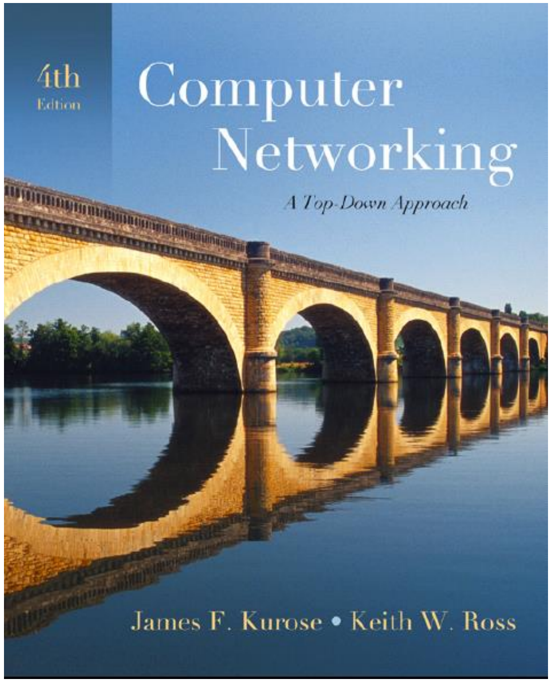
# Multicast



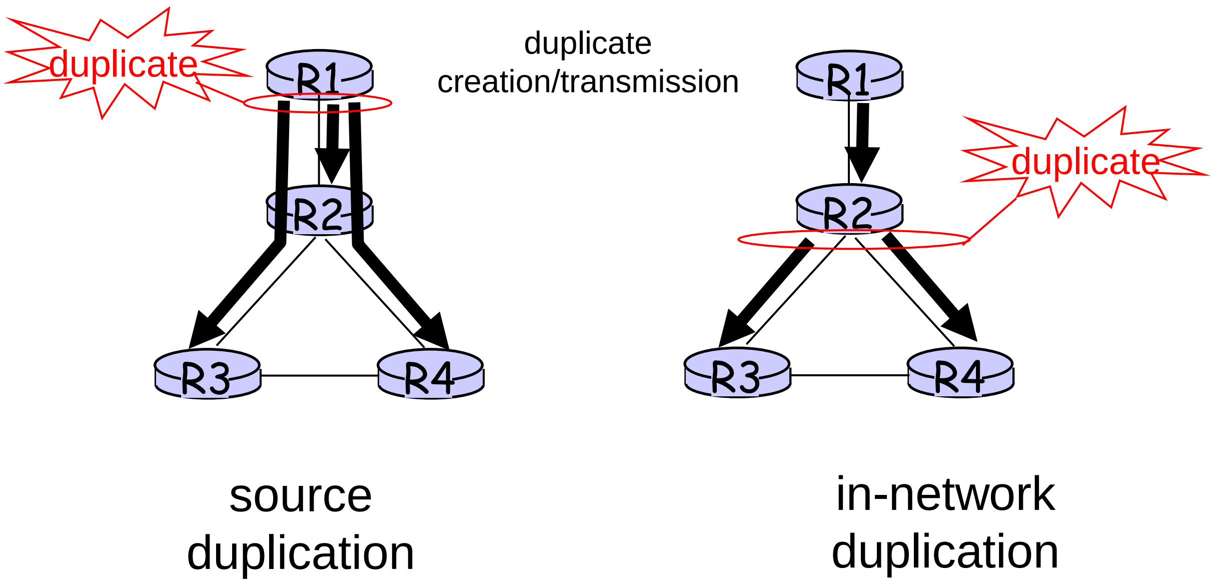

A note on the use of these ppt slides:

The notes used in this course are substantially based on powerpoint slides developed and copyrighted by J.F. Kurose and K.W. Ross, 2007

*Computer Networking: A Top Down Approach*
4th edition.
Jim Kurose, Keith Ross
Addison-Wesley, July 2007.

# Broadcast Routing

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




- Source duplication: how does source determine recipient addresses?

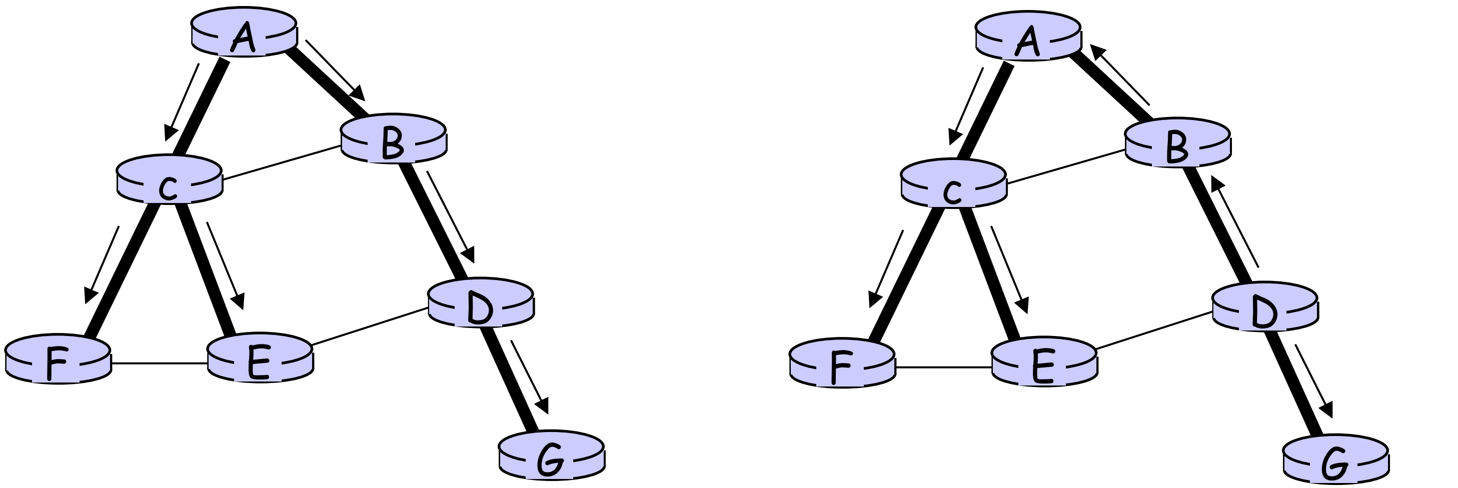
# In-network Duplication

- Flooding: when node receives 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

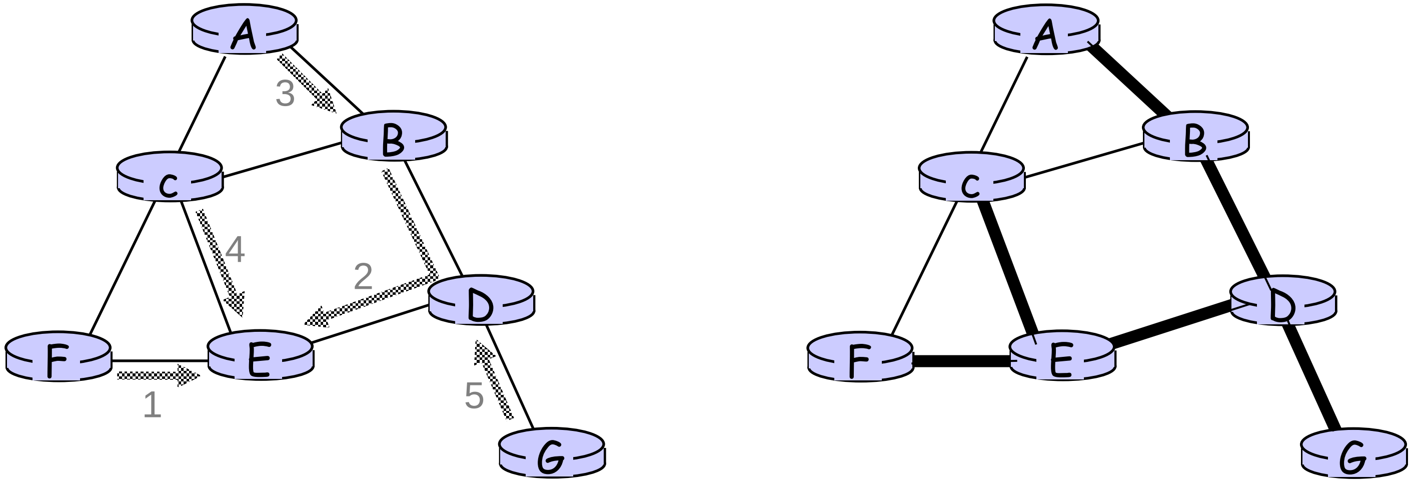



(a) Broadcast initiated at A

(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

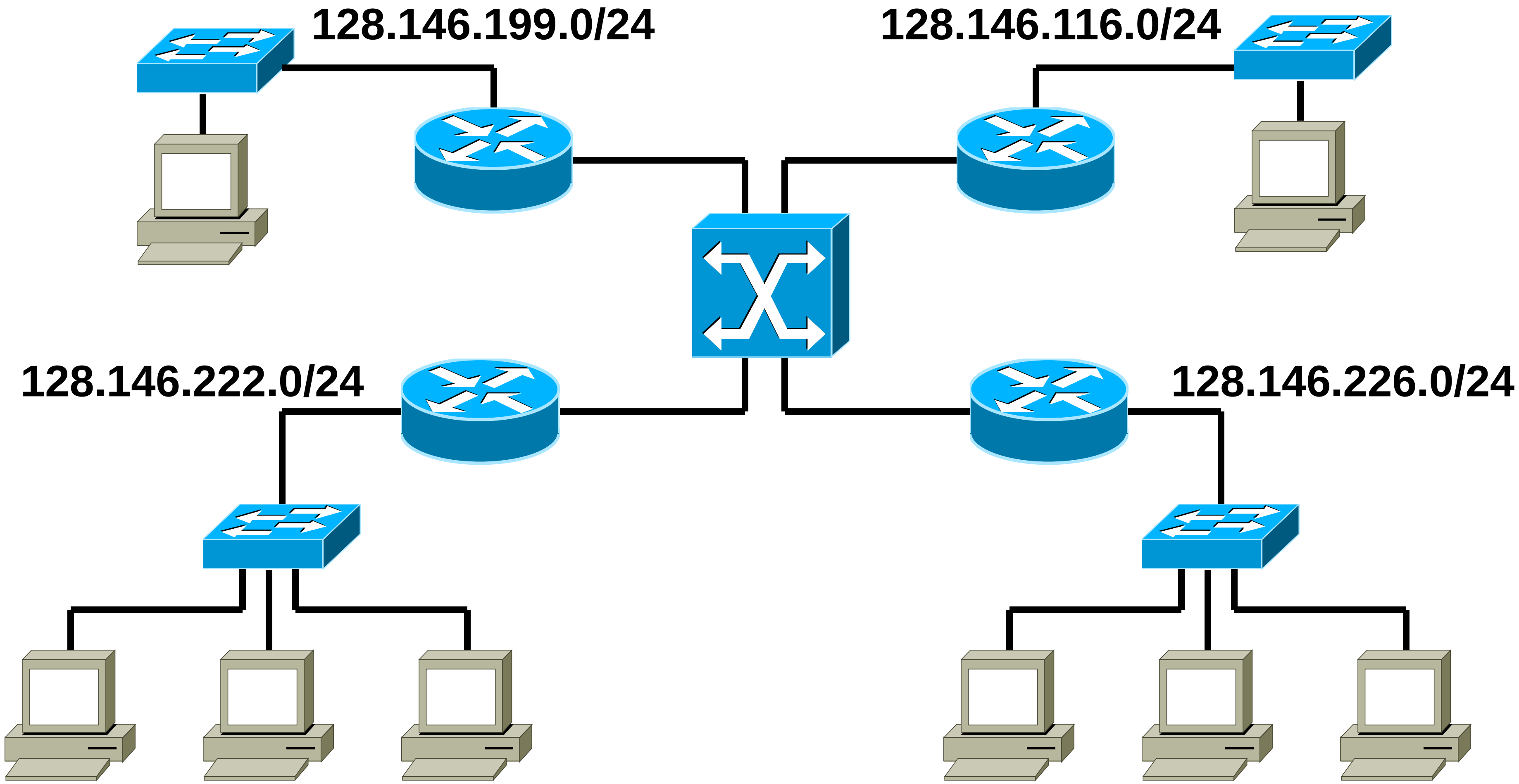



**(a) Stepwise construction of spanning tree** **(b) Constructed spanning tree**

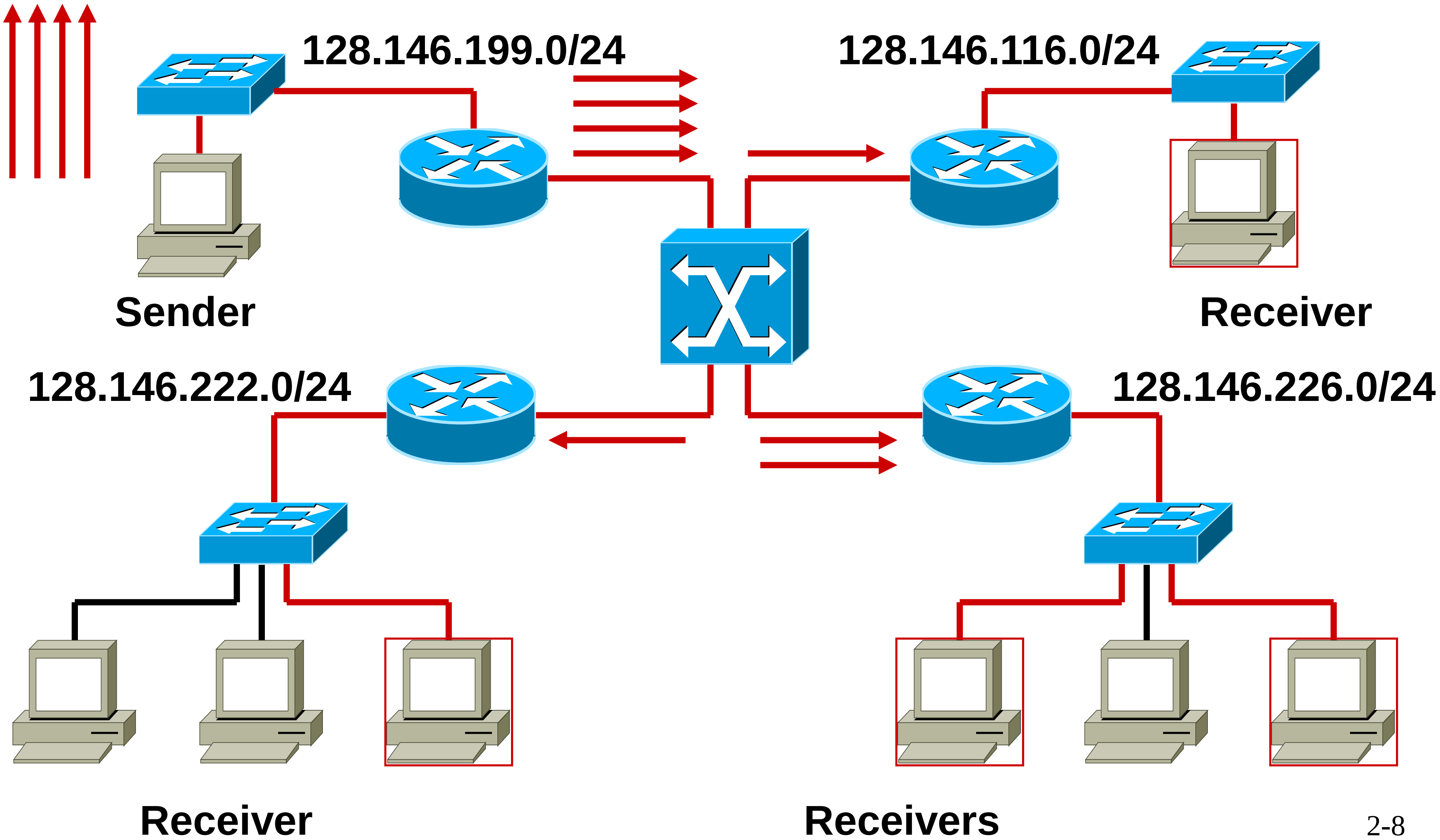
# Why Multicast

- When sending same data to multiple receivers
  - Better bandwidth utilization
  - Less host/router processing
  - Quicker participation
- Application
  - Video/Audio broadcast (One sender)
  - Video conferencing (Many senders)
  - Real time news distribution
  - Interactive gaming

# Unicast/Multicast

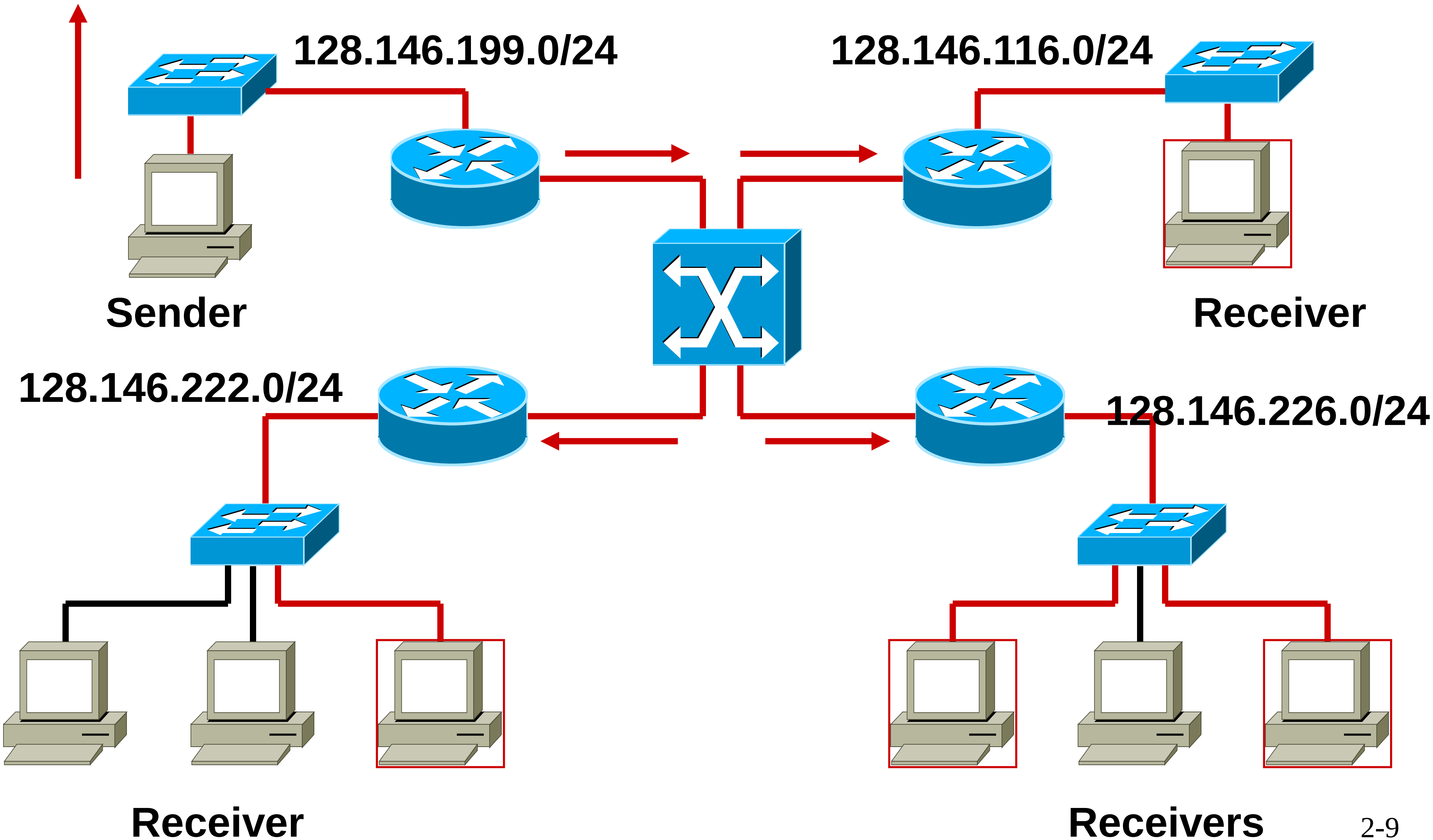


# Unicast



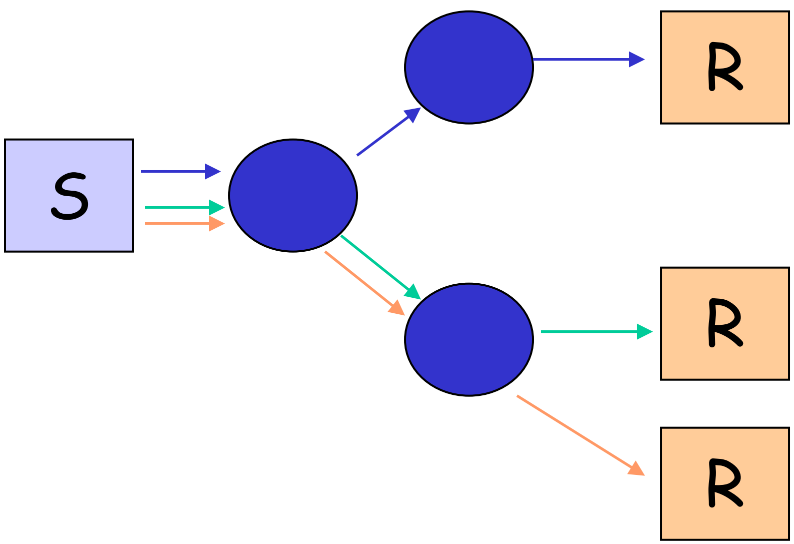
128.146.199.0/24
128.146.116.0/24
Sender
Receiver
128.146.222.0/24
128.146.226.0/24
Receiver
Receivers

# Multicast

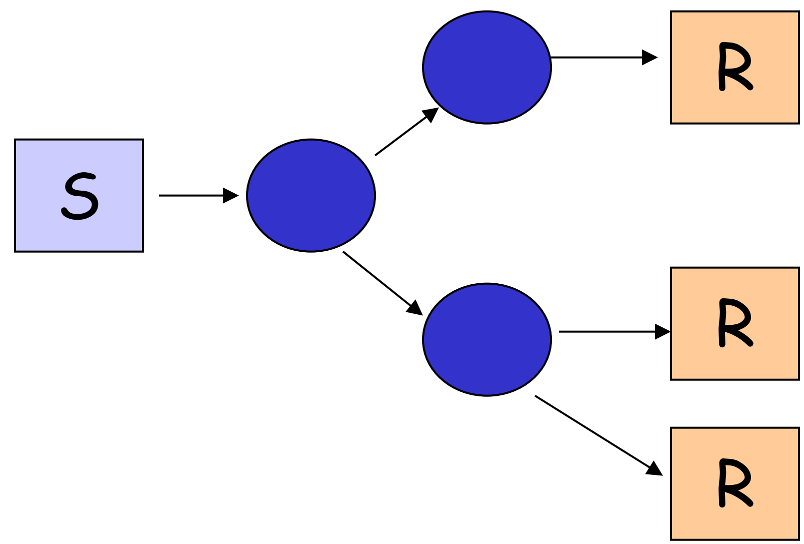

128.146.199.0/24
128.146.116.0/24
Sender
Receiver
128.146.222.0/24
128.146.226.0/24
Receiver
Receivers

# One to Many Communication

- Application-level one to many communication
- Multiple unicasts



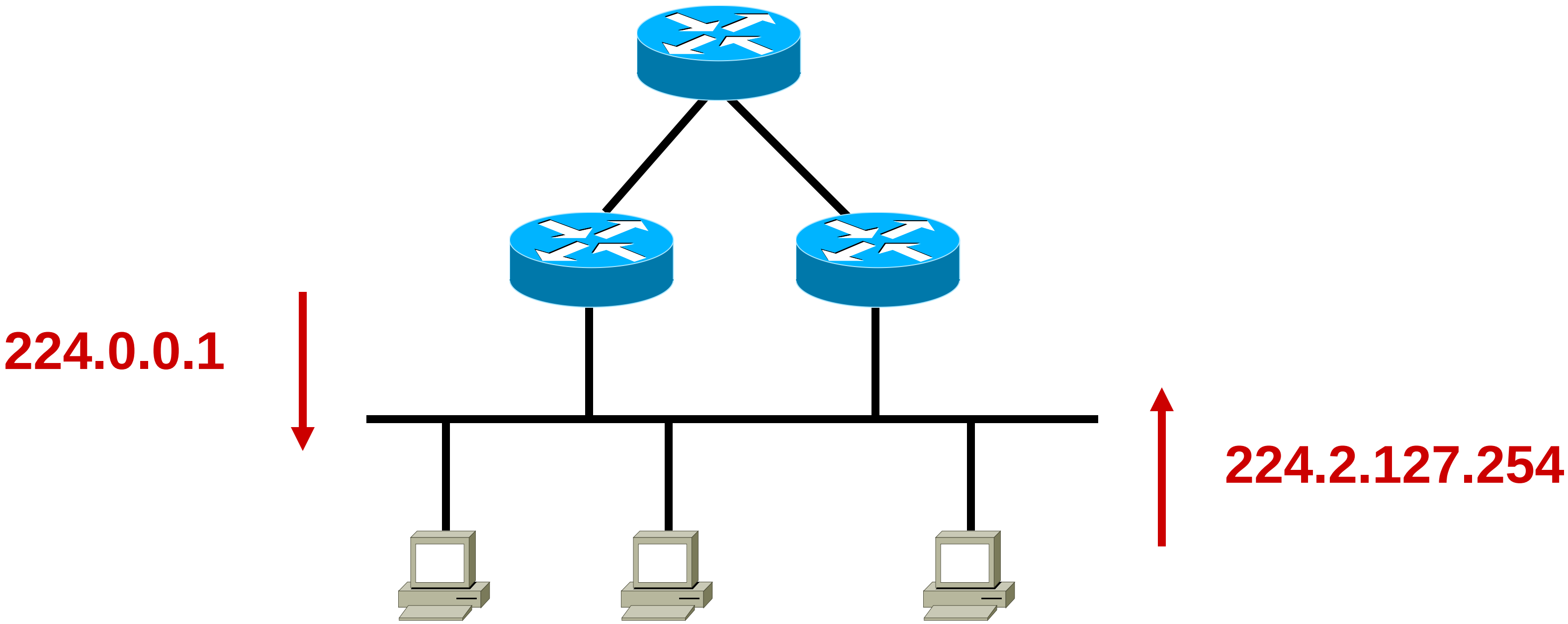

- IP multicast



# Two Major Issues

- Who are the multicast members?
- How to send the packets to the members?

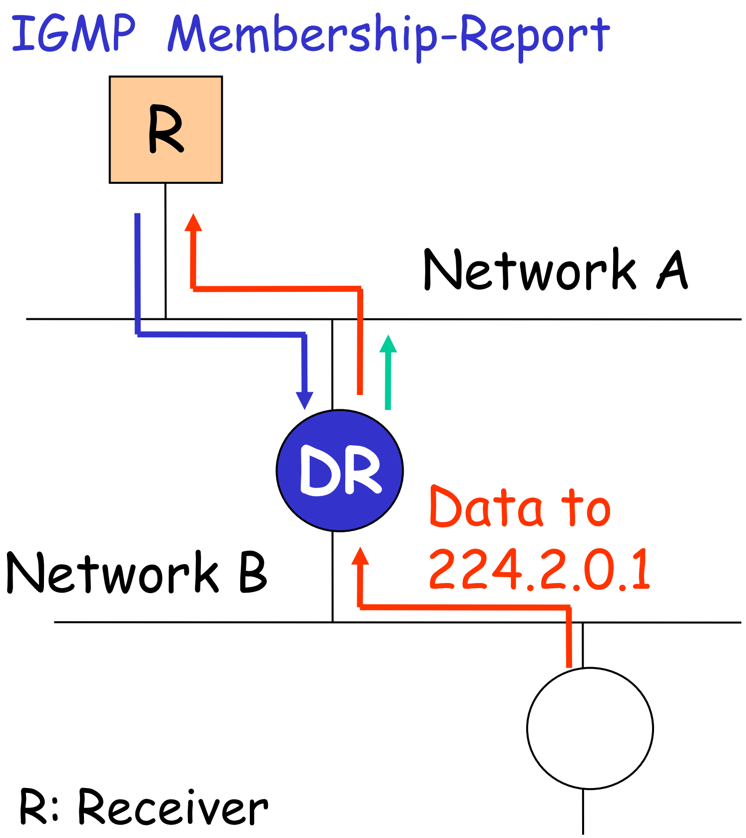
# IGMP




Designated router queries LAN for group membership

Host informs router with IGMP report

# IGMP – Joining a Group



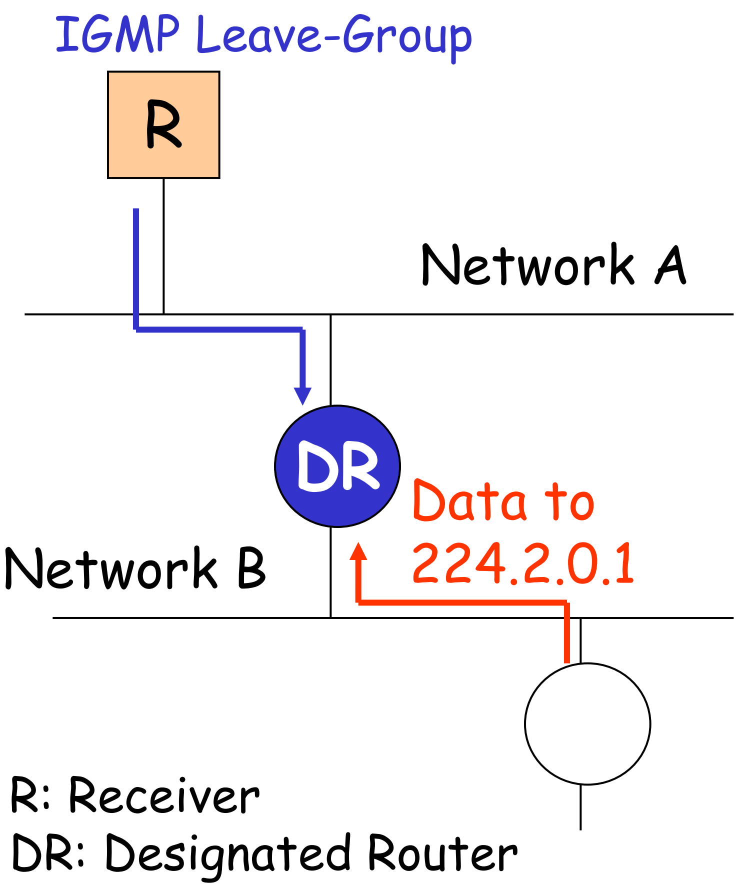

R: Receiver

DR: Designated Router

Example : R joins to Group 224.2.0.1

- R sends IGMP Membership-Report to 224.2.0.1
- DR receives it. DR will start forwarding packets for 224.2.0.1 to Network A
- DR periodically sends IGMP Membership-Query to 224.0.0.1
- R answers IGMP Membership-Report to 224.2.0.1

# IGMP – Leaving a Group

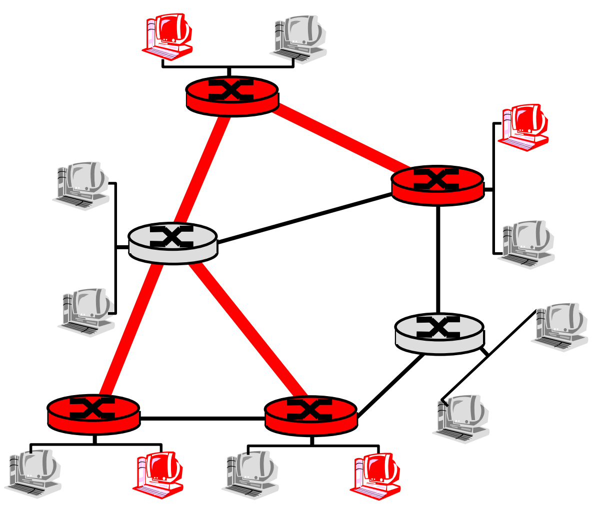



Example : R leaves from a Group 224.2.0.1

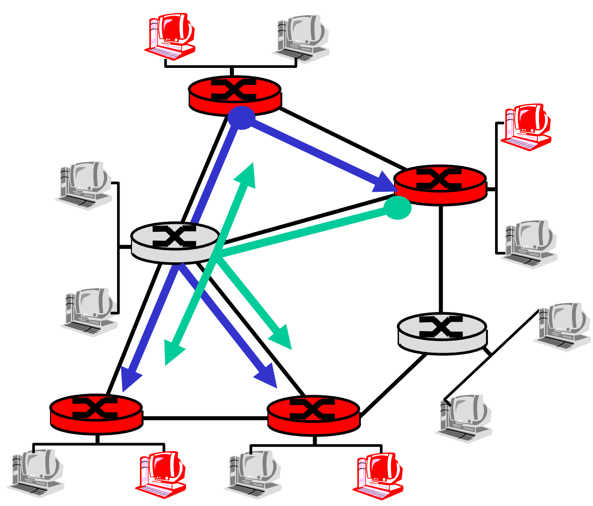
- R sends IGMP Leave-Group to 224.0.0.2
- DR receives it.
- DR stops forwarding packets for 224.2.0.1 to Network A if no more 224.2.0.1 group members on Network A
- Leave-Group is optional

# 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

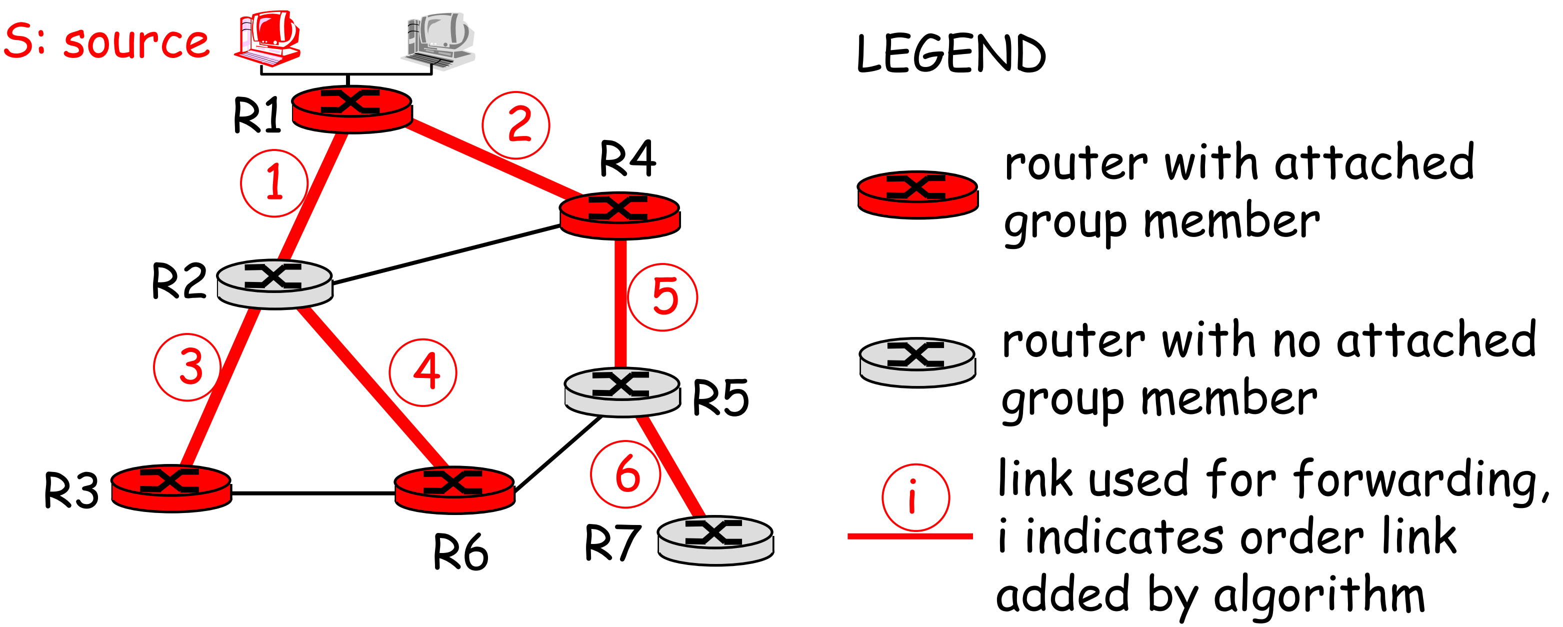
Approaches:

- 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

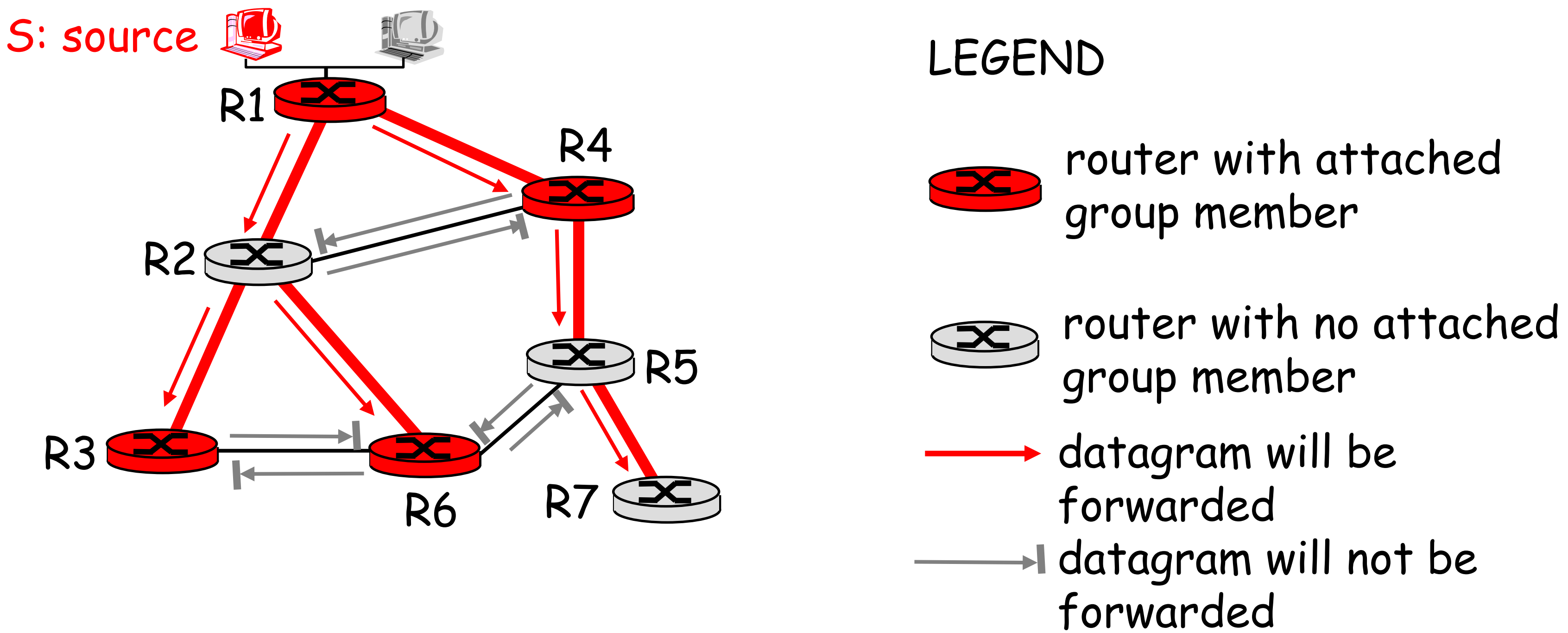


# Reverse Path Forwarding

- 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

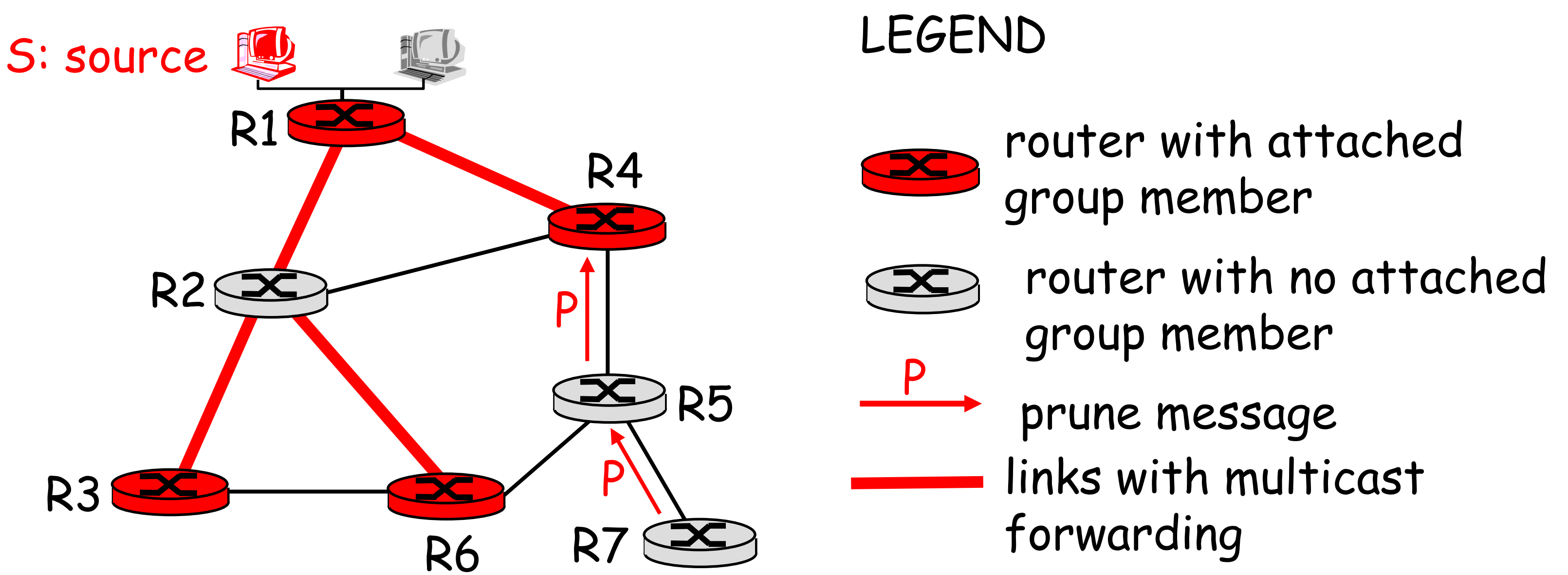
# Reverse Path Forwarding: Example




- 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



# Shared-Tree: Steiner Tree

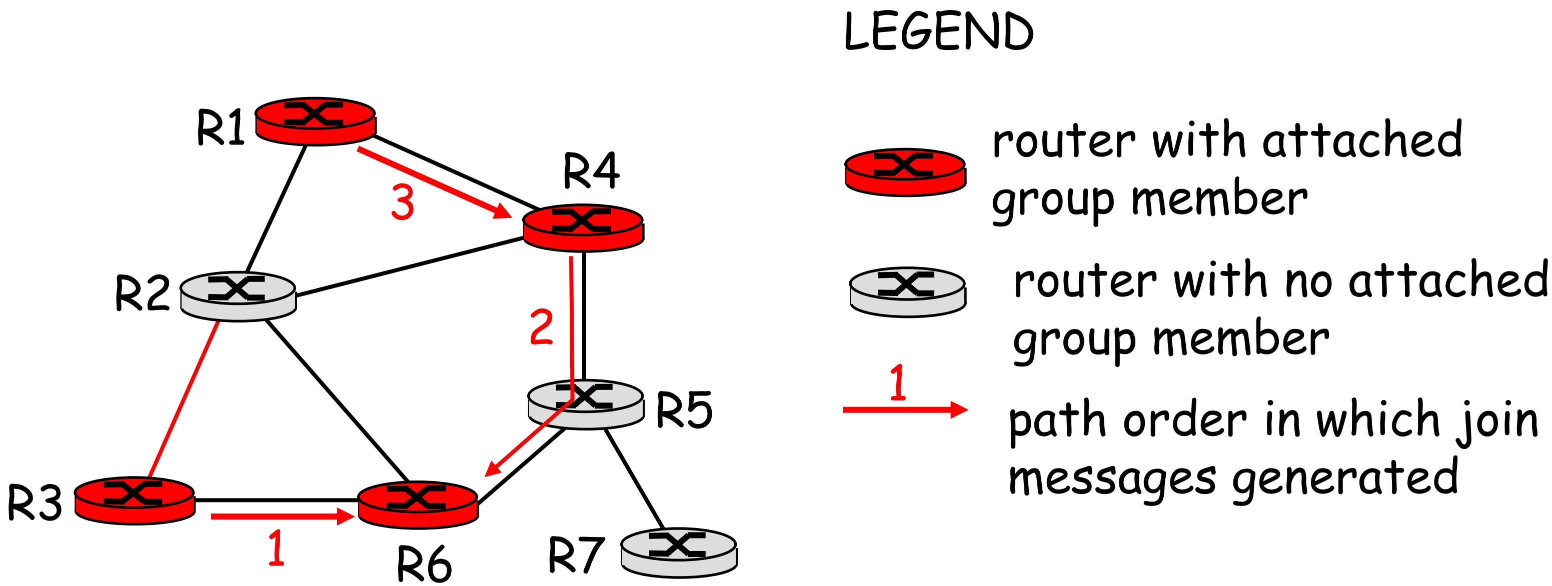
- Steiner Tree: minimum cost tree connecting all routers with attached group members
- Problem is NP-complete
- Excellent heuristics exists
- Not used in practice:
  - Computational complexity
  - Information about entire network needed
  - Monolithic: rerun whenever a router needs to join/leave

# 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 Multicasting Routing: DVMRP

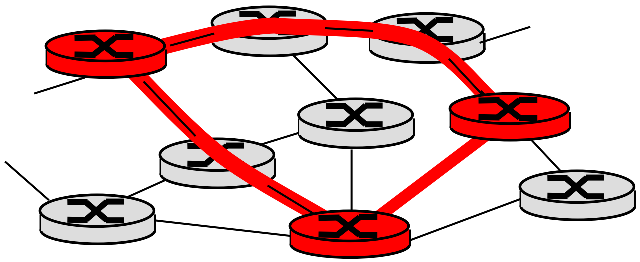
- **DVMRP:** distance vector multicast routing protocol, RFC1075
- *Flood and prune:* reverse path forwarding, source-based tree
  - RPF tree based on DVMRP's own routing tables constructed by communicating DVMRP routers
  - No assumptions about underlying unicast
  - Initial datagram to mcast group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

# DVMRP: continued…

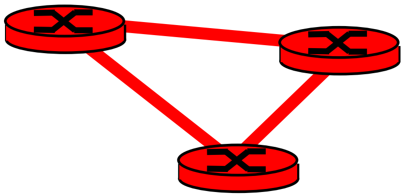
- *Soft state:* DVMRP router periodically (1 min.) "forgets" branches are pruned:
  - Mcast data again flows down unpruned branch
  - Downstream router: reprune or else continue to receive data
- Routers can quickly regraft to tree
  - Following IGMP join at leaf
- Odds and ends
  - Commonly implemented 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*

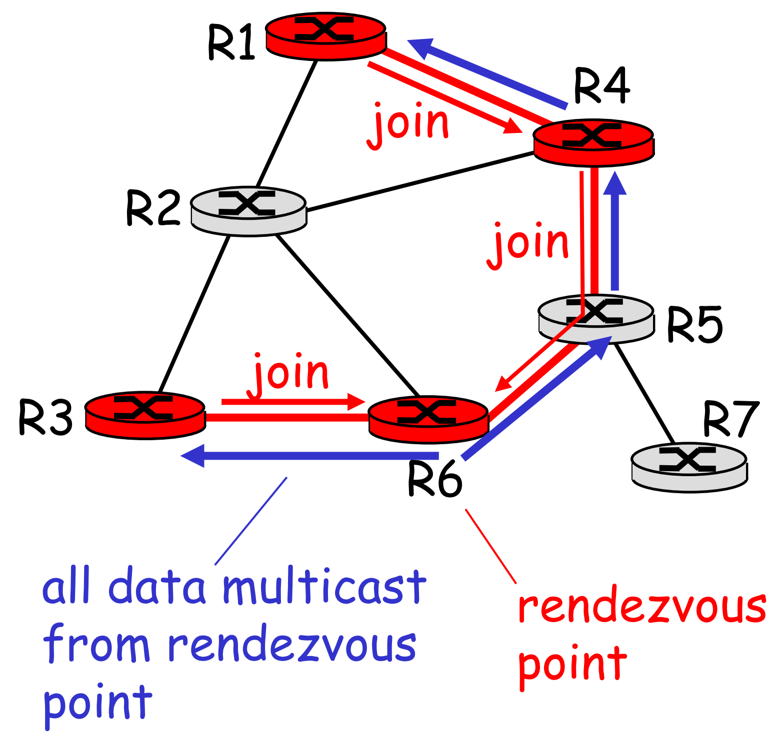
# PIM- Dense Mode

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

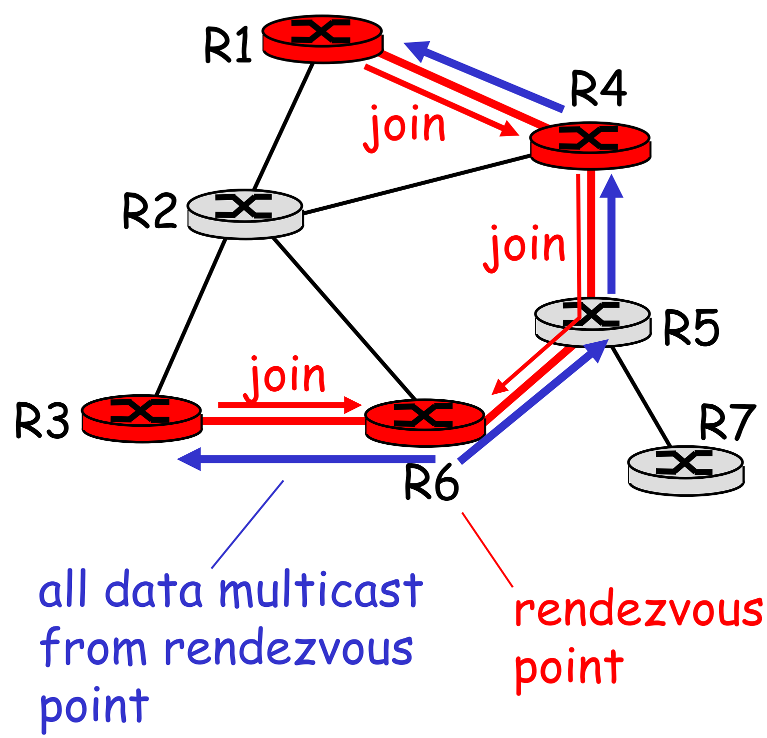
# PIM - Sparse Mode

- 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

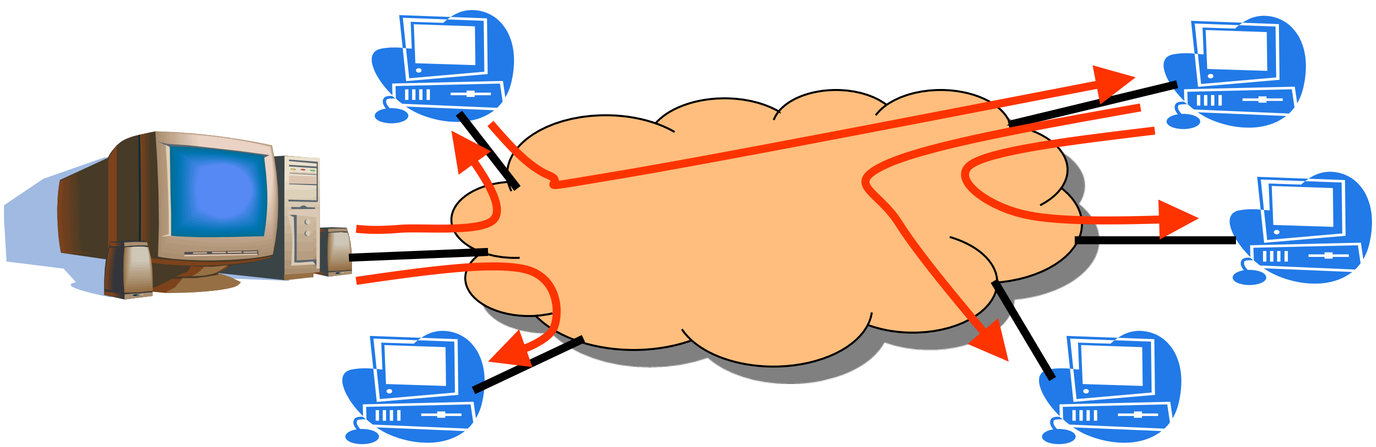
- <u>Sender(s):</u>
  - Unicast data to RP, which distributes down RP-rooted tree
  - RP can extend mcast tree upstream to source
  - RP can send *stop* msg if no attached receivers
    - "no one is listening!"



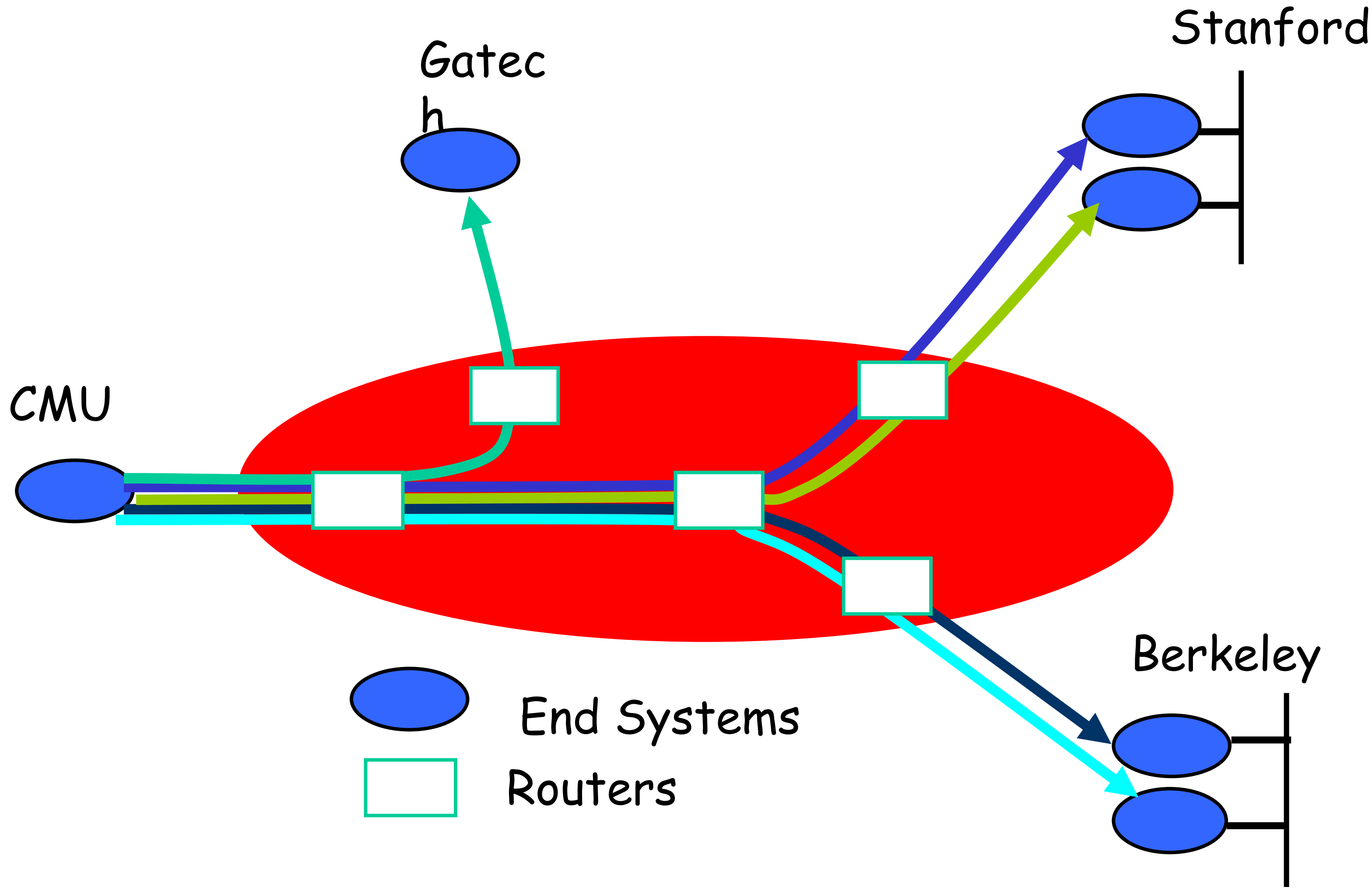
# ALM:
# Application Level Multicast

# End-System Multicast

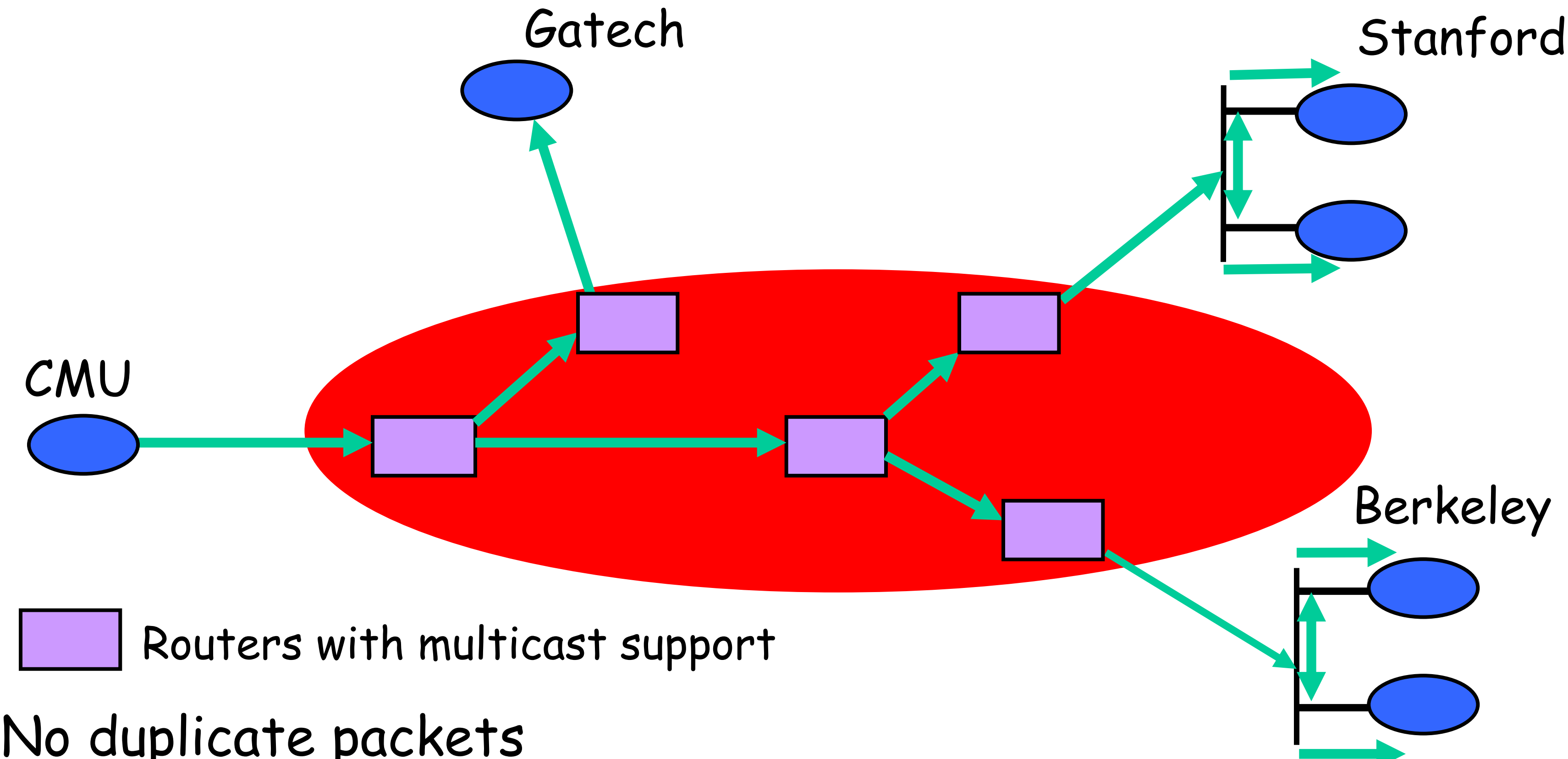
- IP multicast still is not widely deployed
  - Technical and business challenges
  - Should multicast be a *network*-layer service?
- Multicast tree of end hosts
  - Allow end hosts to form their own multicast tree
  - Hosts receiving the data help forward to others



# Unicast Emulation of Multicast



# IP Multicast



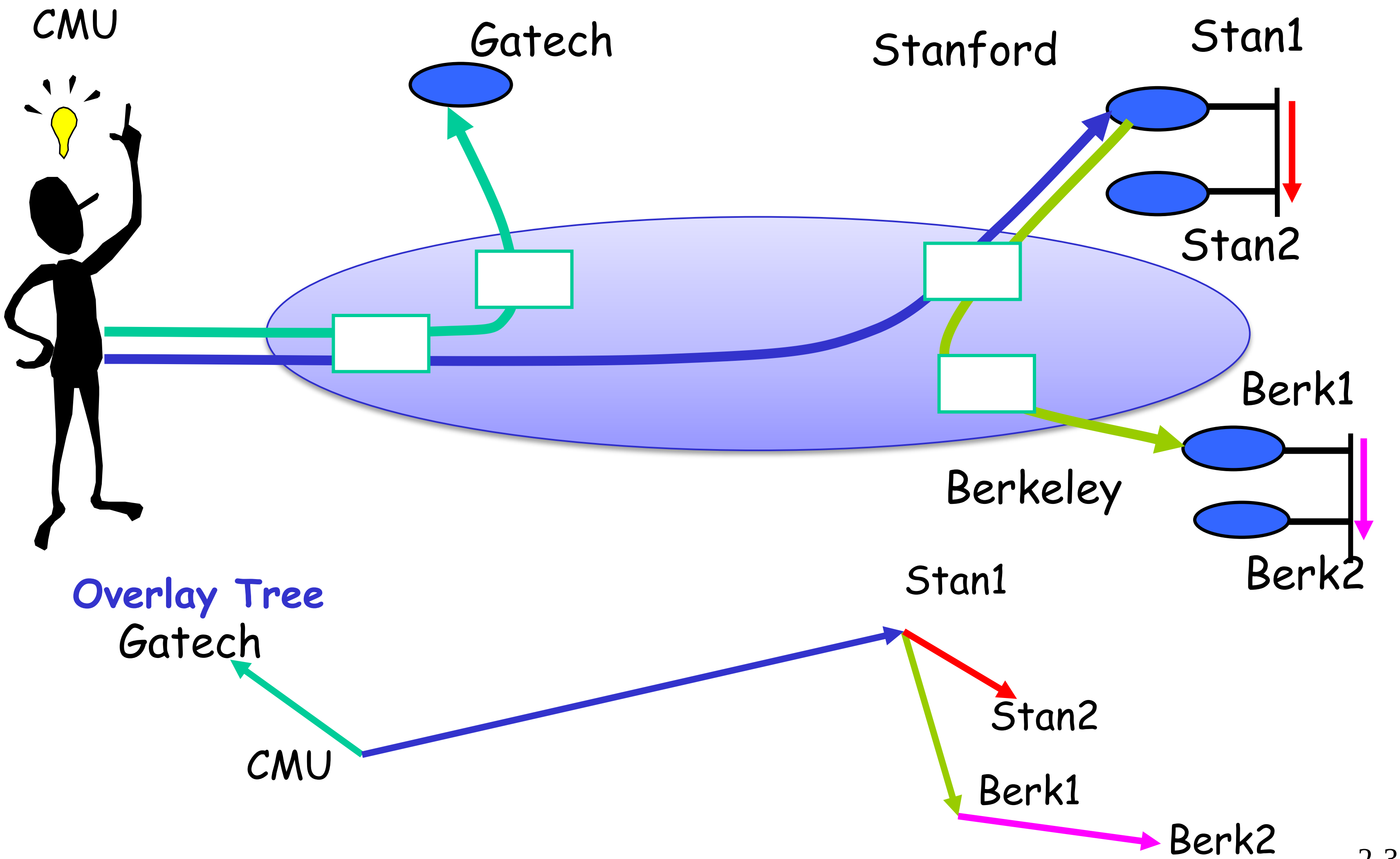

- No duplicate packets
- Highly efficient bandwidth usage

Key Architectural Decision: Add support for multicast in IP layer

# Key Concerns with IP Multicast

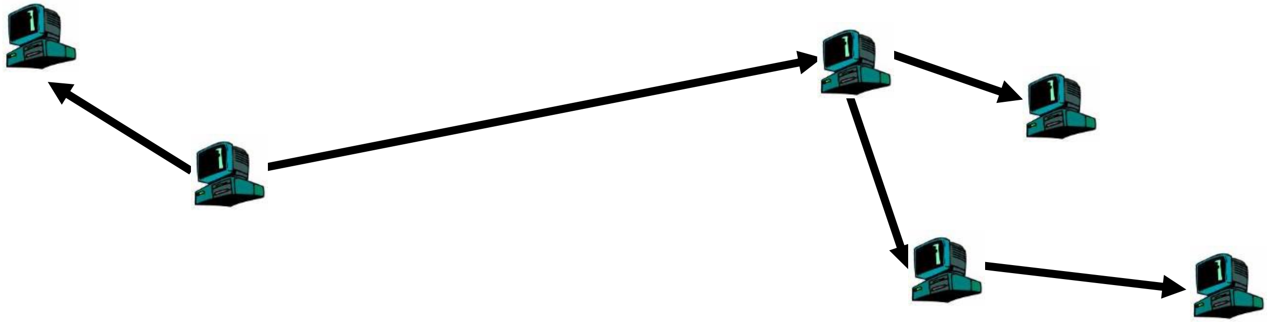
- Scalability with number of groups
  - Routers maintain per-group state
  - Analogous to per-flow state for QoS guarantees
  - Aggregation of multicast addresses is complicated
- Supporting higher level functionality is difficult
  - IP Multicast: best-effort multi-point delivery service
  - End systems responsible for handling higher level functionality
  - Reliability and congestion control for IP Multicast complicated
- Deployment is difficult and slow
  - ISP's reluctant to turn on IP Multicast

# End System Multicast



# Potential Benefits

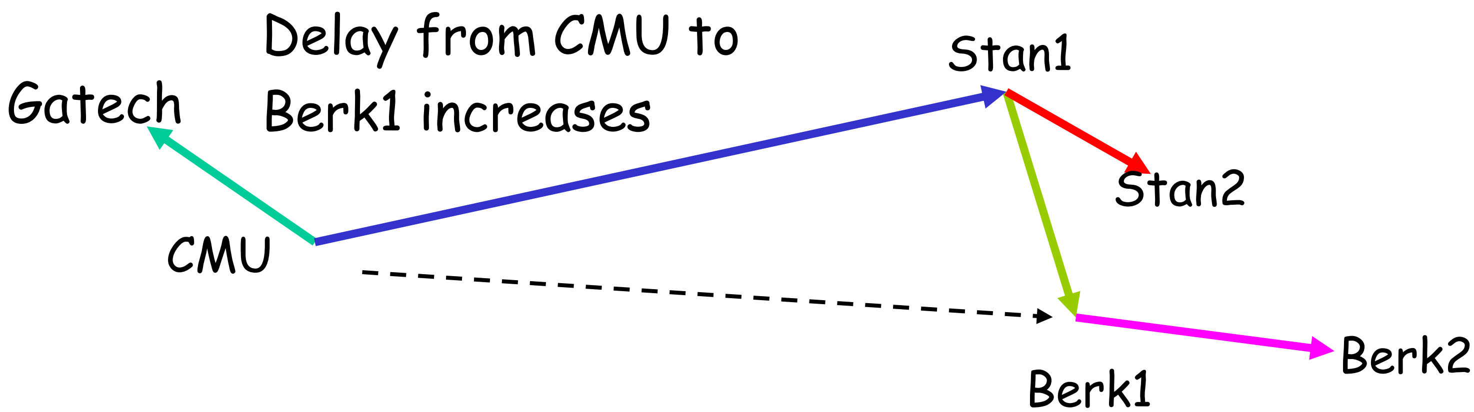
- Scalability
  - Routers do not maintain per-group state
  - End systems do, but they participate in very few groups
- Easier to deploy
- Potentially simplifies support for higher level functionality
  - Leverage computation and storage of end systems
  - For example, for buffering packets, transcoding, ACK aggregation
  - Leverage solutions for unicast congestion control and reliability

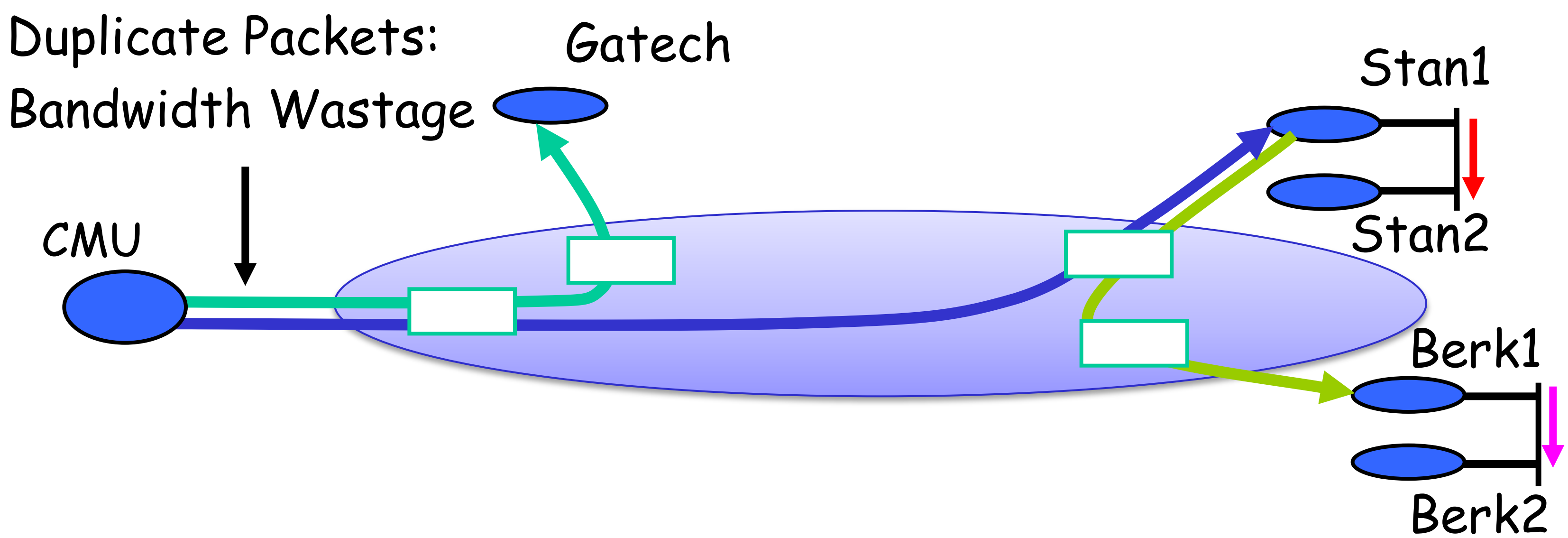


# Design Questions

- Is End System Multicast Feasible?
- Target applications with small and sparse groups
- How to Build Efficient Application-Layer Multicast "Tree" or Overlay Network?
  - Narada: A distributed protocol for constructing efficient overlay trees among end systems
  - Simulation and Internet evaluation results to demonstrate that Narada can achieve good performance

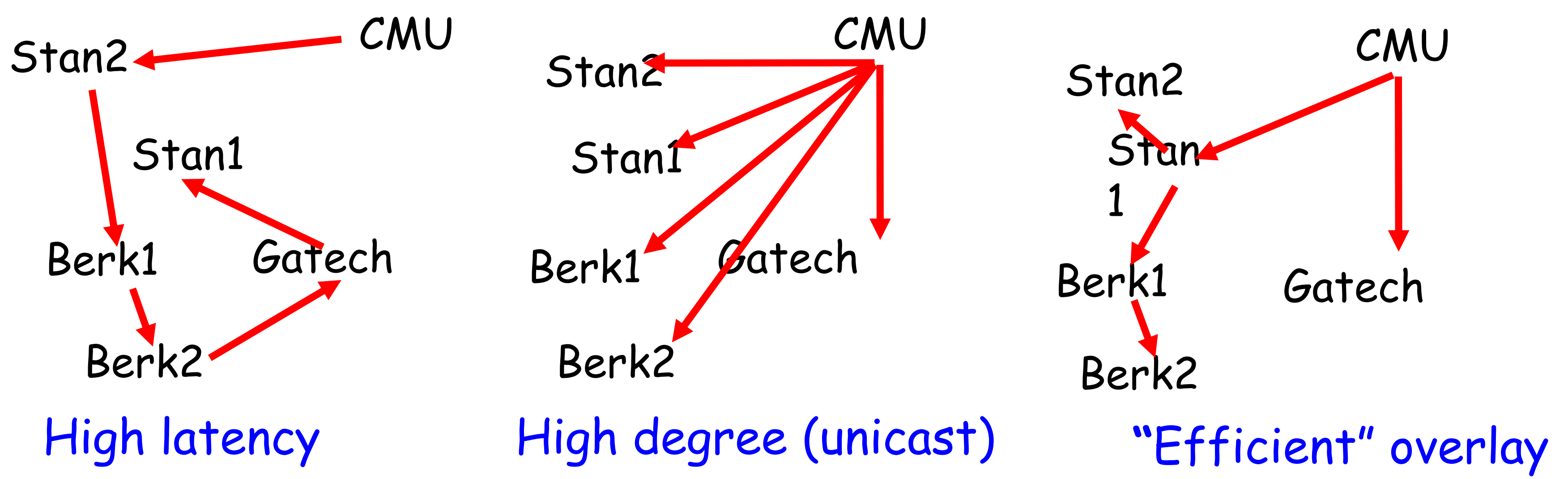
# Performance Concerns



# What is an Efficient Overlay Tree?

- The delay between the source and receivers is small
- Ideally,
  - The number of redundant packets on any physical link is low
- Heuristic used:
  - Every member in the tree has a small degree
  - Degree chosen to reflect bandwidth of connection to Internet



# Why is Self-Organization Hard?

- Dynamic changes in group membership
  - Members may join and leave dynamically
  - Members may die
- Limited knowledge of network conditions
  - Members do not know delay to each other when they join
  - Members probe each other to learn network related information
  - Overlay must self-improve as more information available
- Dynamic changes in network conditions
  - Delay between members may vary over time due to congestion

# Narada Design

Step 1

**"Mesh"**: Richer overlay that may have cycles and includes all group members

• Members have low degrees

• Shortest path delay between any pair of members along mesh is small

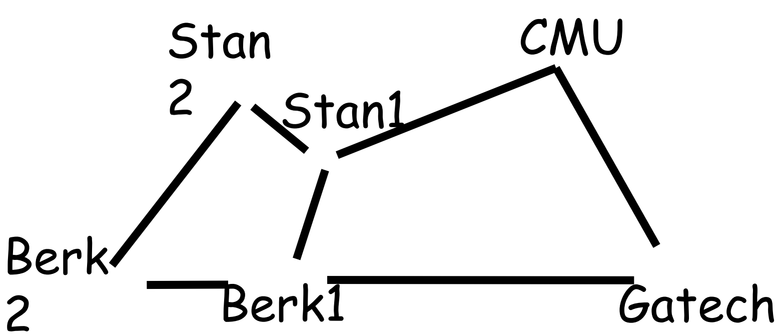
Step 2

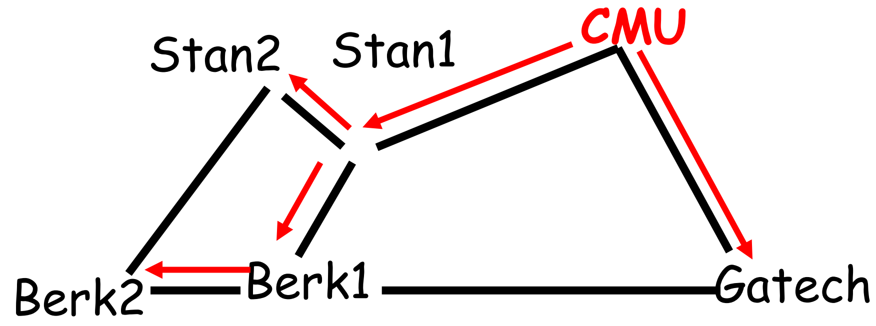
•Source rooted shortest delay spanning trees of mesh

•Constructed using well known routing algorithms

– Members have low degrees

– Small delay from source to receivers

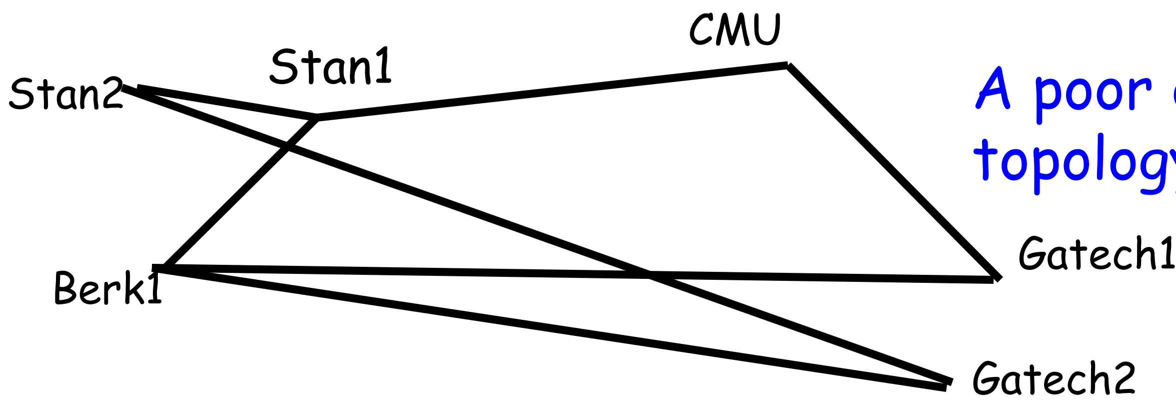


# Narada Components

- Mesh Management:
  - Ensures mesh remains connected in face of membership changes
- Mesh Optimization:
  - Distributed heuristics for ensuring shortest path delay between members  along the mesh is small
- Spanning tree construction:
  - Routing algorithms for constructing data-delivery trees
  - Distance vector routing, and reverse path forwarding

# Optimizing Mesh Quality




A poor overlay topology

- Members periodically probe other members at random
- New Link added if

  Utility Gain of adding link > Add Threshold
- Members periodically monitor existing links
- Existing Link dropped if

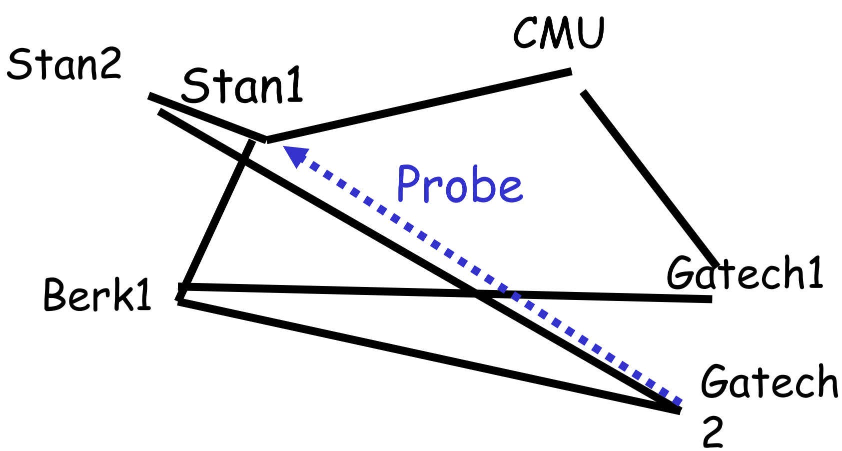
  Cost of dropping link < Drop Threshold

# The Terms Defined

- Utility gain of adding a link based on
  - The number of members to which routing delay improves
  - How significant the improvement in delay to each member is
- Cost of dropping a link based on
  - The number of members to which routing delay increases, for either neighbor
- Add/Drop Thresholds are functions of:
  - Member's estimation of group size
  - Current and maximum degree of member in the mesh

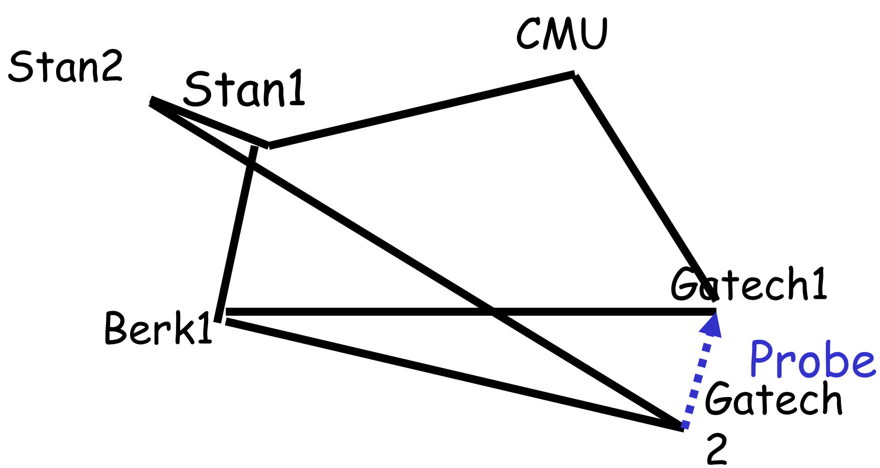
# Desirable Properties of Heuristics

- Stability: A dropped link will not be immediately re-added
- Partition Avoidance: A partition of the mesh is unlikely to be caused as a result of any single link being dropped



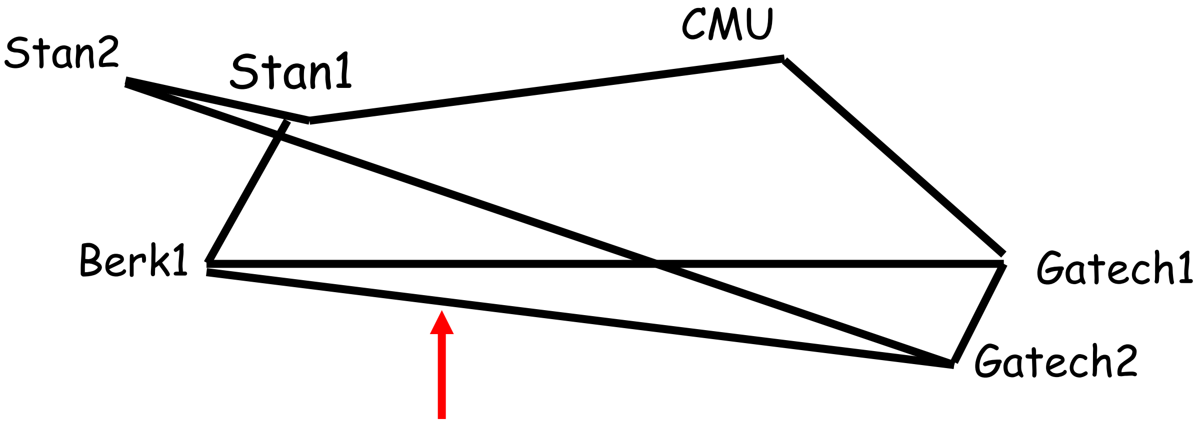

Delay improves to Stan1, CMU but marginally.

Do not add link!



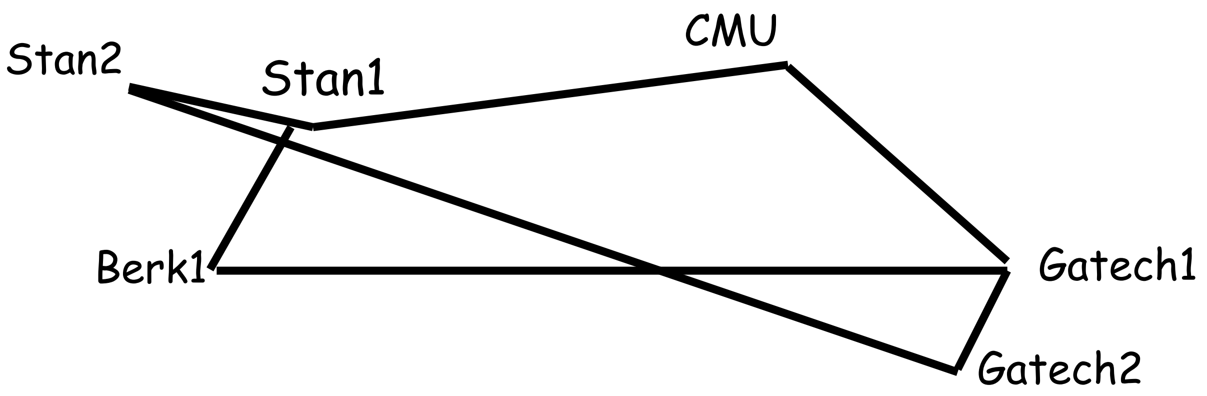

Delay improves to CMU, Gatech1 and significantly.

Add link!




Used by Berk1 to reach only Gatech2 and vice versa.

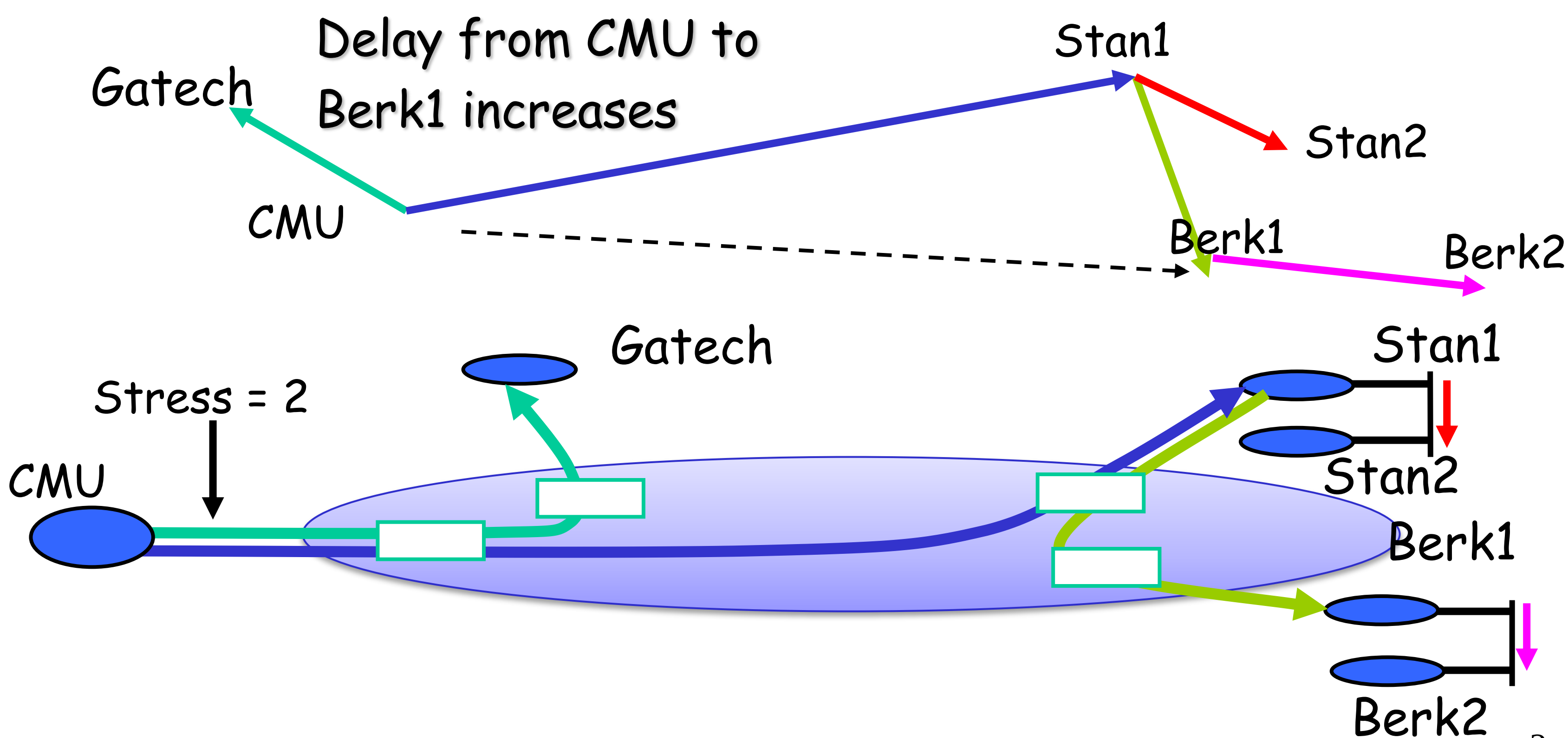
Drop!!




An improved mesh !!

# Performance Metrics

- Delay between members using Narada
- Stress, defined as the number of identical copies of a packet that traverse a physical link



# Factors Affecting Performance

- Topology Model
  - Waxman Variant
  - Mapnet: Connectivity modeled after several ISP backbones
  - ASMap: Based on inter-domain Internet connectivity
- Topology Size
  - Between 64 and 1024 routers
- Group Size
  - Between 16 and 256
- Fanout range
  - Number of neighbors each member tries to maintain in the mesh

# ESM Conclusions

- Proposed in 1989, IP Multicast is not yet widely deployed
  - Per-group state, control state complexity and scaling concerns
  - Difficult to support higher layer functionality
  - Difficult to deploy, and get ISP's to turn on IP Multicast
- Is IP the right layer for supporting multicast functionality?
- For small-sized groups, an end-system overlay approach
  - is feasible
  - has a low performance penalty compared to IP Multicast
  - has the potential to simplify support for higher layer functionality
  - allows for application-specific customizations