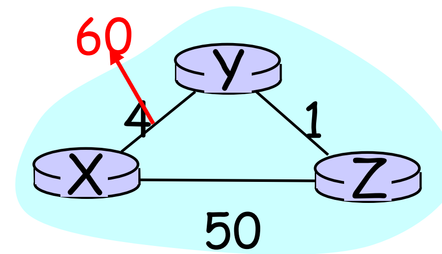


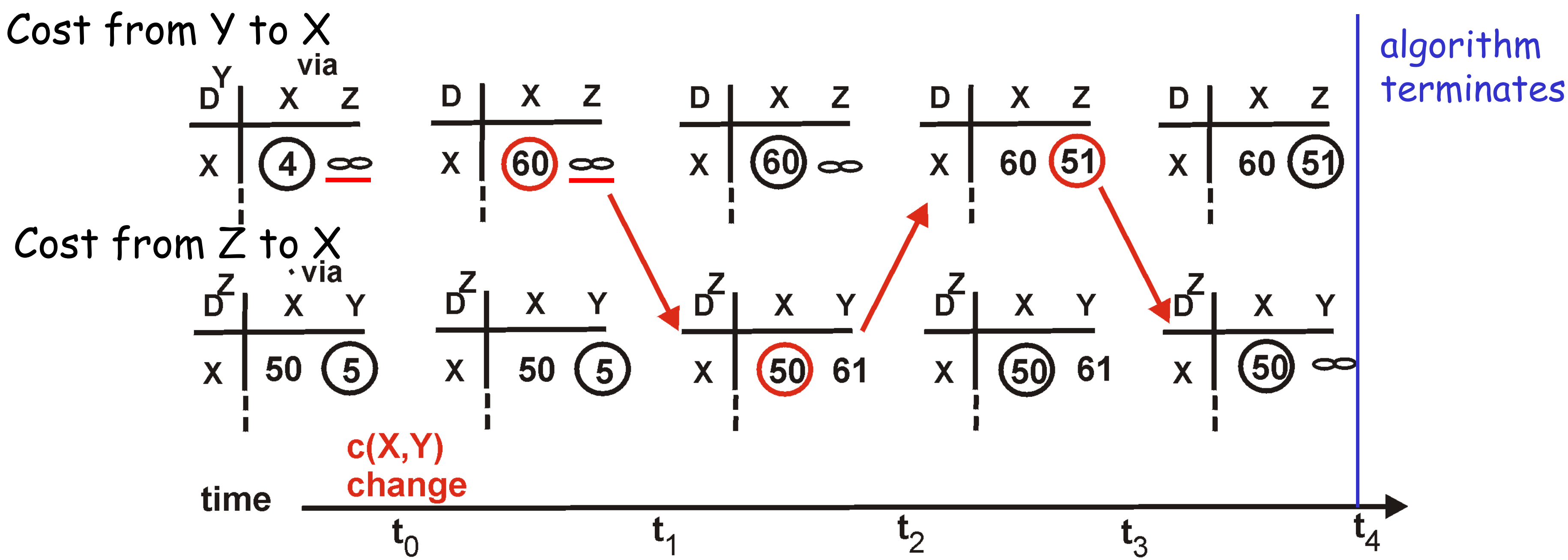


Distance Vector poisoned reverse

If Z routes through Y to get to X (Z->Y->X):

- Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z) -> reserved route is prevented, i.e., Y->Z->Y->X (loop)!





D^Y	X (via X)	Z (via Z)
t_0: X	4	∞
t_1: X	60	∞
t_2: X	60	∞
t_3: X	60	51
t_4: X	60	51

D^Z	X (via X)	Y (via Y)
t_0: X	50	5
t_1: X	50	5
t_2: X	50	61
t_3: X	50	61
t_4: X	50	∞

Comparison of LS and DV algorithms

Message complexity

- LS: with n nodes, E links, O(nE) msgs sent
- DV: exchange between neighbors only
  - convergence time varies

Speed of Convergence

- LS: $O(n^2)$ algorithm requires O(nE) msgs
  - 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 thru network

Hierarchical Routing

Our routing study thus far - idealization

- all routers identical
- network "flat"

… *not* true in practice

scale: with 200 million destinations:

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

administrative autonomy

- internet = network of networks
- each network admin may want to control routing in its own network

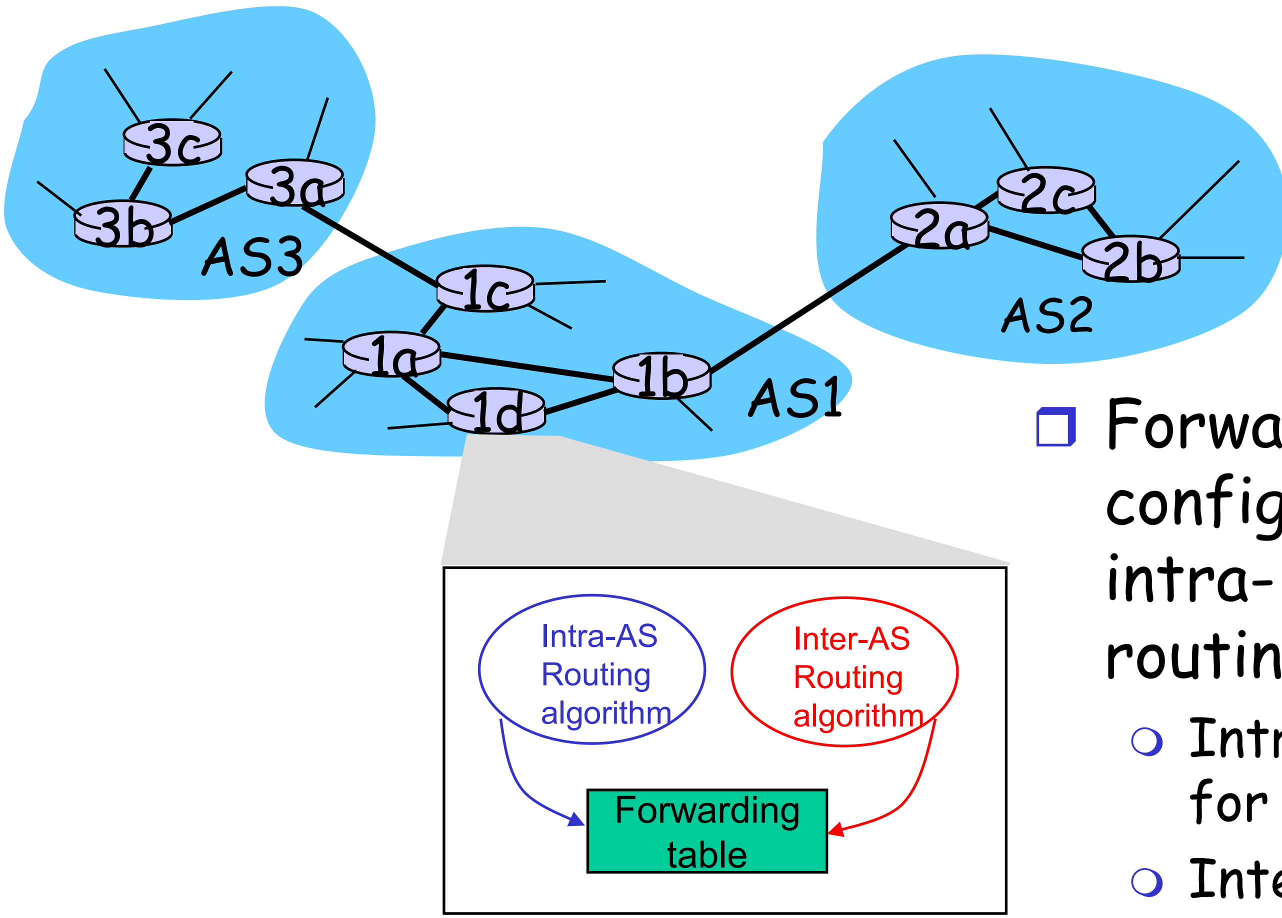
Hierarchical Routing

- 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

- Direct link to router in another AS

Interconnected ASes



- Forwarding table is configured by both intra- and inter-AS routing algorithm
  - Intra-AS sets entries for internal dests
  - Inter-AS & Intra-As sets entries for external dests

Inter-AS tasks

- Suppose router in AS1 receives datagram for which dest is outside of AS1
  - Router should forward packet towards on of the gateway routers, but which one?

AS1 needs:

1. to learn which dests are reachable through AS2 and which through AS3
2. to propagate this reachability info to all routers in AS1

Job of inter-AS routing!