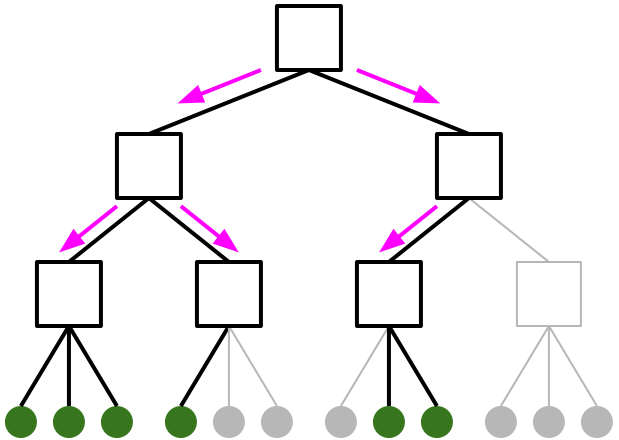


Lecture 23 (Beyond Client-Server 1)

Multicast

CS 168, Fall 2026 @ UC Berkeley

Slides credit: Sylvia Ratnasamy, Rob Shakir, Peyrin Kao

Introducing Multicast

Lecture 22, CS 168, Spring 2026

Introducing Multicast
IP Multicast
- IP Multicast Service Model
- IGMP: Directly-Connected Hosts

DVMRP
- Naive Approach: Flooding
- RPB (Broadcasting): Trees
- RPM (Multicasting): Pruning
- Pros and Cons

Core-Based Trees
- Protocol
- Pros and Cons

IP Multicast Challenges
Overlay-Based Multicast
- Implementation

Motivation: Multicast

Our model so far: The Internet transfers data between hosts.

- Packets are unicast: One sender, one recipient.
- Many applications (e.g. HTTP, DNS, TCP) use the client-server model: One client and one server exchange unicast packets.

Some applications are instead about **groups of hosts** communicating:

- Multi-player games.
- Live content delivery (e.g. Zoom meeting).
- Collaborative documents (e.g. Google Docs).
- AI training (more on this next time).

The unicast, client-server model is an awkward fit for these apps.

- Who is the client? Who is the server?

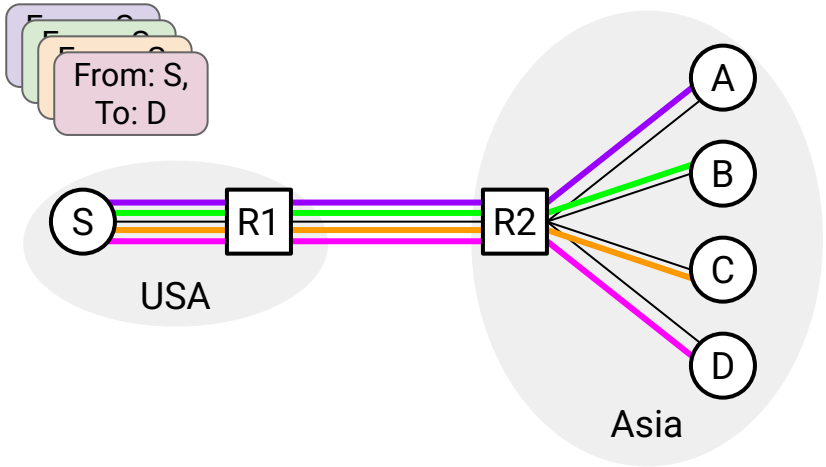
Today's topic: What network protocols exist to support group communication?

Network Support for Multicast

Group communication can be done with unicast only, but it wastes bandwidth.

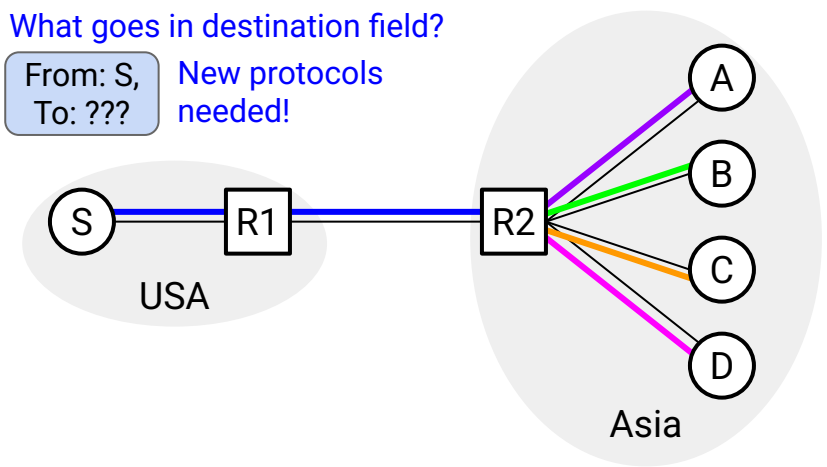
A better approach would avoid sending duplicate copies of a packet along a link.

This multicast model requires extra support from the network.



Unicast-Only Model

Many duplicate copies of the packet sent along the R1-R2 link.



Multicast Model

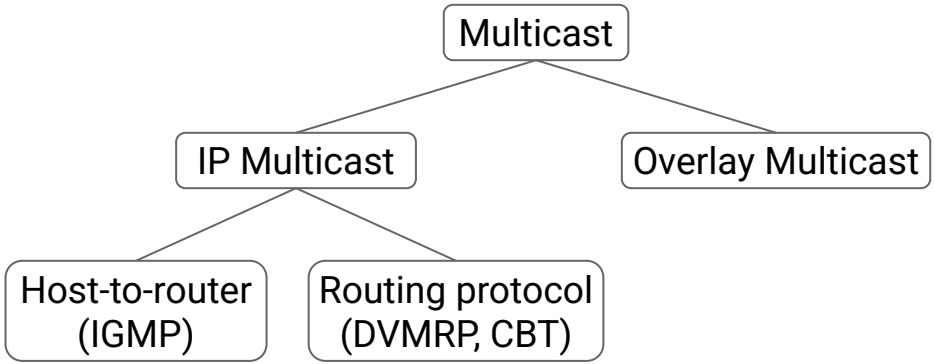
Send a single copy along the R1-R2 link. Then, R2 sends copies to individual hosts.

Implementing Multicast

Where are multicast protocols implemented? There are two options:

- **IP Multicast**: Implemented in routers, at Layer 3.
- **Overlay Multicast**: Implemented in hosts, at Layer 7.

We'll study both options and their trade-offs.



IP Multicast Service Model

Lecture 22, CS 168, Spring 2026

Introducing Multicast

IP Multicast

- **IP Multicast Service Model**
- IGMP: Directly-Connected Hosts

DVMRP

- Naive Approach: Flooding
- RPB (Broadcasting): Trees
- RPM (Multicasting): Pruning
- Pros and Cons

Core-Based Trees

- Protocol
- Pros and Cons

IP Multicast Challenges

Overlay-Based Multicast

- Implementation

Brief History of IP Multicast

IP multicast was the focus of much work in the 1990s and early 2000s.

- Anticipated killer app: Live-streamed TV/radio.

Mixed success in terms of adoption:

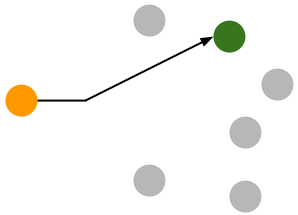
- Modern routers support IP multicast, but operators sometimes disable multicast.
- Used inside a single network (e.g. a datacenter).
- Not used across multiple domains.
- This means you can't send multicast packets on the Internet and expect them to "automatically" work.

Techniques developed can be applied to solve different networking problems.

- Next time: We'll see these techniques applied to AI training.

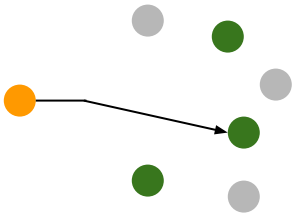
Packet Delivery Models

We've seen four models for delivering a packet:



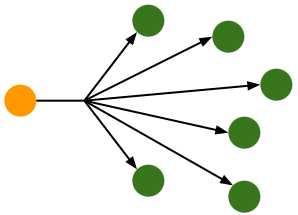
Unicast

Send the packet to exactly one destination.



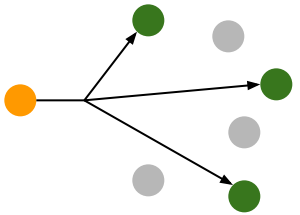
Anycast

Send the packet to one of a set of possible destinations.



Broadcast

Send the packet to all destinations.



Multicast

Send the packet to all members of a group.

Some notes on multicast:

- Hosts can choose to join/leave groups at any time.
- You can send a packet to a group, even if you are not a member of that group.

IP Multicast Model

Each multicast group has an IP address.

- Addresses in the range 224.0.0.0 to 239.255.255.255 are reserved for multicast groups.

Hosts can send 3 different types of packets:

- **Join**: Send a packet to announce that you are joining a group.
- **Leave**: Send a packet to announce that you are leaving a group.
- **Data**: Send a data packet to a multicast group.

Routers must know how to process Join, Leave, and Data packets from hosts.

- Routers must store state. (e.g. "where are the members of group 226.1.2.3?")
- Routers must run protocols to learn how to forward packets.

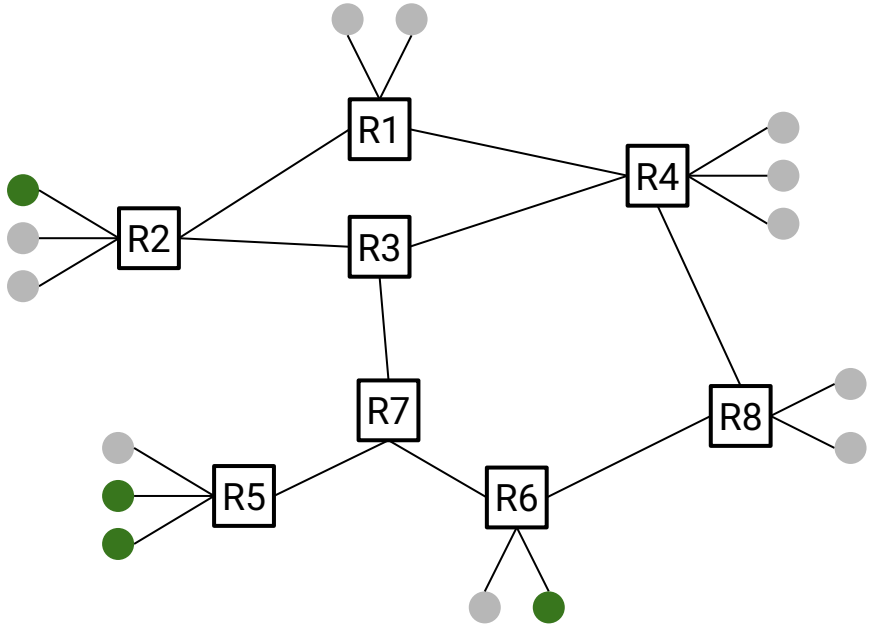
IP Multicast Model: Addressing

Each multicast group has an IP address.

- Addresses in the range 224.0.0.0 to 239.255.255.255 are reserved for multicast groups.

● = Not a member of group 226.1.2.3.

● = Is a member of group 226.1.2.3.



IP Multicast Model: Joining and Leaving

Hosts send a **Join** packet to announce that they are joining a group.

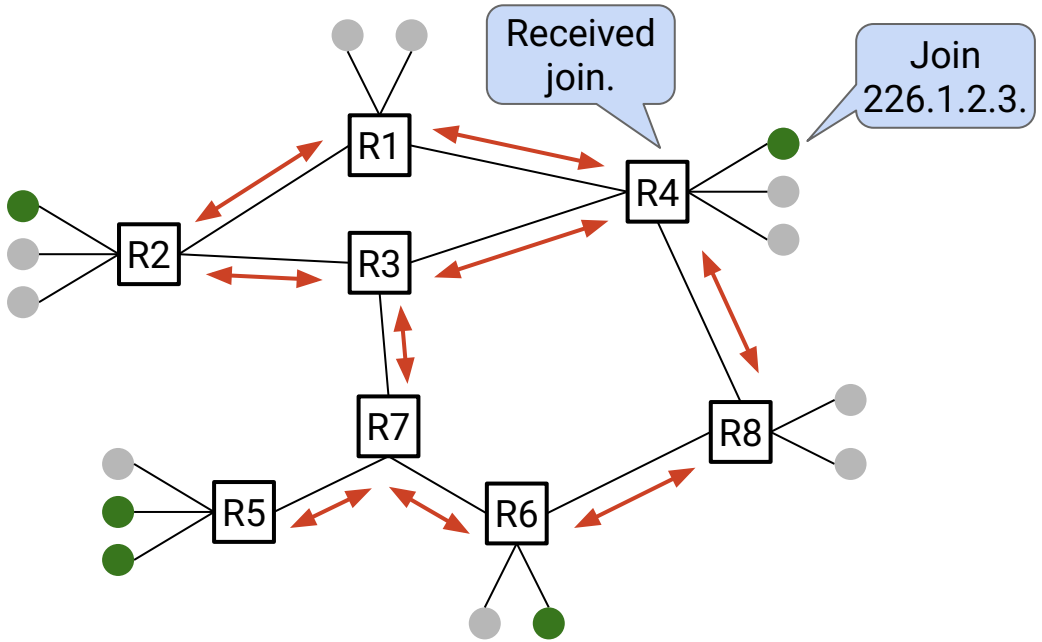
A router receives the Join packet.

Routers exchange messages to update their forwarding tables as needed.

= Not a member of group 226.1.2.3.

= Is a member of group 226.1.2.3.

The group membership has changed, so routers exchange messages to update their forwarding tables.



IP Multicast Model: Joining and Leaving

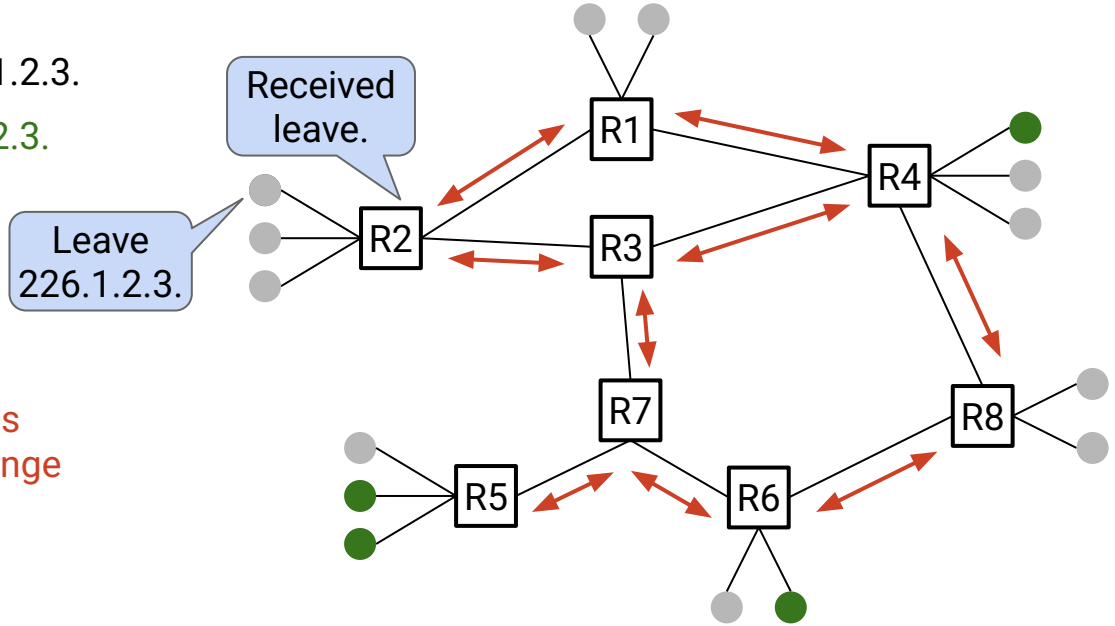
Hosts send a **Leave** packet to announce that they are leave a group.

A router receives the Leave packet.

Routers exchange messages to update their forwarding tables as needed.

= Not a member of group 226.1.2.3.

= Is a member of group 226.1.2.3.



The group membership has changed, so routers exchange messages to update their forwarding tables.

IP Multicast Model: Forwarding

When a router receives a multicast packet:

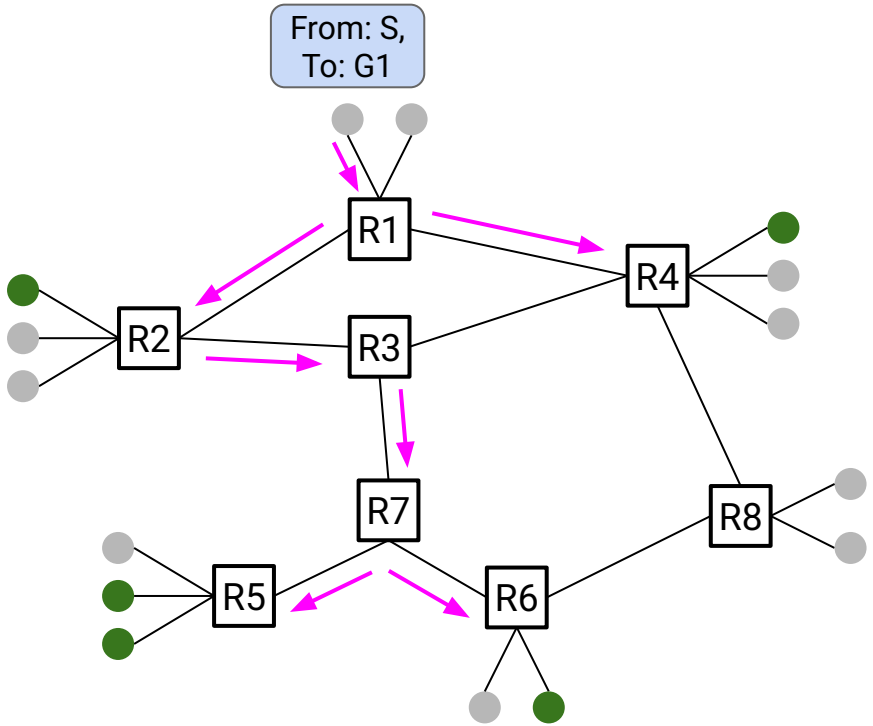
Forward packet along 0, 1, or 2+ links, so that the packet reaches all group members.

● = Not a member of group 226.1.2.3.

● = Is a member of group 226.1.2.3.

Routers forward data packet to all group members.

Notice: Packet is only forwarded once on each link. We want to avoid sending the same packet along a link multiple times.



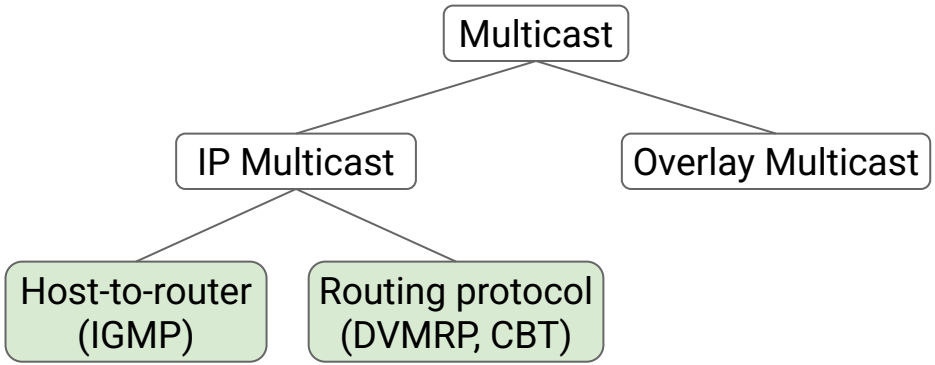
Implementing IP Multicast

The goal of routers:

- Hosts send Join, Leave, and Data packets.
- Routers must coordinate to process these packets and build forwarding tables.
- The forwarding tables say: When I receive a packet with destination group G1, what link(s) do I forward this packet out of?

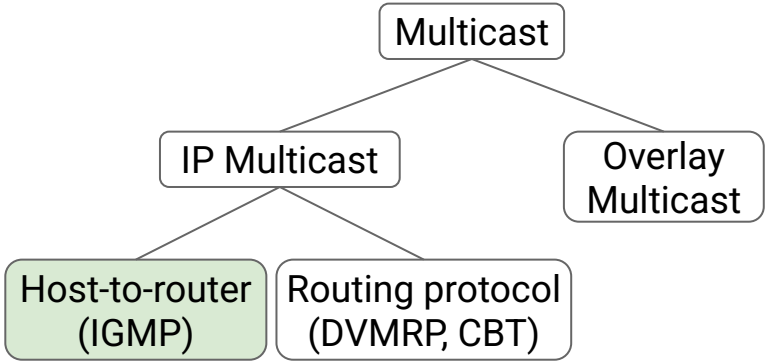
Routers coordinate with two protocols:

- **Host-to-Router**: Learns the group(s) that your directly-connected hosts belong to.
- **Routing**: Routers talk to each other to build forwarding tables.



Implementing IP Multicast: Host-to-Router

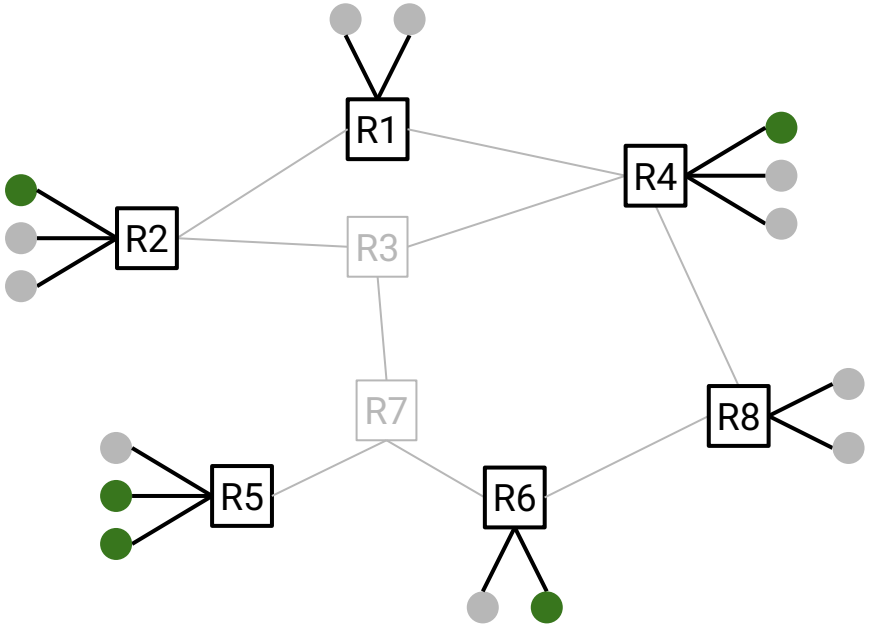
Host-to-Router protocol: Router learns the group(s) that its directly-connected hosts are members of.



Similar to static routing in unicast routing:

Learn about your directly-connected hosts.

Doesn't help you learn about other hosts in the network.

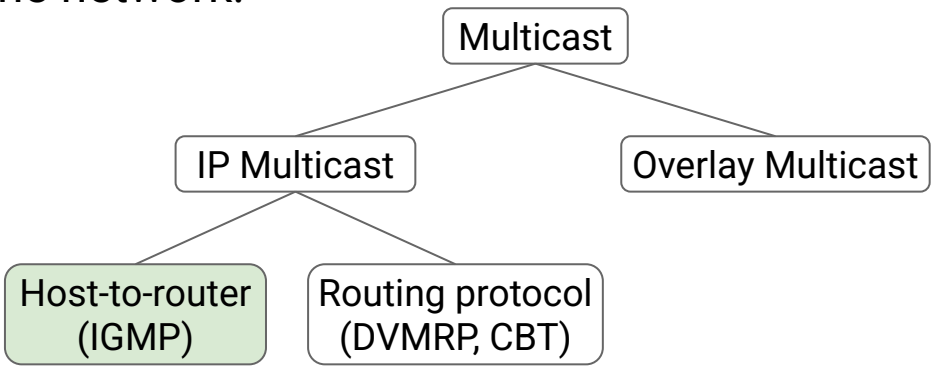


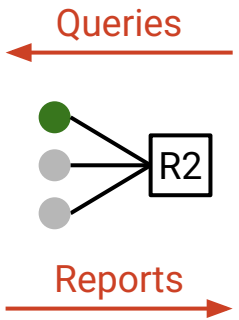
IGMP: Directly-Connected Hosts

The router and its directly-connected hosts periodically exchange messages, so that the router is informed about the hosts' group memberships.

- **Query**: Router periodically asks hosts: "What groups do you belong to?"
- **Reports**: Hosts periodically tell the router: "I belong to these groups."
- If router doesn't receive a Report for a long time, assume the membership has expired.

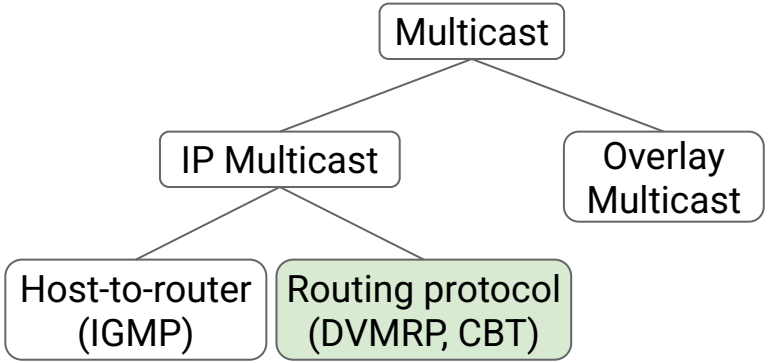
Similar to static routing in unicast: Learn about directly-connected hosts, but not other hosts in the network.





Implementing IP Multicast: Routing Protocol

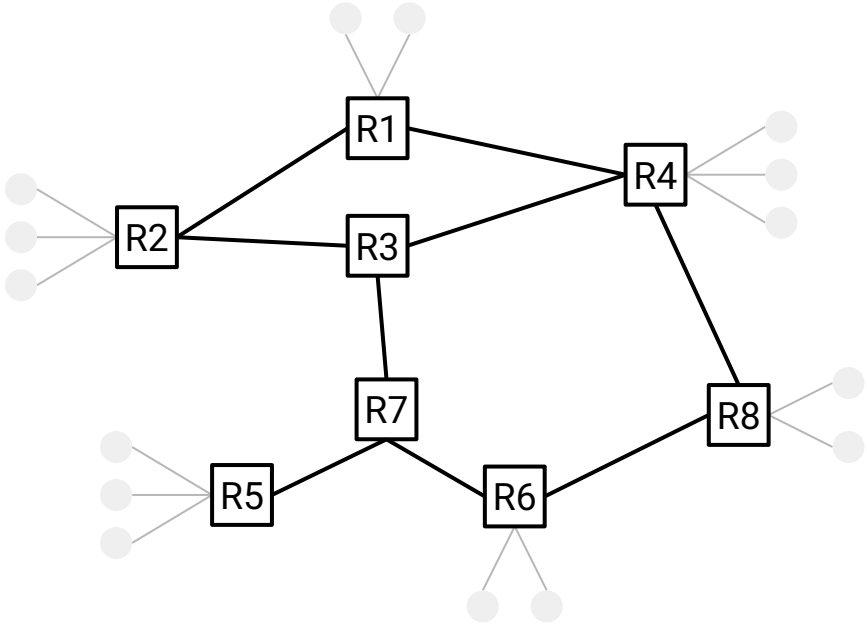
Routing protocol: Routers talk to each other to build forwarding tables, so they learn how to forward packets toward the members of the destination group.



Think of this as the multicast equivalent of distance-vector or link-state.

Two options: DVMRP or CBT.

Pick one (just like picking distance-vector or link-state).



Naive Approach: Flooding

Lecture 22, CS 168, Spring 2026

Introducing Multicast
IP Multicast
- IP Multicast Service Model

DVMRP
- **Naive Approach: Flooding**
- RPB (Broadcasting): Trees
- RPM (Multicasting): Pruning
- Pros and Cons

Core-Based Trees
- Protocol
- Pros and Cons

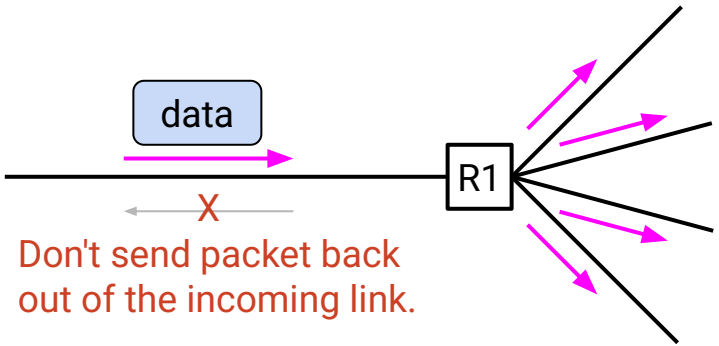
IP Multicast Challenges
Overlay-Based Multicast
- Implementation

Naive Approach: Flooding

When you receive a packet, send it out of all ports, except the incoming port.

Will this work?

Yes. The packet will reach everyone, including all group members.

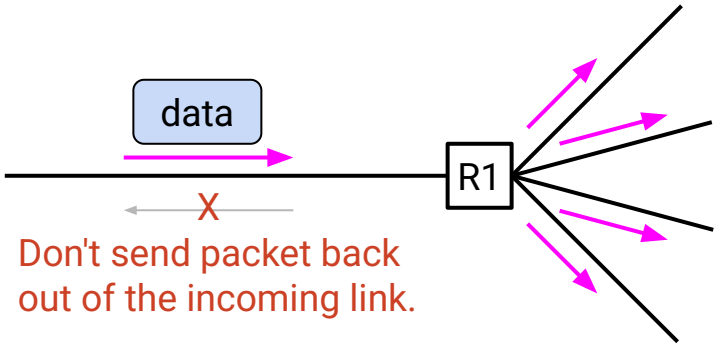


Problems with Flooding

Wastes bandwidth in two ways:

1. Packet is sent along multiple paths to the same router.
2. Packet is sent to non-members.

How would you fix this?

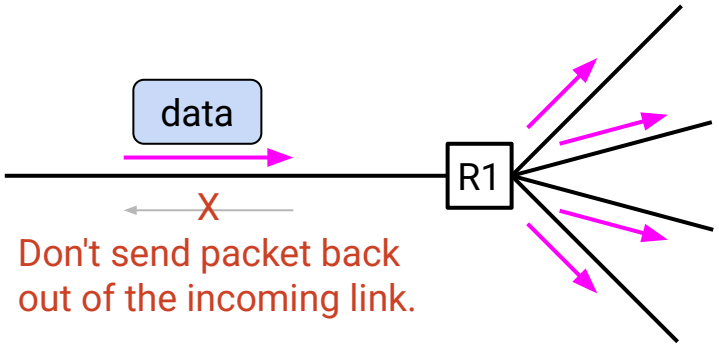


Problems with Flooding

Solutions we will learn about:

1. Packet is sent along multiple paths to the same router.
   - Solution: Send packets along a spanning tree rooted at the sender.
2. Packet is sent to non-members.
   - Solution: Prune branches of the tree with no members.

Put together, these ideas result in DVMRP.



Fine print: Technically, flooding can cause broadcast storms (packet traveling in a loop forever), but this can be solved by having routers discard a packet if they've seen it before.

Our first fix (RPB) will also eliminate broadcast storms by eliminating loops altogether.

Reverse Path Broadcasting (RPB): Trees

Lecture 22, CS 168, Spring 2026

Introducing Multicast
IP Multicast
- IP Multicast Service Model
- IGMP: Directly-Connected Hosts

DVMRP
- Naive Approach: Flooding
- **RPB (Broadcasting): Trees**
- RPM (Multicasting): Pruning
- Pros and Cons

Core-Based Trees
- Protocol
- Pros and Cons

IP Multicast Challenges
Overlay-Based Multicast
- Implementation

Problem: Redundant Paths

Problems with flooding? Wastes bandwidth in two ways:

1. Packet is sent along multiple paths to the same router.
   - Solution: RPB: Send packets along a spanning tree rooted at the sender.
2. Packet is sent to non-members.
   - Solution: RPM: Prune branches of the tree with no members.

Fixing these 2 problems will result in DVMRP.

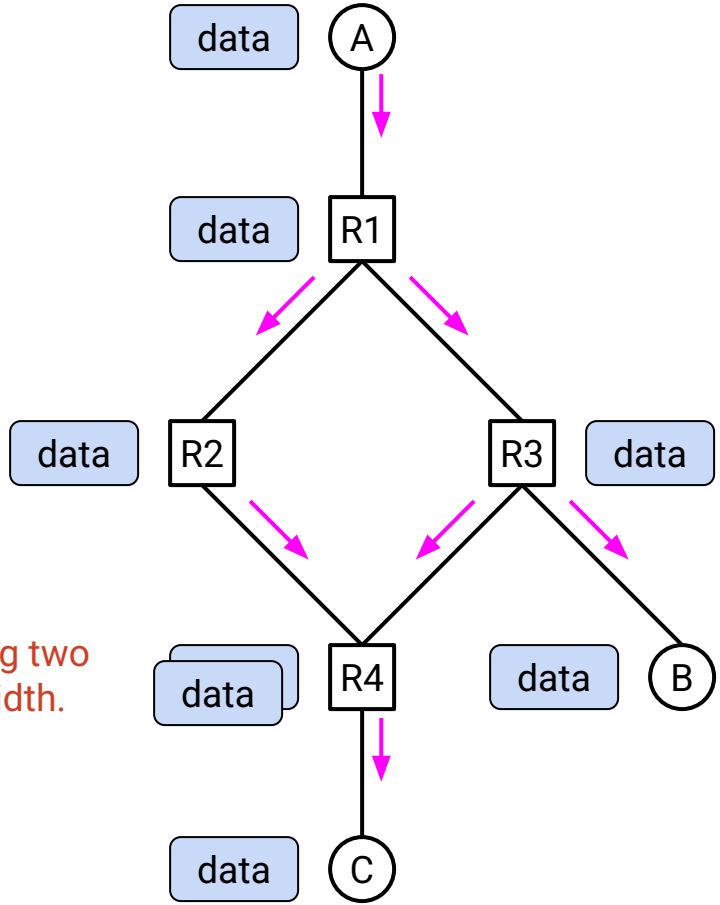
Problem: Redundant Paths

Flooding sends data along redundant paths.

Router(s) receive unnecessary copies of the data. (The copies get discarded.)

Ideally, the data should only take a single path to any router.

In other words: Each router should only receive the data once.



The data was sent to R4 twice, along two different paths! This wastes bandwidth.

Fine print: Technically, R4 will also forward a copy from R3 to R4 to R2, and another copy from R2 to R4 to R3. I didn't draw them for clarity, and it doesn't change anything, since R2 and R3 would receive the duplicates and ignore them.

Fine print: R4 received two copies of the data, but it only forwards one copy to B. The duplicate copy is discarded.

Trees Solve the Redundant Paths Problem

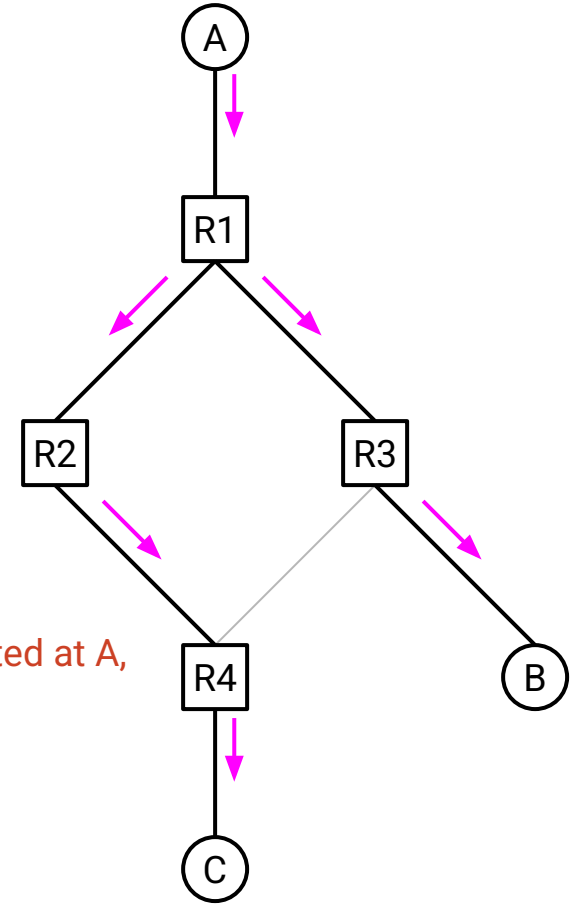
Solution: Build a spanning tree, rooted at the sender.

The tree ensures that the packet only travels along one path to any router.

(*Recall the definition of a tree: Only one path between any pair of nodes.*)

In other words: You only receive the packet once, from your parent.

(*Recall the definition of a tree: Each node has only one parent.*)



This is the spanning tree rooted at A, e.g. R4's parent is R2.

Building Spanning Trees

How do we build a tree rooted at a given host?

- Have we seen spanning trees anywhere else?

Building Spanning Trees

How do we build a tree rooted at a given host?

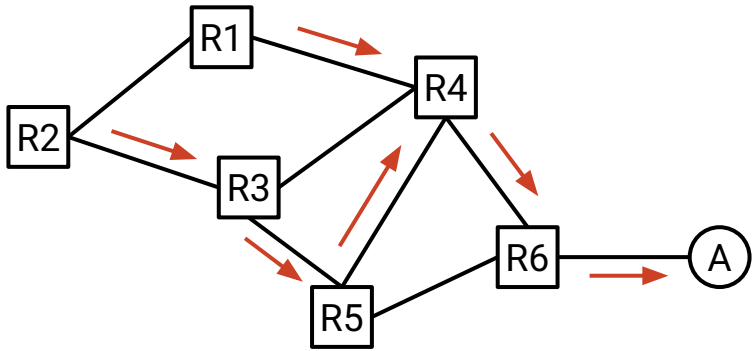
- Have we seen spanning trees anywhere else?
- When we did unicast routing, we built a tree!
- Key idea: Reuse this same tree.

Routing State Validity

A valid directed delivery tree is an **directed spanning tree** rooted at the destination.

- directed = Edges have arrows.
- Tree = No cycles, no disconnected components.
- Spanning = Tree touches every node, so every node can talk to the destination
- All edges point toward destination.

Starting at any node and following edges will reach the destination.



Building Spanning Trees

How do we build a tree rooted at a given host?

- Have we seen spanning trees anywhere else?
- When we did unicast routing, we built a tree!
- Key idea: Reuse this same tree.

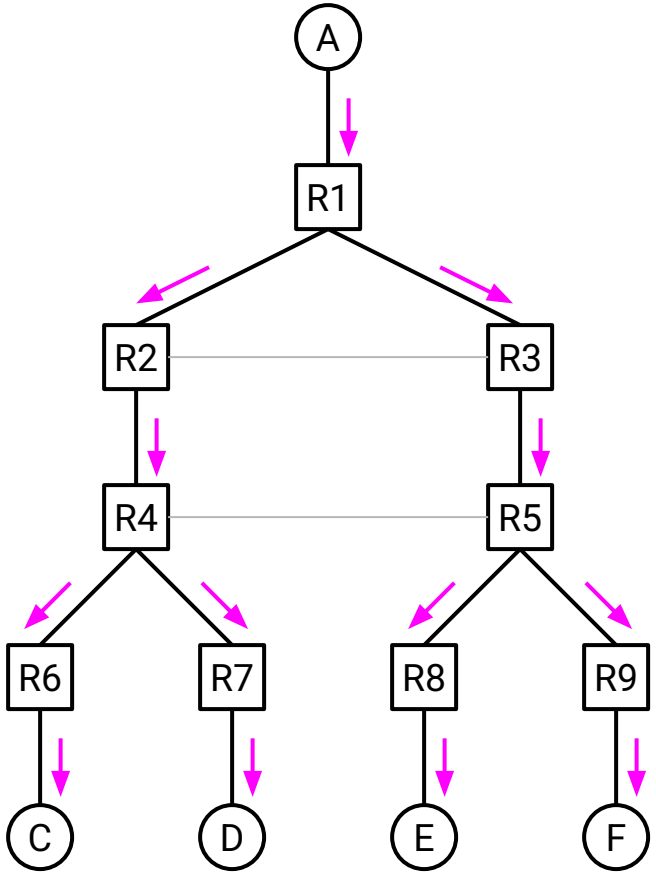
Two interpretations of the same tree:

Unicast:

- Packets flow *up* toward the root.
- The root is the destination.

Multicast:

- Packets flow *down*, from the root to everyone.
- The root is the sender.

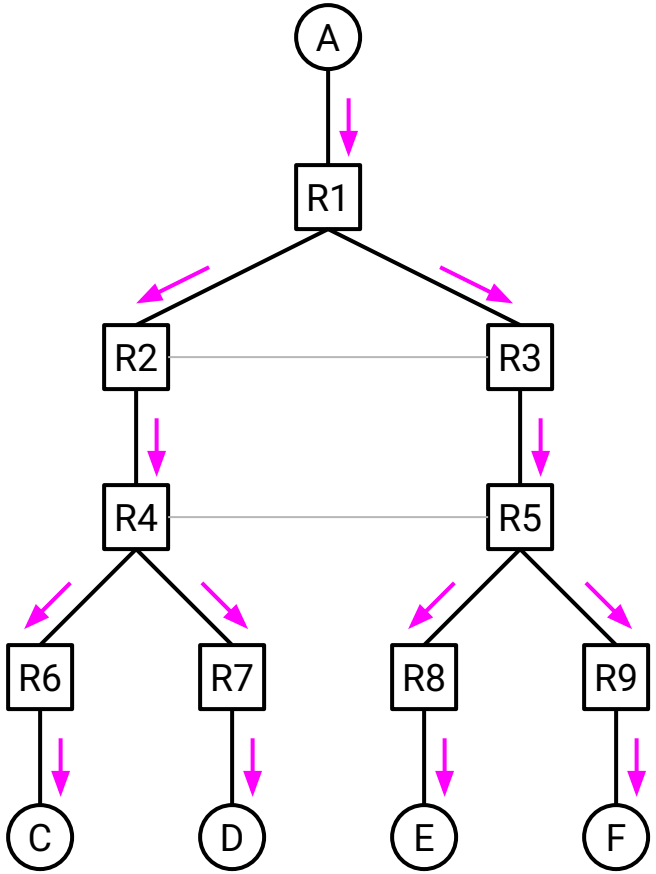


Reverse Path Broadcasting (RPB)

The **Reverse Path Broadcasting (RPB)** rule:

- Build a tree rooted at the sender.
- If you receive a packet from your parent, send it to all your children.
- Otherwise, drop the packet.

This rule solves the redundant paths problem.



Learning Parents and Children

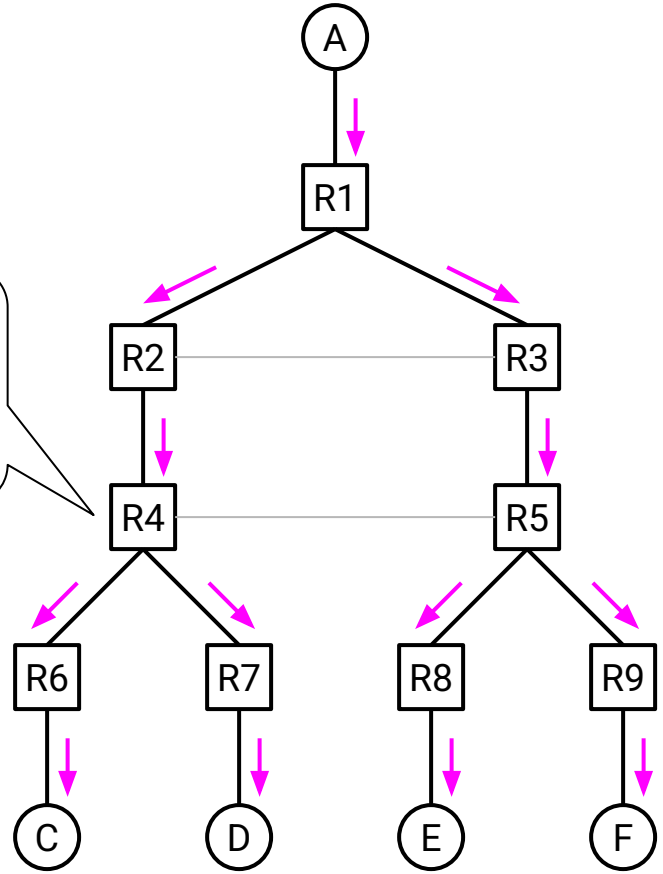
Who is your parent?

- Check your unicast forwarding table.
- Your parent is your next-hop to the root (sender).

I received this packet from **R2**: From: **A**, To: G1 data

My unicast table says the next-hop to **A** is **R2**.
So my parent to A is **R2**. I got this packet from my parent!

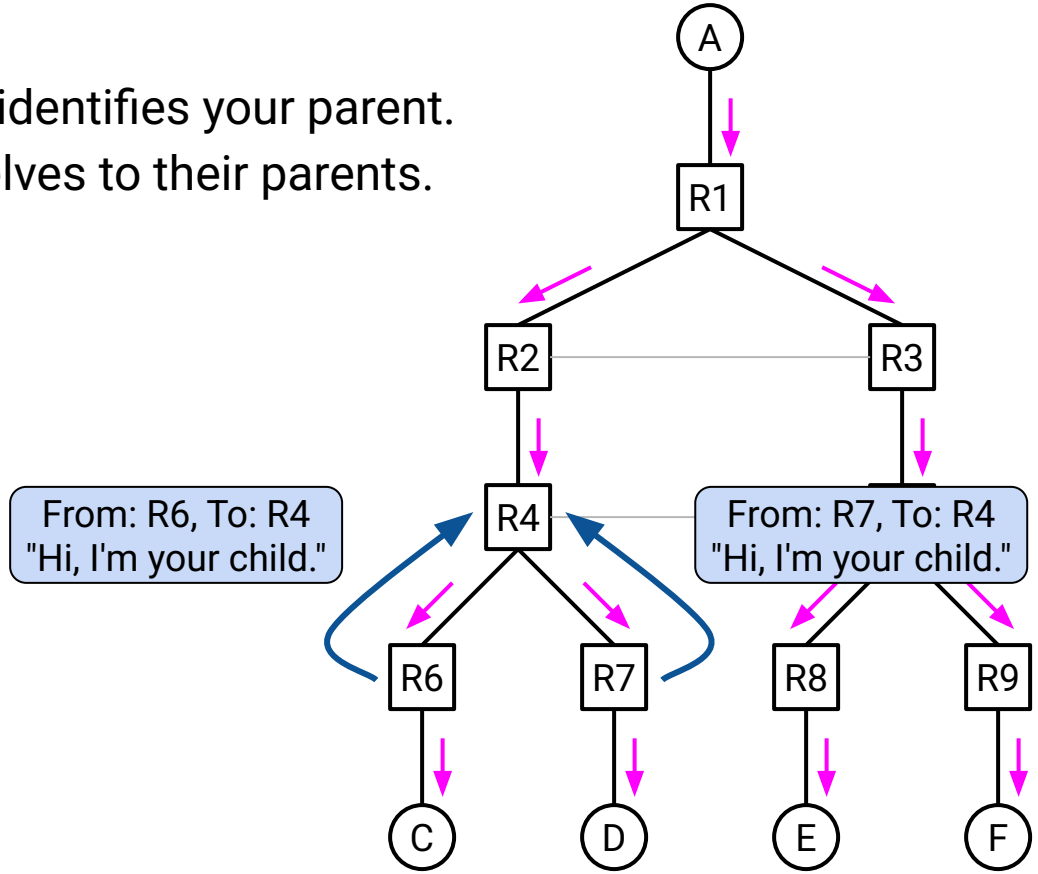
R4's Unicast Table	
To:	Next-Hop:
A	R2



Learning Parents and Children

Who are your children?

- Problem: The unicast table only identifies your parent.
- Solution: Children report themselves to their parents.



Learning Parents and Children

Who are your children?

- Problem: The unicast table only identifies your parent.
- Solution: Children report themselves to their parents.

Routers need an extra table to keep track of their children.

- Unicast table identifies your parent.
- Multicast table (new) identifies your children.

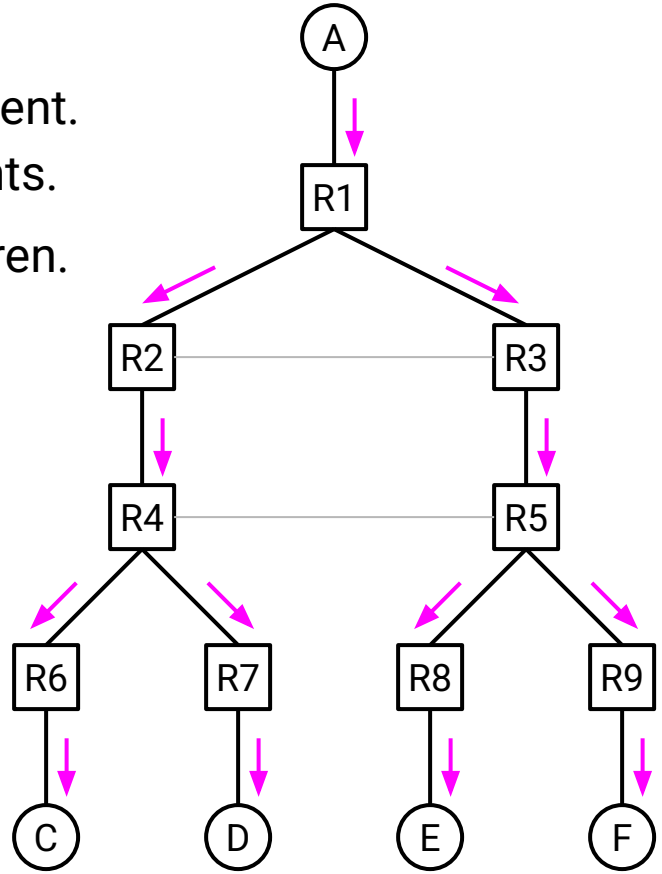
R4's Unicast Table	
To:	Next-Hop:
A	R2

R4's Multicast Table	
From:	Children:
A	R6, R7

R3's Unicast Table	
To:	Next-Hop:
A	R1

R3's Multicast Table	
From:	Children:
A	R5

(Not all tables shown.)



Learning Parents and Children: Recap

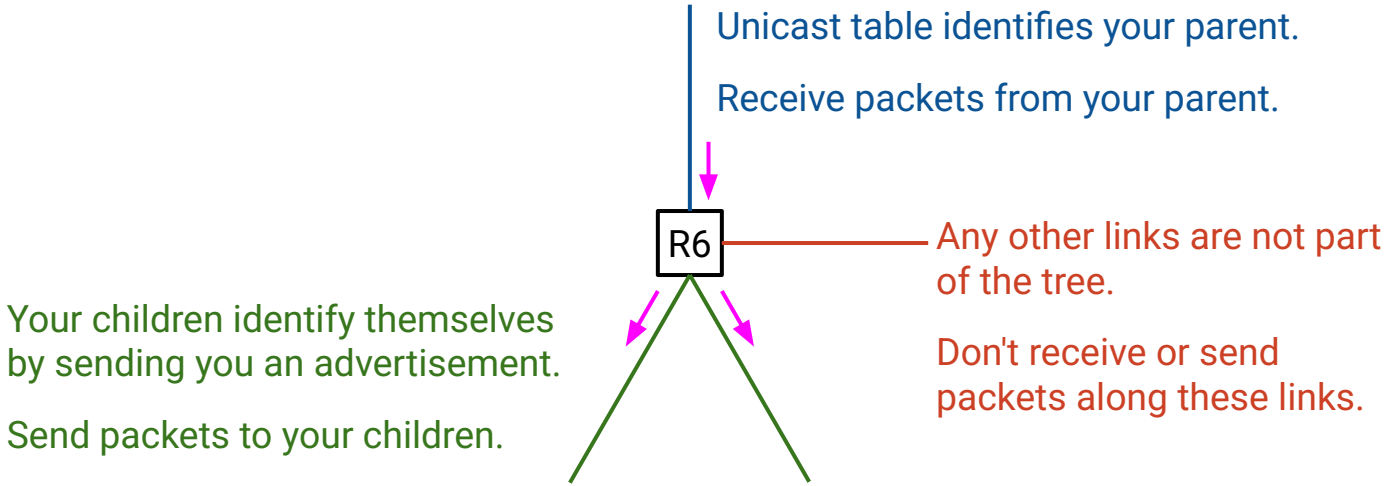
The unicast table has two interpretations:

- **Unicast**: To reach *destination* A, the next-hop is R2.
- **Multicast**: Packets from *sender* A come from parent R2.

The multicast table says: Packets from *sender* A should be forwarded to children R6, R7.

R4's Unicast Table	
To:	Next-Hop:
A	R2

R4's Multicast Table	
From:	Children:
A	R6, R7



One Tree Per Sender

We build one tree per host.

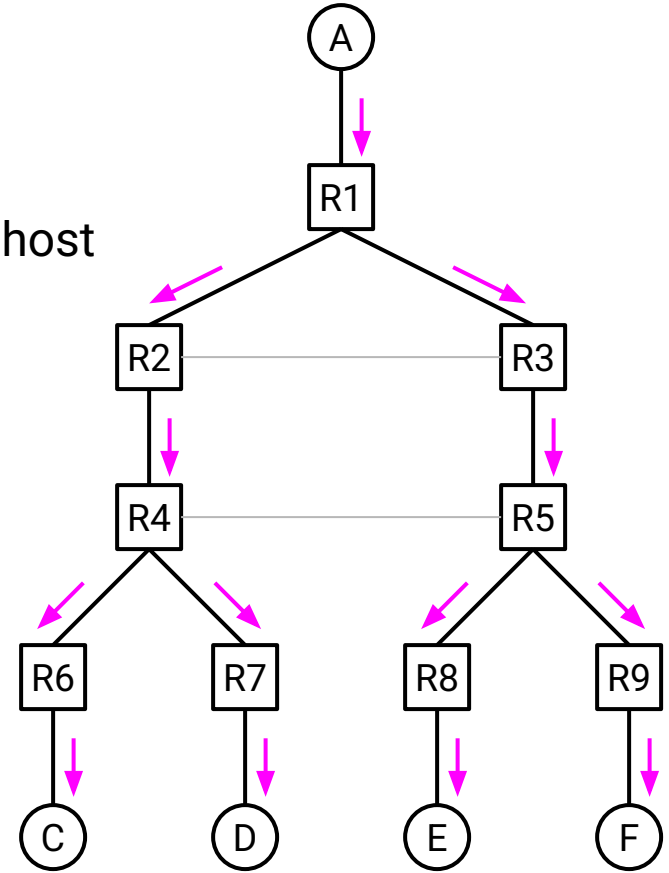
- Unicast uses the tree to send packets *to* that host (destination).
- Multicast uses the tree to send packets *from* that host (sender) to everyone else.

Both forwarding tables need one entry per tree.

R4's Unicast Table	
To:	Next-Hop:
A	R2

R4's Multicast Table	
From:	Children:
A	R6, R7

The tree rooted at A.



One Tree Per Sender

We build one tree per host.

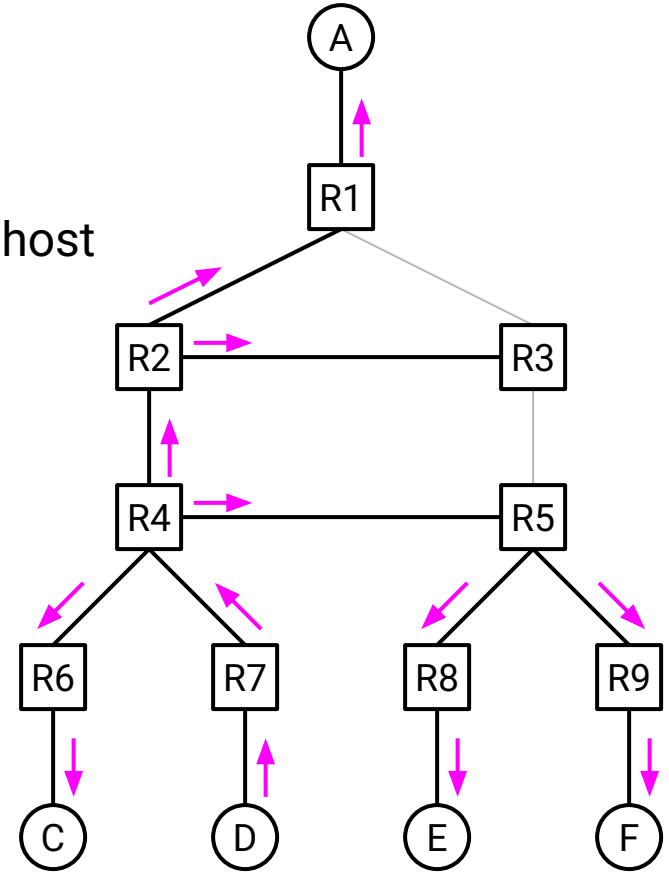
- Unicast uses the tree to send packets *to* that host (destination).
- Multicast uses the tree to send packets *from* that host (sender) to everyone else.

Both forwarding tables need one entry per tree.

R4's Unicast Table	
To:	Next-Hop:
A	R2
D	R7

R4's Multicast Table	
From:	Children:
A	R6, R7
D	R2, R5, R6

The tree rooted at D.
(Looks less like a tree, but it's still a tree.)



Reverse Path Multicasting (RPM): Pruning

Lecture 22, CS 168, Spring 2026

Introducing Multicast
IP Multicast
- IP Multicast Service Model
- IGMP: Directly-Connected Hosts

DVMRP
- Naive Approach: Flooding
- RPB (Broadcasting): Trees
- **RPM (Multicasting): Pruning**
- Pros and Cons

Core-Based Trees
- Protocol
- Pros and Cons

IP Multicast Challenges
Overlay-Based Multicast
- Implementation

Problem: Non-Members Receiving Packet

Problems with flooding? Wastes bandwidth in two ways:

1. Packet is sent along multiple paths to the same router.
   - Solution: RPB: Send packets along a spanning tree rooted at the sender.
2. Packet is sent to non-members.
   - Solution: RPM: Prune branches of the tree with no members.

Fixing these 2 problems will result in DVMRP.

Problem: Non-Members Receiving Packet

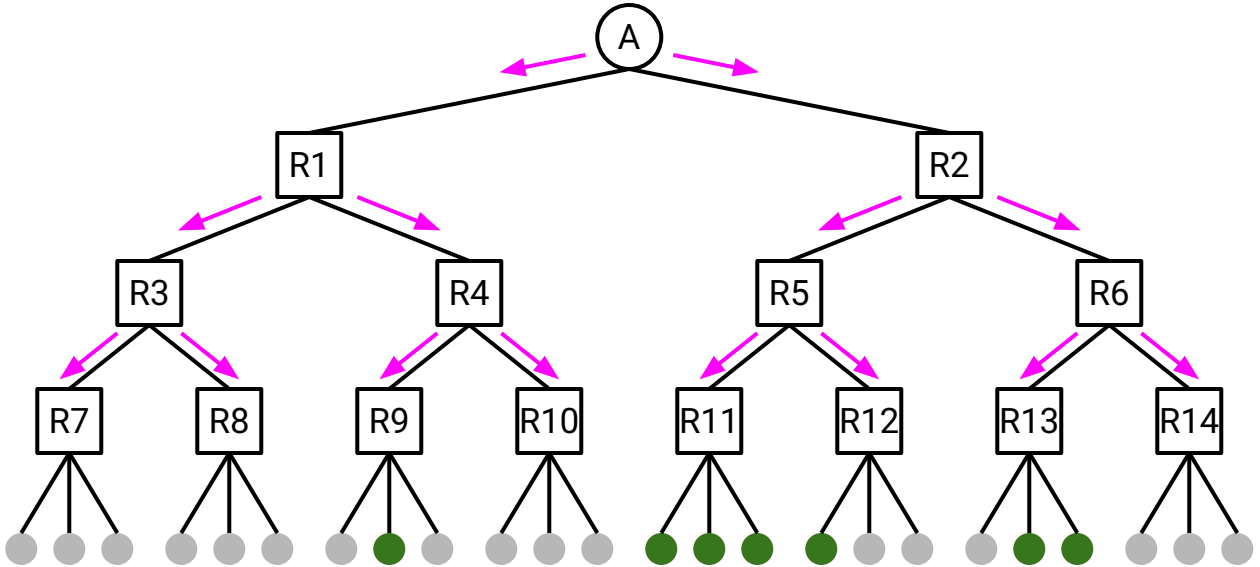
Reverse Path Broadcasting builds a tree from the sender to all hosts.

But not all hosts are group members.

Goal: **Prune** the tree to remove non-members.

= Not a member of group G1.

= Is a member of group G1.



Problem: Non-Members Receiving Packet

Reverse Path Broadcasting builds a tree from the sender to all hosts.

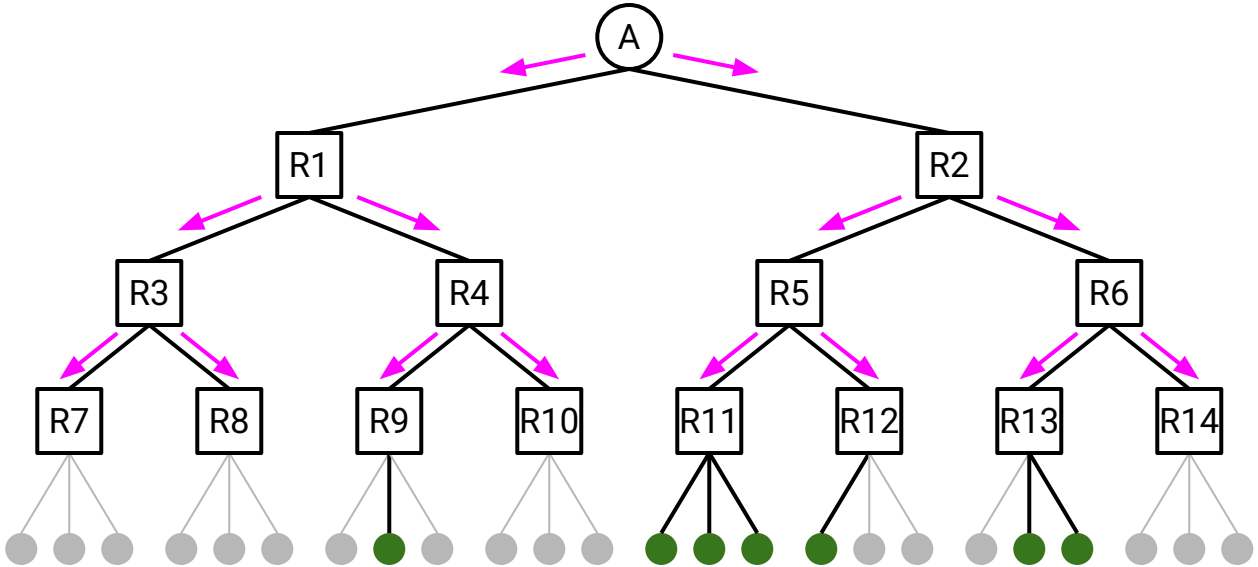
But not all hosts are group members.

Goal: **Prune** the tree to remove non-members.

● = Not a member of group G1.

● = Is a member of group G1.

"Leaf" routers don't need to send the packet to non-member hosts.



Problem: Non-Members Receiving Packet

Reverse Path Broadcasting builds a tree from the sender to all hosts.

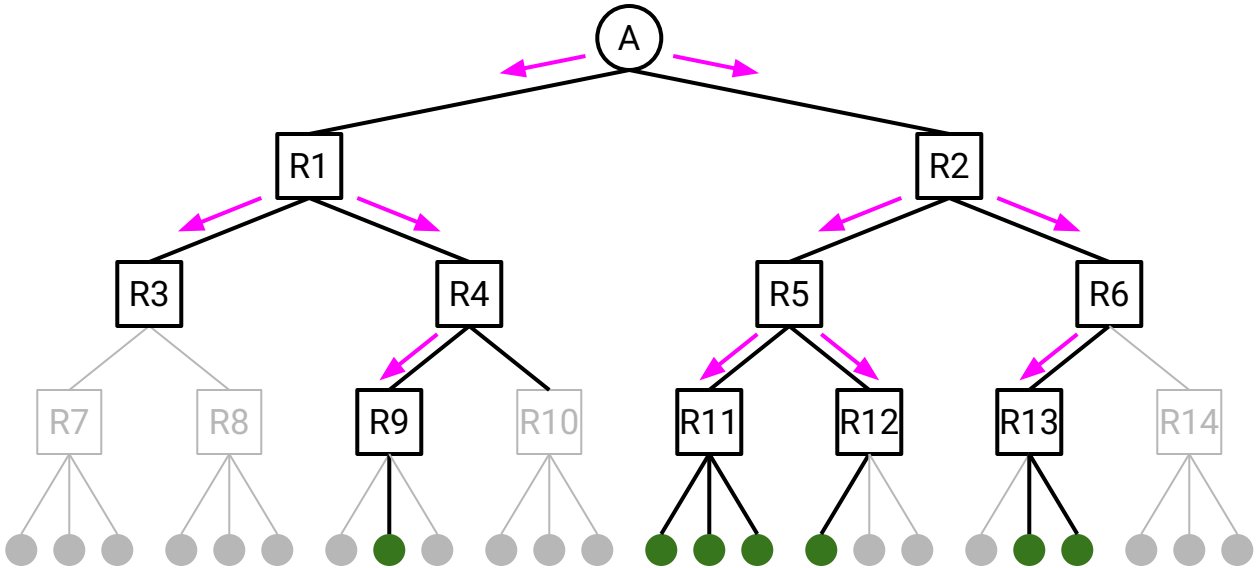
But not all hosts are group members.

Goal: **Prune** the tree to remove non-members.

● = Not a member of group G1.

● = Is a member of group G1.

"Leaf" routers with no children as members can also be pruned from the tree.



Problem: Non-Members Receiving Packet

Reverse Path Broadcasting builds a tree from the sender to all hosts.

But not all hosts are group members.

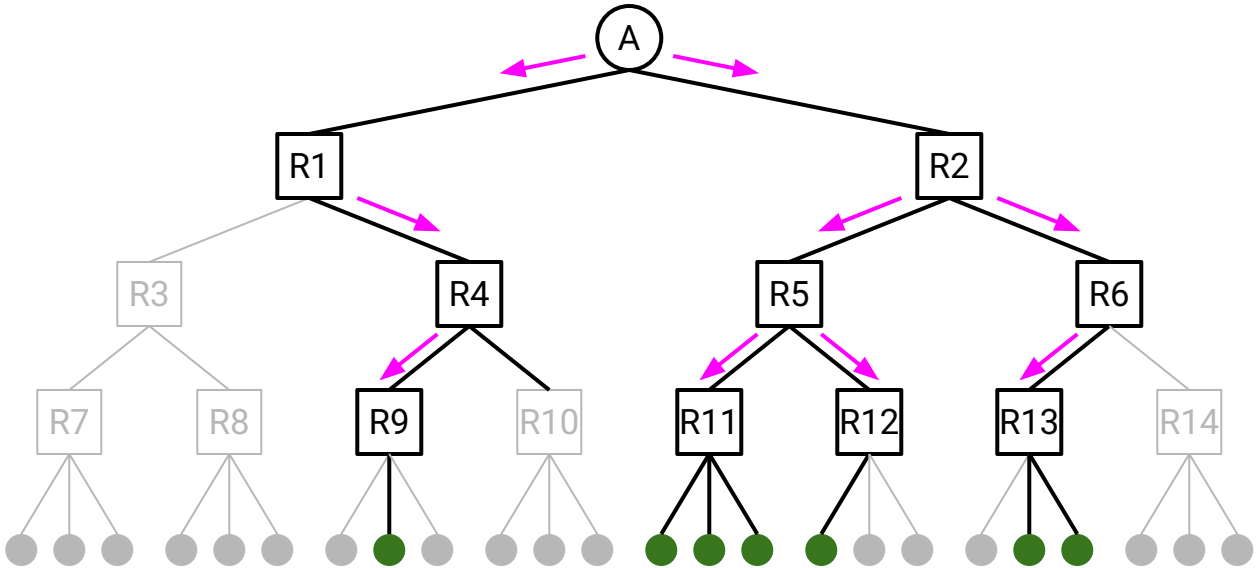
Goal: **Prune** the tree to remove non-members.

● = Not a member of group G1.

● = Is a member of group G1.

"Leaf" routers with no children as members can also be pruned from the tree.

Pruning can happen at higher "layers" as well.

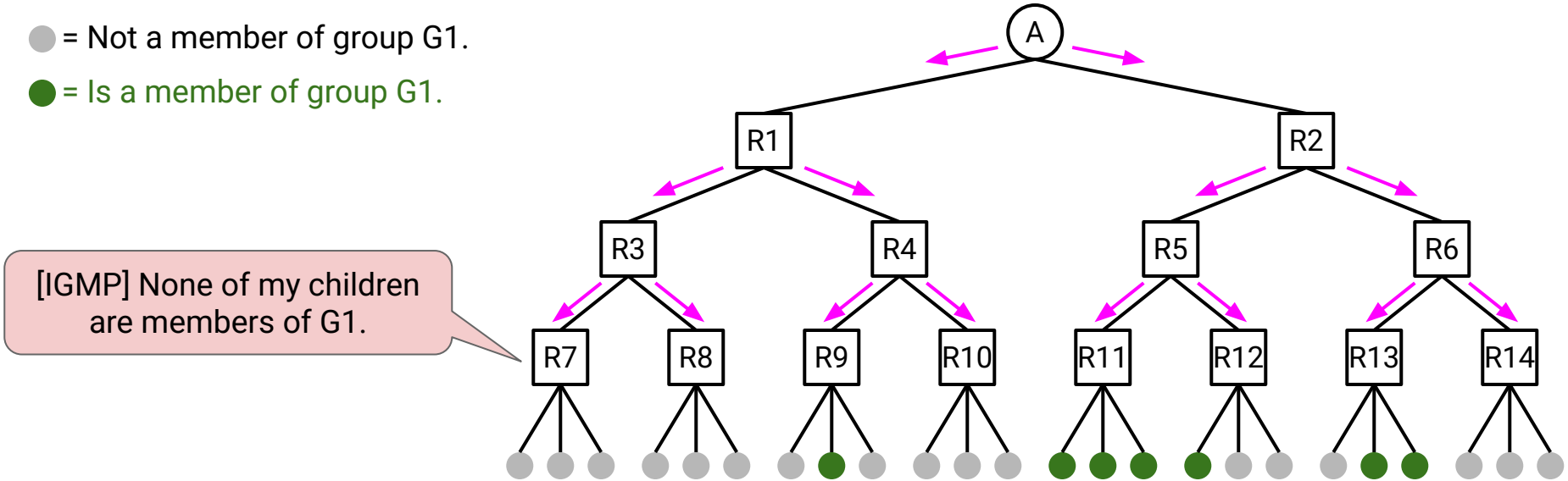


Pruning Directly-Connected Hosts with IGMP

Routers use a **host-to-router protocol** (e.g. IGMP) to check if any directly-connected hosts are group members.

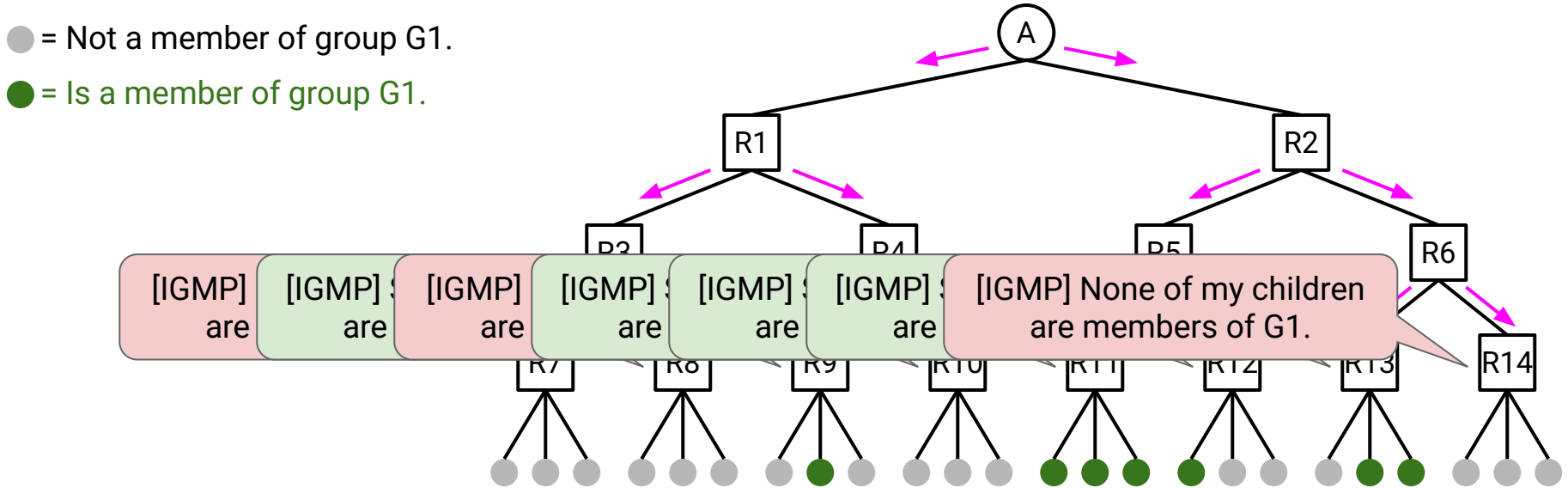
● = Not a member of group G1.

● = Is a member of group G1.



Pruning Directly-Connected Hosts with IGMP

Routers use a **host-to-router protocol** (e.g. IGMP) to check if any directly-connected hosts are group members.



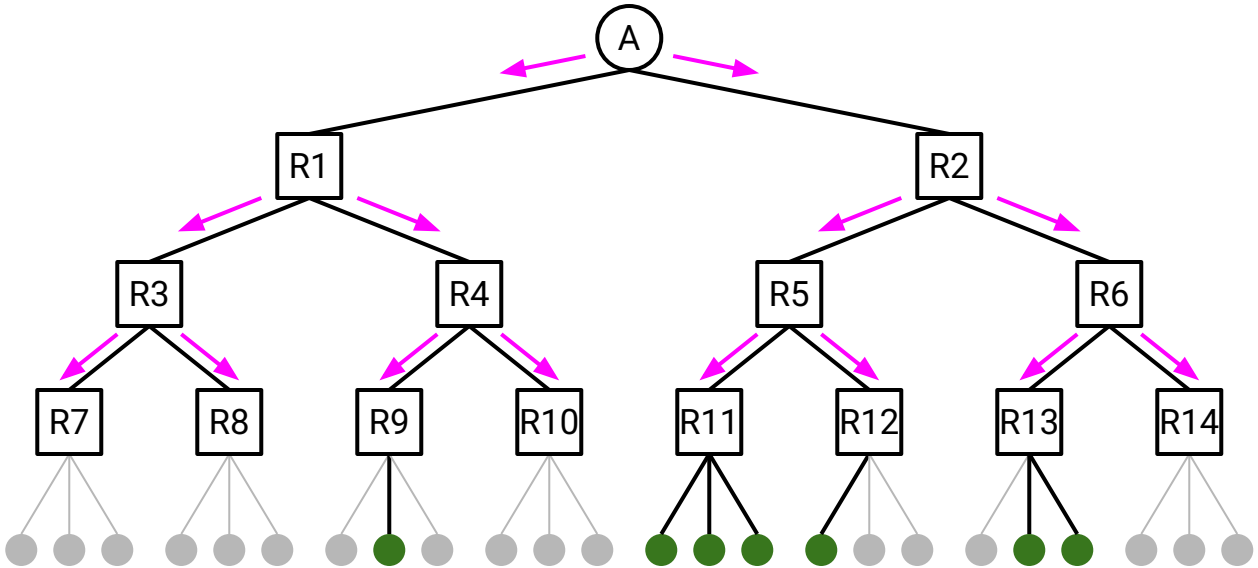
Pruning Directly-Connected Hosts with IGMP

Routers use a **host-to-router protocol** (e.g. IGMP) to check if any directly-connected hosts are group members.

Routers don't need to send packets to directly-connected hosts who are not members.

● = Not a member of group G1.

● = Is a member of group G1.



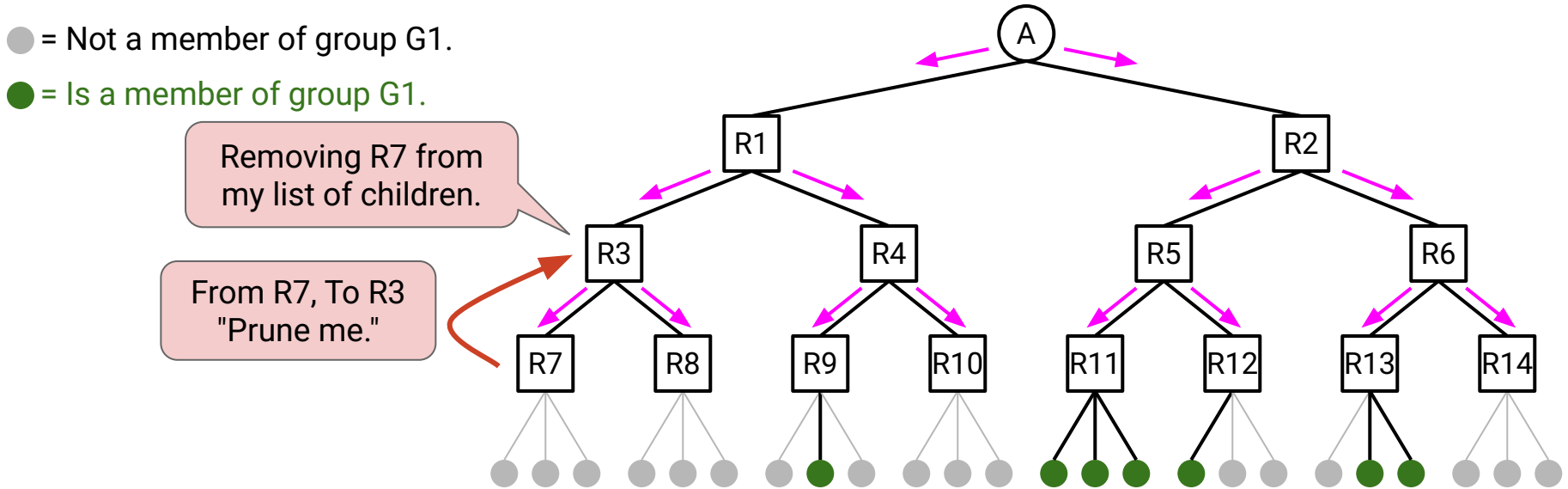
Pruning Routers with Prune Messages

If you're a router, and you notice that:

1. You have no directly-connected group members. *(Use IGMP to check this.)*
2. You have no children on the tree. *(Use your multicast table to check this.)*

Then you can be pruned. Send a prune message to your parent to inform them.

Parent can then remove you from their list of children.



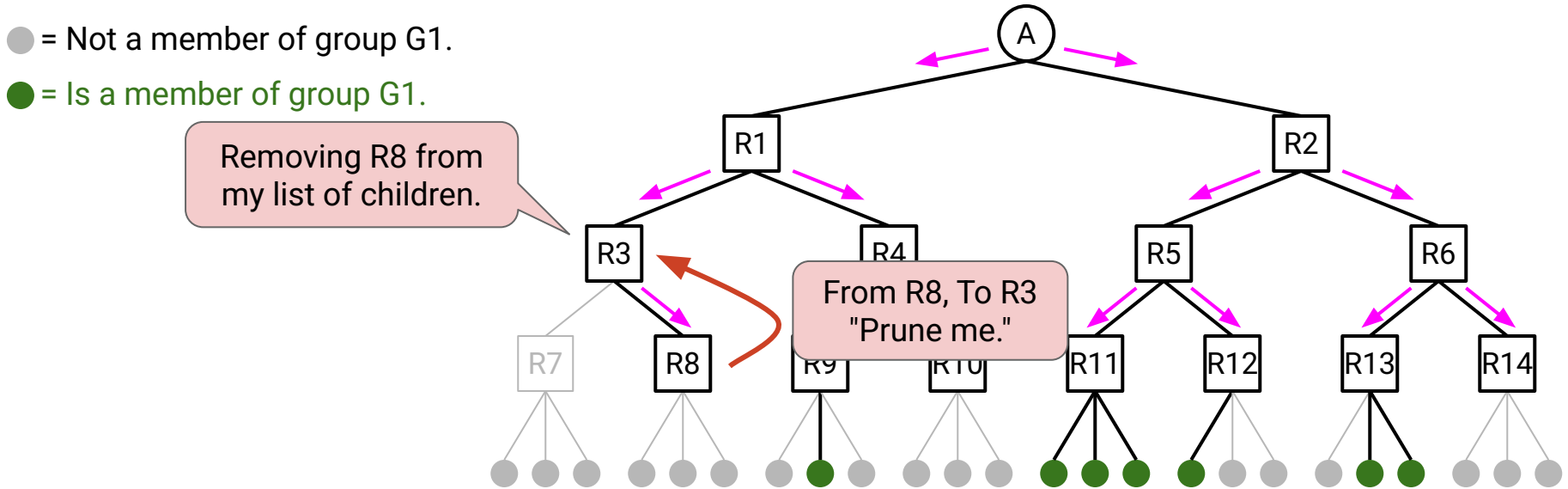
Pruning Routers with Prune Messages

If you're a router, and you notice that:

1. You have no directly-connected group members. *(Use IGMP to check this.)*
2. You have no children on the tree. *(Use your multicast table to check this.)*

Then you can be pruned. Send a prune message to your parent to inform them.

Parent can then remove you from their list of children.



Pruning Routers with Prune Messages

If you're a router, and you notice that:

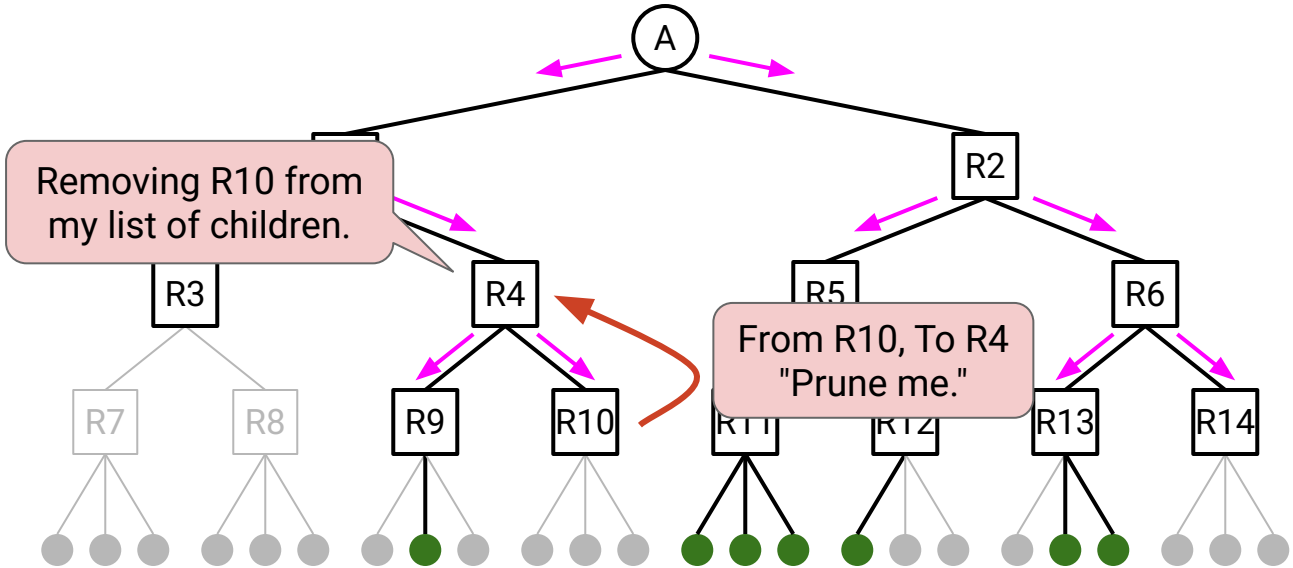
1. You have no directly-connected group members. *(Use IGMP to check this.)*
2. You have no children on the tree. *(Use your multicast table to check this.)*

Then you can be pruned. Send a prune message to your parent to inform them.

Parent can then remove you from their list of children.

● = Not a member of group G1.

● = Is a member of group G1.



Pruning Routers with Prune Messages

If you're a router, and you notice that:

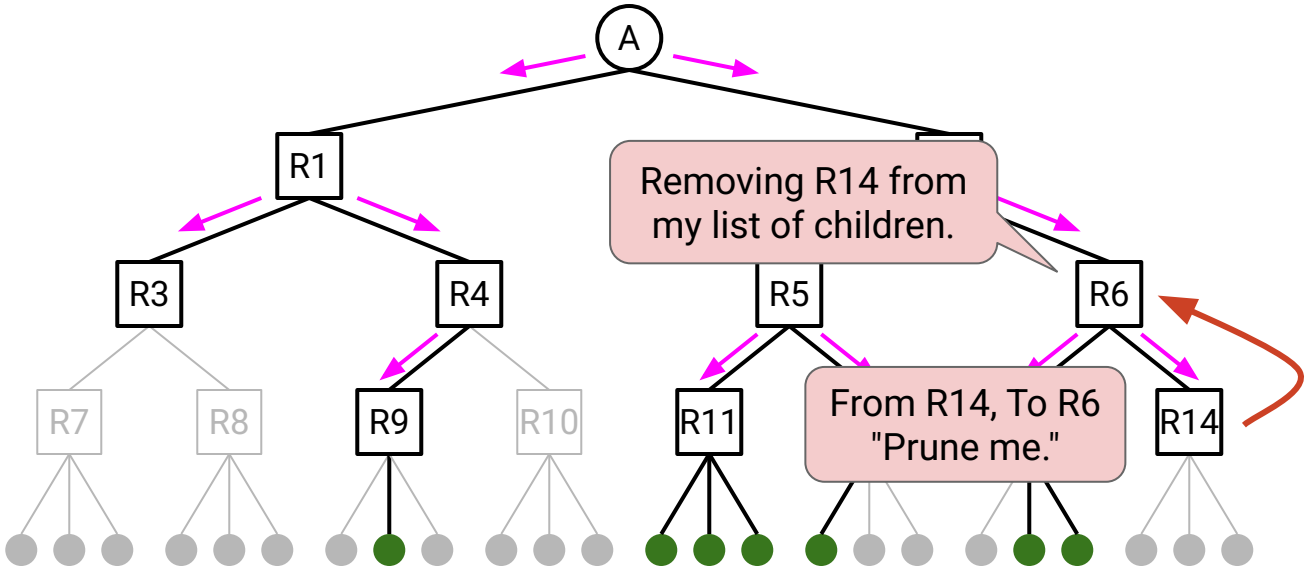
1. You have no directly-connected group members. *(Use IGMP to check this.)*
2. You have no children on the tree. *(Use your multicast table to check this.)*

Then you can be pruned. Send a prune message to your parent to inform them.

Parent can then remove you from their list of children.

● = Not a member of group G1.

● = Is a member of group G1.



Pruning Routers with Prune Messages

If you're a router, and you notice that:

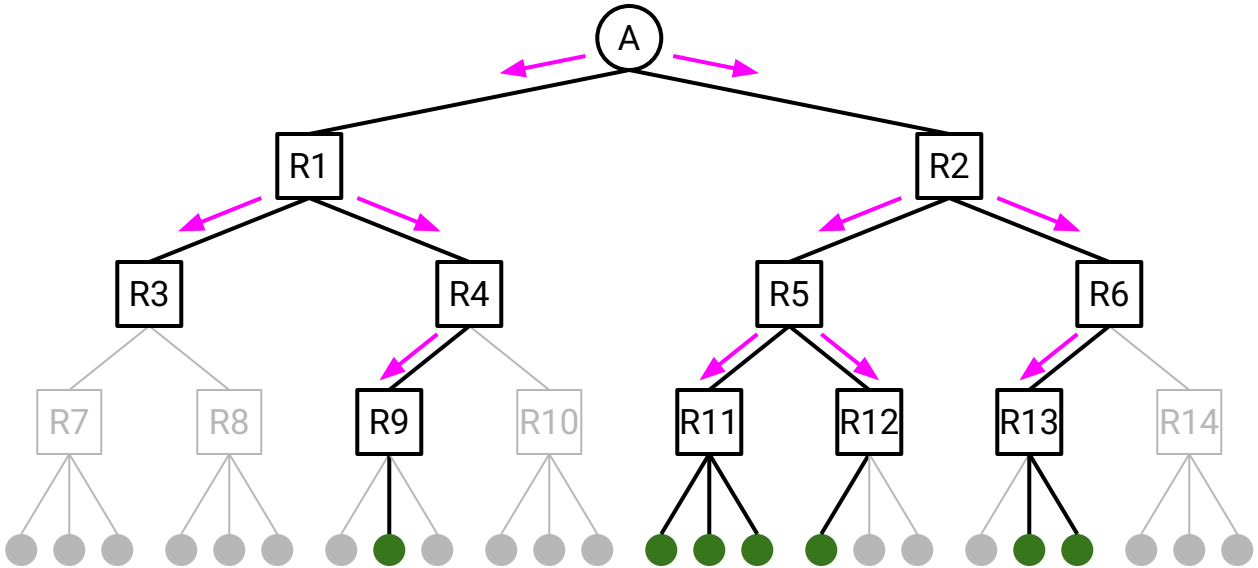
1. You have no directly-connected group members. *(Use IGMP to check this.)*
2. You have no children on the tree. *(Use your multicast table to check this.)*

Then you can be pruned. Send a prune message to your parent to inform them.

Parent can then remove you from their list of children.

● = Not a member of group G1.

● = Is a member of group G1.



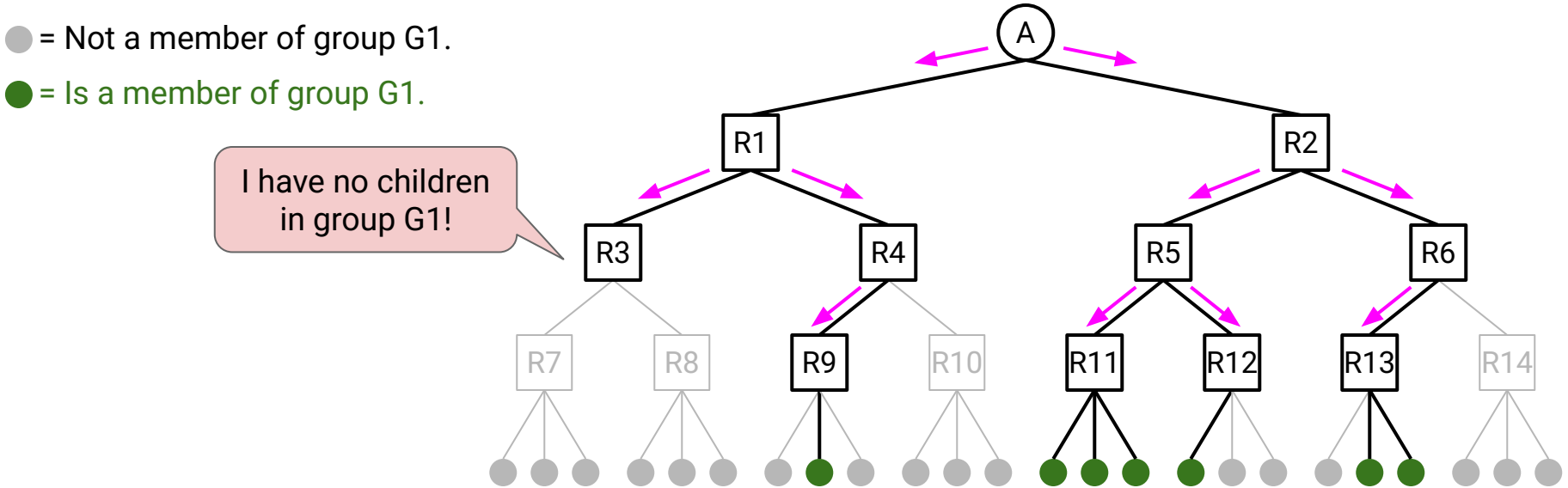
Pruning Routers with Prune Messages

If you're a router, and you notice that:

1. You have no directly-connected group members. *(Use IGMP to check this.)*
2. You have no children on the tree. *(Use your multicast table to check this.)*

Then you can be pruned. Send a prune message to your parent to inform them.

Parent can then remove you from their list of children.



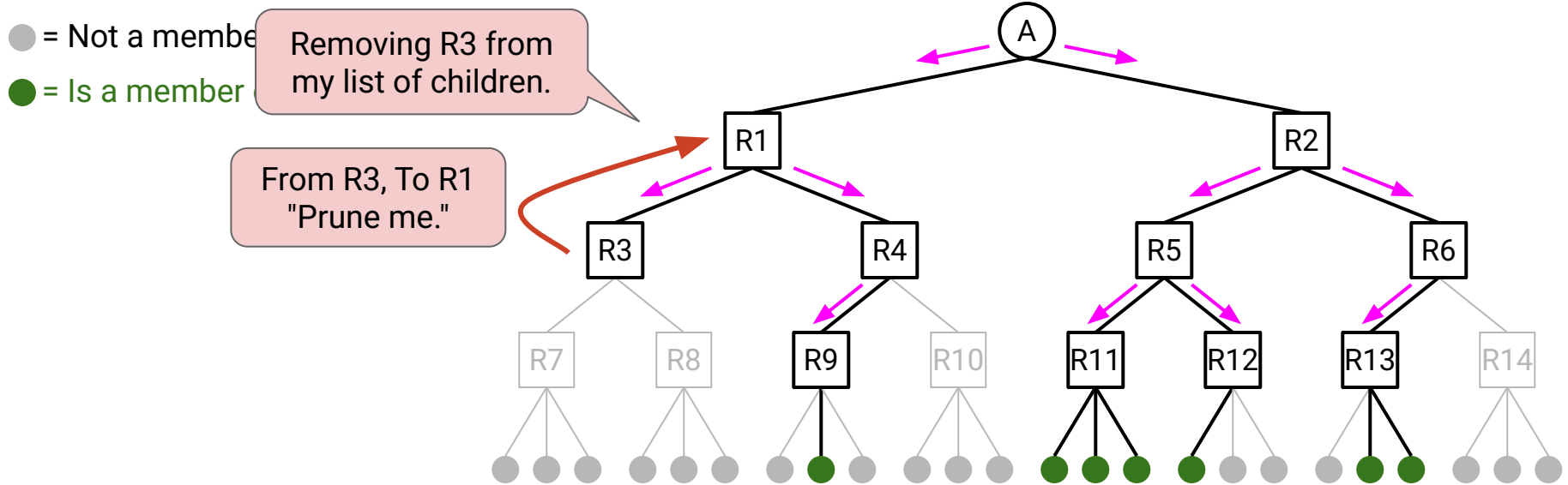
Pruning Routers with Prune Messages

If you're a router, and you notice that:

1. You have no directly-connected group members. *(Use IGMP to check this.)*
2. You have no children on the tree. *(Use your multicast table to check this.)*

Then you can be pruned. Send a prune message to your parent to inform them.

Parent can then remove you from their list of children.



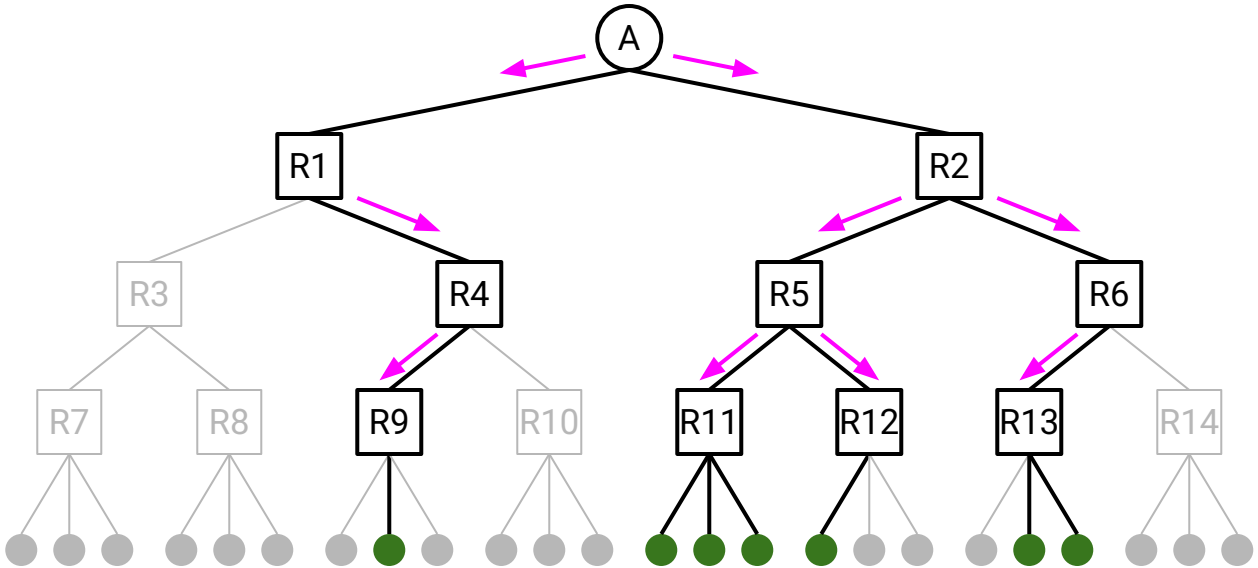
Pruning Routers with Prune Messages

Before pruning: Tree from sender to all hosts.

After pruning: Tree from sender to all group members only.

= Not a member of group G1.

= Is a member of group G1.



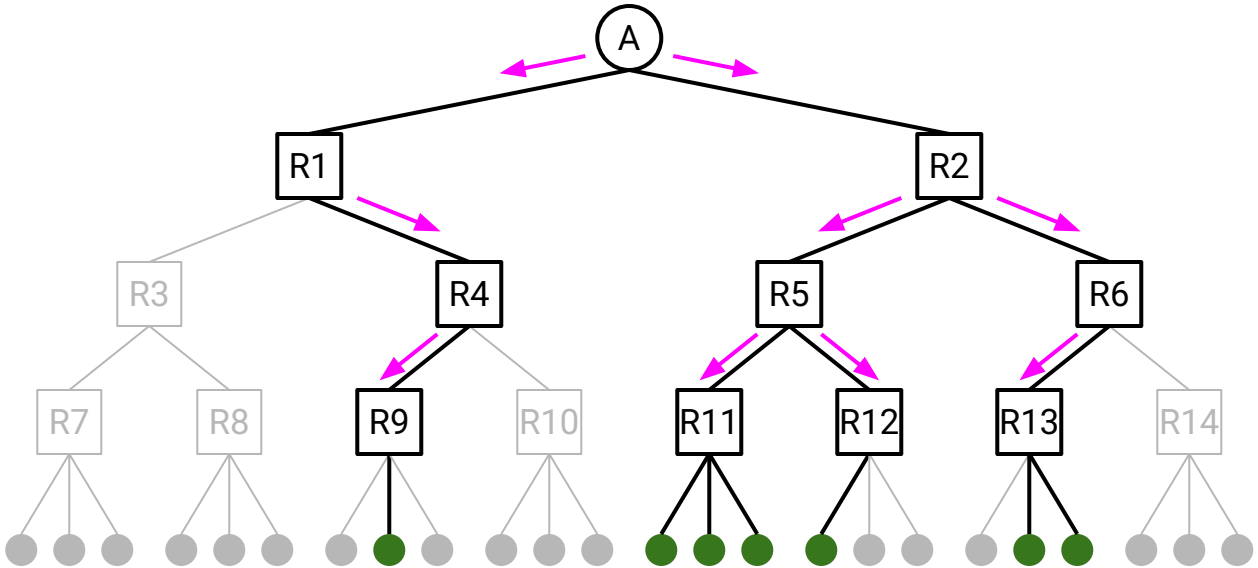
Periodically Clear Prune States

Periodically clear all pruning information (revert to forwarding to all children).

- If your children are still not group members, you'll get pruned again.
- If your children have joined the group, they'll start receiving messages.

● = Not a member of group G1.

● = Is a member of group G1.



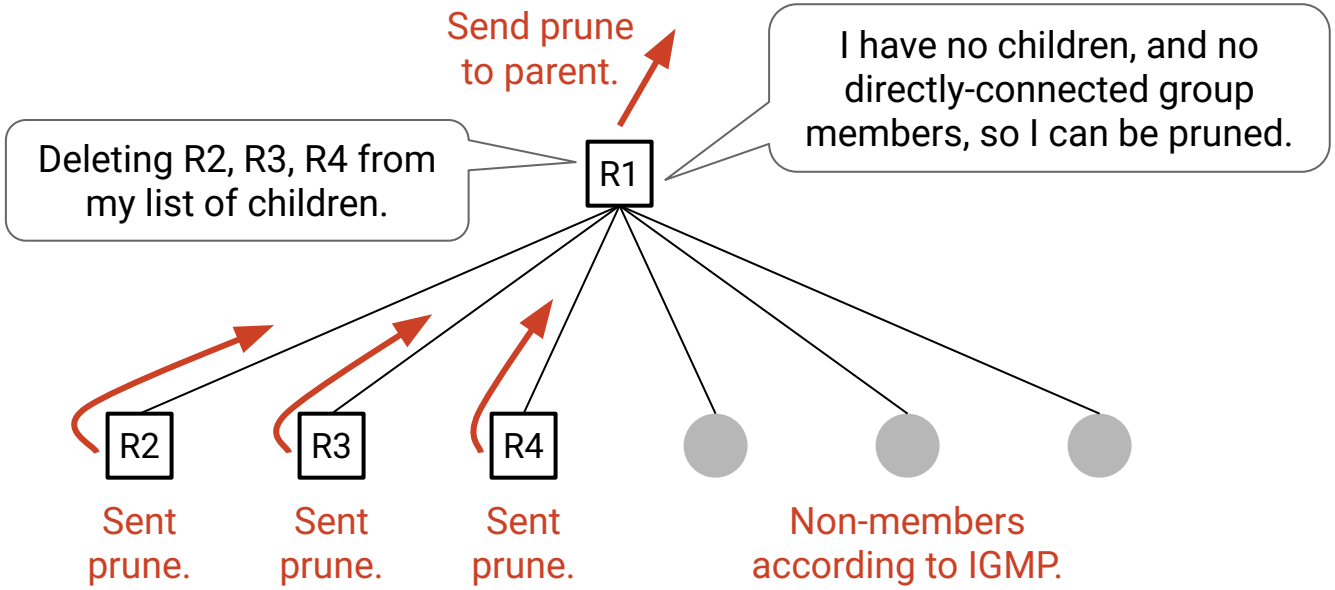
Pruning Routers with Prune Messages

If you're a router, and you notice that:

1. You have no directly-connected group members. *(Use IGMP to check this.)*
2. You have no children on the tree. *(Use your multicast table to check this.)*

Then you can be pruned. Send a prune message to your parent to inform them.

Parent can then remove you from their list of children.



One Tree Per Sender, Per Destination Group

We need to build a tree from *each sender* to *each destination group*.

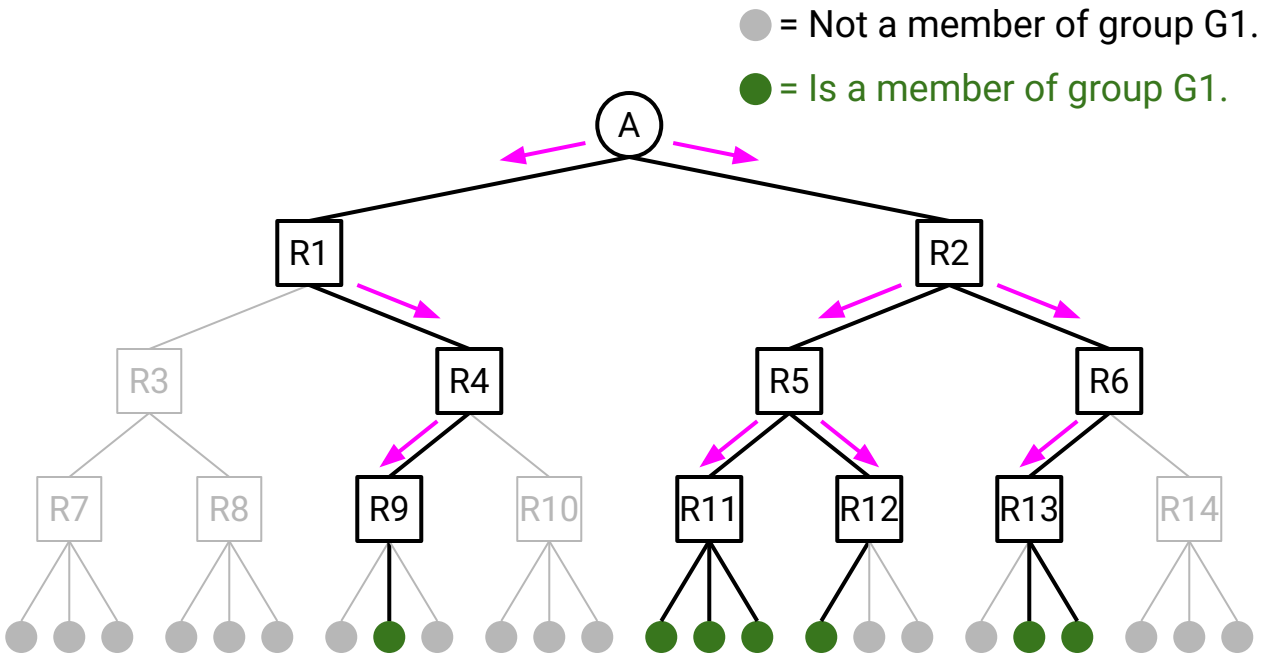
- We need one tree from source A to all members of G1...

R1's Multicast Table		
From:	To:	Children:
A	G1	R4

R4's Multicast Table		
From:	To:	Children:
A	G1	R9

R5's Multicast Table		
From:	To:	Children:
A	G1	R11, R12

(Not all tables shown.)



One Tree Per Sender, Per Destination Group

We need to build a tree from *each sender* to *each destination group*.

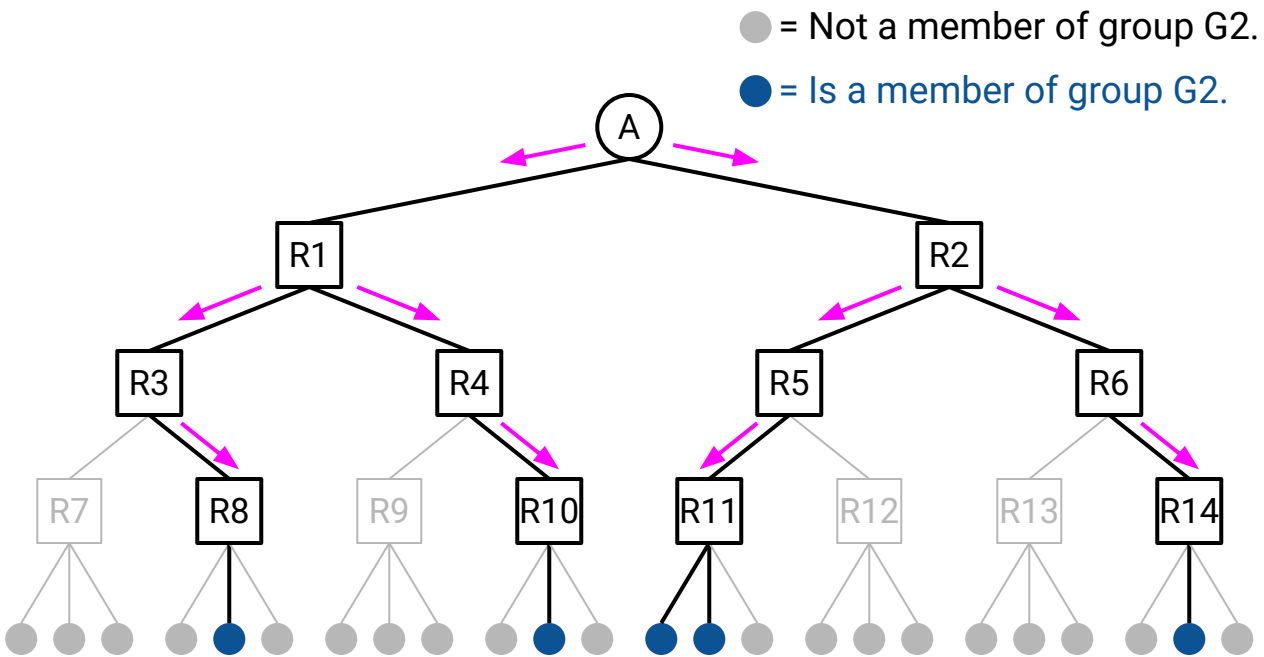
- We need one tree from source A to all members of G1...
- ...and another tree from source A to all members of G2, and so on...

R1's Multicast Table		
From:	To:	Children:
A	G1	R4
A	G2	R3, R4

R4's Multicast Table		
From:	To:	Children:
A	G1	R9
A	G2	R10

R5's Multicast Table		
From:	To:	Children:
A	G1	R11, R12
A	G2	R11

(Not all tables shown.)



Recap: DVMRP

Routing: Build a spanning tree from the source to all hosts.

- Learning parent: Use unicast forwarding table.
- Learning children: Everyone sends advertisement to parent.
  Multicast table has a list of your children.

Pruning: Trim the tree to make a tree from the source to all members only.

- If you receive a prune from a child, remove the child from your multicast table.
- If none of your descendants (directly-connected hosts or children) belong to this group, send a pruning message to your parent.
- Periodically clear all pruning information (revert to forwarding to all children).

Forwarding: Packets are forwarded from your parent to your children.

- Use unicast table to check if packet is from parent.
- If so, use multicast table to forward packet to children. Else, drop the packet.

DVMRP Pros and Cons

Pro: DVMRP is a simple, elegant extension to an existing protocol.

- We reused the unicast forwarding table to save ourselves some work.

Pro: DVMRP gives optimal performance.

- Spanning tree gives us the least-cost paths from source to all group members.
- Multicast packets travel least-cost paths to their destinations.

Con: DVMRP scales poorly.

- Pruning information is periodically cleared.
  Packets are occasionally broadcast to everybody, wasting bandwidth.
- Multicast table needs one entry per source, per destination group.

Core-Based Trees

Lecture 22, CS 168, Spring 2026

Introducing Multicast
IP Multicast
- IP Multicast Service Model
- IGMP: Directly-Connected Hosts

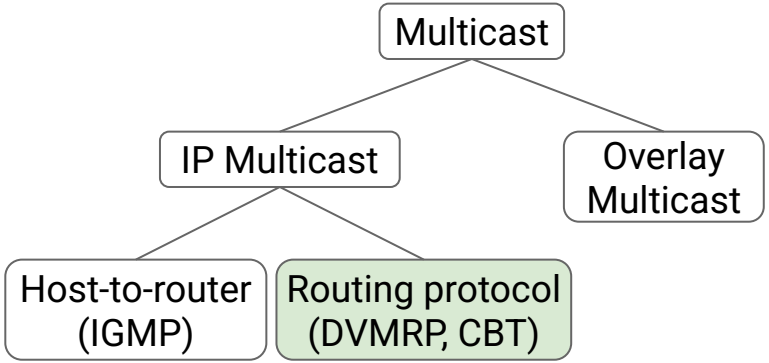
DVMRP
- Naive Approach: Flooding
- RPB (Broadcasting): Trees
- RPM (Multicasting): Pruning
- Pros and Cons

Core-Based Trees
- **Protocol**
- Pros and Cons

IP Multicast Challenges
Overlay-Based Multicast
- Implementation

Next Up: CBT

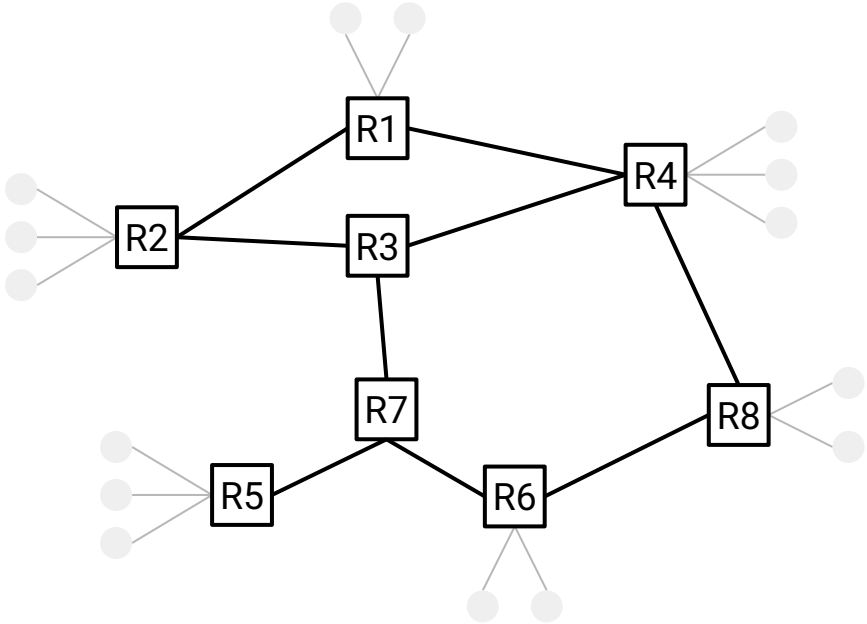
We'll now look at a different routing protocol for routers to learn how to forward multicast packets to their destination groups.



Two options: DVMRP or CBT.

Pick one (just like picking distance-vector or link-state).

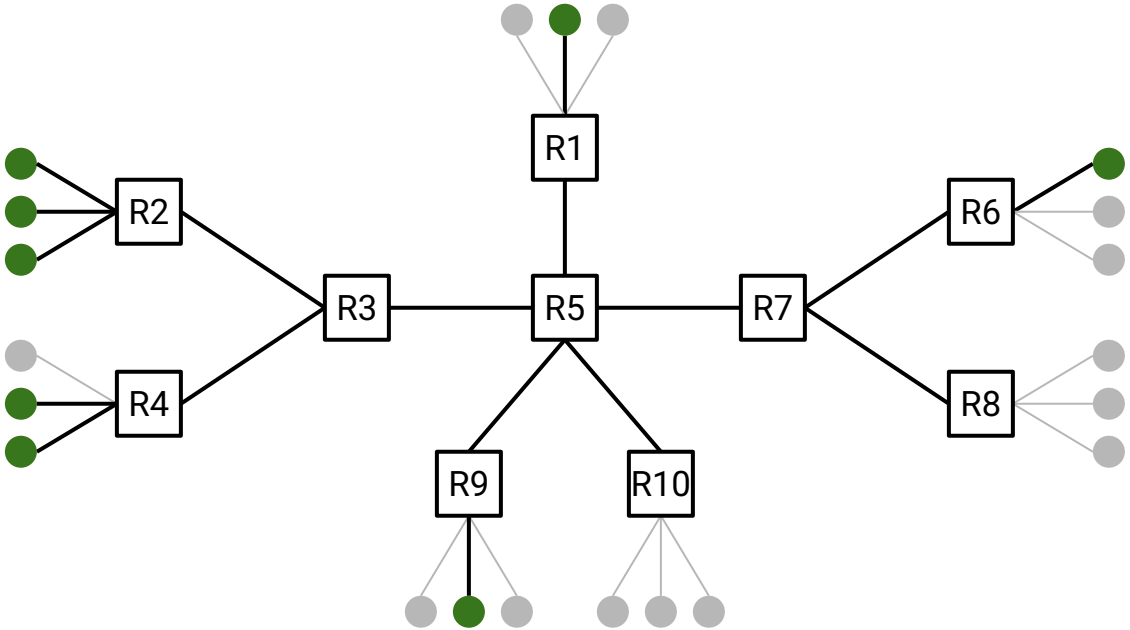
We'll show DVMRP first, then CBT – but remember they are two options for the same goal.



CBT: Definition

A **Core-Based Tree (CBT)** for a given destination group is:

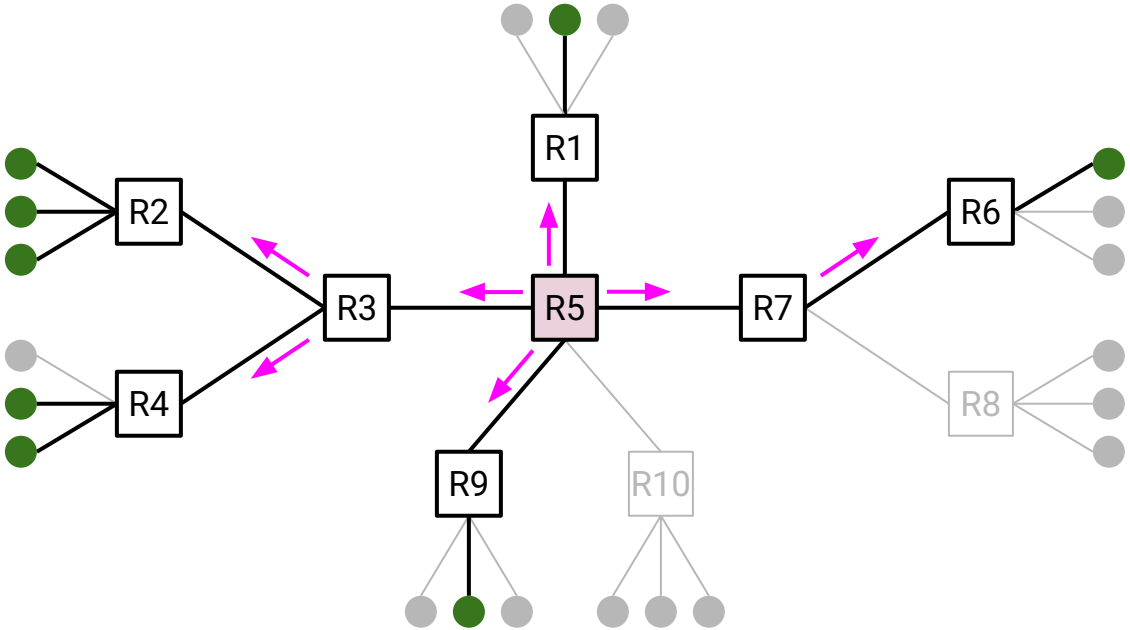
- Rooted at a specific core router.
- Touches every member of that destination group.



CBT: Definition

A **Core-Based Tree (CBT)** for a given destination group is:

- Rooted at a specific core router.
- Touches every member of that destination group.

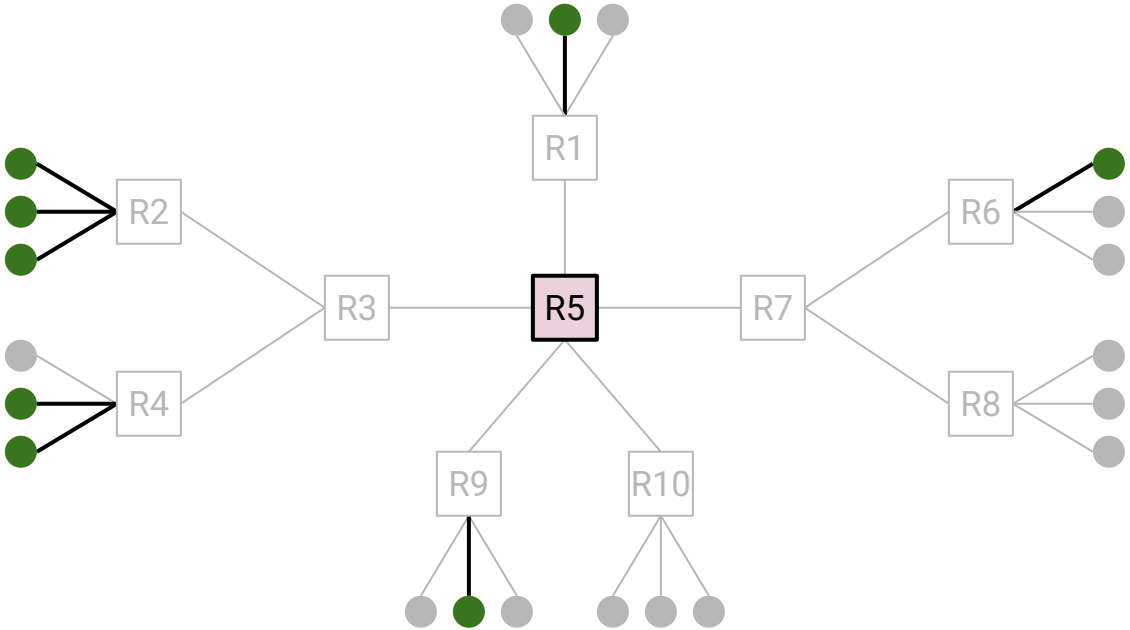


Building a CBT: Choosing Core

First step: Choose a **core router** to be the root of the tree.

- More later on how the core is chosen.

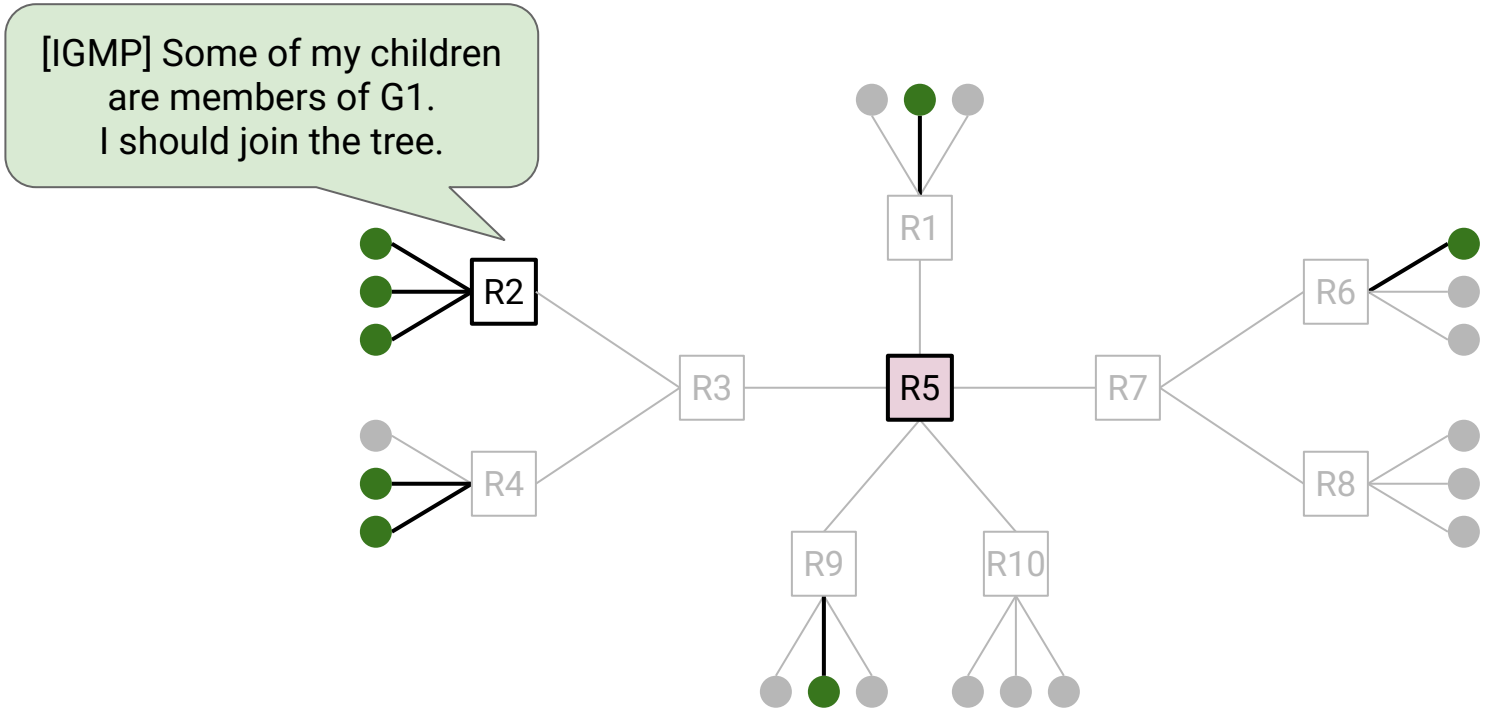
Initially, the core is the only node on the tree.



Building a CBT: Join Messages

If a router needs to join the tree: Unicast a **Join** message to the core.

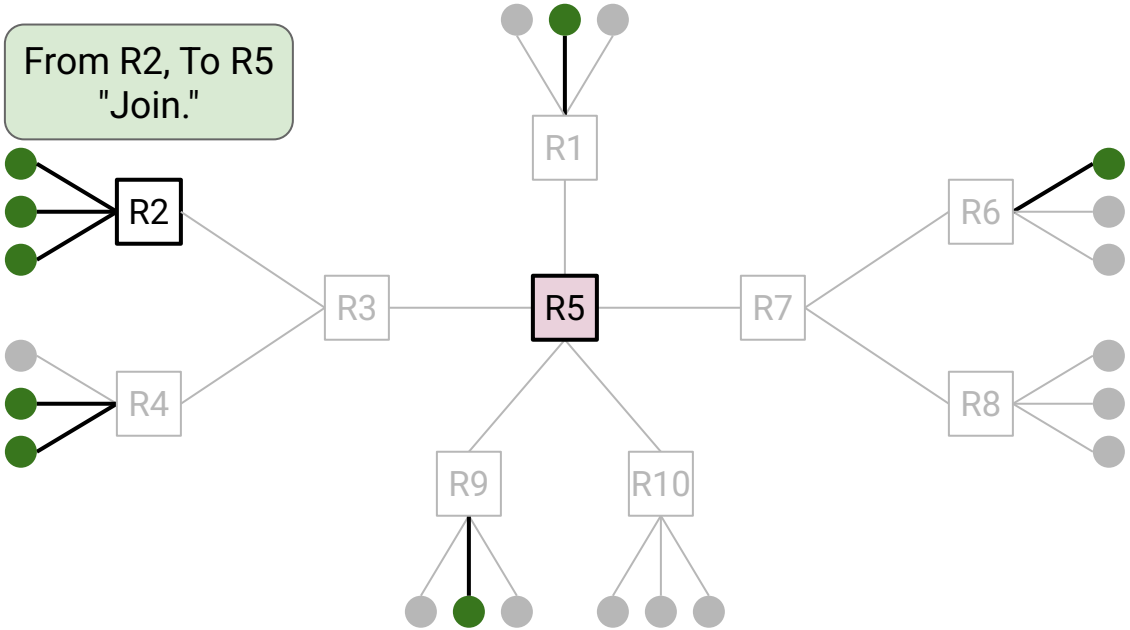
Routers along the path see the message. They join the tree by enabling the incoming link (*the link the message came from*) and the outgoing link (*the link the message is forwarded to*).



Building a CBT: Join Messages

If a router needs to join the tree: Unicast a **Join** message to the core.

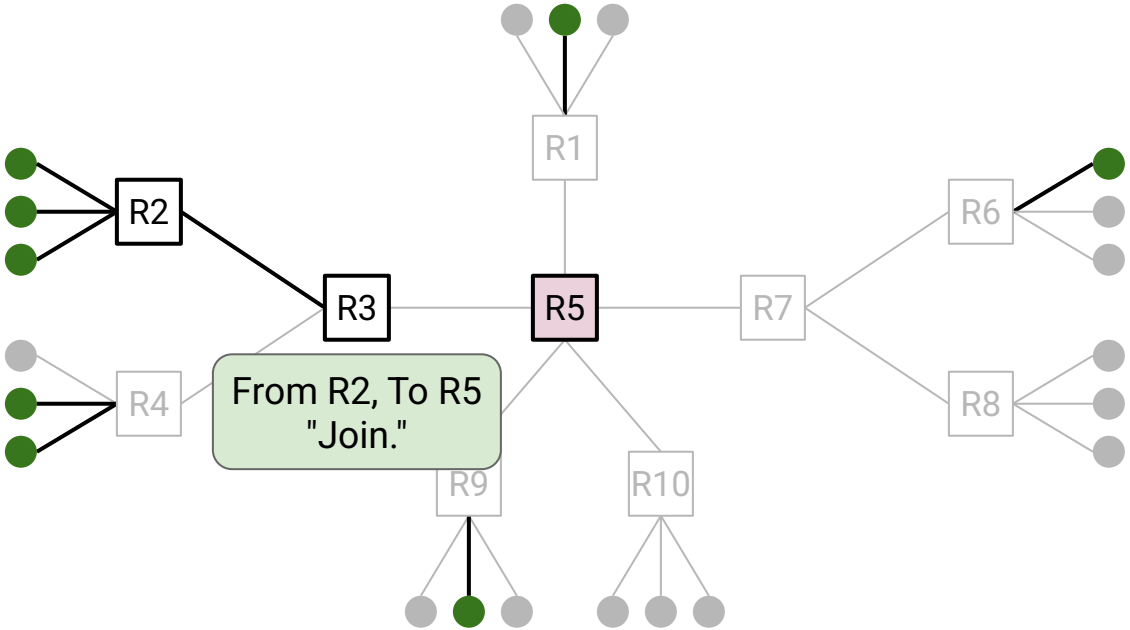
Routers along the path see the message. They join the tree by enabling the incoming link (*the link the message came from*) and the outgoing link (*the link the message is forwarded to*).



Building a CBT: Join Messages

If a router needs to join the tree: Unicast a **Join** message to the core.

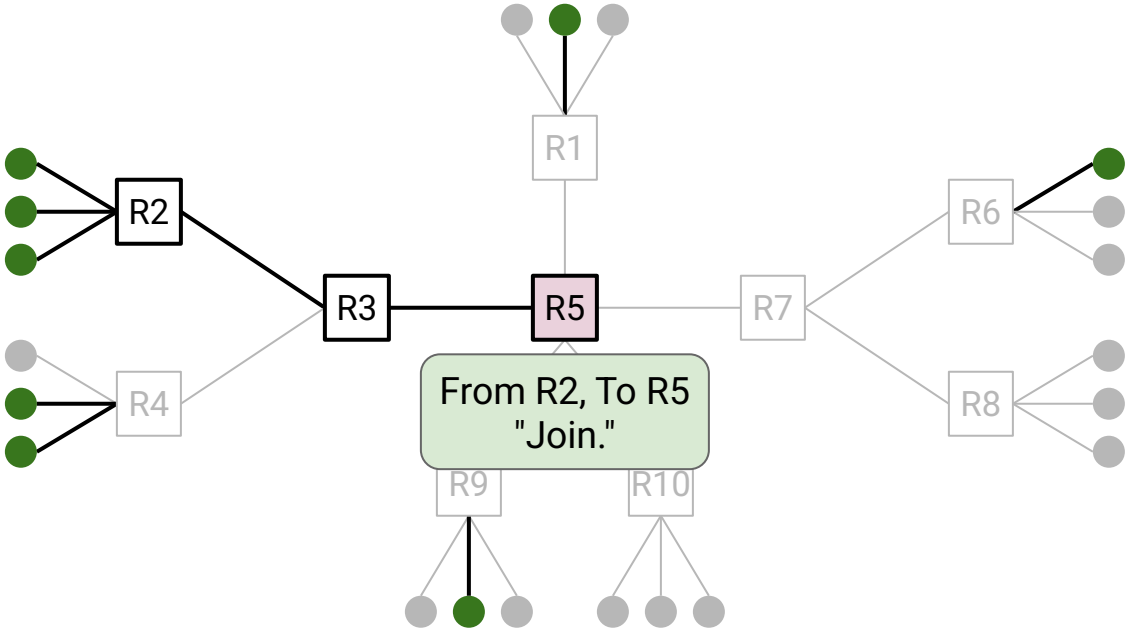
Routers along the path see the message. They join the tree by enabling the incoming link (*the link the message came from*) and the outgoing link (*the link the message is forwarded to*).



Building a CBT: Join Messages

If a router needs to join the tree: Unicast a **Join** message to the core.

Routers along the path see the message. They join the tree by enabling the incoming link (*the link the message came from*) and the outgoing link (*the link the message is forwarded to*).



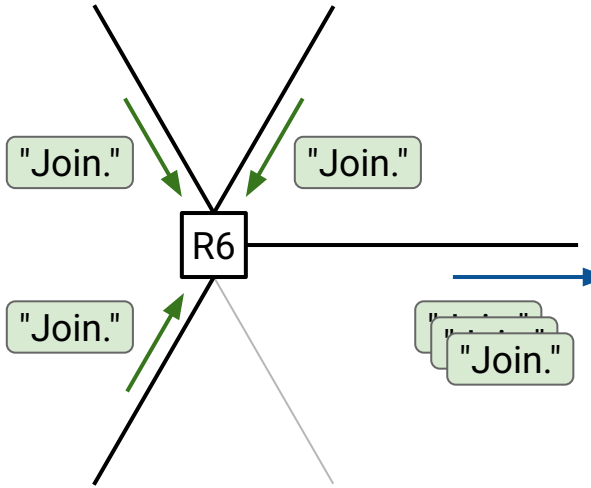
Building a CBT: Join Messages

If a router needs to join the tree: Unicast a **Join** message to the core.

Routers along the path see the message. They join the tree by enabling the incoming link (*the link the message came from*) and the outgoing link (*the link the message is forwarded to*).

When you receive a join message, enable the incoming link.

That link is now part of the tree.



All join messages get forwarded toward the root.

Enable this outgoing link.

It is now part of the tree.

Building a CBT: Quit Messages

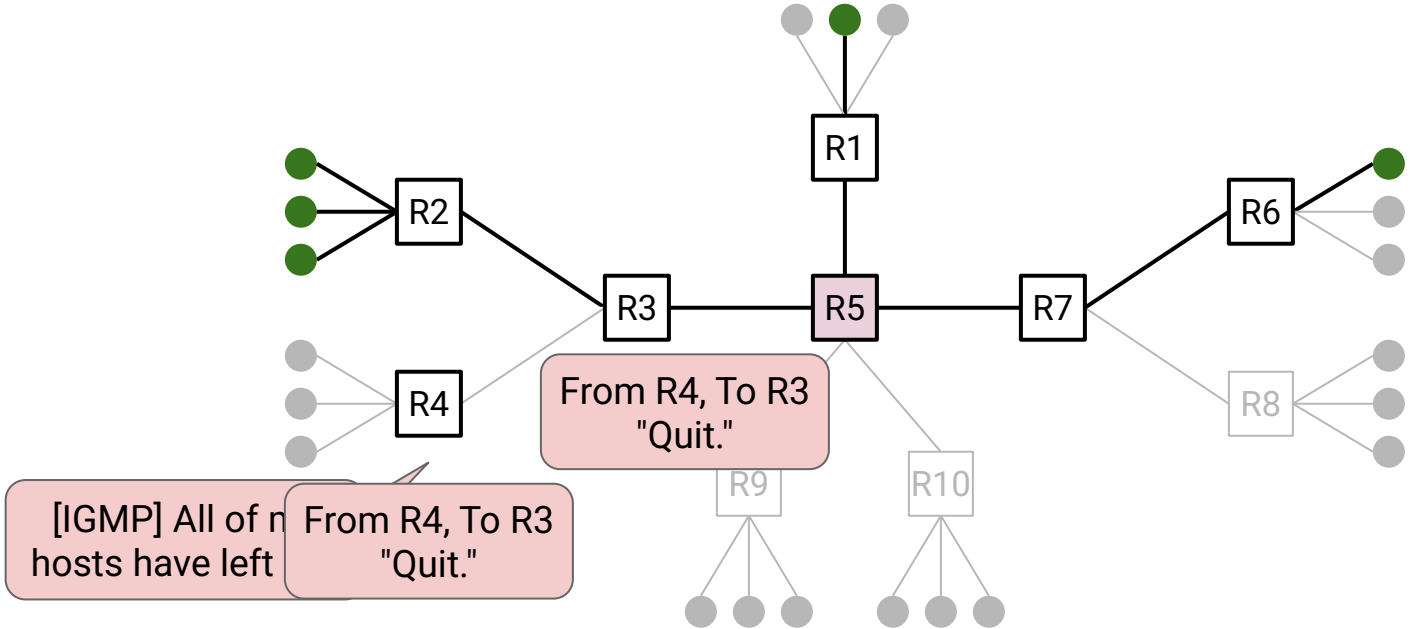
If you are a leaf with no directly-connected members: Send a **Quit** message to your parent. (*Quit message only goes to your direct parent, not all the way to the core.*)

Parent receives Quit message and disables their link to that child.

Building a CBT: Quit Messages

If you are a leaf with no directly-connected members: Send a **Quit** message to your parent. *(Quit message only goes to your direct parent, not all the way to the core.)*

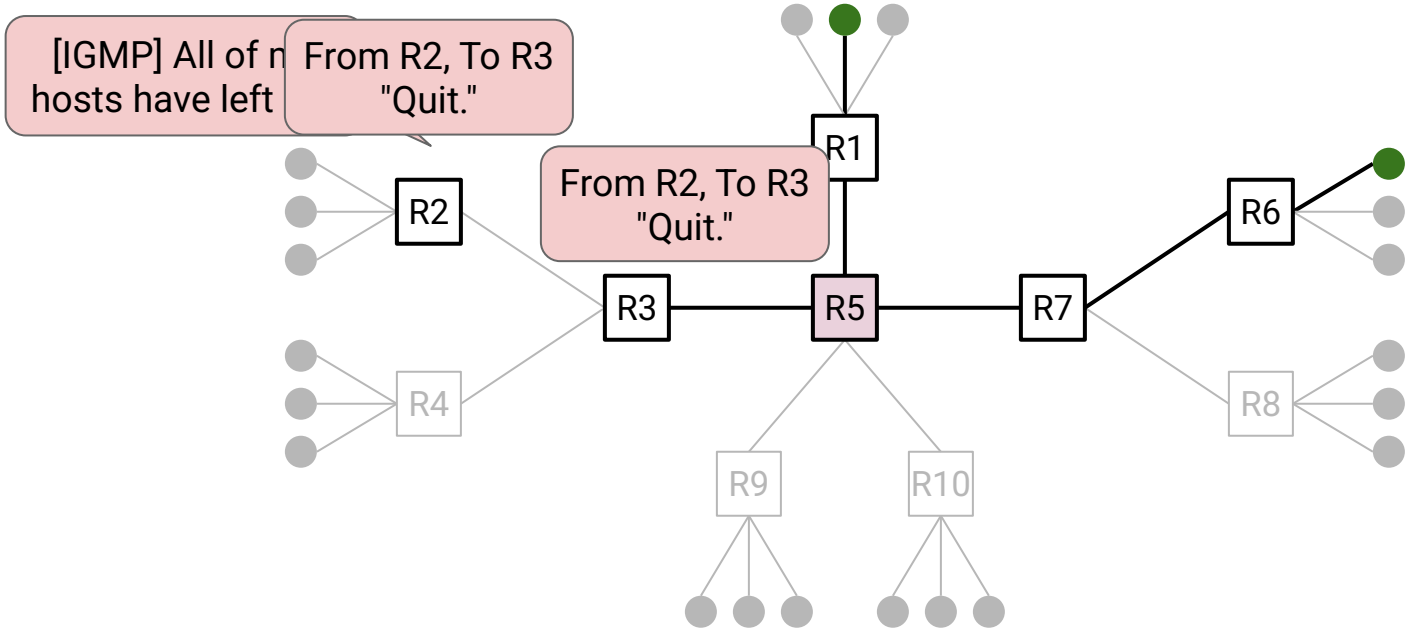
Parent receives Quit message and disables their link to that child.



Building a CBT: Quit Messages

If you are a leaf with no directly-connected members: Send a **Quit** message to your parent. *(Quit message only goes to your direct parent, not all the way to the core.)*

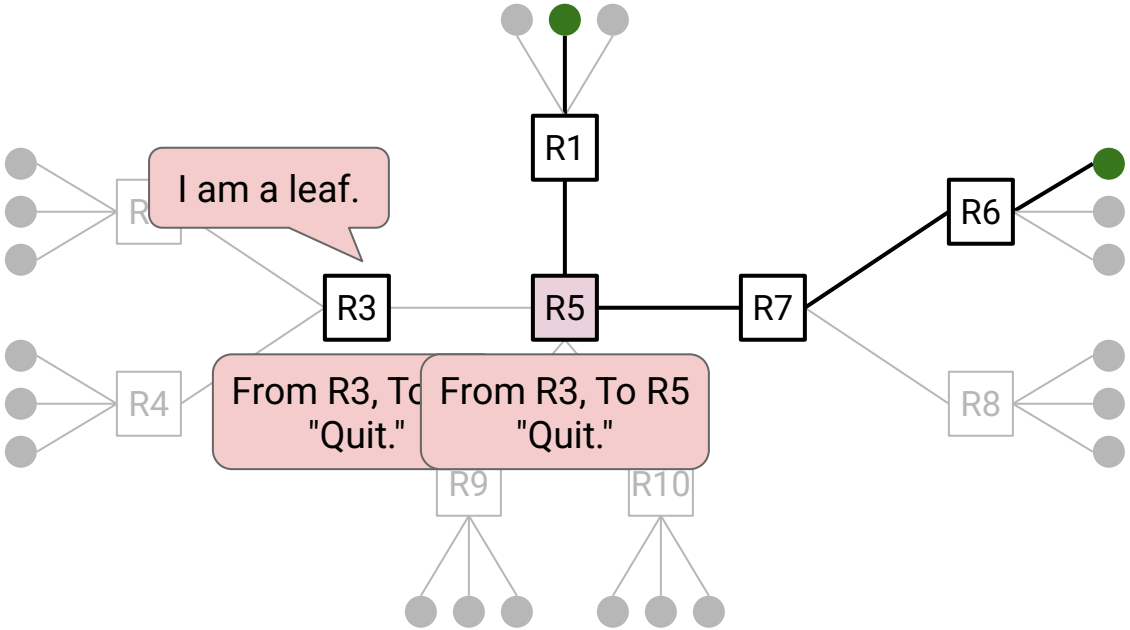
Parent receives Quit message and disables their link to that child.



Building a CBT: Quit Messages

If you are a leaf with no directly-connected members: Send a **Quit** message to your parent. *(Quit message only goes to your direct parent, not all the way to the core.)*

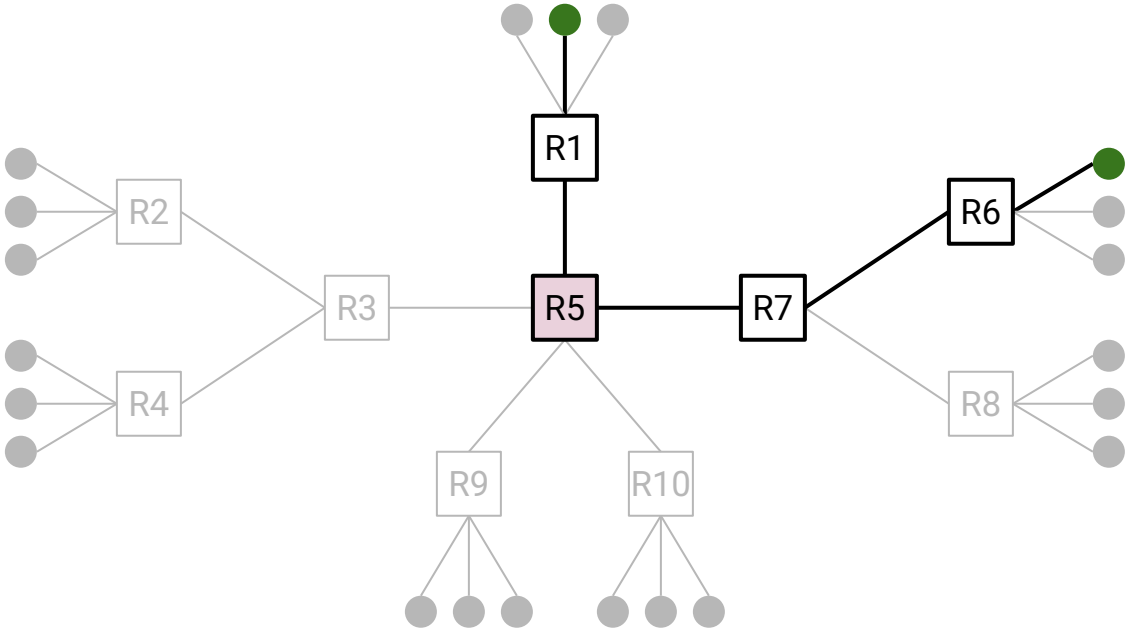
Parent receives Quit message and disables their link to that child.



Building a CBT: Quit Messages

If you are a leaf with no directly-connected members: Send a **Quit** message to your parent. *(Quit message only goes to your direct parent, not all the way to the core.)*

Parent receives Quit message and disables their link to that child.

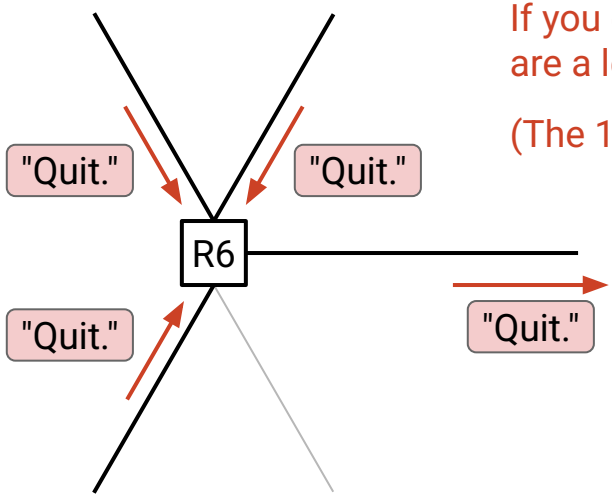


Building a CBT: Quit Messages

If you are a leaf with no directly-connected members: Send a **Quit** message to your parent. *(Quit message only goes to your direct parent, not all the way to the core.)*

Parent receives Quit message and disables their link to that child.

When you receive a Quit message, disable the link you got that message from.



If you only have 1 enabled link, you are a leaf.

(The 1 link is to your parent.)

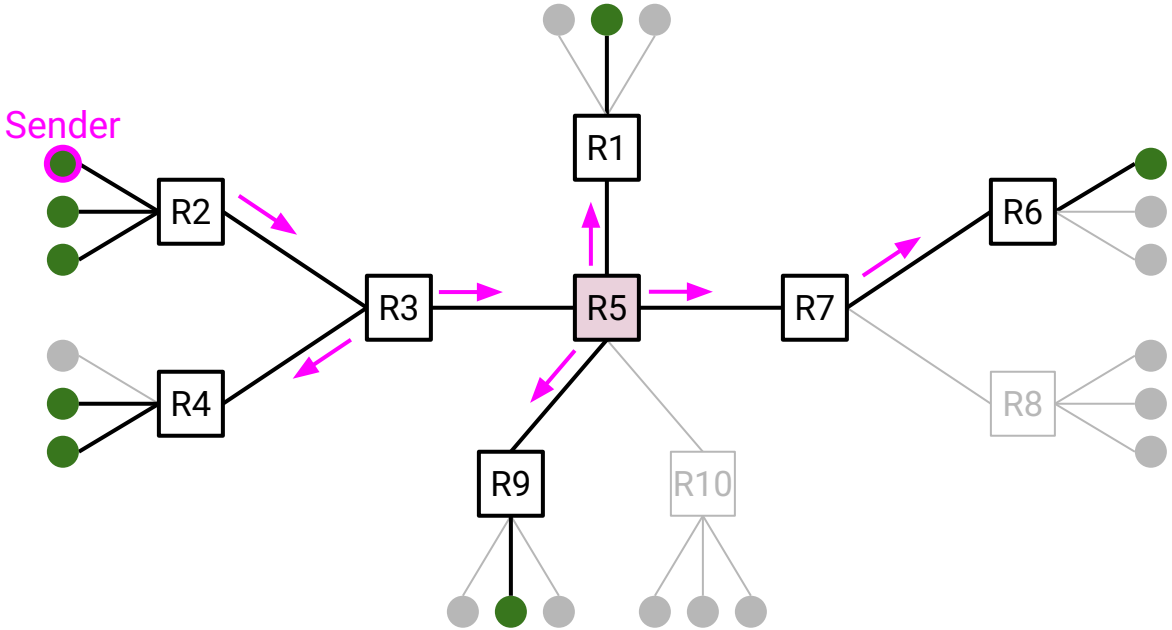
Send a "Quit" message to your parent.

Note: Send "Quit" to your parent, not the root.

Using a CBT: Member Sending Packet

Case 1: You're a group member, and you send a packet to the group.

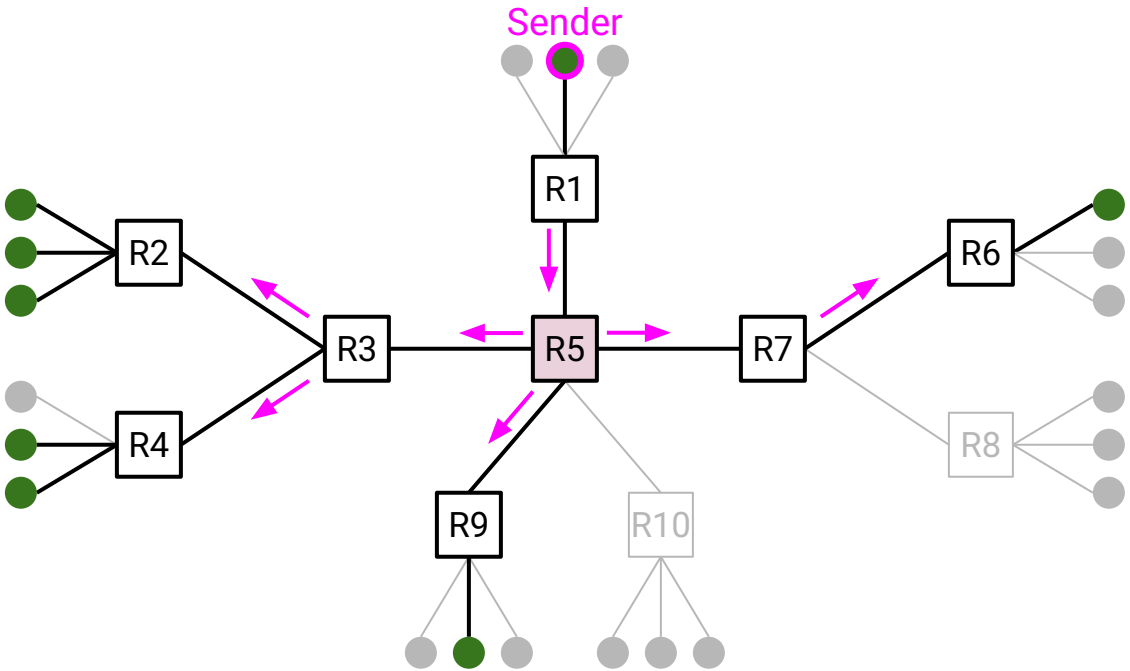
- Broadcast message along the tree to reach every group member.
- Each router floods the packet out of all its enabled tree links.



Using a CBT: Member Sending Packet

Case 1: You're a group member, and you send a packet to the group.

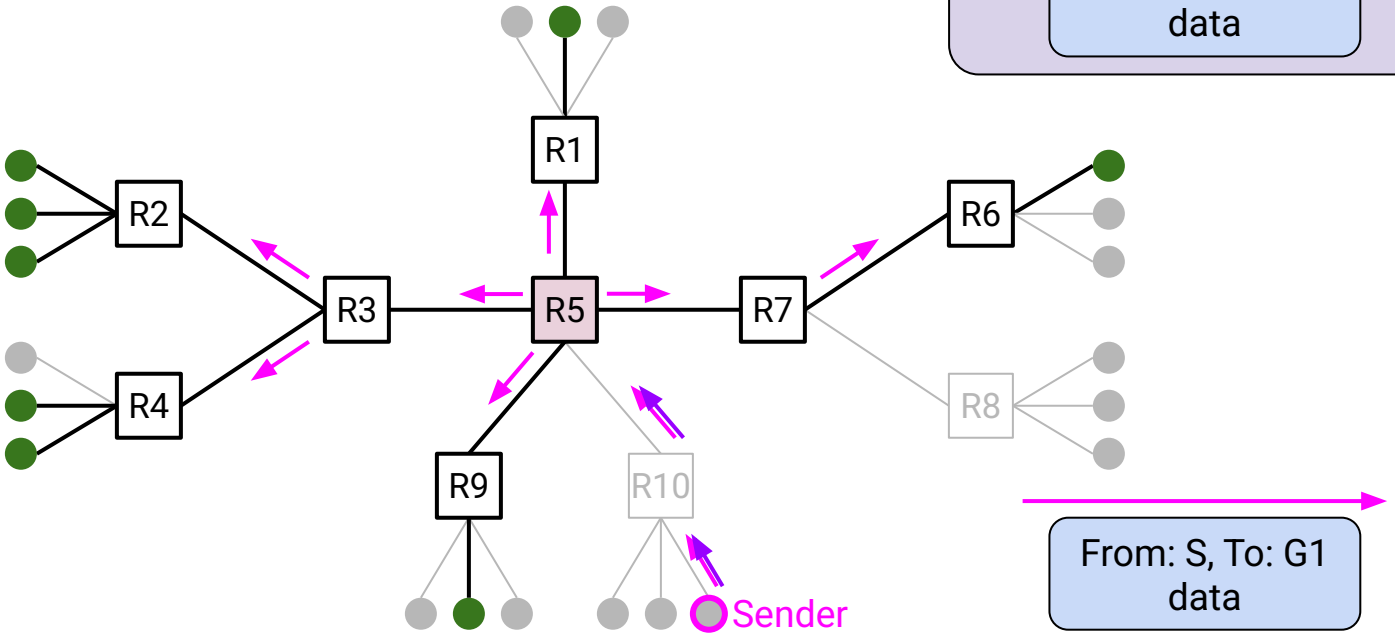
- Broadcast message along the tree to reach every group member.
- Each router floods the packet out of all its enabled tree links.



Using a CBT: Non-Member Sending Packet

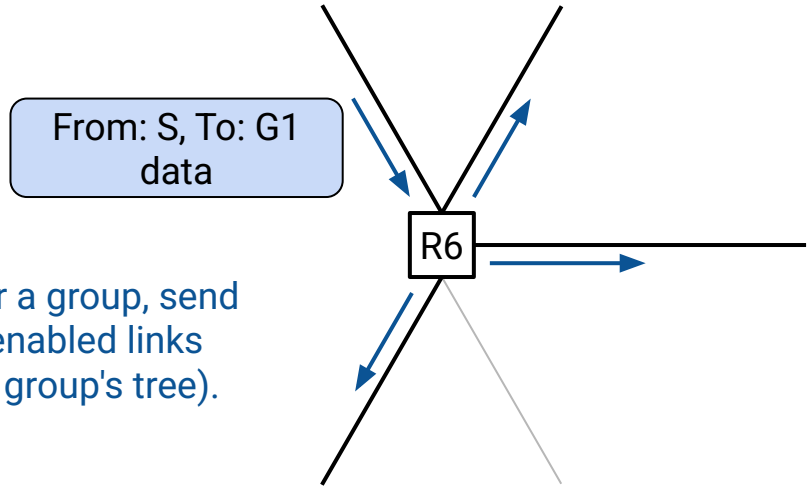
Case 2: You're not a group member, and you send a packet to the group.

- Encapsulate the packet, and unicast it to the core.
- Then, the core broadcasts the packet to everybody on the tree.



Using a CBT: Forwarding Packets

In both cases, when you receive a multicast packet, flood the packet out of all enabled tree links.

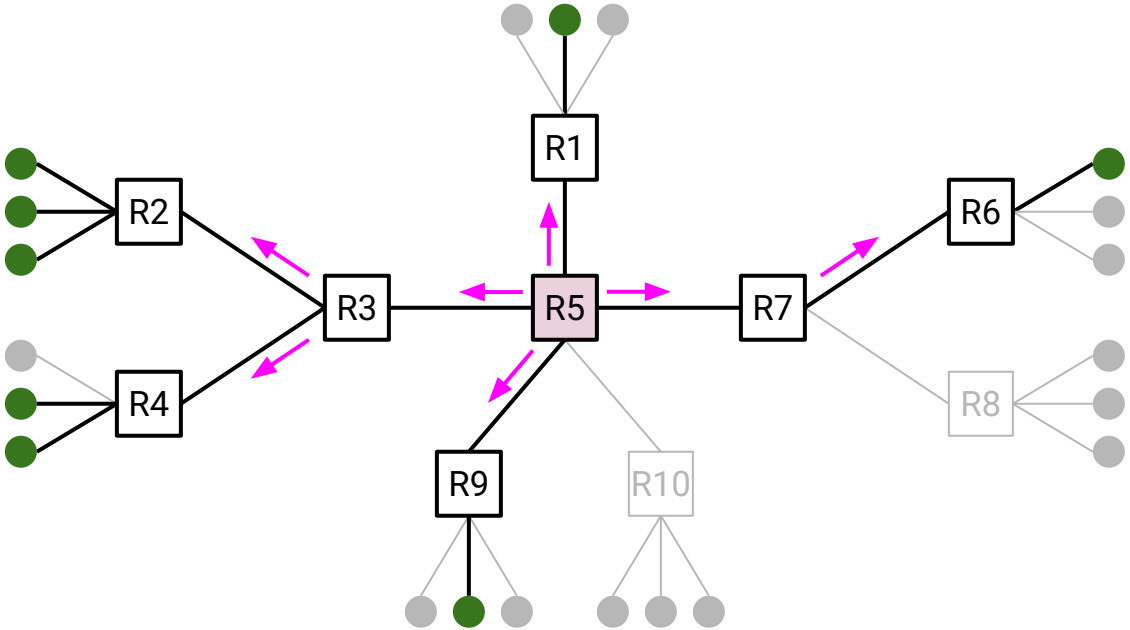


When you receive data for a group, send that data out of all other enabled links (i.e. all other links on that group's tree).

One Tree Per Destination Group

Recall: A CBT for a group must touch every member of that group.

- We build one tree per group.
- Routers must remember the enabled links for each group's tree.



Core-Based Trees: Pros and Cons

Lecture 22, CS 168, Spring 2026

Introducing Multicast
IP Multicast
- IP Multicast Service Model
- IGMP: Directly-Connected Hosts

DVMRP
- Naive Approach: Flooding
- RPB (Broadcasting): Trees
- RPM (Multicasting): Pruning
- Pros and Cons

Core-Based Trees
- Protocol
- **Pros and Cons**

IP Multicast Challenges
Overlay-Based Multicast
- Implementation

Pros of CBT

Main idea: CBT gives better scaling than DVMRP, at the cost of worse efficiency.

Pro: CBT scales better than DVMRP.

- DVMRP needs one tree per sender, per destination group.
- DVMRP periodically clears prune state, causing packets to be broadcast to everyone (including non-members).
- CBT needs one tree per destination group.
- CBT never broadcasts packets to everyone. The tree is constantly updated to only touch current group members.

CBT Pro: Less State

DVMRP requires one tree per sender, per destination group.

CBT requires one tree per destination group. Scales much better!

Destination Group

Sender	G1	G2	G3	G4	G5
A	Tree	Tree	Tree	Tree	Tree
B	Tree	Tree	Tree	Tree	Tree
C	Tree	Tree	Tree	Tree	Tree
D	Tree	Tree	Tree	Tree	Tree
E	Tree	Tree	Tree	Tree	Tree
F	Tree	Tree	Tree	Tree	Tree
G	Tree	Tree	Tree	Tree	Tree

DVMRP: One tree per sender, per destination group.

Each tree shows the shortest path from that sender to all hosts in that destination group.

Destination Group

Sender	G1	G2	G3	G4	G5
A	Tree	Tree	Tree	Tree	Tree
B					
C					
D					
E					
F					
G					

CBT: One tree per destination group.

Each tree connects all hosts in that destination group.

Cons of CBT

Main idea: CBT gives better scaling than DVMRP, at the cost of worse efficiency.

Con: CBT is less efficient than DVMRP.

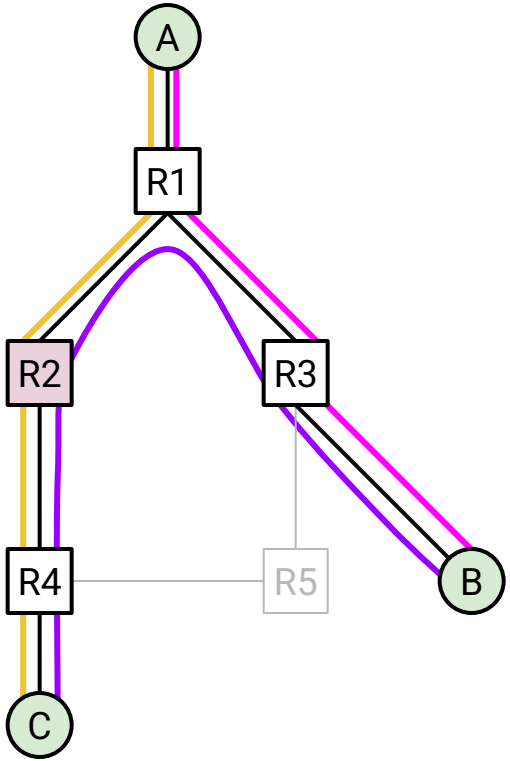
- DVMRP gives us the shortest paths from the sender to all group members.
- DVMRP doesn't require a specific choice of core.
- CBTs don't involve the sender at all. There is no guarantee that packets are taking the shortest path from the sender.
- CBT requires choosing a good core. A bad choice of core can lead to suboptimal paths.

CBT Drawback: Choice of Core

Observations:

- CBT causes packets to take suboptimal paths.
- The choice of core has a big impact on CBT performance.
- Choosing a good core is hard. Core is often manually chosen by operator.

Hosts	Cost	Shortest path?
A to B	3	yes
A to C	4	yes
B to C	5	no



A, B, C are members of G1.

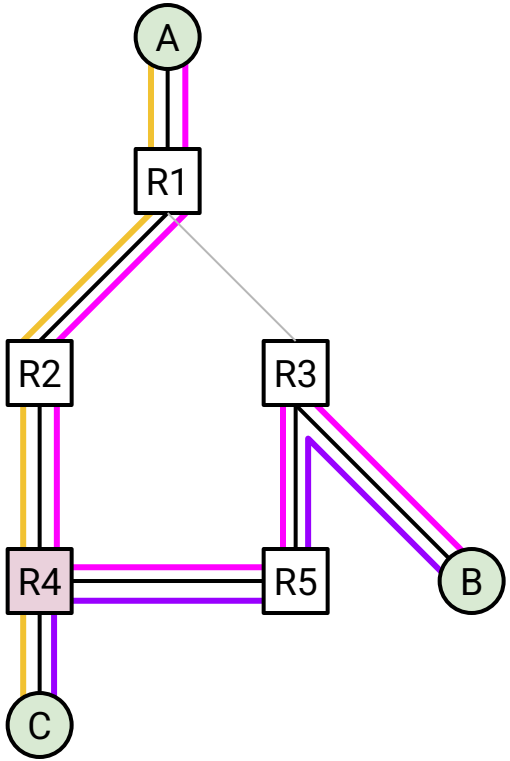
R2 is the core.

CBT Drawback: Choice of Core

Observations:

- CBT causes packets to take suboptimal paths.
- The choice of core has a big impact on CBT performance.
- Choosing a good core is hard. Core is often manually chosen by operator.

Hosts	Cost	Shortest path?
A to B	6	no
A to C	4	yes
B to C	4	yes



A, B, C are members of G1.

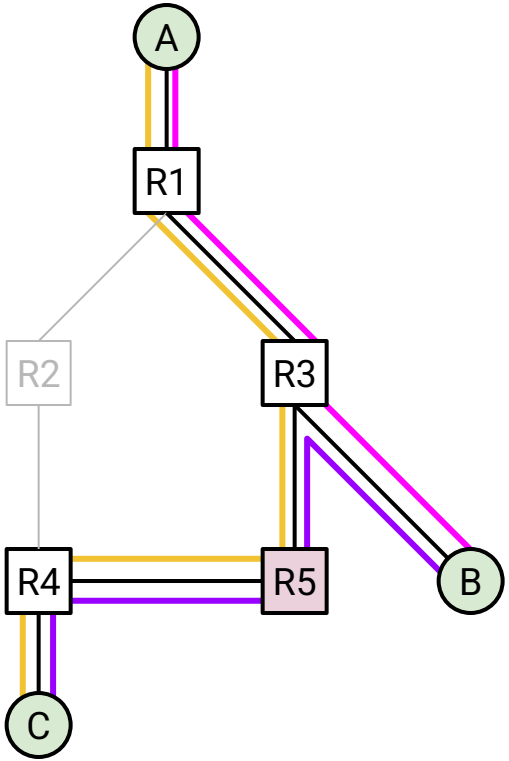
R4 is the core.

CBT Drawback: Choice of Core

Observations:

- CBT causes packets to take suboptimal paths.
- The choice of core has a big impact on CBT performance.
- Choosing a good core is hard. Core is often manually chosen by operator.

Hosts	Cost	Shortest path?
A to B	3	yes
A to C	5	no
B to C	4	yes



A, B, C are members of G1.

R5 is the core.

Other Benefits/Drawbacks of CBT

Main idea: CBT gives better scaling than DVMRP, at the cost of worse efficiency.

Other comparisons between the two protocols:

- DVMRP con: The unicast and multicast protocols are tightly-coupled.
  - DVMRP is an extension to distance-vector. If we changed the unicast protocol (e.g. to link-state), we'd have to change the multicast protocol too.
  - CBT works with any unicast protocol.
- CBT creates a single point of failure at the core.
  - Can be solved by using multiple cores, but introduces more complexity. Won't discuss further.

DVMRP and CBT in Practice

Is DVMRP or CBT better? It depends.

- DVMRP is better if you have one source sending to a large group.
  - You get the shortest paths to all group members.
    The group is large, so you get lots of bandwidth savings.
  - If the group is large, the occasional broadcasting (after pruning is cleared) isn't a big problem.
- CBT is better if you're sending to a small group.
  - CBT avoids broadcasting to non-members.

Both are used today. PIM (Protocol-Independent Multicast) has two variants:

- PIM-DM (Dense Mode): Uses DVMRP for large groups.
- PIM-SM (Sparse Mode): Uses CBT for small groups.

IP Multicast Challenges

Lecture 22, CS 168, Spring 2026

Introducing Multicast
IP Multicast
- IP Multicast Service Model
- IGMP: Directly-Connected Hosts

DVMRP
- Naive Approach: Flooding
- RPB (Broadcasting): Trees
- RPM (Multicasting): Pruning
- Pros and Cons

Core-Based Trees
- Protocol
- Pros and Cons

IP Multicast Challenges

Overlay-Based Multicast
- Implementation

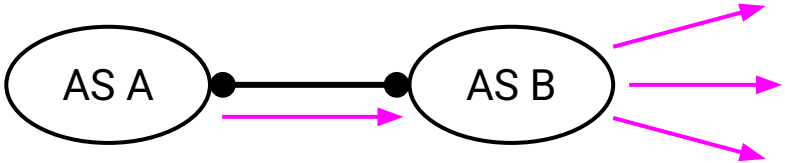
IP Multicast Challenges (1/5): Inter-Domain Routing

Our protocols don't scale to inter-domain routing.

- DVMRP: Pruning is periodically cleared. Packets forwarded to all of Internet?
- CBT: Where to put the core? ASes don't want to depend on a core in another AS.
- In practice, very little adoption of inter-domain IP multicast.

IP Multicast Challenges (2/5): Charging

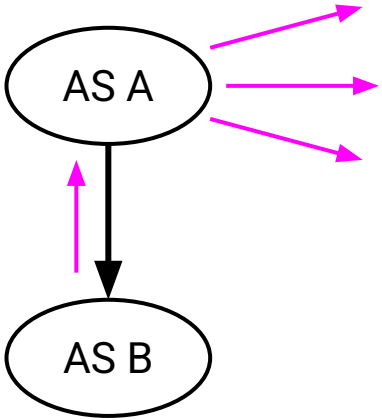
It's hard to design a business model for IP multicast.



A and B are peers, which means they can exchange equal amounts of traffic.

But if A sends a multicast packet to B, then B uses more bandwidth than A.

Should A pay B to forward this packet?



A provides service to B.

If B sends a multicast packet, A has to use more bandwidth (compared to unicast packet).

Should A charge more for the multicast packet?

IP Multicast Challenges (3/5): Congestion Control

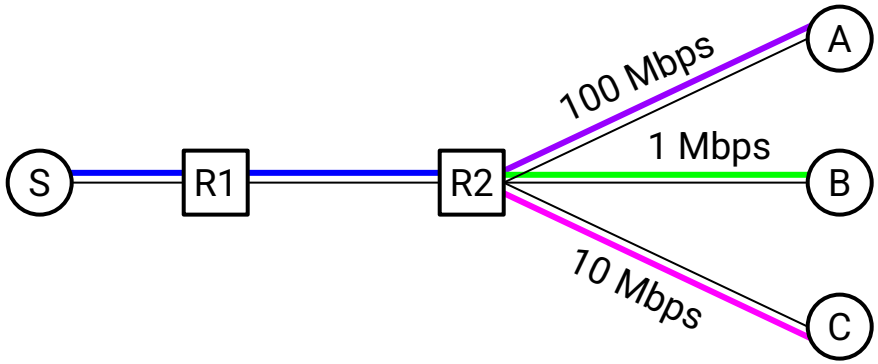
What rate should the sender choose? It's not so obvious.

- 1 Mbps: Too low. Wastes unused capacity on some links.
- 100 Mbps: Too high. Overloads some of the links.

Possible solution: Have different groups depending on performance.

- Example: One group gets high-quality video, one group gets low-quality video.
- Users try joining different groups to see which is best for them.

S sends a multicast packet to a destination group with A, B, C as members.



IP Multicast Challenges (4/5): Reliability

IP multicast is best-effort. Adding reliability introduces more complexity.

- What if the packet reaches some (but not all) group members?
- What if the packet is dropped at the first hop? Nobody receives the packet.

Send an ack when you receive the packet?

- If everyone gets packet, sender gets overwhelmed with tons of acks.

Send a nack if you don't get a packet (e.g. if timer expires)?

- If nobody gets packet, sender gets overwhelmed with tons of nacks.

How do you recover from loss?

- Multicast packet to everyone again? Wastes bandwidth if most received it.
- Unicast packet to people who didn't get it? Wastes bandwidth sending duplicate packets across links.

In practice, we often implement reliability by encoding redundancy into the data.

IP Multicast Challenges (5/5): Security

IP multicast has no access control.

- Anybody can join any group.
- Anybody can send messages to any group.

Possible attack: Denial-of-service.

- Send tons of packets to a group, overwhelming the members of that group.
- More effective than the unicast alternative, where you'd have to send tons of packets to each member separately.

Possible solution: Encrypt messages with a secret key.

- Give a secret key to everyone in the group. How to distribute the key securely?
- If someone leaves the group, they can still use the key to read messages.
- Switch to a new key when someone leaves? How to distribute the new key?

IP Multicast in Practice

Today, IP multicast is mostly used within a single network.

- Inter-domain IP multicast is not widely adopted.
- This means you can't send multicast packets on the Internet and expect them to "automatically" work.

So, how do modern applications support inter-domain group communication?

- Applications can write custom solutions based on the unicast model.
- Example: Use a central relay server. Sender unicasts packet to server, and server unicasts packet to all group members.
- Small applications could just use the naive unicast solution (send separate packets to each group member).
- Another solution: Overlay multicast. (Coming up next!)

Overlay-Based Multicast

Lecture 22, CS 168, Spring 2026

Introducing Multicast
IP Multicast
- IP Multicast Service Model
- IGMP: Directly-Connected Hosts

DVMRP
- Naive Approach: Flooding
- RPB (Broadcasting): Trees
- RPM (Multicasting): Pruning
- Pros and Cons

Core-Based Trees
- Protocol
- Pros and Cons

IP Multicast Challenges

Overlay-Based Multicast
- Implementation

Brief History of Overlay Multicast

1990s and 2000s: Much work on IP multicast.

Early 2000s: Deployment of IP multicast was slow.

- The problems we discussed made deployment hard.

Many startups emerged, seeking alternate solutions.

- FastForward Networks (Berkeley)
- ProxyNet (Berkeley)
- Sightpath (MIT)
- Akamai (MIT)

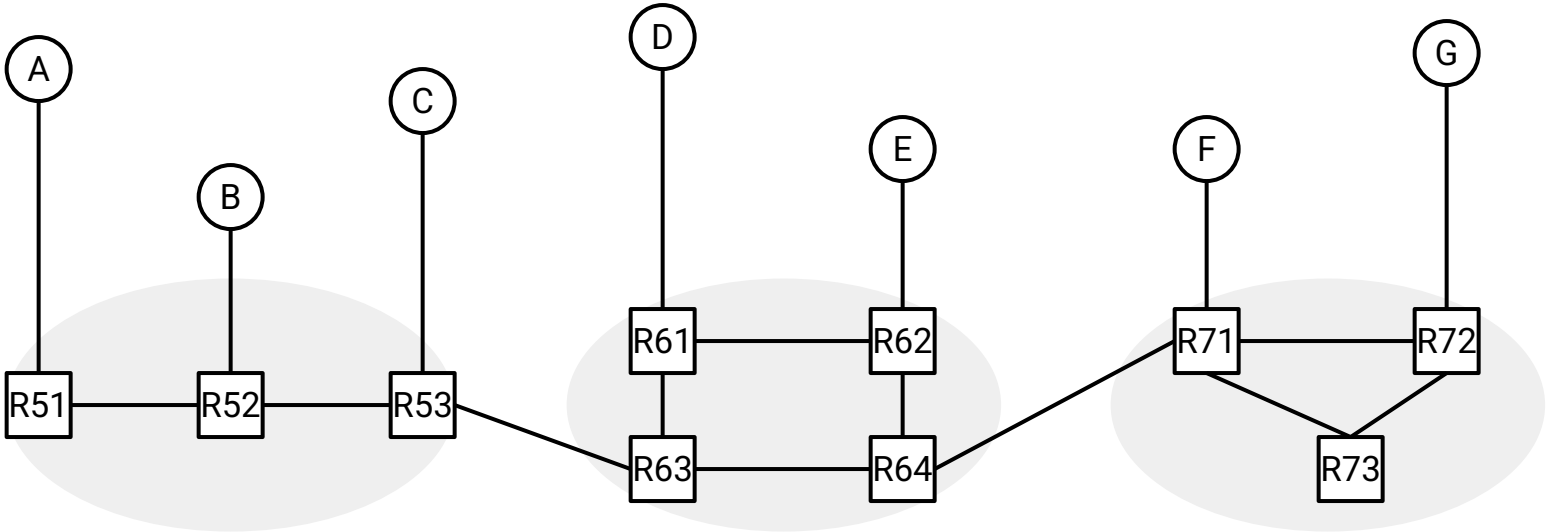
Common theme in these startups: Overlay-based multicast.

- This approach has good enough performance, and is simple enough to deploy, that it is commonly used today.

Overlay Multicast: Key Idea

Problem: No inter-domain support for multicast.

- Example: A and G cannot be part of a group.

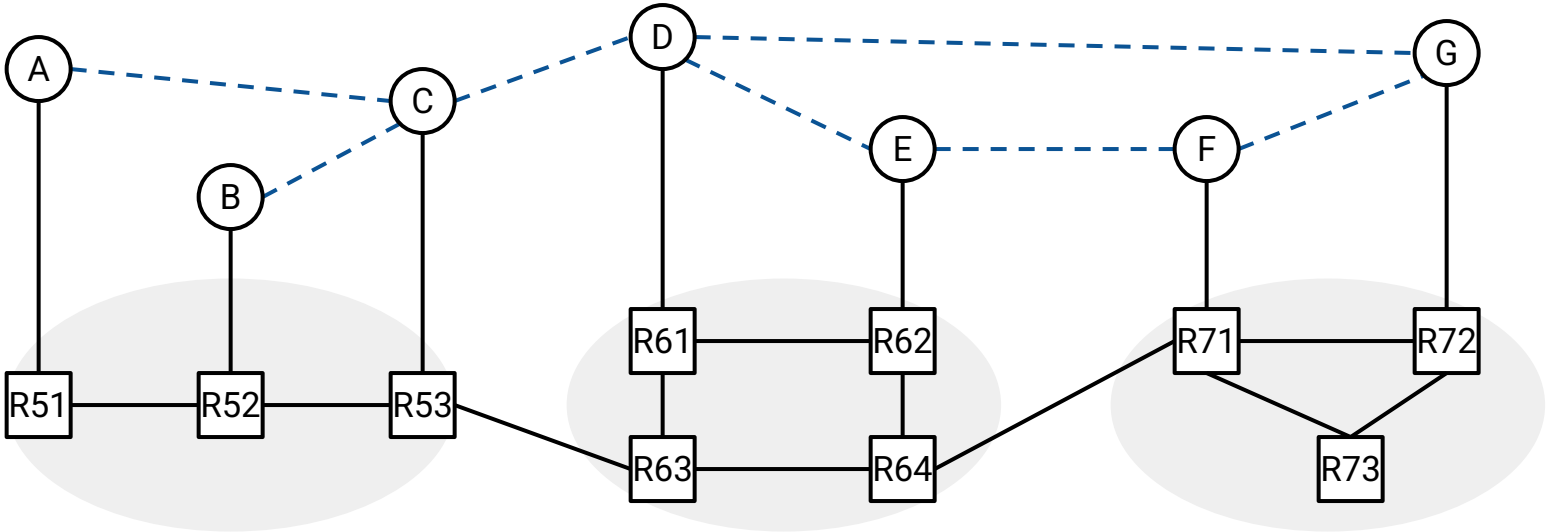


Overlay Multicast: Key Idea

Key idea: Build a **virtual network topology** to connect the hosts together.

- The virtual links are a fiction. Packets still have to be sent along physical links.
- But the virtual links give us the *illusion* that the hosts are in a single network.

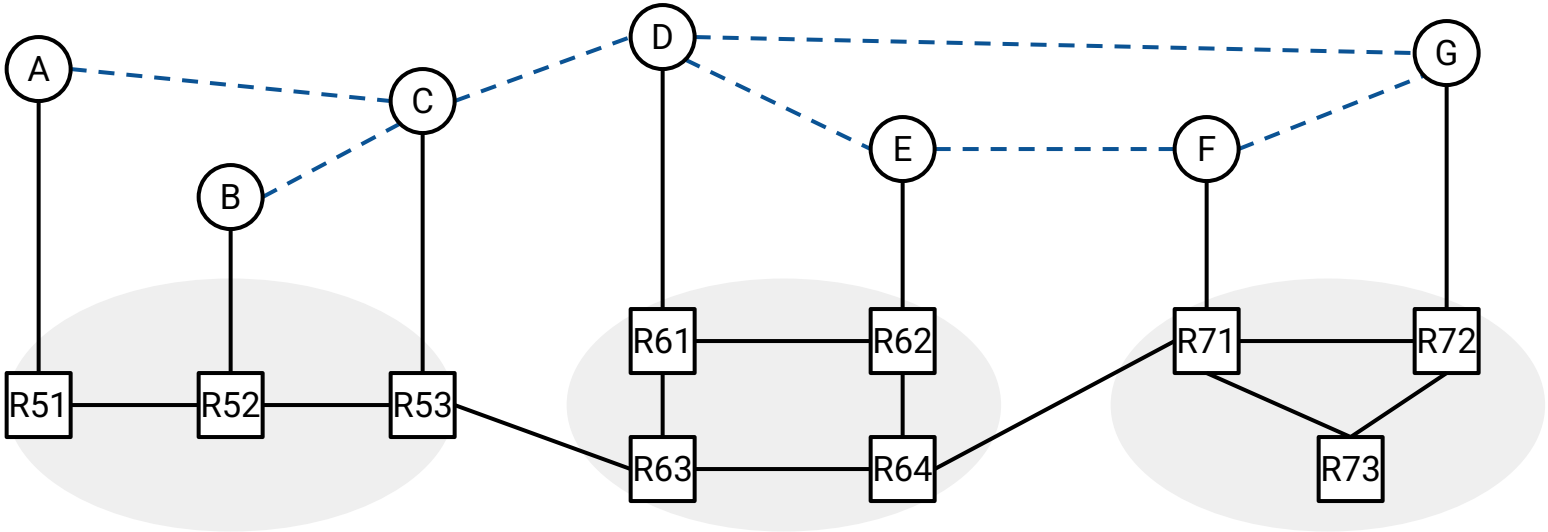
The hosts can run a multicast routing protocol (CBT, DVMRP) to learn how to send multicast packets.



Overlay Multicast: Key Idea

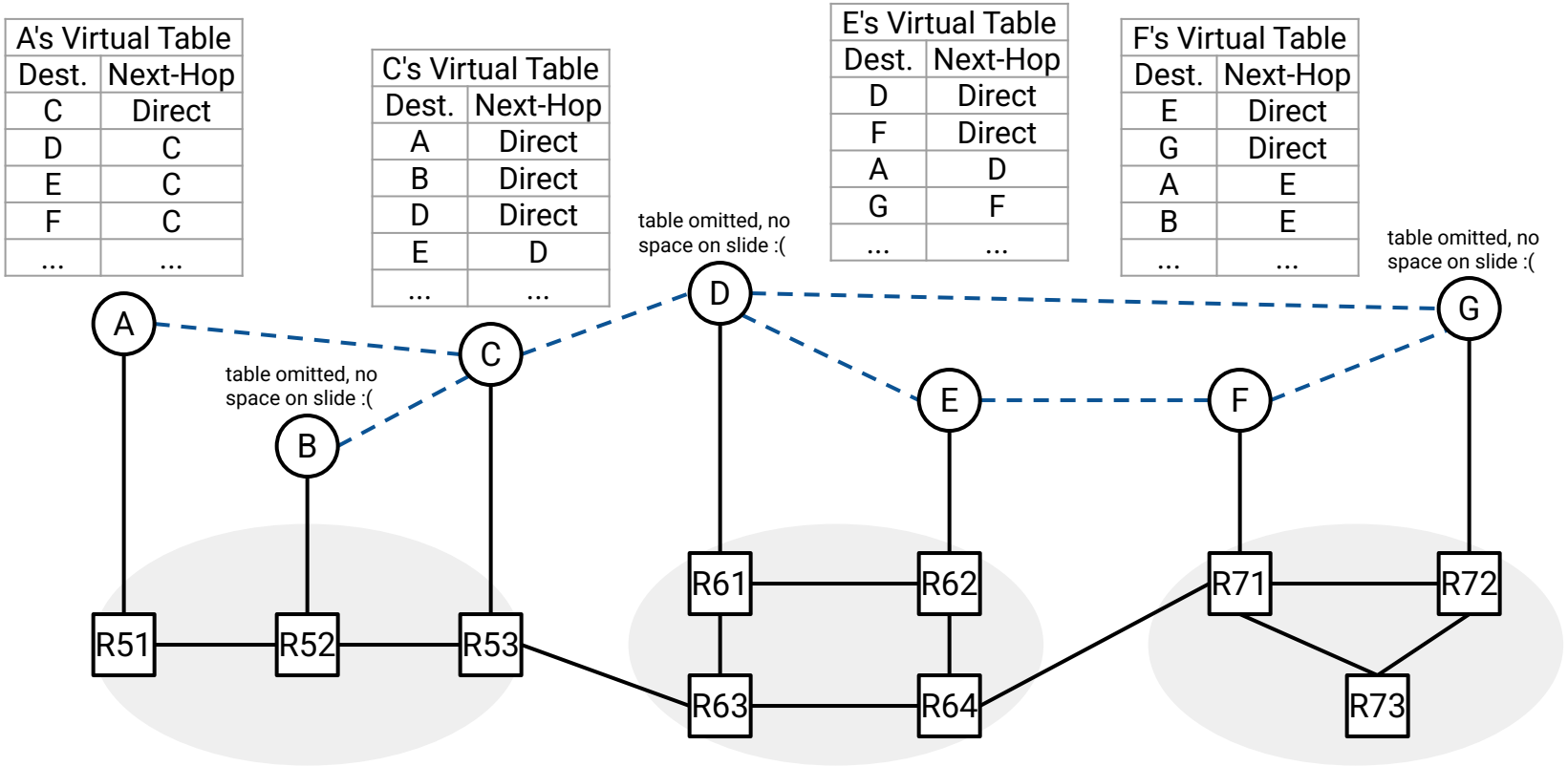
Overlay multicast is implemented in **hosts, not routers**.

- Routers just forward unicast packets, as usual.
- **Virtual routers** in the hosts run routing protocols and populate multicast tables.
- Hosts are now forwarding packets between each other!



Overlay Multicast: Forwarding Packets

Hosts use **virtual forwarding tables** to route packets through the overlay network.



Overlay Multicast: Forwarding Packets

Routers have forwarding tables for hosts, just like usual.

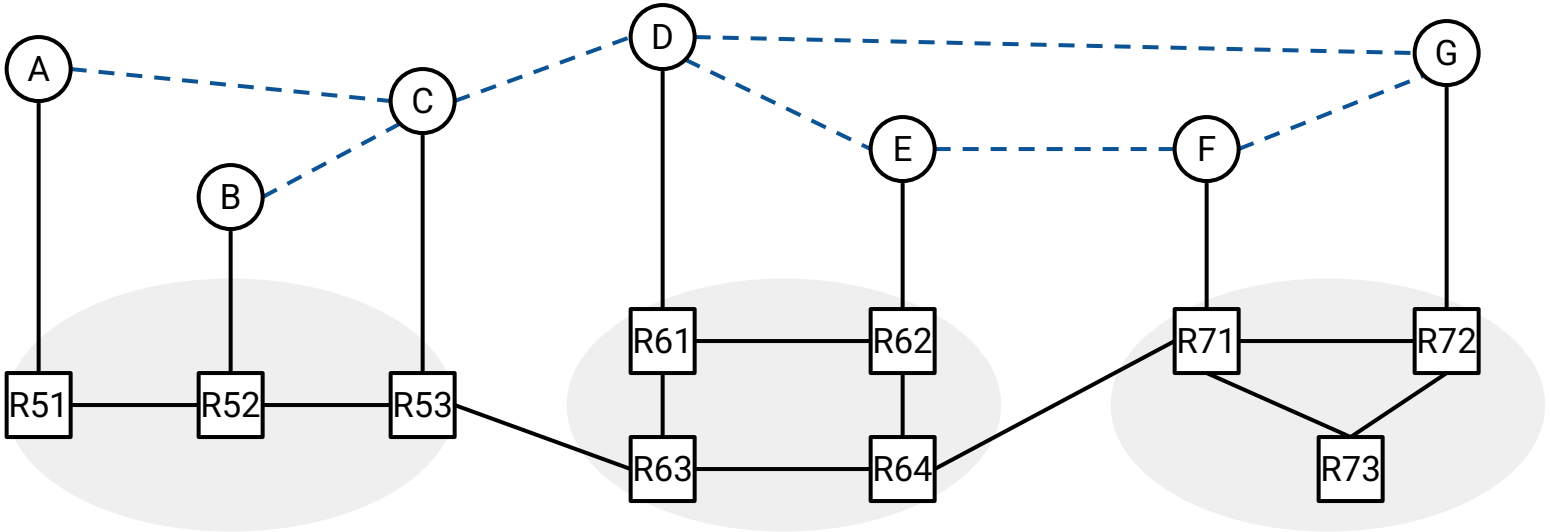
R51's Table	
Dest.	Next-Hop
A	Direct
B	R52
C	R52
D	R52
...	...

R53's Table	
Dest.	Next-Hop
A	R52
B	R52
C	Direct
D	R63
...	...

R61's Table	
Dest.	Next-Hop
B	R63
C	R63
D	Direct
E	R62
...	...

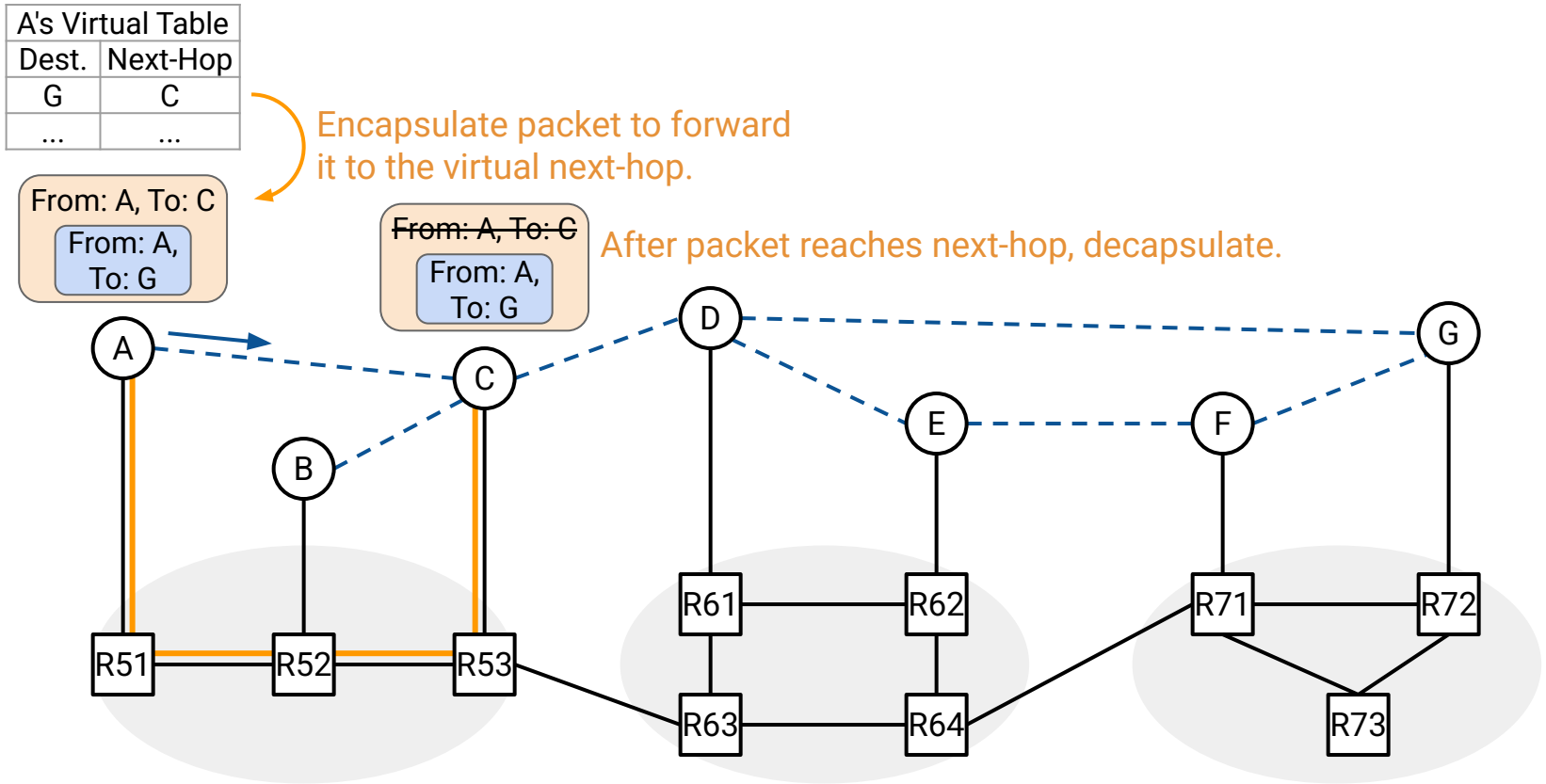
R73's Table	
Dest.	Next-Hop
D	R71
E	R71
F	R72
G	R72
...	...

Tables for the other routers omitted.



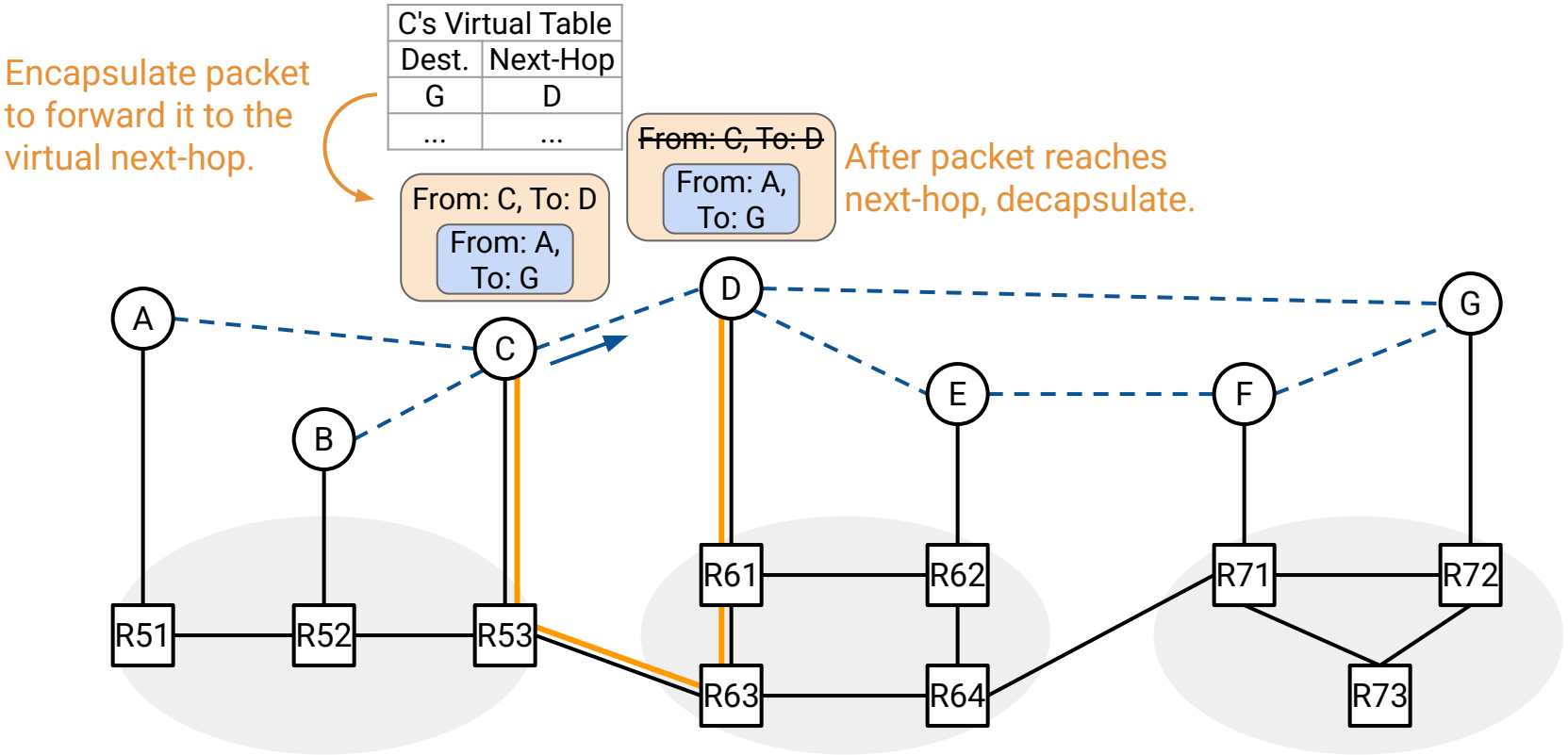
Overlay Multicast: Encapsulation

Virtual routers use **encapsulation** to send packets to the next-hop virtual routers.



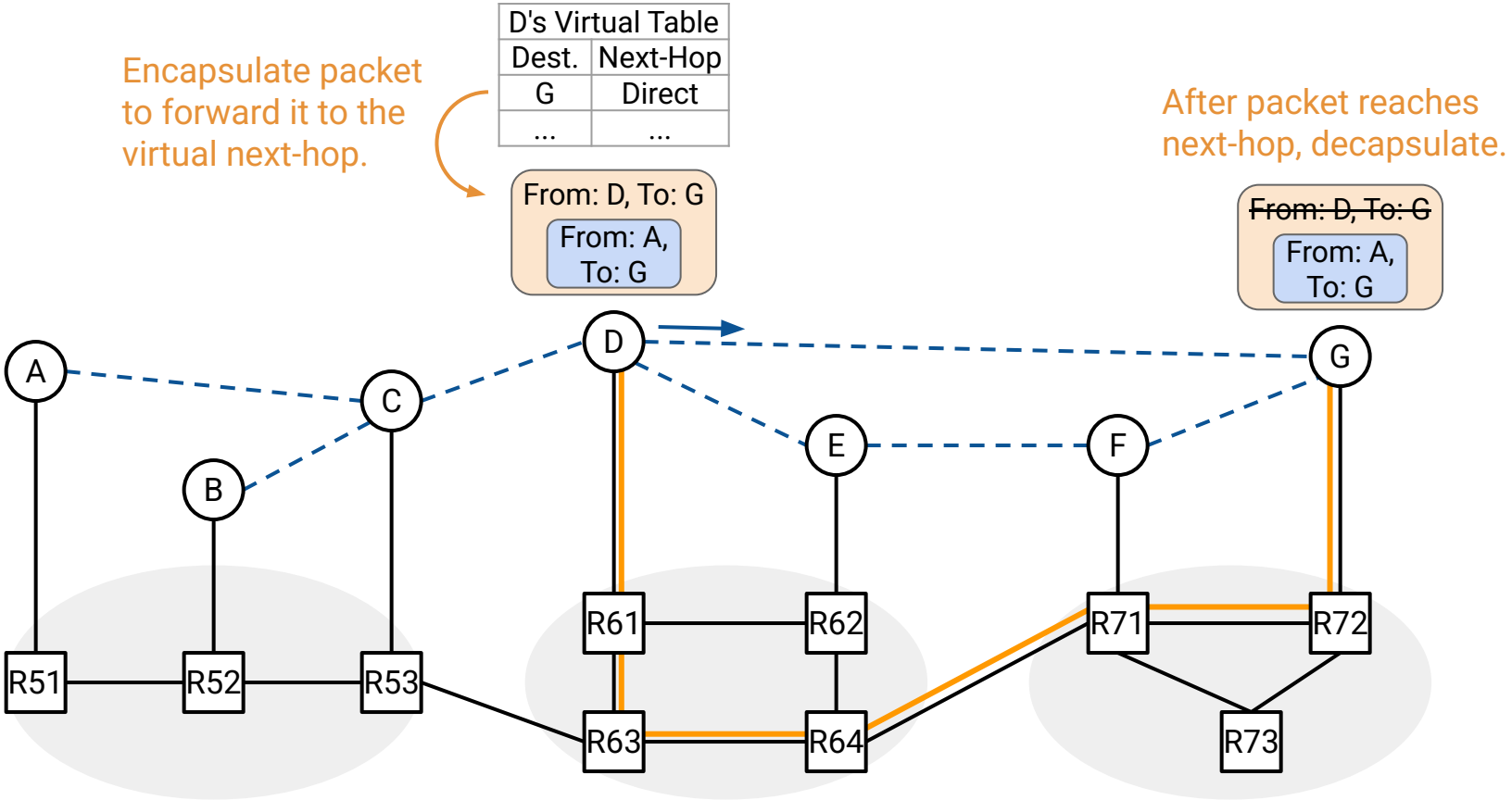
Overlay Multicast: Encapsulation

Virtual routers use **encapsulation** to send packets to the next-hop virtual routers.



Overlay Multicast: Encapsulation

Virtual routers use **encapsulation** to send packets to the next-hop virtual routers.



Implementing Overlay Multicast

Hosts act as **virtual routers**: Run routing protocols, forwards multicast packets.

- Hosts could be end hosts, e.g. your personal laptop.
- Hosts could be machines deployed by some company specifically for multicast. These are still L7 hosts, not L3 routers – their main job is just overlay multicast.

The overlay idea can also be used for other applications besides multicast.

- Peer-to-peer networks: Users share files, with no central server.

Many overlay networks can exist at the same time.

- A host could run multiple virtual routers, and participate in multiple overlay topologies.

Pros of Overlay Multicast

Pro: Overlay multicast is easy to deploy and accommodates diversity in goals.

Overlay multicast:

- Don't need to modify routers.
- Each application can make its own choices (e.g. link costs).
- Can limit which users can participate.
- Can implement custom business models (e.g. track usage and charge accordingly).

IP multicast:

- Must be implemented in routers.
- All routers must be standardized.
- No built-in security.
- No built-in business model.

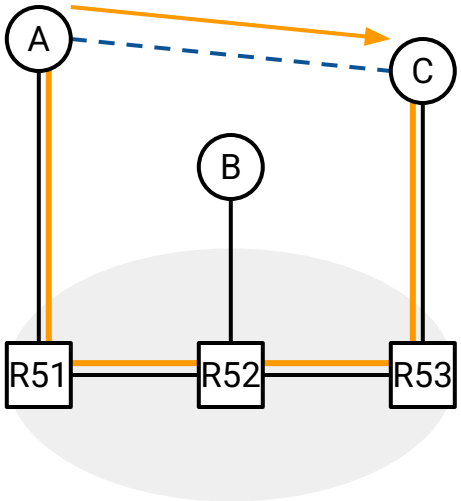
Con: Overlay multicast introduces additional overhead.

- Packets require extra processing, e.g. encapsulating and decapsulating.
- No guarantee that packets take least-cost paths to destinations.
- Application developers must implement multicast themselves.

Overlay Multicast Performance: Stretch

Overlay multicast performance depends on how well the virtual topology matches the physical topology.

- **Stretch**: Ratio of physical path cost to the corresponding virtual path cost.
- Low stretch is good: Physical path cost is similar to virtual path cost.
- High stretch is bad: Physical path is many times longer than virtual path.



Assume all links are assigned cost 1.

Physical path cost: 4

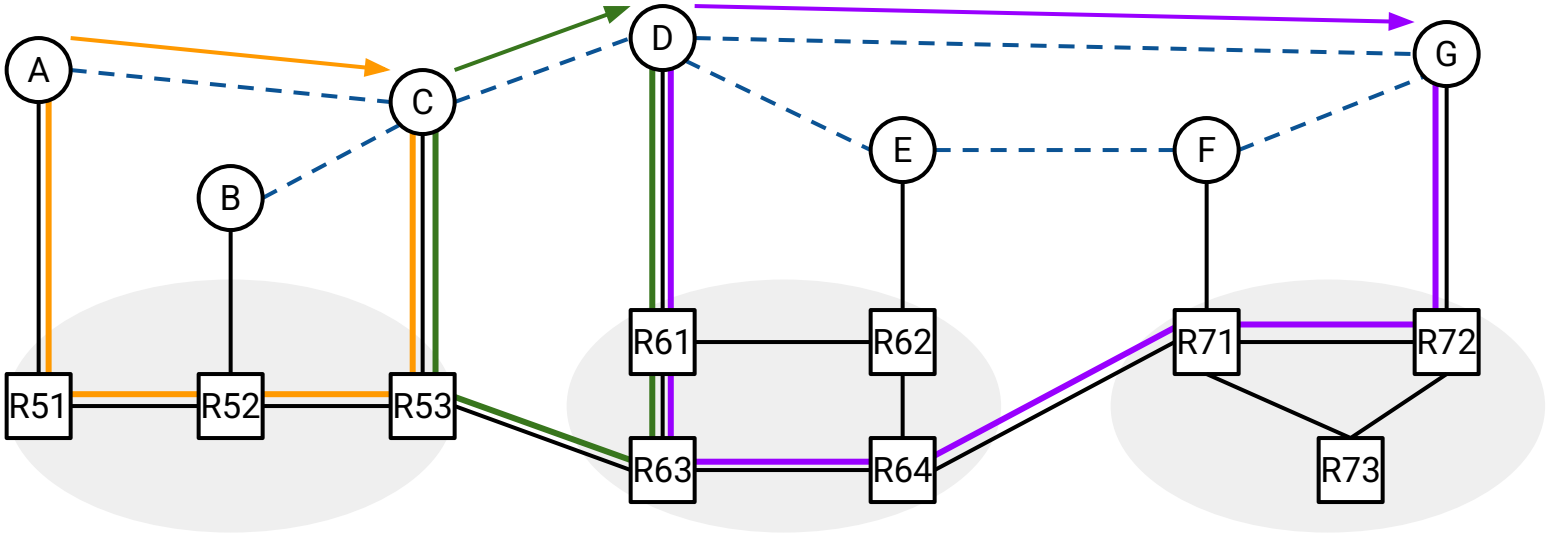
Virtual path cost: 1

Stretch: 4 / 1 = 4x

Overlay Multicast Performance: Stretch

Overlay multicast performance depends on how well the virtual topology matches the physical topology.

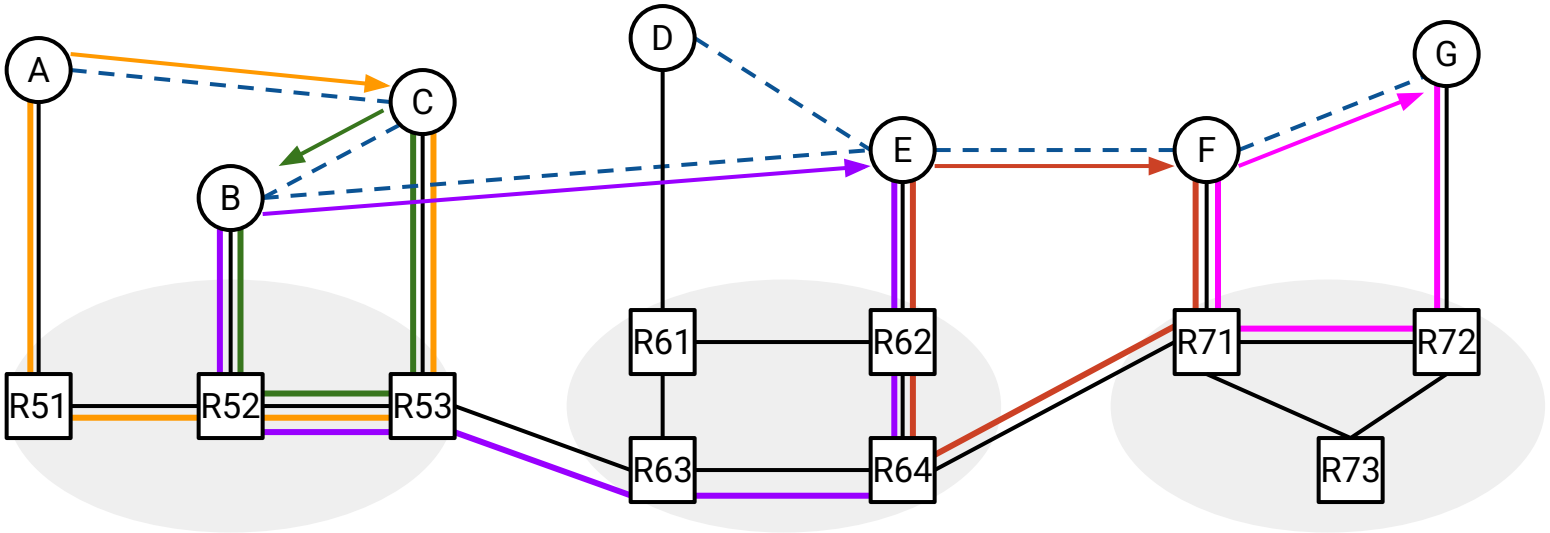
This virtual topology matches well. Packet takes a pretty good path through the underlay network.



Overlay Multicast Performance: Stretch

Overlay multicast performance depends on how well the virtual topology matches the physical topology.

This virtual topology matches poorly. Packet takes a bad path through the underlay network.



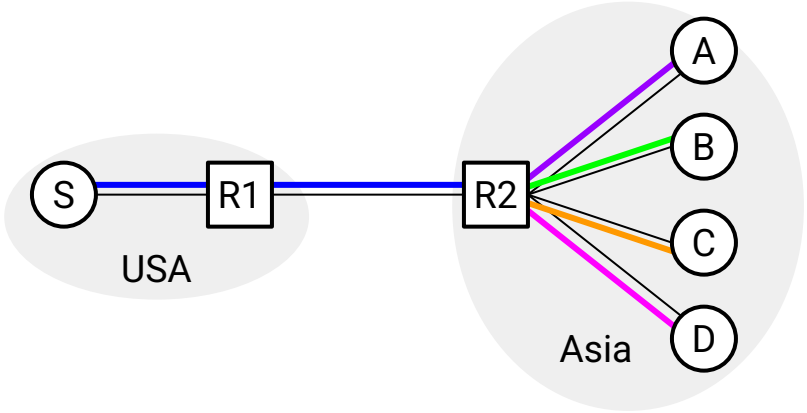
Overlay Multicast Performance

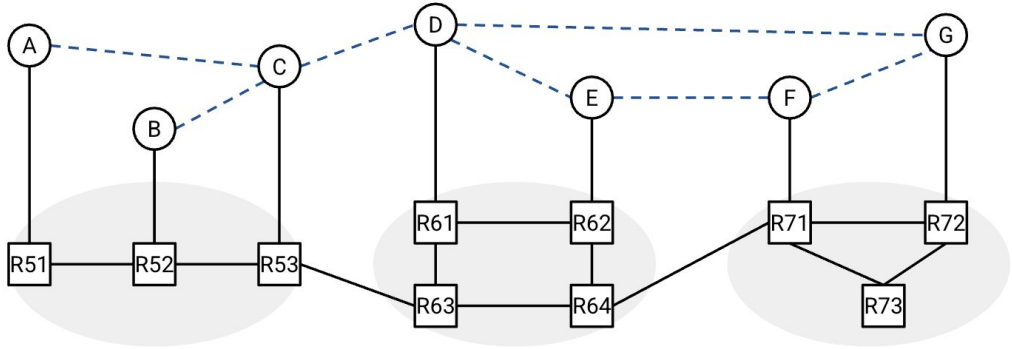
How do you build a low-stretch overlay?

- Manually configured by operator. Common in dedicated networks like datacenters, where the operator controls the topology.
- Self-organizing protocols:
  - Initially, choose neighbors at random.
  - Periodically, search for new candidate neighbors and measure distance to them (e.g. RTT).
  - If the best candidate outperforms your worst current neighbor, swap them.

Summary

Multicast saves bandwidth in group based communication!





DVMRP/CBT: Extend distance vector to multicast, or make a tree with a core node

Challenges of IP Multicast:

- Scaling to Inter-Domain, Business models, Congestion, Reliability, Security

Overlay-Based Multicast:

- Create a fake virtual topology between end-hosts to use as an overlay