Network Layer - Routing

Dr. T. Venkatesh
Dept of CSE, IIT Guwahati

Slides from lectures of Prof. Srini Seshan, CMU and Prof. Jim Kurose, UMass, Amherst

IP Routing - Outline

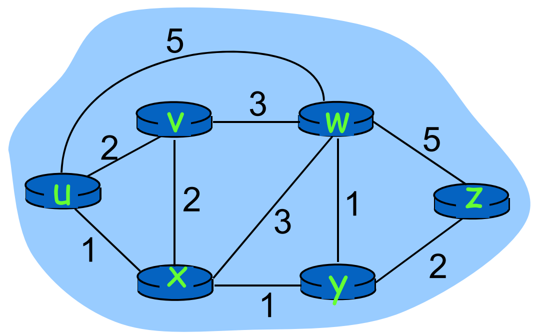
- IP Routing
  - Classification
  - General principles
  - Link state routing
  - Distance vector routing
  - Routing in the internet - heirarchical
  - OSPF, RIP protocols
  - Inter-domain routing
  - BGP
  - Joint routing with inter and intra-domain routing

Routing Protocols

Routing protocol goal: determine "good" paths (equivalently, routes), from sending hosts to receiving host, through network of routers

- path: sequence of routers packets will traverse in going from given initial source host to given final destination host
- "good": least-cost, fastest, least congested
- routing: a "top-10" networking challenge!

Graph Abstraction of the Network



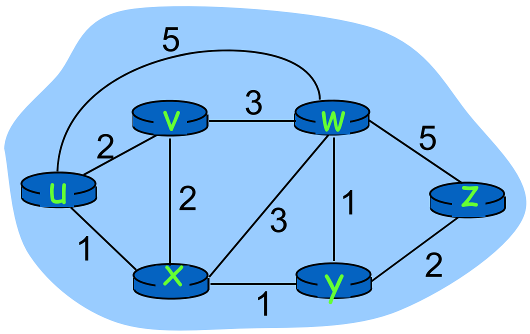
graph: G = (N,E)

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) }

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: Costs



$c(x,x') = \text{cost of link } (x,x')$
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 the least cost path

Routing Algorithms/Protocols

Issues Need to Be Addressed:

- Route selection may depend on different criteria
  - Performance: choose route with smallest delay
  - Policy: choose a route that doesn't cross .gov network
- Adapt to changes in network topology or condition
  - Self-healing: little or no human intervention
- Scalability
  - Must be able to support large number of hosts, routers

Classical Distributed Routing Paradigms

- Hop-by-hop Routing
  - Each packet contains destination address
  - Each router chooses next-hop to destination
    - routing decision made at each (intermediate) hop!
    - packets to same destination may take different paths!
  - Example: IP's default datagram routing
- Source Routing
  - Sender selects the path to destination precisely
  - Routers forward packet to next-hop as specified
    - Problem: if specified path no longer valid due to link failure!
  - Example:
    - IP's loose/strict source route option
    - virtual circuit setup phase (MPLS)

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

Link State Algorithm

- Basic idea: Distribute to all routers
  - Topology of the network
    - Cost of each link in the network
- Each router independently computes optimal paths
  - From itself to every destination
  - periodically or triggered by changes
  - Routes are guaranteed to be loop free if
    - Each router sees the same cost for each link
    - Uses the same algorithm to compute the best path
- gives forwarding table for that node
- iterative: after k iterations, know least cost path to k dest.'s

Topology Dissemination

- Each router creates a set of link state packets (LSPs)
  - Describing its links to neighbors
  - LSP contains
    - Router id, neighbor's id, and cost to its neighbor
- Copies of LSPs are distributed to all routers
  - Using controlled flooding
- Each router maintains a topology database
  - Database containing all LSPs

Dijsktra’s Algorithm

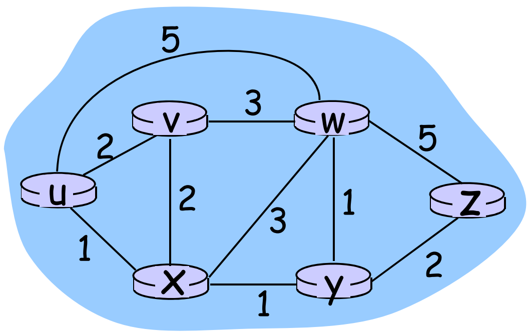
```
Initialization:
  N' = {u}
  for all nodes v
    if v adjacent to u
        then D(v) = c(u,v)
    else D(v) = ∞

 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'
```

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

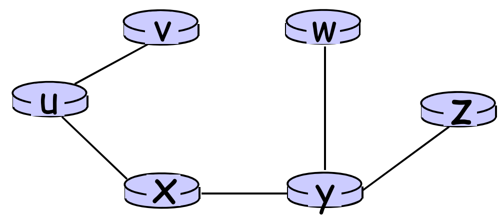
Dijkstra's algorithm: example

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: 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)

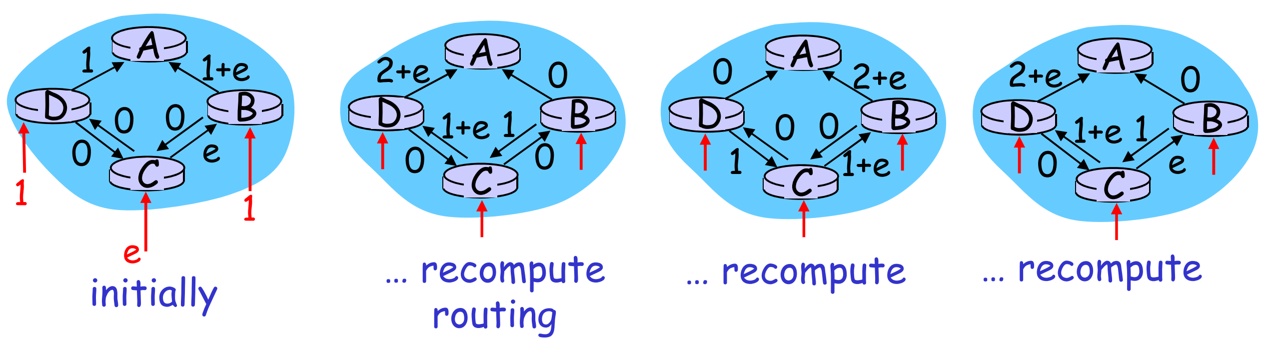
Dijkstra's algorithm, discussion

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(nlogn)

Oscillations possible:

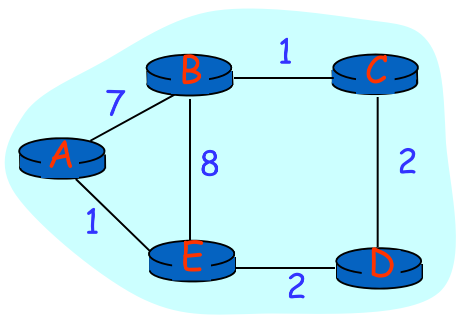
- e.g., link cost = amount of carried traffic



Distance Vector Routing

- Iterative, asynchronous, distributed algorithm
- Local knowledge obtained from neighbours
- Each router maintains a distance table
  - A row for each possible destination
  - A column for each neighbor
    - $D_X(Y,Z)$: distance from X to Y via Z
    - $D_X(Y)$: = $\min_Z \{D_x(Y,Z)\}$: shortest path from X to Y
- Exchanges distance vector with neighbors
  - Distance vector: current least cost from X to each destination
- Computes shortest distance to other nodes based on DV
- A router tells neighbors its distance to every router
  - Communication between neighbors only
- Based on Bellman-Ford algorithm
  - Computes “shortest paths”

Distance Table: Example



$D_E()$		cost to destination via	
	A	B	D
destination A	1	15	12
B	8	8	5
C	9	9	4
D	10	11	2

From Distance Table to Routing Table

Distance table

$D_E()$	A	B	D
A	1	15	12
B	8	8	5
C	9	9	4
D	10	11	2

(columns: cost to destination via; rows: destination)

Routing table (or a distance vector)

destination	Outgoing link to use, cost
A	A ,1
B	D, 5
C	D, 4
D	D, 2

Distance Vector Algorithm

Bellman-Ford equation (dynamic programming)

let

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

then

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

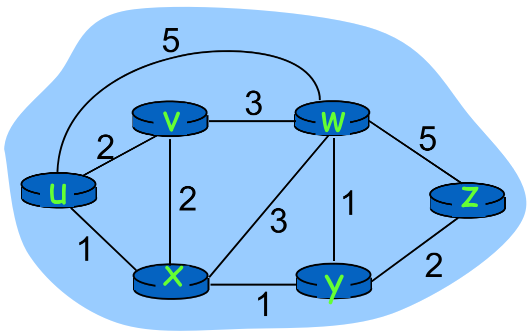
v

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 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 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:

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

- under natural conditions, the estimate $D_x(y)$ *converges to the actual least cost* $d_x(y)$
- *If distance vector changes, send updates to all neighbours*

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

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

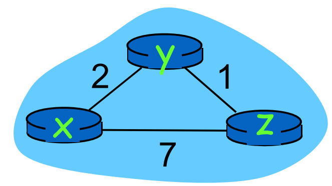
node z table

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

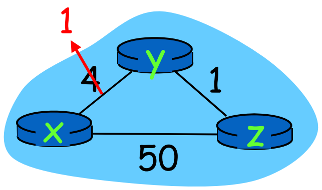
time



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



"good news travels fast"

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

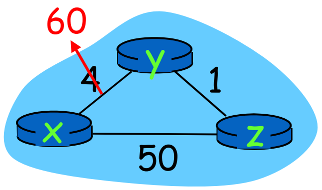
t_1 : *z* receives update from *y*, updates its table, computes new least cost to *x* , sends its neighbors its DV.

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*.

Distance Vector: Link Cost Changes

link cost changes:

- node detects local link cost change
- uses earlier older DV from z to calculate path cost as 6!
- *bad news travels slow* - “count to infinity” problem!
- How many iterations before algorithm stabilizes?



“Fixes” to Count-to-Infinity Problem

- Split horizon
  - A router never advertises the cost of a destination to a neighbor
    - If this neighbor is the next hop to that destination
- Split horizon with poisonous reverse
  - If X routes traffic to Z via Y, then
    - X tells Y that its distance to Z is infinity
      - Instead of not telling anything at all
  - Accelerates convergence
- Will this completely solve count to infinity problem?

Link State *vs* Distance Vector

- Tells everyone about neighbors
- Controlled flooding to exchange link state
- Dijkstra's algorithm
- Each router computes its own table
- May have oscillations
- Open Shortest Path First (OSPF)

- Tells neighbors about everyone
- Exchanges distance vectors with neighbors
- Bellman-Ford algorithm
- Each router's table is used by others
- May have routing loops
- Routing Information Protocol (RIP)

Comparison of LS and DV Algorithms

message complexity

- ***LS:*** with n nodes, E links, O(nE) msgs sent
- ***DV:*** exchange between neighbors only
  - messages count 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

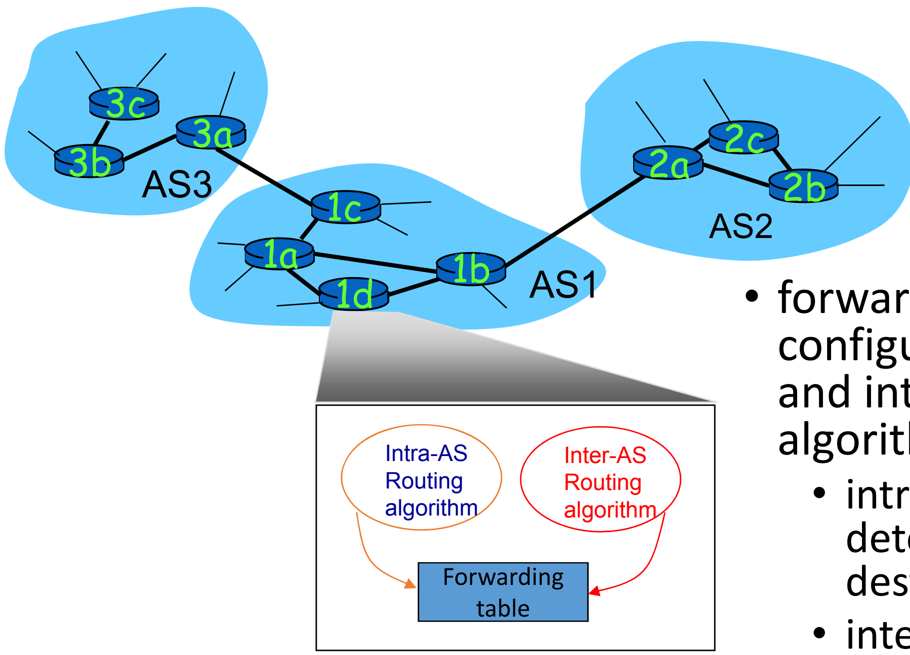
DV:

- DV node can advertise incorrect *path* cost
- each node's table used by others
  - error propagates thru network

Routing in the Internet

- The Global Internet consists of Autonomous Systems (AS) interconnected with each other:
  - Stub AS: small corporation: one connection to other AS’s
  - Multihomed AS: large corporation (no transit): multiple connections to other AS’s
  - Transit AS: provider, hooking many AS’s together
- Two-level routing:
  - Intra-AS: administrator responsible for choice of routing algorithm within network
  - Inter-AS: unique standard for inter-AS routing: BGP

Interconnected ASes



- forwarding table configured by both intra- and inter-AS routing algorithm
  - intra-AS routing determine entries for destinations within AS
  - inter-AS & intra-AS determine entries for external destinations

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, update traffic

Performance:

- Intra-AS: can focus on performance
- Inter-AS: policy may dominate over performance

Will Talk about Inter-AS routing (& BGP) later!

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
  - IS-IS: Intermediate System to Intermediate System (OSI Standard)
  - EIGRP: Extended Interior Gateway Routing Protocol (Cisco proprietary)

RIP (Routing Information Protocol)

- Distance vector algorithm
- Included in BSD-UNIX Distribution in 1982
- Distance metric: # of hops (max = 15 hops)
  - *Can you guess why?*
- Distance vectors: exchanged among neighbors every 30 sec via Response Message (also called **advertisement**)
- Each advertisement: list of up to 25 destination nets within AS

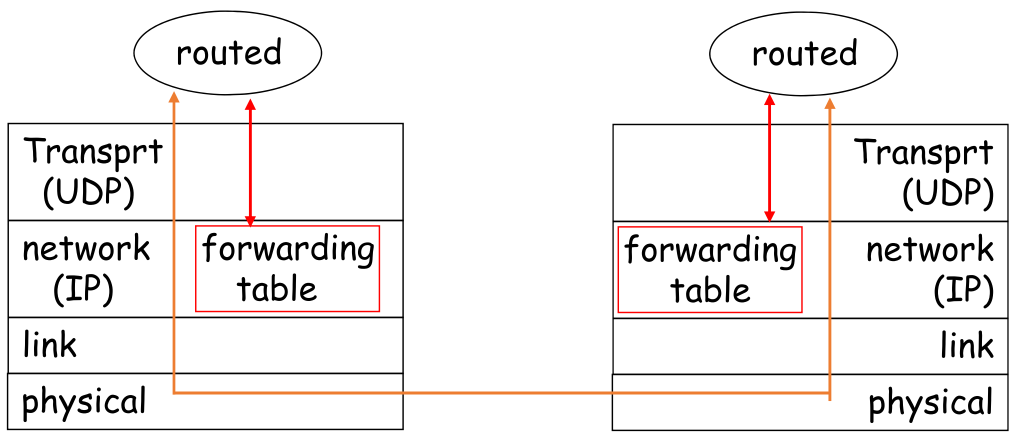
RIP: Link Failure and Recovery

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



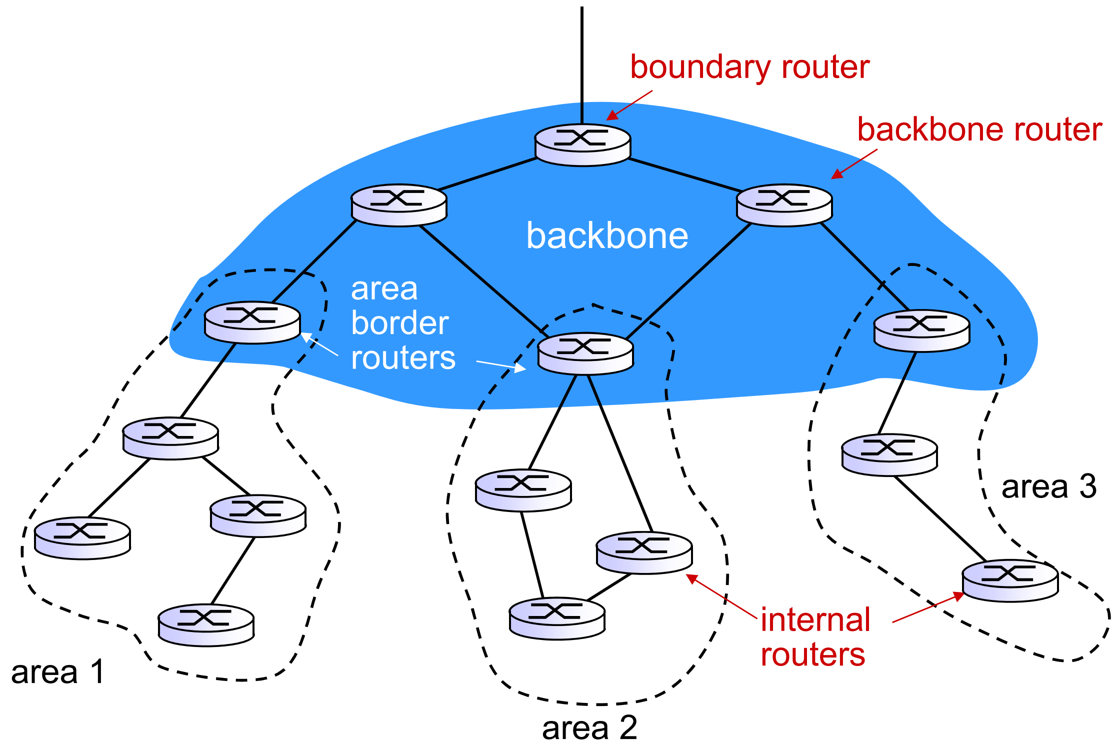
OSPF (Open Shortest Path First)

- uses link-state algorithm
  - link state packet dissemination
  - topology map at each node
  - route computation using Dijkstra's algorithm
- router floods OSPF link-state advertisements to all other routers in *entire* AS
  - carried in OSPF messages directly over IP
  - link state: for each attached link
- *IS-IS routing* protocol: nearly identical to OSPF

OSPF 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 ("Type-of-Services")
  - e.g., satellite link cost set "low" for best effort; high for real time)
- 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 ASes.

BGP (Border Gateway Protocol)

- *The* de facto standard (BGP-4)
- **Path Vector** protocol:
  - similar to Distance Vector protocol
  - each Border Gateway broadcast to neighbors (peers) *entire path* (i.e., sequence of ASes) to destination
  - BGP routes to networks (ASes), not individual hosts
- E.g., Gateway X may announce to its neighbors it "knows" a (AS) path to a *destination network*, Z, via *a series of ASes*:

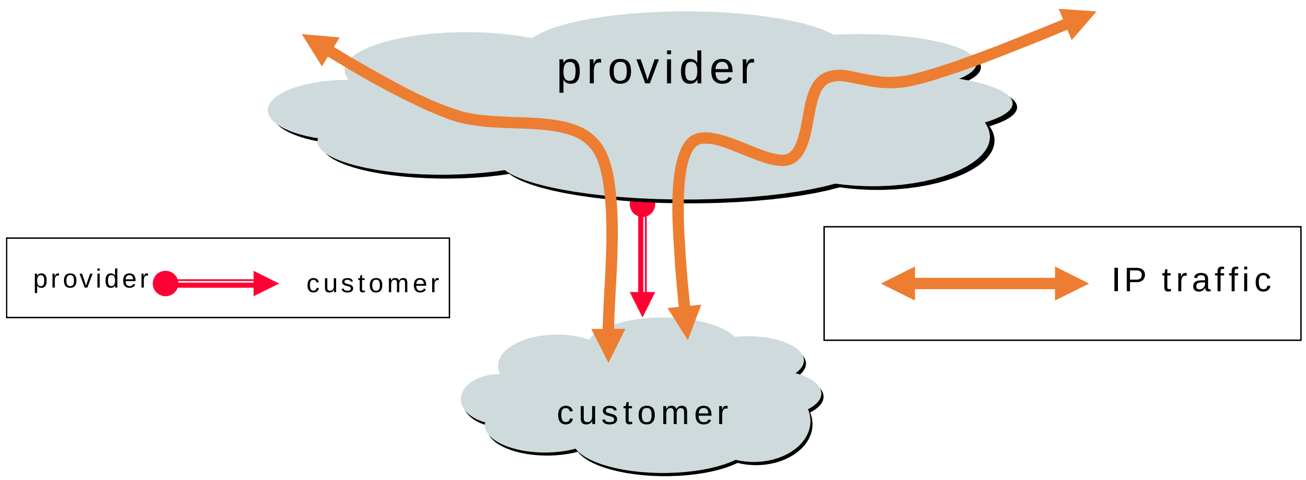
  Path (X,Z) = X,Y1,Y2,Y3,…,Z
- BGP border gateways referred to as BGP *speakers*

BGP Operations: Policy Routing

Functions of a BGP router

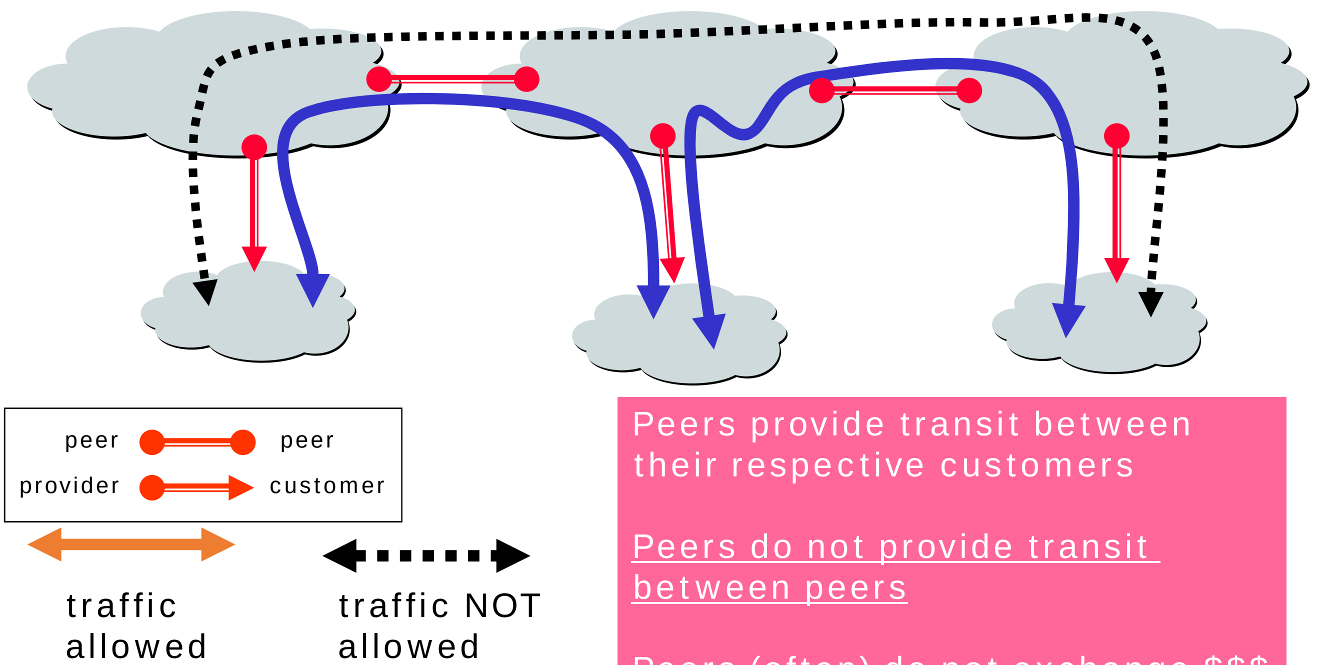
- Receiving and *filtering* route advertisements from directly attached neighbor(s)
  - To accept or not accept route advertisements depends on policies (e.g., whether you "trust" your neighbors)
- Route selection (rank diff. routes to same dest. network).
  - to route to destination X, which path (of several advertised) will be taken?
  - route selection based on policies (e.g., always prefer route advertisement from "good old" neighbor Y)
- Filtering and sending (certain) route advertisements to neighbors
  - what/whether to advertise to your neighbors also depends on policies (e.g., don't tell your neighbor Z that you know a route to destination X)

Customers and Providers



Customer pays provider for access to the Internet

The Peering Relationship

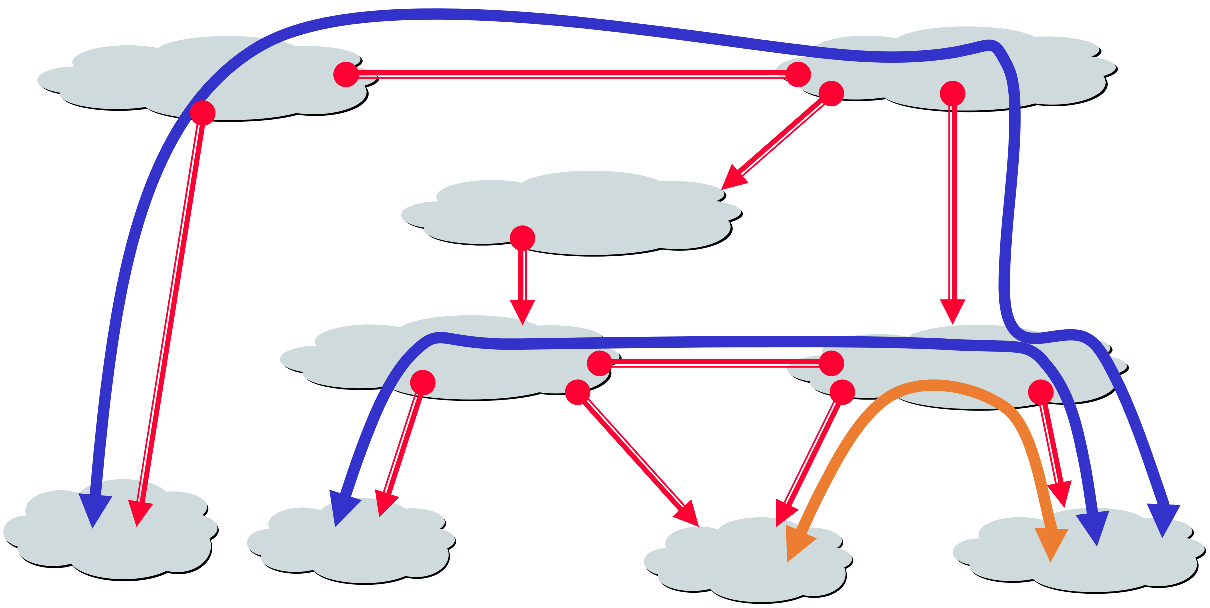


Peers provide transit between their respective customers

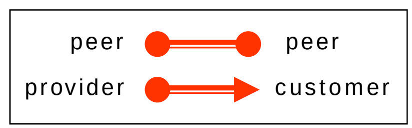
Peers do not provide transit between peers

Peers (often) do not exchange $$$

Peering Provides Shortcuts



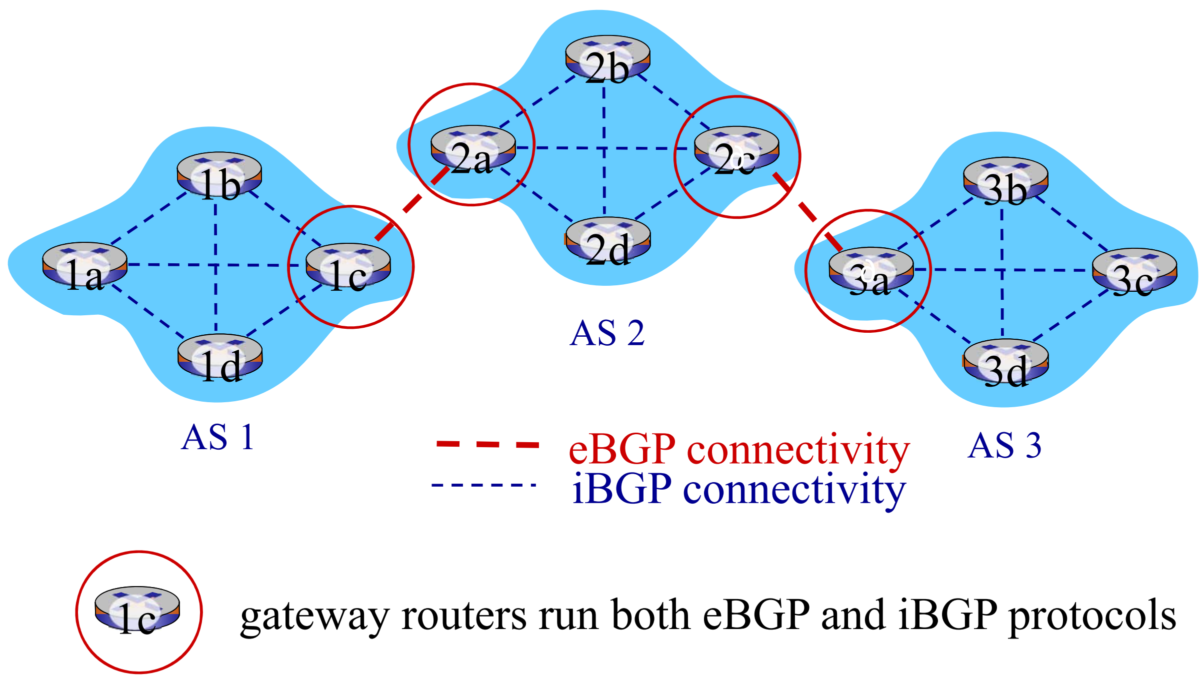
Peering also allows connectivity between
the customers of "Tier 1" providers.



Internet Inter-AS Routing: BGP

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

eBGP, iBGP Connections



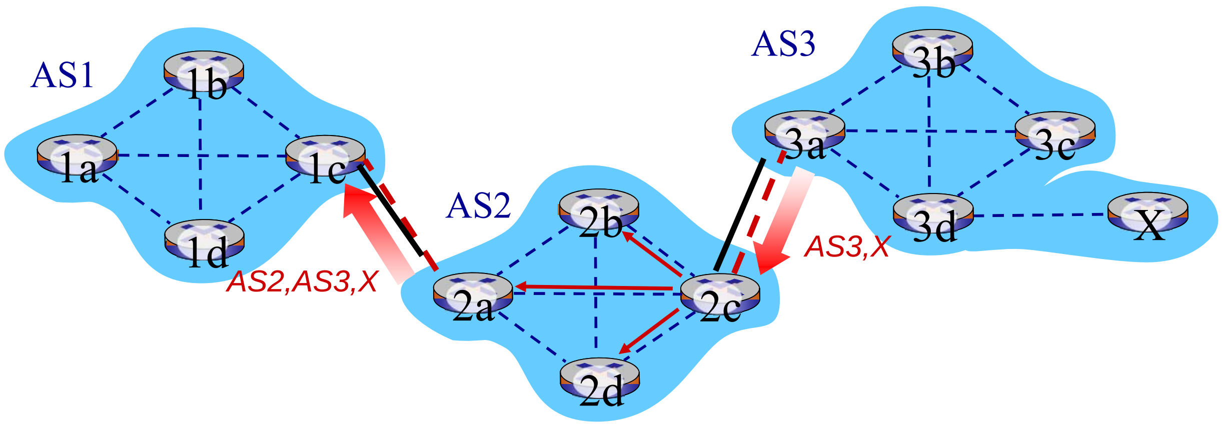
BGP Messages

- BGP messages exchanged using TCP.
- BGP messages:
  - OPEN: opens TCP connection to peer and authenticates sender
  - KEEPALIVE keeps connection alive in absence of UPDATES; also ACKs OPEN request
    - OPEN/KEEPALIVE establish & maintain BGP neighbor relation
  - UPDATE: advertises new path (or withdraws old)
  - NOTIFICATION: reports errors in previous msg; also used to close connection

Path Attributes and BGP Routes

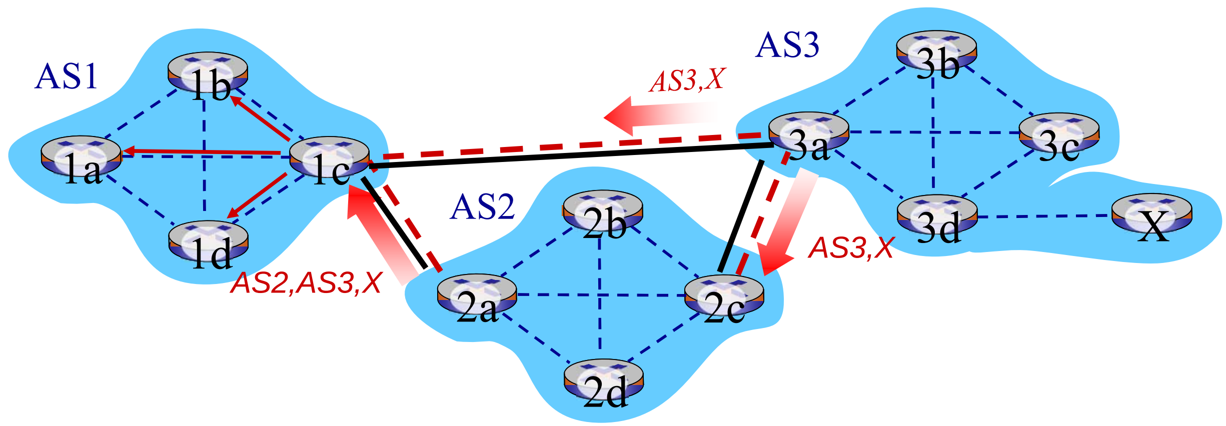
- advertised prefix includes BGP attributes
  - prefix + attributes = "route"
- two important attributes:
  - AS-PATH: list of ASes through which prefix advertisement has passed
  - NEXT-HOP: indicates specific internal-AS router to next-hop AS
- *Policy-based routing:*
  - gateway receiving route advertisement uses *import policy* to accept/decline path (e.g., never route through AS Y).
  - AS policy (*export policy)* also determines whether to *advertise* path to other other neighboring ASes

BGP Path Advertisement



- AS2 router 2c receives path advertisement AS3,X (via eBGP) from AS3 router 3a
- Based on AS2 policy, AS2 router 2c accepts path AS3,X, propagates (via iBGP) to all AS2 routers
- Based on AS2 policy, AS2 router 2a advertises (via eBGP) path AS2, AS3, X to AS1 router 1c

BGP Path Advertisement



gateway router may learn about multiple paths to destination:

- AS1 gateway router 1c learns path *AS2,AS3,X* from 2a
- AS1 gateway router 1c learns path *AS3,X* from 3a
- Based on policy, AS1 gateway router 1c chooses path *AS3,X, and advertises path within AS1 via iBGP*

BGP Attributes

```
Value      Code                              Reference
-----      --------------------------------- ---------
   1       ORIGIN                            [RFC1771]
   2       AS_PATH                           [RFC1771]
   3       NEXT_HOP                          [RFC1771]
   4       MULTI_EXIT_DISC                   [RFC1771]
   5       LOCAL_PREF                        [RFC1771]
   6       ATOMIC_AGGREGATE                  [RFC1771]
   7       AGGREGATOR                        [RFC1771]
   8       COMMUNITY                         [RFC1997]
   9       ORIGINATOR_ID                     [RFC2796]
  10       CLUSTER_LIST                      [RFC2796]
  11       DPA                                  [Chen]
  12       ADVERTISER                        [RFC1863]
  13       RCID_PATH / CLUSTER_ID            [RFC1863]
  14       MP_REACH_NLRI                     [RFC2283]
  15       MP_UNREACH_NLRI                   [RFC2283]
  16       EXTENDED COMMUNITIES                [Rosen]
 ...
 255       reserved for development
```

From IANA: http://www.iana.org/assignments/bgp-parameters

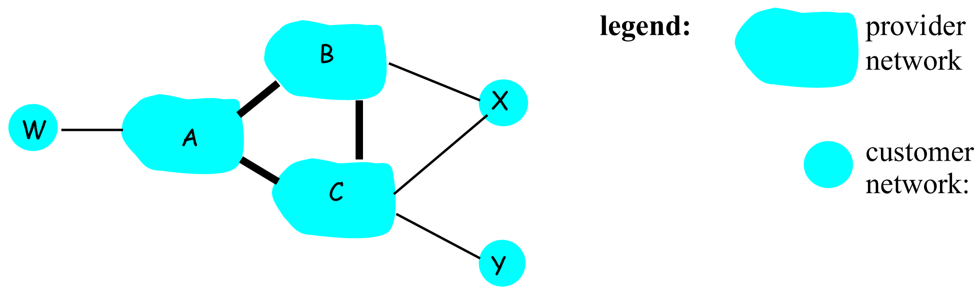
Most important attributes

Not all attributes need to be present in every announcement

Enforcing relationships

- Two mechanisms:
- Export filters
  - Control what you send over BGP
- Import *ranking*
  - Controls which route you prefer of those you hear.
  - "LOCALPREF" – Local Preference Attribute.
  - Hotpotato routing

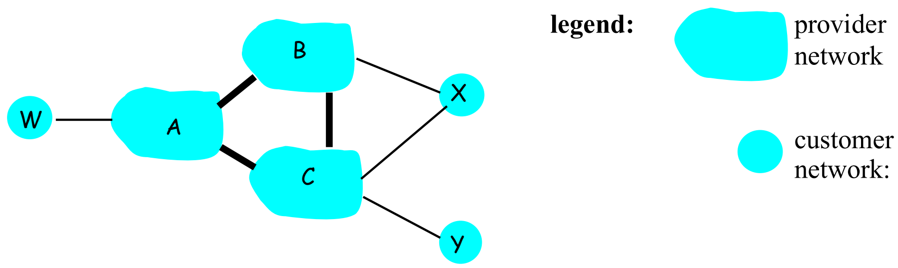
BGP: Controlling Who Routes to You



a simple BGP scenario

- A,B,C are provider networks
- X,W,Y are customer (of provider networks)
- X is dual-homed: attached to two networks
  - C tells X networks belonging to C, i.e., a route to them via C
  - X does not want to carry traffic from B via X to C
  - .. so X will not advertise to B any route to networks in C learned from C

BGP: Controlling Who Routes to You



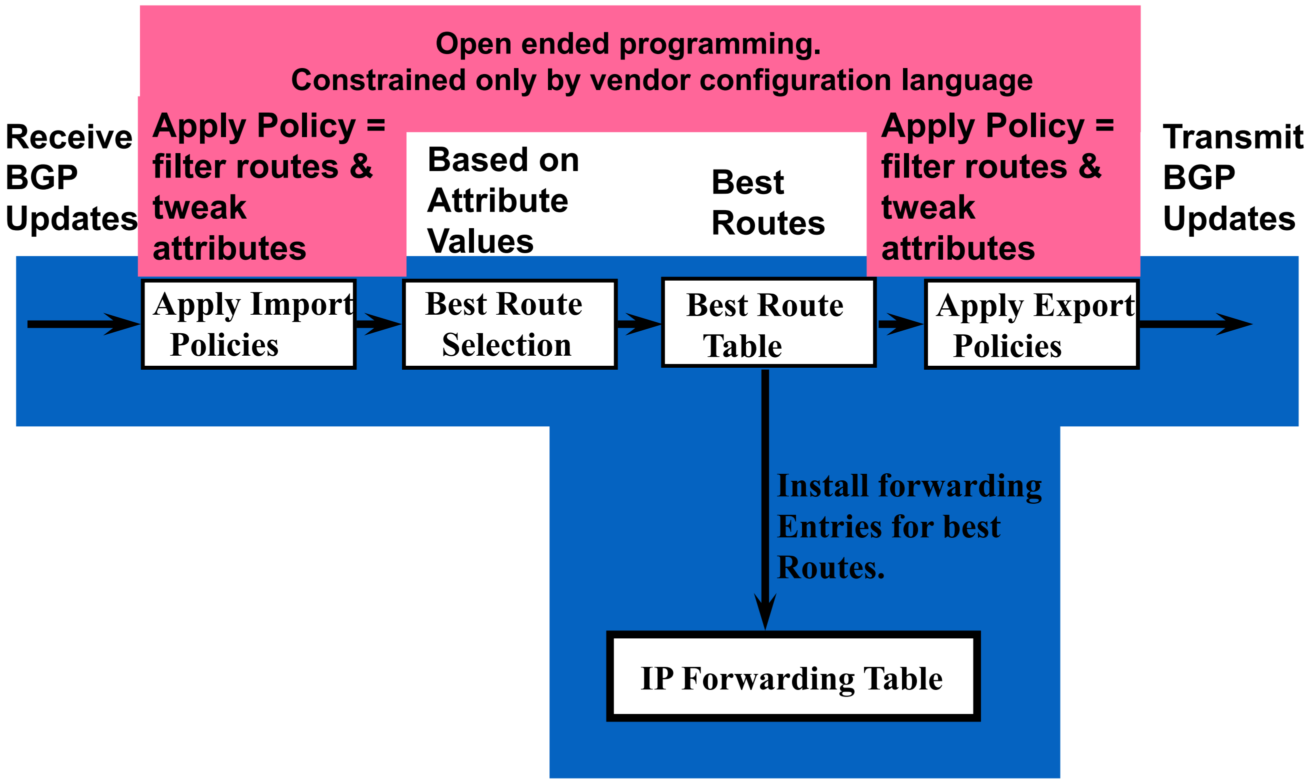
a simple BGP scenario

- 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!

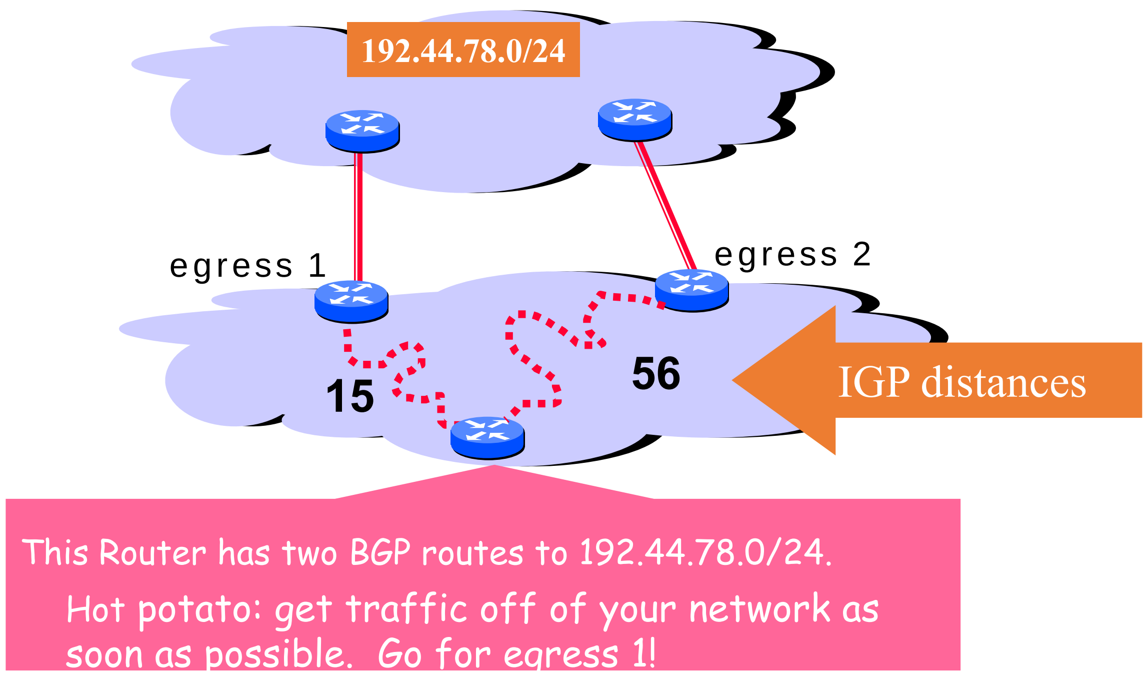
Implementing Customer/Provider and Peer/Peer relationships

- Enforce transit relationships
  - Outbound route filtering
- Enforce order of route preference
  - provider < peer < customer
- Enforce Valley Free Routing
  - Customer not used between two peer providers

BGP Route Processing

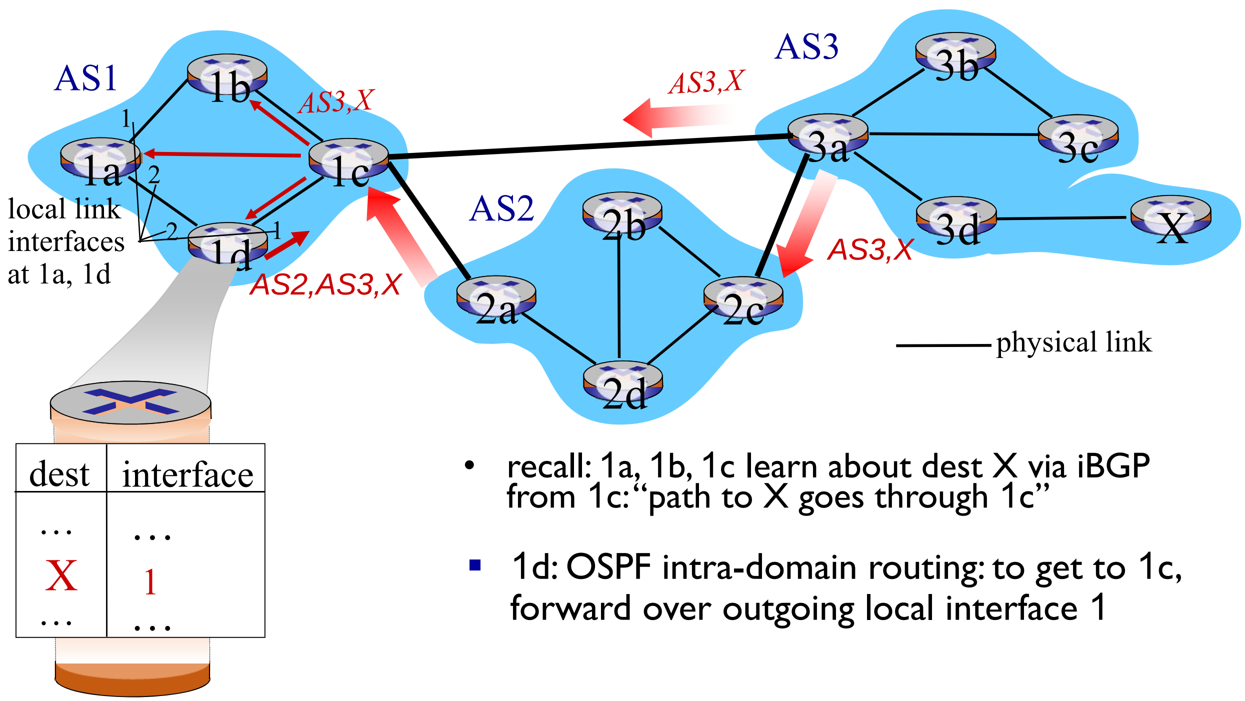


Early Exit or Hot Potato Routing:
Go for the Closest Egress Point



BGP, OSPF, Forwarding Table Entries

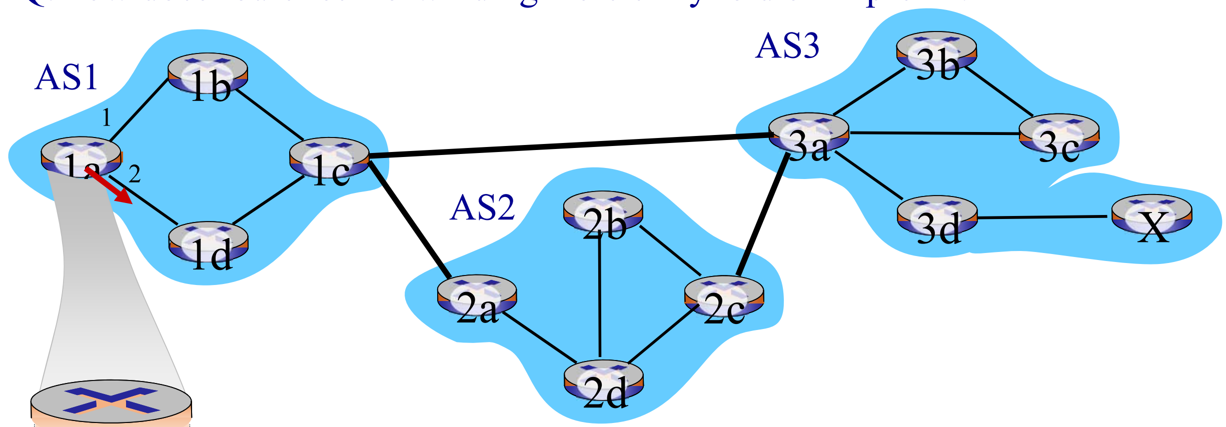
Q: how does router set forwarding table entry to distant prefix?

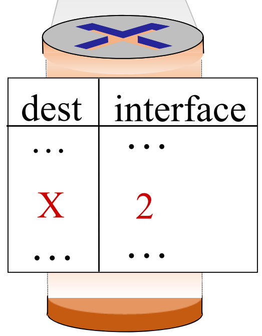


- recall: 1a, 1b, 1c learn about dest X via iBGP from 1c: "path to X goes through 1c"
- 1d: OSPF intra-domain routing: to get to 1c, forward over outgoing local interface 1

BGP, OSPF, Forwarding Table Entries

Q: how does router set forwarding table entry to distant prefix?





- recall: 1a, 1b, 1c learn about dest X via iBGP from 1c: "path to X goes through 1c"

- 1d: OSPF intra-domain routing: to get to 1c, forward over outgoing local interface 1
- 1a: OSPF intra-domain routing: to get to 1c, forward over outgoing local interface 2

Recap: 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:

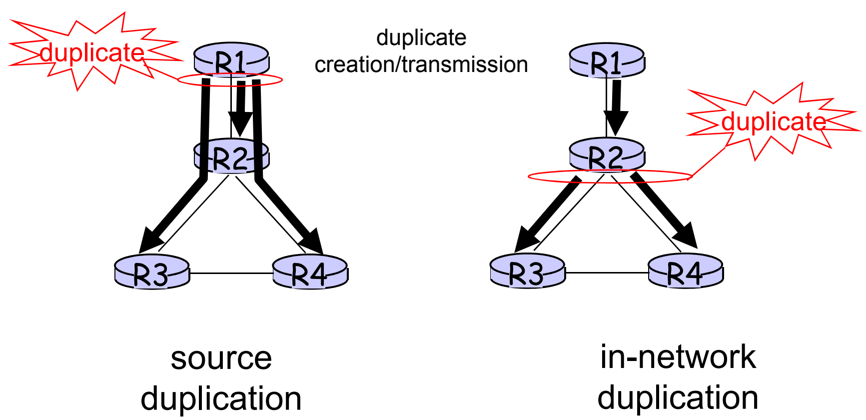
- hierarchical routing saves table size, reduced update traffic

performance:

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

Broadcast Routing

- Deliver packets from source to all other nodes
- Source duplication is inefficient
  - Congestion on outgoing link
  - How to know the addresses?

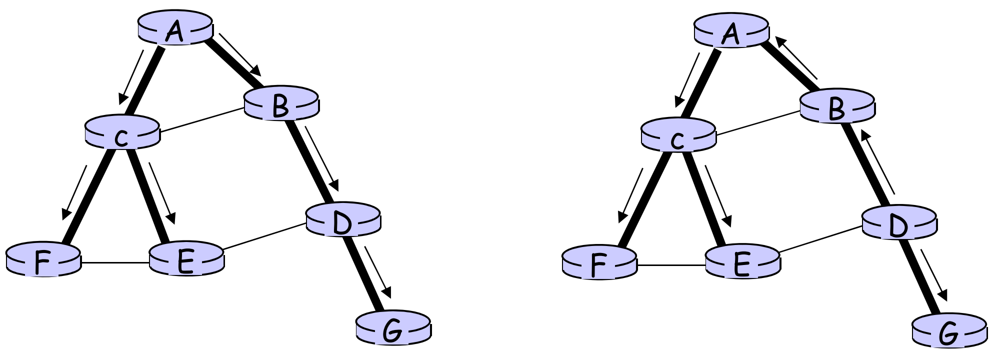


In-network duplication

- Flooding: when node receives broadcast packet, sends copy to all neighbors
  - Problems: looping packets & broadcast storm
- Controlled flooding: broadcast only if it hasn't been done 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

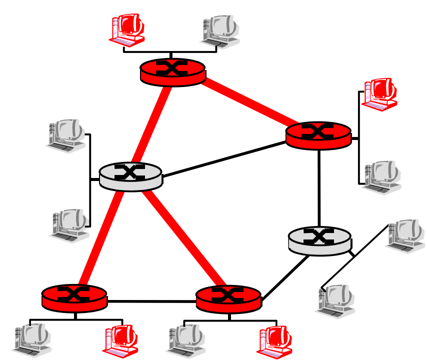
- First construct a spanning tree
- Nodes forward copies only along spanning tree



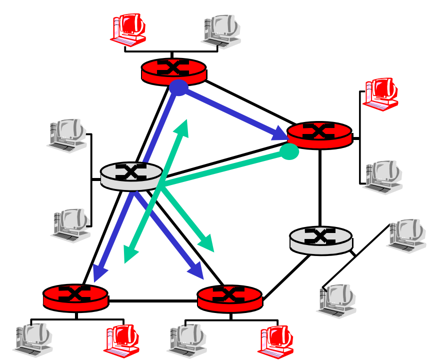
(a) Broadcast initiated at A (b) Broadcast initiated at D

Multicast Routing: Problem Statement

- ***Goal:*** find a tree (or trees) connecting routers having local multicast group members
  - *tree:* not all paths between routers used
  - *source-based:* different tree from each sender to receivers
  - *shared-tree:* same tree used by all group members
- Issues: to identify members of a group and maintain it
  - Membership information maintained by IGMP
  - Receiver driven approach to avoid unwanted messages



Shared tree



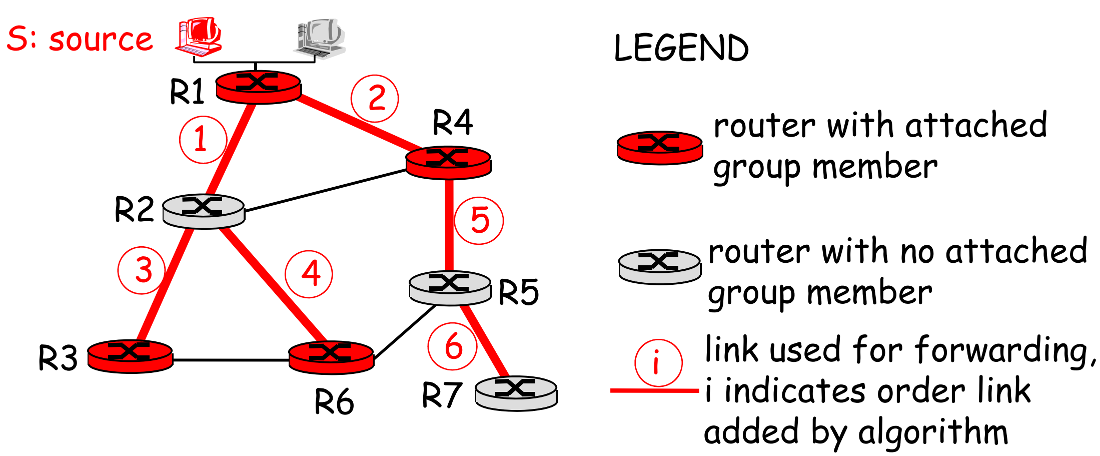
Source-based trees

Approaches for building multicast trees

- **source-based tree:** one tree per source
  - shortest path trees
  - reverse path forwarding
- **group-shared tree:** group uses one tree
  - minimal spanning (Steiner)
  - center-based trees

Shortest Path Tree

- multicast forwarding tree: tree of shortest path routes from source to all receivers
  - Dijkstra's algorithm

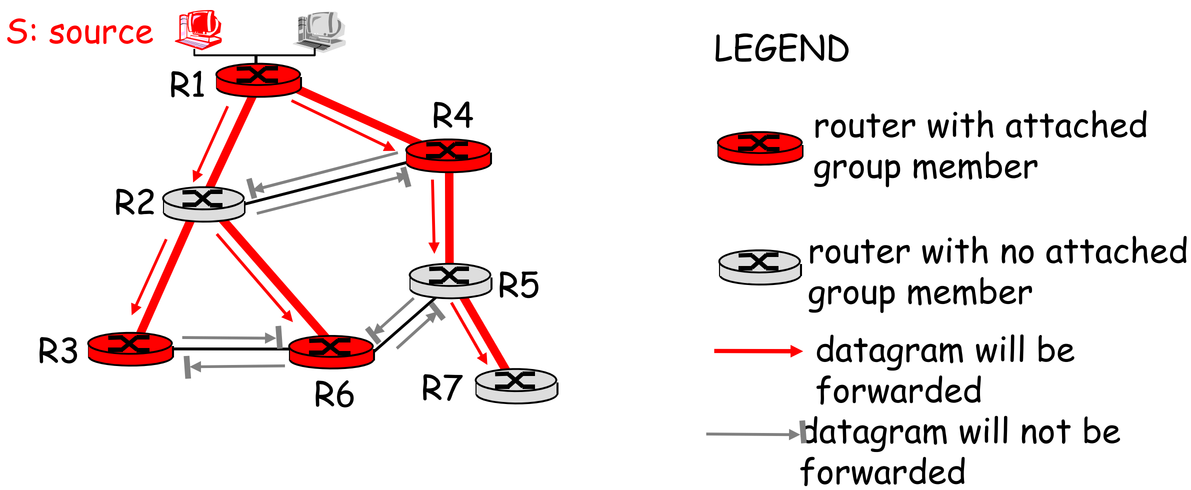


Reverse Path Forwarding

- rely on router's knowledge of unicast shortest path from itself to sender
- each router has simple forwarding behavior:

if (mcast pkt received thru link on shortest path back to source)
then flood datagram onto all outgoing links
else ignore datagram

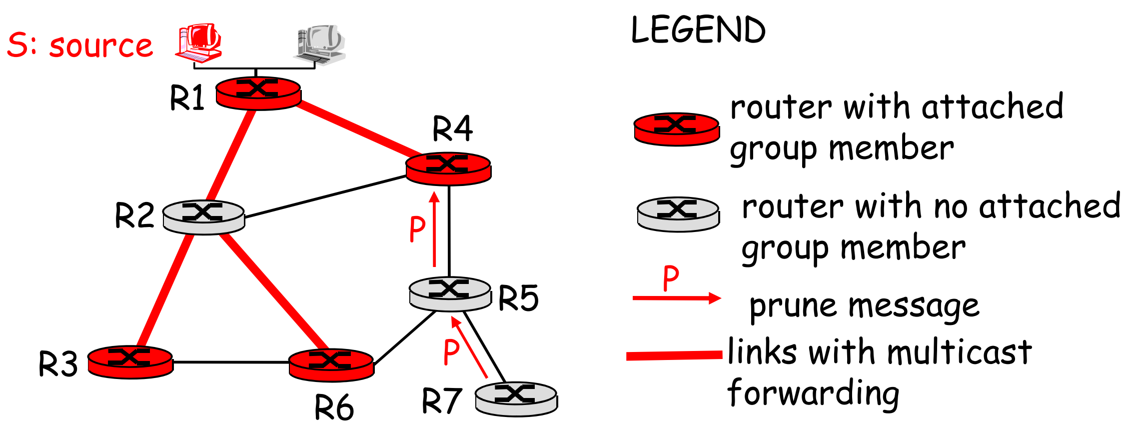
Reverse Path Forwarding: example



- result is a source-specific *reverse* SPT
  - may be a bad choice with asymmetric links

Reverse Path Forwarding: pruning

- forwarding tree contains subtrees with no mcast group members
  - no need to forward datagrams down subtree
  - “prune” msg sent upstream by router with no downstream group members



Shared-Tree: Steiner Tree

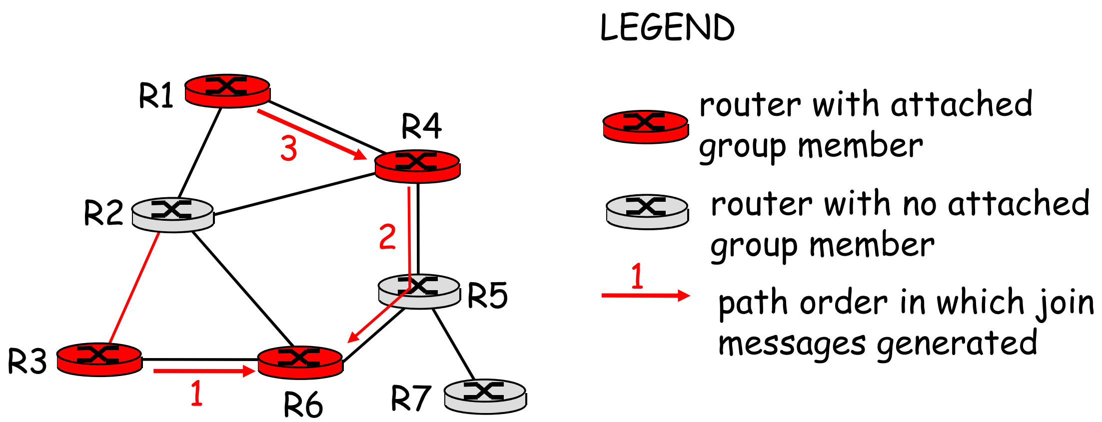
- Steiner Tree: minimum cost tree connecting all routers with attached group members
- problem is NP-complete
- excellent heuristics exists
- not used in practice:
  - computational complexity
  - information about entire network needed
  - monolithic: rerun whenever a router needs to join/leave

Center-based trees

- single delivery tree shared by all
- one router identified as *"center"* of tree
- to join:
  - edge router sends unicast *join-msg* addressed to center router
  - *join-msg* "processed" by intermediate routers and forwarded towards center
  - *join-msg* either hits existing tree branch for this center, or arrives at center
  - path taken by *join-msg* becomes new branch of tree for this router

Center-based trees: an example

Suppose R6 chosen as center:



Internet Group Membership Protocol

- Operates between host and forwarding multicast router
- Group membership information updated at the router with IGMP messages
- Host requests membership for a group address
- Router periodically checks if there are active members in every group
- Membership information used by routers to join or leave multicast tree
- IGMP messages sent as IP datagrams
  - `membership_query, membership_report, leave_group`
- Soft state – If no response, delete the member

Internet Multicasting Routing: DVMRP

- DVMRP: distance vector multicast routing protocol, RFC1075
- *flood and prune:* reverse path forwarding, source-based tree
  - RPF tree based on DVMRP's own routing tables constructed by communicating DVMRP routers
  - initial datagram to mcast group flooded everywhere via RPF
  - routers not wanting group: send upstream prune msgs

PIM: Protocol Independent Multicast

- not dependent on any specific underlying unicast routing algorithm (works with all)
- two different multicast distribution scenarios :

Dense:

- group members densely packed, in "close" proximity.
- group membership by routers *assumed* until routers explicitly prune
- *data-driven* construction on mcast tree (e.g., RPF)

Sparse:

- group members sparsely spread across
- group members "widely dispersed"
- no membership until routers explicitly join
- *receiver- driven* construction of mcast tree (e.g., center-based)