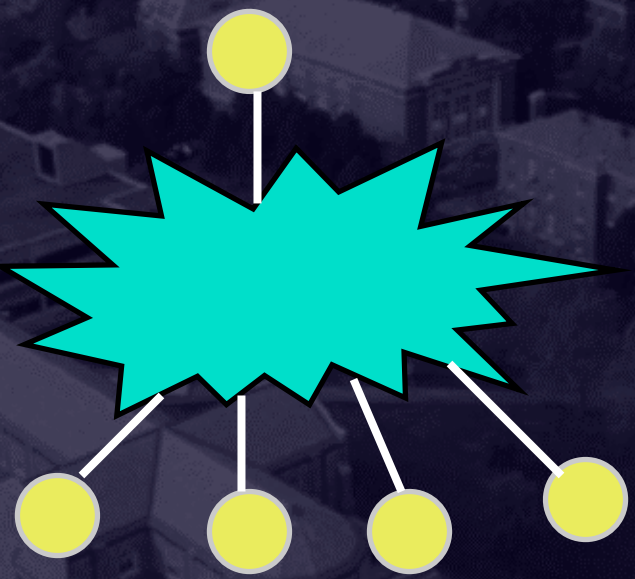
Multicast



Shivkumar Kalyanaraman

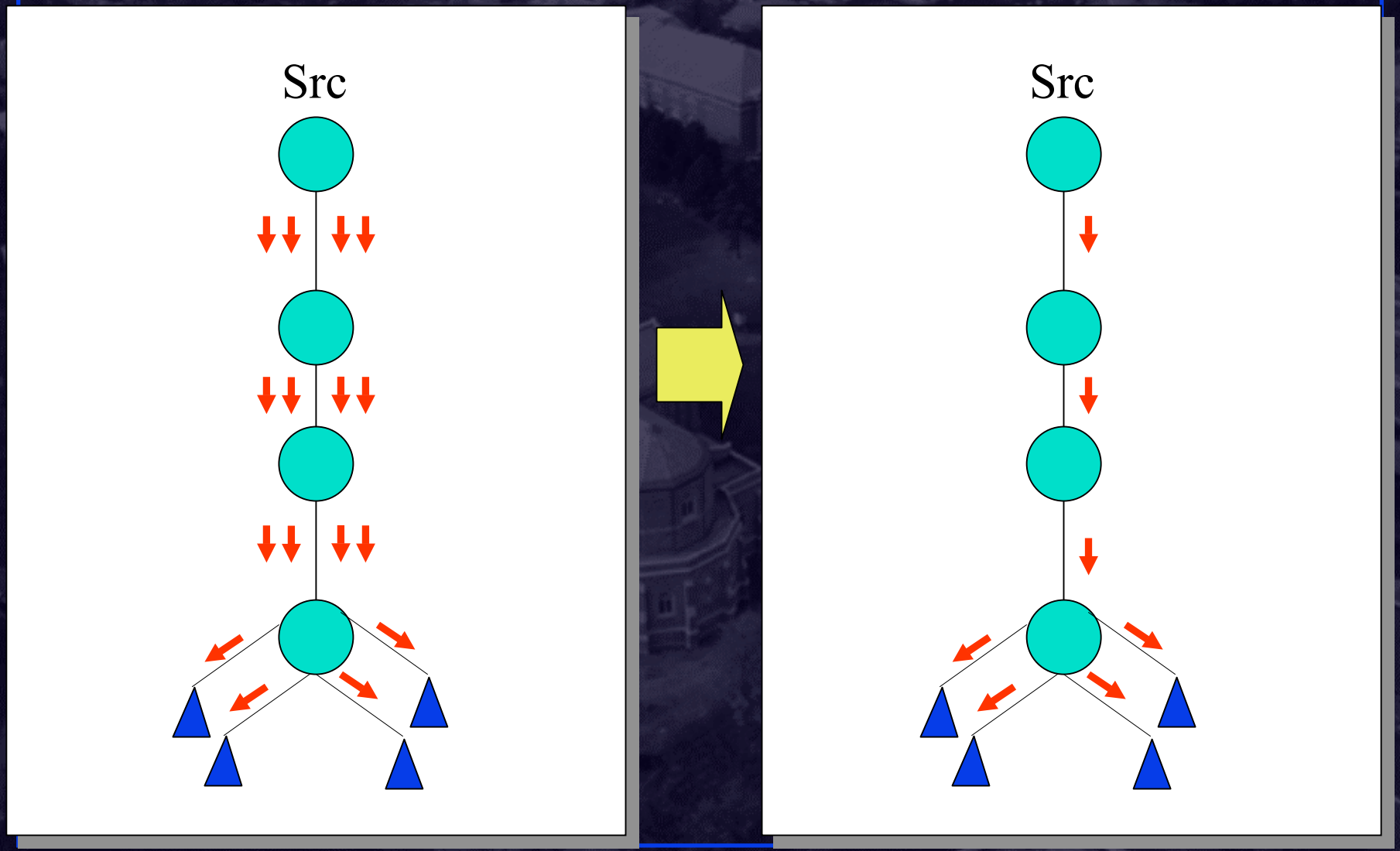
shivkuma@ecse.rpi.edu

http://www.ecse.rpi.edu/Homepages/shivkuma

Adapted in part from Srini Seshan's (CMU) and Ion Stoica (UCB)

- Why multicast ? Multicast apps ...
- Concepts: groups, scopes, trees
- Multicast addresses, LAN multicast
- Group management: IGMP
- Multicast routing and forwarding: MBONE, PIM
- Reliable Multicast Transport Protocols
  - Multicast Congestion Control
- Deployment issues, Source-Specific Multicast (SSM), Application-level Multicast

Multicast = Efficient Data Distribution



Why Multicast ?

- Need for *efficient one-to-many delivery of same data*
- Applications:
  - News/sports/stock/weather updates
  - Distance learning
  - Configuration, routing updates, service location
  - Pointcast-type “push” apps
  - Teleconferencing (audio, video, shared whiteboard, text editor)
  - Distributed interactive gaming or simulations
  - Email distribution lists
  - Content distribution; Software distribution
  - Web-cache updates
  - Database replication

Why Not Broadcast or Unicast?

- Broadcast:
  - Send a copy to every machine on the net
  - Simple, but inefficient
  - All nodes must process packet even if they don't care
  - Wastes *more* CPU cycles of slower machines ("*broadcast radiation*")
  - Network loops lead to "*broadcast storms*"
- Replicated Unicast:
  - Sender sends a copy to each receiver in turn
  - Receivers need to register or sender must be pre-configured
  - Sender is focal point of all control traffic
  - Reliability => per-receiver state, separate sessions/processes at sender

Why IP multicast ?

- *Application-layer relays:*
  - A "relay" node or set of nodes does the replicated unicast function instead of the source
  - Multiple relays can handle "groups" of receivers and reduce number of packets per multicast => efficiency
  - Manager has to manually configure names of receivers in relays etc
  - App-level topology may be sub-optimal
    - But bandwidth is becoming cheaper
    - Becoming more popular in content distribution
- IP Multicast: *replication/multicast engine at the network layer*

Multicast Apps Characteristics

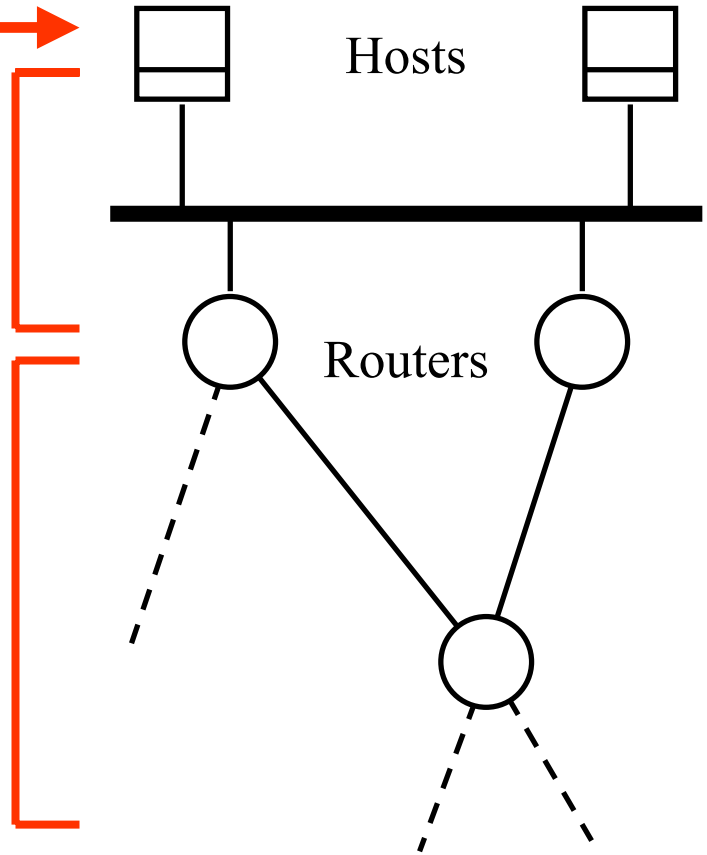
- Number of (simultaneous) senders to the group
- The *size of the groups*
  - *Number* of members (receivers)
  - Geographic extent or *scope*
  - *Diameter* of the group measured in router hops
- The *longevity* of the group
- Number of aggregate packets/second
- The peak/average used by source
- Level of human *interactivity*
  - Lecture mode vs interactive
  - Data-only (eg database replication) vs multimedia

IP Multicast Architecture

Service model

Host-to-router protocol (IGMP)

Multicast routing protocols (various)



IP Multicast model: RFC 1112

- Message sent to multicast "group" (of receivers)
  - *Senders need not be group members*
  - A group identified by a single "group address"
    - Use "group address" instead of destination address in IP packet sent to group
  - Groups can have any size;
  - Group members can be located anywhere on the Internet
  - Group membership is *not explicitly known*
  - Receivers can join/leave at will

IP Multicast Concepts (Continued)

- Packets are not duplicated or delivered to destinations outside the group
  - *Distribution tree* constructed for delivery of packets
  - Packets forwarded "away" from the source
  - No more than one copy of packet appears on any subnet
  - Packets delivered only to "interested" receivers => multicast delivery tree changes dynamically
  - Network has to actively discover paths between senders and receivers

IP Multicast Addresses

- **Class D** IP addresses
  - 224.0.0.0 – 239.255.255.255

- Address allocation:
  - **Well-known** (reserved) multicast addresses, assigned by IANA: 224.0.0.x and 224.0.1.x
    Transient multicast addresses, assigned and reclaimed dynamically, e.g., by "sdr" program
- Each multicast address represents a *group of arbitrary size, called a "host group"*
- There is **no structure** within class D address space like subnetting => **flat address space**

IP Multicast Service — Sending

- Uses normal IP-Send operation, with an IP multicast address specified as the destination
- Must provide sending application a way to:
  - Specify outgoing network interface, if >1 available
  - Specify IP time-to-live (TTL) on outgoing packet
  - Enable/disable loop-back if the sending host is/isnt a member of the destination group on the outgoing interface

IP Multicast Service — Receiving

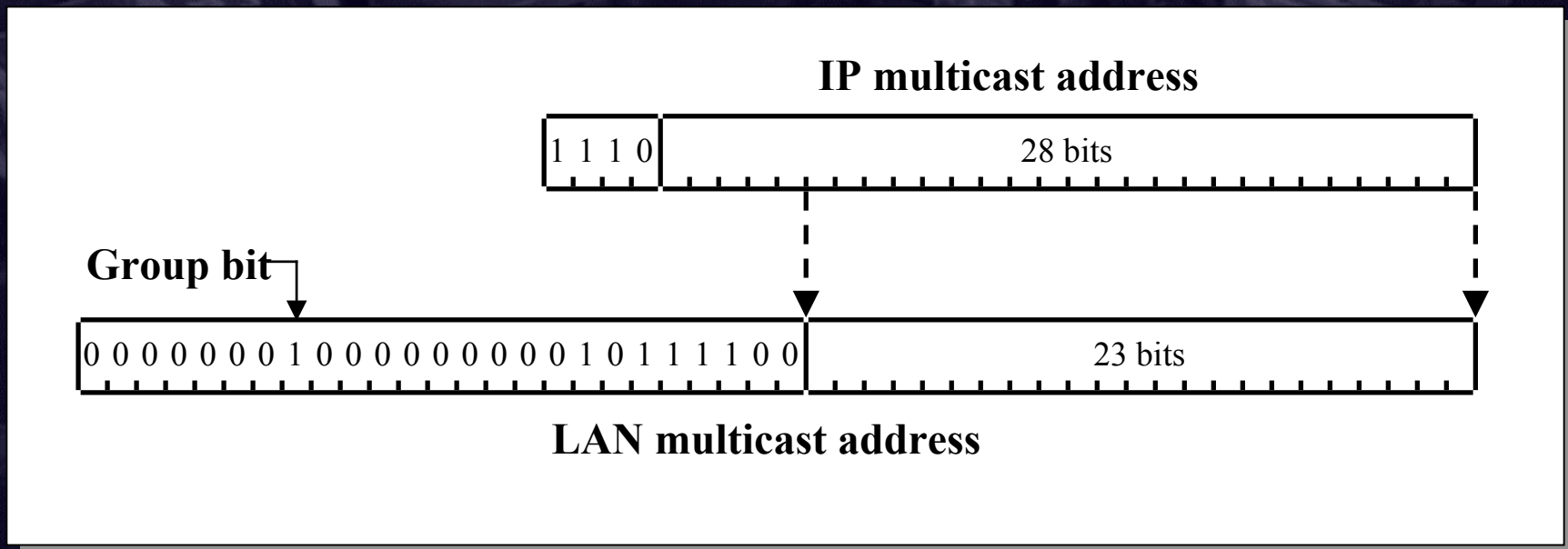
- Two new operations
  - Join-IP-Multicast-Group(group-address, interface)
  - Leave-IP-Multicast-Group(group-address, interface)
- Receive multicast packets for joined groups via normal IP-Receive operation

Link-Layer Transmission/Reception

- **Transmission**
  - IP multicast packet is transmitted as a **link-layer multicast**, on those links that support multicast
  - Link-layer destination address is determined by an algorithm specific to the type of link
- **Reception**
  - Necessary steps are taken to receive desired multicasts on a particular link, such as **modifying address reception filters** on LAN interfaces
  - Multicast **routers** must be able to **receive all IP multicasts on a link**, without knowing in advance which groups will be used

Using Link-Layer Multicast Addresses

- Ethernet and other LANs using 802 addresses:
  - **Direct mapping! Simpler than unicast! No ARP etc.**
  - **32 class D addrs may map to one MAC addr**



- **Special OUI for IETF: 0x01-00-5E.**
- **No mapping needed for point-to-point links**

Multicast over LANs & Scoping

- Multicasts are flooded across MAC-layer bridges along a spanning tree
  - But flooding may steal sending opportunity for non-member stations which want to transmit
  - Almost like broadcast!

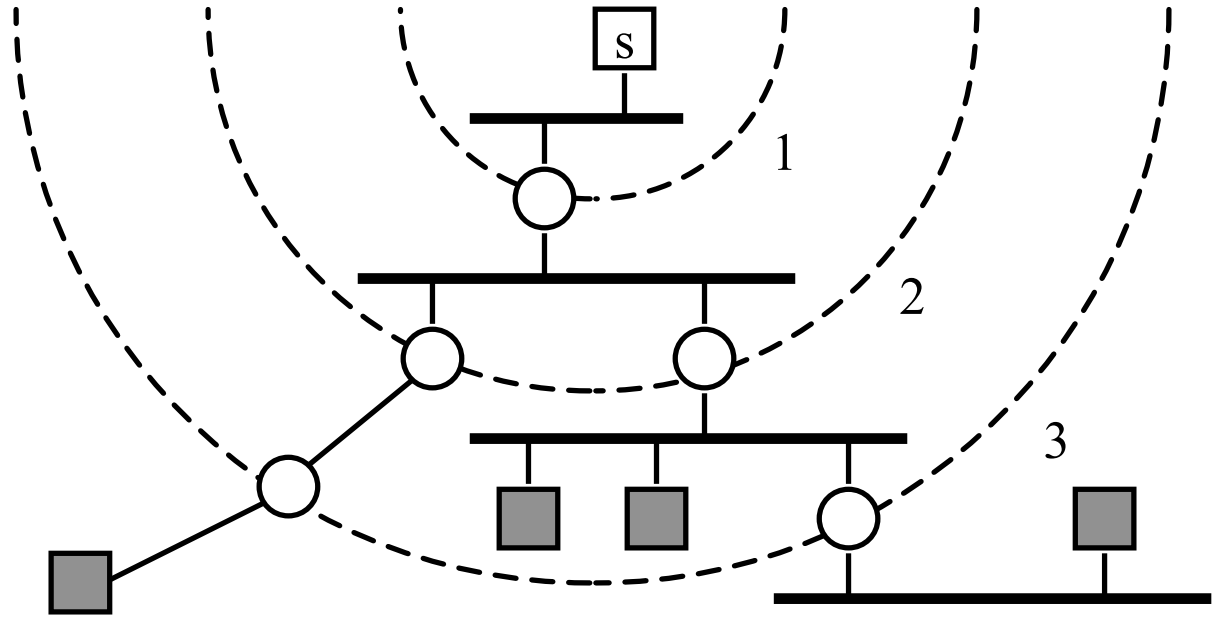
- Scope: How far do transmissions propagate?
- Implicit scoping: Reserved Mcast addresses => don't leave subnet.
  - Also called "*link-local*" addresses

Scope of Multicast Forwarding

- TTL-based scoping:
  - Multicast routers have a configured ***TTL threshold***
  - Mcast datagram dropped if TTL <= TTL threshold
  - Useful as a ***blanket parameter***.
- Administrative scoping:
  - Use a portion of class D address space (239.0.0.0 thru 239.255.255.255)
  - Truly ***local to admin domain***; address reuse possible.
  - In ***IPv6***, scoping is an ***internal attribute*** of an IPv6 multicast address

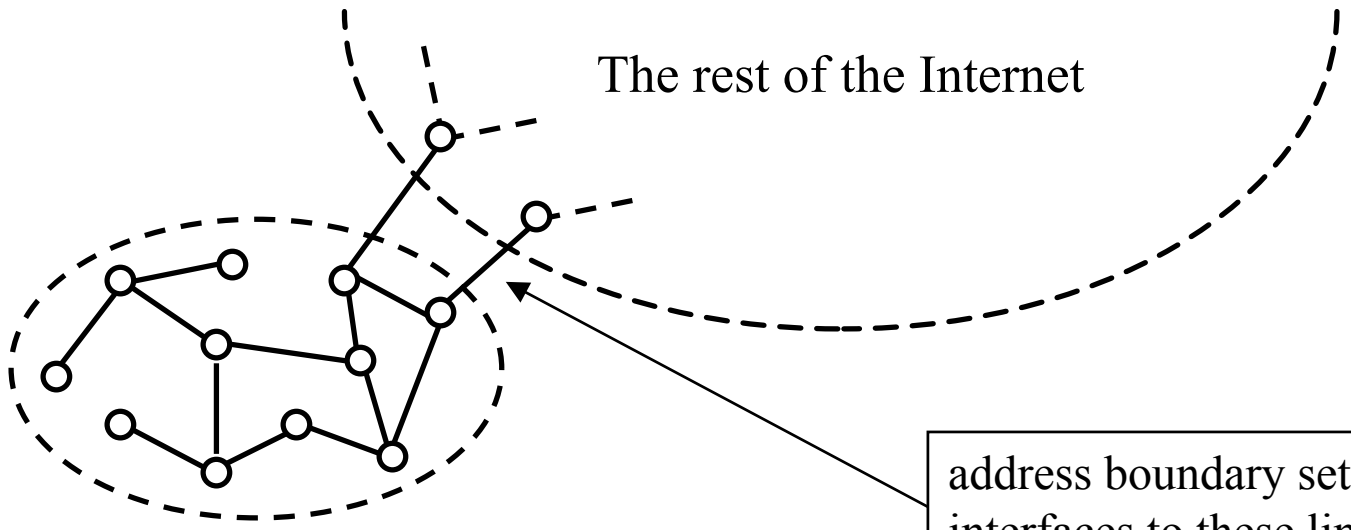
Multicast Scope Control – Small TTLs

- TTL expanding-ring search to reach or find a nearby subset of a group
- Rings can be nested, but not overlapping



Multicast Scope Control

- **Administratively-Scoped** Addresses (RFC 1112)
  - Uses address range 239.0.0.0 — 239.255.255.255
  - Supports **overlapping (not just nested) domains**

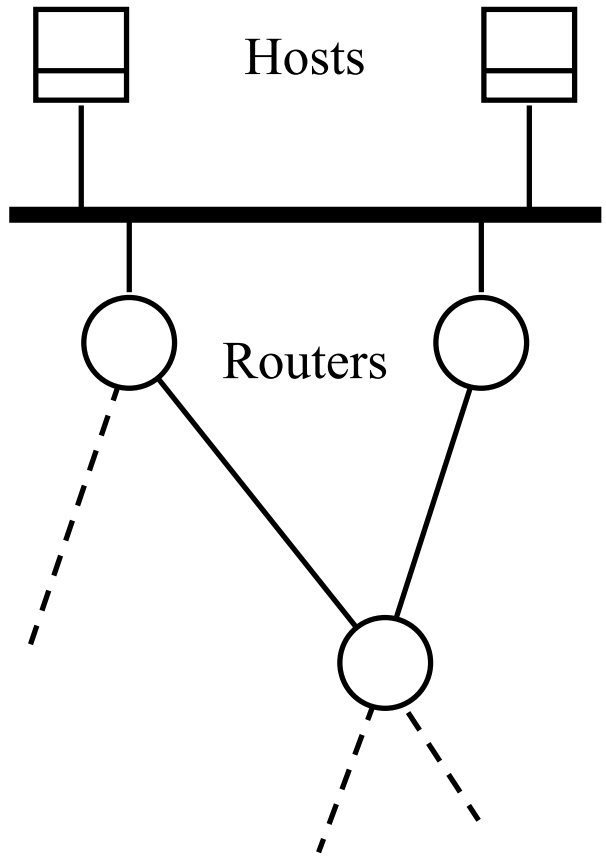


IP Multicast Architecture

Service model

Host-to-router protocol (IGMP)

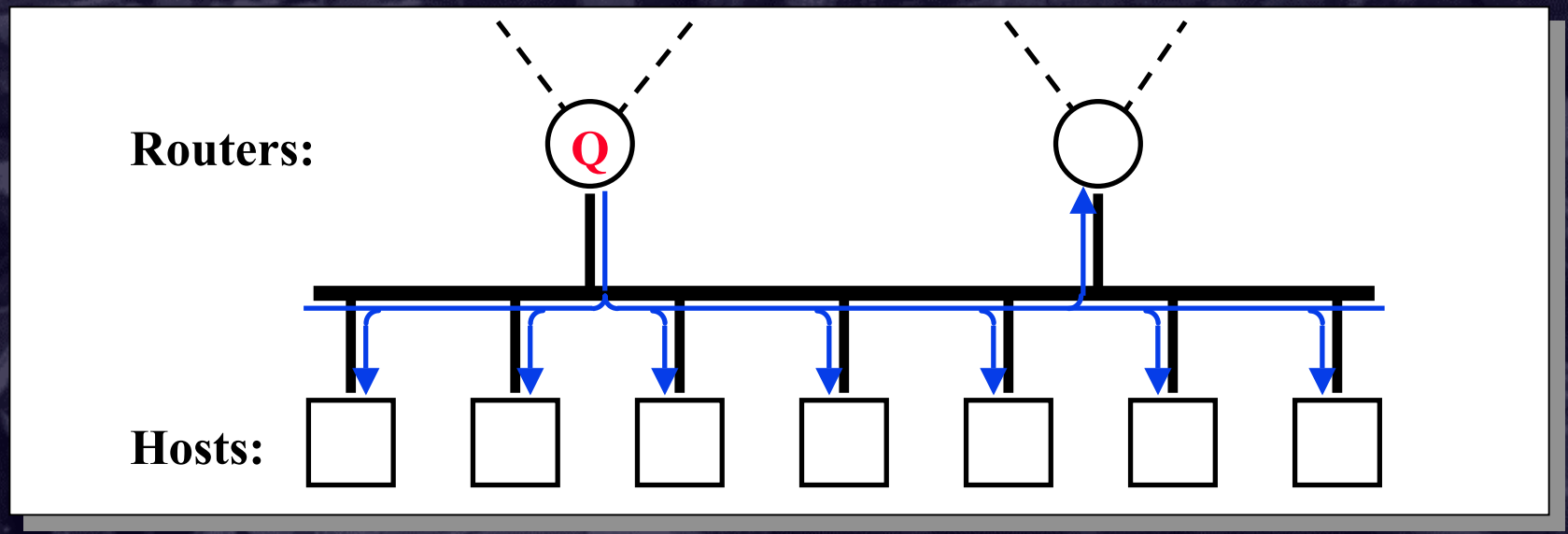
Multicast routing protocols (various)



Internet Group Management Protocol

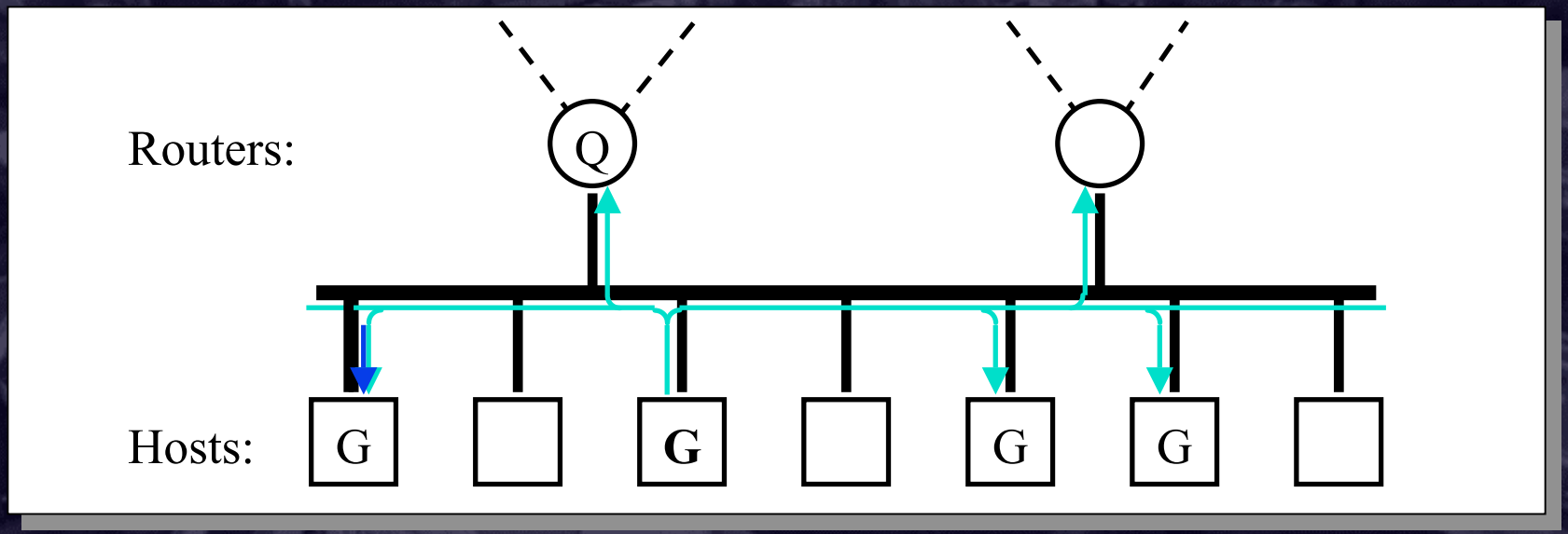
- IGMP: "*signaling*" protocol to establish, maintain, remove groups on a subnet.
- Objective: *keep router up-to-date with group membership of entire LAN*
  - Routers need not know who all the members are, *only that members exist*
- Each host keeps track of which mcast groups are subscribed to
  - Socket API informs IGMP process of all joins

How IGMP Works



- On each link, one router is elected the "**querier**"
- Querier periodically sends a **Membership Query** message to the ***all-systems group (224.0.0.1),*** with ***TTL = 1***
- On receipt, hosts start random timers (between 0 and 10 seconds) for each multicast group to which they belong

How IGMP Works (cont.)



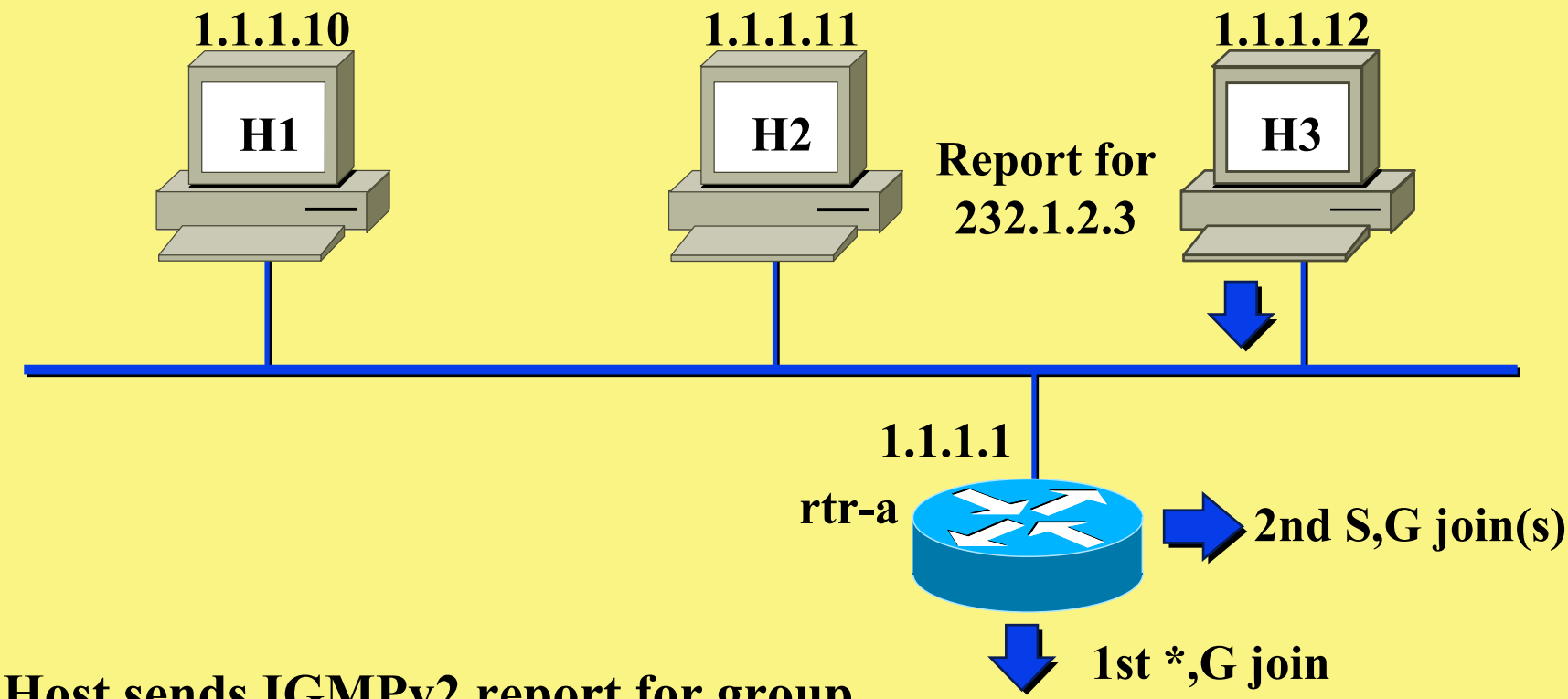
- When a *host's timer for group G expires*, it sends a **Membership Report to group G, with TTL = 1**
- Other members of G hear the report and stop (suppress) their timers
- Routers hear **all** reports, and time out non-responding groups

How IGMP Works (cont.)

- Normal case: only one report message per group present is sent in response to a query
  - Query interval is typically 60-90 seconds
- When a host first joins a group, it sends immediate reports, instead of waiting for a query

- IGMPv2: Hosts may send a "Leave group" message to "all routers" (224.0.0.2) address
  - **Querier responds with a Group-specific Query message: see if any group members are present**
  - **Lower leave latency**

IGMPv2



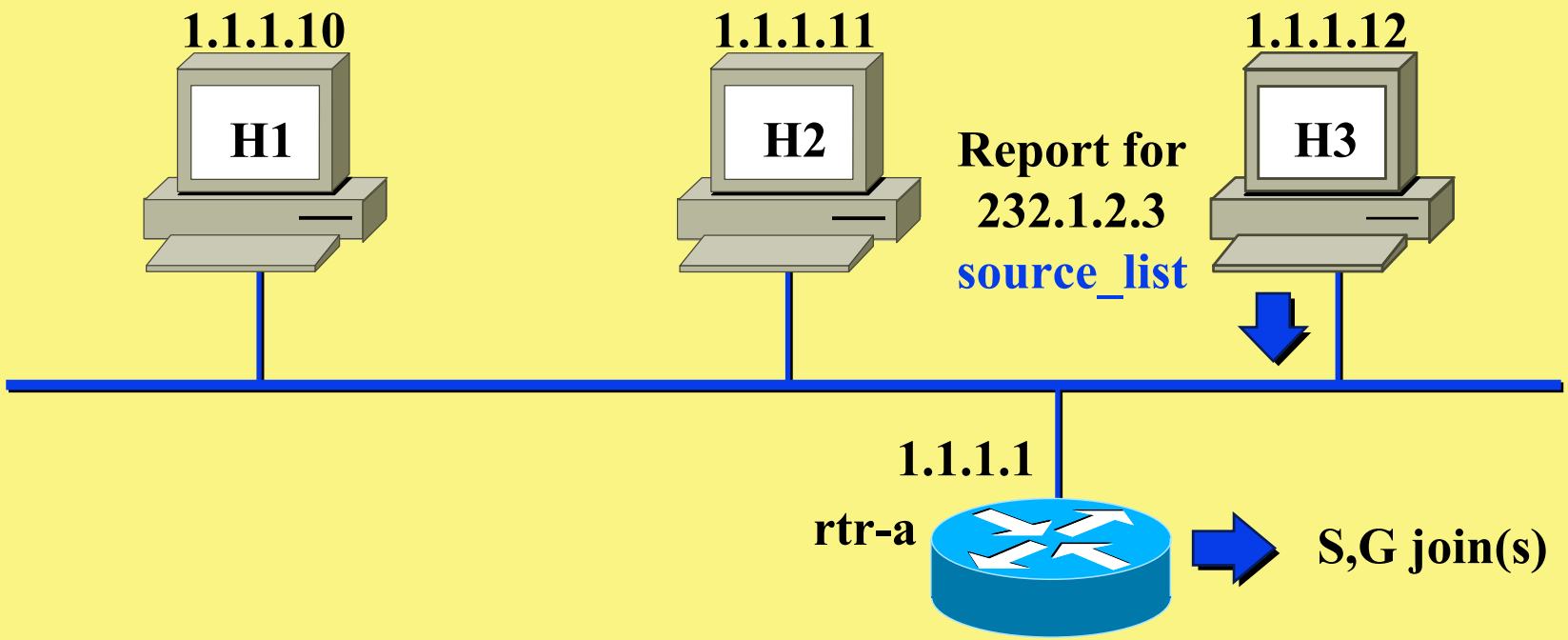
Host sends IGMPv2 report for group

DR adds membership

DR sends *,G join to RP (it has to, it doesn't know the sources)

DR sends S,G join to source (data provides the sources)

IGMPv3



Host sends IGMPv3 report for group which can specify a list of sources to explicitly include.

DR adds membership.

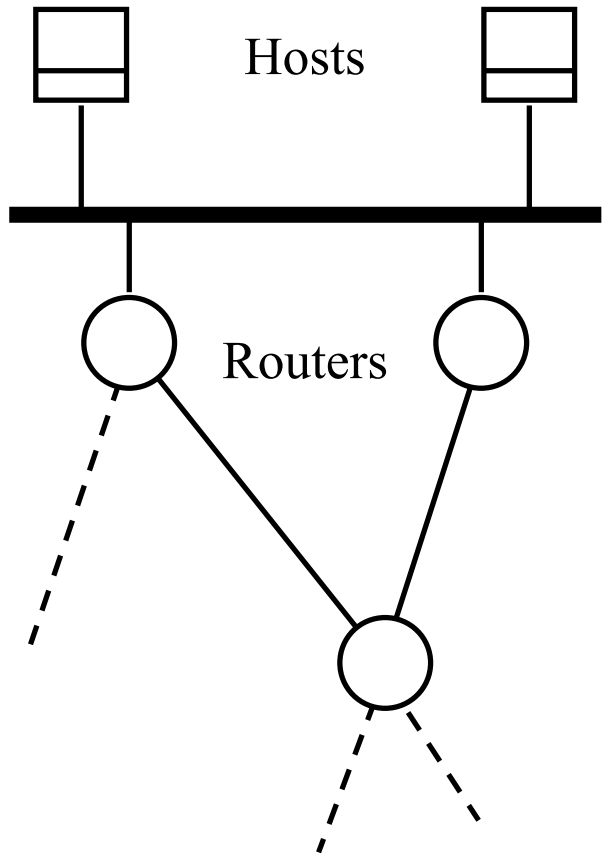
DR sends S,G join directly to sources in the source_list, and is not required to send *,G join to RP (see SSM discussion later)

IP Multicast Architecture

Service model

Host-to-router protocol
(IGMP)

Multicast routing protocols



Multicast Routing

- Basic objective – *build distribution tree for multicast packets*
  - The "leaves" of the distribution tree are the subnets containing at least one group member (detected by IGMP)

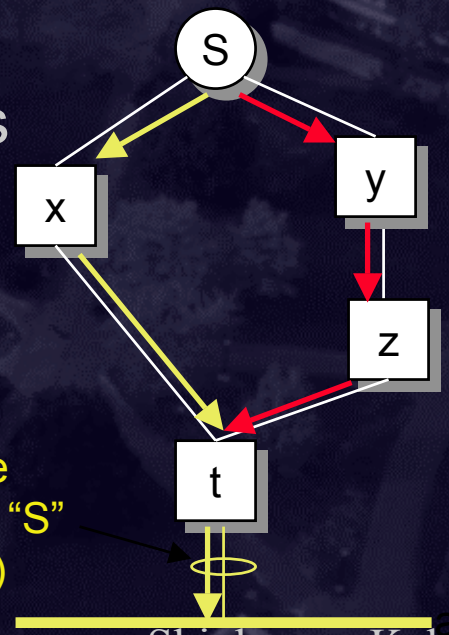
- Multicast service model makes it hard
  - *Anonymity*
  - *Dynamic join/leave*

Simple Mcast Routing Techniques

- Flood and prune
  - Begin by flooding traffic to entire network
  - Prune branches with no receivers
  - Examples: DVMRP, PIM-DM
  - *Unwanted state where there are no receivers*
- Link-state multicast protocols
  - Routers advertise groups for which they have receivers to entire network
  - Compute trees on demand
  - Example: MOSPF
  - *Unwanted state where there are no senders*

How to Flood Efficiently ?

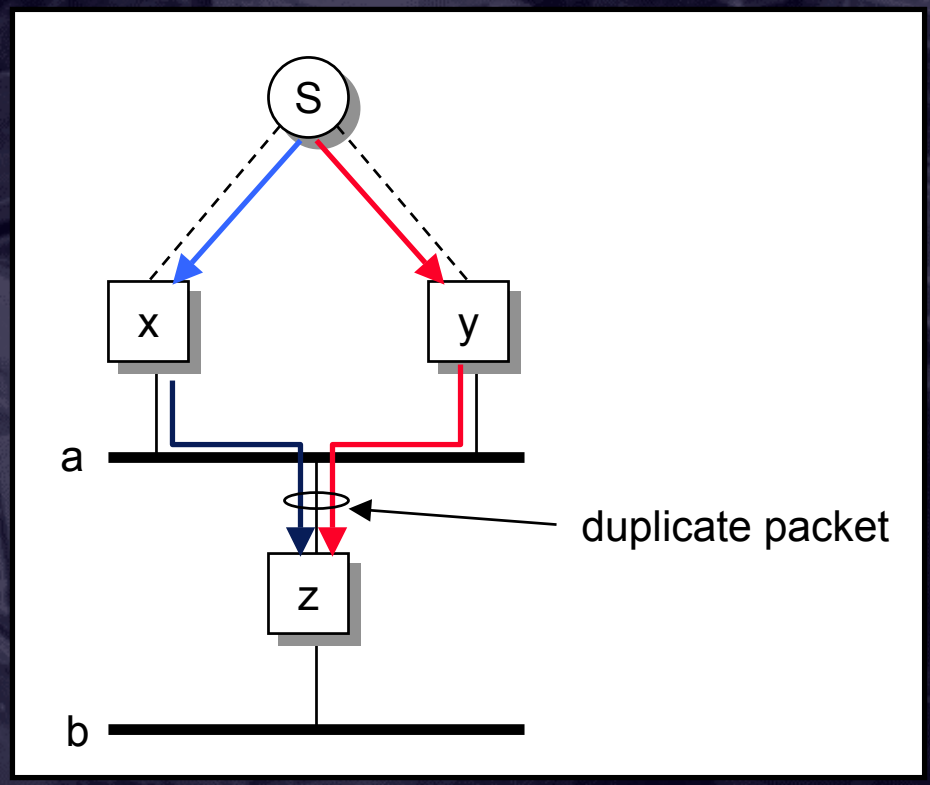
- A router forwards a packet from source (S) **iff** it arrives via the shortest path from the router back to S
  - Reverse path check!
- Packet is replicated out all but the incoming interface
- Reverse shortest paths easy to compute → just use info in DV routing tables
  - DV gives shortest reverse paths
  - Efficient if costs are symmetric



Forward packets that arrive on shortest path from "t" to "S" (assume symmetric routes)

Problem

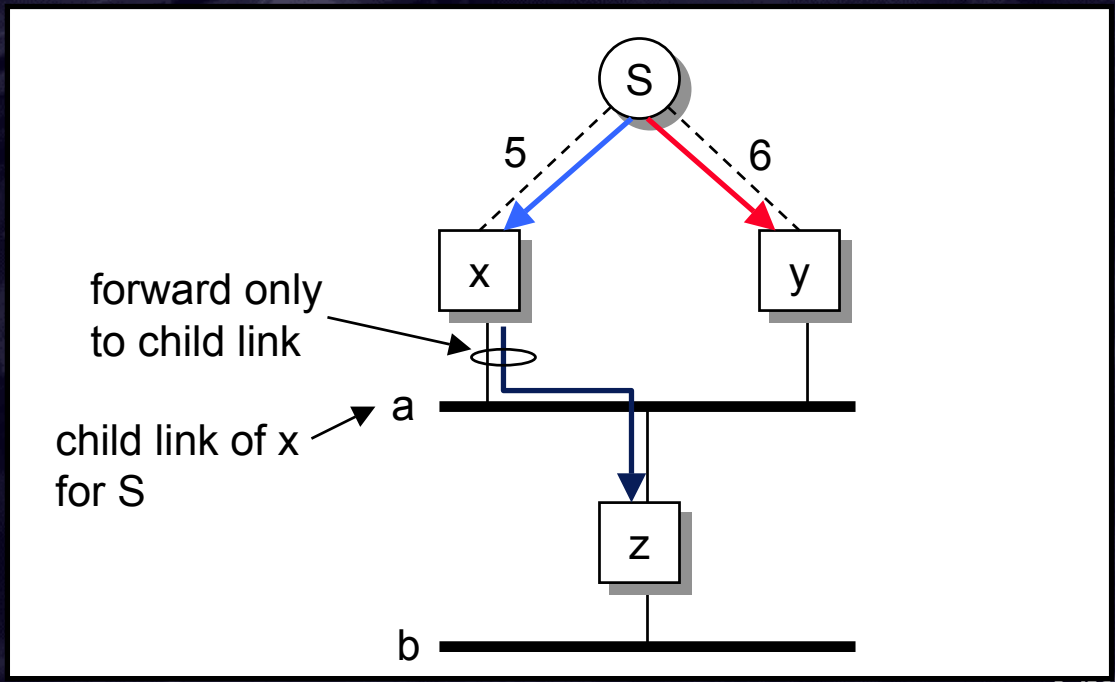
- Flooding can cause a given packet to be sent multiple times over the same link: can filter better than this!



- Solution: Reverse Path Broadcasting

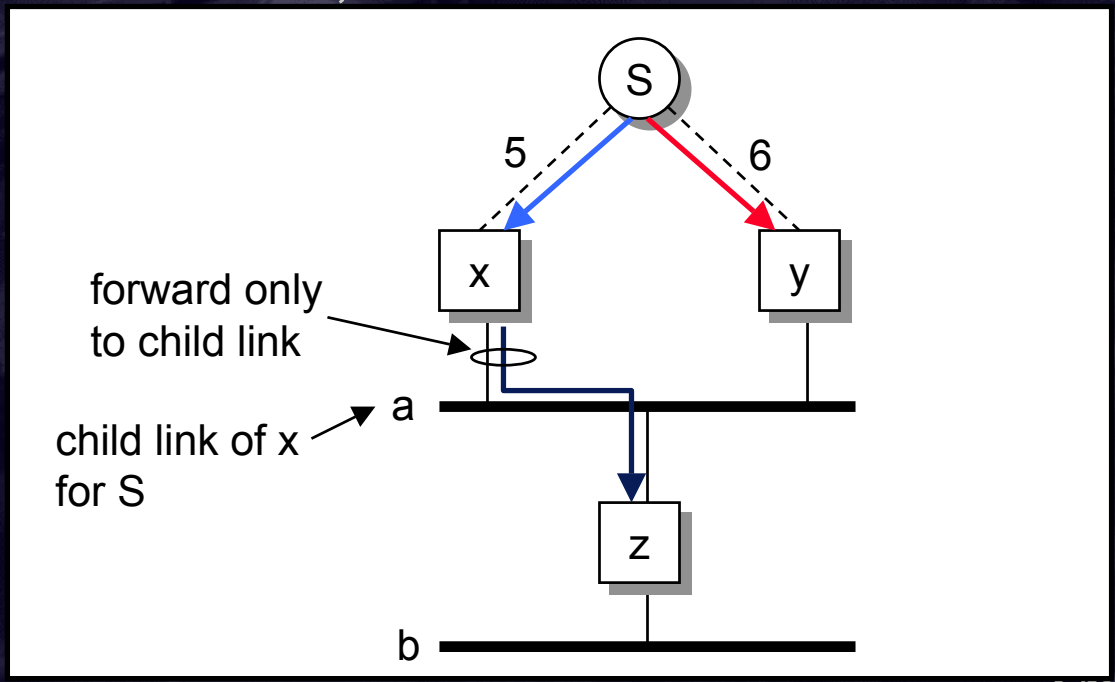
Reverse Path Broadcasting (RPB)

- Basic idea: forward a packet from S only on child links for S
- Child link of router x for source S: *link that has x as parent on the shortest path from the link to S*



How to Find Child Links?

- Routing updates !
  - If z tells x that it can reach S through y,
  - and if the cost of this path is >= the cost of the path from z to S through x,
  - then x knows that the link to z is a child link
- In case of tie, lower address wins



Problem

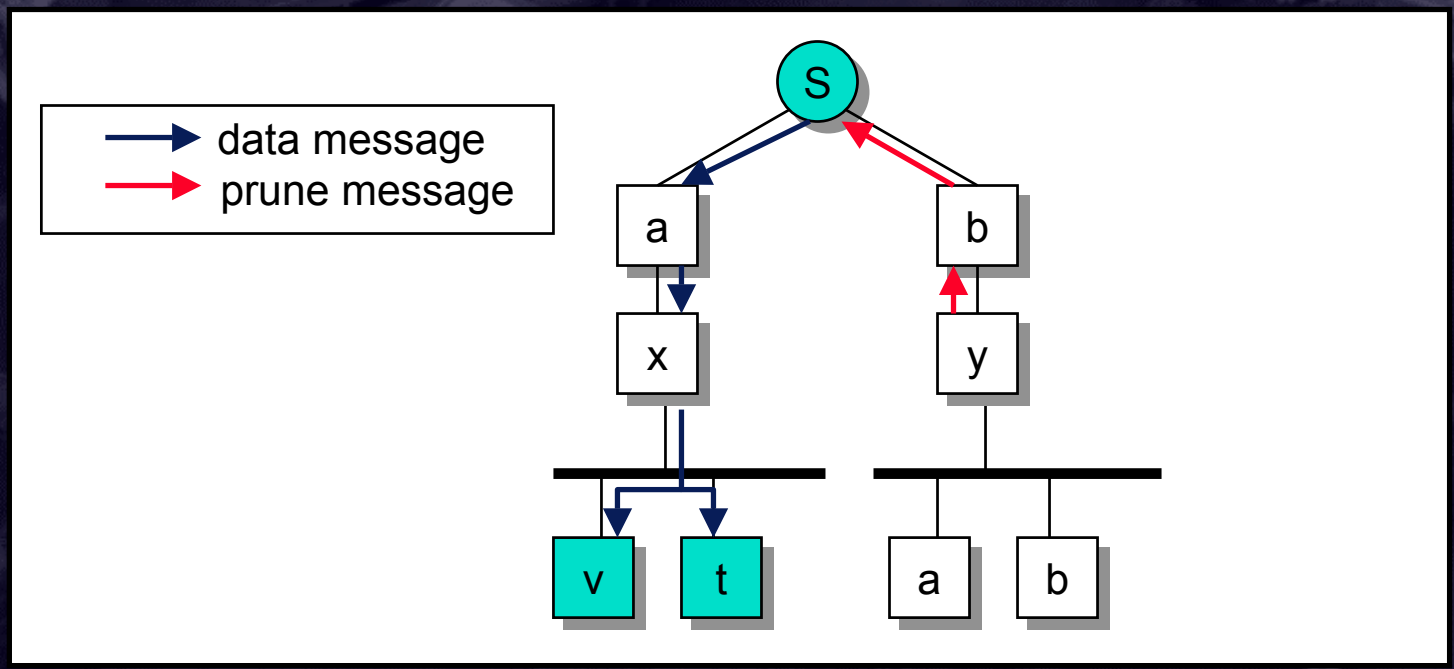
- This is still a broadcast algorithm – the traffic goes everywhere – lousy filtering!
- First order solution:
  - ***Truncated RPB***

Truncated RPB

- Don't forward traffic onto networks with no receivers
  1. Identify leaves (see below)
  2. Detect group membership in leaf (IGMP)
- Identify leaves
  - Leaf links are the child links that no other router uses to reach source S
  - Use periodic updates of form:
    - "this is my next-link to source S"
  - If child is not the "next-link" for anyone, it is a leaf

Reverse Path Multicast (RPM)

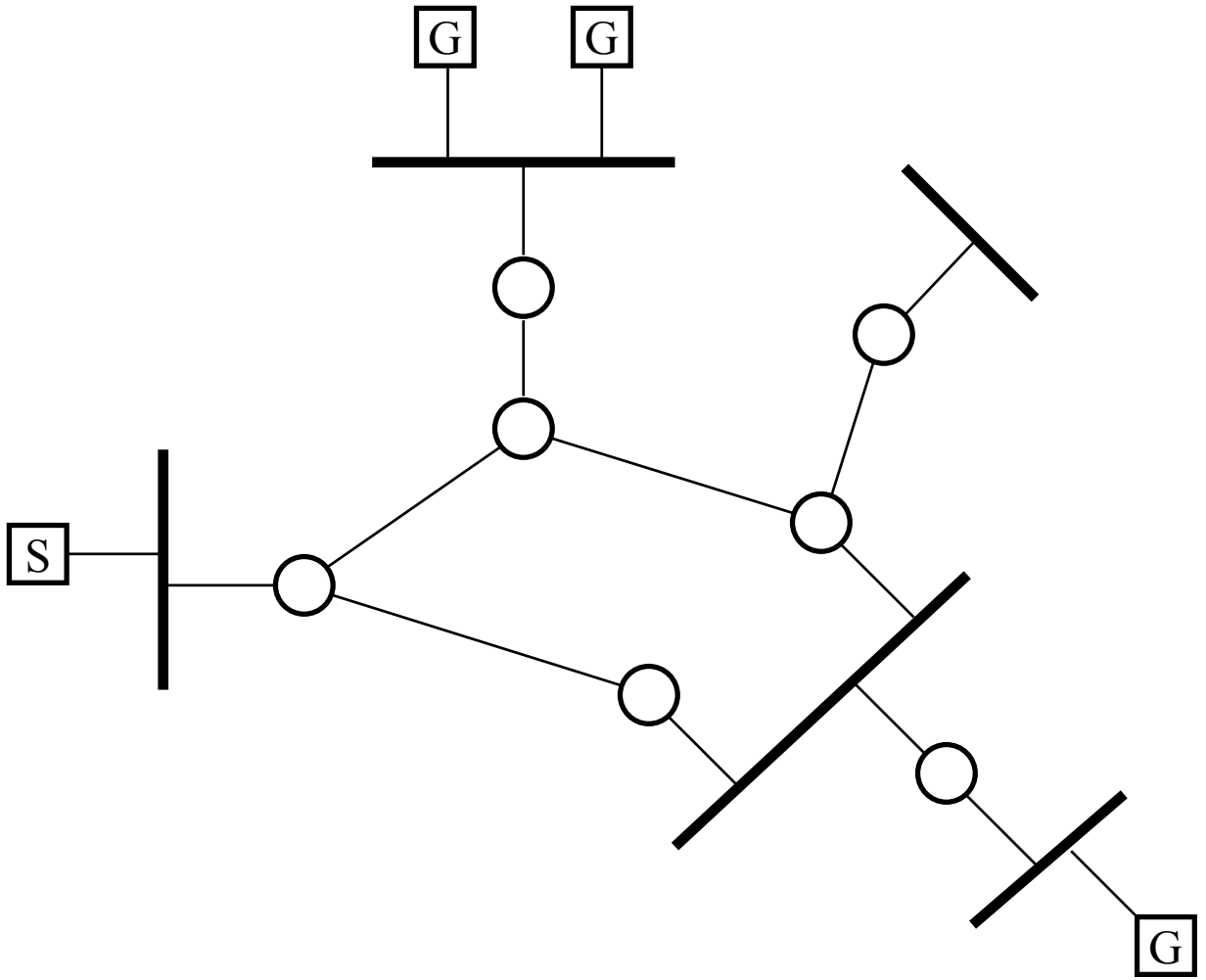
- Prune back transmission so that only absolutely necessary links carry traffic
- Use on-demand pruning so that router group state scales with number of active groups



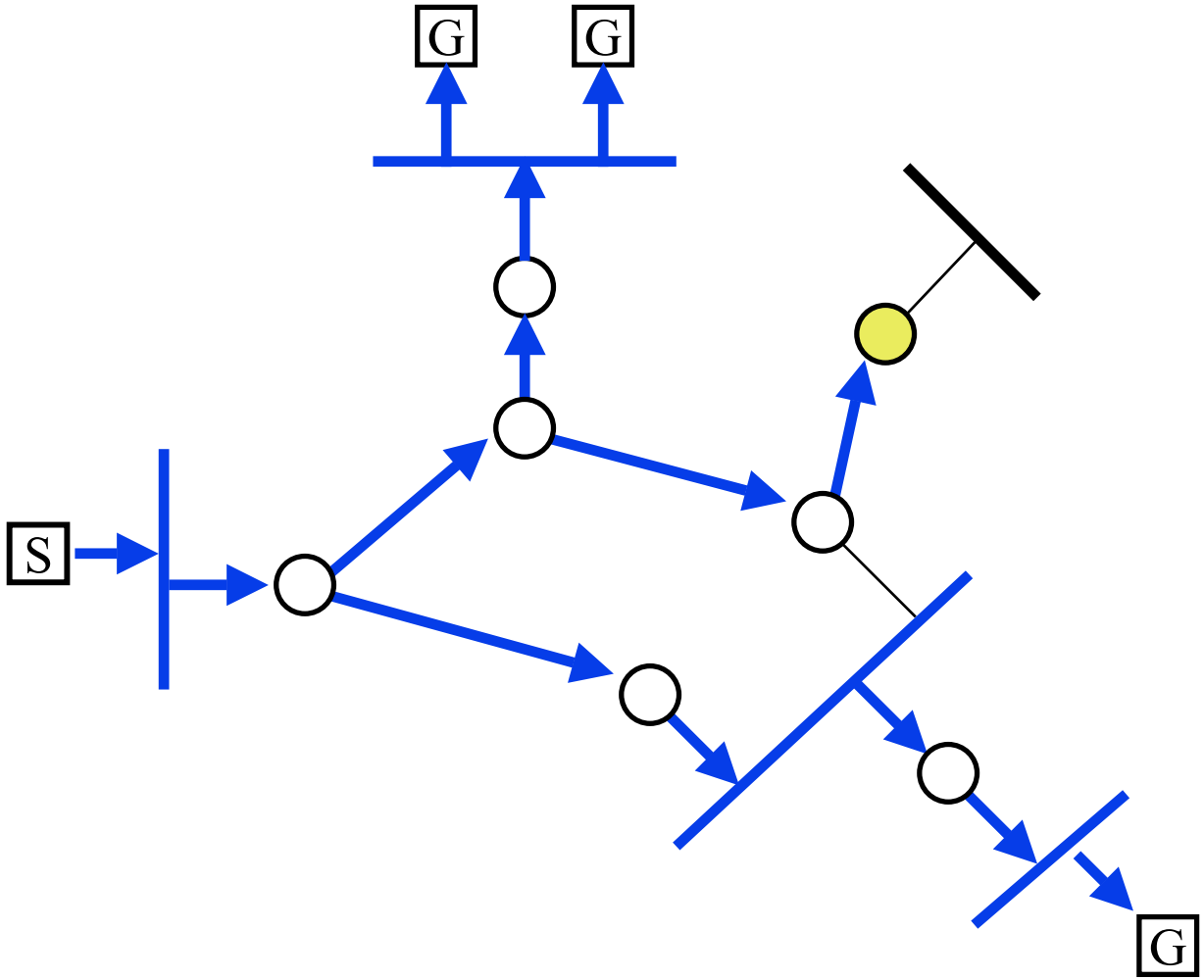
Basic RPM Idea

- Prune (Source,Group) at leaf if no members
  - Send Non-Membership Report (NMR) up the tree
- If all children of router R prune (S,G)
  - Propagate prune for (S,G) to parent R
- On timeout:
  - Prune dropped
  - Flow is reinstated
  - Down stream routers re-prune
  - Note: this is a soft-state approach
- Grafting: Explicitly reinstate sub-tree when
  - IGMP detects new members at leaf, or when a child asks for a graft.

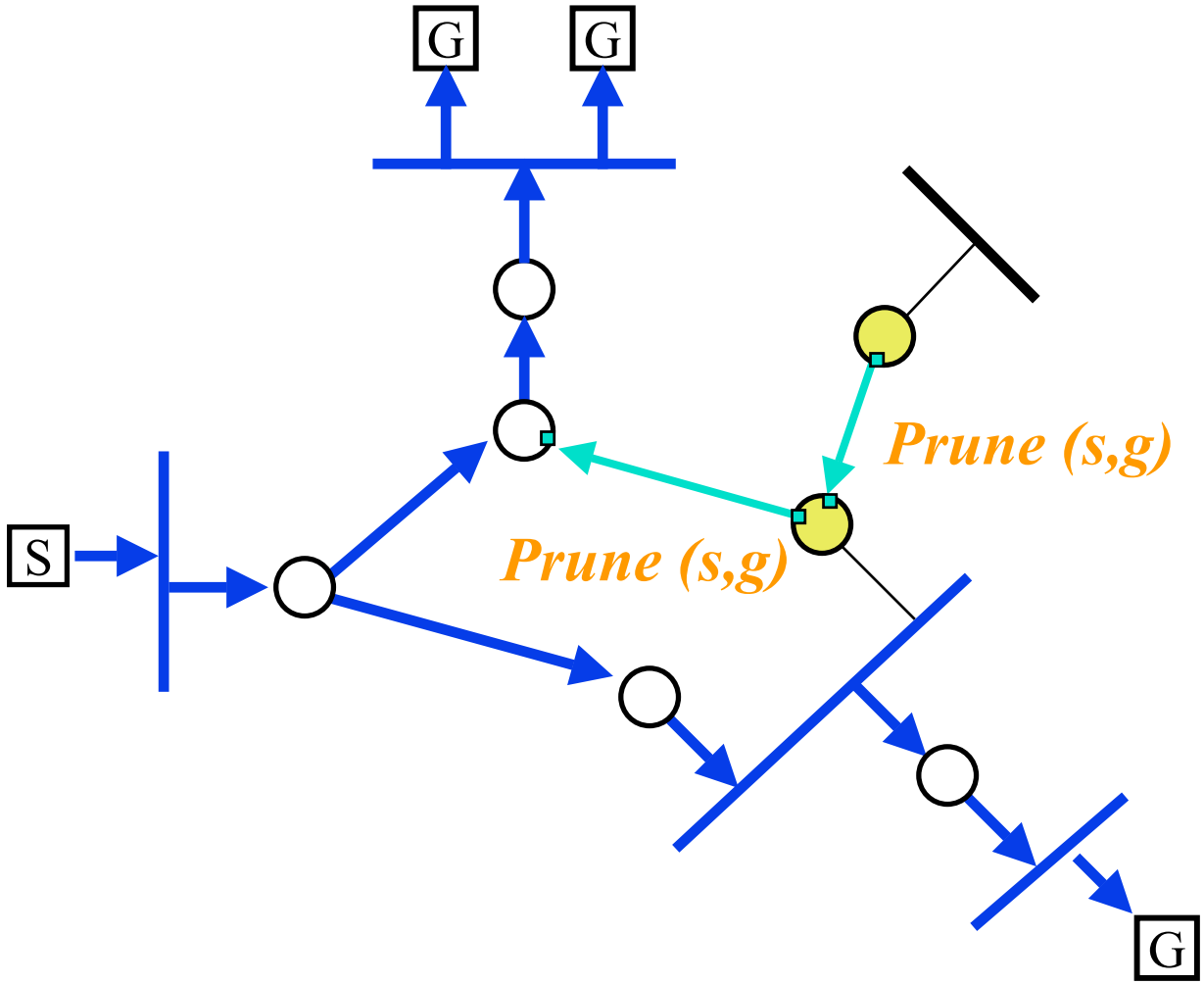
Putting it together: Topology



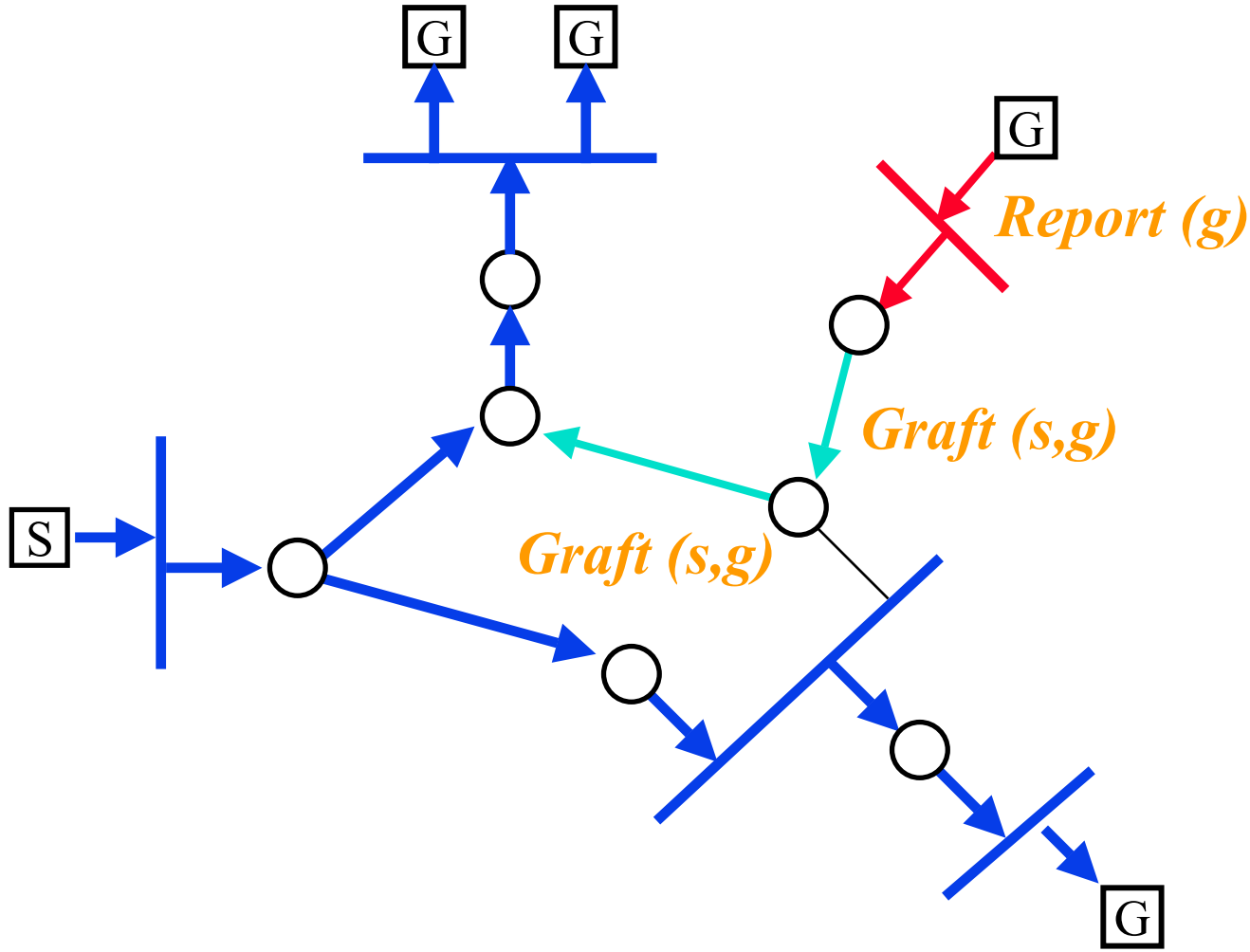
Flood with *Truncated* Broadcast



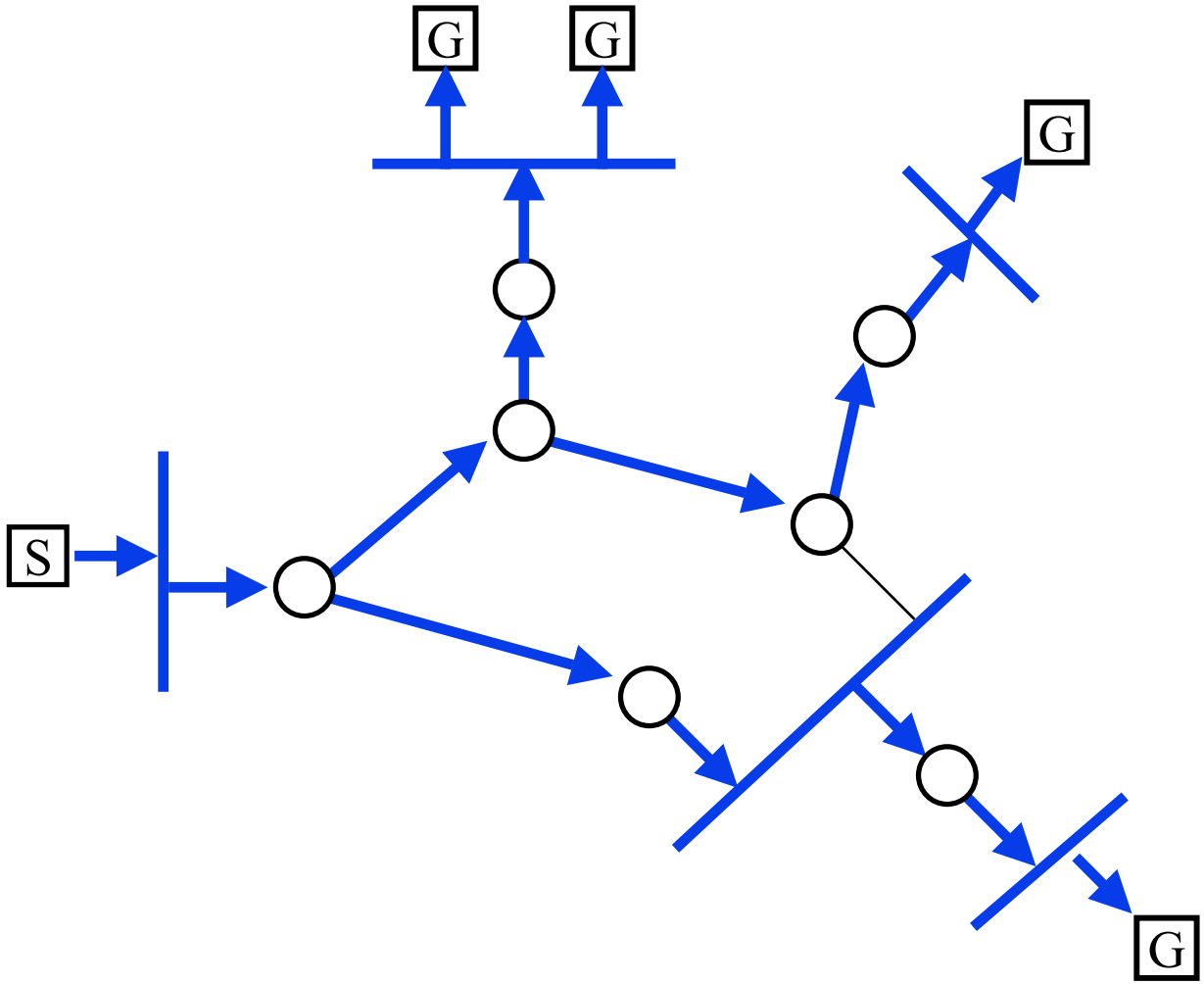
Pruning



Grafting



After Grafting Complete



Distance-Vector Multicast Routing

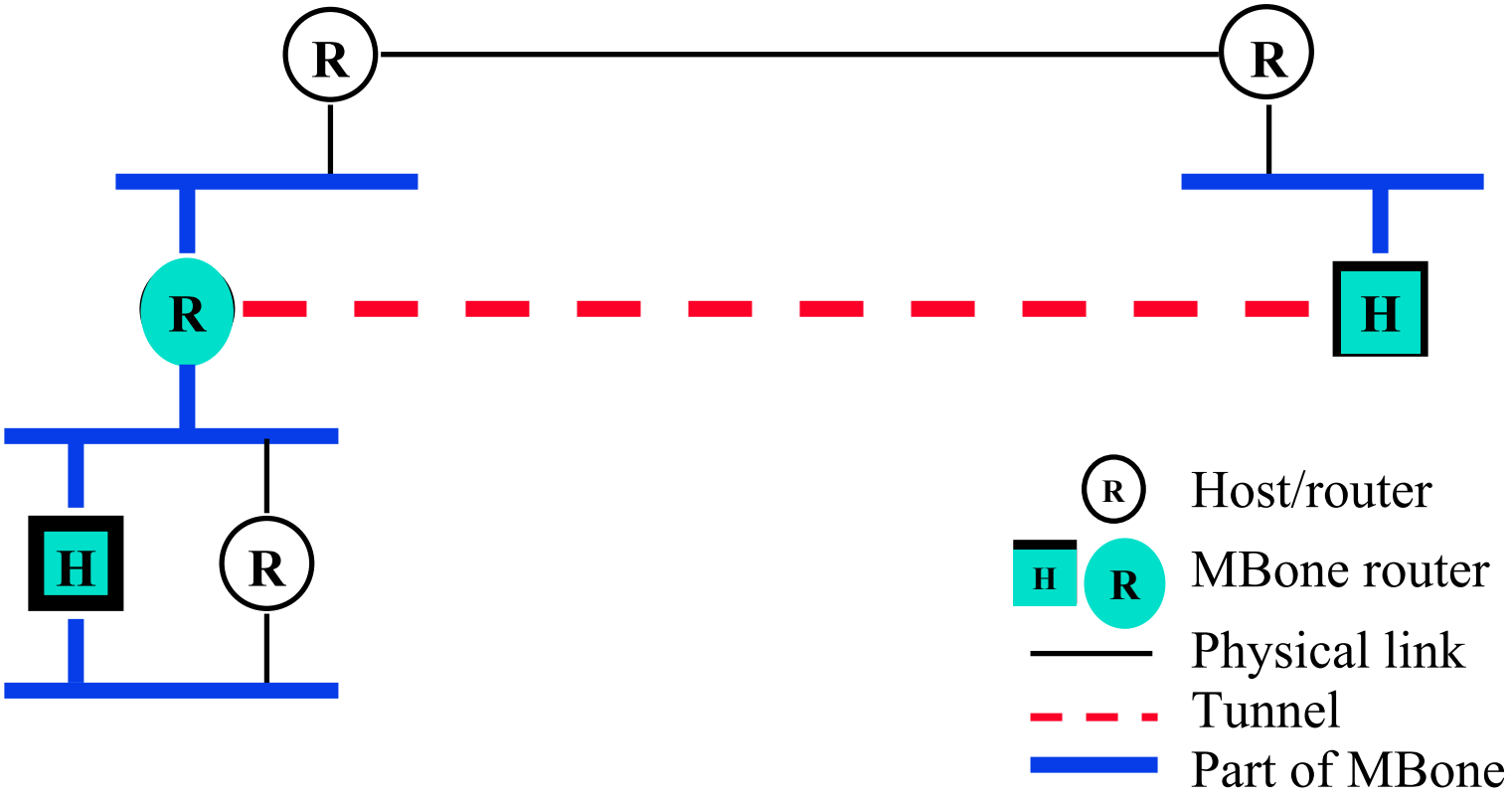
- DVMRP consists of two major components:
  - **A conventional distance-vector routing protocol (like RIP)**
  - **A protocol for determining how to forward multicast packets, based on the *unicast* routing table**
- DVMRP router forwards a packet if
  - **The packet *arrived from the link used to reach the source of the packet***
    - **Reverse path forwarding check – RPF**
  - **Packet forwarded ONLY to child links**
  - **If downstream links have not pruned the tree**

DVMRP limitations

- Like distance-vector protocols, affected by count-to-infinity and transient looping
  - Multicast trees more vulnerable than unicast !
- Shares the scaling limitations of RIP. New scaling limitations:
  - (S,G) state in routers: even in pruned parts!
  - Broadcast-and-prune has an initial broadcast.
  - Limited to few senders. Many small groups also undesired. Why ?
- No hierarchy: flat routing domain

Multicast Backbone (MBone)

- An *overlay network* of IP multicast-capable routers using DVMRP
- Tools: sdr (session directory), vic, vat, wb



MBone Tunnels

- A method for *sending multicast packets through multicast-ignorant routers*
- IP multicast packet is encapsulated in a unicast IP packet (IP-in-IP) addressed to far end of tunnel:

IP header, dest = unicast	IP header, dest = multicast	Transport header and data…

- Tunnel acts like a virtual point-to-point link
  - Intermediate routers see only outer header
  - Tunnel endpoint recognizes IP-in-IP (protocol type = 4) and de-capsulates datagram for processing
- Each end of tunnel is manually configured with unicast address of the other end

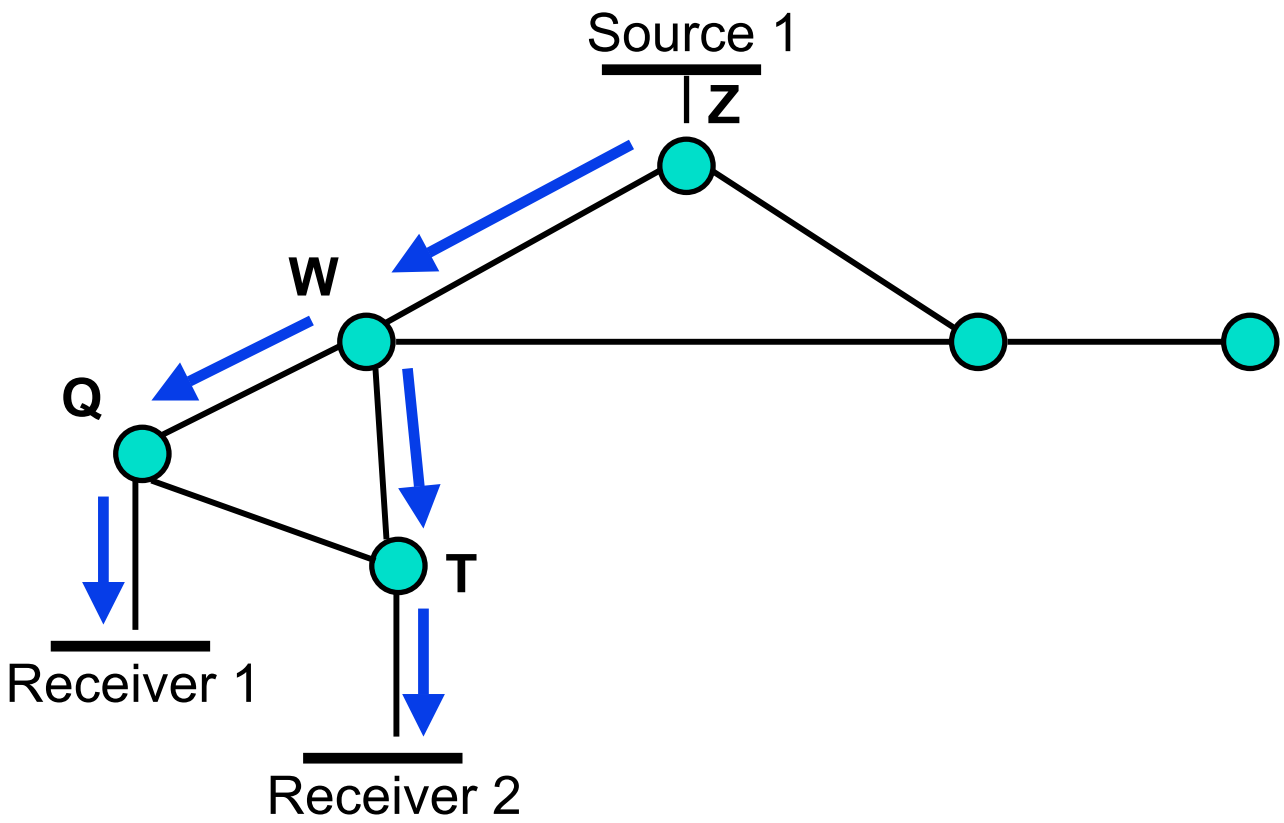
Simple Routing Techniques (recall)

- Flood and prune
  - Begin by flooding traffic to entire network
  - Prune branches with no receivers
  - Examples: DVMRP, PIM-DM
  - *Unwanted state where there are no receivers*
- Link-state multicast protocols
  - Routers advertise groups for which they have receivers to entire network
  - Compute trees on demand
  - Example: MOSPF
  - *Unwanted state where there are no senders*

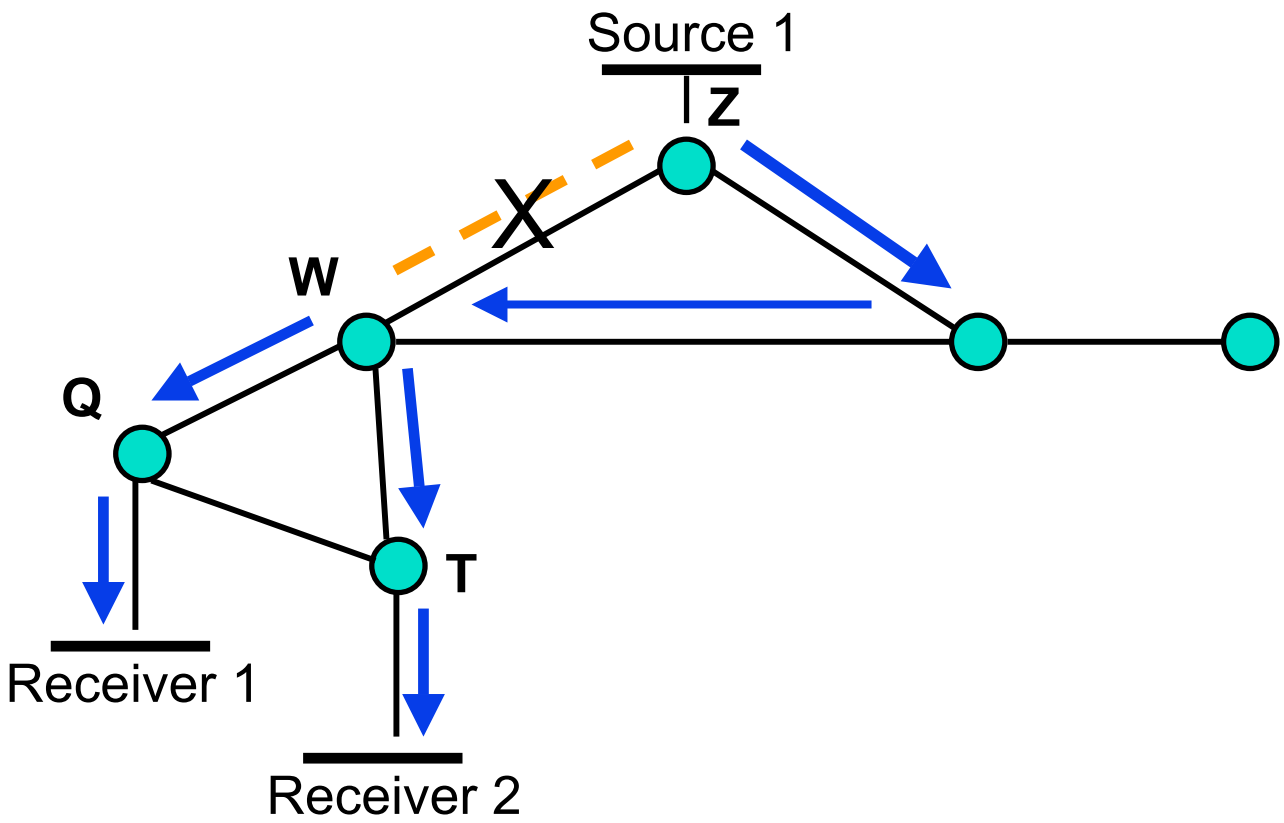
Multicast OSPF (MOSPF)

- Extend OSPF to support multicast
- Multicast-capable routers flag link state routing advertisements
- Link-state packets include multicast group addresses to which local members have joined
- Routing algorithm augmented to compute shortest-path distribution tree from a source to any set of destinations

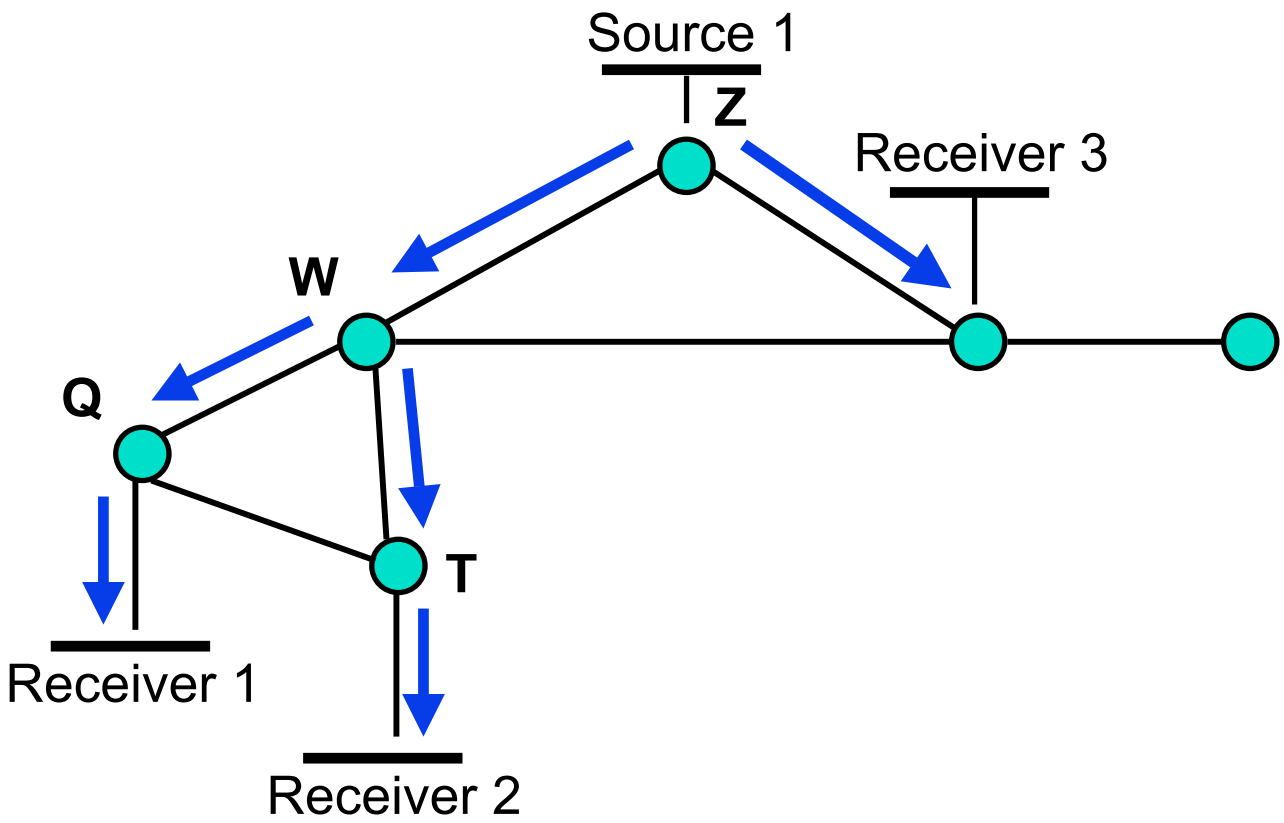
MOSPF: Example



Link Failure/Topology Change



Group Membership Change



MOSPF: Impact on Route Computation

- Can't pre-compute all source multicast trees
- Compute tree on-demand when first packet from a source S to a group G arrives
- New link-state advertisement
  - May lead to addition or deletion of outgoing interfaces if it contains different group addresses
  - May lead to re-computation of entire tree if links are changed

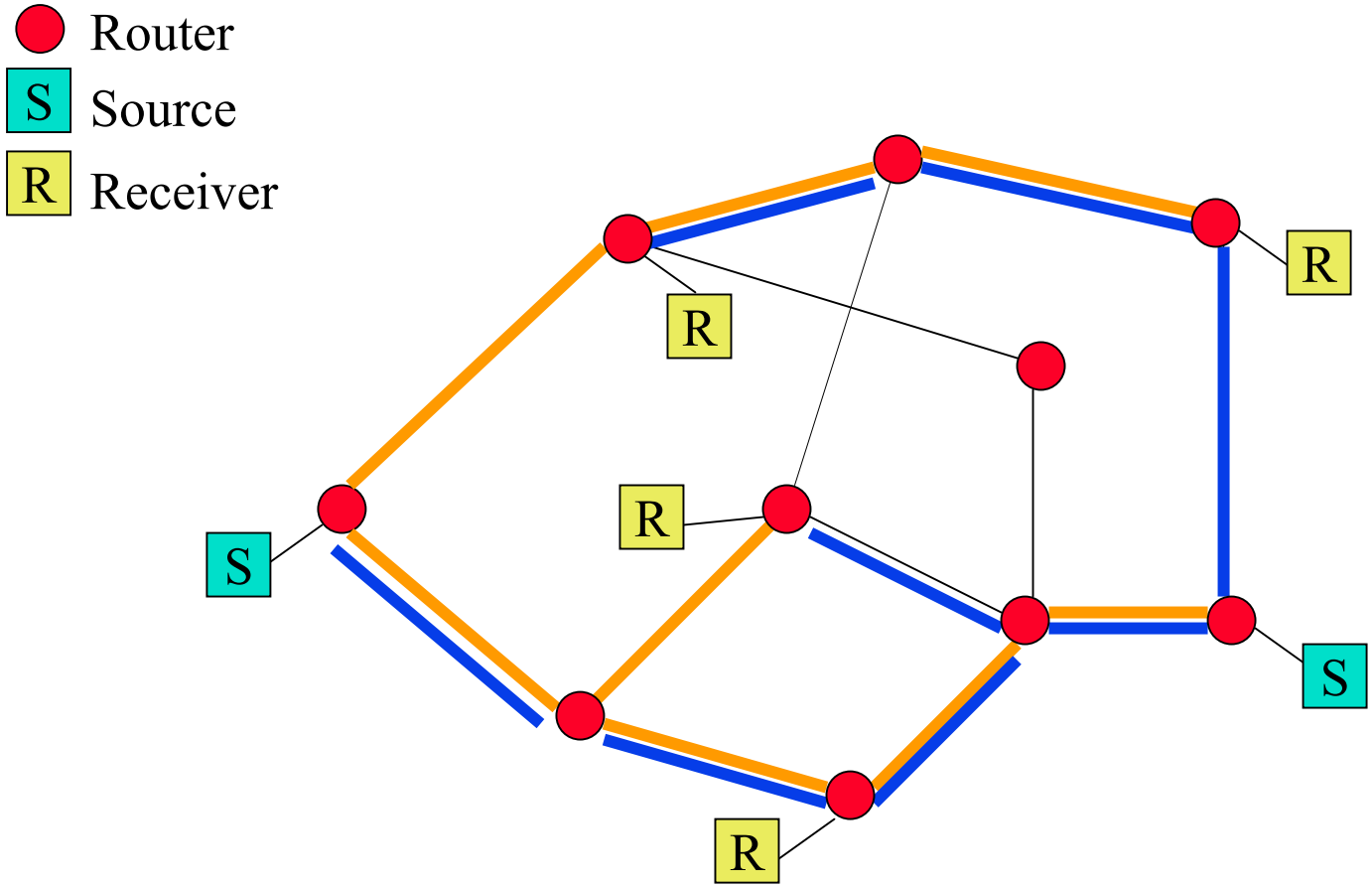
Routing Techniques (taxonomy)

- Tree building methods:
  - ***Data-driven*:** calculate the tree only when the first packet is seen. Eg: DVMRP, MOSPF
  - ***Control-driven*:** Build tree in background before any data is transmitted. Eg: CBT
- Join-styles:
  - ***Explicit-join:*** The leaves explicitly join the tree. Eg: CBT, PIM-SM
  - ***Implicit-join:*** All subnets are assumed to be receivers unless they say otherwise (eg via tree pruning). Eg: DVMRP, MOSPF

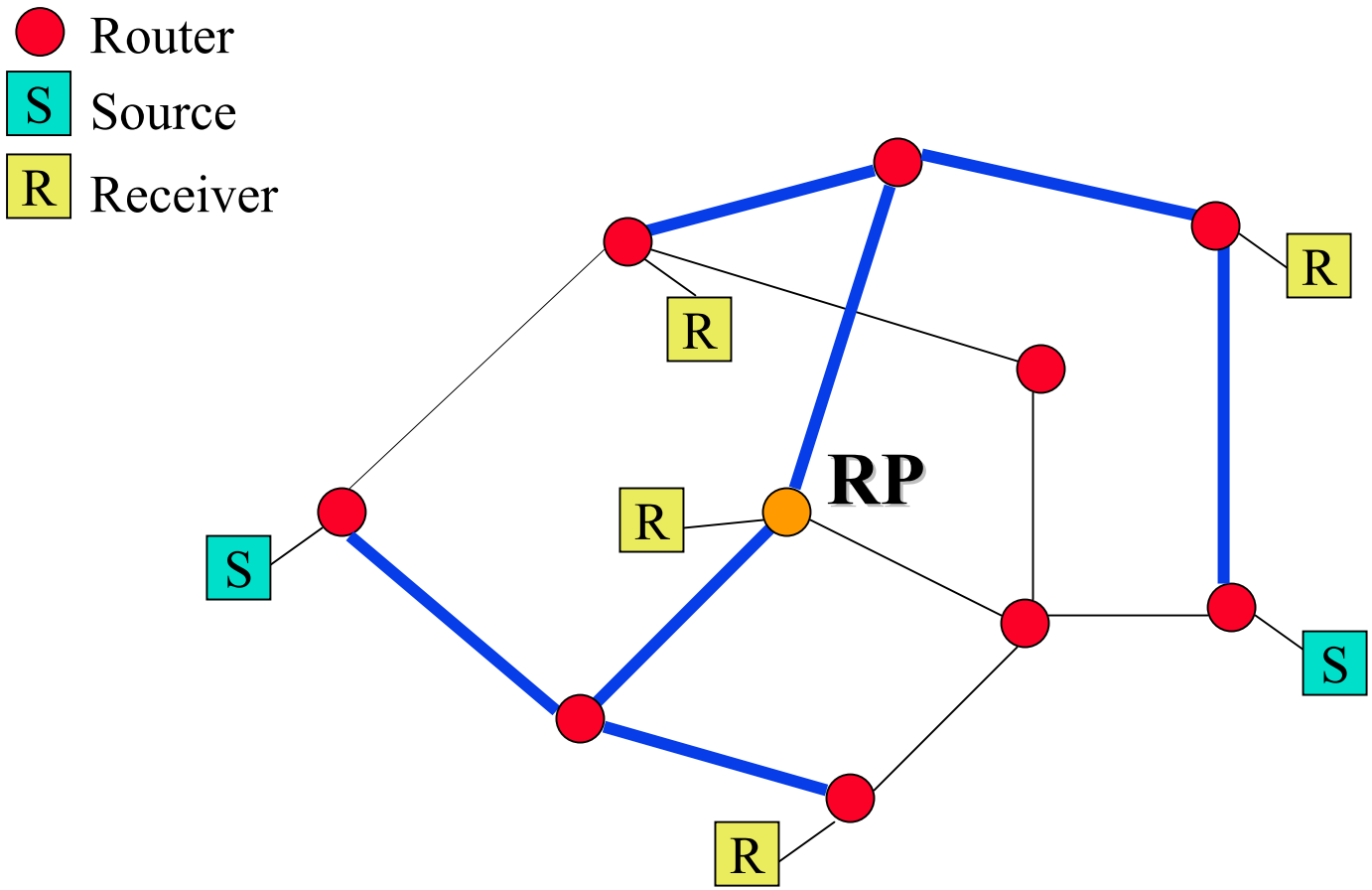
Shared vs. Source-based Trees

- Source-based trees
  - Separate shortest path tree *for each sender*
  - (S,G) state at intermediate routers
  - Eg: DVMRP, MOSPF, PIM-DM, PIM-SM
- Shared trees
  - *Single tree shared by all members*
  - Data flows on same tree regardless of sender
  - (*,G) state at intermediate routers
  - Eg: CBT, PIM-SM

Source-based Trees



A Shared Tree



Shared vs. Source-Based Trees

- **Source-based trees**
  - Shortest path trees – low delay, better load distribution
  - More state at routers (per-source state)
  - Efficient in *dense-area* multicast

- **Shared trees**
  - Higher delay (bounded by factor of 2), traffic concentration
  - Choice of core affects efficiency
  - Per-group state at routers
  - Efficient for *sparse-area* multicast

Core-based Routing Protocols

- Specify “meeting place” aka “core” or “rendezvous point (RP)”
- Sources send initial packets to core
- Receivers join group at core
- Requires mapping between multicast group address and “meeting place”
- Examples: CBT, PIM-SM

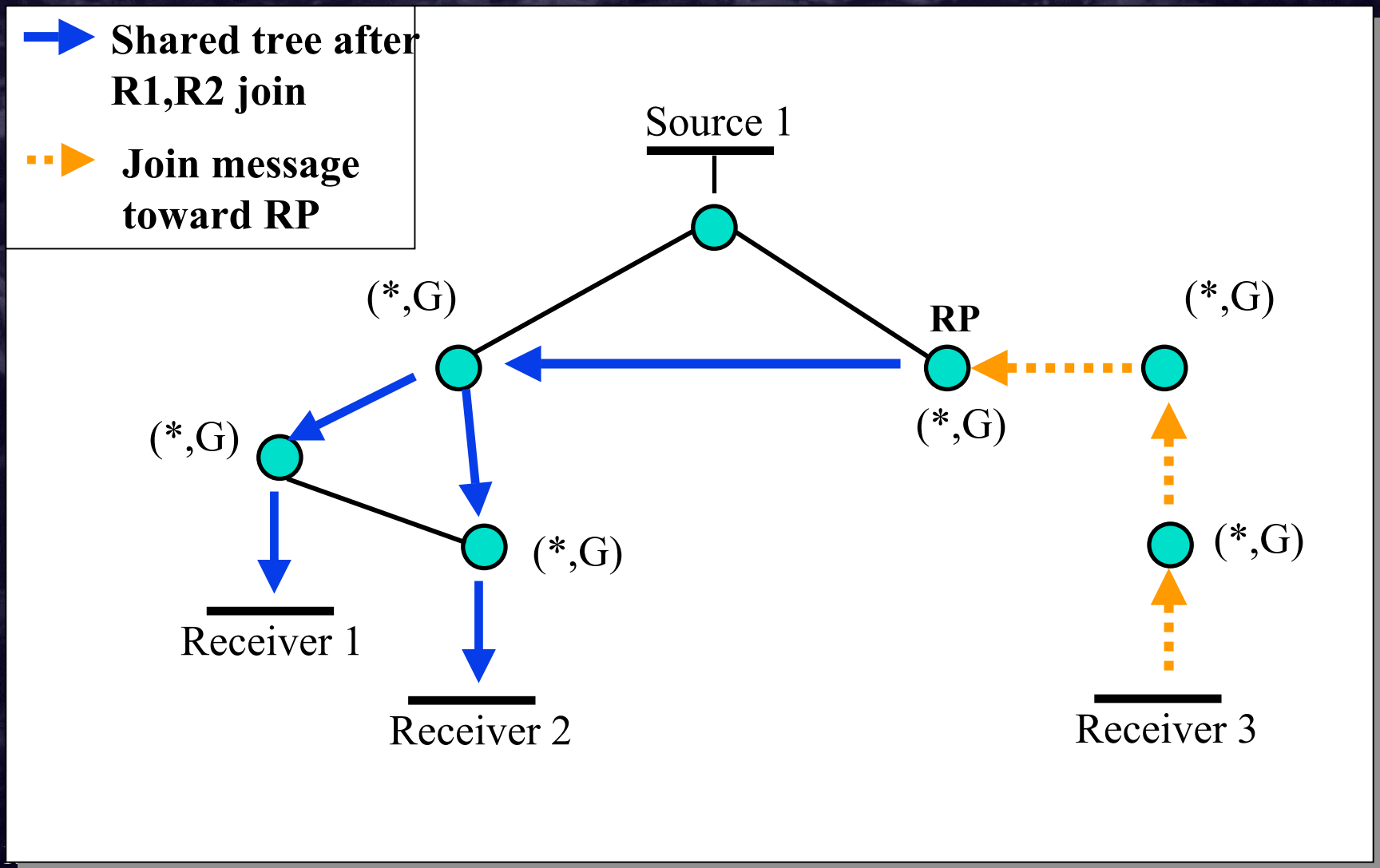
Protocol Independent Multicast (PIM)

- Support for both shared and per-source trees
- Dense mode (per-source tree)
  - Similar to DVMRP
- Sparse mode (shared tree)
  - Core = rendezvous point (RP)
- Independent of unicast routing protocol
  - Just uses unicast forwarding table

PIM Protocol Overview

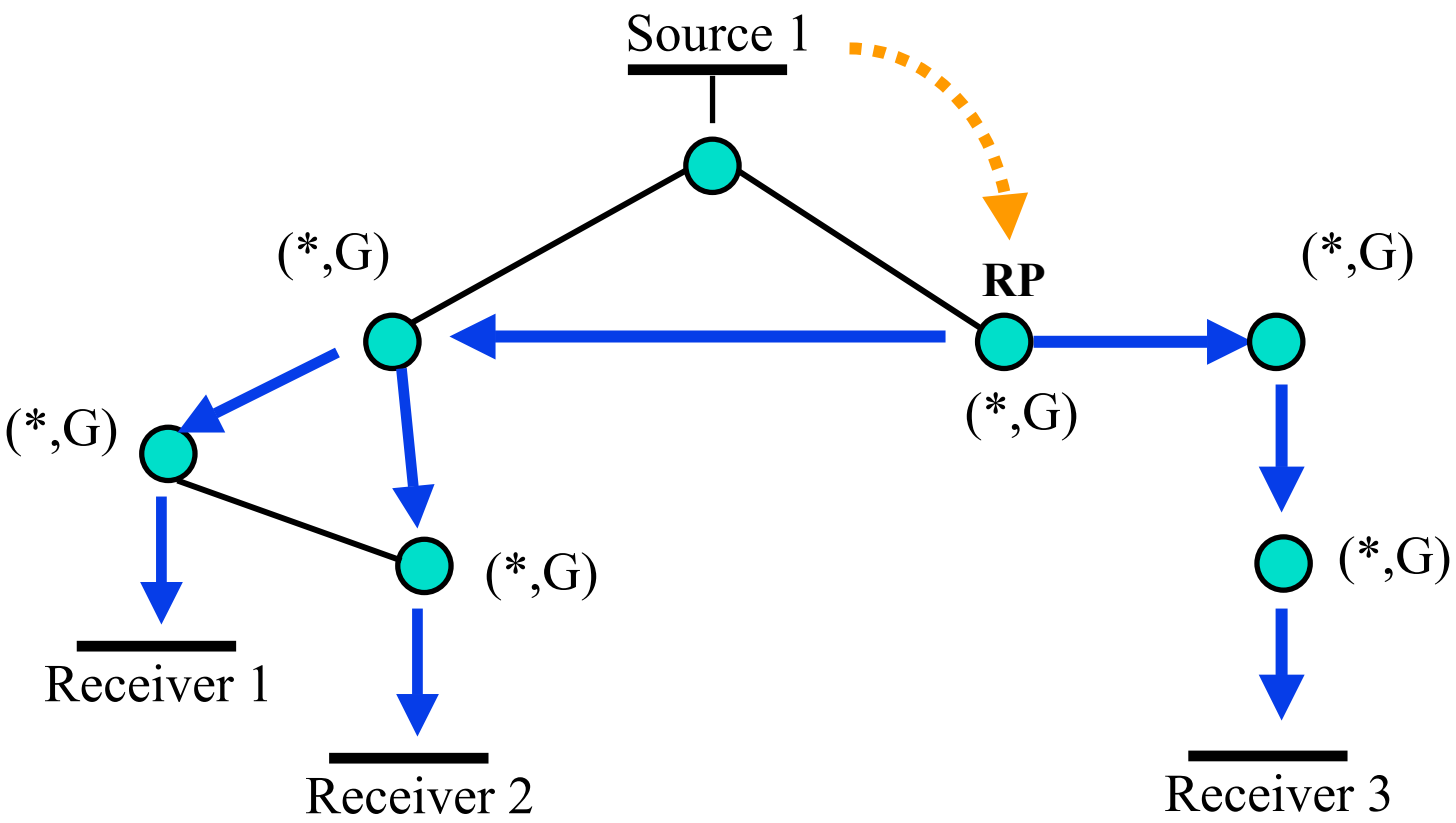
- Basic protocol steps
  - Routers with local members Join toward Rendezvous Point (RP) to join shared tree
  - Routers with local sources encapsulate data in Register messages to RP
  - Routers with local members may initiate data-driven switch to source-specific shortest path trees
- PIM v.2 Specification (RFC2362)

PIM Example: Build Shared Tree



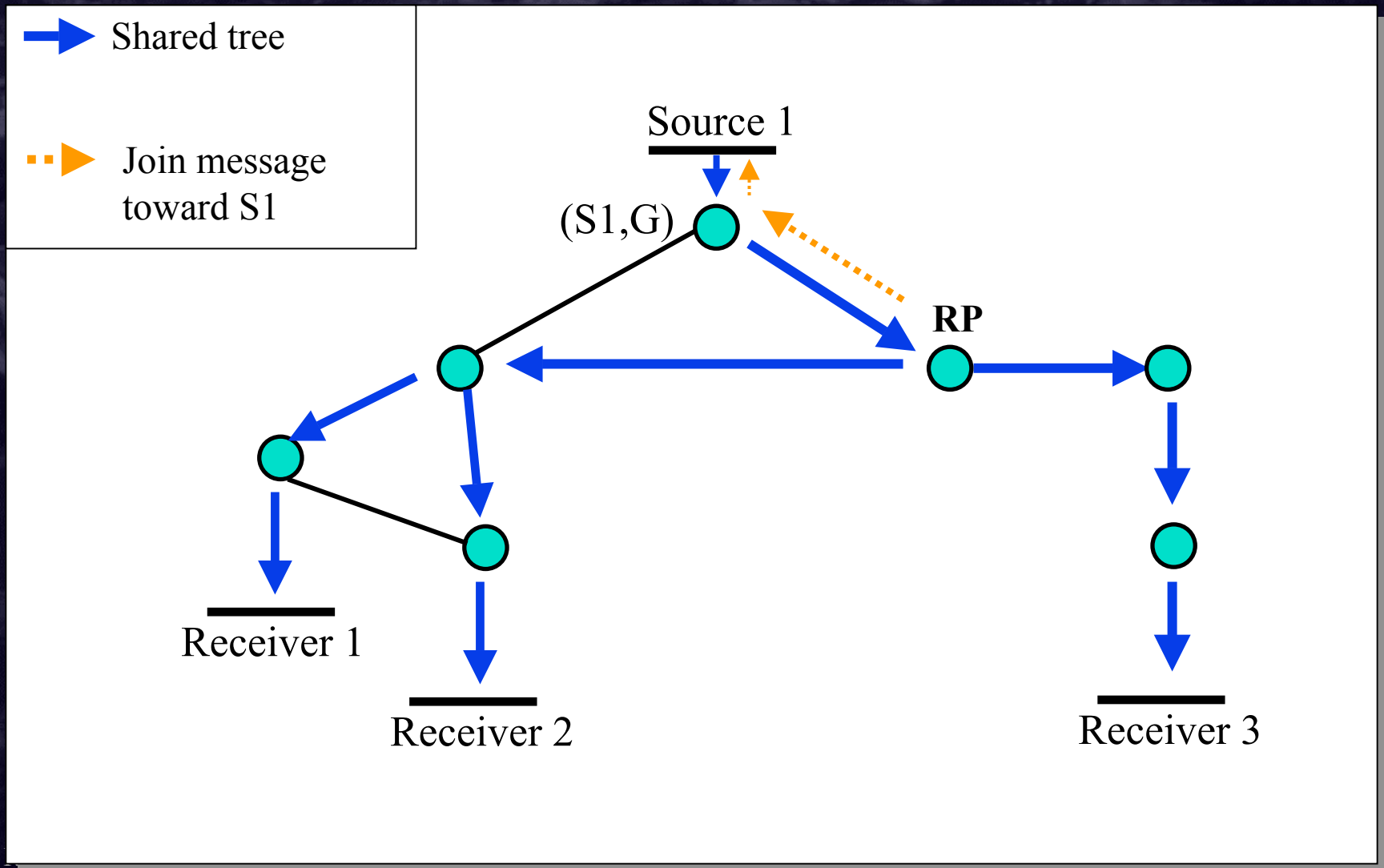
Data Encapsulated in Register

Unicast encapsulated data packet to RP in **Register**

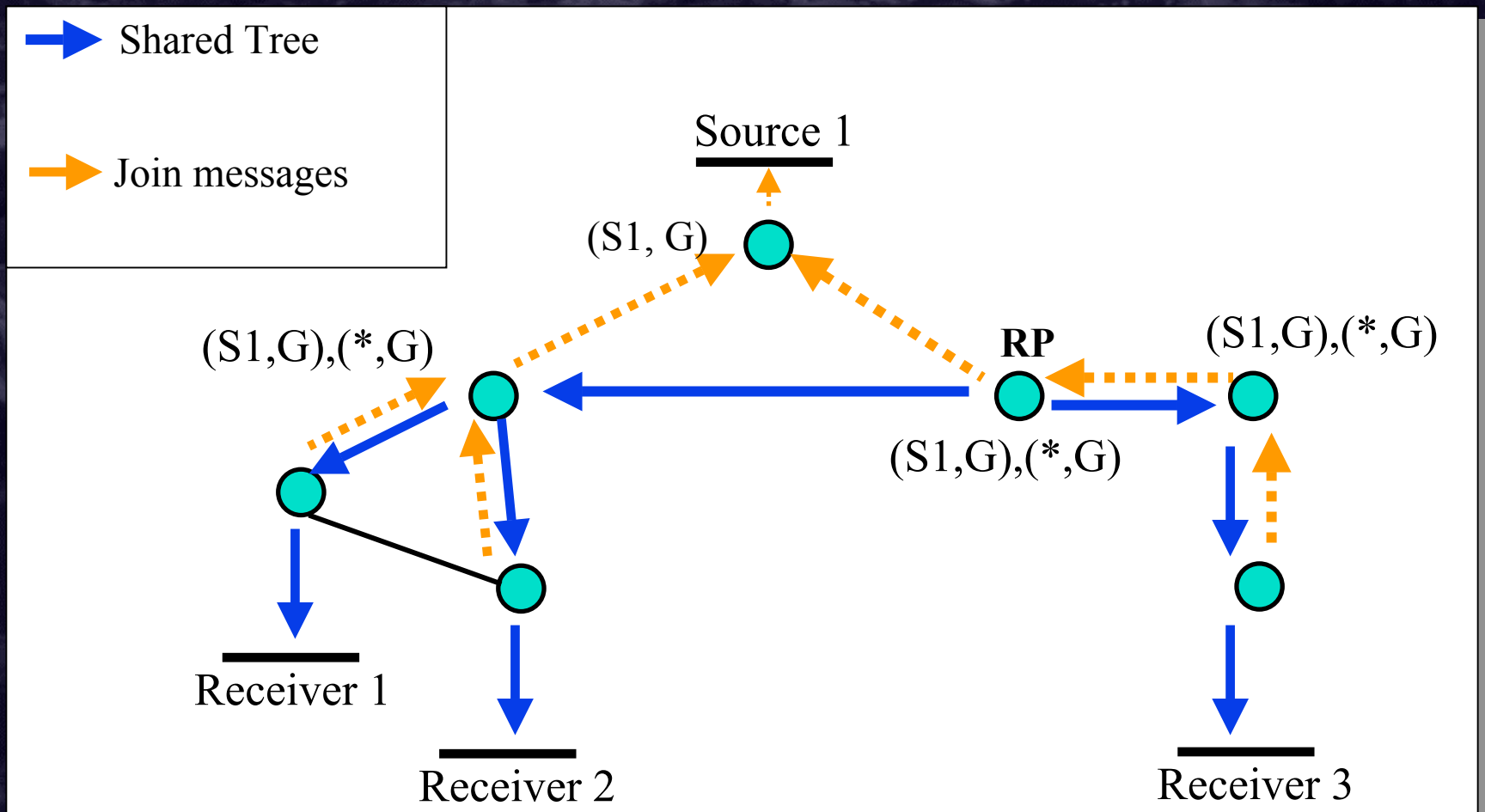


RP de-capsulates, forwards down shared tree

RP Send Join to High Rate Source

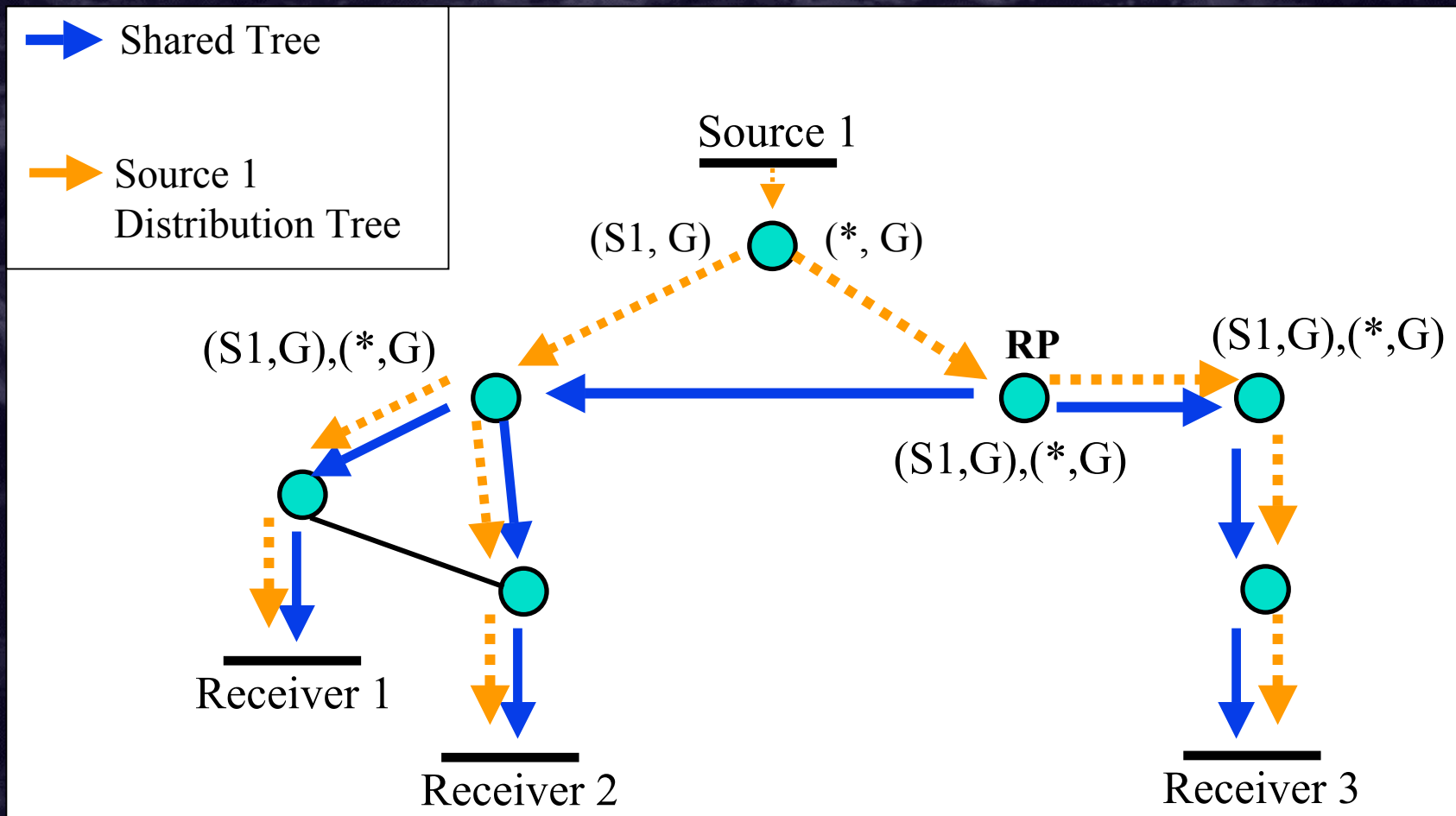


Build Source-Specific Distribution Tree



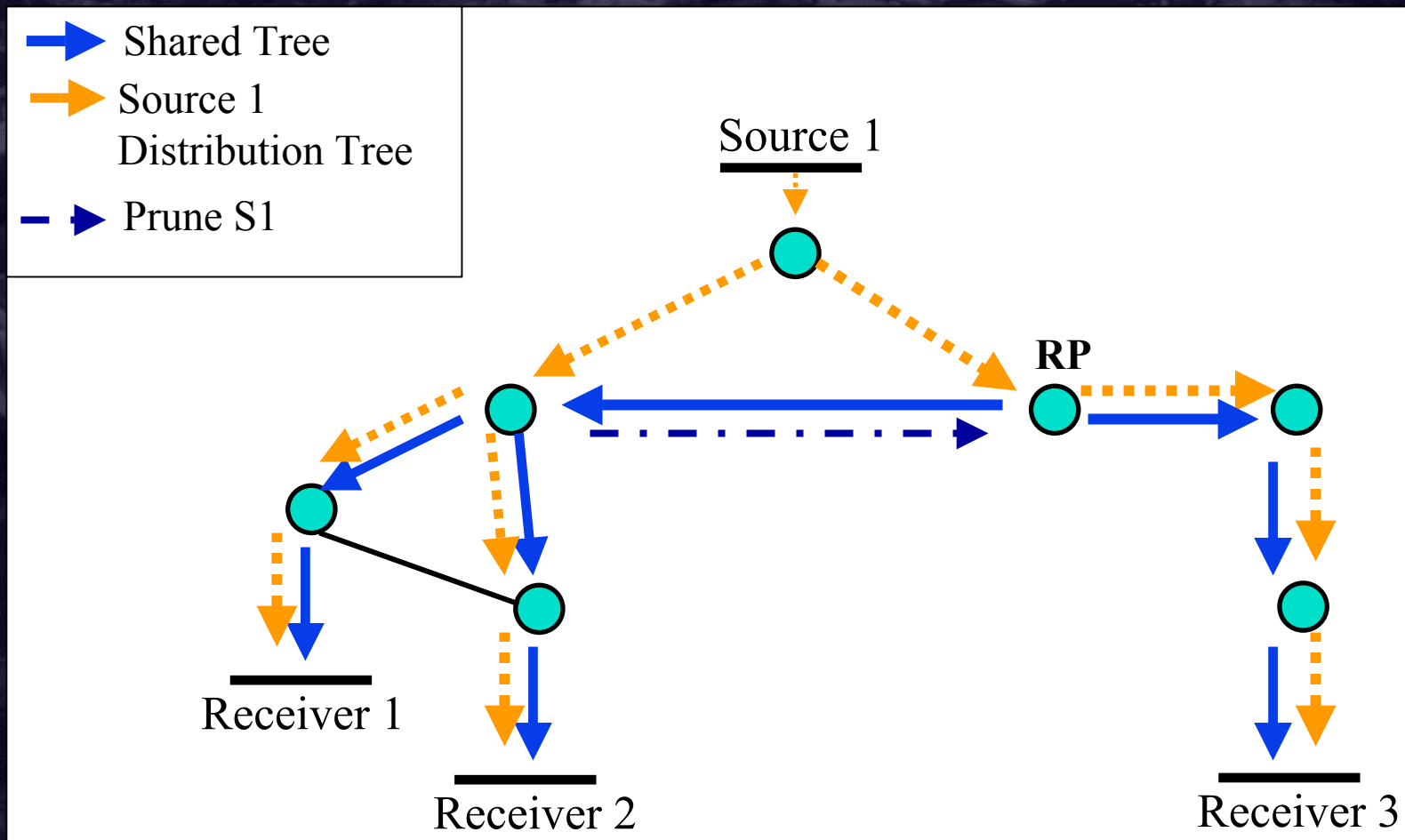
Build source-specific tree for high data rate source

Forward On "Longest-match" Entry



Source-specific entry is "longer match" for source S1 than is Shared tree entry that can be used by any source

Prune S1 off Shared Tree



Prune S1 off shared tree where if S1 and RP entries differ

Multicast Address Allocation

- Unicast IP addresses
  - Allocated statically to hosts
  - Allocated hierarchically by hosts' org
- Multicast IP addresses
  - Allocated per session
  - Groups cross admin boundaries

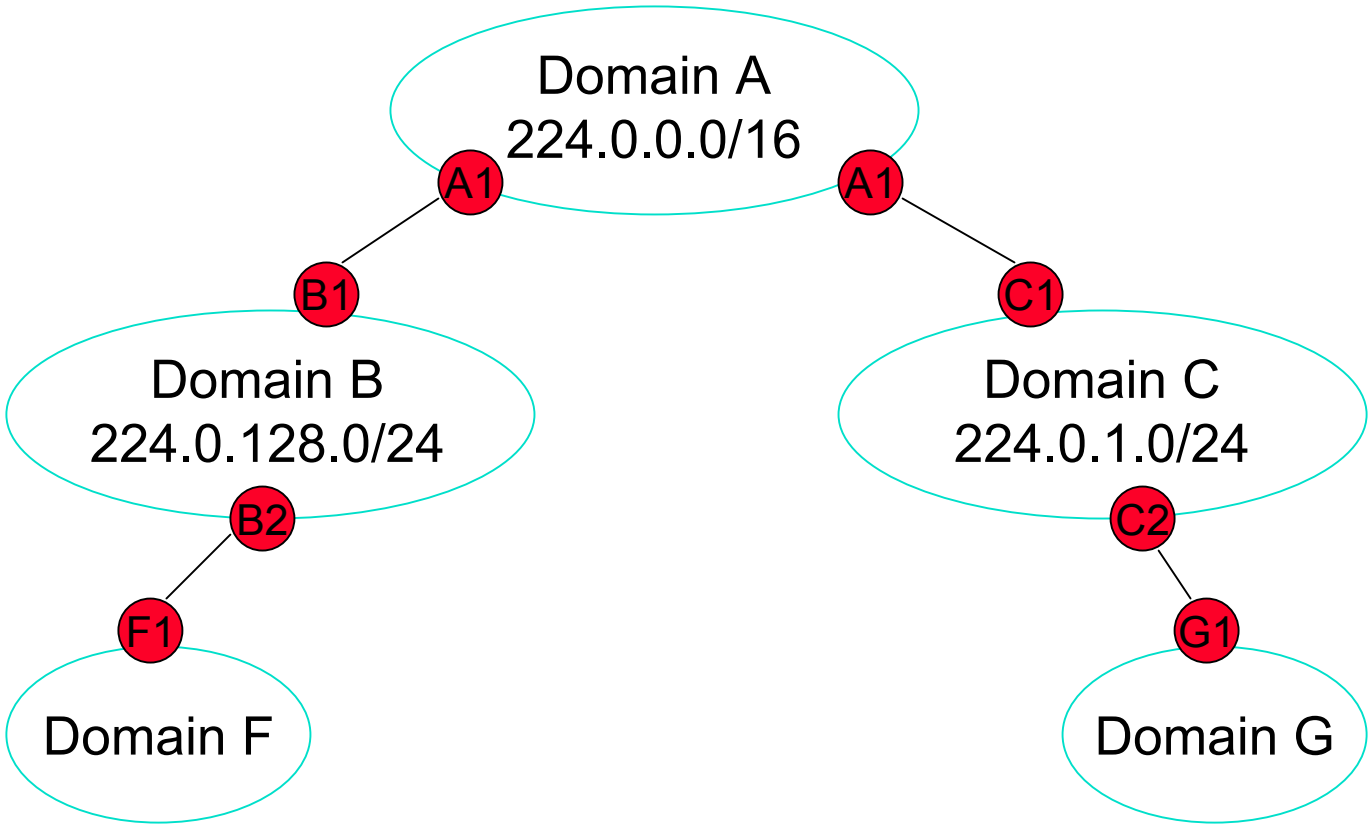
Address Allocation Solutions

- Central server?
  - Blocks of addresses for internal groups
  - Leases for global addresses
- Random allocation?
  - Send to group to ask if address is in use
  - Need conflict resolution protocol

Multicast Address-Set Claim (MASC)

- Address allocation goals
  - Efficient utilization of address space
  - Aggregation of routing entries
  - Robust and scalable
- Distributed, collaborative allocation
  - Dynamic claim and listen protocol
  - Claim “prefixes” from parent
  - Communicate prefixes to local addresss allocation

MASC


Domain A
224.0.0.0/16
A1
A1
B1
C1
Domain B
224.0.128.0/24
Domain C
224.0.1.0/24
B2
C2
F1
G1
Domain F
Domain G

Prefix Selection

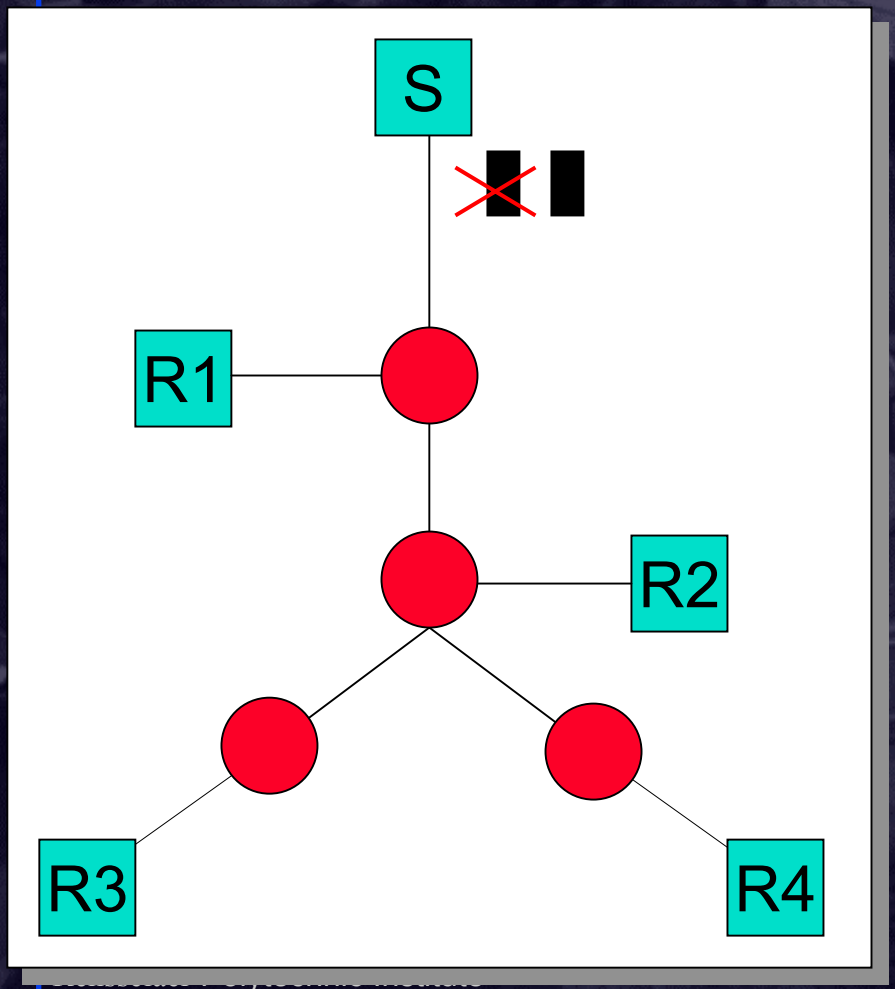
- Collect available prefixes
- Remove those that have been claimed
- Randomly select one of remaining prefixes with shortest mask (largest range)
- Claim first sub-prefix of desired size

Reliable Multicast: The Goal

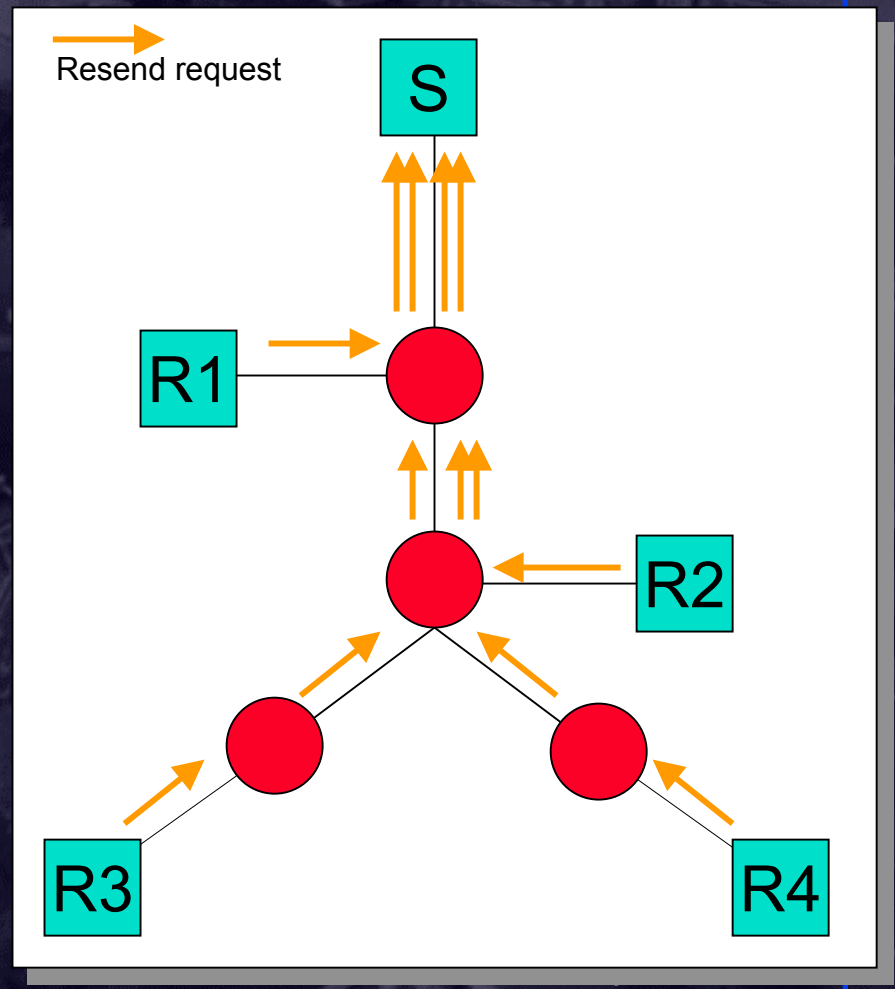
- Implement reliability on top of IP multicast
  - Don't necessarily care about ordering or byte-stream abstraction
- Why is this hard ?
  - Sender cannot keep state for unknown number of dynamic receivers
    - Remember open & dynamic group semantic?
  - Algorithms like TCP that estimate path properties such as RTT and congestion window don't generalize to trees.
    - Remember: TCP is only for a unicast session!
  - Has to address (N)ACK implosions

Implosion

Packet 1 is lost



All 4 receivers request a resend

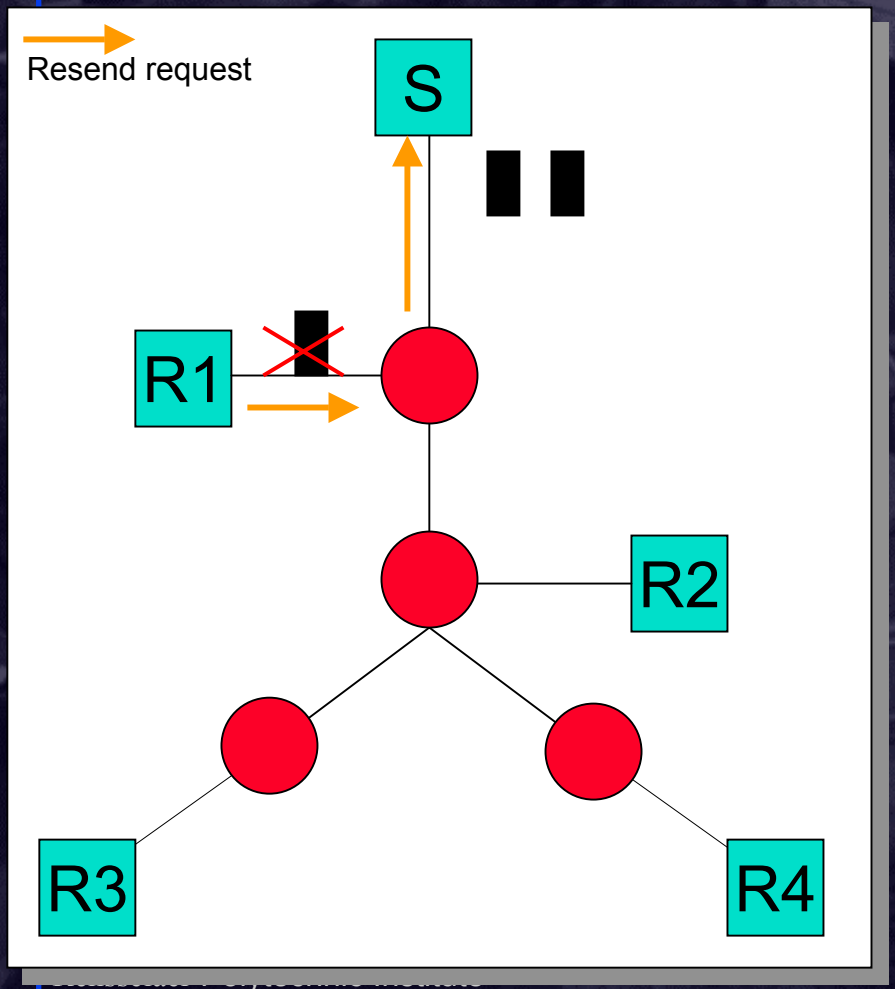


Retransmission

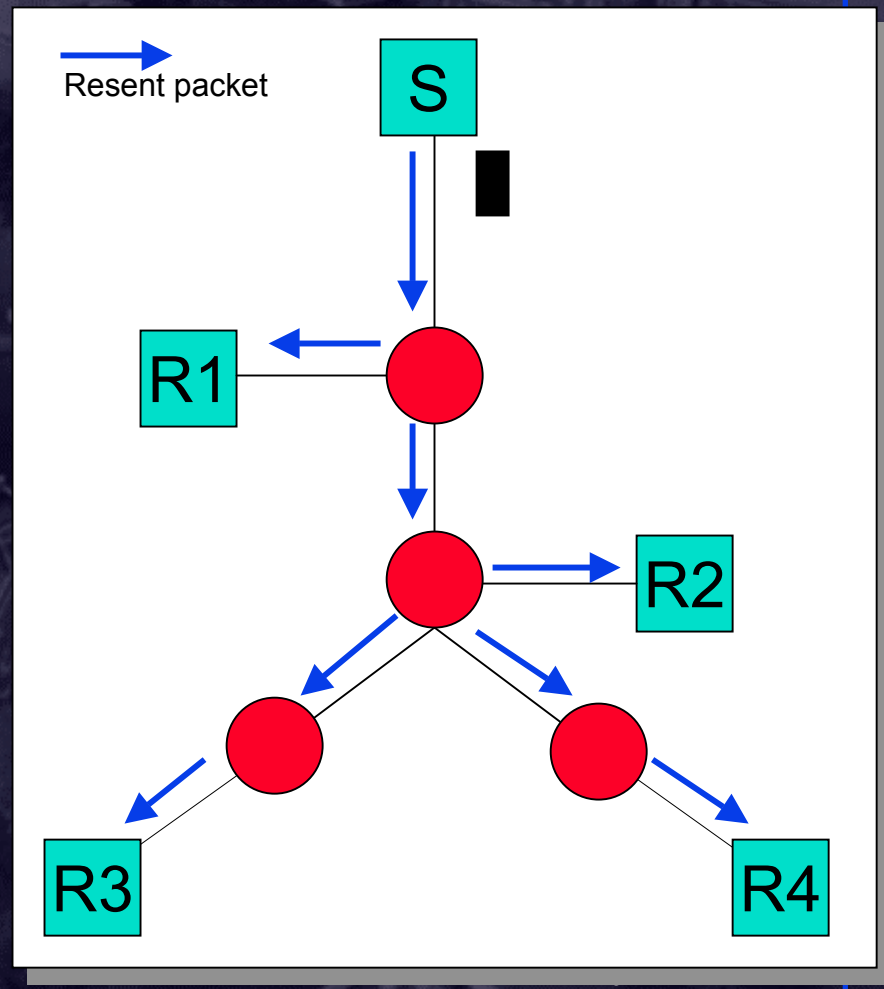
- Re-transmitter
  - Options: sender, other receivers
- How to retransmit
  - Unicast, multicast, scoped multicast, retransmission group, …
- Problem: retransmissions (aka repairs) may reach destinations that don't require a retransmission
  - A.k.a "exposure" problem
  - Solution: subcast the re-transmission only to receivers that need it.

Why Subcast? Exposure problem…

Packet 1 does not reach R1;
Receiver 1 requests a resend

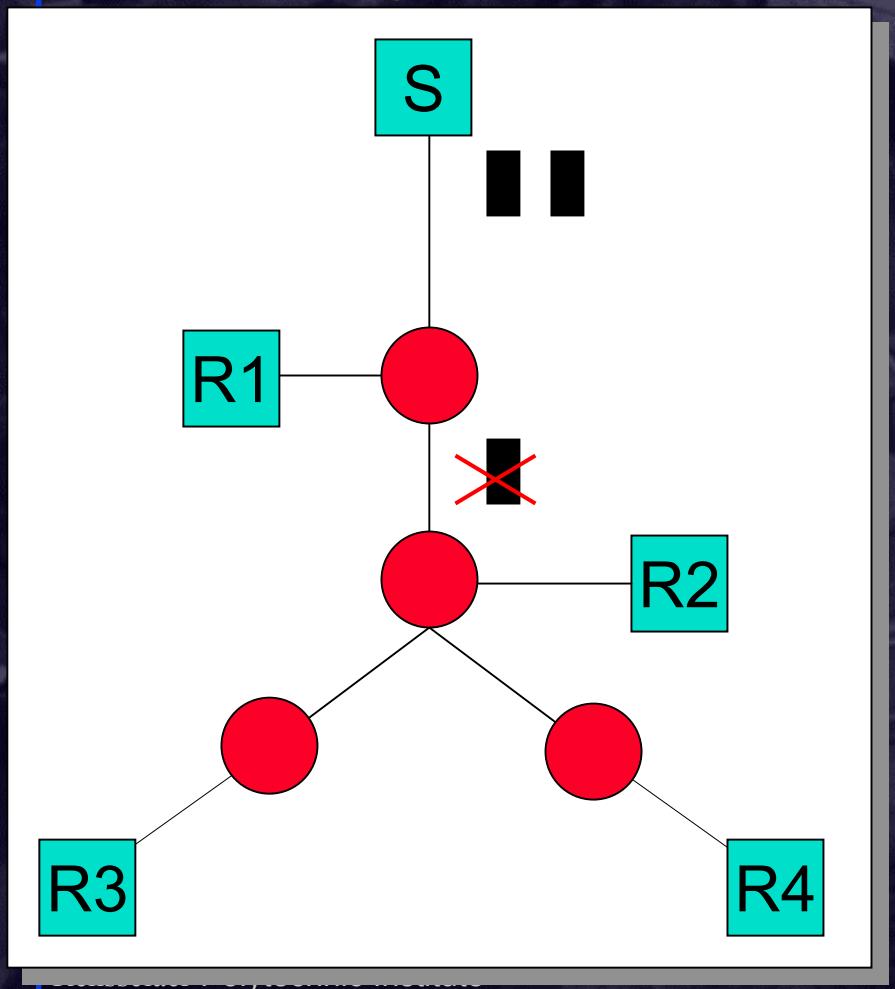


Packet 1 resent to all 4 receivers

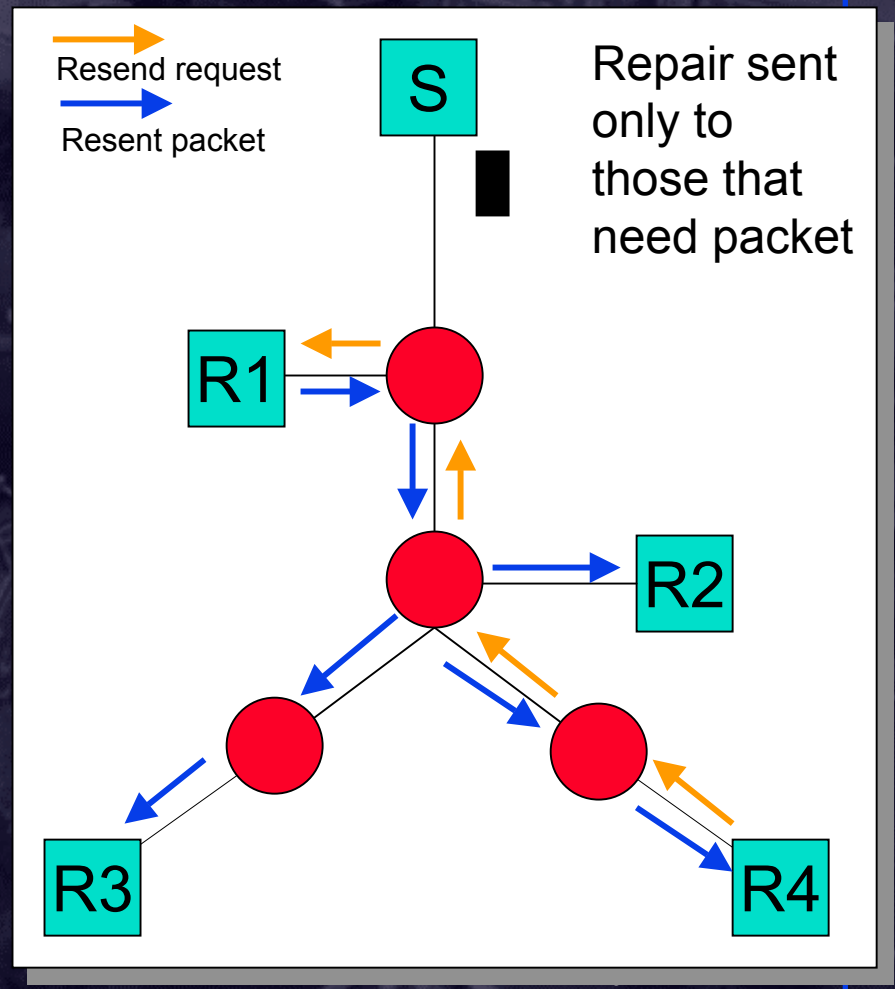


Ideal Recovery Model

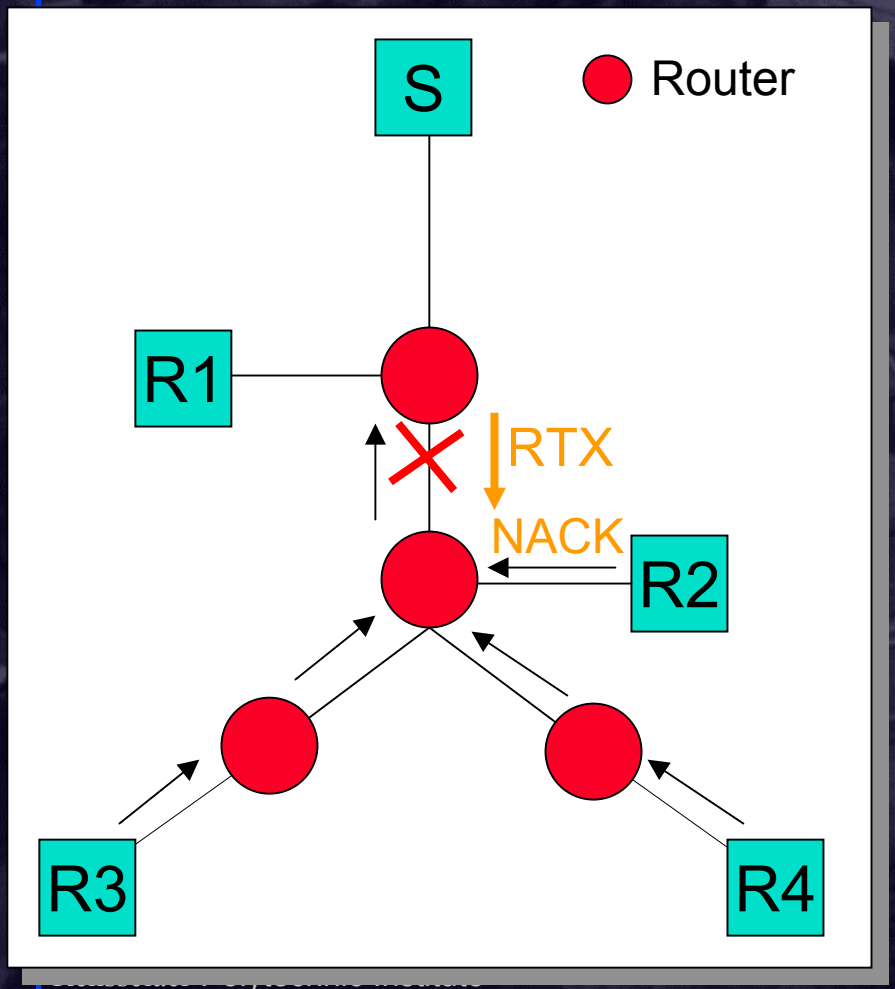
Packet 1 reaches R1 but is lost before reaching other Receivers



Only one receiver sends NACK to the nearest S or R with packet



Using the Routers



- Routers do transport level processing:
  - Buffer packets
  - Combine ACKs
  - Send retransmissions
- Model solves implosion and exposure, but not scalable
- Violates end-to-end argument

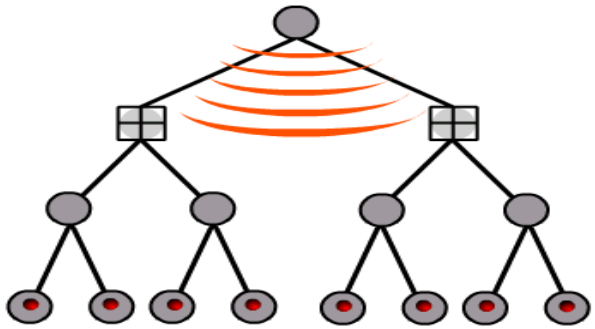
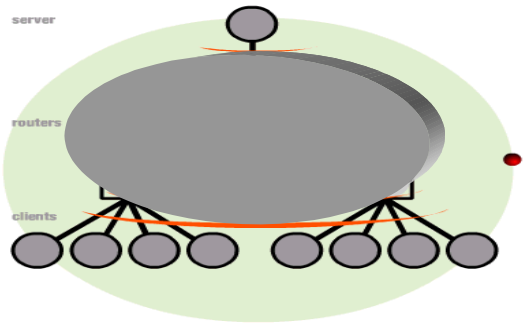
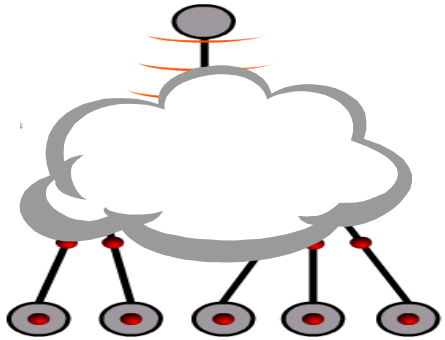
Reliable Multicast Transport: Issues

- Retransmission can make reliable multicast as inefficient as replicated unicast
  - (N)ACK-implosion if all destinations ack at once
  - "Crying baby": a bad link affects entire group
- Heterogeneity: receivers, links, group sizes
- Anonymous/Open/Dynamic Group Model:
  - Source does not know # of destinations, and destinations may vanish
- Multicast applications do not need strong reliability of the type provided by TCP.
  - Can tolerate some reordering, delay, etc

RMT building blocks: RFC 3048

- NACK only: Eg: SRM uses only end-to-end mechanisms.
- Tree-based ACK: aggregators reduce reverse traffic. Eg: RMTP-II
- Asynchronous Layered Coding (ALC): use of forward-error correction (FEC), and no feedback, aka "proactive" FEC
- Router assist: use of NAKs but router support for aggregation. Eg: PGM
  - FEC retransmissions (aka reactive FEC) instead of data retransmissions

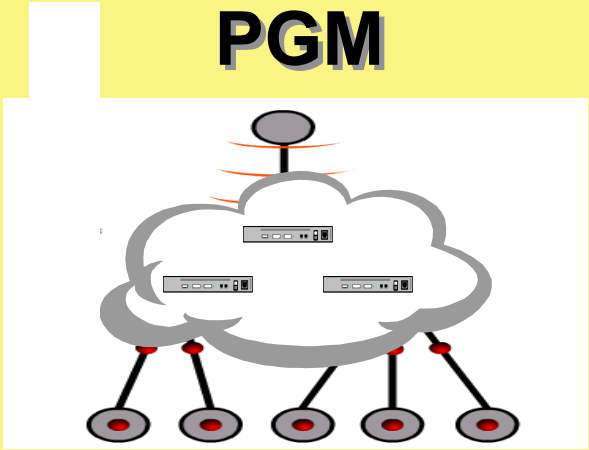
Classes of RMT Protocols

TRACK	Ring	NACK-Only
RMTP-II	RMP	SRM
		
•Tree with Hard State	•Ring Algorithm	•No Hierarchy
•ACK & NACK & FEC	•ACK & NACK	•Multicast NACK & FEC
•Confirmed Delivery	•Many to Many	•Simple to Implement
•Requires Configuration	•Lowest Scalability	•Moderate Scalability

Classes of RMT Protocols

Router-Assist

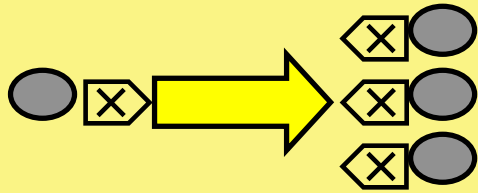
PGM



- Tree with Soft State
- NACK & FEC
- Scaleable & Auto Config
- Req. New Router Code

ALC (FEC)

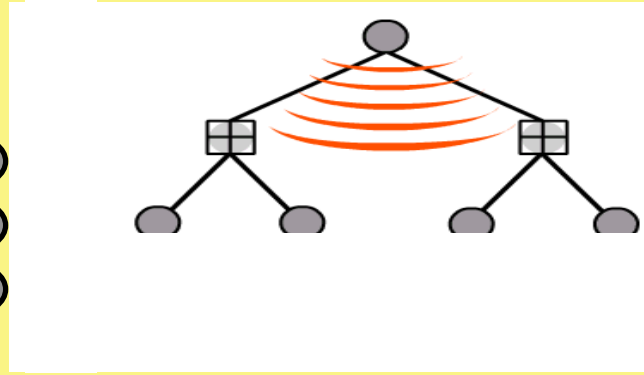
Digital Fountain



- No Return Channel
- FEC Only
- Highest Scalability & Supports Hetergeneity
- Best for non real time or semi-reliable video

File Transfer

MFTP



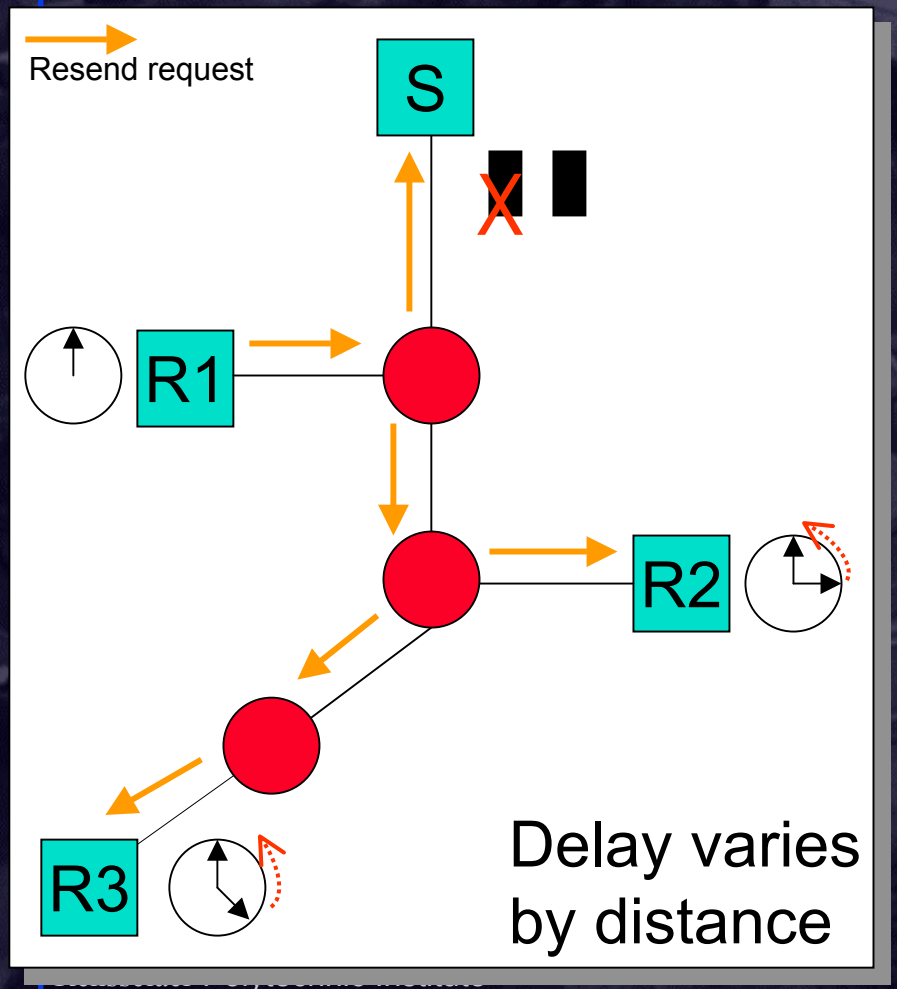
- One-level TRACK
- ACK & NACK
- Simple to Implement
- Non Real Time

SRM

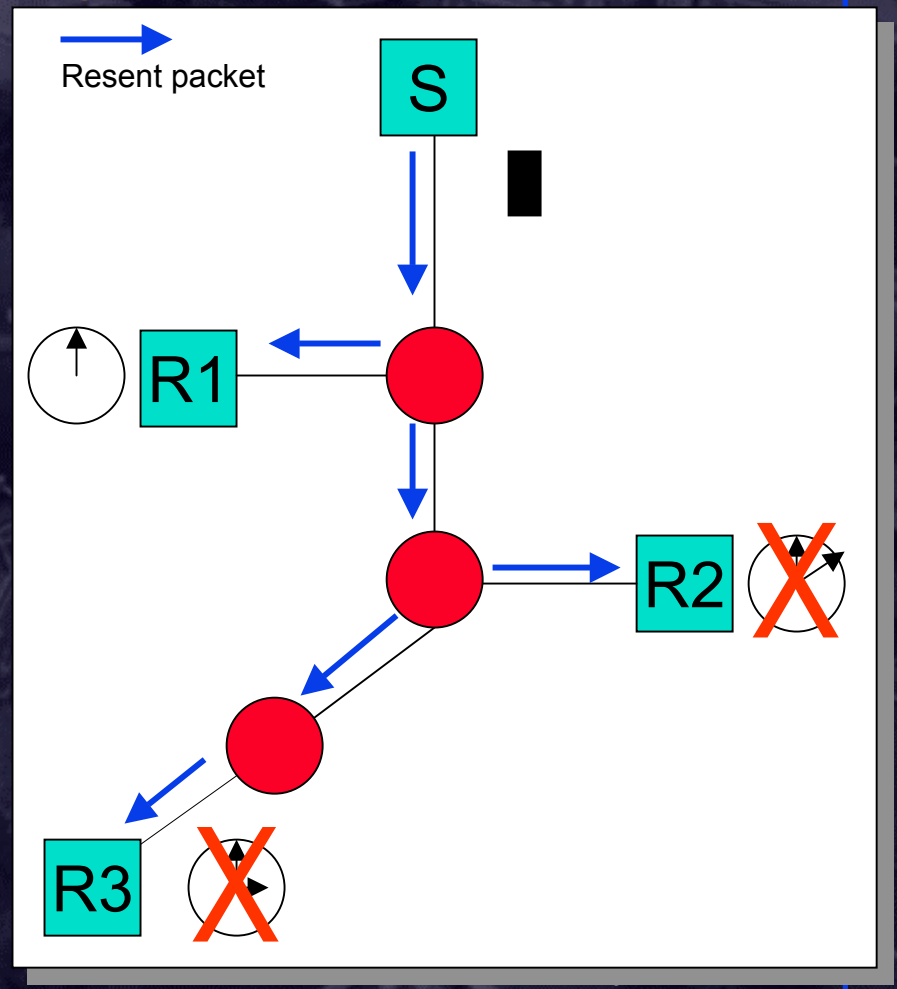
- Originally designed for *wb*
- Receiver-reliable
  - NACK-based
- Every member may multicast NACK or retransmission

SRM Request Suppression

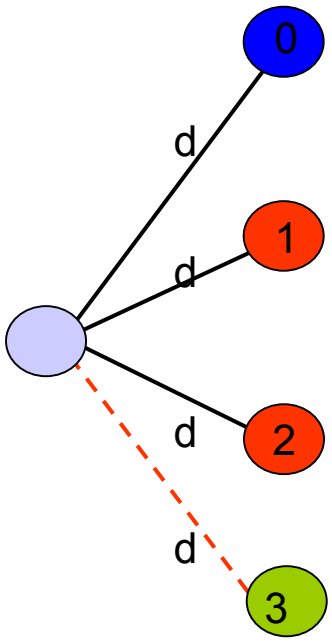
Packet 1 is lost; R1 requests resend to Source and Receivers

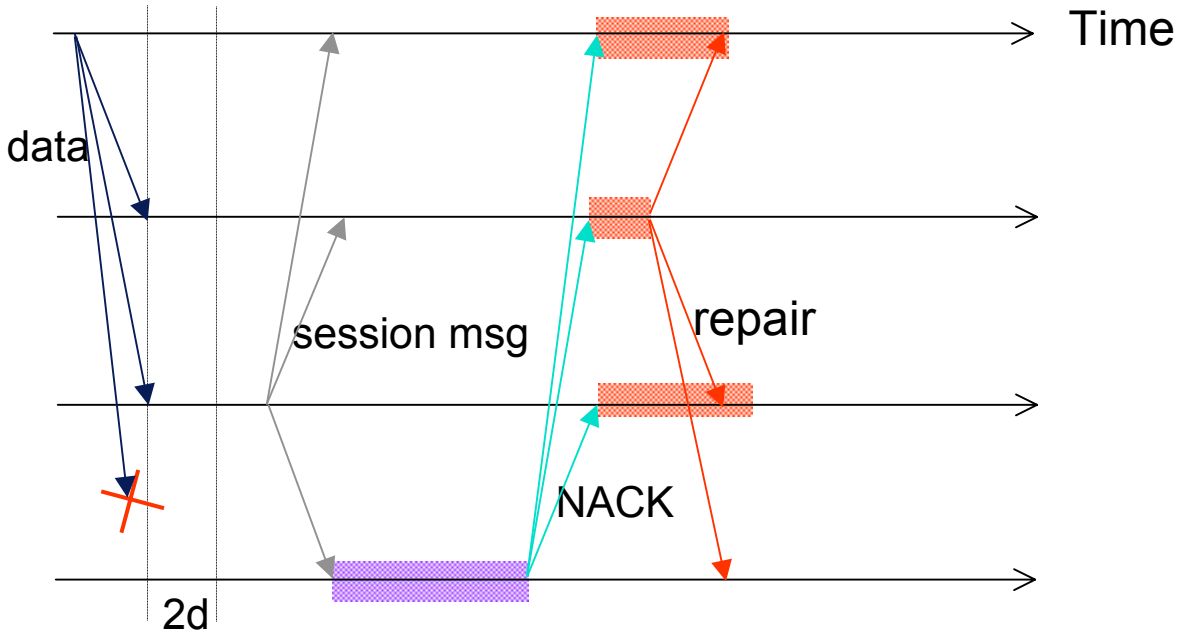


Packet 1 is resent; R2 and R3 no longer have to request a resend



SRM: Stochastic Suppression





$$\text{Delay} = U[0, D_2] \times d_{S,R}$$

= Sender

= Repairer

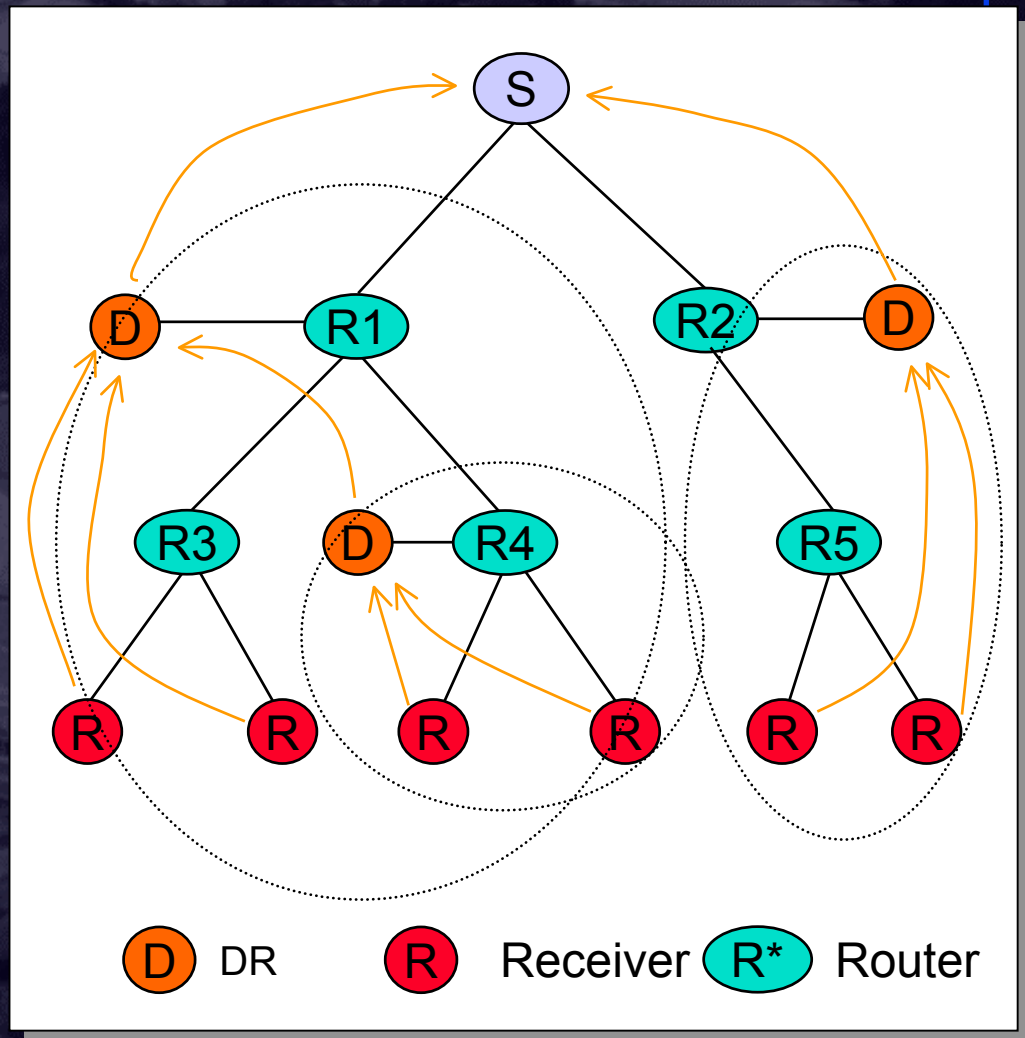
= Requestor

SRM: Summary

- All members get all the data that has been sent to the the multicast group (*packet reliability*)
- Repair requests/responses are multicast.
- Scope of repair requests/responses can be TTL limited or a separate "local recovery group" can be formed
- Techniques to avoid implosion of repair requests, and reduce control traffic: NAK backoff timers
- NACK/Retransmission suppression
  - Delay before sending
  - Delay based on RTT estimation
  - Deterministic + Stochastic components
- Periodic session messages
  - Full reliability
  - Estimation of distance matrix among members

RMTP: Fixed Hierarchy

- Rcvr unicasts periodic ACK to its Designated Receiver (DR)
- DR unicasts its own ACK to its parent
- Rcvr chooses closest statically configured (DR)
- Mcast or unicast retransmission
  - Based on percentage of requests
  - Scoped mcast for local recovery

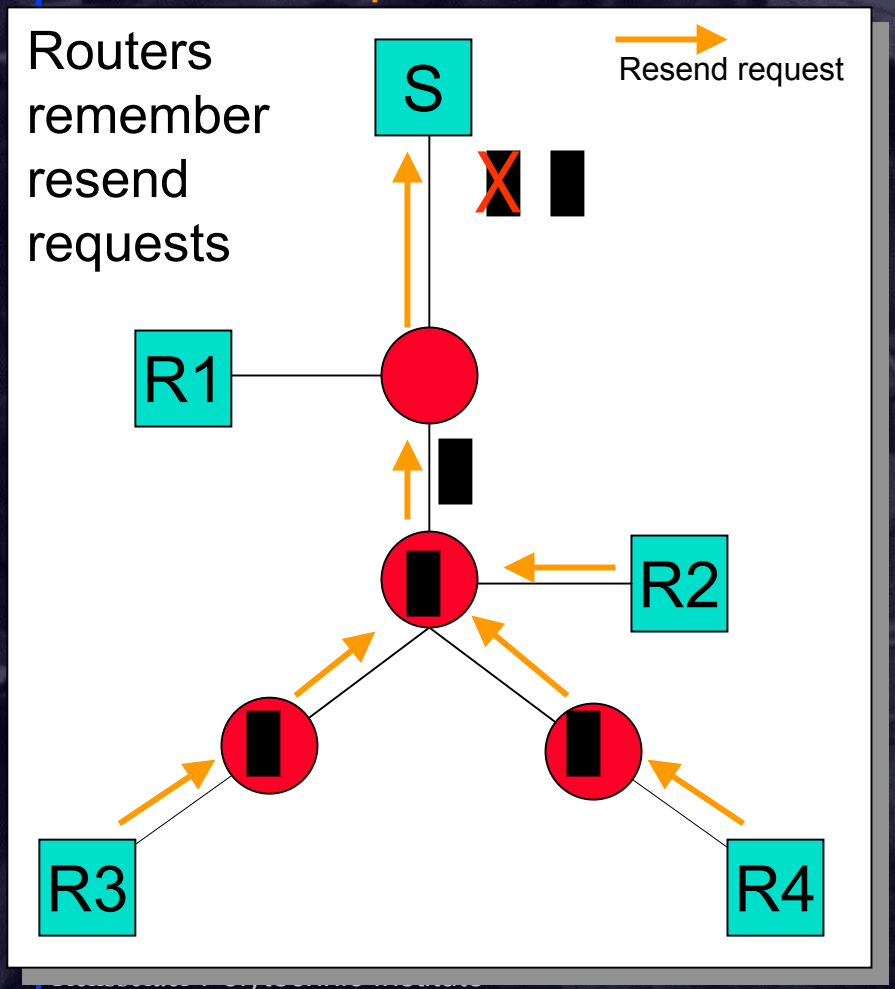


RMTP: Comments

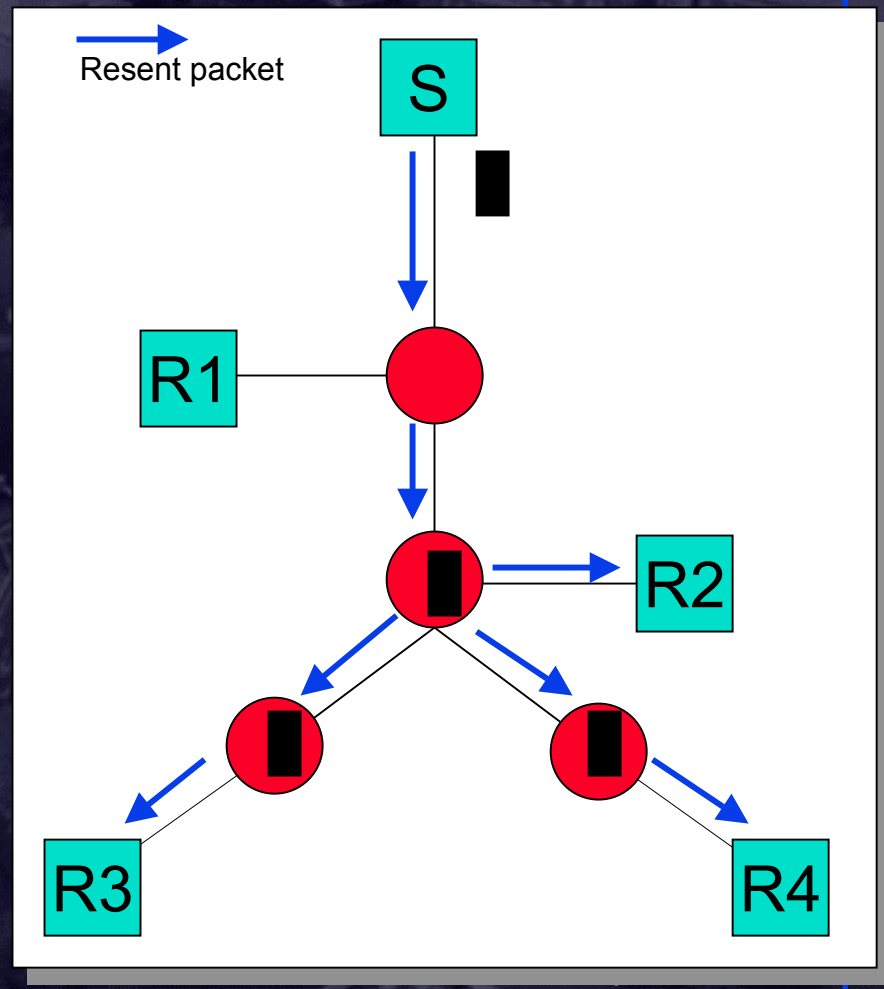
- +: Heterogeneity
  - Lossy link or slow receiver will only affect a local region
- –: Position of DR critical
  - Static hierarchy cannot adapt local recovery zone to loss points

Pragmatic General Multicast

Packet 1 reaches only R1;
R2, R3, R4 request resends



Packet 1 resent to R2, R3, R4;
Not resent to R1

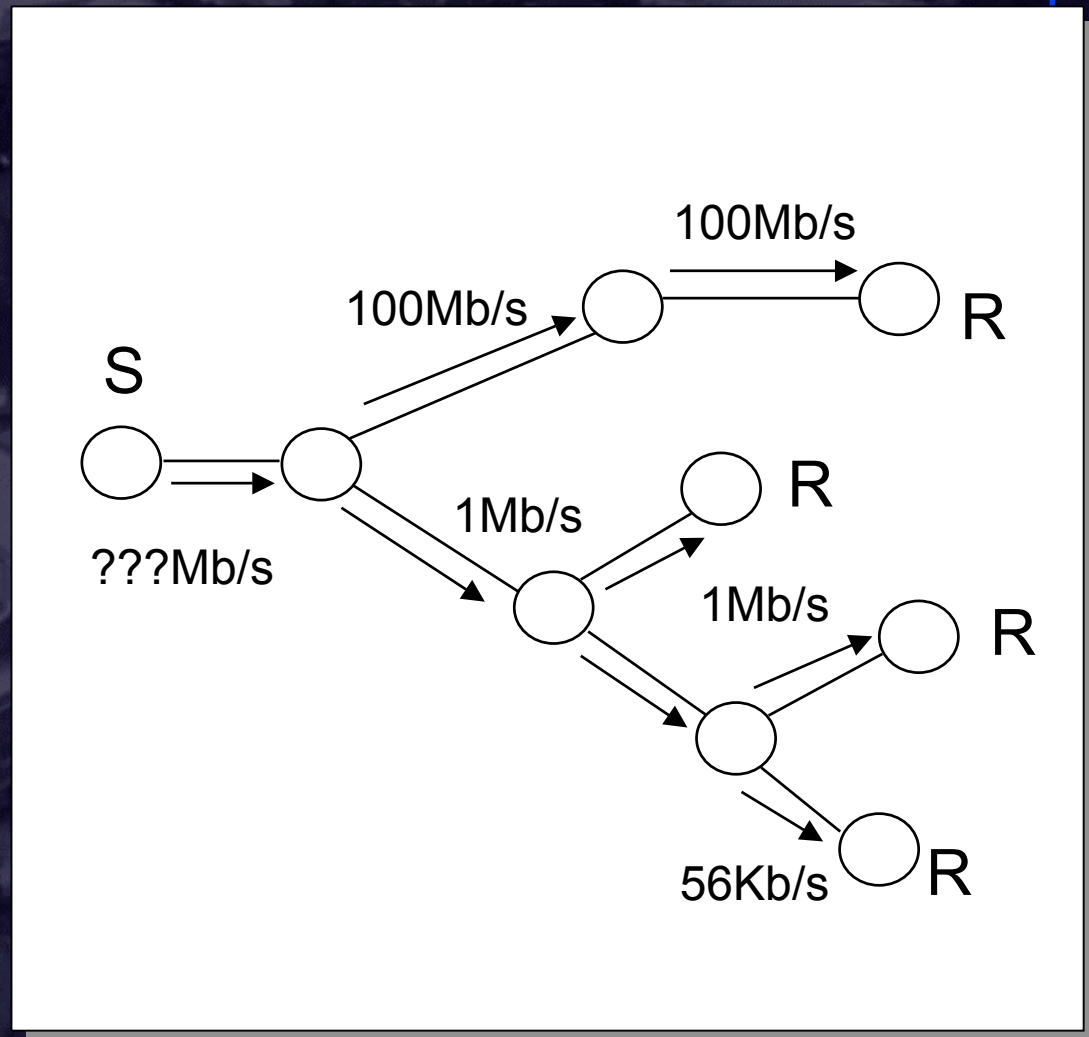


ALC Protocol: Digital Fountain

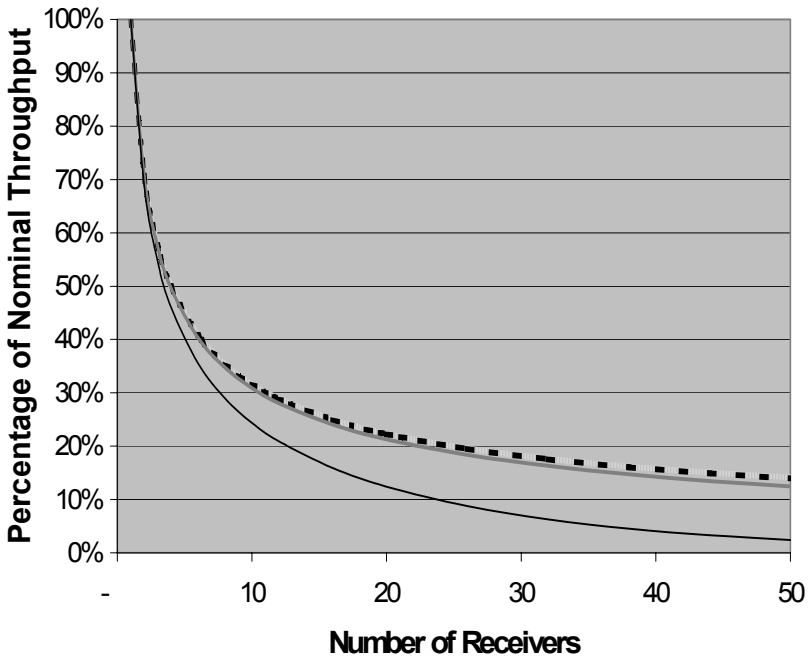
- Instead of using a single multicast group, split data over multiple groups
  - Example: Groups at 8, 16, 32, 64, 128, 256 Kbps
  - Receivers can receive from 8 - 504 Kbps
- Requires a way of splitting data up
  - Hierarchical video codecs
  - File transfer with Tornado FEC codes
- Supports very large and heterogeneous groups, simple to implement
- Requires sparse mode routing protocols

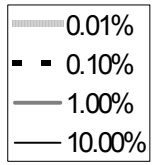
Multicast Congestion Control

- What if receivers have very different bandwidths?
- Send at max?
- Send at min?
- Send at avg?



Single Rate: Drop To Zero Problem



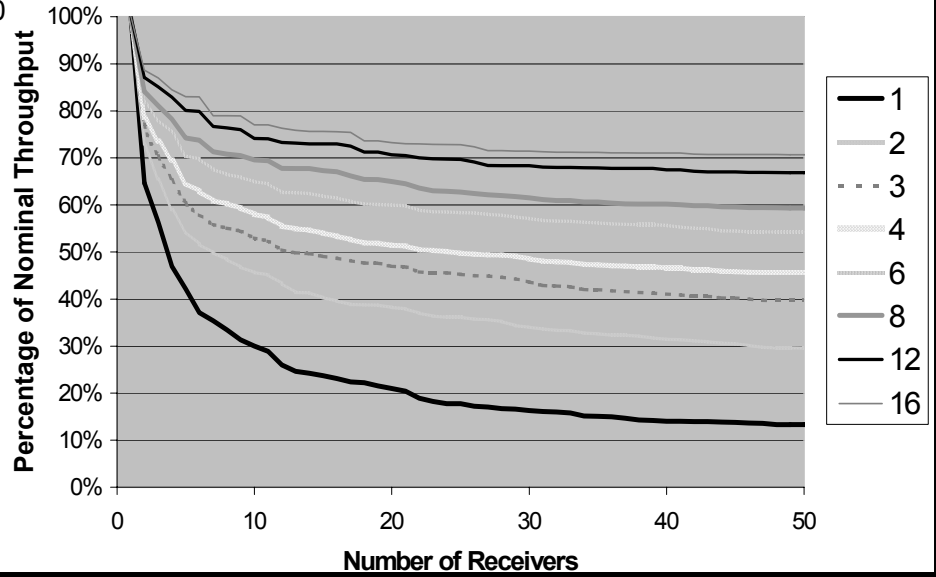


Assumptions

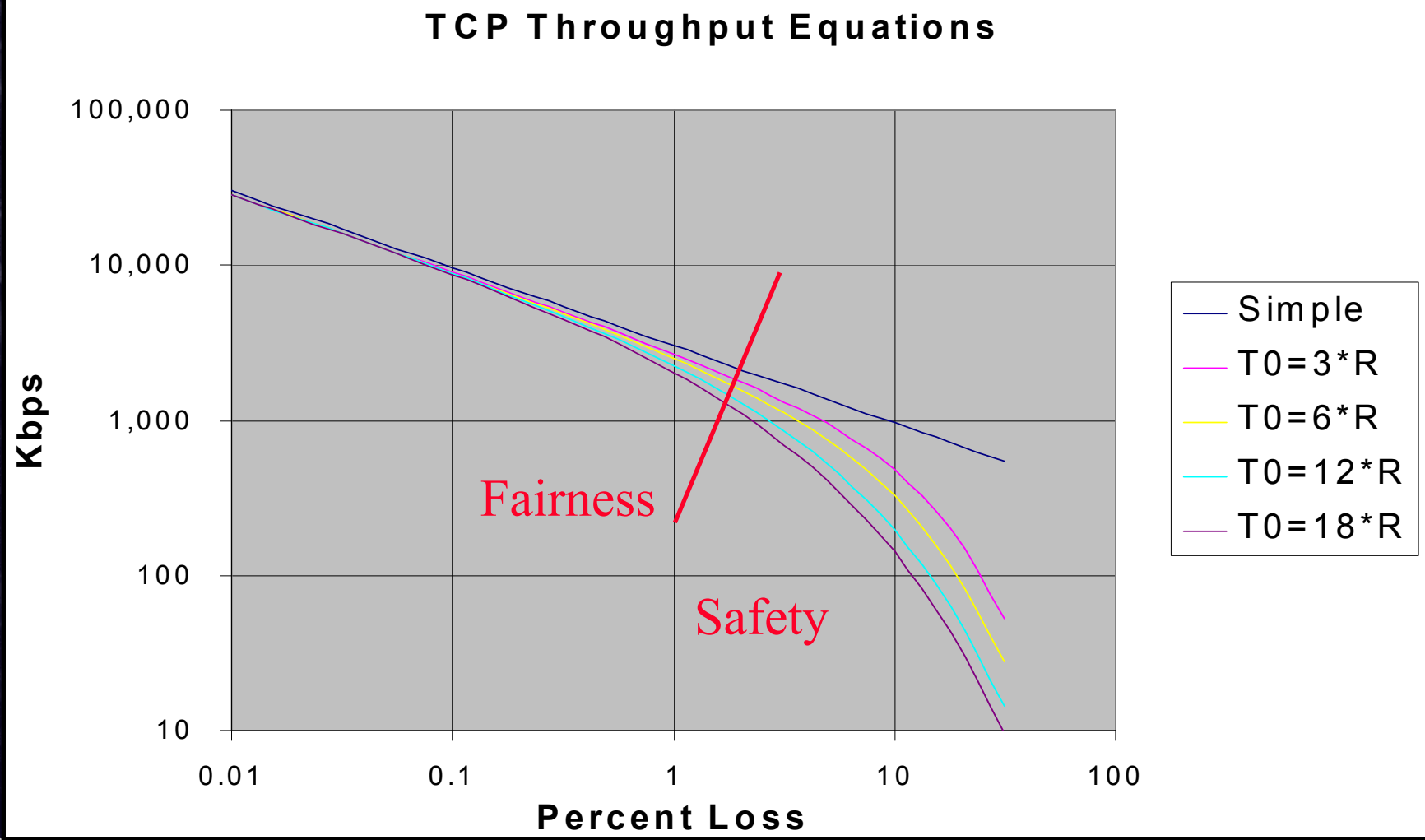
- All receivers have same loss rate (worst case)
- All other assumptions are optimistic, providing an upper bound

Above: Drop-towards-zero on a TCP Time Scale, for different loss rates

Right: Drop-towards-zero For different averaging periods



Single Rate: TCP Friendliness



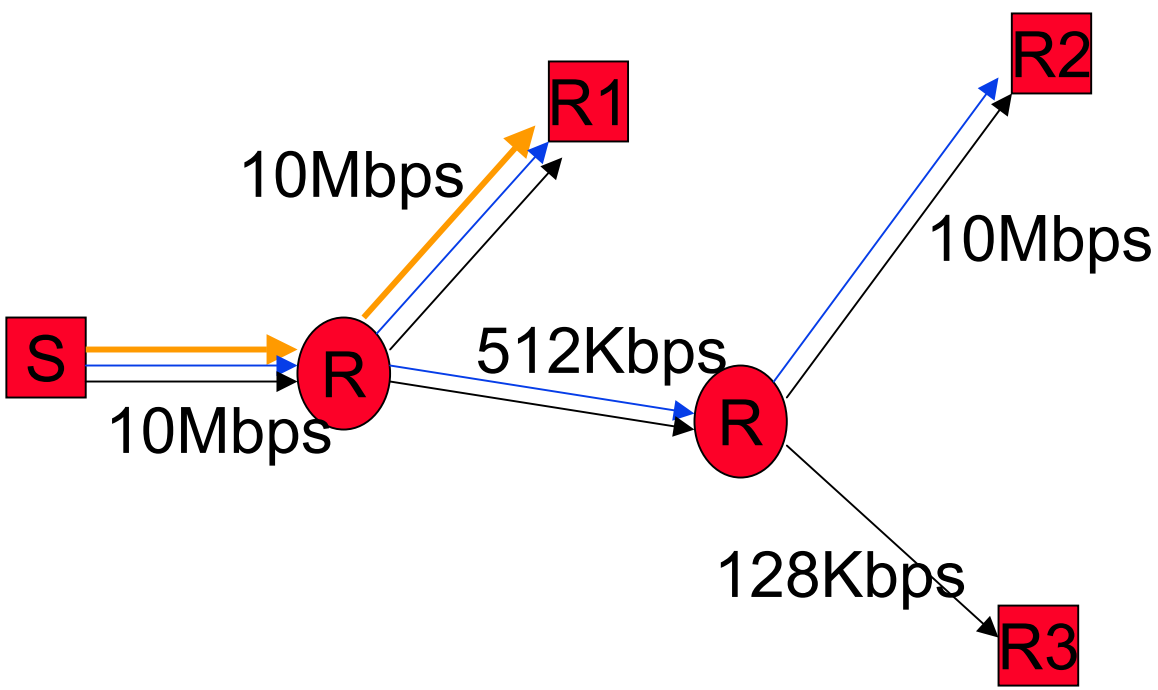
PGM Congestion Control (PGMCC)

- Receivers measure their loss and RTT
- The slowest receiver sends back an ACK on each data packet (I.e. ack clock)
- The sender runs TCP congestion control algorithms using these ACKs
- Results in faster responsiveness but somewhat high variation in send rate
- Works well for small groups, or if only a single receiver at a time is congested

Multi-Rate: Receiver Adaptation

- Receiver-driven Layered Multicast (RLM)
  - Layered video encoding
  - Each layer uses its own mcast group
- Receiver subscribes to max group that will get through with minimal drops
- Dynamically adapt to available capacity
  - Use packet losses as congestion signal
  - On congestion, receivers drop a layer
  - On spare capacity, receivers add a layer
  - Join experiments used for shared learning
- Assume no special router support
  - Packets dropped independently of layer

Layered Media Streams



R1 joins layer 1,
joins layer 2
joins layer 3

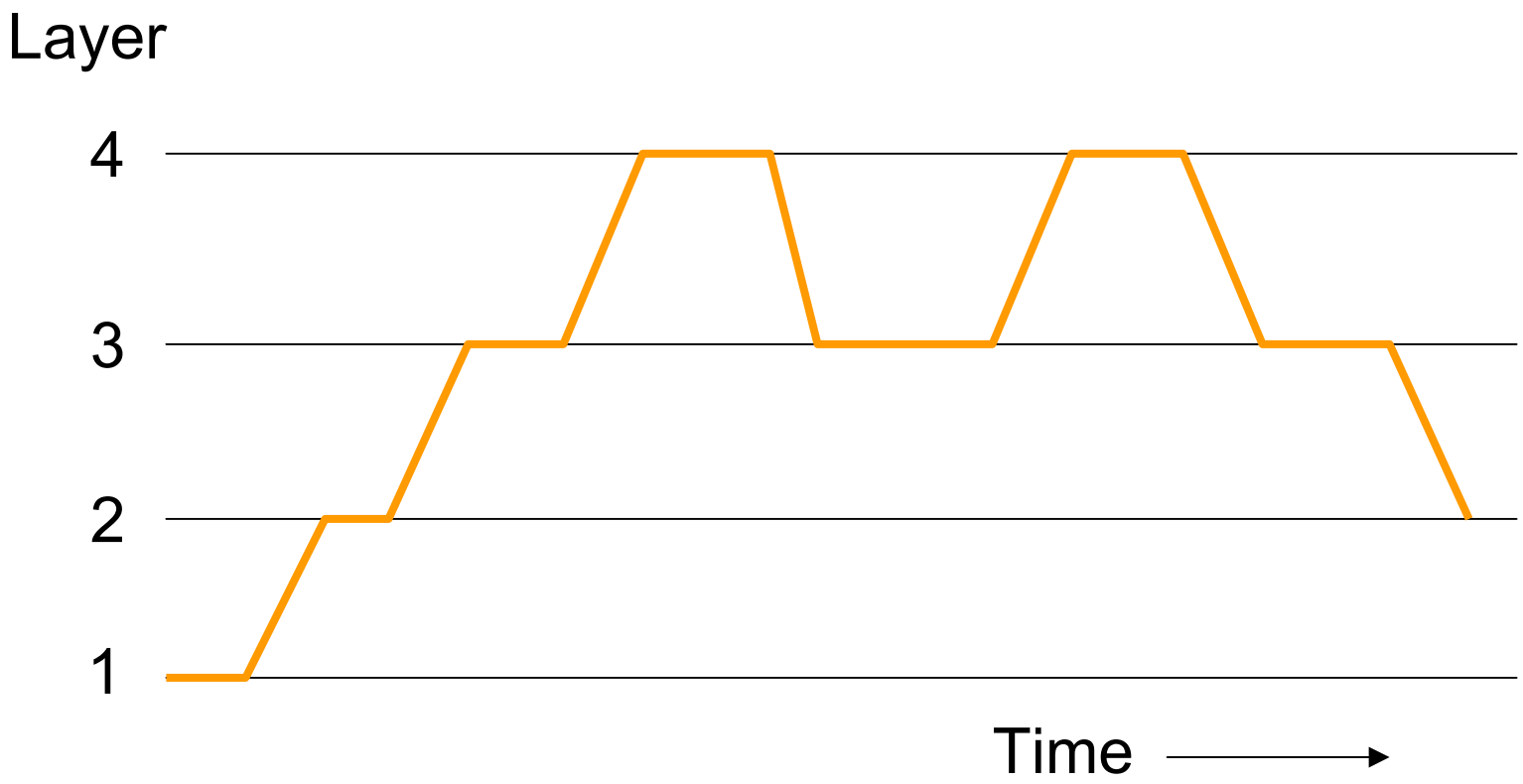
R2 join layer 1,
join layer 2
fails at layer 3

R3 joins layer 1,
fails at layer 2

RLM Join Experiment

- Receivers periodically try subscribing to higher layer
- If enough capacity, no congestion, no drops → Keep layer (& try next layer)
- If not enough capacity, congestion, drops → Drop layer (& increase time to next retry)
- What about impact on other receivers?

Join Experiments



RLM Receiver Coordination

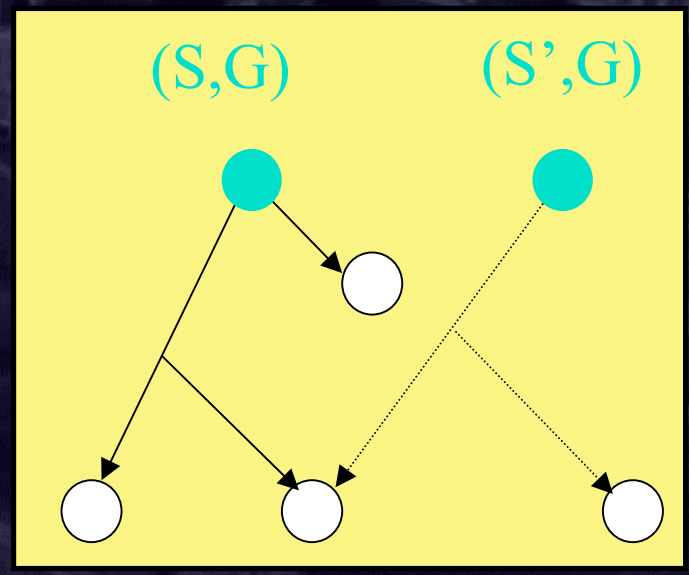
- Receiver advertises intent to add layer
- Other receivers
  - Avoid conflicting experiments
  - If experiment fails, will see increased drops => don't try adding layer! (shared learning)
  - OK to try adding lower layer during higher layer experiment
    - Won't cause drops at higher layer!

IP Multicast: Concerns

- Deployment is difficult and slow
  - ISP's reluctant to turn on IP Multicast
- Open Group Model: Anyone can join a Group
- Inter-domain routing: MSDP doesn't scale
- Address allocation is also complex
- PIM-SM requires a Rendezvous Point (RP): subject to attack
- Multicast tried to solve too many problems…

Single-Source Multicast (SSM)

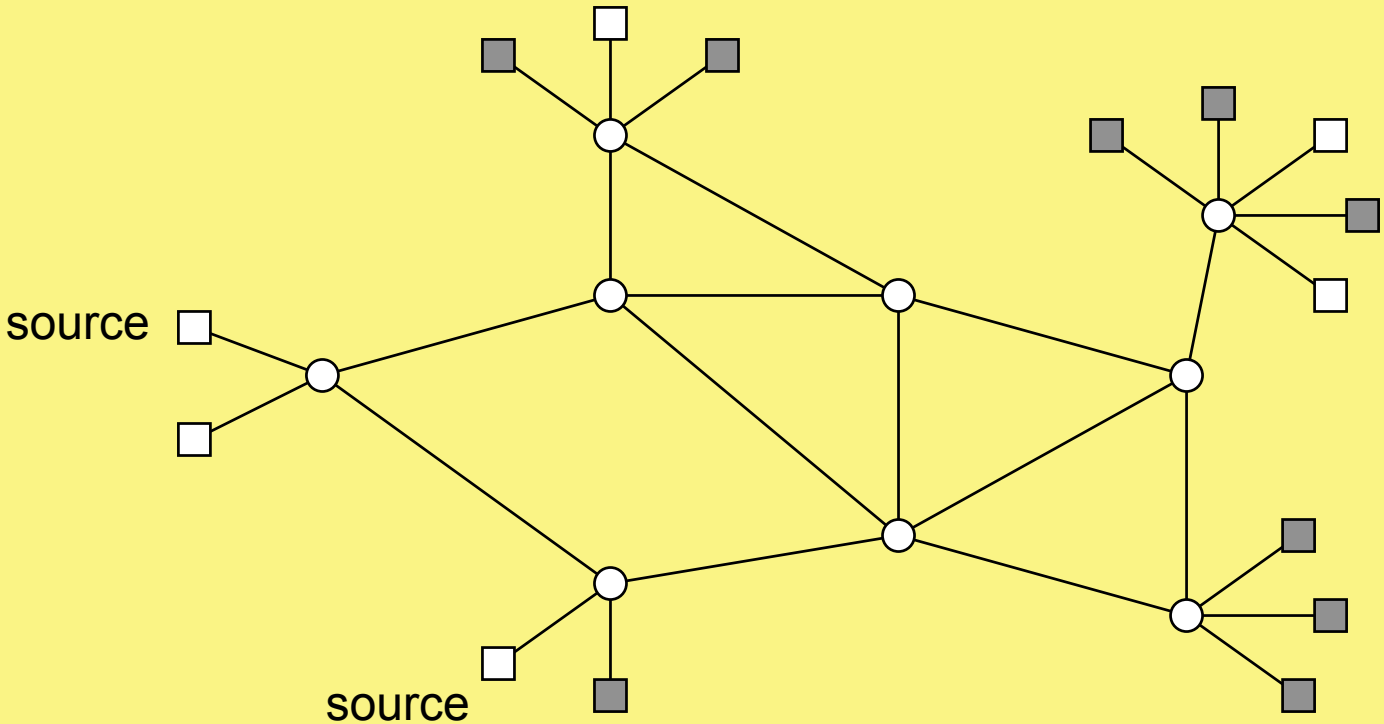
- Source-specific channel (S,G)
  - only S can send to G
  - another source S' must use a separate channel (S',G)
  - hosts join channels, so a member joining only (S,G) will NOT receive traffic from S'
- Current infrastructure uses Any-Source Multicast (ASM)
  - any source can send to any group at any time



Why SSM?

- Network Operator
  - trivial address allocation (16 million addresses per host)
  - no network-layer source discovery (PIM RP and/or MSDP moved to the application layer)
  - ✓ overcomes two significant obstacles to deployment
- Content Provider
  - exclusive access to multicast groups (no interruptions)
  - permanent multicast groups (easy to advertise)
  - ✓ provides better service

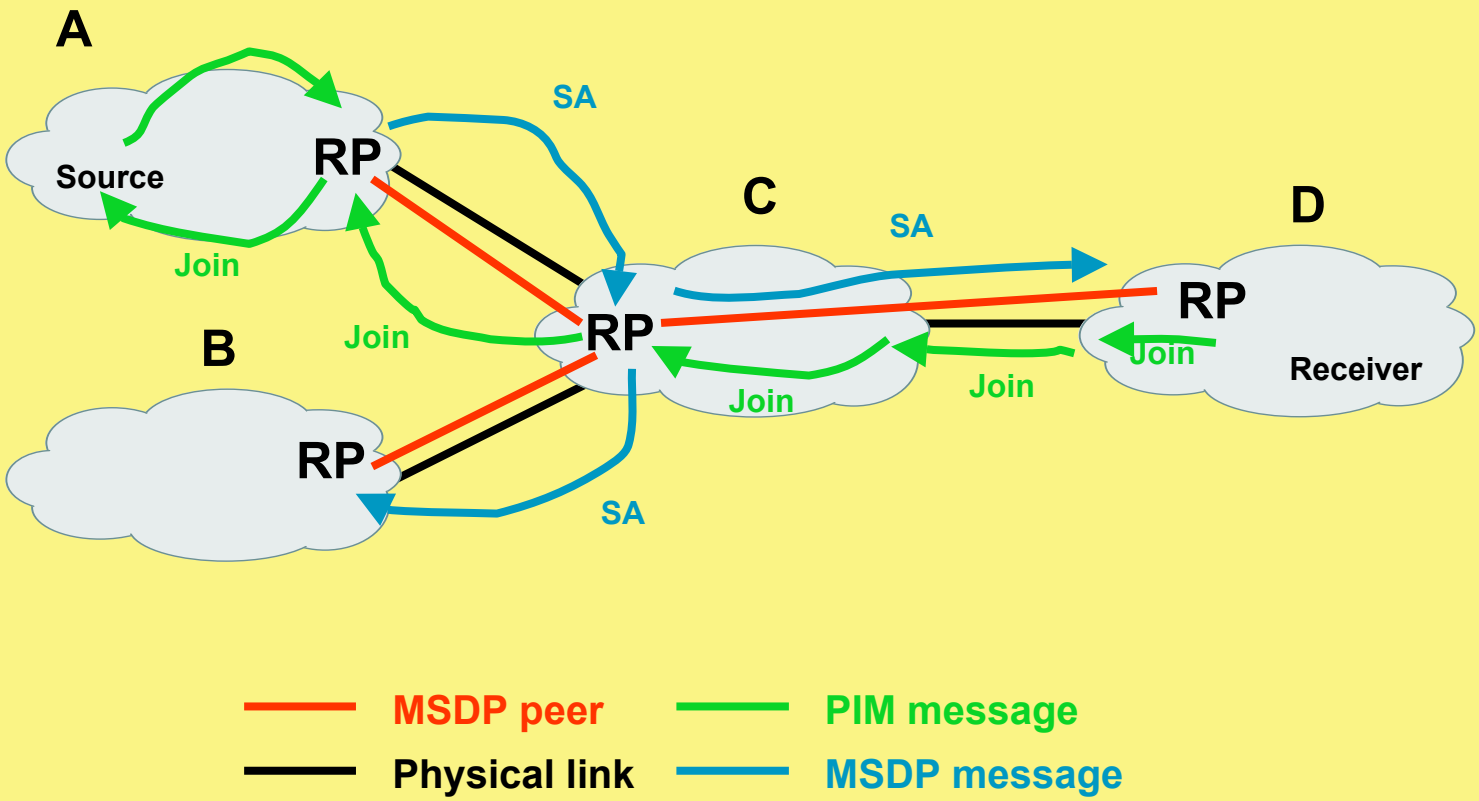
Problem: How to find the Source?



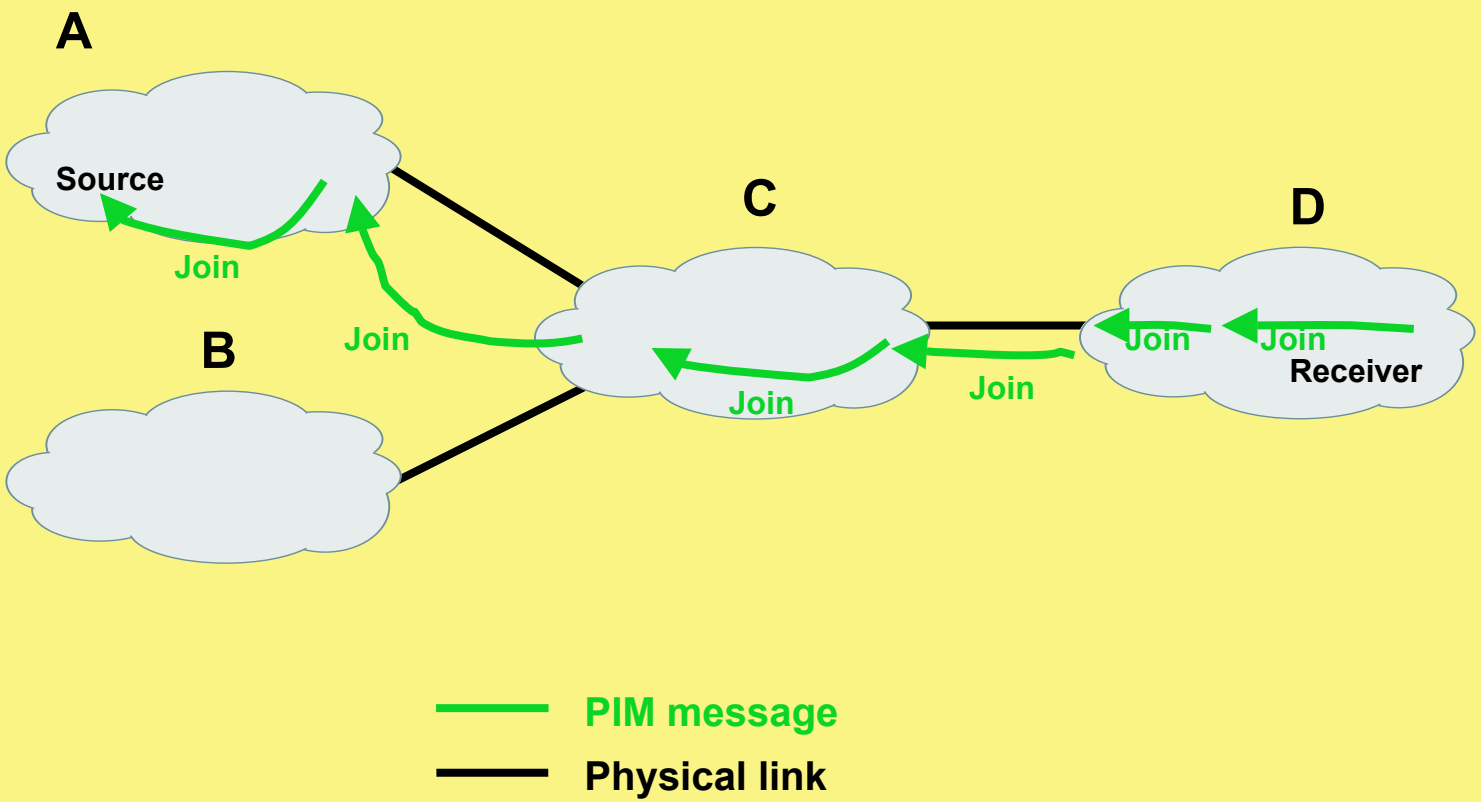
How to Find the Sources?

- broadcast everywhere
  - receivers decide when they do **not** want the traffic
- any source multicast (ASM) [PIM-SM/MBGP/MSDP/IGMPv2]
  - use a rendezvous point (RP)
  - receivers send joins along reverse path to RP
  - sources send traffic to RP
- source specific multicast (SSM) [PIM/MBGP/IGMPv3]
  - require receivers to already know source(s)
  - use some out-of-band mechanism

How MSDP works with PIM-SM



How SSM Works



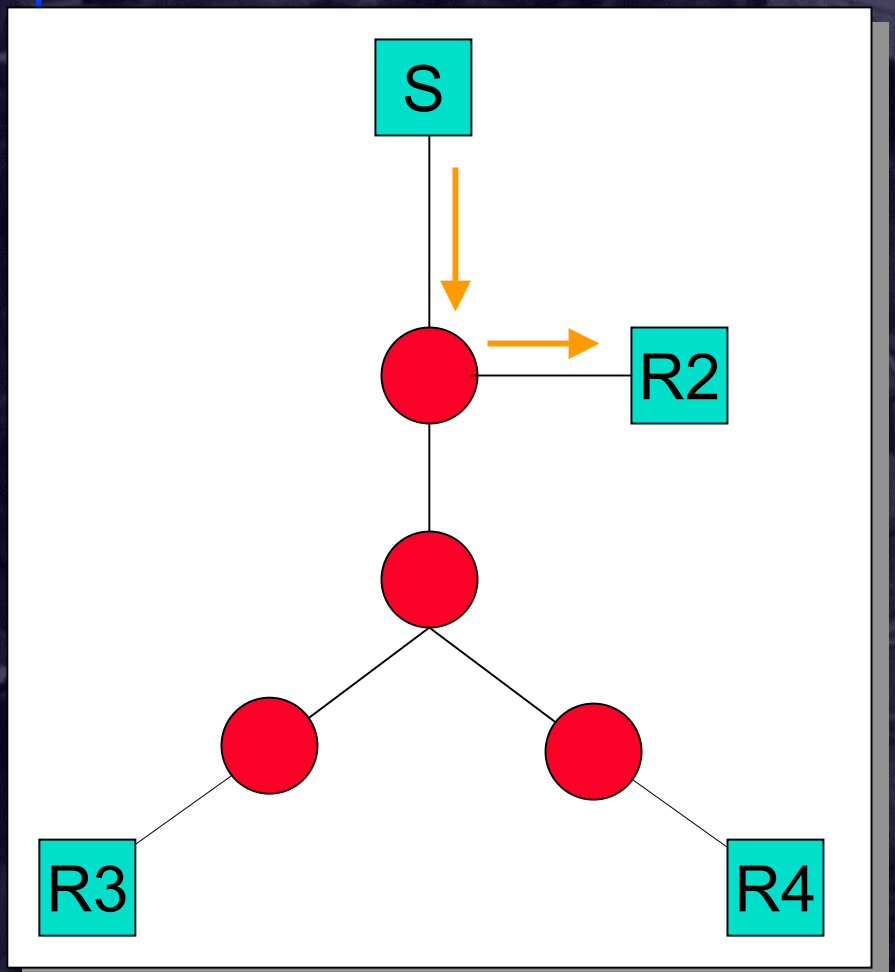
SSM Advantages (cont'd)

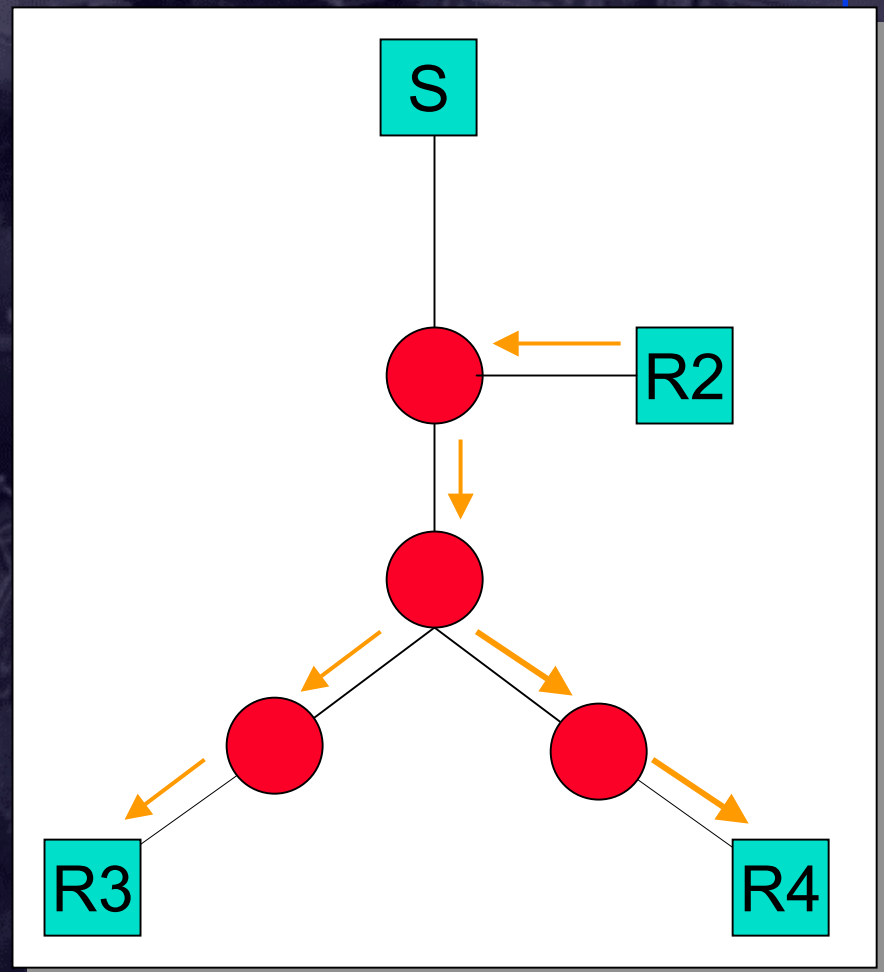
- No RP, No need for MSDP
- All joins are (S,G), so no need for Class D address allocation
- Receivers find out about sources through out-of-band means (such as a web site)
- SSM-only implementations are much simpler than the full PIM-SM
  - No RP, No Bootstrap RP Election
  - No Register state machine
  - No need to keep (*,G), (S,G,rpt) and (*,*,RP) state
  - No (*,G) Assert State

Application-level Multicast

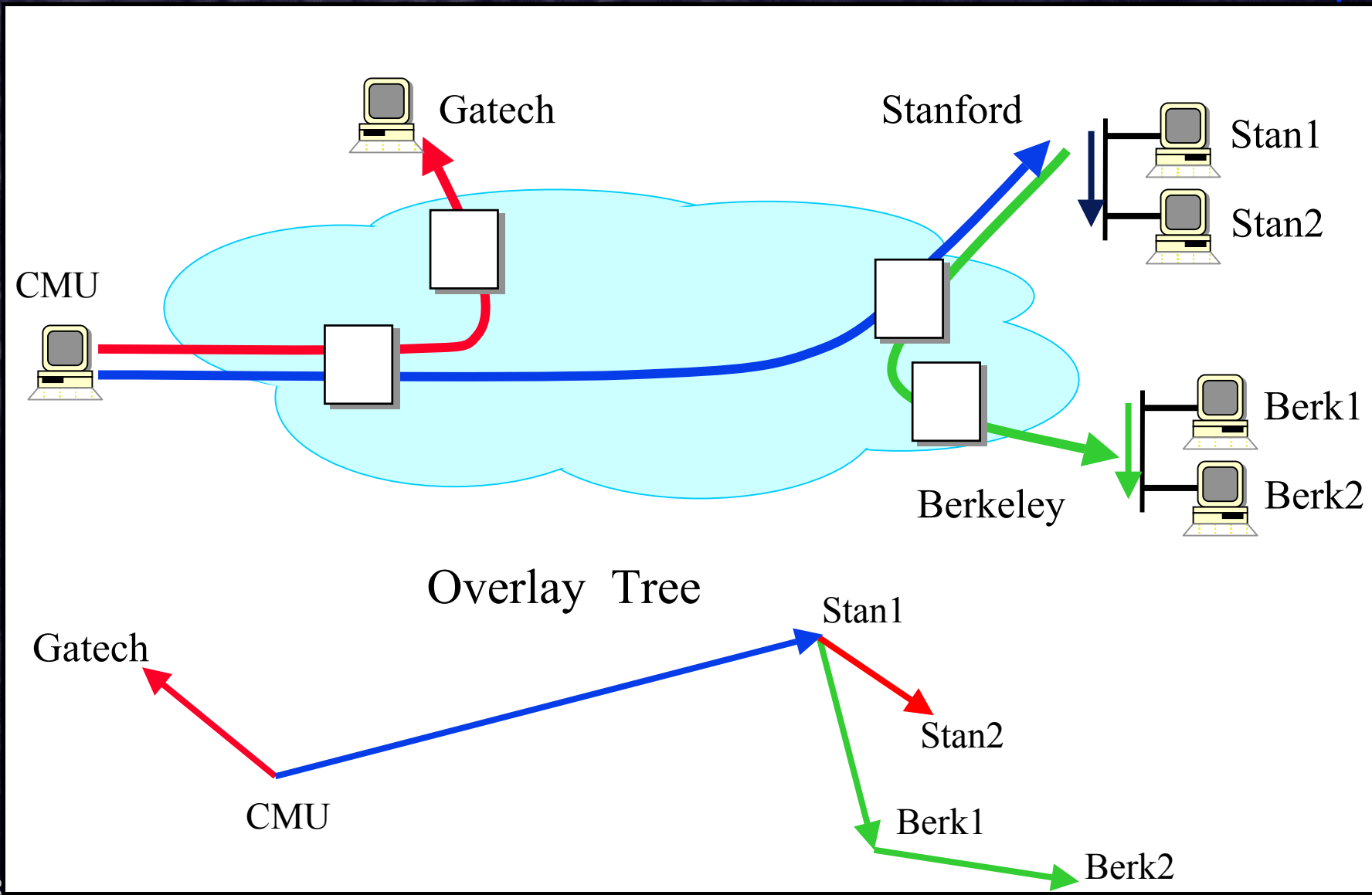
- *Do we really need network level multicast?*
  - Efficiency
  - Logical rendezvous
- Can we get efficiency by creatively using the actual members or a few infrastructure nodes?

Application-level Multicast

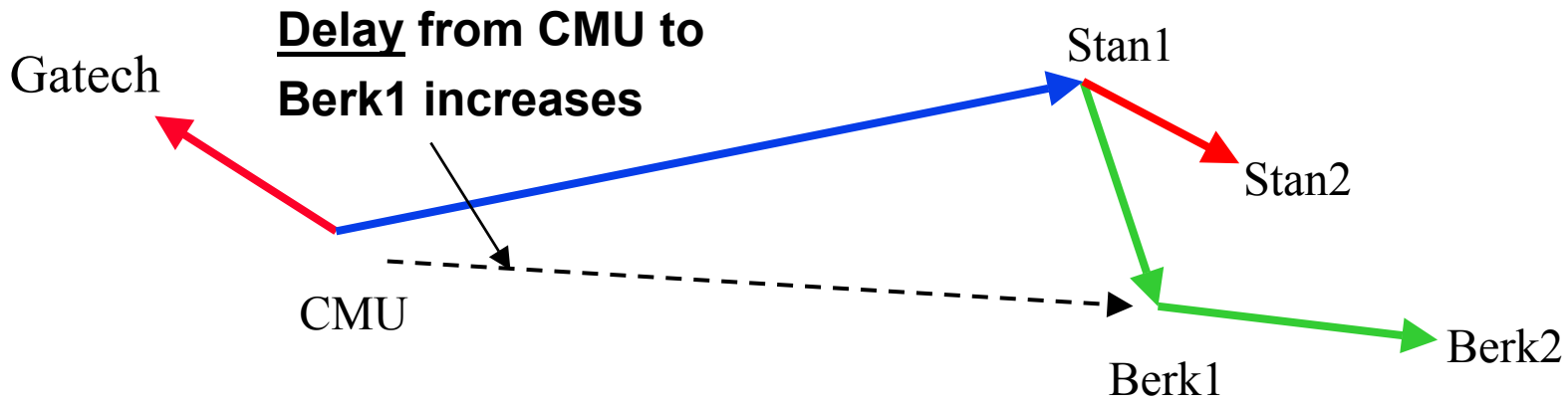


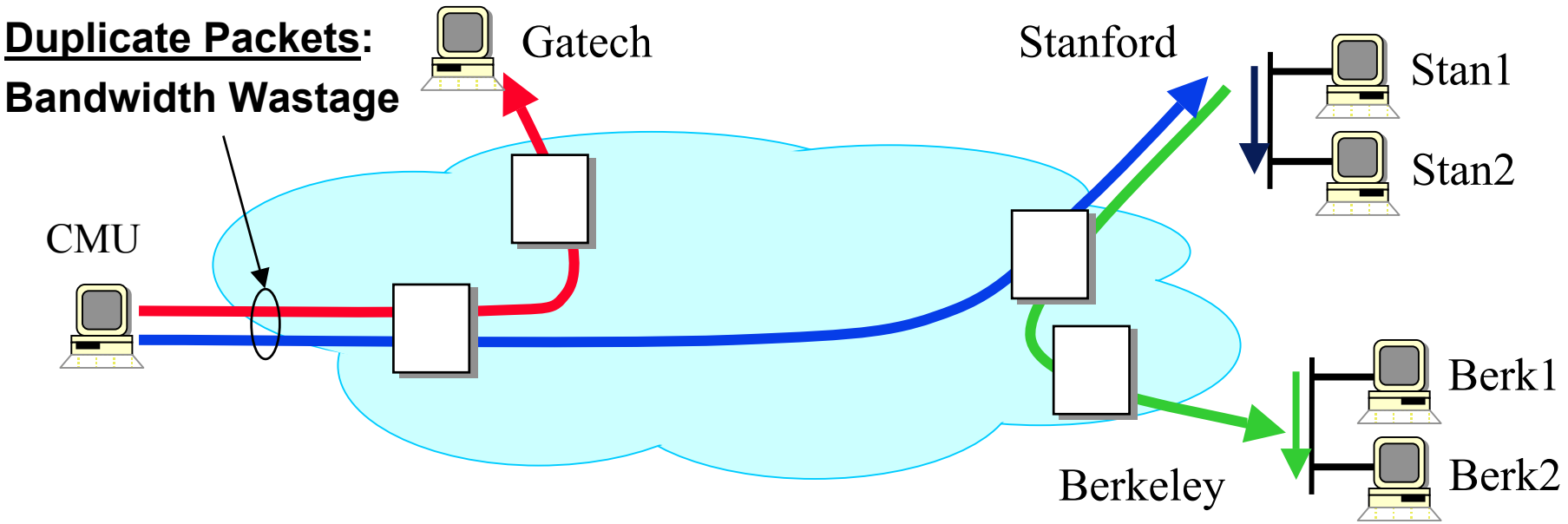


Narada: End System Multicast



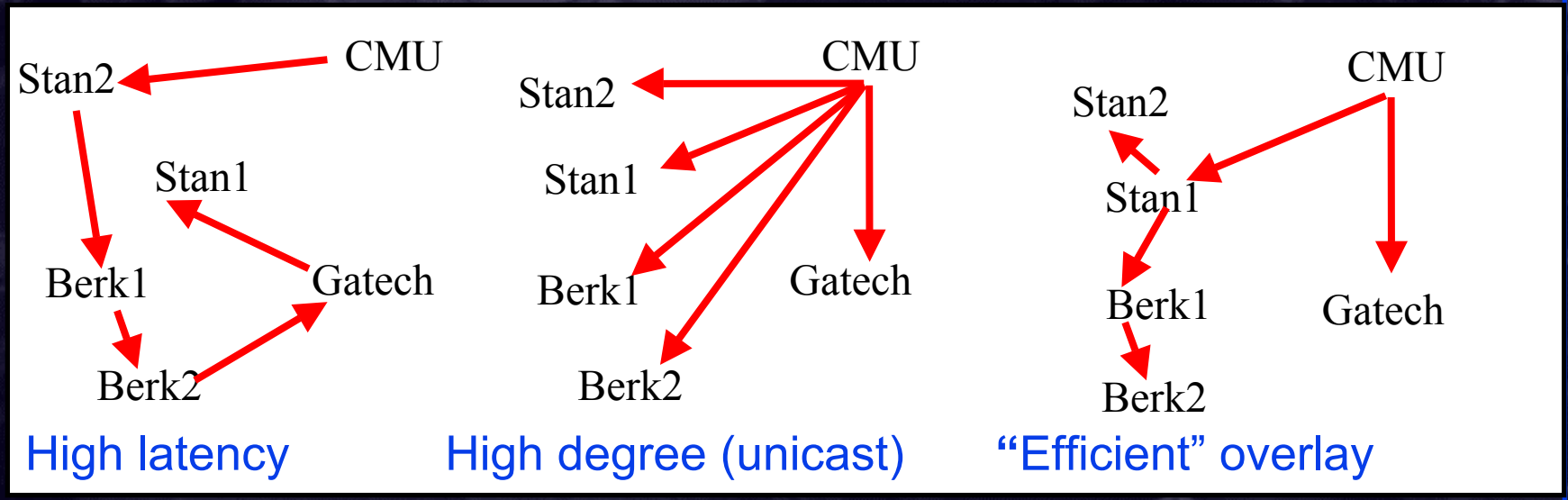
Performance Concerns





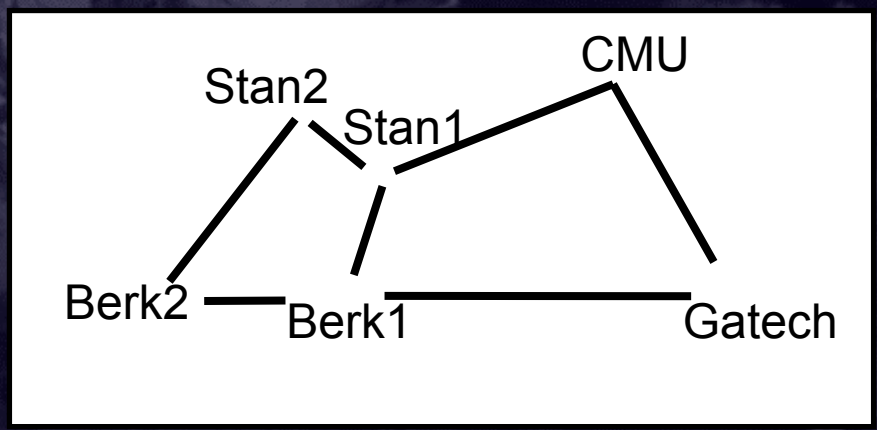
Overlay Tree

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



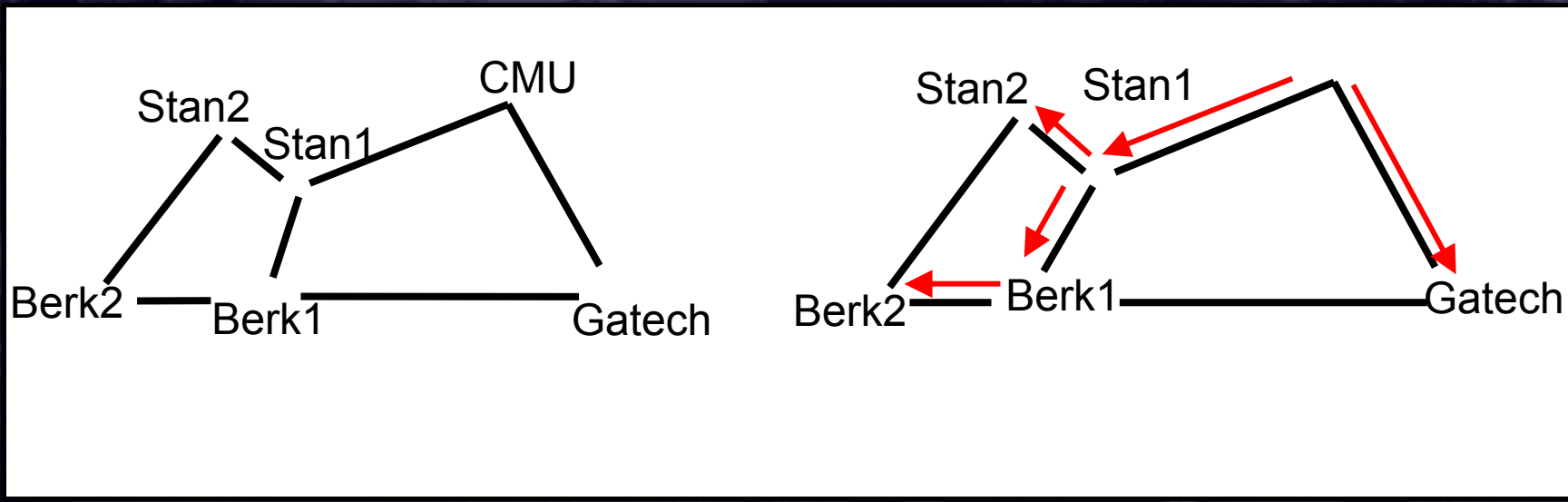
Mesh

- Advantages:
  - Offers a richer topology → robustness; don't need to worry to much about failures
  - Don't need to worry about cycles
- Desired properties
  - Members have low degrees
  - Shortest path delay between any pair of members along mesh is small



Overlay Trees

- Source routed minimum spanning tree on mesh
- Desired properties
  - Members have low degree
  - Small delays from source to receivers



Summary

- IP multicast issues and applications
- Multicast over LANs and scoping
- IGMP
- Multicast Routing and MBONE
- Reliable multicast transports
- Multicast Congestion Control
- SSM, Overlay Multicast