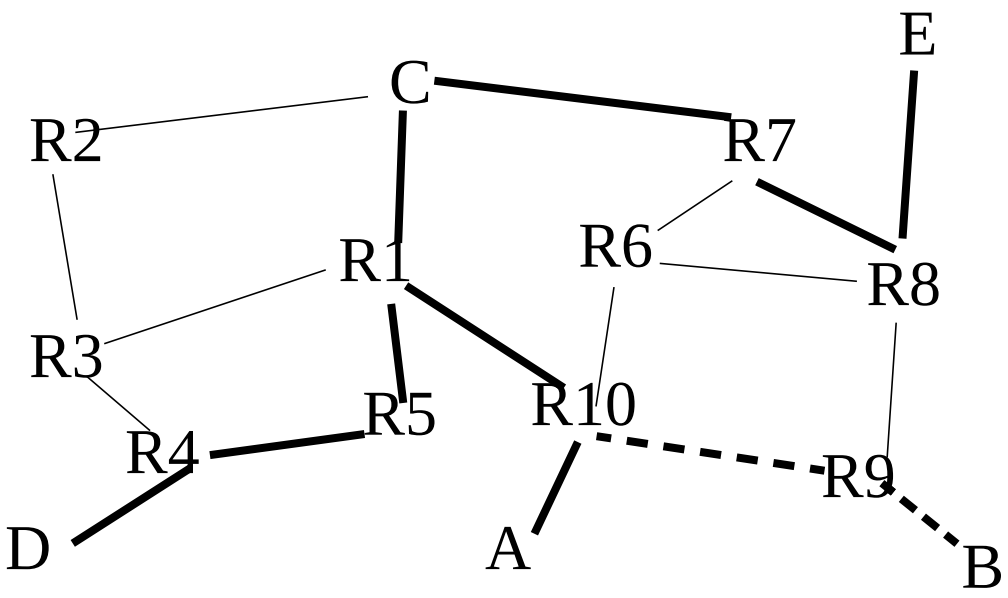
Thoughts on Multicast

Radia Perlman, SunLabs

radia.perlman@sun.com

Basic Core Based Trees

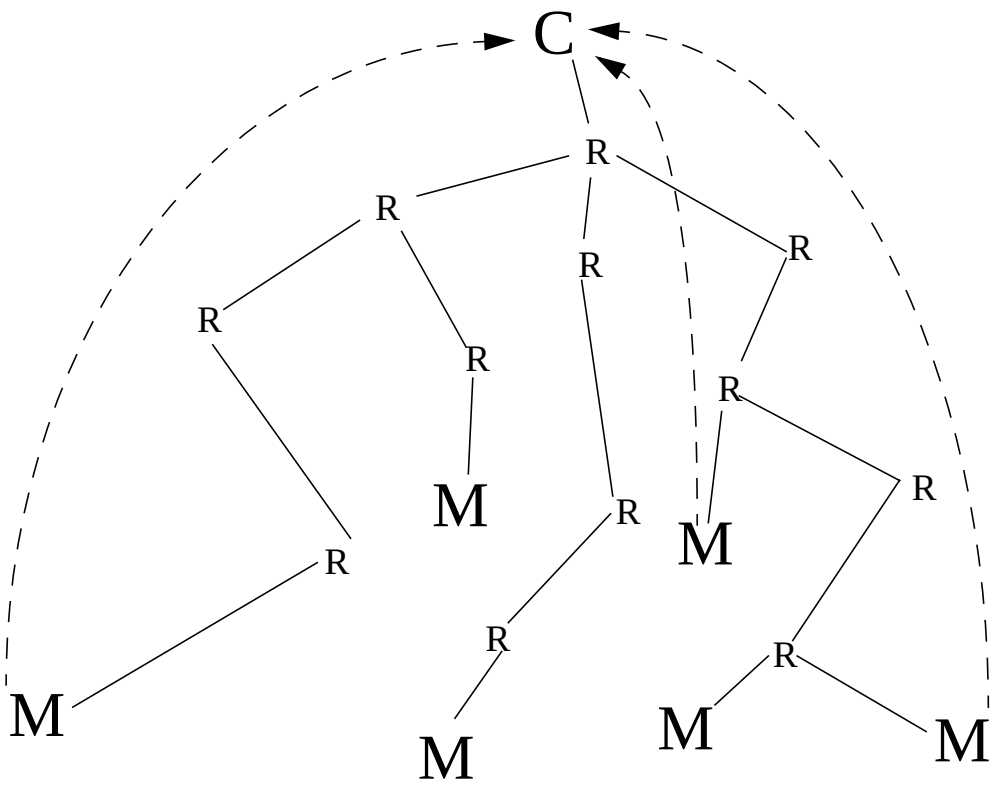
- Choose special node for group M called “Core”
- To join M, send “join” towards the Core
- Routers along the way keep state
- If a router already knows about M, limb just gets grafted onto tree



PIM

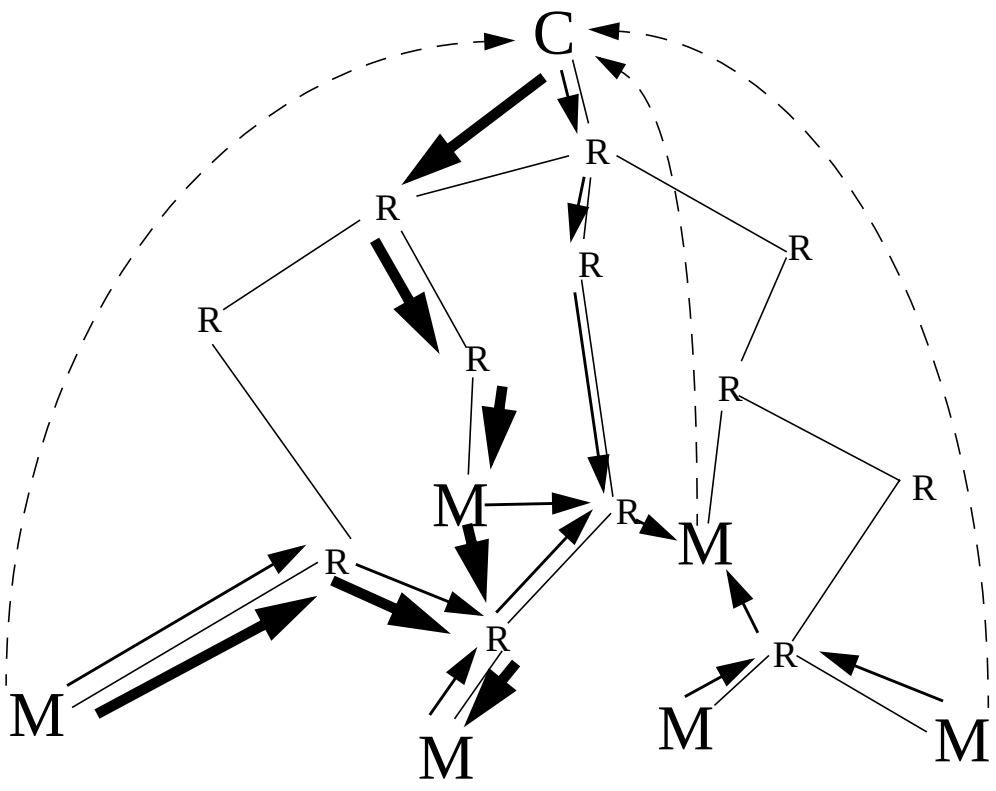
- Automatically create per-source trees
  - Each rcv joins S if “enough” traffic from S
  - Result: Core tree needs to be unidirectional (and therefore very suboptimal)
  - k times more state for routers, complex alg
- Routers have to be able to map M to core addr
  - “core capable routers” advertise within domain (mercifully not within whole net!)
  - hash alg: maps M to one of set of routers
  - likely to be very suboptimal (core anywhere)
  - multicast addresses assigned to domains, so interdomain can find domain that “owns” M
  - multicast address ranges passed in interdomain routing protocol

Unidirectional Shared Tree



- Core chosen arbitrarily from set of “core capable routers” rather than to be close to the group

Plus Per-Source Trees



- Instead of one tree per group, multiply by number of transmitters

Alternative Proposal

- No per-source trees. Shared bidirectional tree.
- Core is (by default) a member of group, or else consciously (well) chosen for that group
- Core fixed at time of group creation
- Core address found by member when finding M, through whatever means you find groups
- Cheaper because
  - no need to keep state about per source trees
  - no advertisements by core capable routers
  - no passing mult. adds in interdomain rtg.
  - shared tree bidirectional, and good
- Simpler administration
  - addresses just need to be unique (not aggregatable). Won’t run out.

Assigning Mult. Addresses

- Have hierarchy of address servers. Give each top level guy 1/n of space
- Have lower level guys get blocks from higher
  - avoid bottleneck
  - higher level guy not have to keep track of as many address timers
  - have address server close and convenient when needed
- To get address, find any mult-add-server, and ask for address, and time limit. After that time, it can reassign address
- Addresses have no topological significance
- If necessary, have some “local” addresses that don’t escape out of the domain

Creating a Group

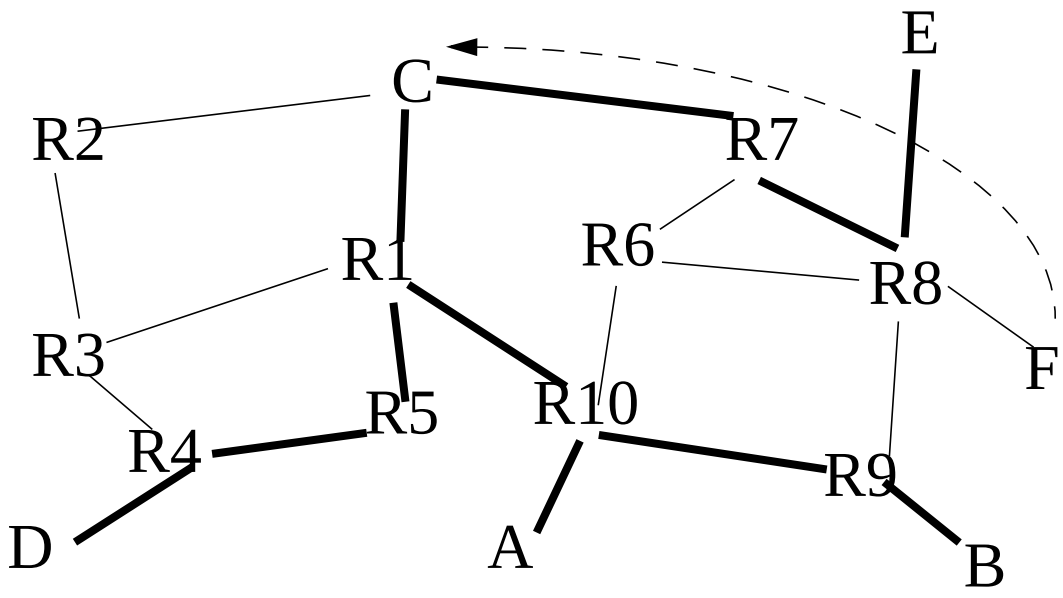
- Find a name
- Acquire an address M from an address server
- Choose core C.
- Register group name, M, and C in the directory (SDR, etc.), or explicitly distribute (email, phone, web page, etc.)
- In some cases, create multiple groups
  - For super-availability
  - For simple backup
  - For super delay-sensitive apps, create group for each high-volume Xmitter
  - Can spread traffic so it isn't all on one tree
  - Still less state in routers than always separate group for each Xmitter, plus shared tree!

Joining a Group

- Find group in directory, or be “invited” by explicit email, phone, web page ad, etc.
- Your node joins that group, two possible ways:
  - Specify M and C in IGMP reply,
  - or (if “impossible to modify IGMP”) bypass IGMP entirely and send join towards C
- If multiple members on LAN, and members send joins, joins won’t go beyond first router (same number of packets as IGMP)
- Routers keep state:
  - M
  - C
  - expected port to C
  - ports downstream from C

Transmitting to M

- If you're a member, simply transmit with destination address M
- With bidirectional tree, it's equally efficient no matter where you inject the packet from
- If you're not a member, tunnel the packet to C

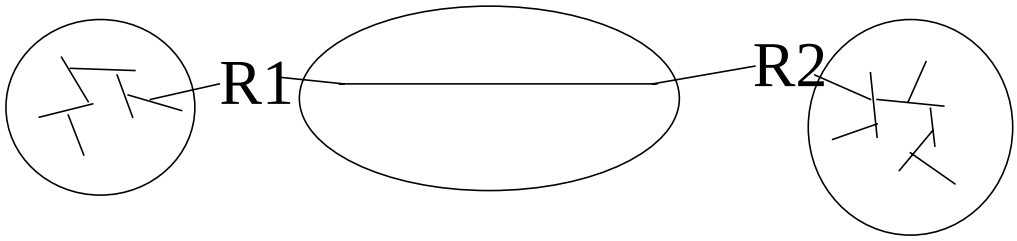


Can't Router Look up Core?

- Proposed compromise: have router look up core in directory (based on multicast address M), rather than endnode (looking it up based on human-understandable name)
- But endnode looks it up based on hierarchical NAME, more scalable than reverse lookup on multicast address
- Endnode already has to look up the group. Why make the router also do that work?
- Some groups may not be public (find out through email or phone or web page)

Interdomain Groups

- Everything works. No special protocol needed
- No need to advertise multicast addresses interdomain, since unicast routing is all you need to find core
- If you want to optimize, when the core is outside the domain, use one exit point from domain for each IP prefix

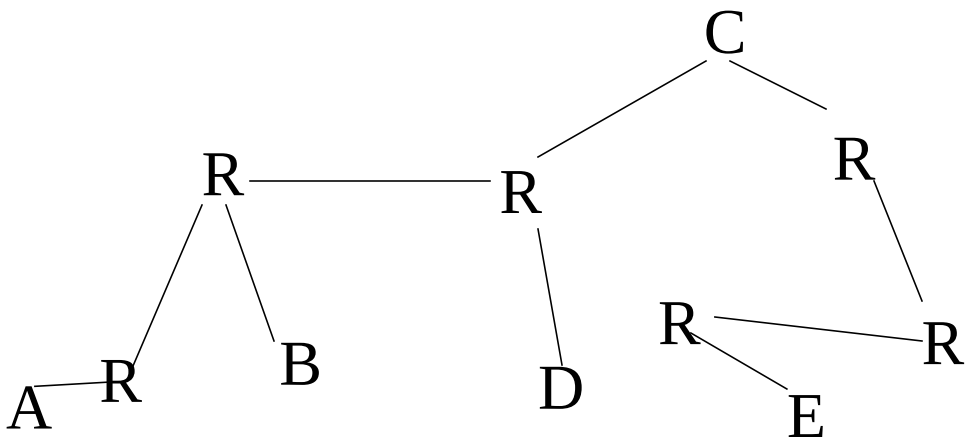


Per-Source Trees

- No better than shared tree if metric is total cost for network to deliver packet. A tree is a tree (provided it's bidirectional)
- Only metric with per-source tree better is delay from source to each receiver
- In what applications does this really matter? Is it worth building the design around this, adding k times as much state, for these apps?
- For those apps, can create multiple groups, one for each *location* (not necessarily source) from which delay matters so much. Listen on all those trees. Xmit to appropriate tree
- Note: if application is so delay-sensitive, it's not going to work if a distant member joins, and can optimize with a few trees (prev bullet)

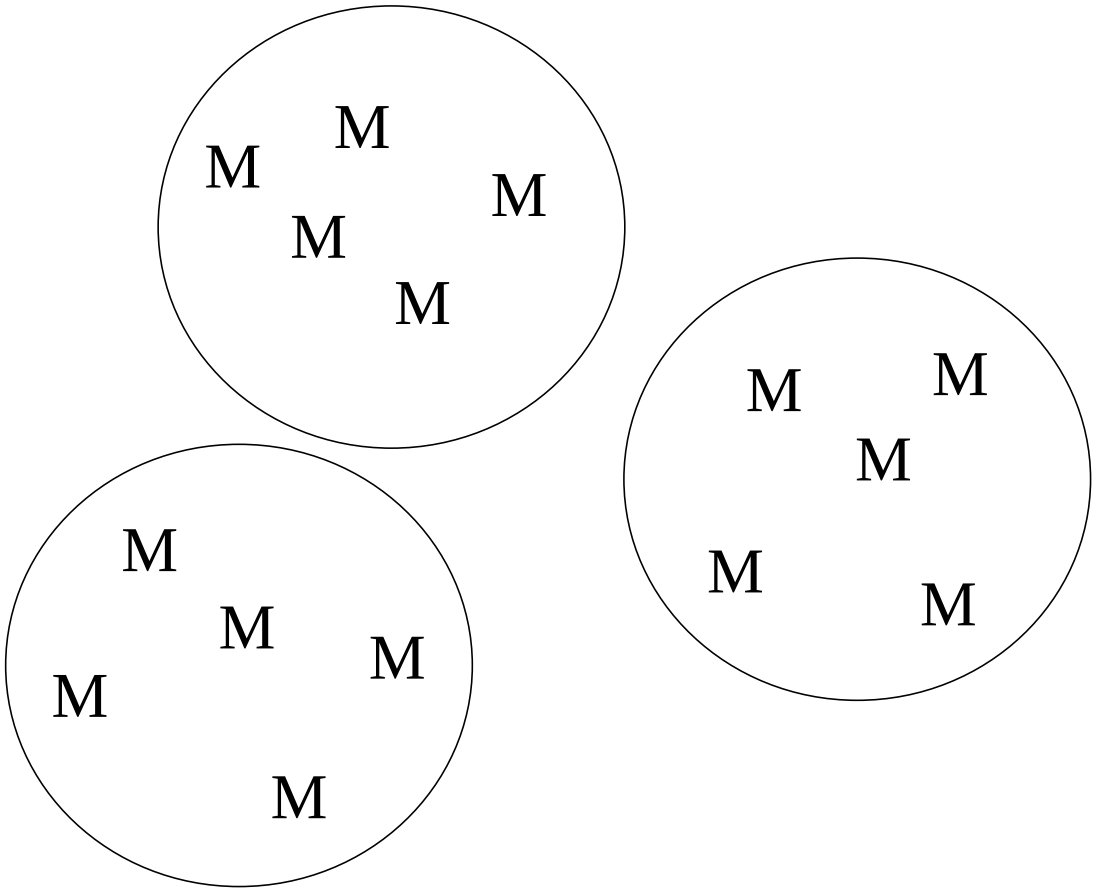
Per-Source Trees

- It does **NOT** minimize network bandwidth to use a per-source tree (assuming a sensible core, e.g., one of the members, and a bidirectional tree)!



- You can always explicitly set up multiple trees from “important” sources or locations, if important
- No “bandwidth” bottleneck around core..it’s just another node in the tree

Per-Location rather than Per-Source



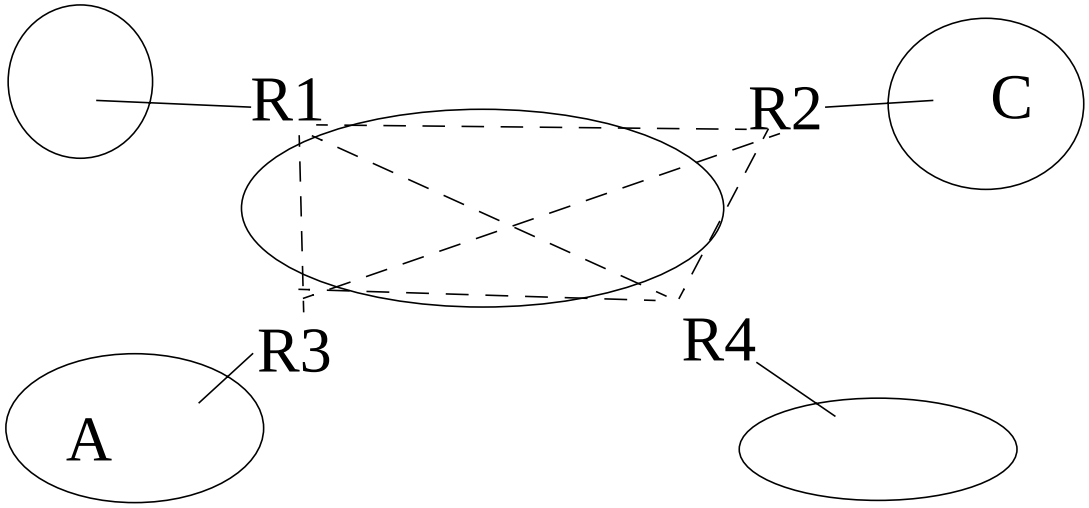
- Three trees rather than fifteen, for equivalent effect. Planned rather than automatically created

Detecting Tree Failures

- Failures to detect:
  - path to core broken
  - core dies
  - branch unused (receiver died)
  - loop
- Keep-alives in both directions. UP if at least one child port has received keep-alive recently
- If packet loops..., delete port. Rcvrs will rejoin
- For dead core, whichever appropriate:
  - manually recreate new group (no rtg mech.)
  - proactively create k groups, diff multicast addresses, send all data k times (for super-reliability needed), or switch if necessary
  - create backup group when needed

Minimizing State in Backbones

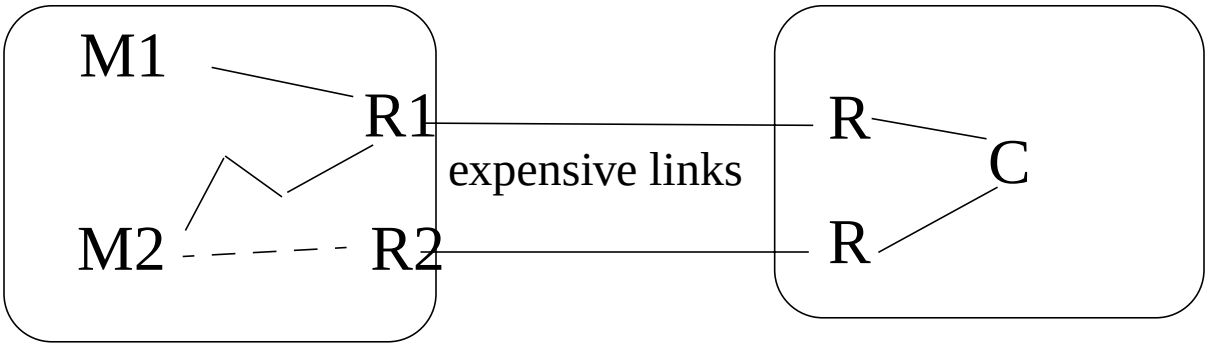
- Use tunnels



- Only the border routers need to know about groups, and only those groups with members in their domain that go outside their domain

Intermediate Cores

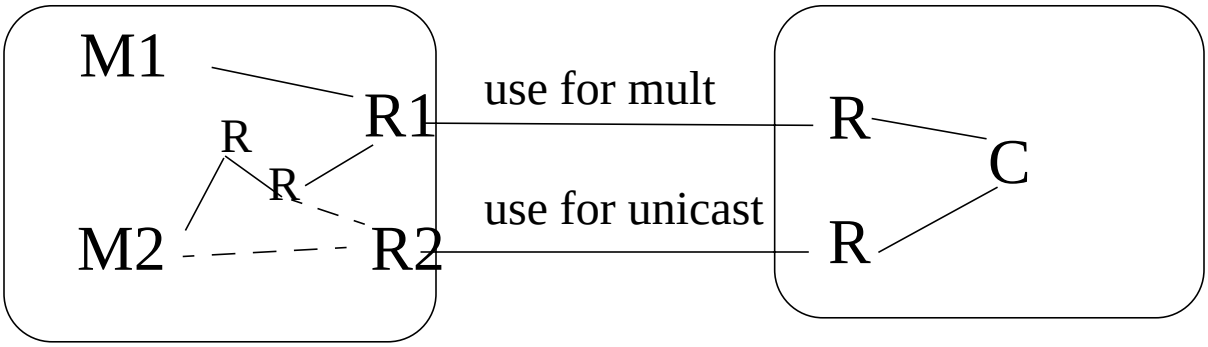
- Create near-optimal trees, both from delay from sender point of view, and from bandwidth on network point of view



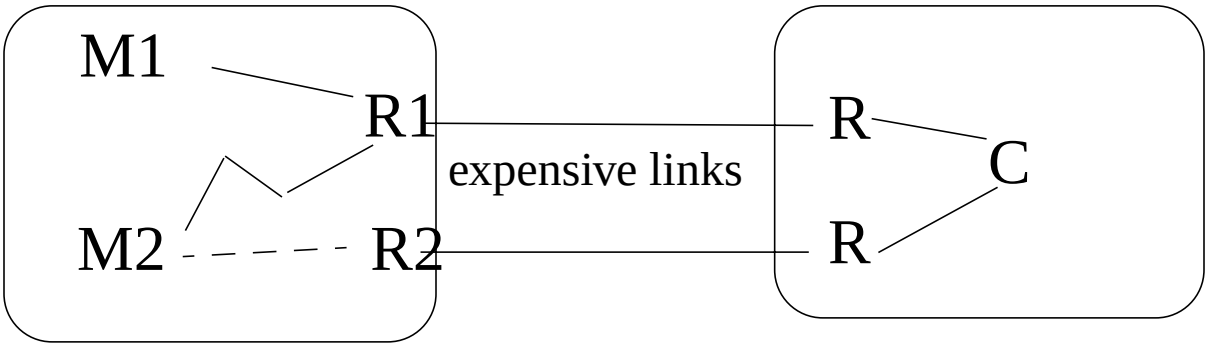
- If C is source, per source tree would have M2 join through R2. Not much better for delay than through R1, and certainly for network resources much worse through R2
- This is probably better solved by routers
  - Routers do work (find core in IGMP reply)
  - Routers route join packets differently from unicast (and core is specified in join)

Routing Decision for Join

- If policy says “use this link for multicast” and “use that link for unicast”, the join should go over the multicast link



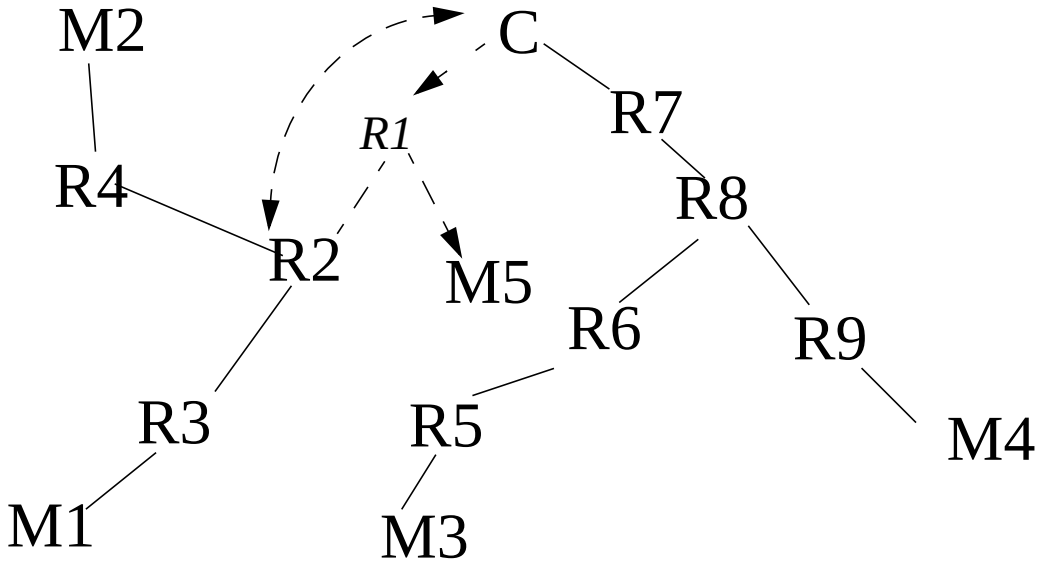
- Routing algorithm can decide good intermediate cores when there are expensive links



■

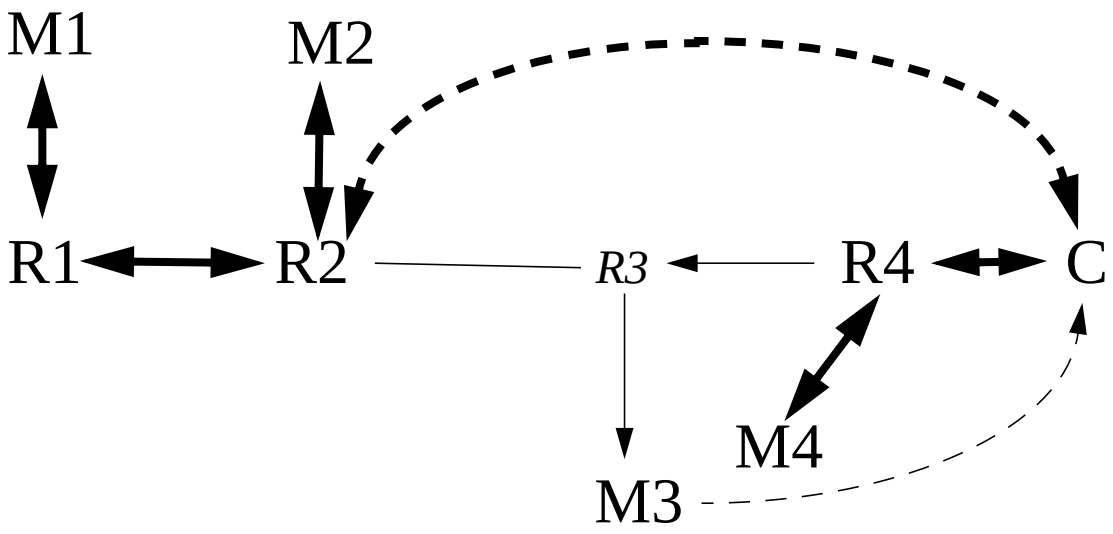
Migration: Mixing Uni and Bi-Directional Trees

- Certainly better if all routers (in a domain) are running the same routing protocols
- But can make it work. Start with “bidirectional join”. If next router can’t handle it, create tunnel to core, to bypass that router



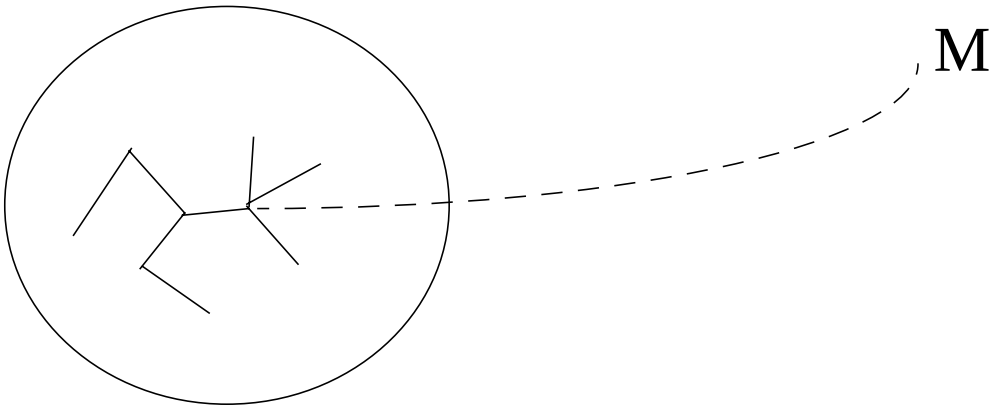
Mixing Uni-and Bidirectional Trees

- Another picture

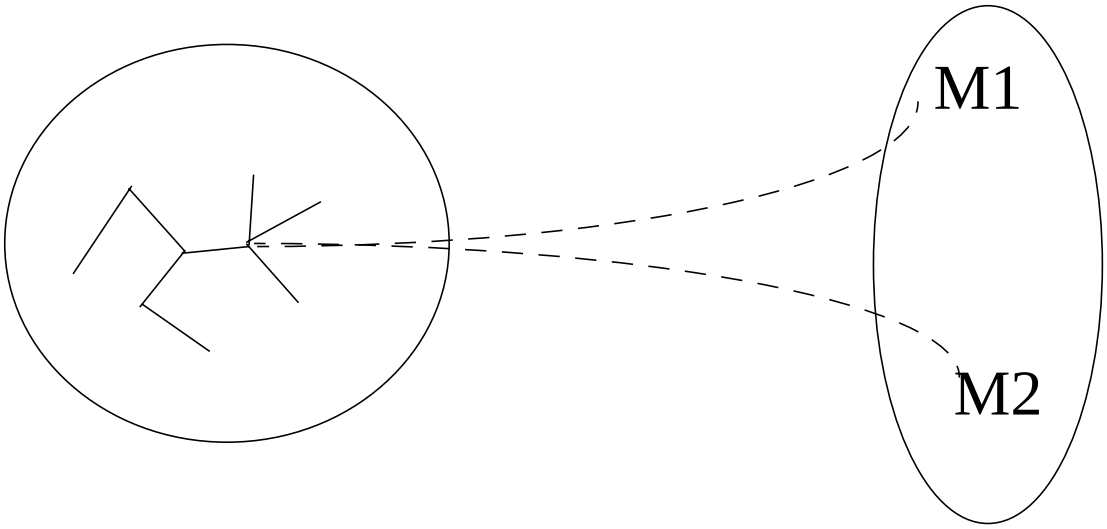


Fun with Tunnels

- You formed a group with a “local” address, and then someone outside wants to join
- Or, someone is so far away it’s not worth having all the routers on the path keeping state about the tree
- Solution: Tunnels(to core or any member, or to any node that will become a member to help out)



More Fun With Tunnels



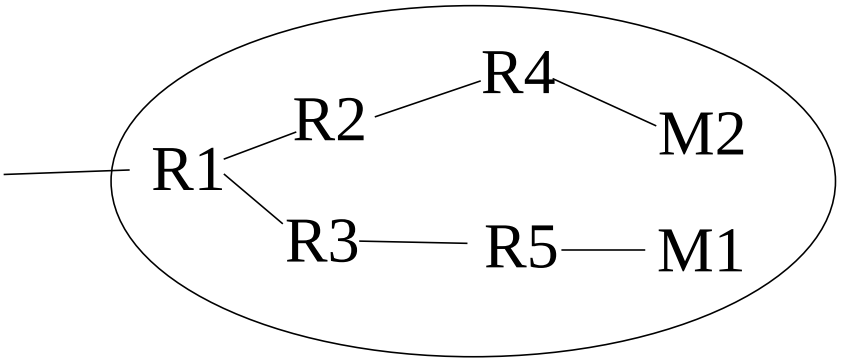
- When request tunnel, say your domain. Then tunnel endpoint can notice multiple tunnels from same domain and tell them to consolidate (form to each other, tunnel the two groups together with one tunnel).

Security

- Hiding data from unauthorized receivers
  - Don’t try to prevent them from rcv’ing traffic
  - Instead encrypt the traffic
  - Have shared group key. Group coordinator authenticates members, gives them key. Various fancy scalable algorithms exist for changing keys when member joins (so can’t decrypt pre-join traffic) and leaves (so can’t decrypt future traffic)
- Preventing unauthorized transmitters
  - if all members trusted, use group shared key
  - else, use public key. Receivers (and routers) can filter out bad traffic. Routers can filter as much as possible. Can have one shared xmit-public key, or one per xmitter, certified with one public key

Filtering Bad Traffic

- Want to get rid of as much bad (invalidly signed) traffic as possible
- Firewall is excellent place to do it
- If it can't handle load, some bad stuff will leak through, get caught by routers downstream, and ultimately by receivers
- If trust everyone in domain, can use flag in packet to indicate "packet verified". Clear on entrance to domain, set if packet checked by any router (so don't do expensive check twice)



■

Questions

- Can core be endnode?
  - Yes. If core only has a single port in the tree, it will not be forwarding packets (except those tunneled to it).
- How should core be chosen for optimal tree?
  - You can't go too wrong if core is a member of the group. If the group has high-bandwidth transmitter, have that be the core.
- IGMP impossible to modify in endnodes
  - Really, impossible to modify endnodes? Why discuss IPv6?
  - Can bypass IGMP. Send join at application layer so don't need to modify kernel
  - First router could look up core, for members that don't say it in IGMP

Questions, Cont'd

- With aggregatable multicast addresses (AMA), is the table in the routers smaller than with this proposal?
  - No. Forwarding state still has to be per group with AMA. The only thing AMA gets is knowing the direction of the group based on multicast address, but in this proposal, the direction of the group is based on core's unicast address.
- How does this compare to Dave Cheriton's Express model?
  - His has per source trees, with multicast addr source specific. Addr allocation is trivial. But difficult for something like a conference call, since you have to know all possible Xmitters, and routers have lots more state, since there has to be a separate tree for each source

Summary

- Bidirectional shared tree with core a member or well-chosen is far better than shared tree with arbitrary “core-capable” router as core
- Address allocation much simpler. No need for addresses to be aggregatable
- Get rid of some bandwidth-intensive protocols
  - core capable router advertisements
  - BGMP, MASC, AAP
- Complex protocols not necessary
  - hashing scheme to agree on core router
  - mixed (*,G) and (S,G) stuff
  - dynamically acquiring aggregatable multicast address ranges, looking for collisions
- Less state in routers