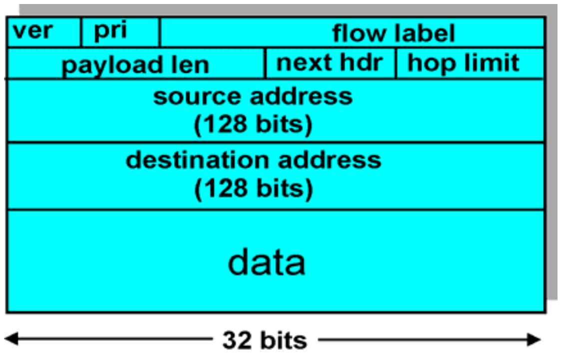
IPv6

- Initial motivation: 32-bit address space soon to be completely allocated.
- Additional motivation:
  - Header format helps speed processing/forwarding
  - Header changes to facilitate QoS (service classes)
  - Reduction of routing table size
  - Multicast support
  - Support for mobile hosts
  - Support coexistence with other protos, e.g., IPv4

  IPv6 datagram format:
  - Fixed-length 40 byte 7 element header
  - No fragmentation allowed

IPv6 Header

Version: IPv4 or IPv6
Priority: Identify priority among datagrams in flow
Flow Label: Identify datagrams in same “flow.”
(concept of“flow” not well defined).
Next header: Identify upper layer protocol for data

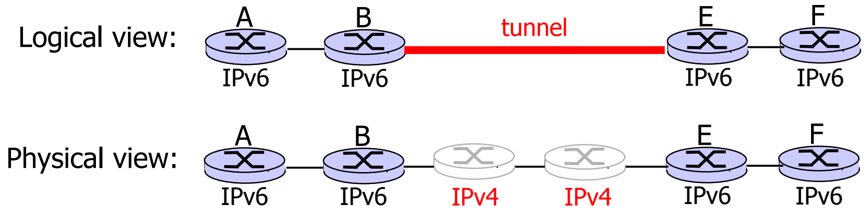


Other Changes From IPv4

- *Checksum*: Removed entirely to reduce processing time at each hop
- *Options:* Allowed, but outside of header, indicated by “Next Header” field
- *ICMPv6:* New version of ICMP
  - Additional message types, e.g. “Packet Too Big”
  - Multicast group management functions
- *Extension header examples:*
  - Routing, fragmentation, authentication, encrypted security payload, destination options

Transition From IPv4 To IPv6

- Not all routers can be upgraded simultaneous
  - No “flag days”
  - How will the network operate with mixed IPv4 and IPv6 routers?
- *Tunneling:* IPv6 carried as payload in IPv4 datagram among IPv4 routers



Network layer: Status

- Network layer functions
- IP
- Routing and forwarding
- NAT
- ARP
- IPv6
- Routing

Internet routing

Our routing study thus far – idealization

- All routers identical
- Network "flat"

… *not* true in practice

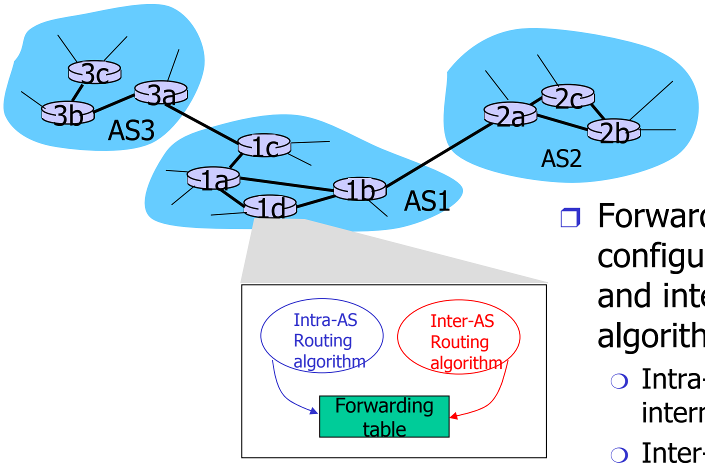
Scale: With 200 million destinations:

- 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
- Aggregate routers into regions, "autonomous systems" (AS)
- Routers in same AS run same routing protocol
  - "Inter-AS" routing protocol
  - Routers in different AS can run different inter-AS routing protocol

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

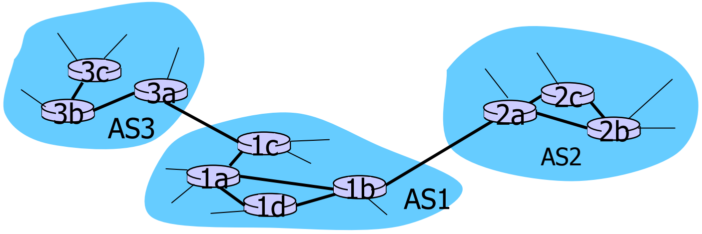
Inter-AS Tasks

- Suppose router in AS1 receives datagram for dest outside of AS1
  - Router should forward packet towards an AS-border router, 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!



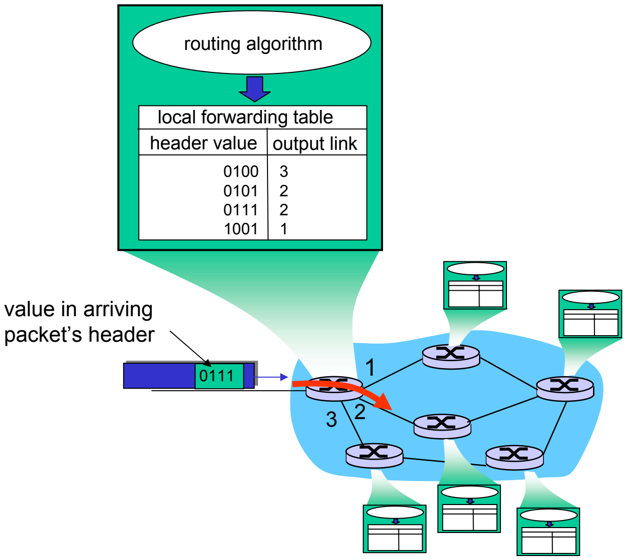
Intra-AS routing

- ❒ Also known as Interior Gateway Protocols (IGP)
- ❒ Most common Intra-AS routing protocols:
  - ❍ RIP: Routing Information Protocol
    - Distance vector protocol (based on Bellman-Ford)
    - Routers periodically exchange reachability info with their neighbors
    - Distance metric: hop count
    - Advantage: simple, minimal communication overhead
    - Disadvantage: long convergence times, loop detection

Intra-AS routing protocols

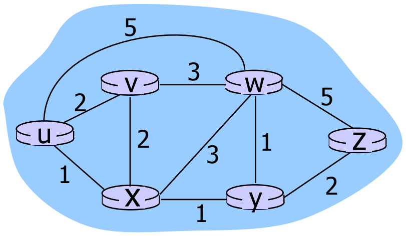
- ❍ OSPF: Open Shortest Path First
  - Link state protocol (based on Dijkstra)
  - Routers periodically **flood** immediate reachability information to all other routers
  - Distance metric: administrative weight
  - Advantage: fast convergence
  - Disadvantage: complexity and communication overhead
- ❍ ISIS: Intermediate-System-to-Intermediate-System (ISO 10589) (link state)
- ❍ IGRP: Interior Gateway Routing Protocol (Cisco proprietary) (distance vector)
- ❍ EIGRP: Enhanced Interior Gateway Routing Protocol (Cisco proprietary) (enhanced distance vector)

Interplay between routing and forwarding



header value	output link
0100	3
0101	2
0111	2
1001	1

Graph abstraction



Graph: $G = (N,E)$

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

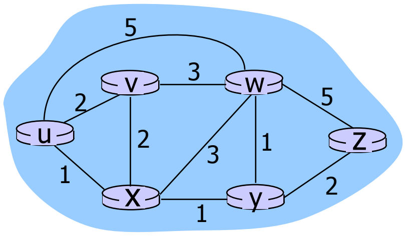
E = set of links $= \{ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) \}$

Path: Sequence of edges (routers)

Remark: Graph abstr. 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 can be always 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: alg. that finds "good" path
(typically: 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 routing table for that node
- Iterative: after k iterations, know least cost path to k dest.’s

Notation:

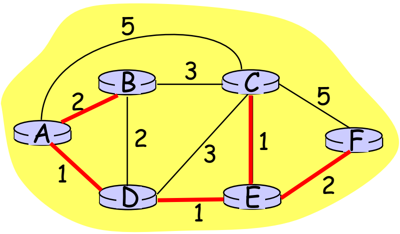
- $c(i,j)$: Link cost from node i to j. Cost infinite if not direct neighbors
- $D(v)$: Current value of cost of path from source to dest. v
- $p(v)$: Predecessor node along path from source to v
- $N’$: Set of nodes whose least cost path definitively known

Dijsktra’s algorithm

```
Initialization for A:
  N’ = {A}
  for all nodes v
    if v adjacent to A
      then D(v) = c(A,v)
      else D(v) = 1

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’
```

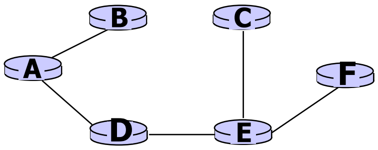
Dijkstra’s algorithm: Example

Step	start N’	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	infinity	infinity
1	AD	2,A	4,D		2,D	infinity
2	ADE	2,A	3,E			4,E
3	ADEB		3,E			4,E
4	ADEBC					4,E
5	ADEBCF					



Dijkstra’s algorithm: Example (2)

Resulting shortest-path tree from u:



Resulting forwarding table in u:

destination	link
B	(A,B)
D	(A,D)
E	(A,D)
C	(A,D)
F	(A,D)

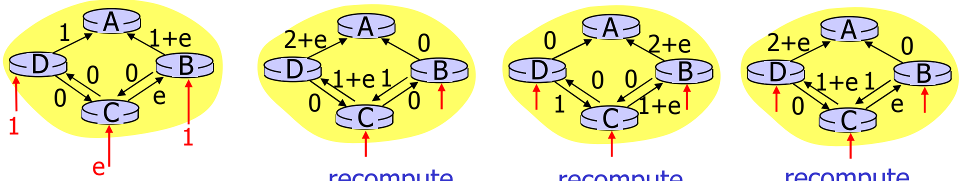
Dijkstra’s algorithm: Discussion

Algorithm complexity: n nodes

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

Oscillations possible:

- E.g., link cost = amount of carried traffic



initially

… recompute routing

… recompute

… recompute

Distance vector algorithm

Bellman-Ford Equation (dynamic programming)

Define

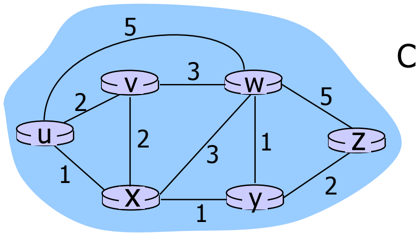
$d_x(y)$:= cost of least-cost path from x to y

Then

$$\boxed{d_x(y) = \min \{c_v(x,v) + d_v(y) \}}$$

where $\min$ is taken over all neighbors v of x

Bellman-Ford: Example



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

Bellman-Ford 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 yields minimum is next
hop in shortest path ➜ forwarding table

Distance vector algorithm (2)

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

Distance vector algorithm (3)

Basic idea:

- Each node periodically sends its own distance vector estimate to neighbors
- When a node 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 "natural" conditions the estimates of $D_x(y)$ converge to the actual least cost $d_x(y)$

Distance vector algorithm (4)

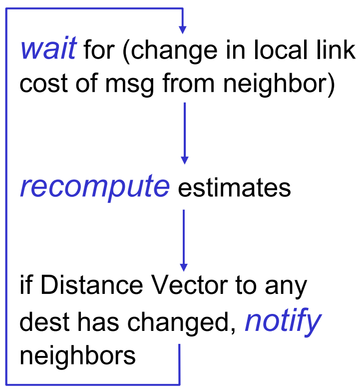
Iterative, asynchronous:

- Each local iteration caused by:
  - Local link cost change
  - DV update message from neighbor

Distributed:

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

Each node:



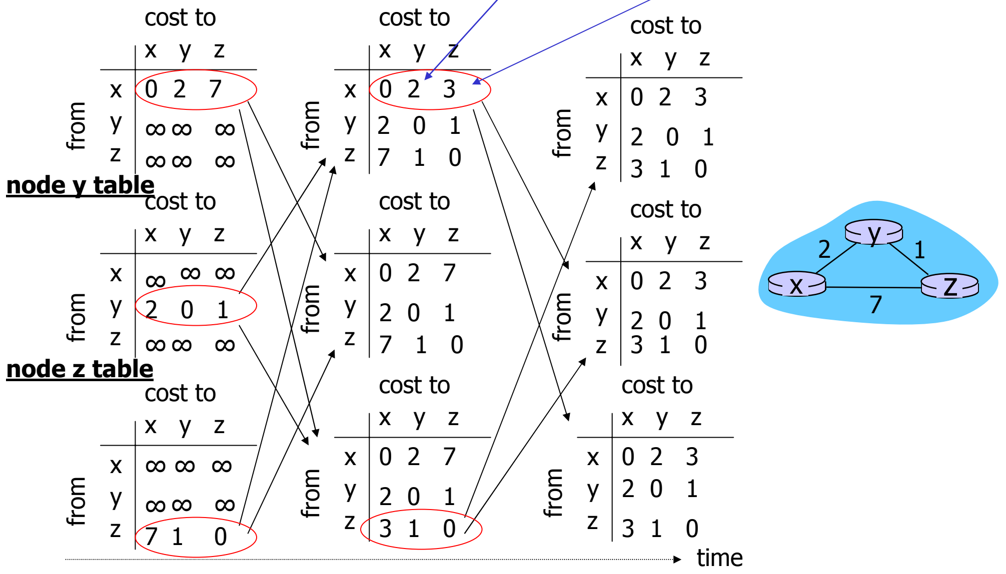
wait for (change in local link cost of msg from neighbor)

recompute estimates

if Distance Vector 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

cost to (from)

	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

cost to (from)

	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

cost to (from)

	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node y table

cost to (from)

	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

cost to (from)

	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

cost to (from)

	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node z table

cost to (from)

	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

cost to (from)

	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

cost to (from)

	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

Distance vector routing: Overview

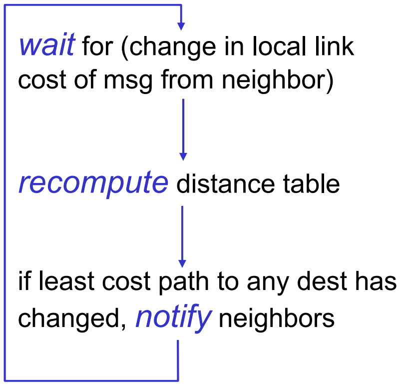
Iterative, asynchronous: each local iteration caused by:

- Local link cost change
- Message from neighbor: its least cost path change from neighbor

Distributed:

- Each node notifies neighbors *only* when its least cost path to any destination changes
  - Neighbors then notify their neighbors if necessary

Each node:



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

recompute distance table

if least cost path to any dest has changed, *notify* neighbors

Distance vector algorithm

At each node, x:

```
Initialization:
 for all destinations $y$ in N:
     $D_x(y) = 1$                     if $y$ is not a neighbor
     $D_x(y) = c(x,y)$                if $y$ is a neighbor
 for each neighbor w
     $D_w(y) = 1$ for all destinations z in N
 for each neighbor $w$
     send distance vector $D_x = [D_x(y): y \text{ in } N]$ to $w$
```

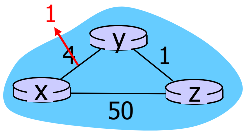
Distance vector algorithm (2.):

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

    for each $y$ in N:
            $D_x(y) = \min_v\{c(x, v) + D_v(y)\}$

    if $D_x(y)$ changed for any destination $y$
            send DV $D_x = [D_x(y)]: y \text{ in } N]$ to all neighbors

forever
```

Distance vector (DV): Link cost changes

Link cost changes:

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

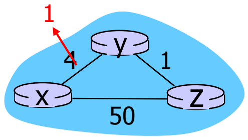


"Good news travels fast"

At time t_0, y detects the link-cost change, updates its DV, and informs its neighbors.

At time t_1, z receives the update from y and updates its table. It computes a new least cost to x and sends its neighbors 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.



node y table

cost to

from	x	y	z
x			
y	~~4~~ 1	0	1
z	5	1	0

cost to

from	x	y	z
x			
y	1	0	1
z	5	1	0

cost to

from	x	y	z
x			
y	1	0	1
z	2	1	0

node z table

cost to

from	x	y	z
x			
y	4	0	1
z	5	1	0

cost to

from	x	y	z
x			
y	1	0	1
z	~~5~~ 2	1	0

cost to

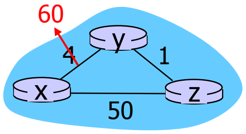
from	x	y	z
x			
y	1	0	1
z	2	1	0

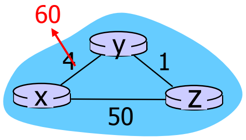
time

Distance vector: Link cost changes (2.)

Link cost changes:

- Good news travels fast
- Bad news travels slow





$D_y(x) = \min\{c(y,x) + D_x(x), c(y,z) + D_z(x)\}$
$= \min\{60 + 0 , 1 + 5\} = 6$

$D_y(x) = \min\{c(y,x) + D_x(x),$
$c(y,z) + D_z(x)\}$
$= \min\{60 + 0 , 1 + 7\} = 8$

node y table

cost to

from	x	y	z
x			
y	~~4~~ 6	0	1
z	5	1	0

cost to

from	x	y	z
x			
y	6	0	1
z	5	1	0

cost to

from	x	y	z
x			
y	~~6~~ 8	0	1
z	7	1	0

node z table

cost to

from	x	y	z
x			
y	4	0	1
z	5	1	0

cost to

from	x	y	z
x			
y	6	0	1
z	~~5~~ 7	1	0

cost to

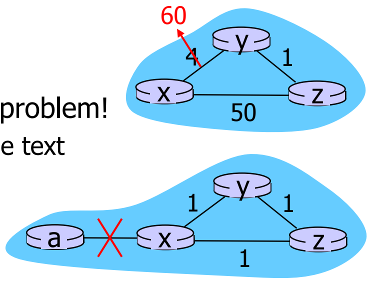
from	x	y	z
x			
y	6	0	1
z	7	1	0

time

Distance vector: Link cost changes (3.)

Link cost changes:

- Good news travels fast
- Bad news travels slow - "count to infinity" problem!
  - 44 iterations before algorithm stabilizes: see text
- What happens here?



Poissoned 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)
- Will this completely solve count to infinity problem?

Comparison of LS and DV algorithms

Message complexity

- LS: with n nodes, E links, $O(nE)$ msgs sent each
- DV: $O(d)$ messages, many times
  - d is node degree

Speed of Convergence

- LS: $O(n \log n)$ 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:

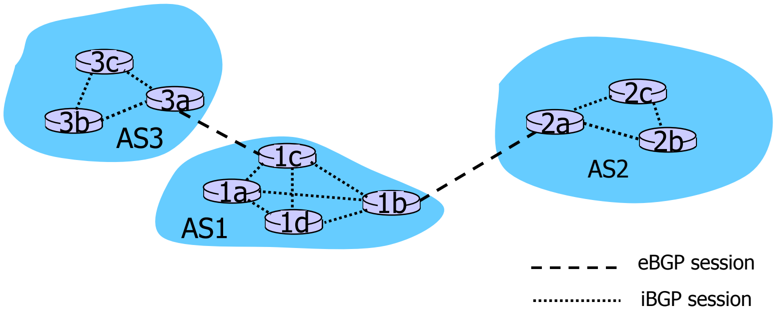
- Node can advertise incorrect *path* cost
- Each node's table used by others: error propagate through network

Internet Inter-AS routing: BGP

- *The* de facto standard: Border Gateway Protocol (BGP)
- BGP provides each AS a means to:
  1. Obtain subnet reachability information from neighboring ASs
  2. Propagate reachability information to all routers in the AS
  3. Determine "good" routes to subnets based on reachability information and routing policy.
- Allows a subnet to advertise its existence to rest of the Internet: *"I am here"*
- Issues:
  - Which routing algorithm?
  - How are routes advertised?
  - How to implement routing policies?

BGP Basics

- Pairs of routers (BGP peers) exchange routing info over semi-permanent TCP connections: BGP sessions
- Note that BGP sessions do not correspond to physical links.
- When AS2 advertises a prefix to AS1, AS2 is *promising* it will forward any datagrams destined to that prefix towards the prefix.
  - AS2 can aggregate prefixes in its advertisement



BGP is a path vector protocol

- Distance vector algorithm with extra information
  - Two important attributes:
    - AS-PATH: contains all ASs along the way: AS 67 AS 17
    - NEXT-HOP: Indicates the specific internal-AS router to next-hop AS.
  - Path can be used to make routing decisions, e.g., to avoid loops
  - Pure distance vector does not enable policies
  - Link state does not scale and exposes policies
- When advertising a prefix, advert includes BGP attributes
  - Prefix + other attributes = "route"
- When gateway router receives route advertisement, uses ingress filters to accept/decline
  - Can make decision based on ASes on path, e.g., to avoid loops

BGP messages

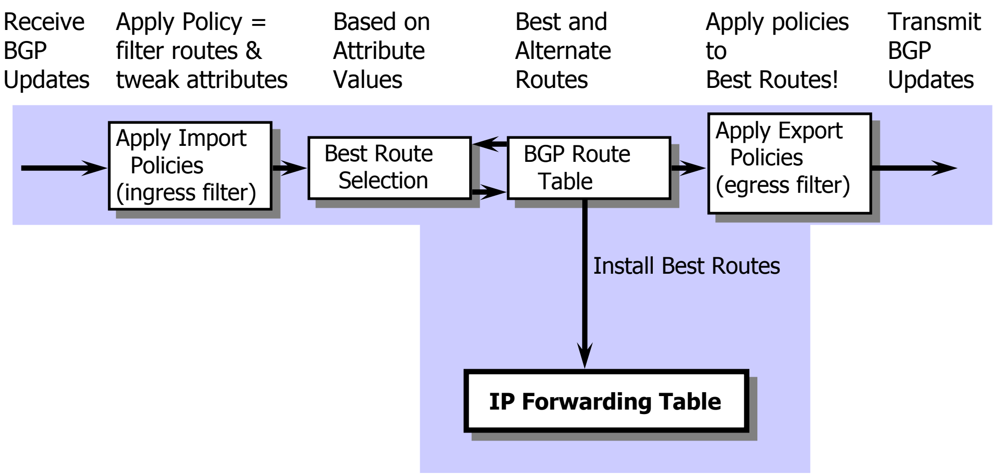
Peers exchange BGP messages using TCP

- OPEN:
  - Opens TCP conn. to peer
  - Authenticates sender
- UPDATE:
  - Advertises new routes (or withdraws old)
- KEEPALIVE:
  - Keeps conn alive in absence of UPDATES, ACKs OPEN request
- NOTIFICATION:
  - Reports errors in previous msg; closes a connection

Process:

- Initialization: Open ⇨ Updates for all routes
- Ongoing: Updates for changed routes

BGP route processing



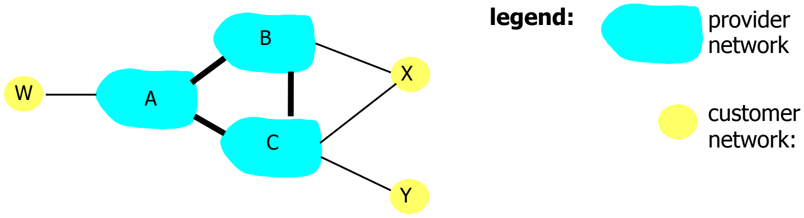
Routing policy

- Reflects goals of network provider
  - Which routes to accept from other ASes
  - How to manipulate the accepted routes
  - How to propagate routes through network
  - How to manipulate routes
    before they leave the AS
  - Which routes to send to another AS

Routing policy: Examples

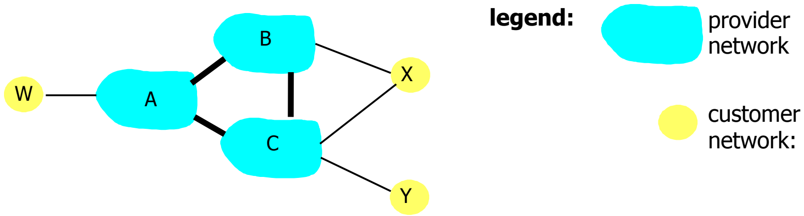
- Honor business relationships
  (E.g., customers get full-table; peers only customer prefixes)
  (E.g., prefer customer routes over peer routes over upstream routes)
- Allow customers a choice of route
  (E.g., on customer request do not export prefix to AS x, etc.)
- Enable customer traffic engineering
  (E.g., prepend x times to all peers or to specified AS)
- Enable DDoS defense for customers
  (E.g., blackholing by rewriting the next hop)
- ...

BGP routing policy



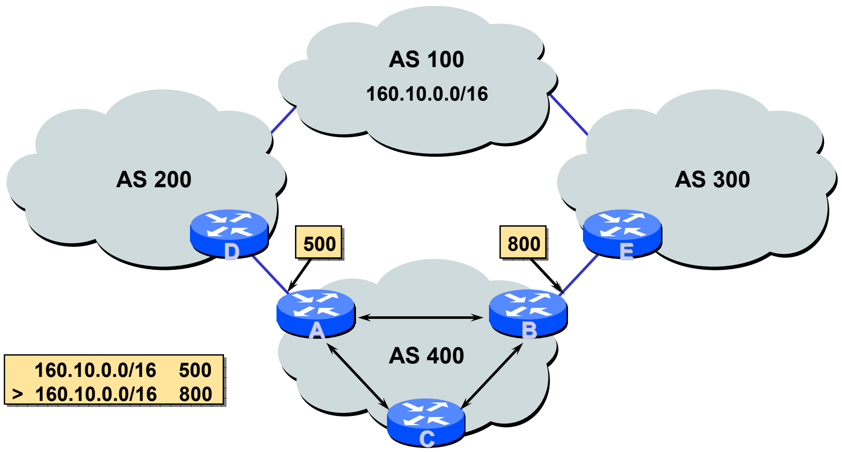
- A,B,C are provider networks
- X,W,Y are customer (of provider networks)
- X is dual-homed: attached to two networks
  - X does not want to route from B via X to C
  - .. so X will not advertise to B a route to C

BGP routing policy (2.)



- A advertises to B the path AW
- B advertises to X the path BAW
- Should B advertise to C the path BAW?
  - 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!

Local preference attribute



- Path with highest local preference wins
- Allows providers to prefer routes

BGP route selection

- ❒ Router learn > 1 route to some prefix
- ❒ Router must select best route.
- ❒ Elimination rules:
  1. Local preference value attribute: policy decision
  2. Shortest AS-PATH
  3. Best MED (multi-exit-discriminator)
  4. Closest NEXT-HOP router: hot potato routing
  5. Additional criteria
  6. IP address of peer

Different types of ASes

- ❒ Providers: Offer connectivity to direct customer
  offer transit to other ISPs
- ❒ Customers: Buy connectivity from providers
- ❒ Peers: Exchange customers traffic at no cost
- ❒ Siblings: others

	Own Routes	Customer's Routes	Sibling's Route	Provider's Route	Peer's Route
Exporting to a Provider	✕	✕	✕		
Exporting to a Customer	✕	✕	✕	✕	✕
Exporting to a Peer	✕	✕	✕		

OSPF (Open Shortest Path First)

- "Open": Specification publicly available
- Uses the Link State algorithm
  - State: per router info about itself and attached networks
  - OSPF advertisements: propagates state
  - Link state database: state of all routers
  - Topology map: derived from link state database

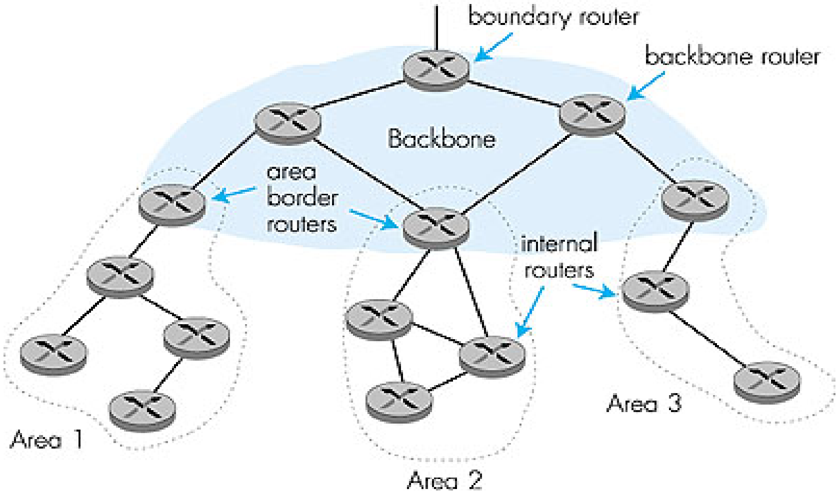
OSPFv2: Components

- Who is my neighbor?
  - Hello Protocol
- With whom I want to talk? (LAN!!!)
  - Designated router/Backup designated router concept
- What info am I missing?
  - Database synchronization
- How do I distribute info?
  - Advertisements disseminated to entire Autonomous System (via reliable flooding)
  - OSPF messages directly over IP (rather than TCP or UDP)
- Route computation
  - From link state database with Dijkstra's algorithm
  - Supports equal-cost path routing

OSPF “advanced” features

- Security: All OSPF messages are authenticated (to prevent malicious intrusion)
- Multiple same-cost paths allowed
- For each link, multiple cost metrics for different TOS (eg, satellite link cost set “low” for best effort; high for real time)
- Integrated uni- and multicast support:
  - Multicast OSPF (MOSPF) uses same topology data base as OSPF
- Hierarchical OSPF in large domains

Hierarchical OSPF



Hierarchical OSPF (2.)

- Two-level hierarchy: Local area and backbone.
  - Link-state advertisements only in respective areas.
  - Nodes in each area have detailed area topology; only know direction (shortest path) to networks in other areas.
- Area Border routers "summarize" distances to networks in the area and advertise them to other Area Border routers.
- Backbone routers: Run an OSPF routing algorithm limited to the backbone.
- Boundary routers: Connect to other ASs.

Why different Intra- and 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

We need BOTH!