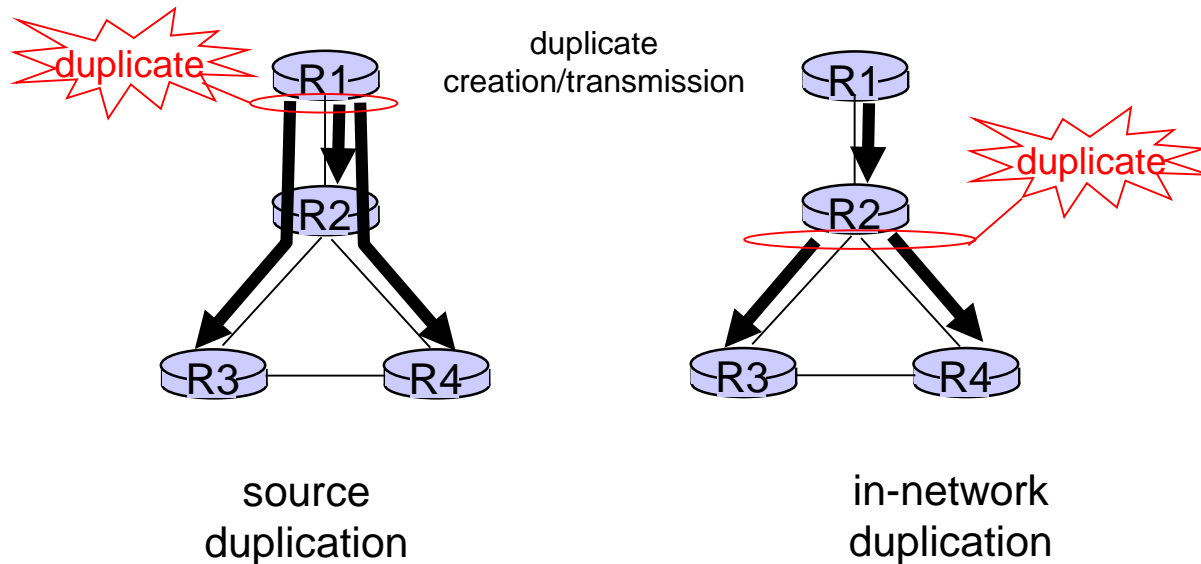# CS 280:
Network Layer:

Broadcast and Multicast Routing

John Magee
21 November 2016

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 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
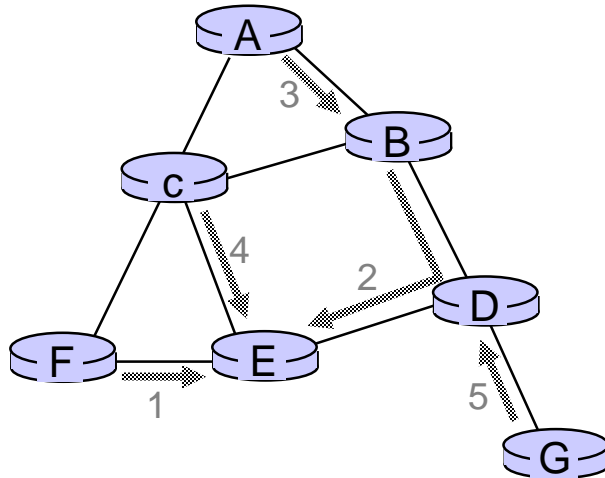❖ nodes then forward/make copies only along spanning tree
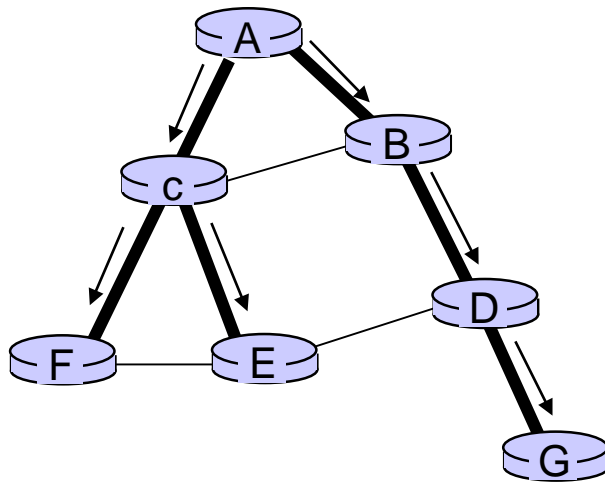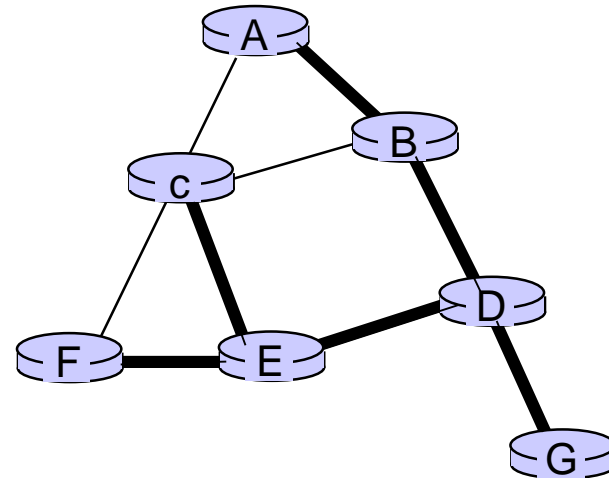

(a) broadcast initiated at A
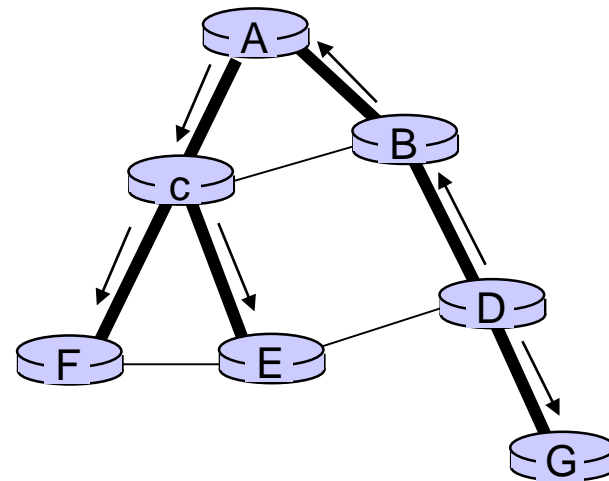
(b) broadcast initiated at D

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 routing: problem statement

goal: find a tree (or trees) connecting routers having local mcast group members

❖ *tree:* not all paths between routers used

❖ *shared-tree:* same tree used by all group members

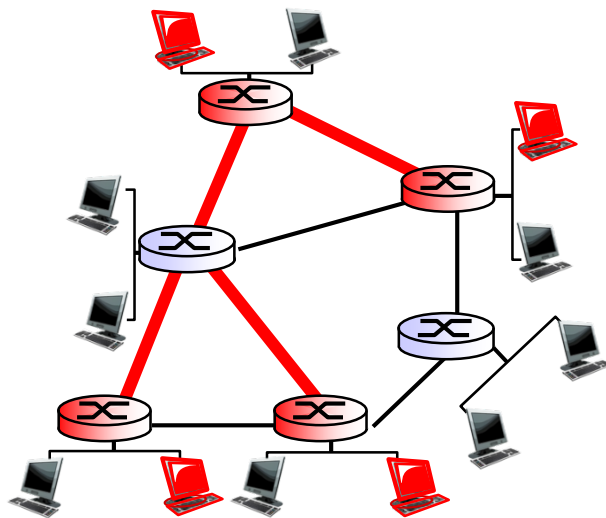❖ *source-based:* different tree from each sender to rcvrs

shared tree

source-based trees

legend

group member

not group member
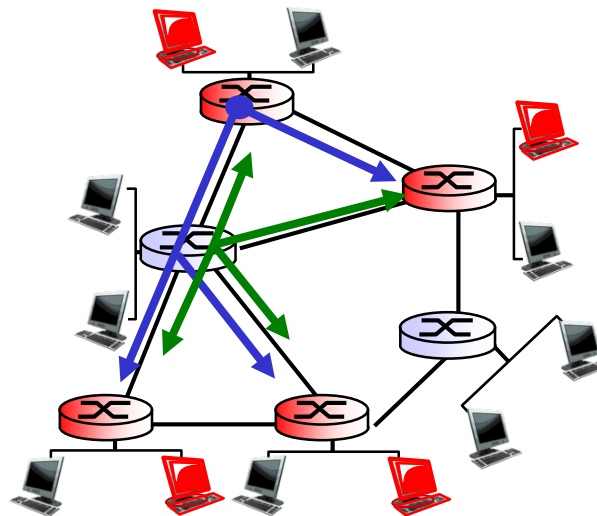
router with a group member

router without group member

Approaches for building mcast trees

approaches:

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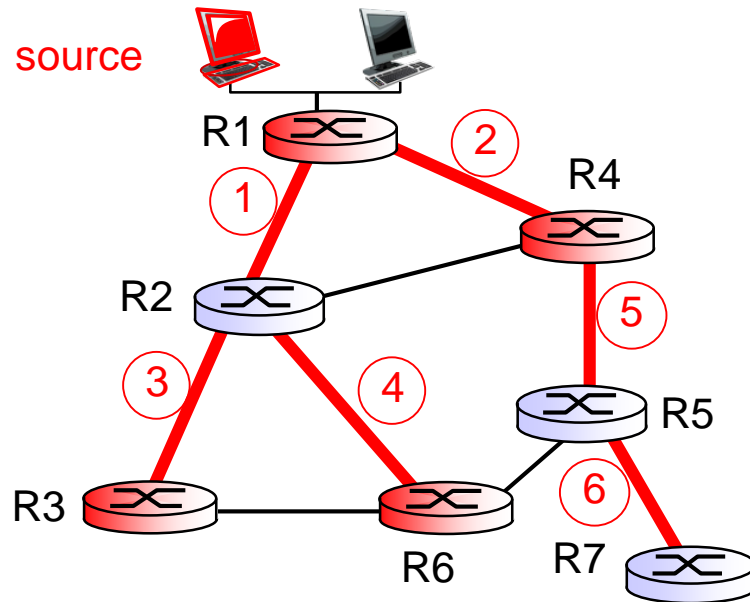
❖ *group-shared tree:* group uses one tree
- minimal spanning (Steiner)
- center-based trees

…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

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

LEGEND

router with attached group member

router with no attached group member

⟶ datagram will be forwarded

⟶ datagram will not be forwarded
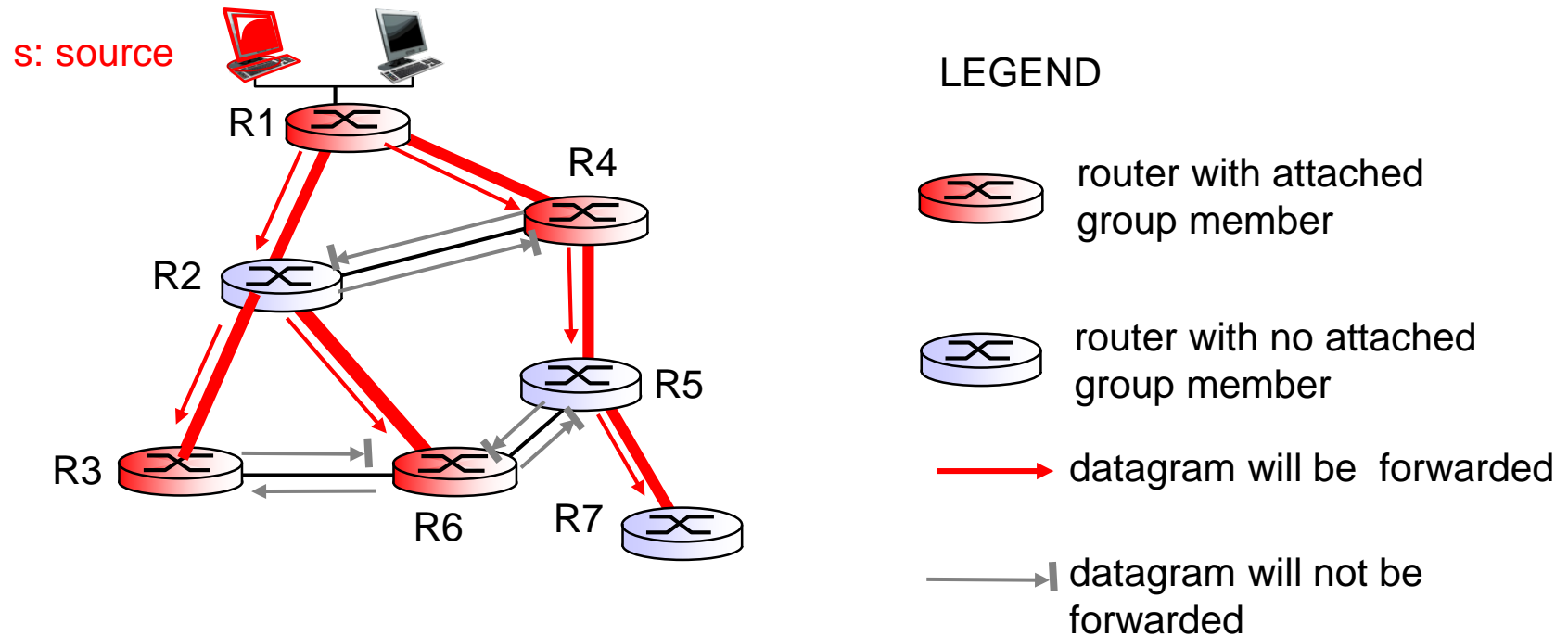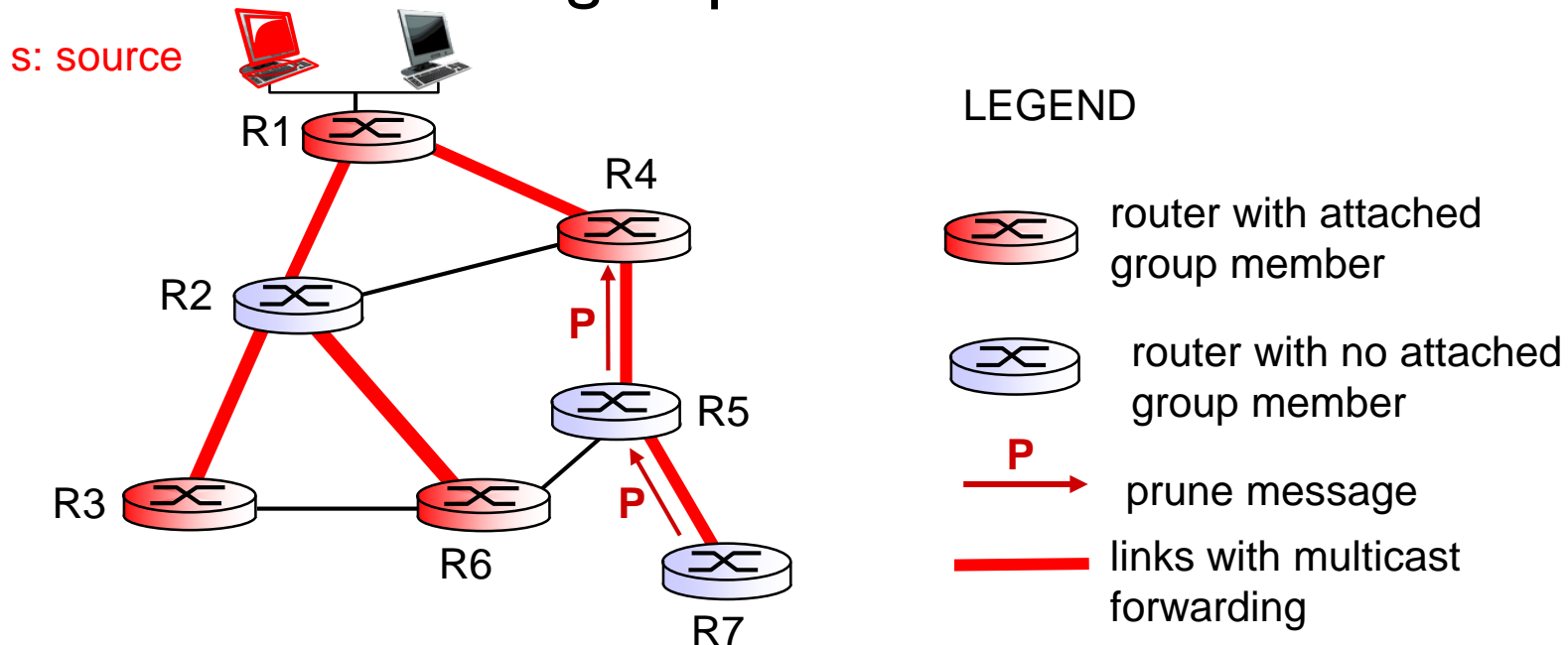
❖ result is a source-specific *reverse* SPT
  ▪ may be a bad choice with asymmetric links

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

R4

R2

R3

R6

R5

R7

P

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: example

suppose R6 chosen as center:
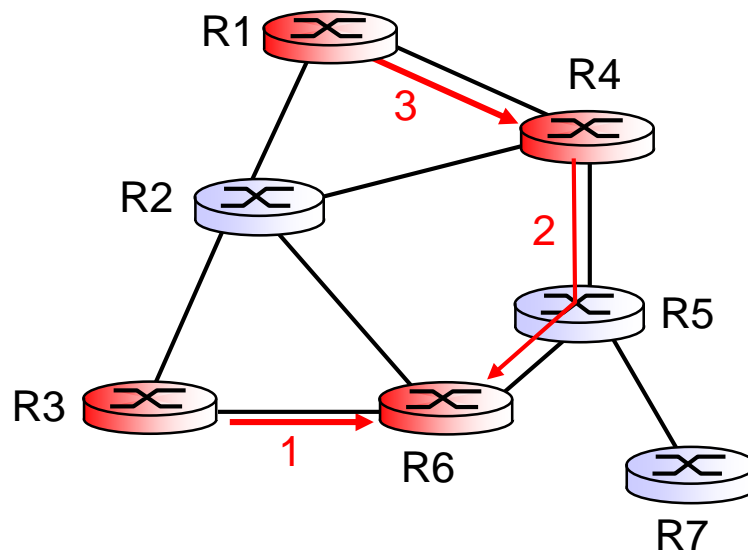
LEGEND

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 router

Tunneling

Q: how to connect "islands" of multicast routers in a "sea" of unicast routers?
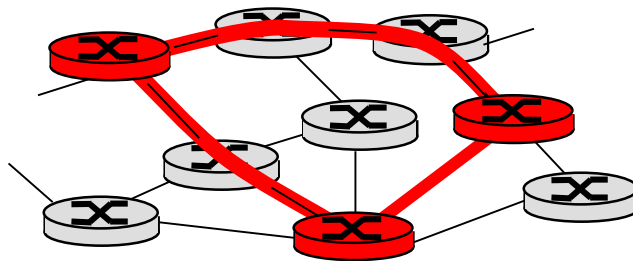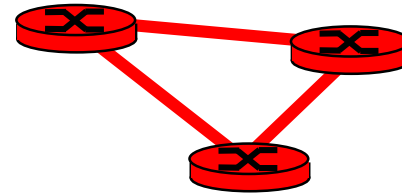
physical topology   logical topology

❖ mcast datagram encapsulated inside "normal" (non-multicast-addressed) datagram

❖ normal IP datagram sent thru "tunnel" via regular IP unicast to receiving mcast router (recall IPv6 inside IPv4 tunneling)

❖ 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

all data multicast from rendezvous point

rendezvous point
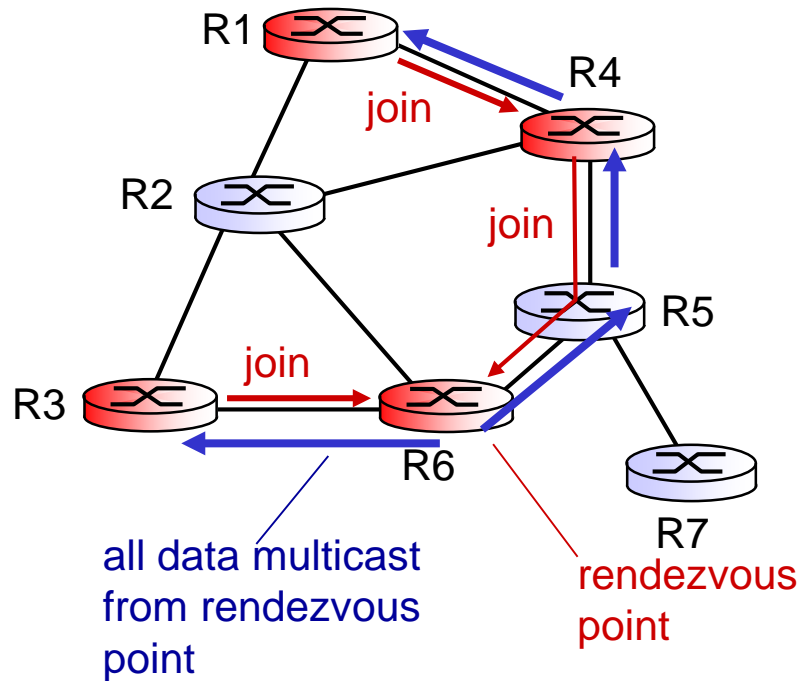
PIM - sparse mode

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!"

R1

R4

R2

R3

R5

R6

R7

join

join

join

all data multicast from rendezvous point

rendezvous point