Master Course Computer Networks IN2097

Prof. Dr.-Ing. Georg Carle
Christian Grothoff, Ph.D.
Dr. Nils Kammenhuber

Chair for Network Architectures and Services
Institut für Informatik
Technische Universität München
http://www.net.in.tum.de

Chapter 4: Network Layer

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

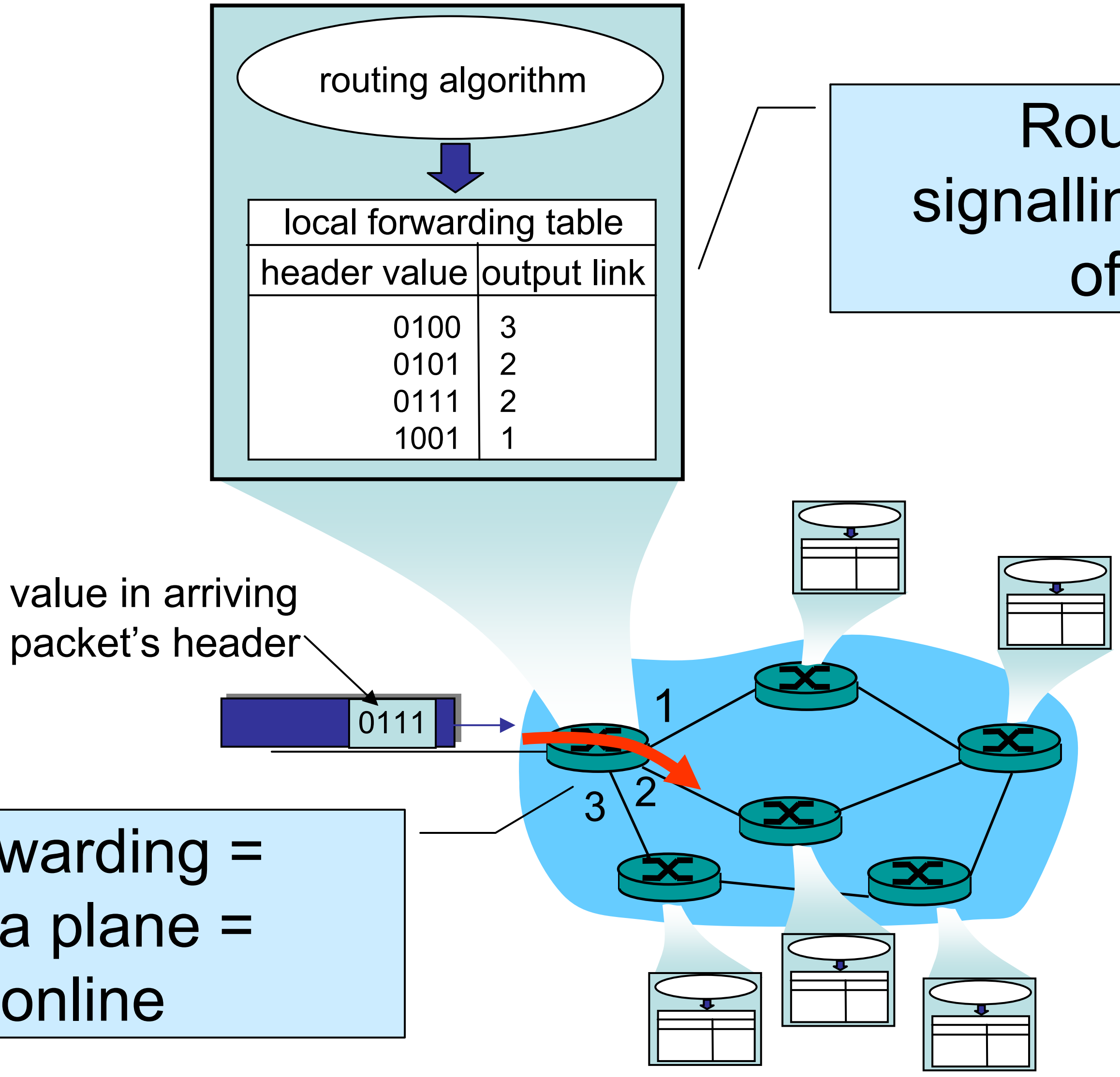
Part 2

- IPv6
- Virtual circuit and datagram networks
- What’s inside a router

Part 3

- **Routing algorithms**
  - Link state
  - Distance Vector
  - Path Vector
  - Hierarchical routing
- Internet routing protocols
  - RIP
  - OSPF
  - BGP
- Business considerations
  - Policy routing
  - Traffic engineering

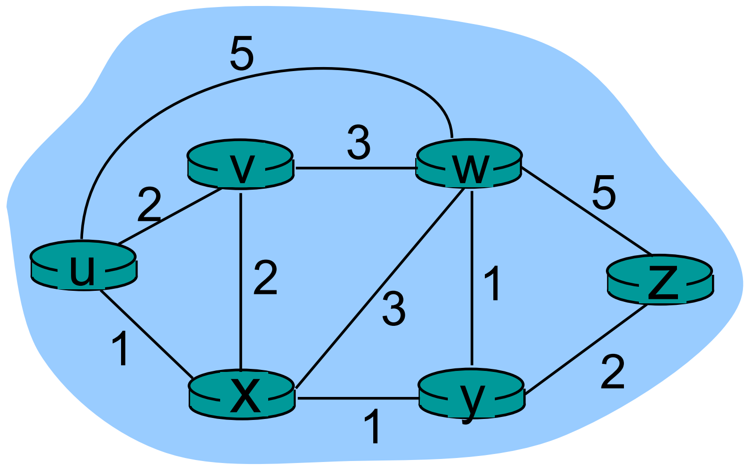
Recall: Interplay between routing and forwarding



Routing = signalling plane = offline

Forwarding = data plane = online

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 and link cost info
- *link state* algorithms (L-S)

Decentralized:

- Router only knows physically-connected neighbors and link costs to neighbors
- Iterative process of computation = exchange of info with neighbors
- *distance vector* algorithms (D-V)
- Variant: *path vector* algorithms

Static or dynamic?

Static:

- Routes change slowly over time

Dynamic:

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

Chapter 4: Network Layer

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

- IPv6
- Virtual circuit and datagram networks
- What's inside a router

Part 3

- **Routing algorithms**
  - **Link state**
  - Distance Vector
  - Path Vector
  - Hierarchical routing
- Internet routing protocols
  - RIP
  - OSPF
  - BGP
- Business considerations
  - Policy routing
  - Traffic engineering

A Link-State Routing Algorithm

- Net topology and link costs made known to each node
  - Accomplished via *link state broadcasts*
  - All nodes have same info
- Each node independently computes least-cost paths from one node (“source”) to all other nodes
  - Usually done using Dijkstra’s shortest-path algorithm
    - refer to any algorithms & data structures lecture/textbook
    - n nodes in network $\Rightarrow$ $O(n^2)$ or $O(n \log n)$
  - Gives forwarding table for that node
- Result:
  - All nodes have the same information,
  - … thus calculate the same shortest paths,
  - … hence obtain consistent forwarding tables

Chapter 4: Network Layer

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

- IPv6
- Virtual circuit and datagram networks
- What's inside a router

Part 3

- **Routing algorithms**
  - Link state
  - **Distance Vector**
  - Path Vector
  - Hierarchical routing
- Internet routing protocols
  - RIP
  - OSPF
  - BGP
- Business considerations
  - Policy routing
  - Traffic engineering

Distance Vector Algorithm

- No node knows entire topology
- Nodes only communicate with neighbours (i.e., no broadcasts)
- Nodes *jointly* calculate shortest paths
  - Iterative process
  - Algorithm == protocol
- Distributed application of Bellman-Ford algorithm
  - refer to any algorithms&data structures lecture/textbook

Distance Vector Algorithm

Bellman-Ford Equation (dynamic programming)

Let

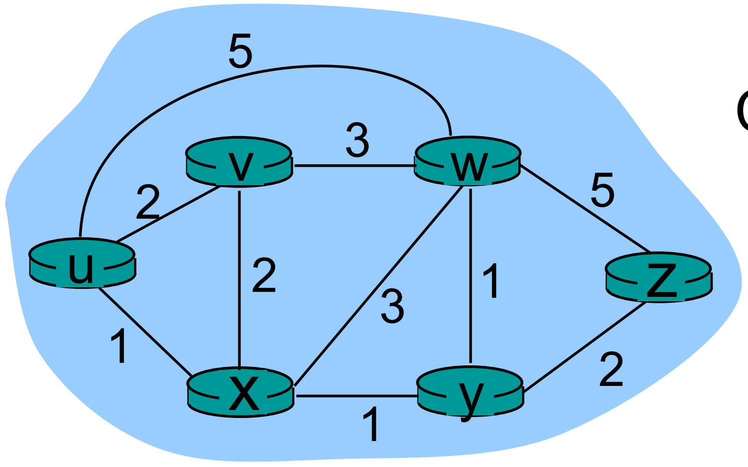
- $c(x,y)$:= cost of edge from x to y
- $d_x(y)$:= cost of least-cost path from x to y

Then

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

where min is taken over all neighbours 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 that achieves minimum is next hop in shortest path
→ forwarding table

Distance Vector Algorithm

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

Distance vector algorithm (4)

Basic idea:

- From time-to-time, each node sends its own distance vector estimate D to neighbors
  - Asynchronously
- When a node x receives new DV estimate from neighbour, 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, these estimates $D_x(y)$ *converge to 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 neighbour

Distributed:

- ❑ Each node notifies neighbors *only* when its DV changes
  - neighbours then notify their neighbours if this caused *their* DV to change
  - etc.

Each node:

Forever:

wait for (change in local link cost *or* message arriving from neighbour}

recompute estimates

if (DV to any destination has changed) { *notify* neighbours }

Distance Vector Algorithm (6)

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

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

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

node y table

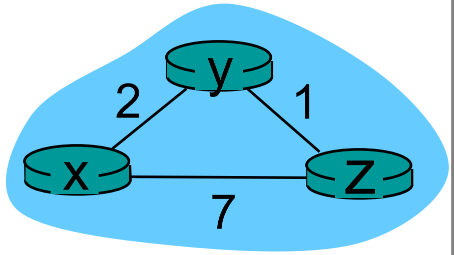
cost to

from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

node z table

cost to

from	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0



time

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

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

node x table

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

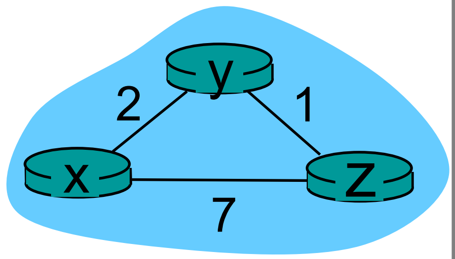
cost to

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

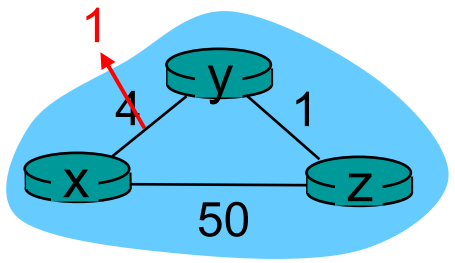
time



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



"good news travels fast"

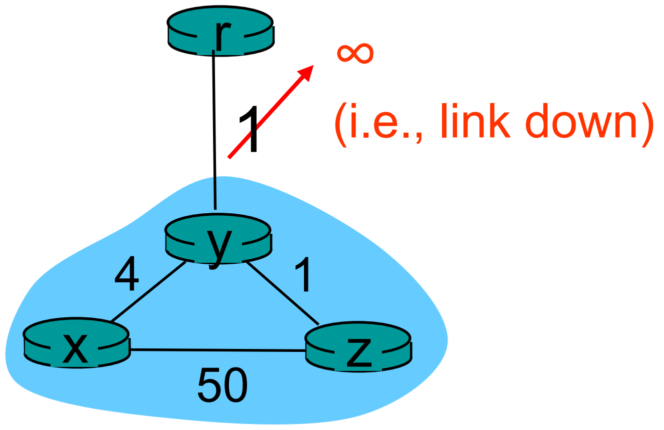
At time t_0, *y* detects the link-cost change, updates its DV, and informs its neighbors.

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*.

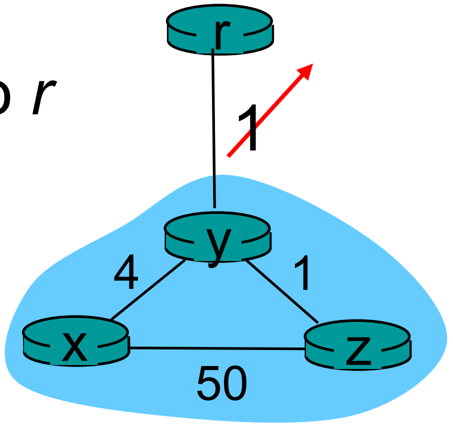
Distance Vector: link cost changes (2)

- But: bad news travels slow — “count to infinity” problem!
- In example: Many iterations before algorithm stabilizes!
  1. Cost increase for *y*→*r*:
     - *y* consults DV,
     - *y* selects “cheaper” route via *z* (cost 2+1 = 3),
     - Sends update to *z* and *x* (cost to *r* now 3 instead of 1)
  2. *z* detects cost increase for path to *r*:
     - was 1+1, is now 3+1
     - Sends update to *y* and *x* (cost to *r* now 4 instead of 2)
  3. *y* detects cost increase, sends update to *z*
  4. *z* detects cost increase, sends update to *y*
  5. ….



Distance Vector: Solutions that only half work

- Finite infinity: Define some number to be ∞ (in RIP: 16 := ∞)
- Split Horizon:
  - Tell to a neighbour that is part of a best path to a destination that the destination cannot be reached
  - If *z* routes through *y* to get to *r*
    z tells *y* that its own (i.e., *y*'s) distance to *r* is infinite (so *y* won't route to *r* via *z*)



- Poisoned Reverse:
  - In addition, *actively* advertise a route as unreachable to the neighbour from which the route was learned
- (Warning: Terms often used interchangeably!)
- Often help, but cannot solve all problem instances
- Can significantly increase number of routing messages

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

Chapter 4: Network Layer

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

- IPv6
- Virtual circuit and datagram networks
- What's inside a router

Part 3

- **Routing algorithms**
  - Link state
  - Distance Vector
  - **Path Vector**
  - Hierarchical routing
- Internet routing protocols
  - RIP
  - OSPF
  - BGP
- Business considerations
  - Policy routing
  - Traffic engineering

Path Vector protocols

- Problem with D-V protocol:
  Path cost is "anonymous" single number
- Path Vector protocol:
  - For each destination, advertise entire path (=sequence of node identifiers) to neighbours
  - Cost calculation can be done by looking at path
  - Easy loop detection: Does my node ID already appear in the path?
- Not used very often
  - only in BGP …
  - … and BGP is much more complex than just paths!

Chapter 4: Network Layer

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

- IPv6
- Virtual circuit and datagram networks
- What’s inside a router

Part 3

- **Routing algorithms**
  - Link state
  - Distance Vector
  - Path Vector
  - **Hierarchical routing**
- Internet routing protocols
  - RIP
  - OSPF
  - BGP
- Business considerations
  - Policy routing
  - Traffic engineering

Hierarchical Routing

Our routing study thus far = idealisation

- all routers identical
- network "flat"

… *not* true in practice!

Scale = billions of destinations:

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

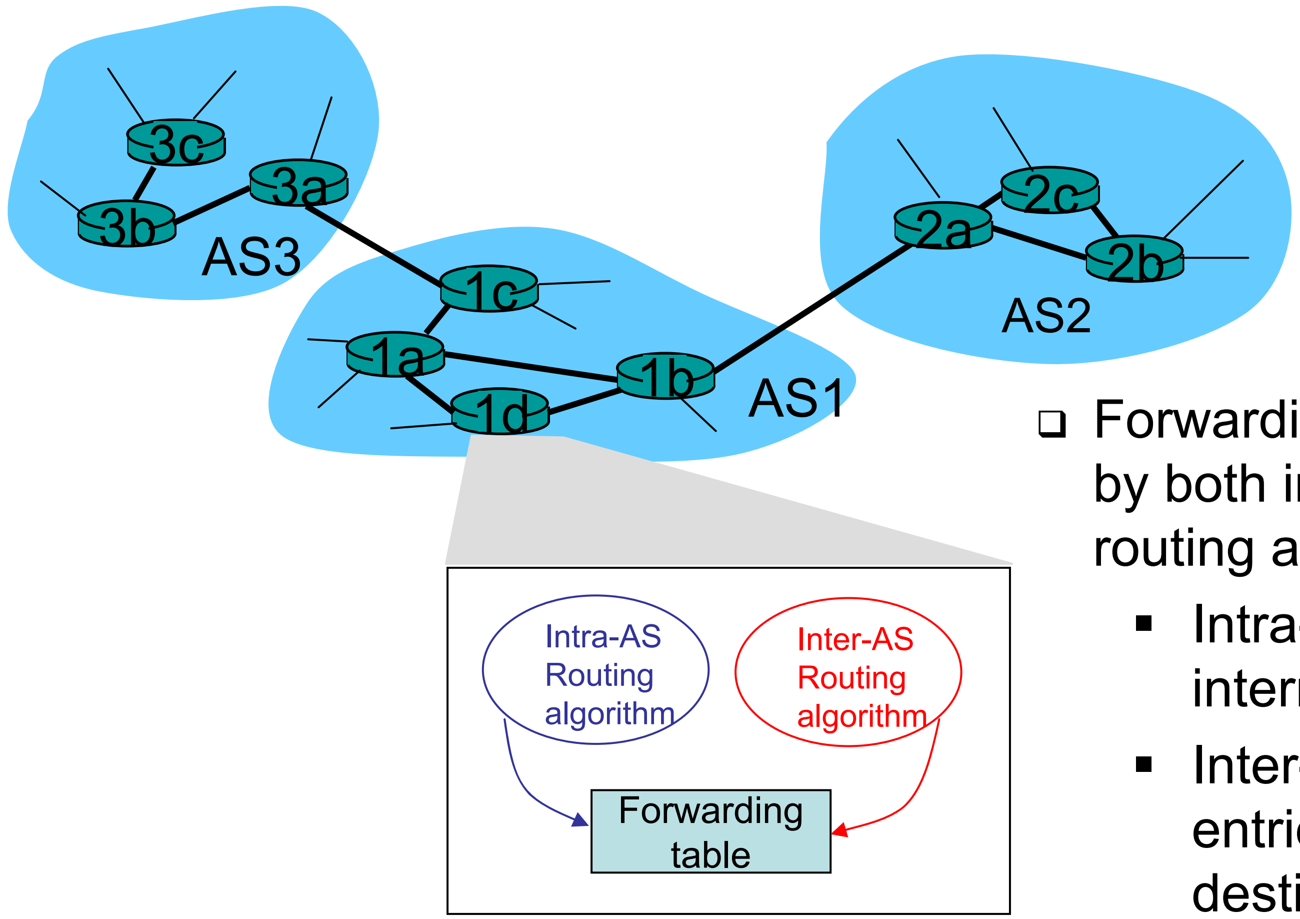
Administrative autonomy

- Internet = network of networks
- Each network admin may want to control routing in its own network — no central administration!

Hierarchical Routing

- Aggregate routers into regions called
  “autonomous systems” (short: AS; plural: ASes)
- Routers in same AS run same routing protocol
  - = “intra-AS” routing protocol (also called “intradomain”)
  - Routers in different ASes can run different intra-AS routing protocols
- ASes are connected: via gateway routers
  - Direct link to [gateway] router in another AS
    = “inter-AS” routing protocol (also called “interdomain”)
  - Warning: Non-gateway routers need to know about inter-AS routing as well!

Interconnected ASes



- Forwarding table configured by both intra- *and* inter-AS routing algorithm:
  - Intra-AS sets entries for internal destinations
  - Inter-AS *and* intra-AS set entries for external destinations

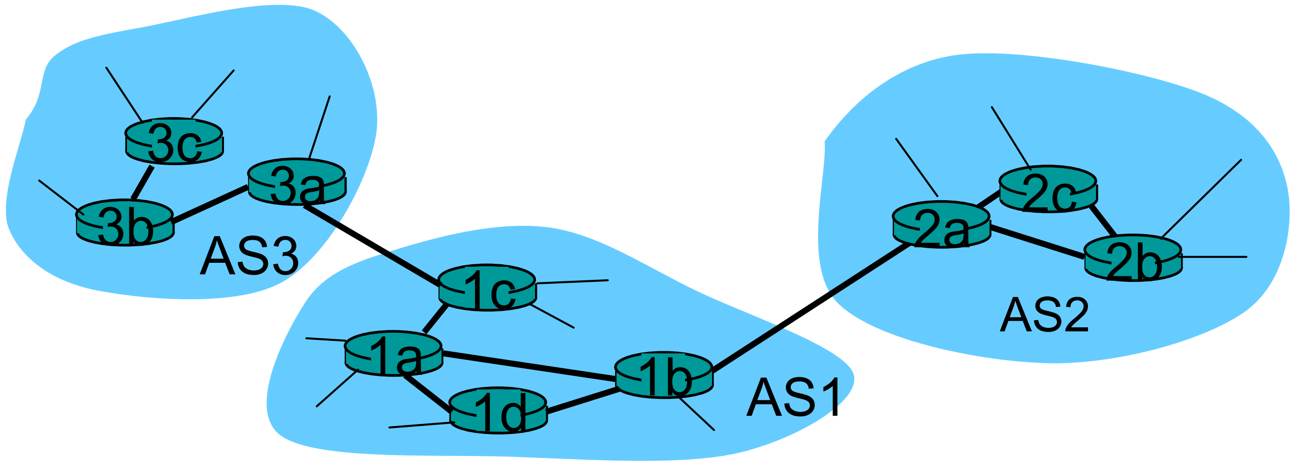
Inter-AS tasks

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

AS1 must:

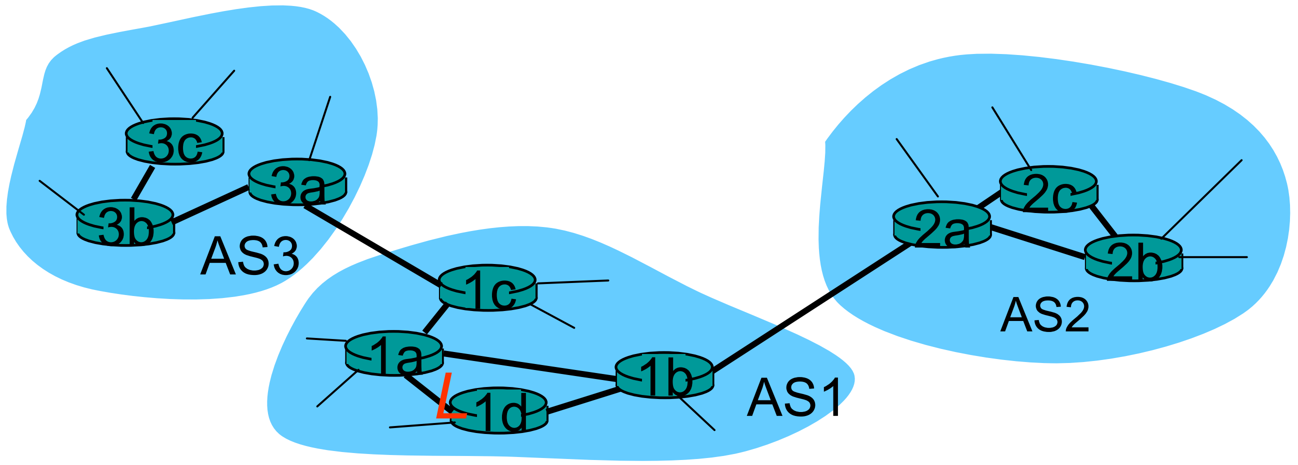
1. learn which dests are reachable through AS2, which through AS3
2. propagate this reachability info *to all* routers in AS1 (i.e., not just the gateway routers)

Job of inter-AS routing!



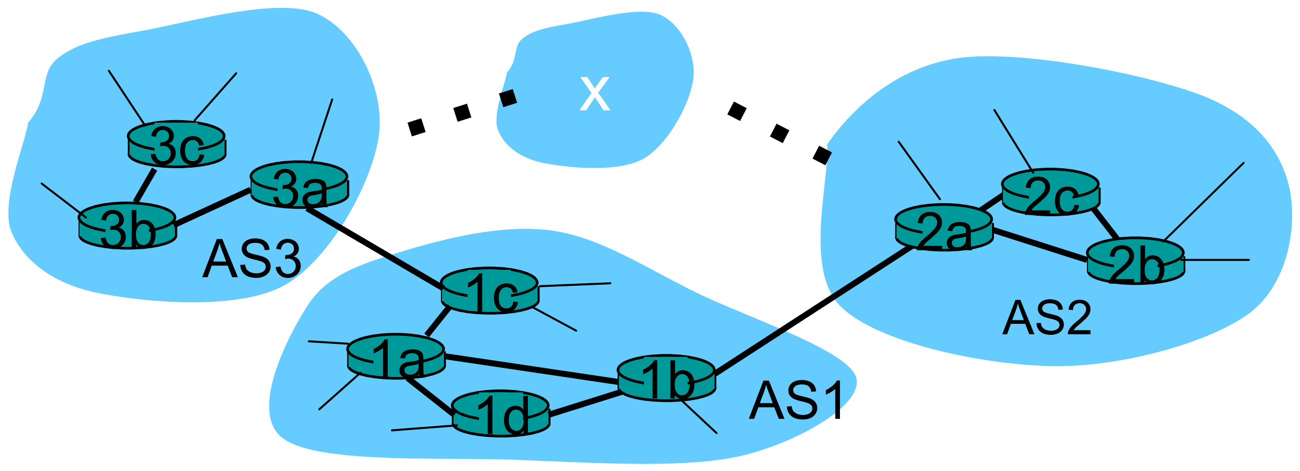
Example: Setting forwarding table in router 1d

- Suppose AS1 learns (via inter-AS protocol) that subnet *x* is reachable via AS3 (gateway 1c) but not via AS2.
- Inter-AS protocol propagates reachability info to all internal routers.
- Router 1d determines from intra-AS routing info that its interface *I* (i.e., interface to 1a) 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 towards which gateway it should forward packets for destination x.
  - This is also job of inter-AS routing protocol!



Interplay of inter-AS and intra-AS routing

- Inter-AS routing
  - Only for destinations outside of own AS
  - Used to determine gateway router
  - Also: Steers transit traffic
    (from AS *x* to AS *y* via our own AS)
- Intra-AS routing
  - Used for destinations within own AS
  - Used to reach gateway router for outside destinations

Chapter 4: Network Layer

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

- IPv6
- Virtual circuit and datagram networks
- What's inside a router

Part 3

- Routing algorithms
  - Link state
  - Distance Vector
  - Path Vector
  - Hierarchical routing
- **Internet routing protocols**
  - (RIP)
  - OSPF
  - BGP
- Business considerations
  - Policy routing
  - Traffic engineering

Intra-AS Routing

- Also known as Interior Gateway Protocols (IGP)
- Most common Intra-AS routing protocols:
  - RIP: Routing Information Protocol — DV (typically small systems)
  - OSPF: Open Shortest Path First — hierarchical LS (typically medium to large systems)
  - IS-IS: Intermediate System to Intermediate System — hierarchical LS (typically medium-sized ASes)
  - (E)IGRP: (Enhanced) Interior Gateway Routing Protocol (Cisco proprietary) — hybrid of LS and DV

Chapter 4: Network Layer

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

- IPv6
- Virtual circuit and datagram networks
- What’s inside a router

Part 3

- Routing algorithms
  - Link state
  - Distance Vector
  - Path Vector
  - Hierarchical routing
- **Internet routing protocols**
  - (RIP)
  - **OSPF**
  - BGP
- Business considerations
  - Policy routing
  - Traffic engineering

OSPF (Open Shortest Path First)

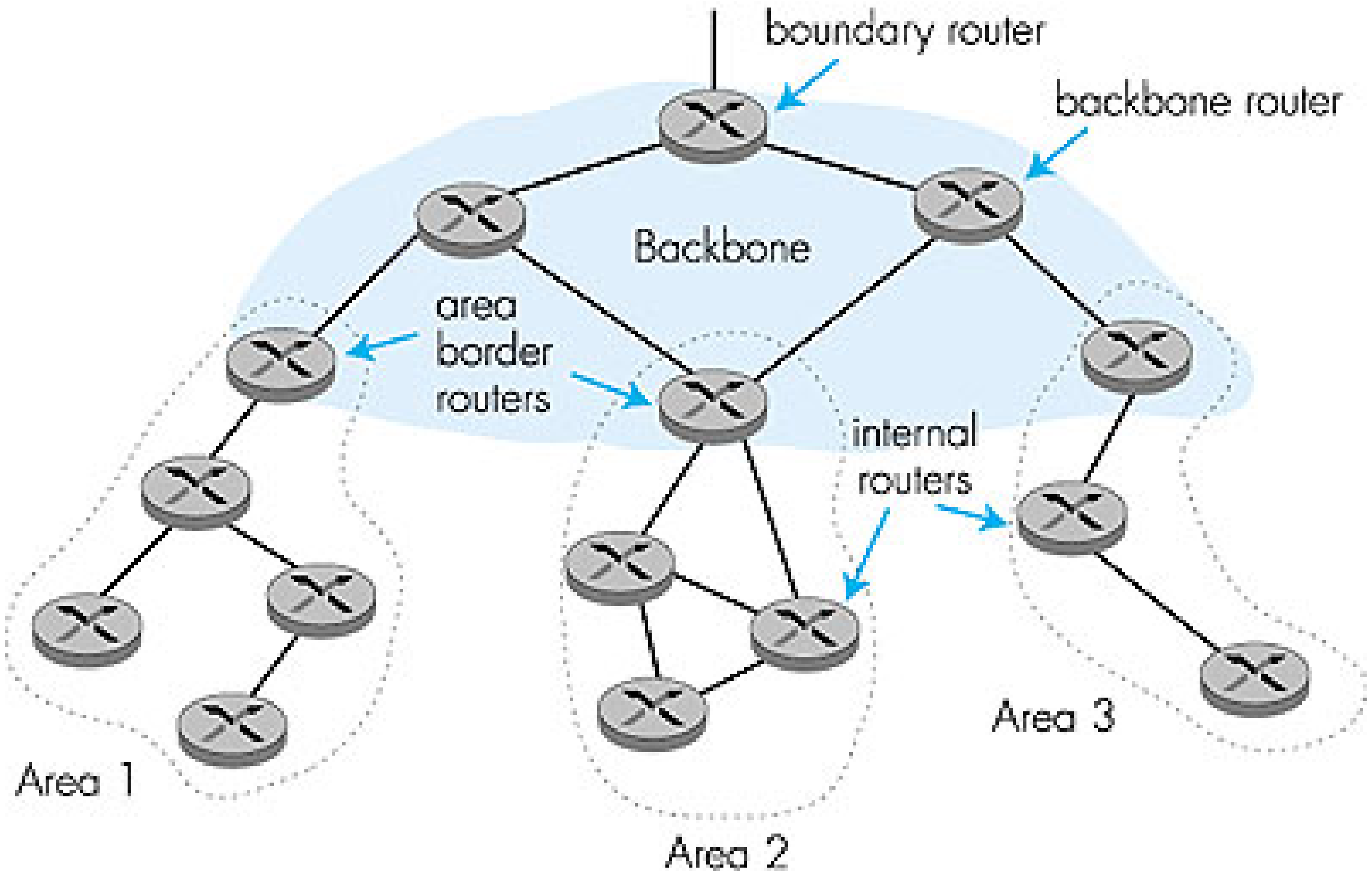
- “open”: publicly available
- uses Link State algorithm
  - LS packet dissemination
  - topology map at each node
  - route computation using Dijkstra’s algorithm

- OSPF advertisement carries one entry per neighbor router
- advertisements disseminated to entire AS (via flooding)
  - carried in OSPF messages directly over IP (rather than TCP or UDP

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; 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

- OSPF *can* create a two-level hierarchy similar to inter-AS and intra-AS routing within an AS
  - Two levels: local *areas* and the *backbone*
  - Link-state advertisements only within local area
  - Each node has detailed area topology; but only knows direction (shortest path) to networks in other areas
- *Area border routers:* "summarize" distances to networks in own area; advertise distances to other Area Border routers
- *Backbone routers:* run OSPF routing limited to backbone
- *Boundary routers:* connect to other ASes

Chapter 4: Network Layer

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

- IPv6
- Virtual circuit and datagram networks
- What's inside a router

Part 3

- Routing algorithms
  - Link state
  - Distance Vector
  - Path Vector
  - Hierarchical routing
- **Internet routing protocols**
  - RIP
  - OSPF
  - **BGP**
- Business considerations
  - Policy routing
  - Traffic engineering

Internet inter-AS routing: BGP

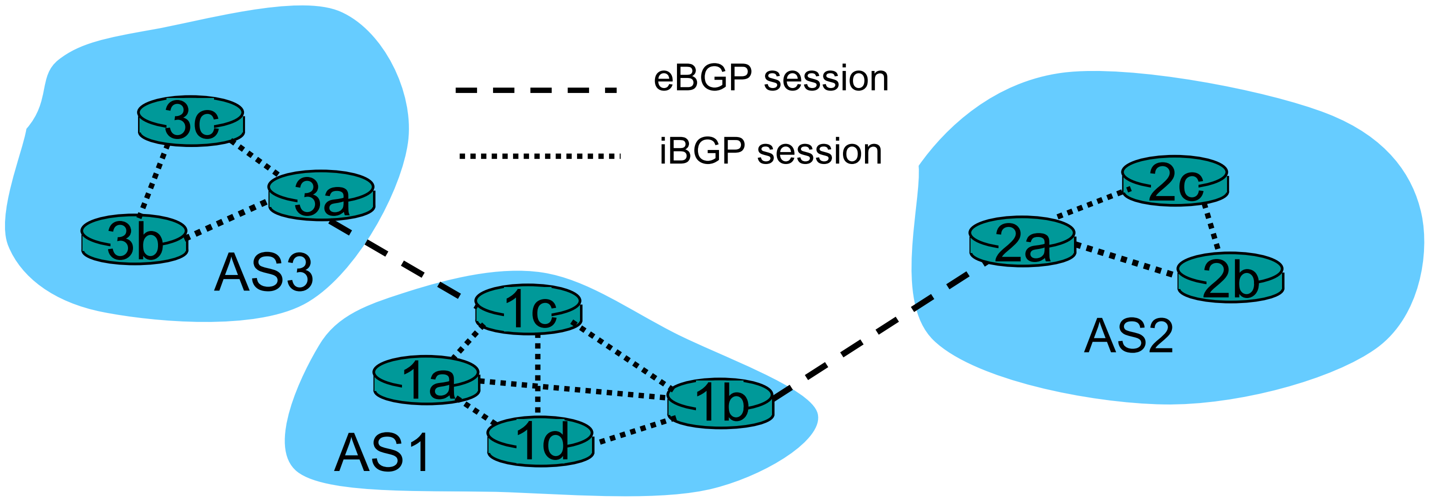
- BGP (Border Gateway Protocol):
  The de facto standard for inter-AS routing
- BGP provides each AS a means to:
  1. Obtain subnet reachability information from neighbouring ASes.
  2. Propagate reachability information to all AS-internal routers.
  3. Determine “good” routes to subnets based on reachability information and policy.
- Allows an AS to advertise the existence of an IP prefix to rest of Internet: *“This subnet is here”*

BGP basics

- Pairs of routers (BGP peers) exchange routing info over semi-permanent TCP connections: BGP sessions
  - BGP sessions need not correspond to physical links!
- When AS2 advertises an IP prefix to AS1:
  - AS2 *promises* it will forward IP packets towards that prefix
  - AS2 can aggregate prefixes in its advertisement

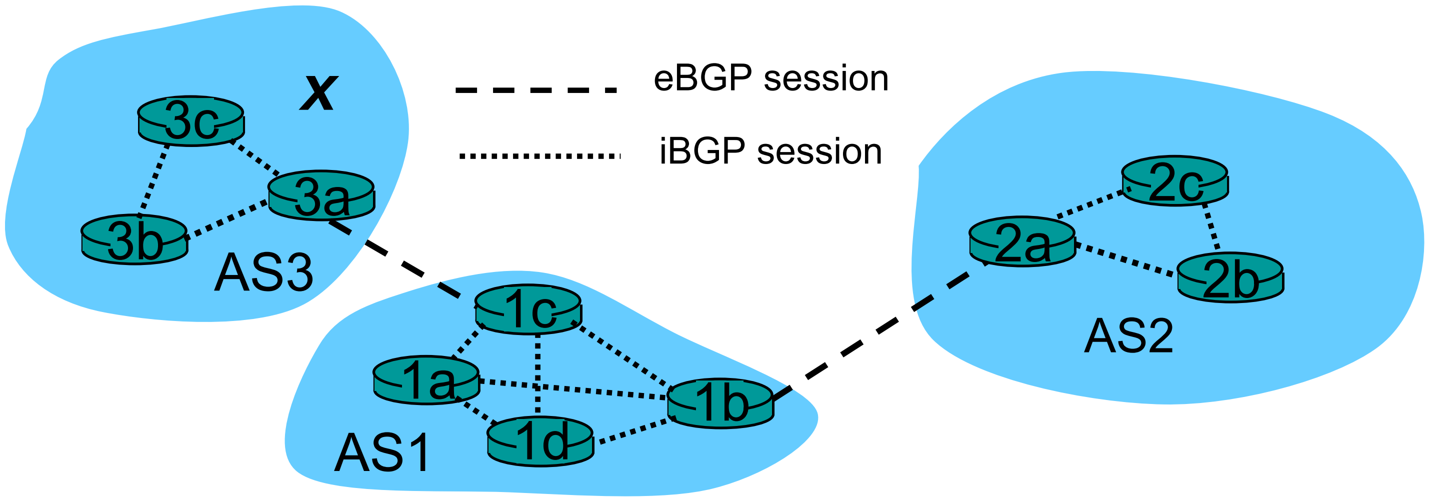
eBGP and iBGP

- External BGP: between routers in *different* ASes
- Internal BGP: between routers in *same* AS
  - Remember: In spite of intra-AS routing protocol, *all* routers need to know about external destinations (not only border routers)
- No different protocols — just slightly different configurations!



Distributing reachability info

- Using eBGP session between 3a and 1c, AS3 sends reachability info about prefix *x* to AS1.
  - 1c can then use iBGP to 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 *x*, it creates entry for prefix in its forwarding table.



Slides subject to change after this point until Monday!

Path attributes & 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)
- when gateway router receives route advertisement, uses import policy to accept/decline.

BGP route selection

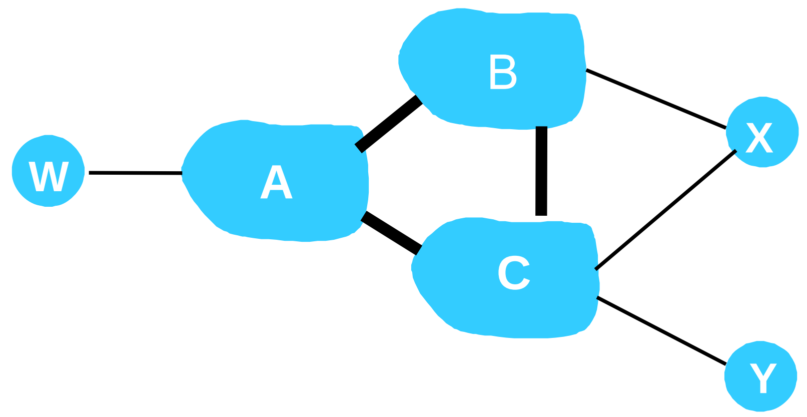
- router may learn about more than 1 route to some prefix. Router must select route.
- elimination rules:

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 using TCP.
- 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

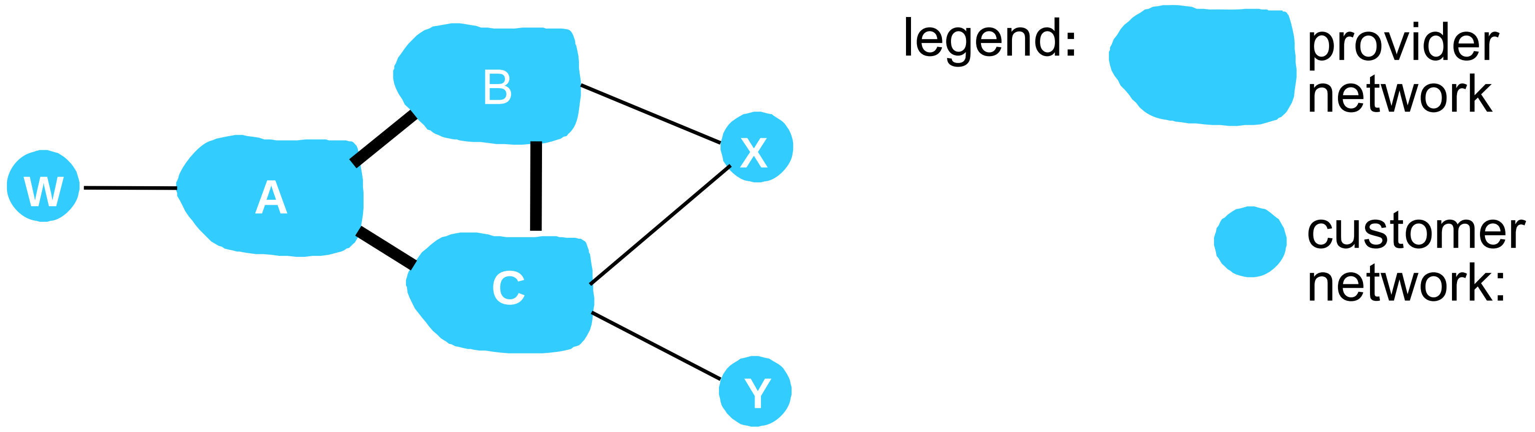


legend: ● provider network

● customer network:

- 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- 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

Chapter 4: Network Layer

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

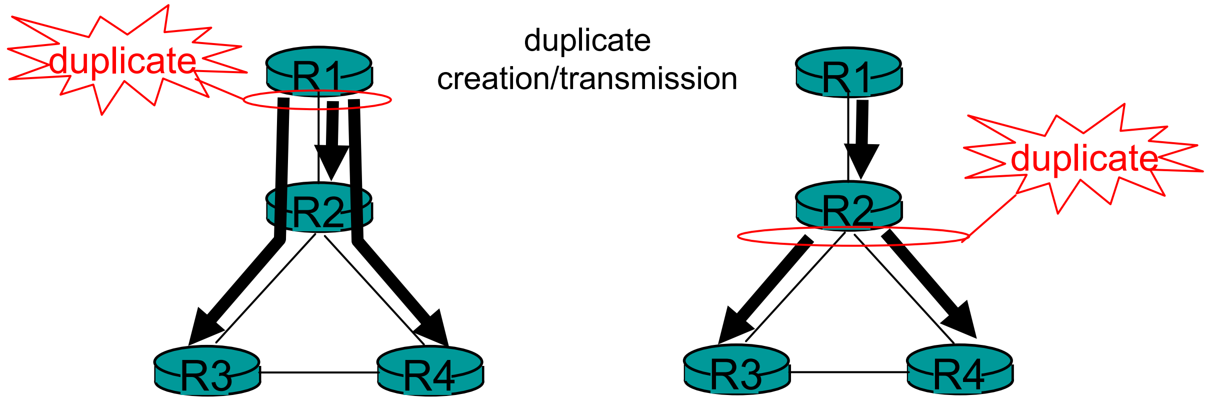
- IPv6
- Virtual circuit and datagram networks
- What's inside a router

Part 3

- Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- Routing in the Internet
  - RIP
  - OSPF
  - BGP
- **Broadcast and multicast routing**

Broadcast Routing

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



source duplication

in-network duplication

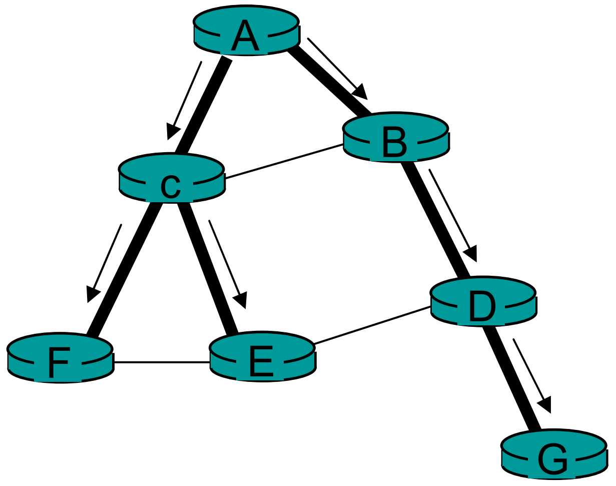
- source duplication: how does source determine recipient addresses?

In-network duplication

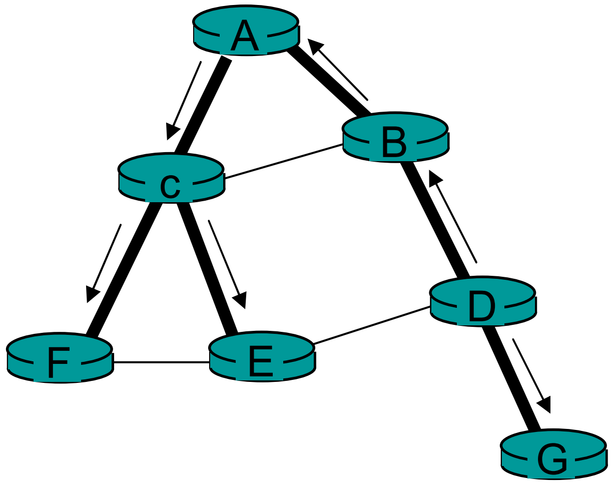
- flooding: when node receives brdcst pckt, sends copy to all neighbors
  - Problems: cycles & broadcast storm
- controlled flooding: node only brdcsts pkt if it hasn’t brdcst same packet before
  - Node keeps track of pckt ids already brdcsted
  - Or reverse path forwarding (RPF): only forward pckt 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



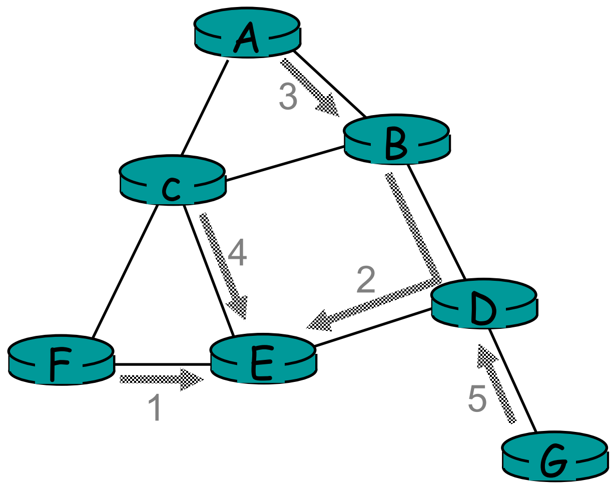
(a) Broadcast initiated at A



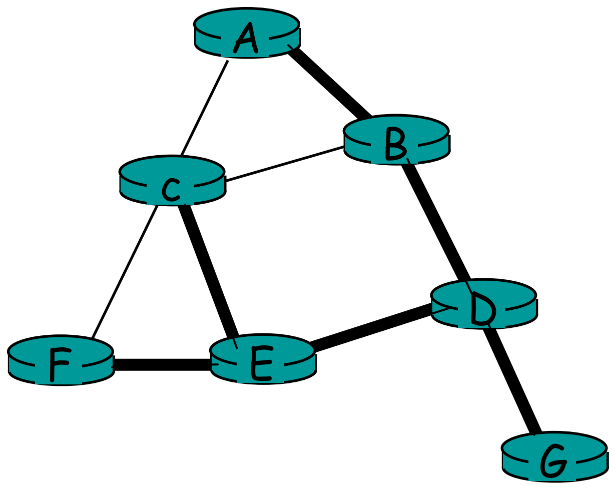
(b) Broadcast initiated at D

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



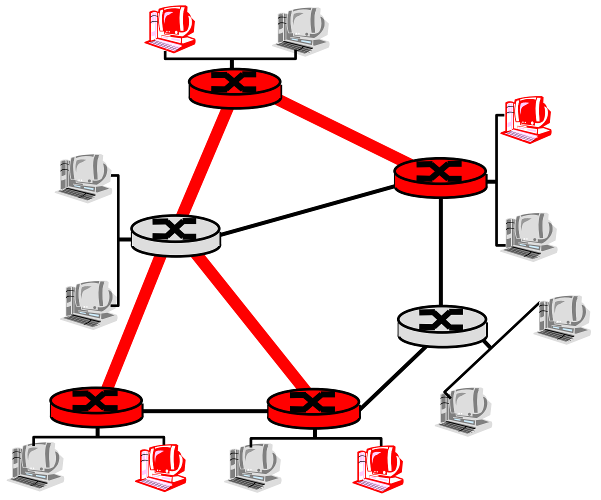
(a) Stepwise construction of spanning tree

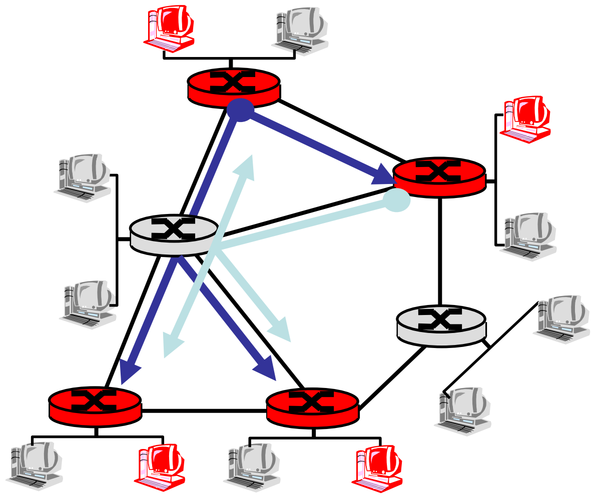


(b) Constructed spanning tree

Multicast Routing: Problem Statement

- ***Goal:*** find a tree (or trees) connecting routers having local mcast group members
  - *tree:* not all paths between routers used
  - *source-based:* different tree from each sender to rcvrs
  - *shared-tree:* same tree used by all group members


Shared tree


Source-based trees

Approaches for building mcast trees

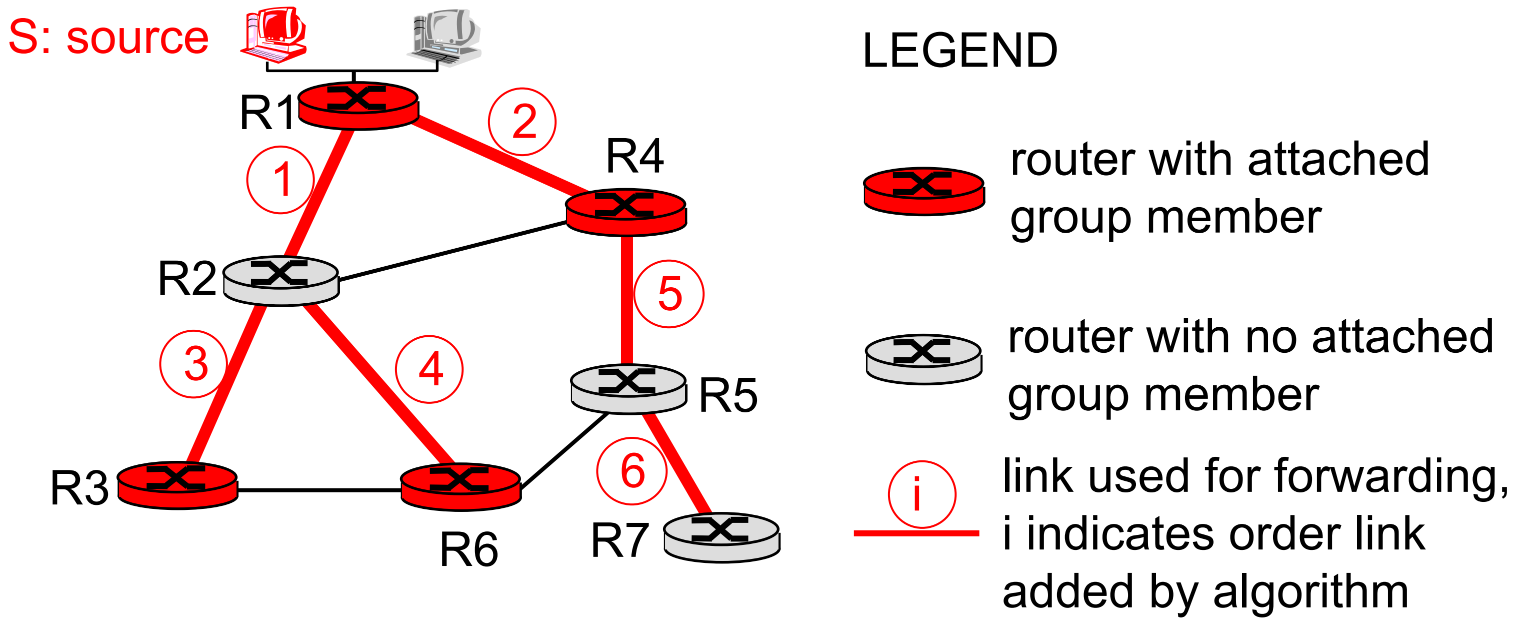
Approaches:

- 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

…we first look at basic approaches, then specific protocols adopting these approaches

Shortest Path Tree

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



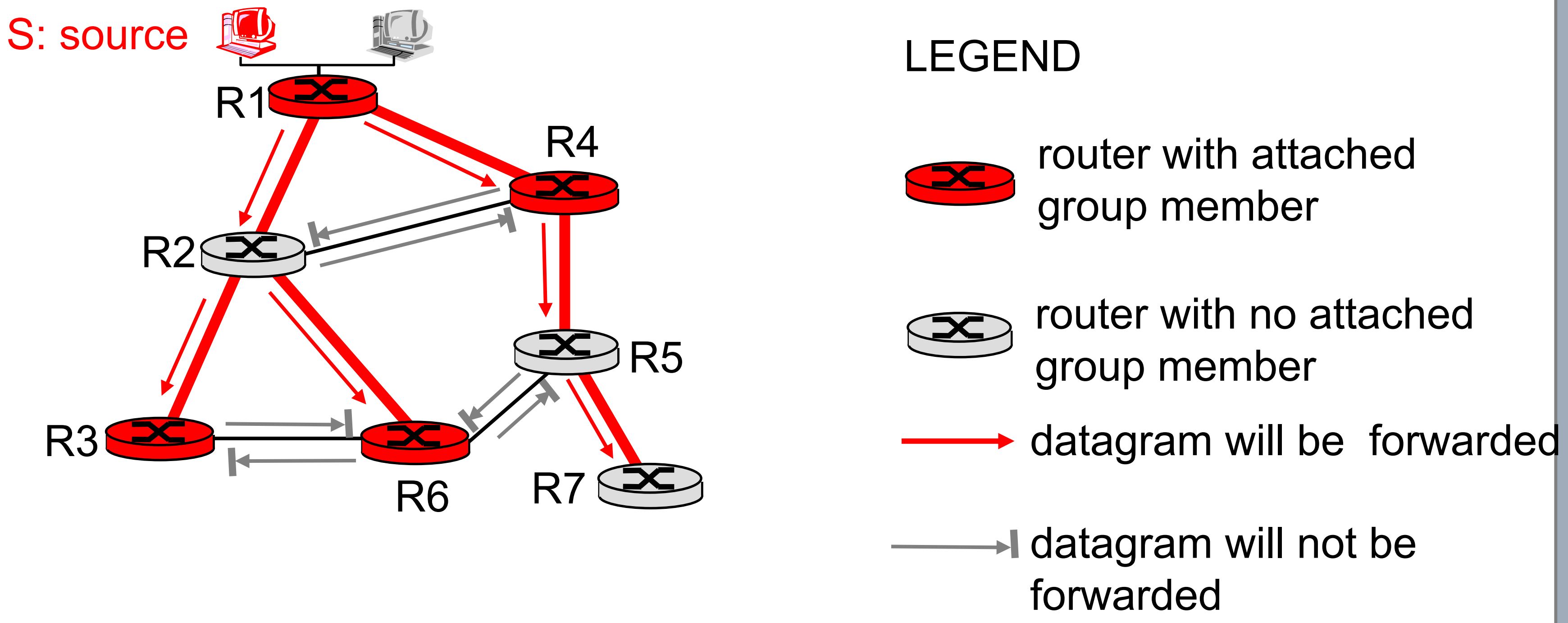
S: source
R1
R2
R3
R4
R5
R6
R7
1
2
3
4
5
6
LEGEND
router with attached group member
router with no attached group member
i
link used for forwarding, i indicates order link added by algorithm

Reverse Path Forwarding

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

```
if (mcast datagram received on incoming link on shortest path
    back to center)
  then flood datagram onto all outgoing links
  else ignore datagram
```

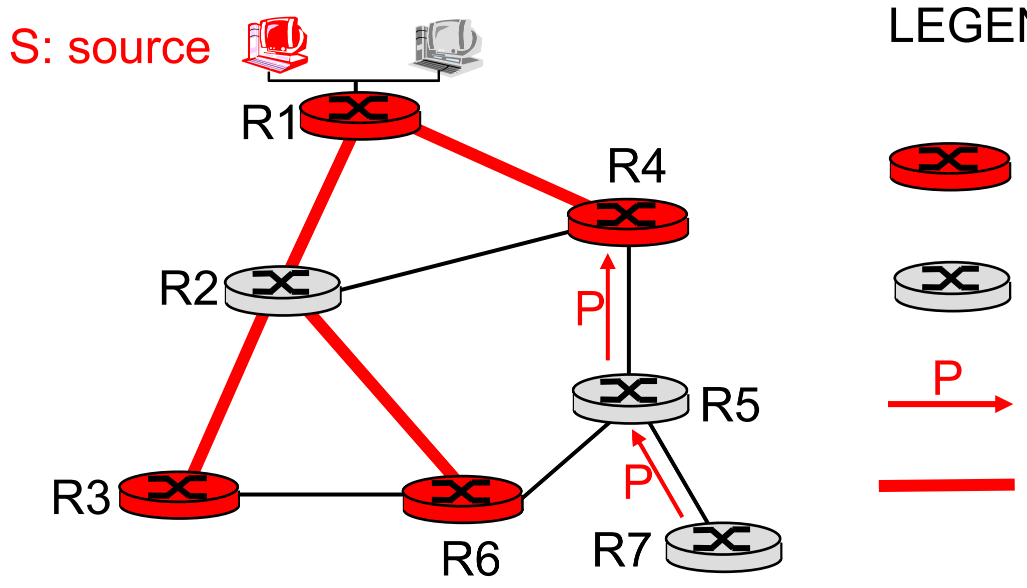
Reverse Path Forwarding: example


S: source
R1
R2
R3
R4
R5
R6
R7
LEGEND
router with attached group member
router with no attached group member
datagram will be forwarded
datagram will not be forwarded

- 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" msgs sent upstream by router with no downstream group members


S: source
R1
R2
R3
R4
R5
R6
R7
P
P
LEGEND
router with attached group member
router with no attached group member
P
prune message
links with multicast forwarding

Shared-Tree: Steiner Tree

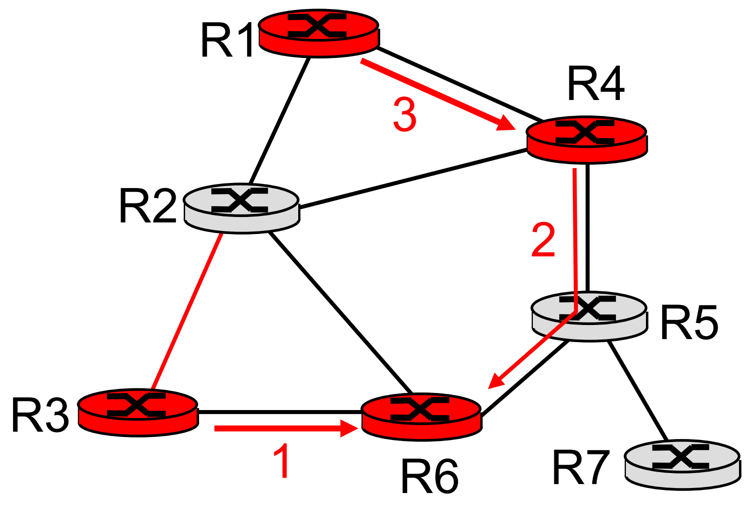
- Steiner Tree: minimum cost tree connecting all routers with attached group members
- problem is NP-complete
- excellent heuristics exists
- not used in practice:
  - computational complexity
  - information about entire network needed
  - monolithic: rerun whenever a router needs to join/leave

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:



LEGEND

🟥	router with attached group member
⬜	router with no attached group member
1 →	path order in which join messages generated

Internet Multicasting Routing: DVMRP

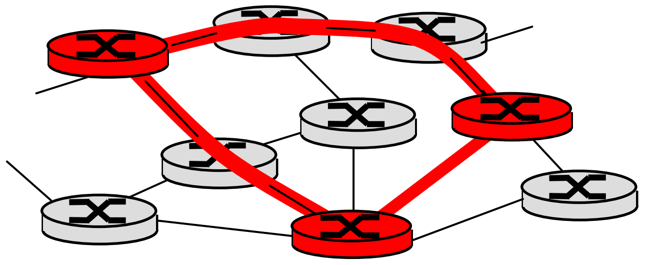
- DVMRP: distance vector multicast routing protocol, RFC1075
- *flood and prune:* reverse path forwarding, source-based tree
  - RPF tree based on DVMRP's own routing tables constructed by communicating DVMRP routers
  - no assumptions about underlying unicast
  - initial datagram to mcast group flooded everywhere via RPF
  - routers not wanting group: send upstream prune msgs

DVMRP: continued…

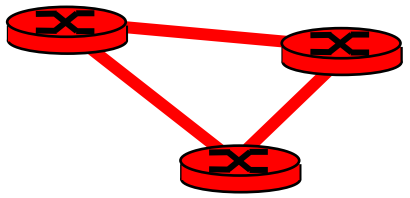
- *soft state:* DVMRP router periodically (1 min.) “forgets” branches are pruned:
    - mcast data again flows down unpruned branch
    - downstream router: reprune or else continue to receive data
- routers can quickly regraft to tree
    - following IGMP join at leaf
- odds and ends
    - commonly implemented in commercial routers
    - Mbone routing done using DVMRP

Tunneling

Q: How to connect "islands" of multicast routers in a "sea" of unicast routers?



physical topology



logical topology

- mcast datagram encapsulated inside "normal" (non-multicast-addressed) datagram
- normal IP datagram sent thru "tunnel" via regular IP unicast to receiving mcast router
- receiving mcast router unencapsulates to get mcast datagram

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.
- bandwidth more plentiful

Sparse:

- # networks with group members small wrt # interconnected networks
- group members “widely dispersed”
- bandwidth not plentiful

Consequences of Sparse-Dense Dichotomy:

Dense

- group membership by routers *assumed* until routers explicitly prune
- *data-driven* construction on mcast tree (e.g., RPF)
- bandwidth and non-group-router processing *profligate*

Sparse:

- no membership until routers explicitly join
- *receiver- driven* construction of mcast tree (e.g., center-based)
- bandwidth and non-group-router processing *conservative*

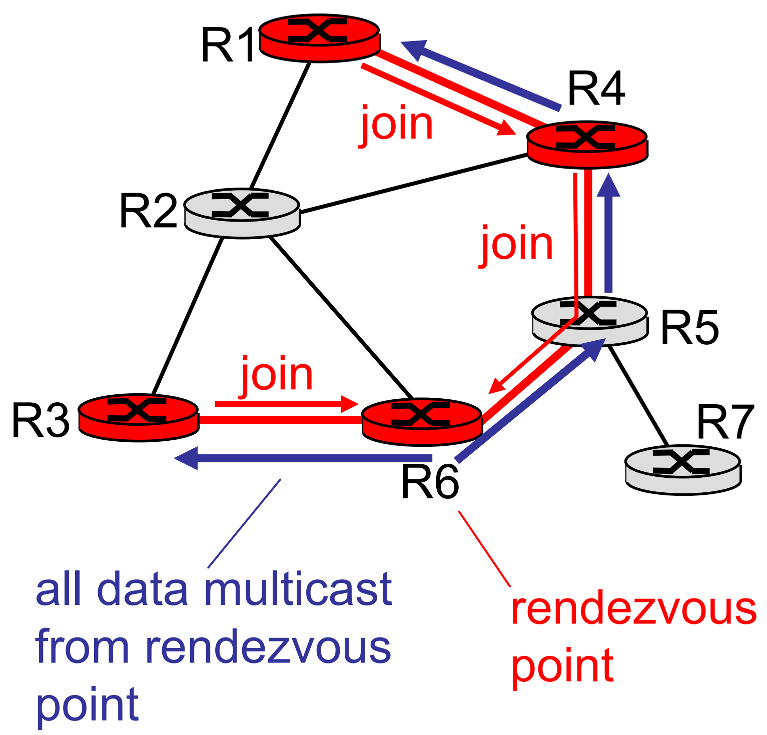
PIM- Dense Mode

flood-and-prune RPF, similar to DVMRP but

- underlying unicast protocol provides RPF info for incoming datagram
- less complicated (less efficient) downstream flood than DVMRP reduces reliance on underlying routing algorithm
- has protocol mechanism for router to detect it is a leaf-node router

PIM - Sparse Mode

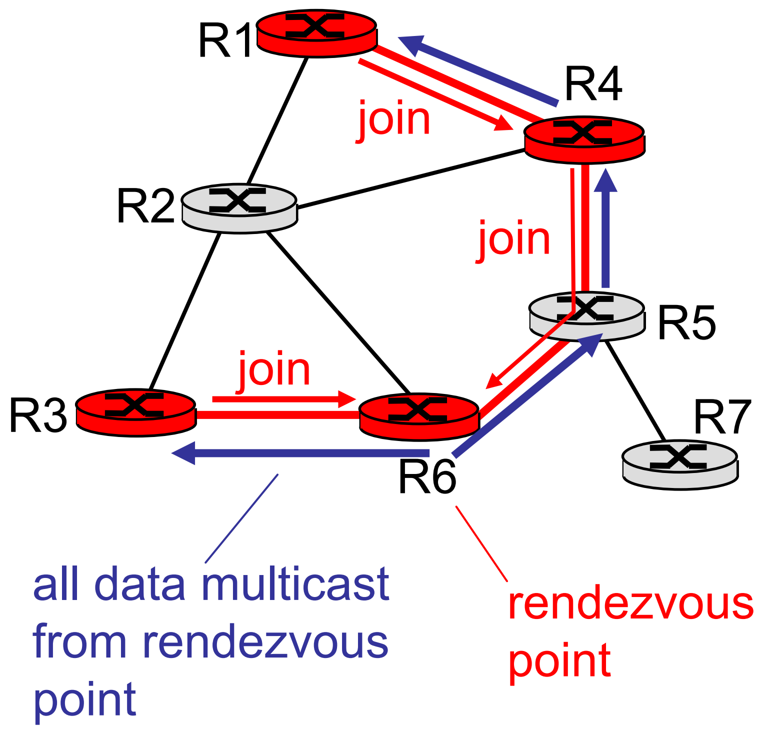
- center-based approach
- router sends *join* msg to rendezvous point (RP)
  - intermediate routers update state and forward *join*
- after joining via RP, router can switch to source-specific tree
  - increased performance: less concentration, shorter paths



PIM - Sparse Mode

sender(s):

- unicast data to RP, which distributes down RP-rooted tree
- RP can extend mcast tree upstream to source
- RP can send *stop* msg if no attached receivers
    - "no one is listening!"



Chapter 4: Network Layer

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

- IPv6
- Virtual circuit and datagram networks
- What's inside a router

Part 3

- Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- Routing in the Internet
  - RIP
  - OSPF
  - BGP
- Broadcast and multicast routing