Dijsktra's Algorithm

notation:
- c(x,y): link cost from node x to y; cost= ∞ if not direct neighbors
- D(n): current value of cost of path from source to "n"
- P(n): predecessor node along path from source to n
- N': set of nodes whose least cost path definitively known

```
1  Initialization:
2    N' = {s}
3    for all nodes "n"
4      If "n" adjacent to "s"
5        then D(n) = c(s,n)
6      else D(n) = ∞
7
8  Loop
9    Find "m" not in N' such that D(m) is a minimum
10   Add "m" to N'
11   update D(m) for all "n" adjacent to "m" and not in N' :
12     D(n) = min( D(n), D(m) + c(m,n) )
13   /* new cost to "n" is either old cost to "n" or known
14    shortest path cost to "m" plus cost from "m" to "n" */
15 until all nodes in N'
```


Dijkstra's algorithm: example (Step1)

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	∞

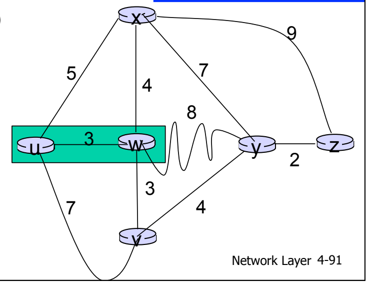
notation:
- D(n): current value of cost of path from source to "n"
- N': set of nodes whose least cost path definitively known

11 update D(m) for all "n" adjacent to "m" and not in N' :

12 **D(n) = min(D(n), D(m) + c(m,n))**

N'(uw) → update D(v), D(x), D(y), D(z)

D(v)=min(D(v), D(w)+c(w,v))
=min(7,3+3)=6
D(x)=min(D(x), D(w)+c(w,x))
=min(5,3+4)=5
D(y)=min(D(y), D(w)+c(w,y))
=min(∞, 3+8)=11
D(z)=min(D(z), D(w)+c(w,z))
=min(∞, 3+∞)=∞



Dijkstra's algorithm: example (Step 2)

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

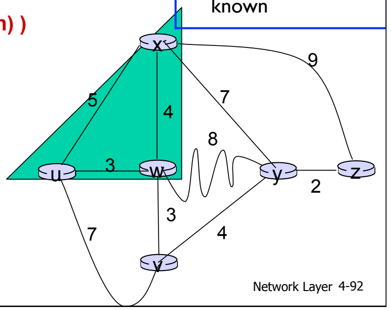
notation:
- D(n): current value of cost of path from source to "n"
- N': set of nodes whose least cost path definitively known

11 update D(m) for all "n" adjacent to "m" and not in N' :

12 **D(n) = min(D(n), D(m) + c(m,n))**

N'(uwx) → update D(v), D(y), D(z)

D(v)=min(D(v), D(x)+c(x,v))
=min(6,5+∞)=6
D(y)=min(D(y),D(x)+c(x,y))
=min(11,5+7)=11
D(z)=min(D(z), D(x)+c(x,z))
=min(∞,5+9)=14



Dijkstra's algorithm: example (Step 3)

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

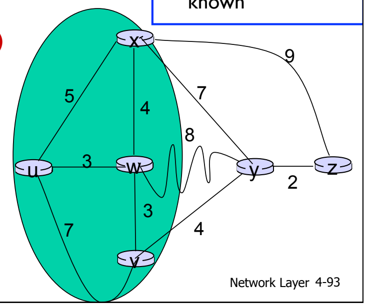
notation:
- D(n): current value of cost of path from source to "n"
- N': set of nodes whose least cost path definitively known

11 update D(m) for all "n" adjacent to "m" and not in N' :

12 **D(n) = min(D(n), D(m) + c(m,n))**

N'(uwxv) → update D(y), D(z)

D(y)=min(D(y),D(v)+c(v,y))
=min(11, 6+4)=10
D(z)=min(D(z), D(v)+c(v,z))
=min(14, 6+∞)=14



Dijkstra's algorithm: example (Step 4)

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

notation:

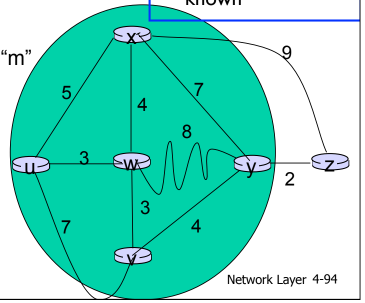
- D(n): current value of cost of path from source to "n"
- N': set of nodes whose least cost path definitively known

11 update D(m) for all "n" adjacent to "m" and not in N' :

12 **D(n) = min(D(n), D(m) + c(m,n))**

N'(uwxvy) → update D(z)

D(z)=min(D(z), D(y)+c(y,z))
=min(14, 10+2)=12



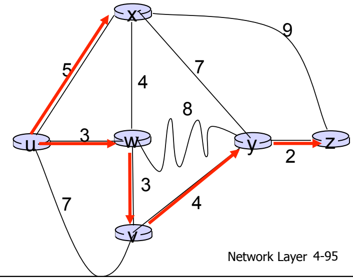
Dijkstra's algorithm: example (Step 5)

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:

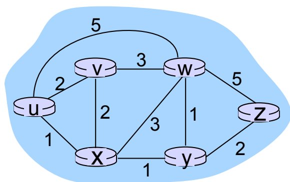
- construct shortest path tree by tracing predecessor nodes
- ties can exist (can be broken arbitrarily)

destination	link
x	(u,x)
w	(u,w)
v	(u,w)
y	(u,w)
z	(u,w)



Dijkstra's algorithm: another example

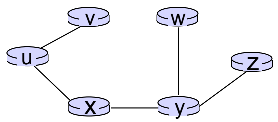
Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



destination	link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

Dijkstra's algorithm: example (2)

resulting shortest-path tree from u:



resulting forwarding table in u:

destination	link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

Chapter 4: outline

Distance vector algorithm

Bellman-Ford equation (dynamic programming)

let

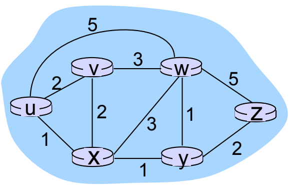
$d_x(y)$:= least-cost path from x to y

then

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

- $d_v(y)$: cost from neighbor v to destination y
- $c(x,v)$: 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 says:

$$d_u(z) = \min \{ c(u,v) + d_v(z), c(u,x) + d_x(z), c(u,w) + d_w(z) \}$$

$$= \min \{2 + 5, 1 + 3, 5 + 3\} = 4$$

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:

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

- 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)

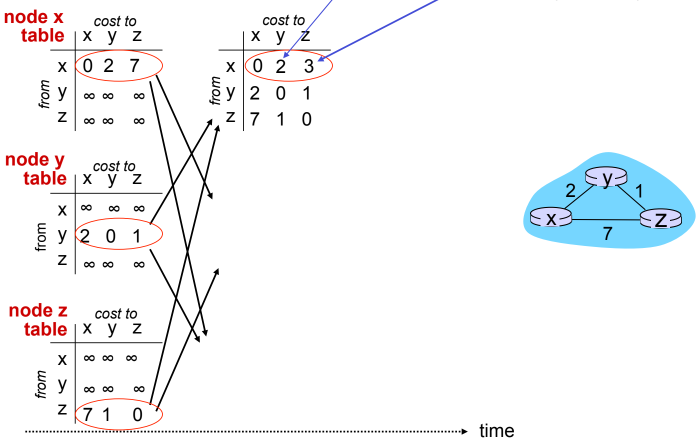
recompute estimates

$D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\}$

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

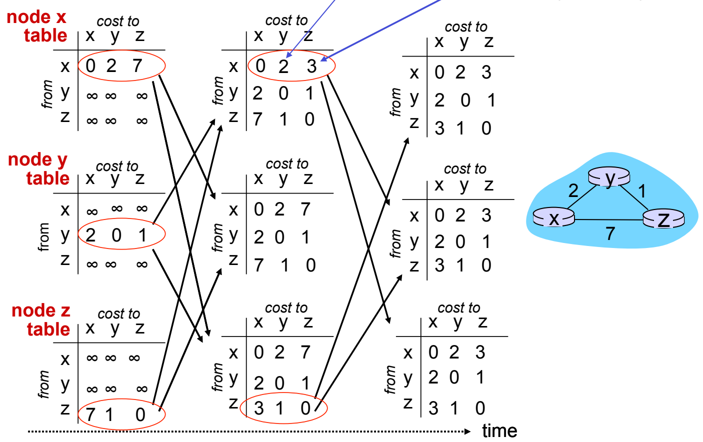
$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$
$= \min\{2+0 , 7+1\} = 2$

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



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

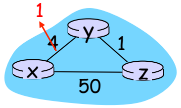
$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$
$= \min\{2+1 , 7+0\} = 3$

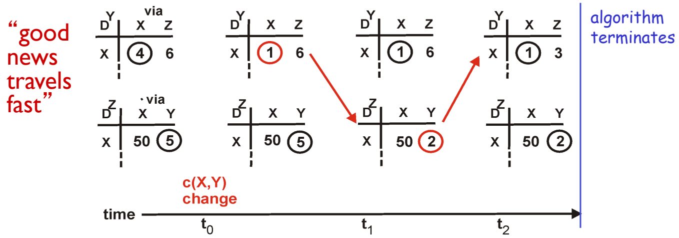


Distance vector: link cost changes (1)

link cost changes:

- node detects local link cost change
- updates routing info, recalculates distance vector
- if DV changes, notify neighbors

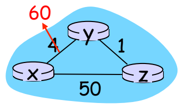


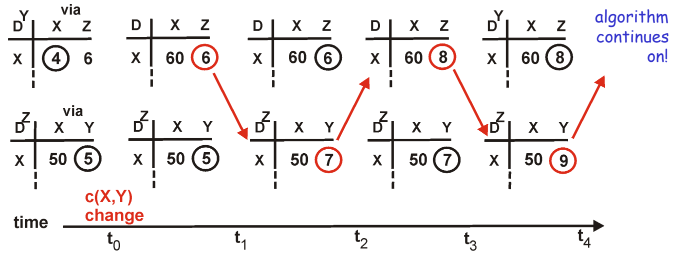


Distance vector: link cost changes (2)

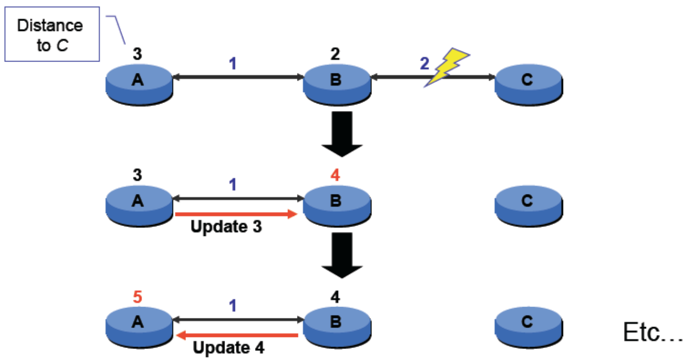
link cost changes:

- node detects local link cost change
- *bad news travels slow* - "count to infinity" problem!
- 44 iterations before algorithm stabilizes





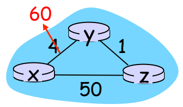
Distance vector: count to infinity

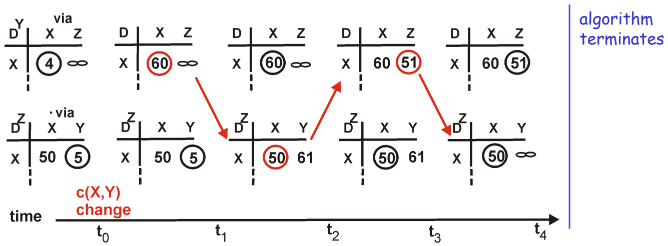


Distance vector: Poison Reverse

If Z routes through Y to get to X :

- Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)
- Still, can have problems when more than 2 routers are involved





Comparison of LS and DV algorithms

	Link State	Distance Vector
size of (update) routing info	small, contains only neighbours' link costs ☺	potentially long distance vectors
communication overhead	flood to all nodes – overhead O(N*E), where N = # of nodes, E = # of edges	send distance vectors only to neighbours – O(N*K) if each of N routers has K neighbours ☺
convergence speed	do NOT need to recalculate LSP's before forwarding ⇒ faster ☺	takes a while to propagate changes to rest of network
space requirements	maintains entire topology in a link database – O(N*K) if each of N routers has K neighbours	maintains only neighbours' states – O(K) distance vectors ☺
computational complexity per one destination	O(N*(N-1)/2)=O(N²)	O(N*K*Diameter) ☺
computational robustness	each router computes paths on its own – no error propagation ☺	routers compute paths collectively – errors propagate
security / fault tolerance	false/corrupt LSPs can be flooded to all routers	false/corrupt LSPs can be flooded to all routers

Chapter 4: outline

Hierarchical routing

our routing study thus far - idealization
- all routers identical
- network "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!

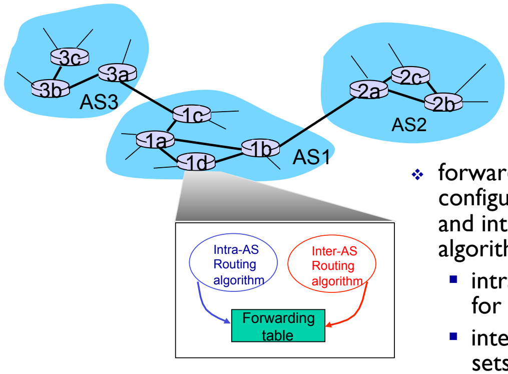
administrative autonomy
- internet = network of networks
- each network admin may want to control routing in its own network

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 dests
  - inter-AS & intra-AS sets entries for external dests

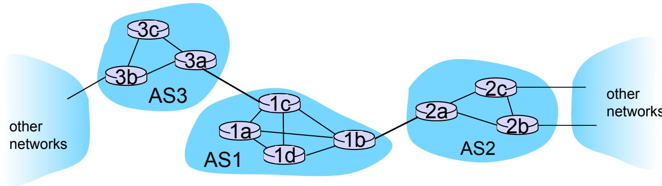
Inter-AS tasks

- suppose router in AS1 receives datagram destined outside of AS1:
  - router should forward packet to gateway router, but which one?

AS1 must:

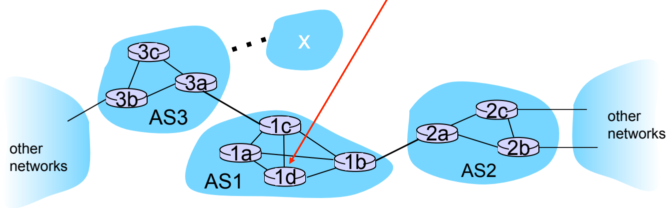
1. learn which dests are reachable through AS2, which through AS3
2. propagate this reachability info to all routers in AS1

job of inter-AS routing!



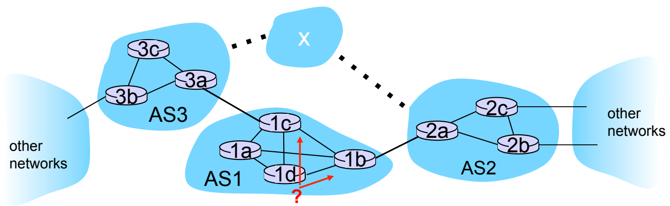
Example: setting forwarding table in router 1d

- suppose AS1 learns (via inter-AS protocol) that subnet *x* 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)*



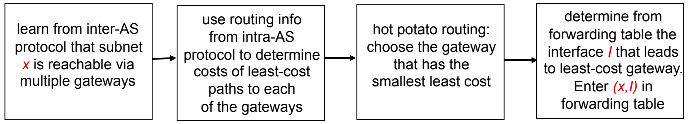
Example: choosing among multiple ASes

- 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 towards which gateway it should forward packets for dest x
  - this is also job of inter-AS routing protocol!
- *hot potato routing: send* packet towards closest of two routers.



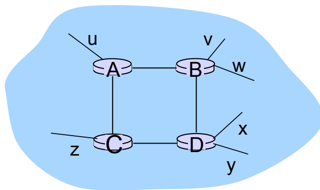
Chapter 4: outline

Intra-AS Routing

- also known as *interior gateway protocols (IGP)*
- most common intra-AS routing protocols:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - IGRP: Interior Gateway Routing Protocol (Cisco proprietary)

RIP (Routing Information Protocol)

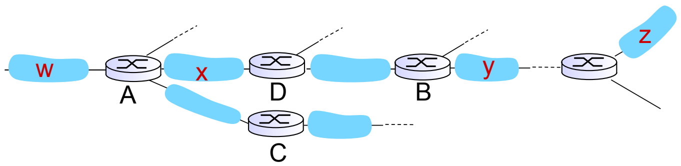
- distance vector algorithm
  - distance metric: # hops (max = 15 hops), each link has cost 1
  - DVs exchanged with neighbors every 30 sec in response message (aka advertisement)
  - each advertisement: list of up to 25 destination *subnets* *(in IP addressing sense)*



from router A to destination *subnets:*

subnet	hops
u	1
v	2
w	2
x	3
y	3
z	2

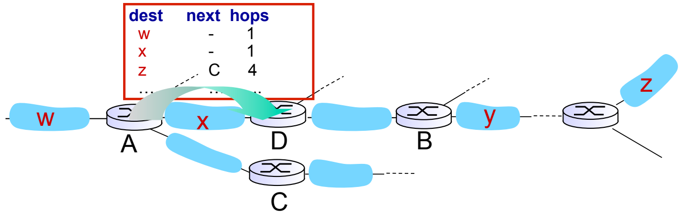
RIP: example



routing table in router D

destination subnet	next router	# hops to dest
w	A	2
y	B	2
z	B	7
x	--	1
....	....	....

RIP: example



routing table in router D

destination subnet	next router	# hops to dest
w	A	2
y	B	2
z	~~B~~ A	~~7~~ 5
x	--	1
....	....	....

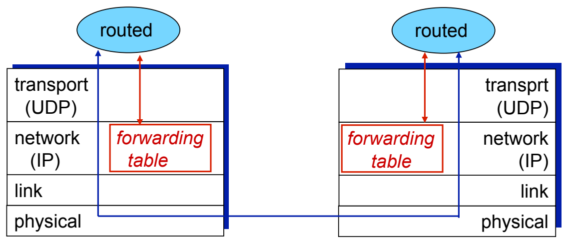
RIP: link failure, recovery

if no advertisement heard after 180 sec --> neighbor/ link declared dead

- routes via neighbor invalidated
- new advertisements sent to neighbors
- neighbors in turn send out new advertisements (if tables changed)
- link failure info quickly (?) propagates to entire net
- *poison reverse* used to prevent ping-pong loops (infinite distance = 16 hops)

RIP table processing

- RIP routing tables managed by *application-level* process called route-d (daemon)
- advertisements sent in UDP packets, periodically repeated



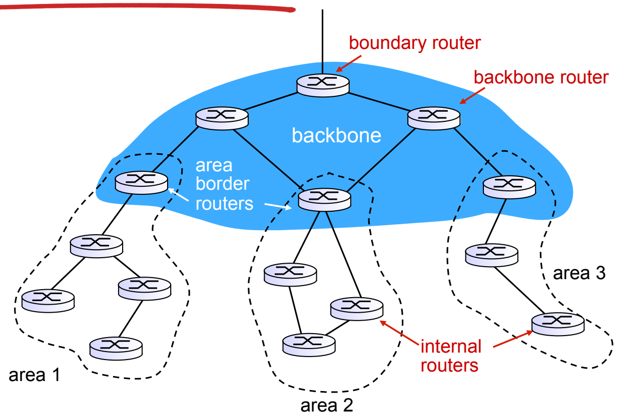
OSPF (Open Shortest Path First)

- "open": publicly available
- uses link state algorithm
  - LS packet dissemination
  - topology map at each node
  - route computation using Dijkstra's algorithm
- OSPF advertisement carries one entry per neighbor
- advertisements flooded to *entire* AS
  - carried in OSPF messages directly over IP (rather than TCP or UDP
- *IS-IS routing* protocol: nearly identical to OSPF

OSPF "advanced" features (not in RIP)

- *security:* all OSPF messages authenticated (to prevent malicious intrusion)
- multiple same-cost paths allowed (only one path in RIP)
- for each link, multiple cost metrics for different TOS (e.g., satellite link cost set "low" for best effort ToS; high for real time ToS)
- integrated uni- and multicast support:
  - Multicast OSPF (MOSPF) uses same topology data base as OSPF
- hierarchical OSPF in large domains.

Hierarchical OSPF



Hierarchical OSPF

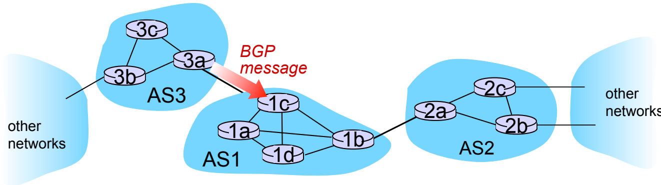
- *two-level hierarchy:* local area, backbone.
  - link-state advertisements only in area
  - each nodes has detailed area topology; only know direction (shortest path) to nets in other areas.
- *area border routers:* "summarize" distances to nets in own area, advertise to other Area Border routers.
- *backbone routers:* run OSPF routing limited to backbone.
- *boundary routers:* connect to other AS's.

Internet inter-AS routing: BGP

- BGP (Border Gateway Protocol): *the* de facto inter-domain routing protocol
  - "glue that holds the Internet together"
- BGP provides each AS a means to:
  - eBGP: obtain subnet reachability information from neighboring ASs.
  - iBGP: propagate reachability information to all AS-internal routers.
  - determine "good" routes to other networks based on reachability information and policy.
- allows subnet to advertise its existence to rest of Internet: *"I am here"*

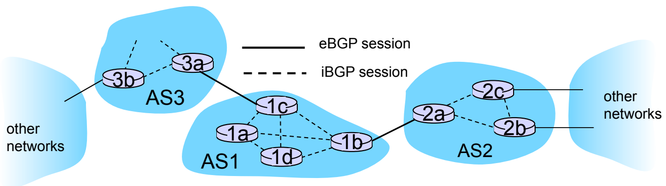
BGP basics

- BGP session: two BGP routers ("peers") exchange BGP messages:
  - advertising *paths* to different destination network prefixes ("path vector" protocol)
  - exchanged over semi-permanent TCP connections
- when AS3 advertises a prefix to AS1:
  - AS3 *promises* it will forward datagrams towards that prefix
  - AS3 can aggregate prefixes in its advertisement



BGP basics: distributing path information

- using eBGP session between 3a and 1c, AS3 sends prefix reachability info to AS1.
  - 1c can then use iBGP do distribute new prefix info to all routers in AS1
  - 1b can then re-advertise new reachability info to AS2 over 1b-to-2a eBGP session
- when router learns of new prefix, it creates entry for prefix in its forwarding table.



Path attributes and BGP routes

- advertised prefix includes BGP attributes
  - prefix + attributes = "route"
- two important attributes:
  - AS-PATH: contains ASs through which prefix advertisement has passed: e.g., AS 67, AS 17
  - NEXT-HOP: indicates specific internal-AS router to next-hop AS. (may be multiple links from current AS to next-hop-AS)
- gateway router receiving route advertisement uses import policy to accept/decline
  - e.g., never route through AS x
  - *policy-based* routing

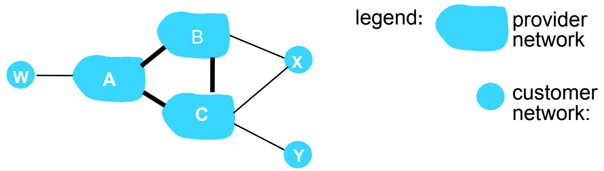
BGP route selection

- router may learn about more than 1 route to destination AS, selects route based on:
  1. local preference value attribute: policy decision
  2. shortest AS-PATH
  3. closest NEXT-HOP router: hot potato routing
  4. additional criteria

BGP messages

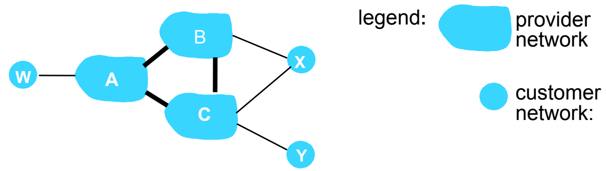
- BGP messages exchanged between peers over TCP connection
- BGP messages:
  - OPEN: opens TCP connection to peer and authenticates sender
  - UPDATE: advertises new path (or withdraws old)
  - KEEPALIVE: keeps connection alive in absence of UPDATES; also ACKs OPEN request
  - NOTIFICATION: reports errors in previous msg; also used to close connection

BGP routing policy



- A,B,C are *provider networks*
- X,W,Y are customer (of provider networks)
- X is *dual-homed:* attached to two networks
  - X does not want to route from B via X to C
  - .. so X will not advertise to B a route to C

BGP routing policy (2)



- A advertises path AW to B
- B advertises path BAW to X
- Should B advertise path BAW to C?
  - No way! B gets no "revenue" for routing CBAW since neither W nor C are B's customers
  - B wants to force C to route to w via A
  - B wants to route *only* to/from its customers!

Why different Intra-, Inter-AS routing ?

policy:

- inter-AS: admin wants control over how its traffic routed, who routes through its net.
- intra-AS: single admin, so no policy decisions needed

scale:

- hierarchical routing saves table size, reduced update traffic

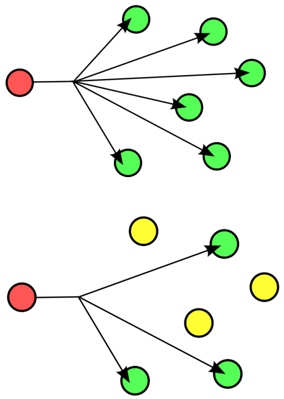
performance:

- intra-AS: can focus on performance
- inter-AS: policy may dominate over performance

Chapter 4: outline

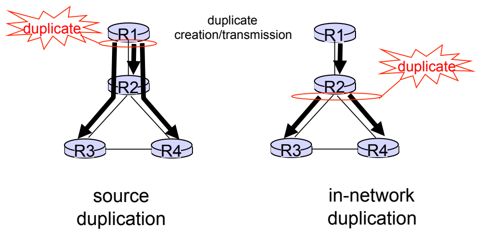
Broadcast and multicast routing

- One to many communications
- Broadcast – delivering a packet sent from a source node to all other nodes in the network
- Multicast – a single source node to send a copy of a packet to a subset of the other network nodes.



Broadcast

- deliver packets from source to all other nodes
- source duplication is inefficient:



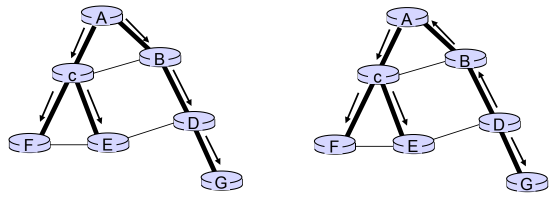
- source duplication: how does source determine recipient addresses?

In-network duplication

- *flooding:* when node receives broadcast packet, sends copy to all neighbors
  - problems: cycles & broadcast storm
- *controlled flooding:* node only broadcasts pkt if it hasn't broadcast same packet before
  - node keeps track of packet ids already broadcasted
  - or reverse path forwarding (RPF): only forward packet if it arrived on shortest path between node and source
- *spanning tree:*
  - no redundant packets received by any node

Spanning tree

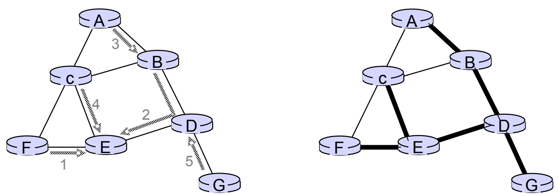
- first construct a spanning tree
- nodes then forward/make copies only along spanning tree



(a) broadcast initiated at A (b) broadcast initiated at D

Spanning tree: creation

- center node
- each node sends unicast join message to center node
  - message forwarded until it arrives at a node already belonging to spanning tree



(a) stepwise construction of spanning tree (center: E) (b) constructed spanning tree

Multicast

- *goal:* sending a packet to a group of members
- Benefits of multicast
  - better bandwidth utilization
  - less host/router processing
  - quicker participation
- Applications
  - Video/Audio broadcast (One sender)
  - Video conferencing (Many senders)
  - Real time news distribution
  - Interactive gaming

Internet multicast

- Senders transmit IP datagrams to a "host group"
- "Host group" identified by a class D IP address
- Members of host group could be present anywhere in the Internet
- Members join and leave the group and indicate this to the routers
- Routers listen to all multicast addresses and use multicast routing protocols to manage groups
- Routing protocols:
  - DVMRP: distance vector multicast routing protocol, RFC1075
  - PIM: protocol independent multicast

Chapter 4: *done!*

4.1 introduction
4.2 virtual circuit and datagram networks
4.3 what's inside a router
4.4 IP: Internet Protocol
- datagram format, IPv4 addressing, ICMP, IPv6

4.5 routing algorithms
- link state, distance vector, hierarchical routing

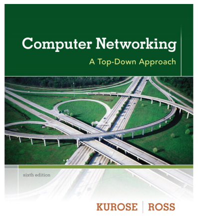
4.6 routing in the Internet
- RIP, OSPF, BGP

4.7 broadcast and multicast routing

- understand principles behind network layer services:
  - network layer service models, forwarding versus routing how a router works, routing (path selection), broadcast, multicast
- instantiation, implementation in the Internet

A note on these slides

Part of PPT slides were adopted from Prof. Natalija Vlajic' early CSE3214 course and the rest were adopted from the book "Computer Networking: A Top Down Approach" 6th Edition by Jim Kurose and Keith Ross



Computer Networking: A Top Down Approach
6th edition
Jim Kurose, Keith Ross
Addison-Wesley
March 2012