CS168

Beyond Client-Server (part 1)

Sylvia Ratnasamy

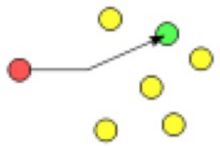
Fall 2024

"The goal of the Internet is to transfer data between hosts"

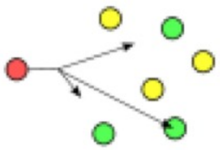
- Our view so far: **unicast** delivery
  - From a source host to a specified destination host
- Foundation for **client-server** applications
  - Ideal for request-response exchanges between two specific hosts
  - Can layer advanced features on top: reliability (TCP), naming (URLs), *etc.*
- But some apps involve **groups** of hosts communicating
- What (if any!) support should the network provide such apps?

Packet Delivery Models

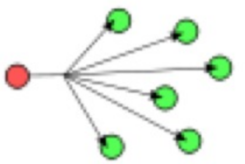
- **Unicast**: sending a packet to exactly one destination



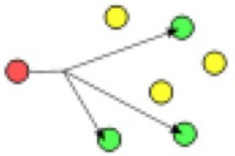
- **Anycast**: sending a packet to one of a set of possible destinations



- **Broadcast**: sending a packet to all hosts



- **Multicast**: sending a packet to multiple hosts
  - But not all hosts, only those that want it!



Multicast

- One-to-many delivery is common in many applications
  - Collab and meeting tools
  - Live content delivery
  - Multiplayer games
  - ML training and scientific apps
  - Discovery (“oy, where’s the printer?”)

- Perennial debate: at what layer should we implement multicast?
  - L3, in switches: complex but optimal
  - L7, in hosts: simpler but less optimal

- We’ll study both options and their tradeoffs

Topics

- IP Multicast: one-to-many delivery at L3
  - The IP multicast service model
  - Multicast routing
  - Other challenges
- ”Overlay” multicast: one-to-many delivery at L7
  - Approach
  - Pros and cons of overlays

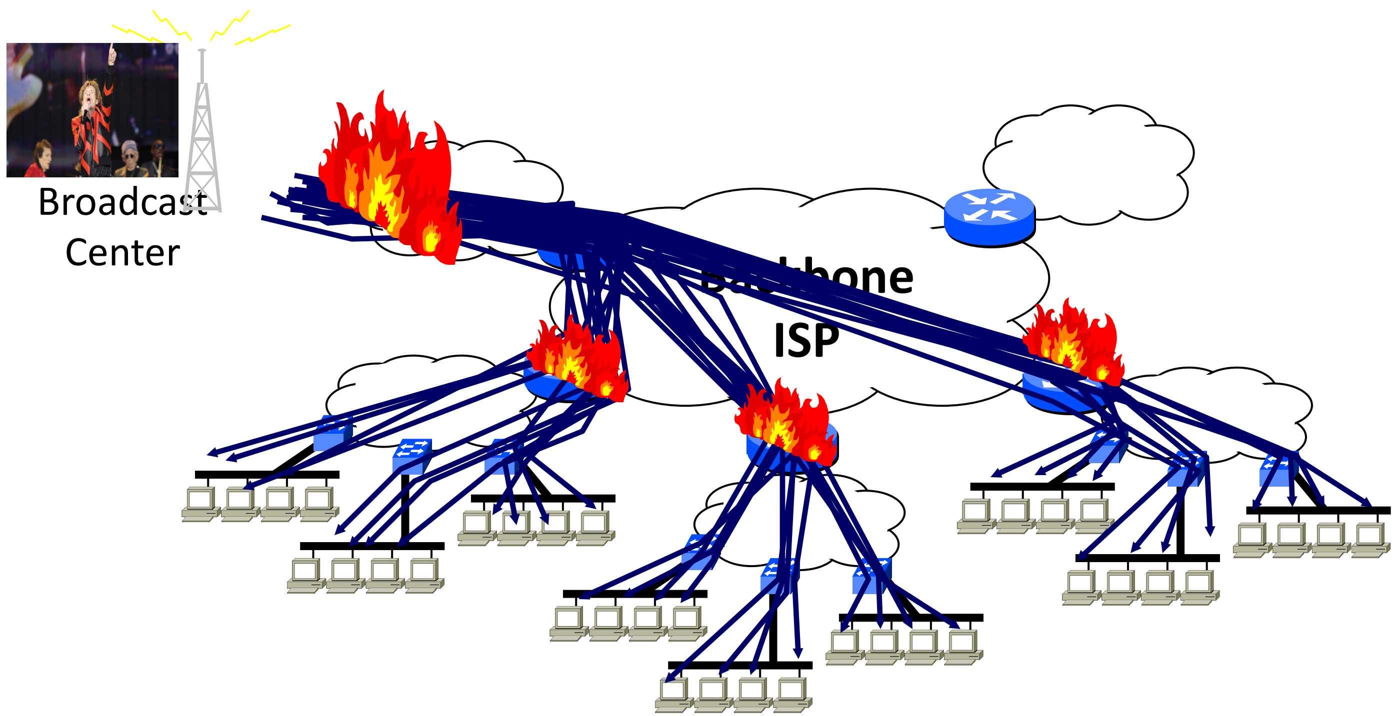
IP Multicast: a bit of history

- Focus of much work in the mid-90s to 2000s
- Anticipated killer app: Internet TV/Radio
- Mixed success in terms of adoption
  - Implemented in all routers
  - Fair bit of use within one domain/network
  - Little/no inter-domain deployment or availability as an e2e service to users
- Techniques developed are now part of our design arsenal
  - Relevant again with collective communication in AI clusters (next lecture)

Early motivation: Internet TV/Radio

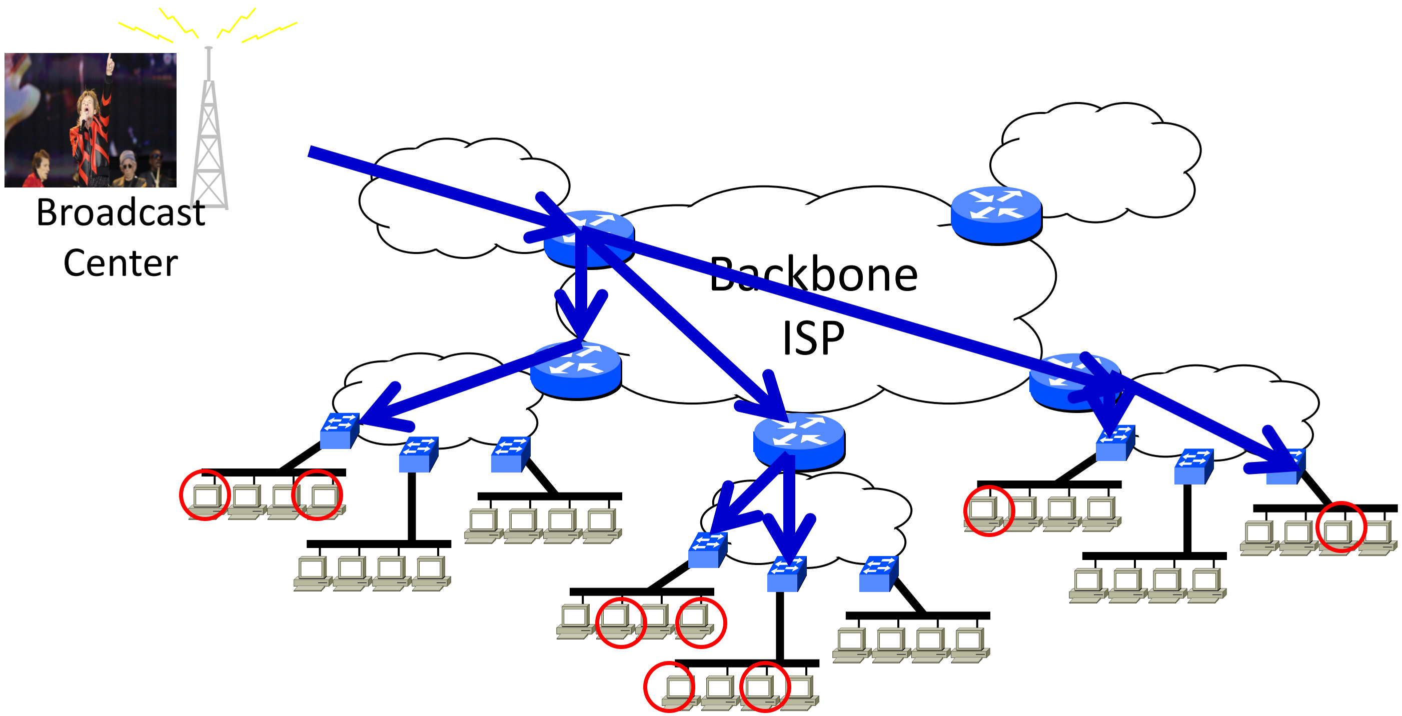
- Internet concert
  - 100 million simultaneous online users
  - Could we do this with parallel unicast streams?
- Bandwidth usage
  - If each stream was 1Mbps, concert requires 100 Tbps
  - This is what the server and outgoing link needs to handle
- Coordination is challenging
  - Hard to keep track of each listener as they come and go
- IP Multicast was designed to address both problems

Unicast approach does not scale…



Instead replicate packets along a delivery tree

- Router forwards an incoming packet on multiple outgoing links
- At most one copy of a data packet per link



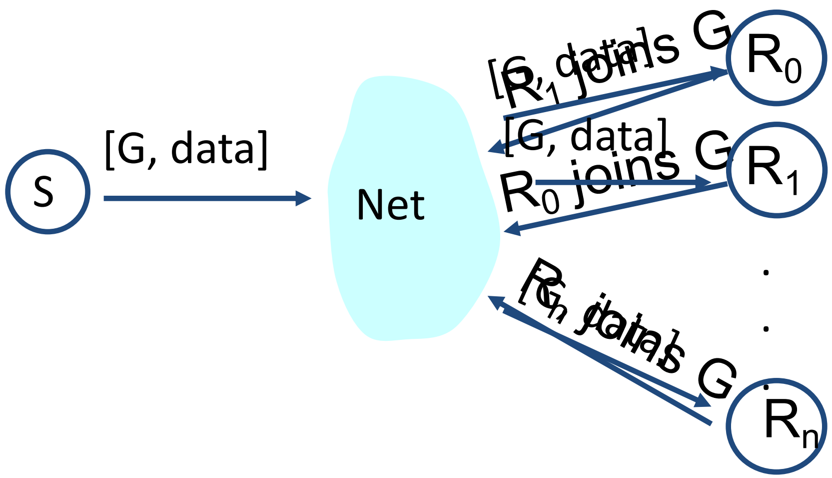
- Routers keep track of groups in real-time
- Routers compute trees and forward packets along them

Multicast Addresses

- Multicast “group” defined by IP address
  - Multicast addresses look like unicast addresses
  - Must be in range 224.0.0.0 to 239.255.255.255
- Using multicast IP addresses
  - Sender sends to the IP address
  - Receivers join the group based on IP address
  - Network delivers packets from sender to receivers

IP Multicast Service Model

- Interface: join/leave(G), sendto(G)
  - Receivers join or leave group with address G
  - Sender sends packets to address G



IP Multicast Service Model

- Interface: join/leave(G), sendto(G)
  - Receivers join or leave group with address G
  - Sender sends packets to address G

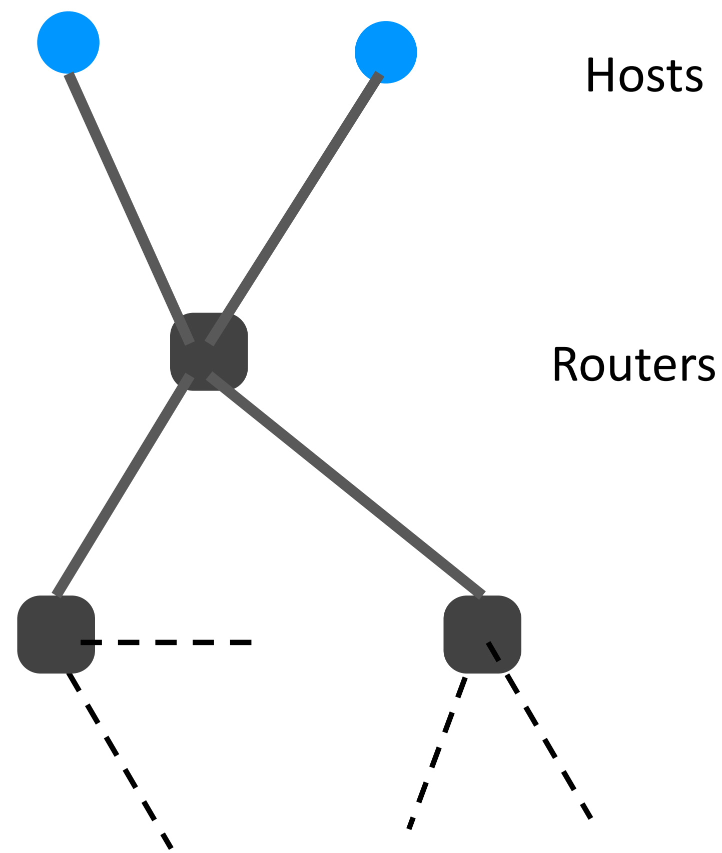
- Characteristics
  - Best effort delivery semantics
  - Dynamic group membership: hosts call join/leave at will
  - Group address hides details of membership from sender (*indirection*)
  - Access control not built-in: anyone can join or send

Implementing the Service Model

Service model

Host-to-Router protocol
(Internet Group Membership Protocol, IGMP)

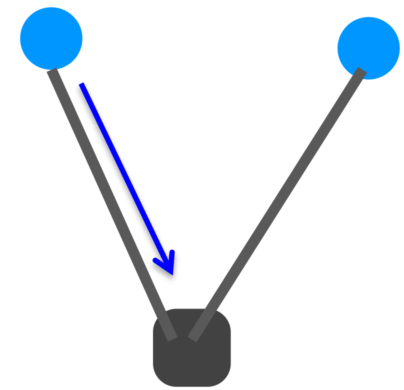
Multicast routing protocols
(DVMRP, CBT)



Host-to-Router

Service model

Host-to-Router protocol
(Internet Group Membership Protocol, IGMP)



Hosts

Routers

(G, outgoing-link, TTL)

(1) Host locally broadcasts its membership in G

(2) First-hop router installs/refreshes forwarding state for G

Router periodically broadcasts a membership query (“still a member of G?”)

Topics

- IP Multicast: one-to-many delivery at L3
  - The IP multicast service model
  - Multicast routing
  - Other challenges
- ”Overlay” multicast: one-to-many delivery at L7
  - Approach
  - Pros and cons of overlays

The Key Challenge: Delivery Tree

- Need a tree from each source to all receivers
- Only want one copy of packet over any link
- Would like an *efficient* tree
  - Path from source to receiver is close to the shortest path
- Two different approaches
  - DVMRP: extension to distance-vector routing
  - Core-Based Trees: new approach
- **Common theme: exploit existing unicast routing tables!**

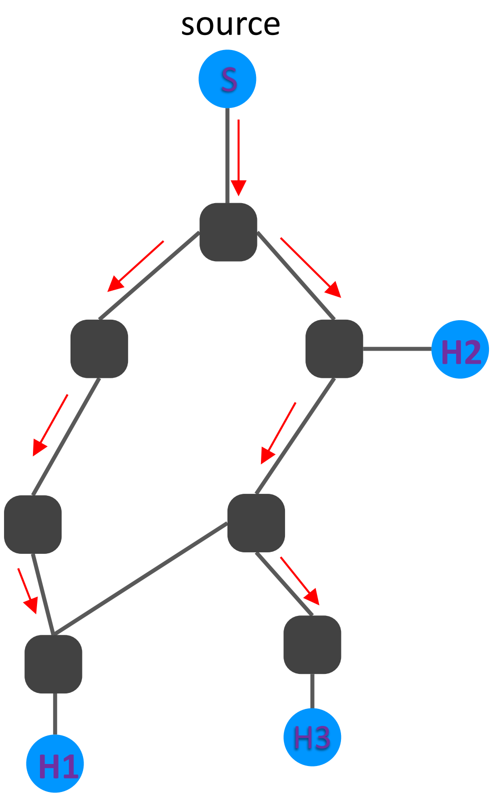
Distance Vector Multicast Routing Protocol (DVMRP)

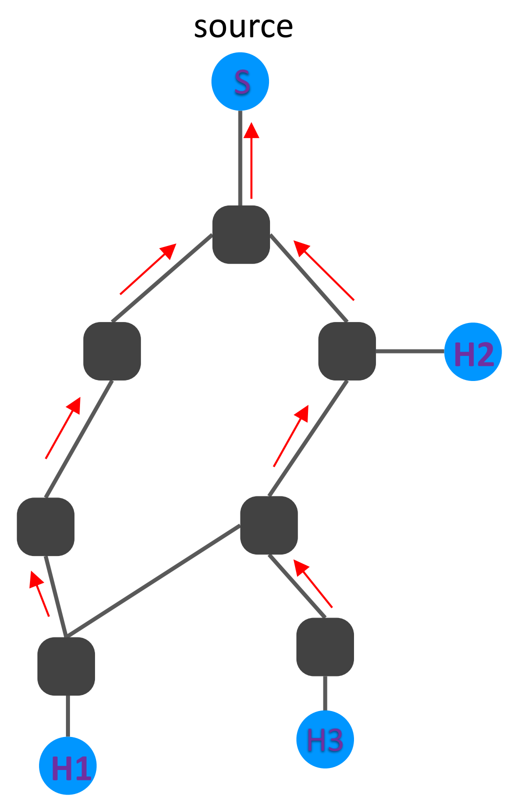
- Extension to DV routing
- Challenge: dynamic and open group membership
  - Don't know a priori which hosts will be a source
  - Receivers can join / leave the group at will
- DVMRP's strategy: build source-rooted trees *on demand*

Constructing a Delivery Tree

- You are given unicast routing tables
- How do you build a delivery tree?
- And how do you do that in real-time?
  - As a packet arrives, you must be able to look at packet and decide where to send it
  - No additional routing state beyond unicast
- *Think about it….*

DVMRP





What DVMRP wants:

- Delivery tree **<u>from</u>** S
- To current group members

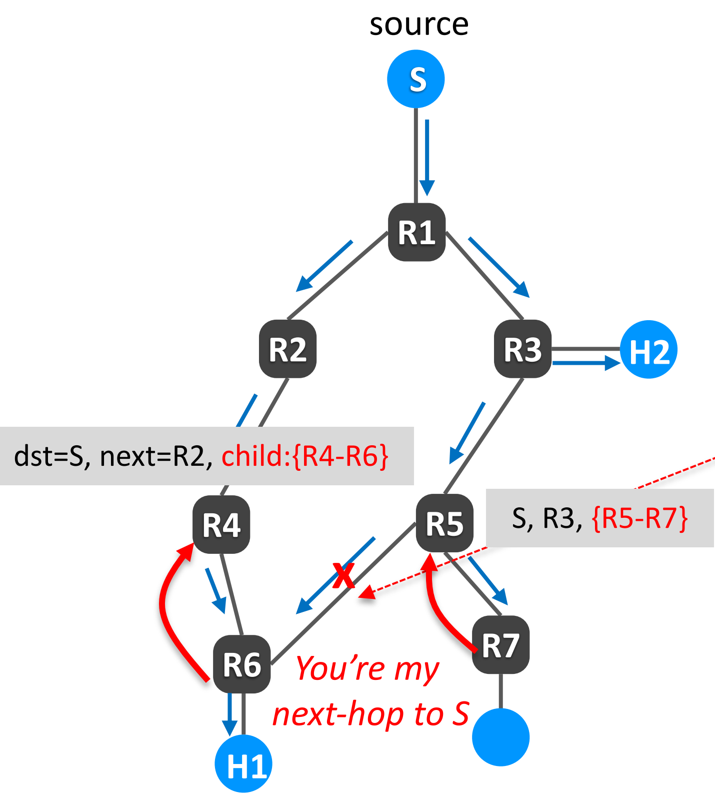
What Distance-Vector gives us:

- Every router has next-hop **<u>to</u>** S (i.e., the *reverse* path)

Use Reverse Paths!

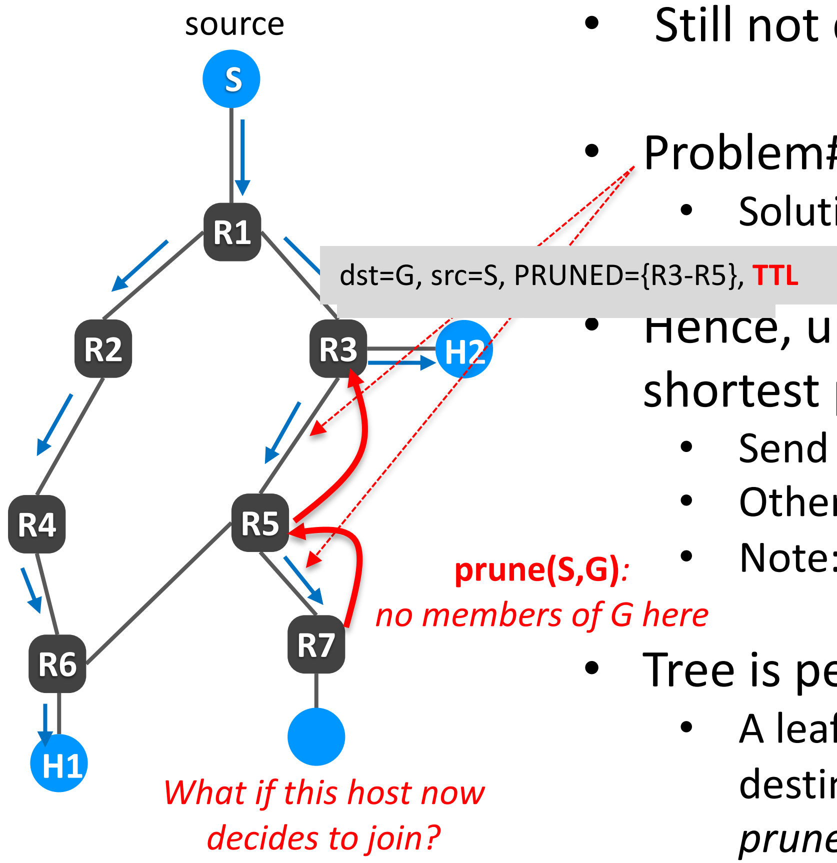
- Arriving packet has:
  - Destination: a multicast group address (G)
  - Source: host sending packet (S)
- Packet arrives at router on incoming port P
  - Would router reach S by sending out that port?
- If no: drop
- If yes: forward out all other ports

Developing DVMRP



- If incoming link is on shortest path **to** source
  - Send on all links except incoming
  - Otherwise, drop
  - Note: no new state at routers!
- But we're not done yet ...
- Problem#1: not exactly a tree
  - Solution: routers inform next-hops of their choice
  - And routers store which links have child nodes in the delivery tree
  - Note: added child state to *existing* FIB entries
- Hence, updated rule: if incoming link is on shortest path **to** source
  - Send on all *child* links for that source
  - Otherwise, drop

Developing DVMRP



- Still not done …
- Problem#2: wasted bandwidth!
  - Solution: “prune” unnecessary branches in the tree
- Hence, updated rule: if incoming link is on shortest path **to** source
  - Send on all child links that have not been pruned
  - Otherwise, drop
  - Note: added FIB entries for each *(source, group)*
- Tree is periodically <u>pruned</u>
  - A leaf router that gets a packet from source S for destination G but has no local members, sends a *prune(S, G)* to its parent
  - If router R receives prunes from all its children, then R sends a *prune(S, G)* to its parent
  - Prune state is deleted after TTL expires

DVMRP Review

- Packets are initially broadcast everywhere
  - Using reverse paths to create delivery “tree”
- Leaf nodes send prunes if they have no members
  - Prunes travel toward source (using forward path)
- **Result**
  - When all prunes have been sent, then all packets from source S travel the subtree that connects S to all members of the group

Critiquing DVMRP

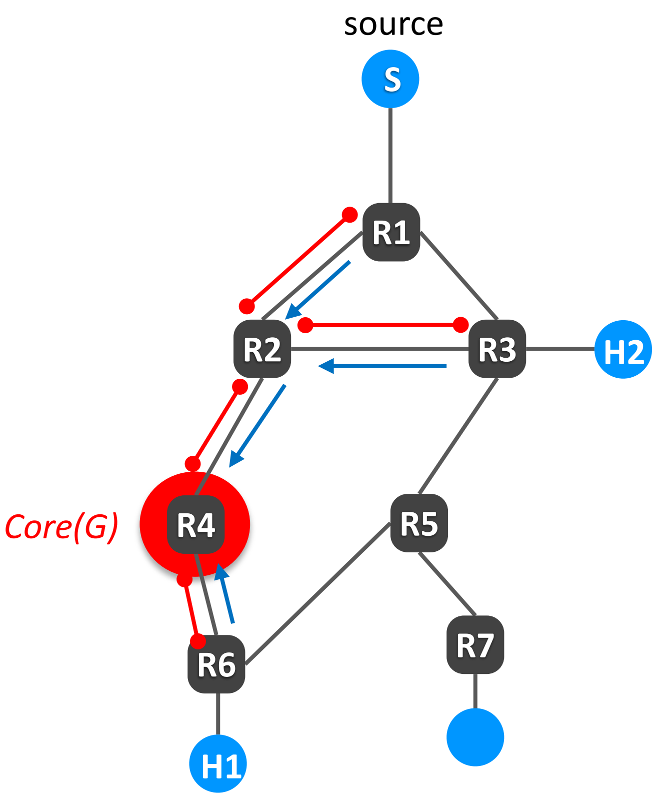
- Scalability
  - Expiring prune state means we periodically flood the entire network ☹
  - O(*#sources* x *#groups*) forwarding entries per router ☹
  - Even a router with no downstream members in G must maintain state for G ☹
  - But only incurred when sources/groups are actively in use ☺
- Architectural approach
  - Simple, elegant extensions to an existing routing algorithm ☺
  - Switching to new unicast routing algorithm (e.g., LS) means rethinking how we implement multicast → need a multicast extension for every unicast solution ☹
  - The difference between extending an architecture vs. layering on top of it!
- Core-Based Trees address these limitations (coming up)

Questions?

Core-Based Trees (CBT)

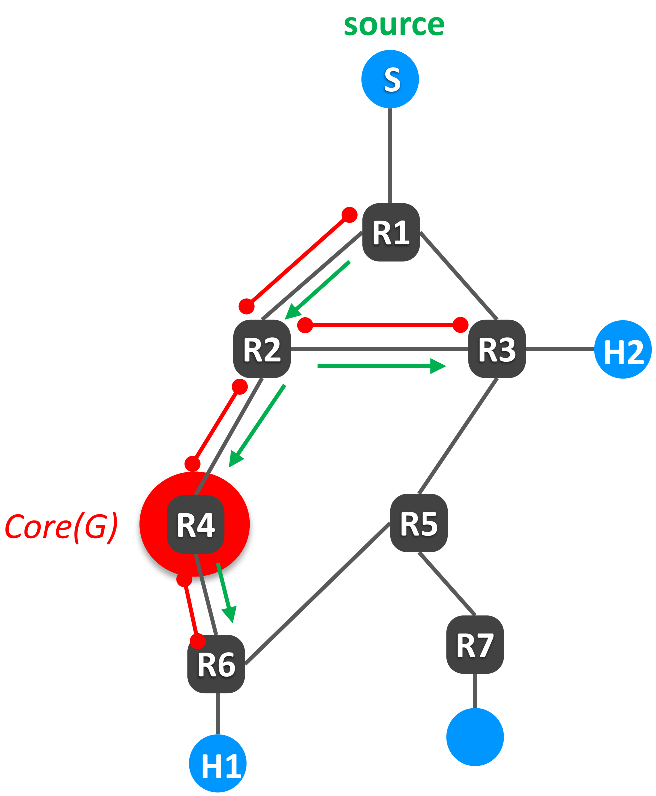
- Idea: build a **single shared tree** for all sources (*vs.* per-source trees)
- One “core” router acts as rendezvous point for the group
  - Core has a unicast IP address: C
  - Assume: mapping from group address G to C is known somehow (e.g., DNS)
- Build tree from all members to that core
  - Member A sends a *join* message to C via regular unicast
  - Routers along the A-to-C path install forwarding entries
  - Hence, tree construction is *receiver-driven*
- Data packets follow the tree; no flooding

Building a Core-Based Tree



Joins are unicast-routed towards the core, until they arrive at a router already on the tree

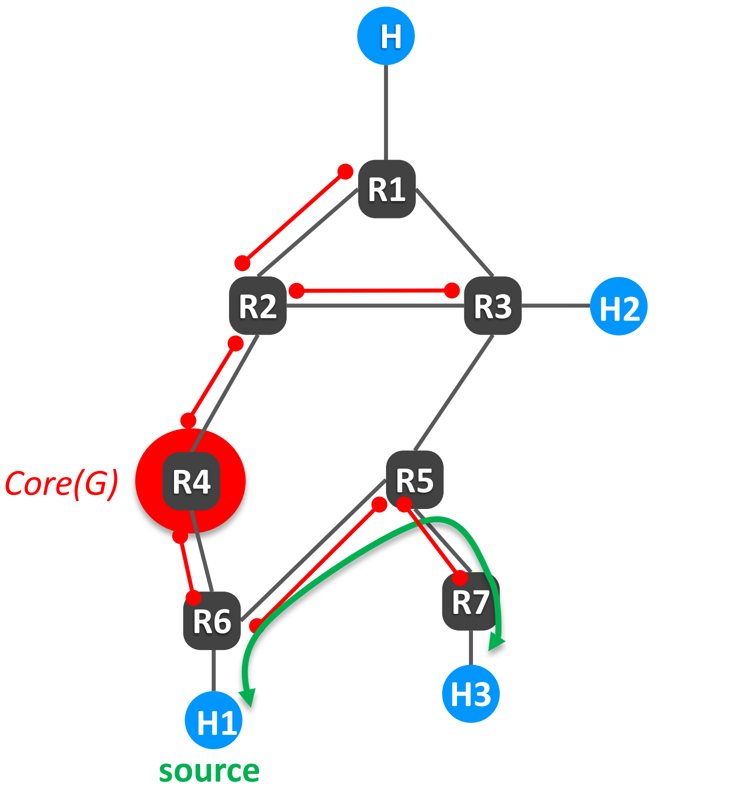
Building a Core-Based Tree



Packets forwarded along the shared tree

Dealing with link/router failure? Other dynamics?

Core placement is key to efficiency



Sending Packets

- Members:
  - Send on tree (broadcast)

- Nonmembers:
  - Encapsulate packet and send to core
    - Using core's IP address
  - Core then sends it on tree

Critiquing CBT

- Scalability ☺
  - No flooding!
  - O(G) state at routers
  - Routers not on the path have no state!
  - (Can we do better?)
- Efficiency ☹
  - Paths often sub-optimal (i.e., not shortest path)
  - Performance sensitive to core placement
- Fault tolerance ☹
  - Failure of the core is problematic! What about routers on the tree?
- Architectural ☺
  - CBT works with any unicast routing protocol

Review of Multicast Routing

- DVMRP:
  - Per-source trees (reverse path!)
  - Flood then prune
  - Issues: scalability - state and flooding

- CBT:
  - Shared tree (all sources have same tree)
  - Built by receiver joins sent to core
  - Any sender can reach tree by going to core

- In routers today
  - DVMRP → Protocol Independent Multicast – Dense Mode (PIM-DM)
  - CBT → Protocol Independent Multicast – Sparse Mode (PIM-SM)

Questions?

Topics

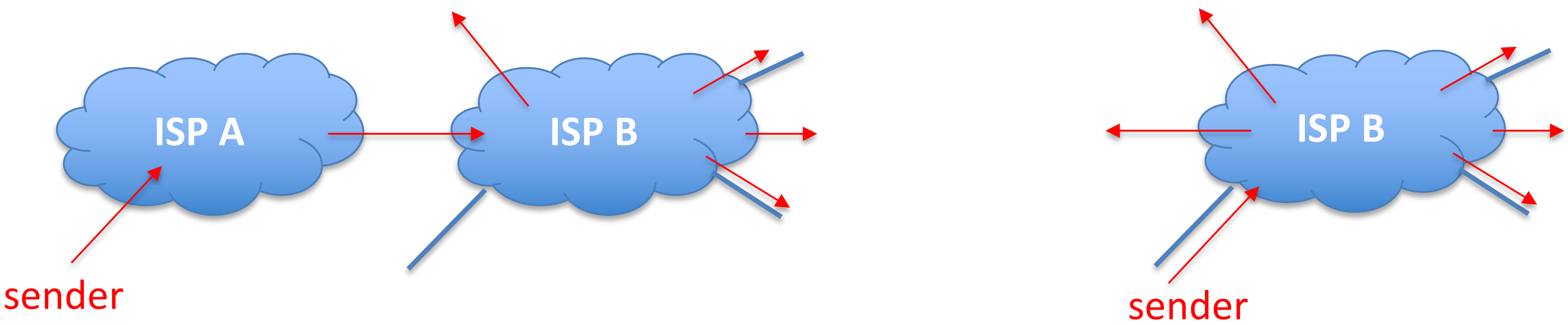
- IP Multicast: one-to-many delivery at L3
  - The IP multicast service model
  - Multicast routing
  - Other challenges with IP multicast
- ”Overlay” multicast: one-to-many delivery at L7
  - Approach
  - Pros and cons of overlays

Inter-domain Routing?

- Using DVMRP? Can’t flood then prune in a global network
- Using CBT? Where do we place the core?
  - Domains don’t want to depend on a core in another network
- Very hard problem. Much work but limited adoption in practice.

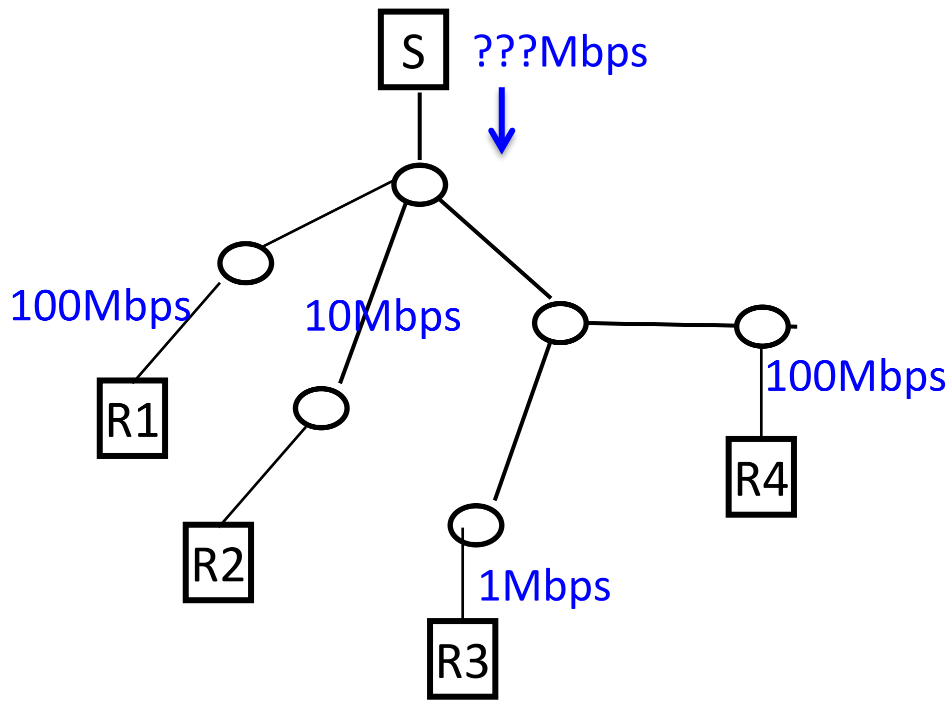
Charging?

- At odds with ISP's model of billing based on input rate
  - One packet in, many packets out!



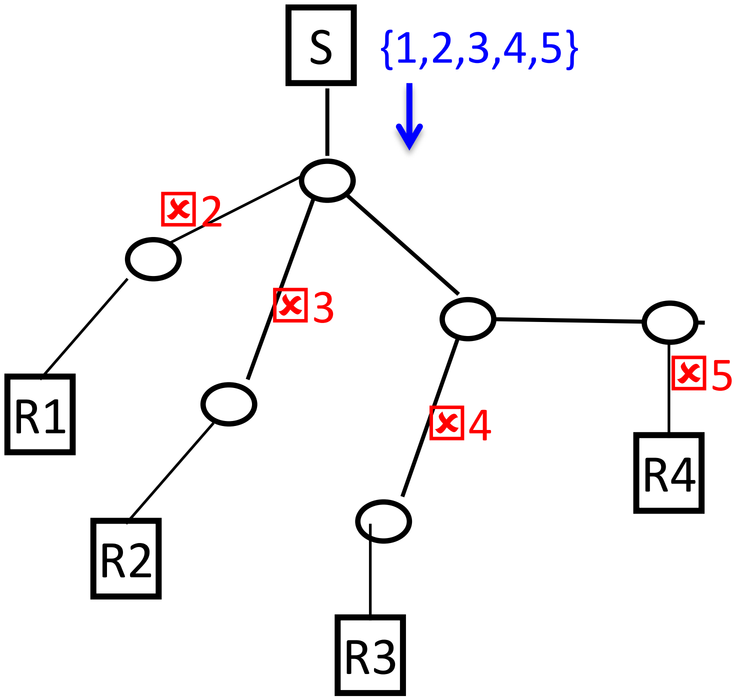
- Plus, the abstraction offers no indication of group size
  - Makes it hard to support new billing models

Congestion Control?

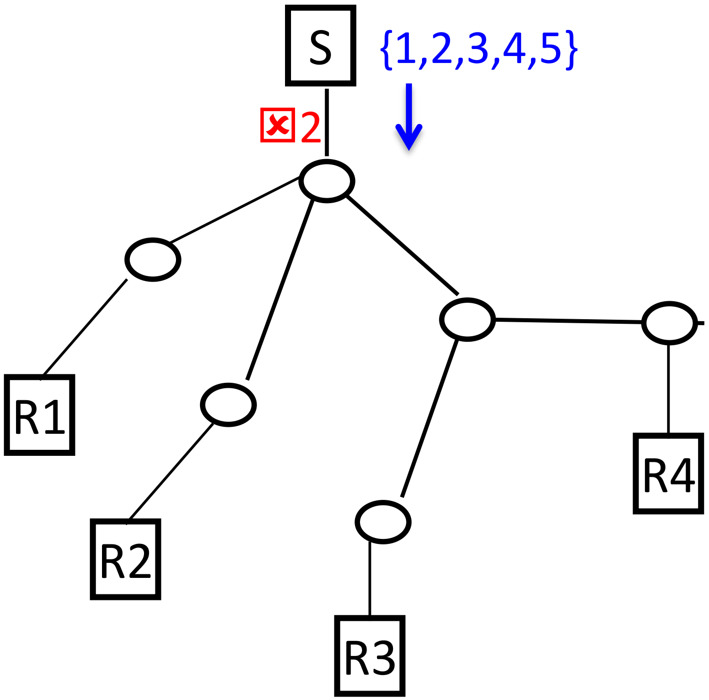


Reliability?

Problem: uncorrelated loss



Problem: correlated loss



Security?

- Scalable access control proved very difficult
  - Open service model allows a malicious sender to overwhelm many receivers without consuming much bandwidth at the sender
  - Dynamic group membership makes it hard to control where traffic does/doesn’t reach

Recap.

- IP Multicast
  - Compelling service model and use-cases
  - But raised many hard problems (also many creative solutions)
- Today: selectively deployed, usually in a single domain
  - E.g., AT&T U-Verse, high-frequency trading datacenters, local area discovery
- Instead, each app pursued its own simpler approach
  - Central relay server (gaming)
  - N-way unicasts (small-scale conferencing)
  - Overlay based solutions (coming up)

Topics

- IP Multicast: one-to-many delivery at L3
  - The IP multicast service model
  - Multicast routing
  - Other challenges with IP multicast
- ”Overlay” multicast: one-to-many delivery at L7
  - Approach
  - Pros and cons of overlays

More History: In the early 2000s ...

- Deploying IP Multicast is slow going

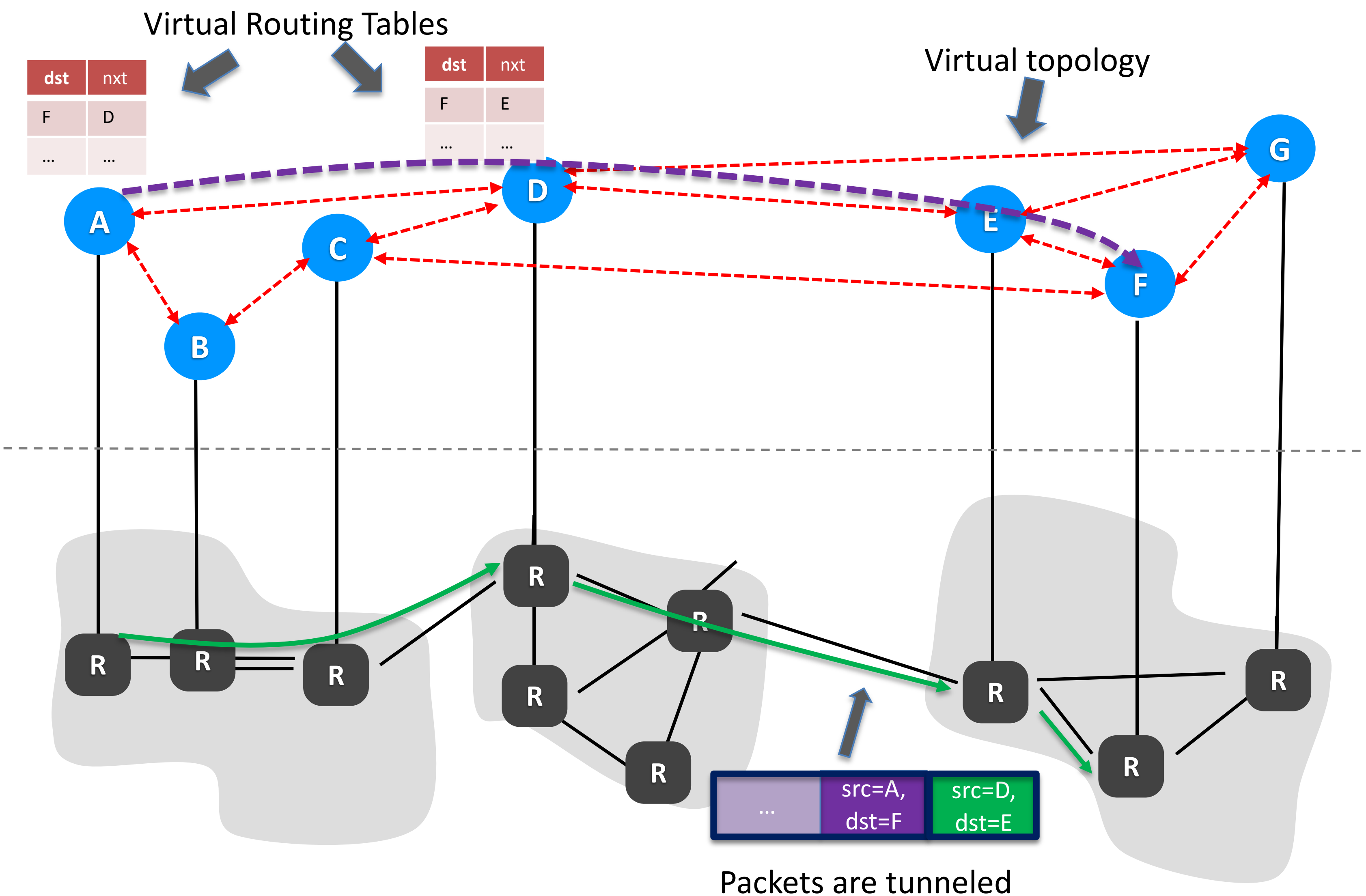
- Many startups emerge
  - FastForward Networks (Berkeley)
  - ProxyNet (Berkeley)
  - Sightpath (MIT)
  - **Akamai** (MIT)

- Common theme: “overlay” based multicast

Overlay-Based Multicast

- Idea:
  - Move multicast support out of the network layer (and switches) and into the application layer (and hosts)

Overlay-Based Multicast



Overlay-Based Multicast

- Idea:
  - Move multicast support out of the network layer (and switches) and into the application layer (and hosts)

- Do this by:
  - Constructing a virtual topology between hosts
  - Run a multicast routing algorithm on overlay topology
  - Packets are tunneled between nodes in the overlay

More Generally: Overlay Networks

- A network
  - Defined by addressing, routing, and service model for communication
- Overlay network
  - A logical network built on top of a physical network
  - With potentially different addressing, routing, etc., from underlay
- Nodes are often end hosts
  - End users, as part of the application (e.g., “peer to peer” file sharing)
  - Or dedicated proxies, deployed by a service (e.g., CDNs)
- Many overlay networks may coexist at once
  - Over the same physical network
  - Each offering a different service/application

Benefits of (Multicast) Overlays

- Ease of deployment
  - Do not have to modify existing routers or protocols
  - Can leverage existing infrastructure for bootstrapping
  - Do not have to deploy at all nodes or process all traffic
  - No need for standardization

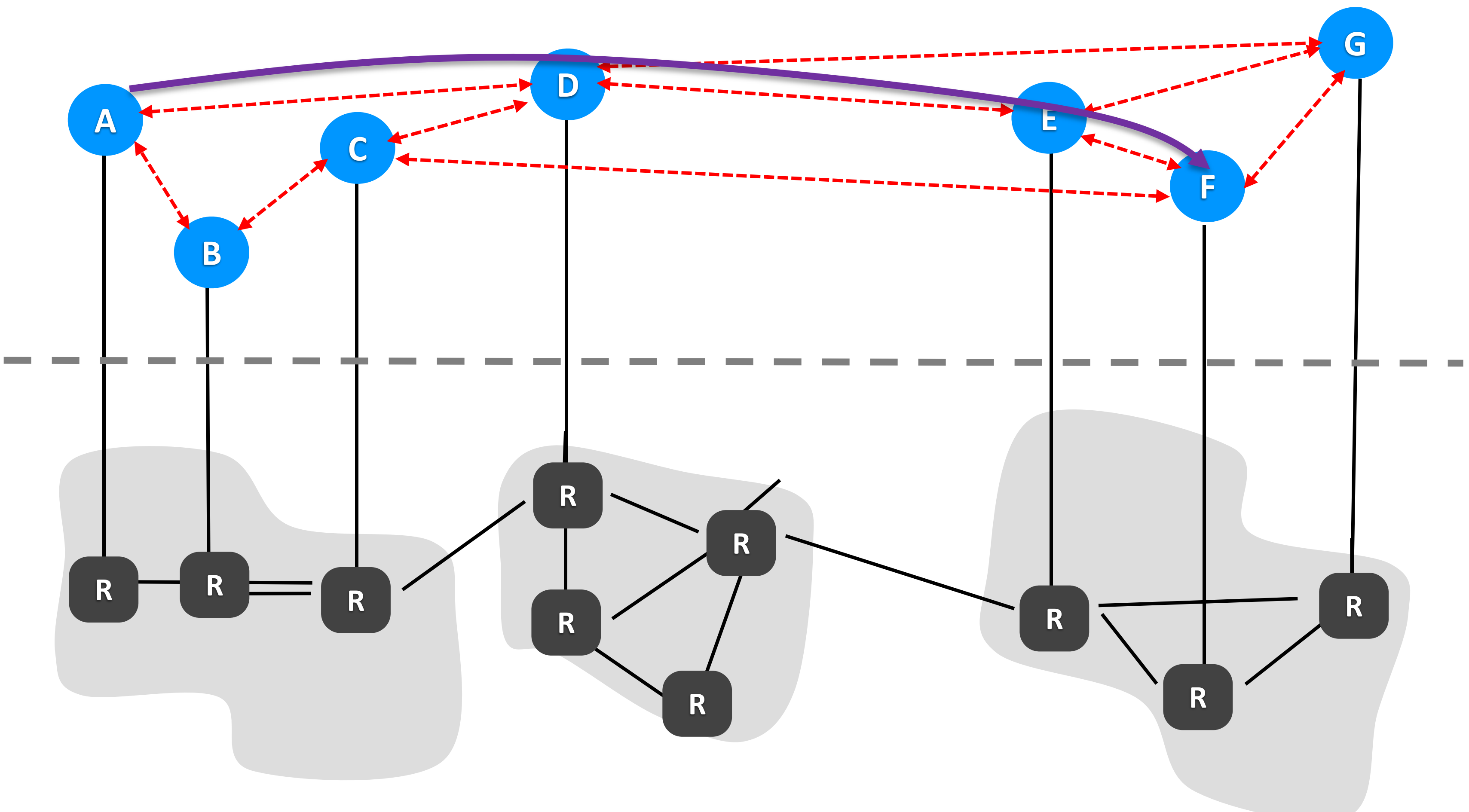
- Accommodates diversity in goals
  - E.g., route selection based on latency, throughput, etc.
  - E.g., limit participation to trusted hosts

- Accommodates diversity in business models (or lack thereof)
  - End-system or P2P: driven by user adoption (e.g., evading copyright)
  - Proxies: in support of 3rd party services (e.g., CDN providers)

Drawbacks

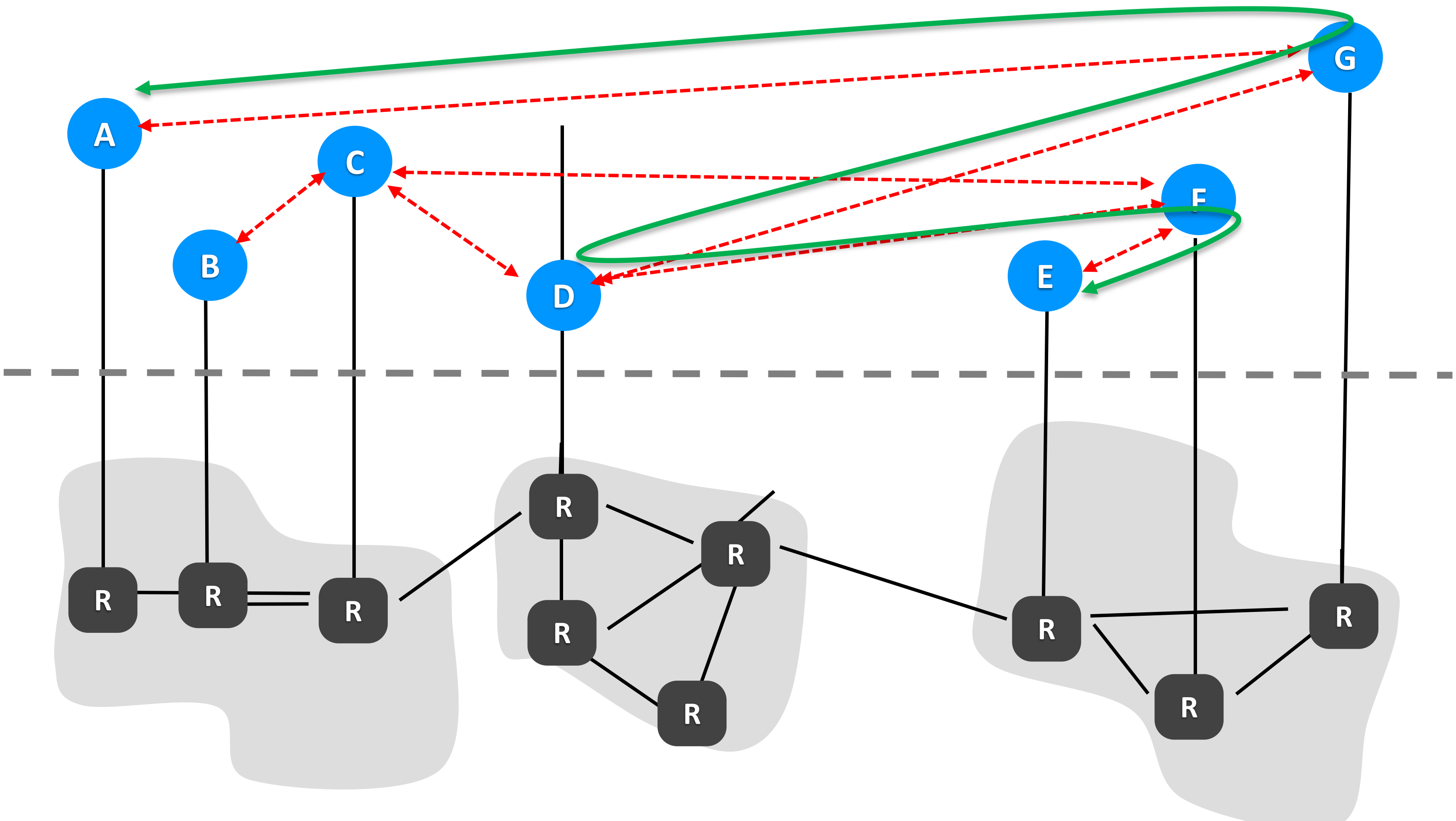
- Sub-optimal performance

Performance with Overlay-Based Multicast



Performance of overlay multicast depends on how well the virtual topology (overlay) matches the underlying physical topology (underlay)

Performance with Overlay-Based Multicast



Performance of overlay multicast depends on how well the virtual topology (overlay) matches the underlying physical topology (underlay)

Performance of an overlay

- Measured by average path *"stretch"*
  - Stretch = ratio of path length on overlay to path length on underlay

- Many possible solutions for building low-stretch overlays
  - Manual
    - very common in dedicated overlay infrastructures (e.g., CDNs)
  - Self-organizing:
    - Initially, every node starts with k random neighbors
    - Periodically, a node does a "random walk" to discover new candidate nbrs.
    - Measure your performance to candidate nbrs.
    - If best candidate outperforms your worst current neighbor, swap them

Drawbacks

- Sub-optimal performance
- Adds overhead
  - +1 layer in the networking stack (headers, processing)
- Not “built in” to the Internet, hence app developers must arrange for the availability of multicast infrastructure

Summary

- IP Multicast
  - Compelling service model
  - But realizing the service model raised hard problems
  - Ultimately fell short on widespread adoption

- Overlay-based multicast often “good enough” and simpler to deploy

- Lessons: consider the design alternatives and tradeoffs!