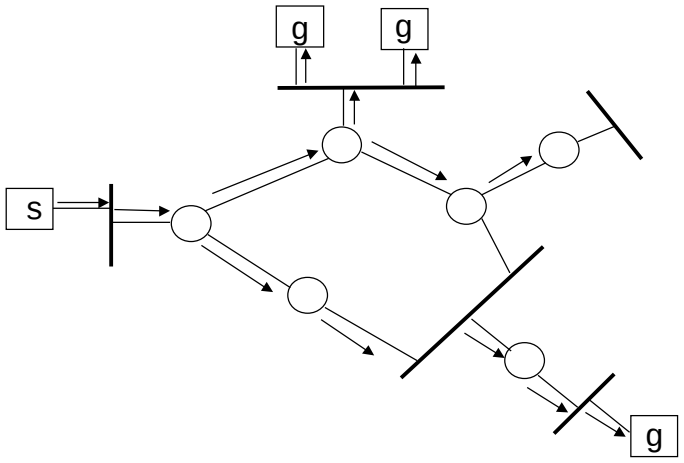
CSE/EE 461 Lecture 20
More Multicast

Tom Anderson
tom@cs.washington.edu
Peterson, Chapter 4.4

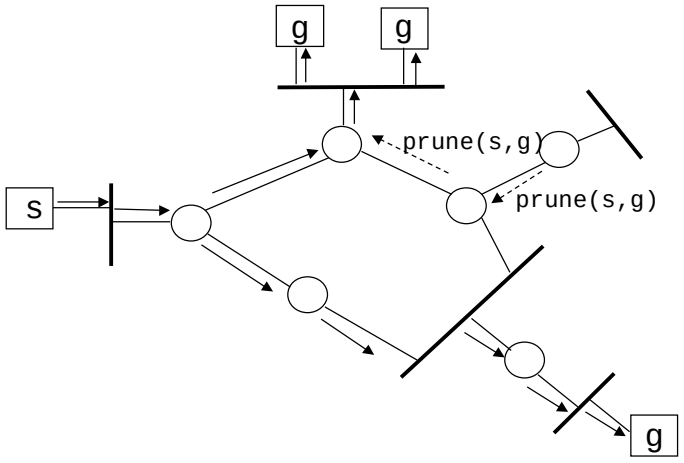
Reverse Path Multicast (RPM)

- Use distance vector to set up a broadcast tree
- Prune off branches of the tree where there are no receivers
- "Broadcast and prune"
  - Use IGMP to tell if LAN if no members
  - If no children are members, propagate prune to parent in tree
- Assume membership and prune if wrong vs. assume non-membership and explicit join

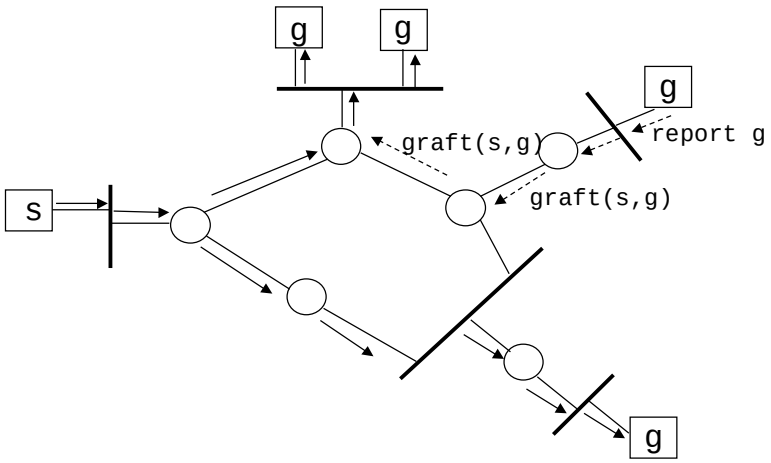
Phase 1: Truncated Broadcast



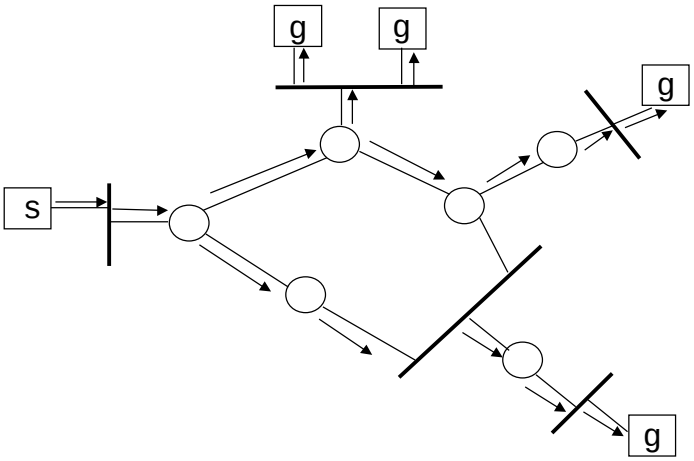
Phase 2: Pruning



Phase 3: Grafting



Phase 4: Steady State



Hierarchical Broadcast and Prune

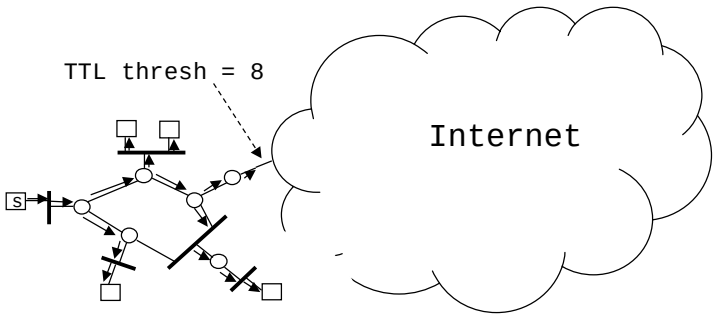
- Reverse Path Flooding
  - Discard incoming packet if not from reverse path
  - Multicast incoming packet to all borders
- Reverse Path Multicast
  - For each neighbor AS, compute if we're on its reverse path to source
  - Multicast incoming packets to all border routers for those AS's
  - Propagate prunes across the AS back towards the source

Scope Control Motivation

- Efficiency with reverse path multicast
  - sender prunes receivers
- Administrative control over listeners
  - anyone can listen to multicast conversation!
  - snooping more difficult in unicast
- Coordinate sub-group actions
  - elect a leader/suppress duplicate actions
  - locate nearest receiver

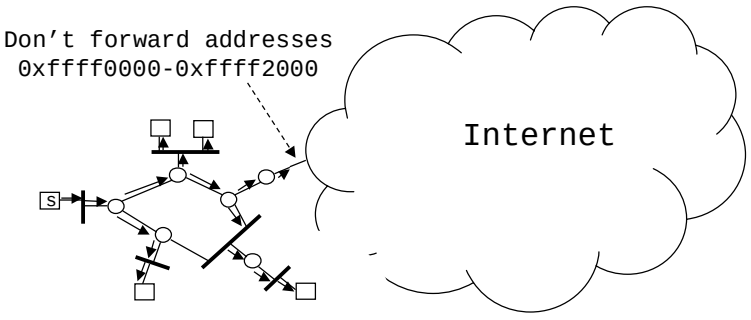
Scope Control Mechanism #1

- Administrative TTL boundaries
  - Sender uses TTL = max local diameter
  - At border router, forward pkts out iff > TTLmax



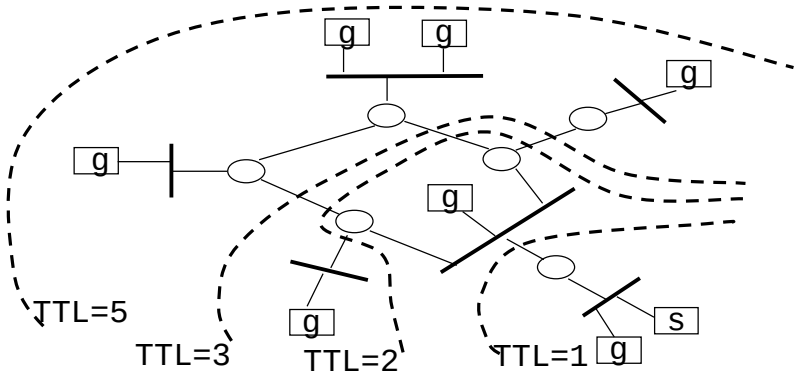
Scope Control Mechanism #2

- Allocate block of “local” addresses
  - At border router, forward only global addresses



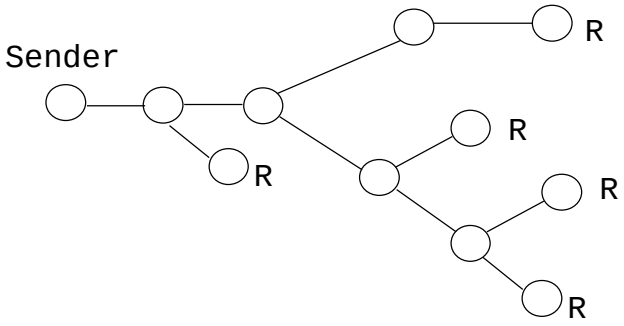
Expanding Ring Multicast

- Locate “nearest” receiver by sending to more and more of group



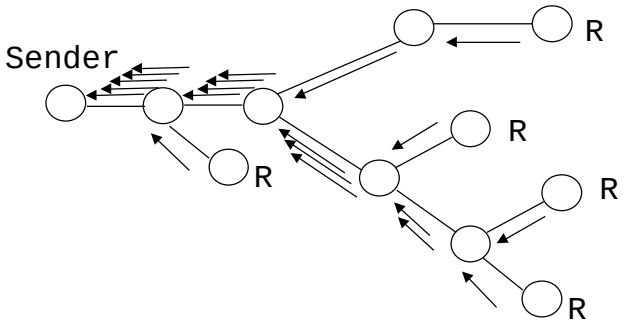
Reliable Multicast

- How do we make sure each receiver gets a copy of each message?



Ack Implosion

- If each receiver acks each packet, sender gets overwhelmed!

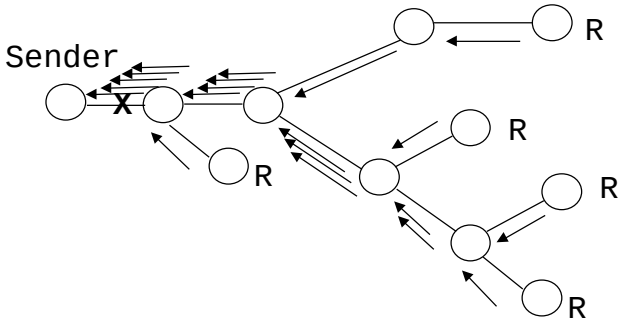


Negative Acks

- Possible solution: only send back to source if *missing* data
  - missing sequence number (2, 3, 5, 6, 7, …)
  - ping if no data being sent, to detect if missing last packet
- Fewer packets if losses are infrequent
  - note TCP uses acks for pacing new sends

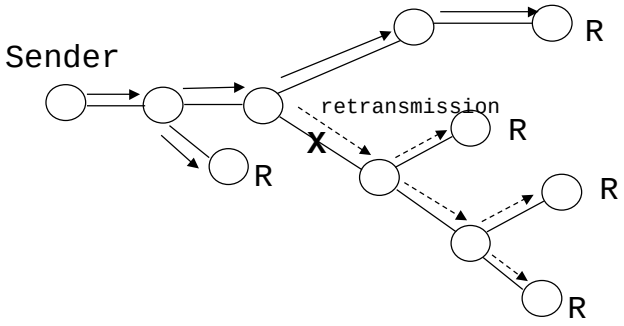
Nack Implosion

- If lose packet near sender, overwhelm sender with nacks!



Hop by Hop Retransmission

- Router keeps copy of all packets
- Resends if negative ack or timeout



Scalable Reliable Multicast

- Use multicast services to (scalably!) recover from packet losses!
  - If missing packet, multicast NACK
    - anyone get the packet?
  - Receivers with packet will multicast reply
    - anyone else missing the packet?
  - Doesn't matter who NACKs and who replies
    - anyone missing the packet can get the reply
- Assumes packets are signed by source
  - otherwise, any receiver in group could supply bogus packets that appear to come from sender

SRM Scalability?

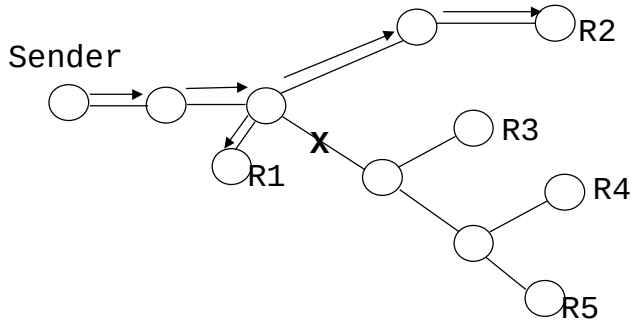
- If everyone multicasts NACK
  - NACK implosion everywhere!
- If everyone multicasts reply
  - data implosion everywhere!
- Goal: minimize simultaneous NACKs and replies
  - want one node to quickly NACK, reply
  - others to stay silent

SRM Scalability

- Use random delay before sending NACK/reply
  - want at least one node to send (short delay)
  - want at most one node to send (long delay)
- Bias delay to reduce competition
  - NACK delay based on distance to source
  - Reply delay based on distance to NACK
  - distance estimated using periodic session messages

SRM Example

- R3 detects loss, multicasts NACK
- R1 sees NACK, multicasts reply

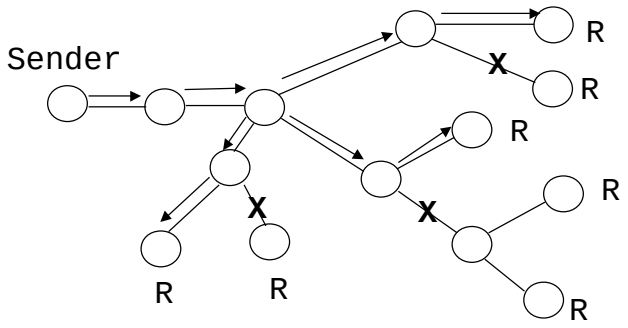


SRM Timer Adaptation

- Want system to be robust to topologies, group sizes, congestion
  - Adapt average delays to minimize redundant NACKs, replies
  - Analogous to RTT estimation in TCP
- Examples
  - if too many NACKs, increase average delay
  - if NACK once, reduce delay so NACK again

What if multiple drops?

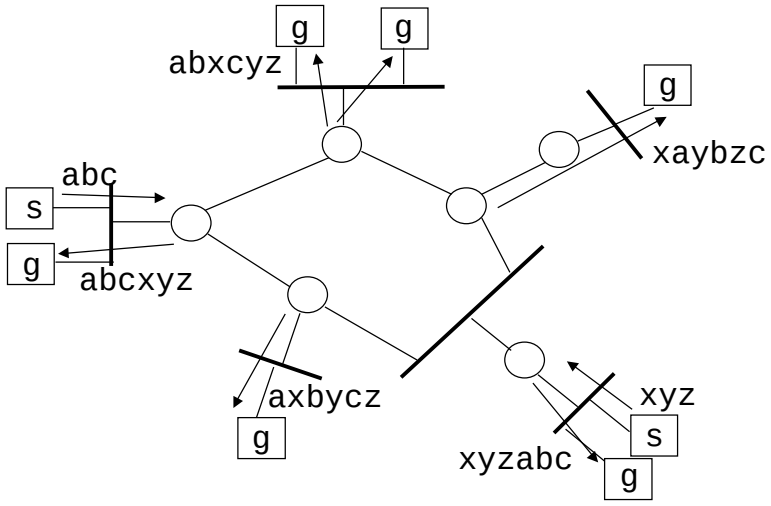
- Can use TTL expanding ring search for local recovery



Multicast Packet Ordering

- Easy to order unicast packets => seq #s
- Easy to order multicast packets from a single source => seq #s
- What if multiple sources?
  - Packets can arrive in different order at different receivers
  - Is this bad?
  - If so, what can we do to fix it?

Multicast Ordering Example



Example: Email Groups