Lecture 6: Multicast

- Challenge: how do we efficiently send messages to a group of machines?
  - Need to revisit all aspects of networking
  - Last time
    - Routing
  - This time
    - Reliable delivery
    - Ordered delivery
    - Congestion control

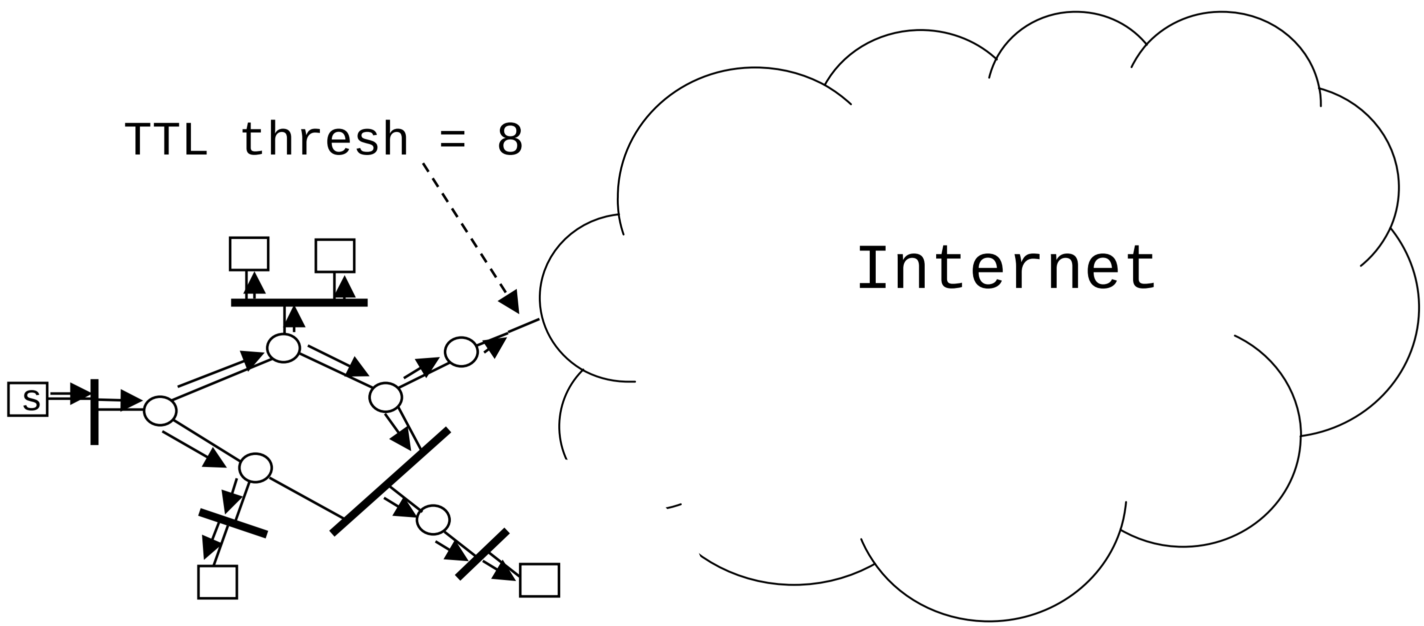
How should you route traffic in an application-level multicast tree?

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

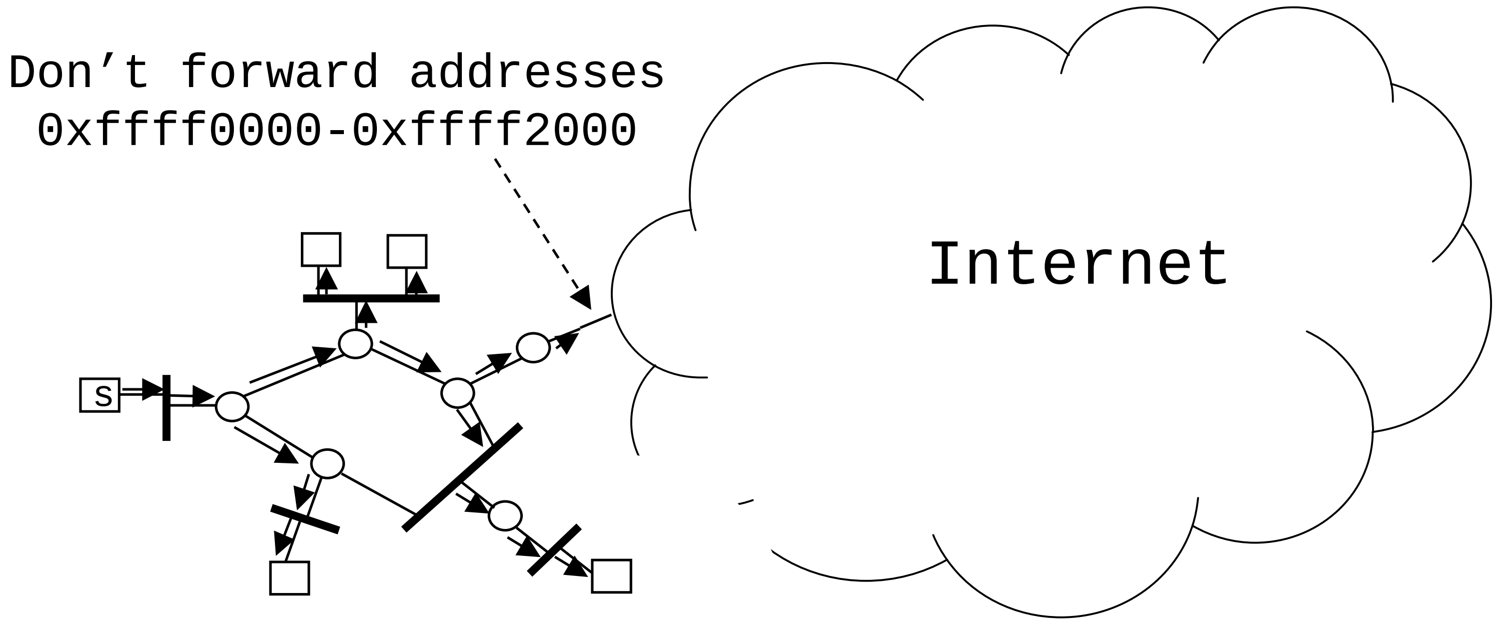
Scope Control Mechanism #1

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



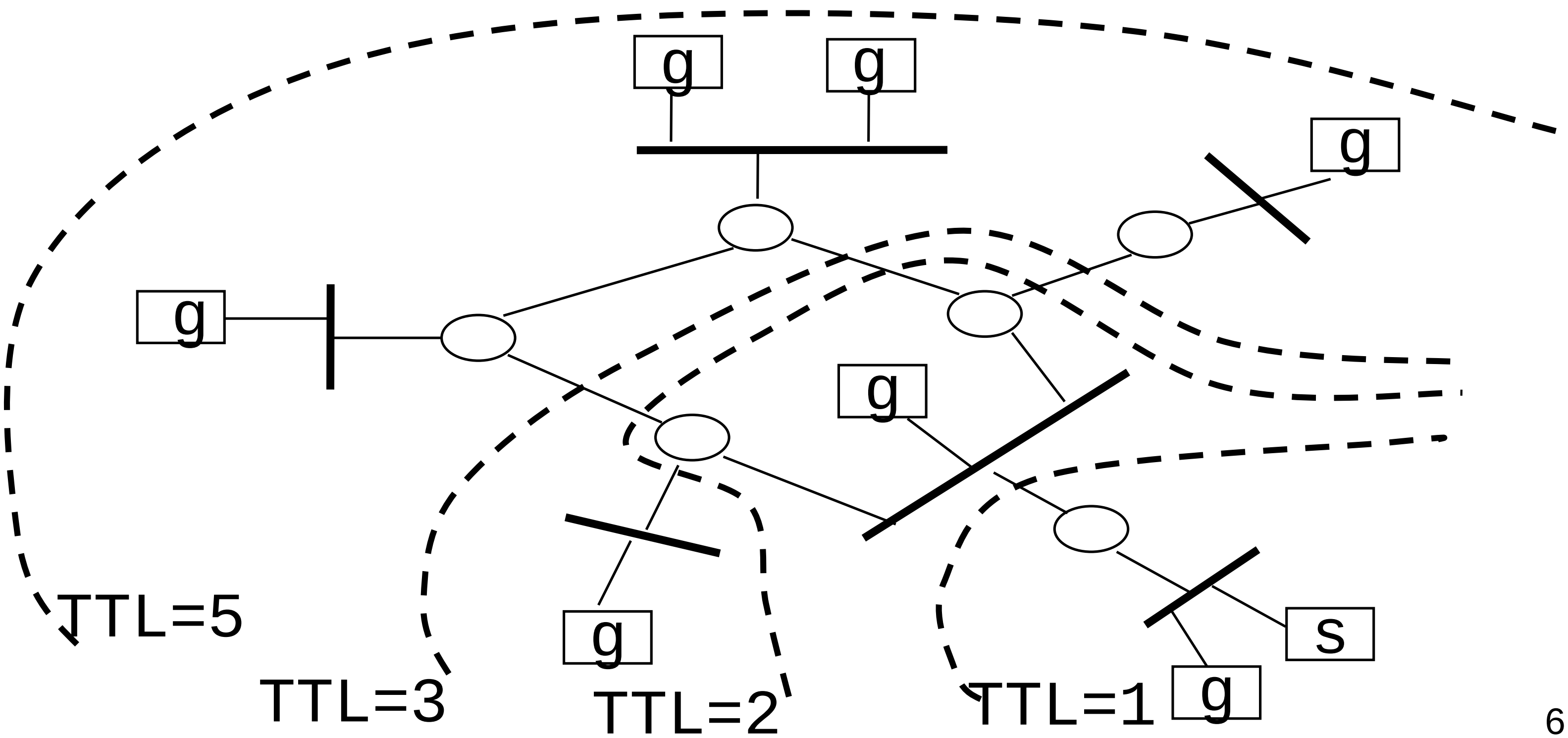
Scope Control Mechanism #2

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



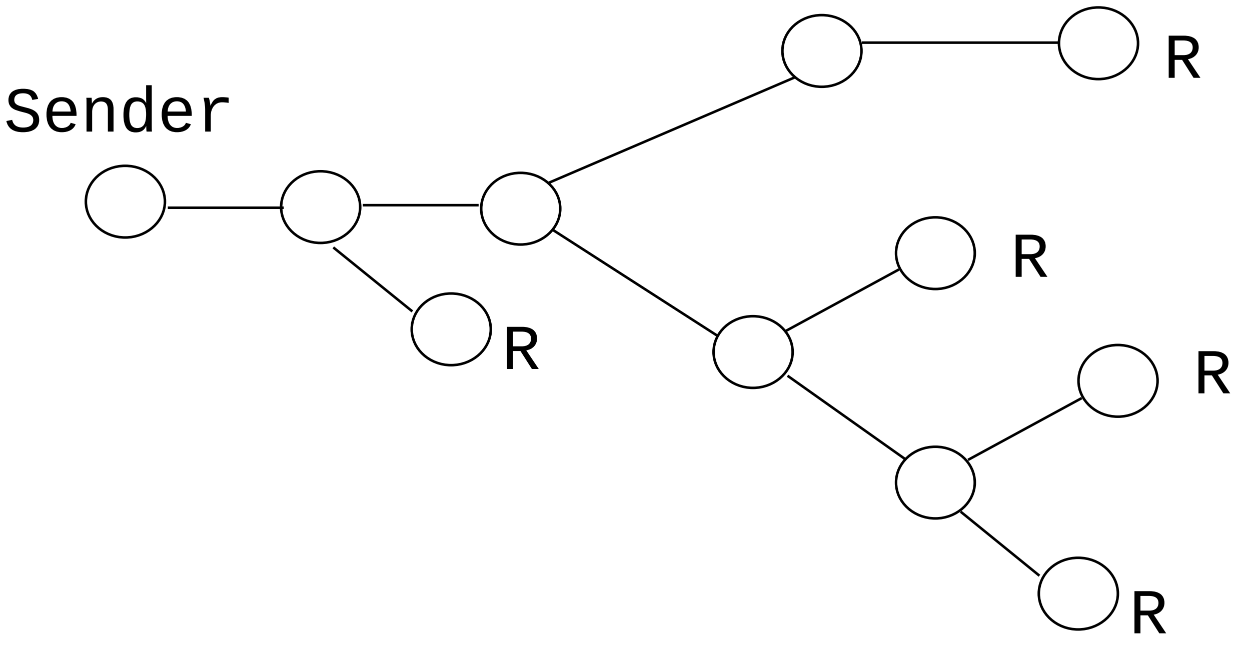
Expanding Ring Multicast

- Locate "nearest" receiver by sending to more and more of group



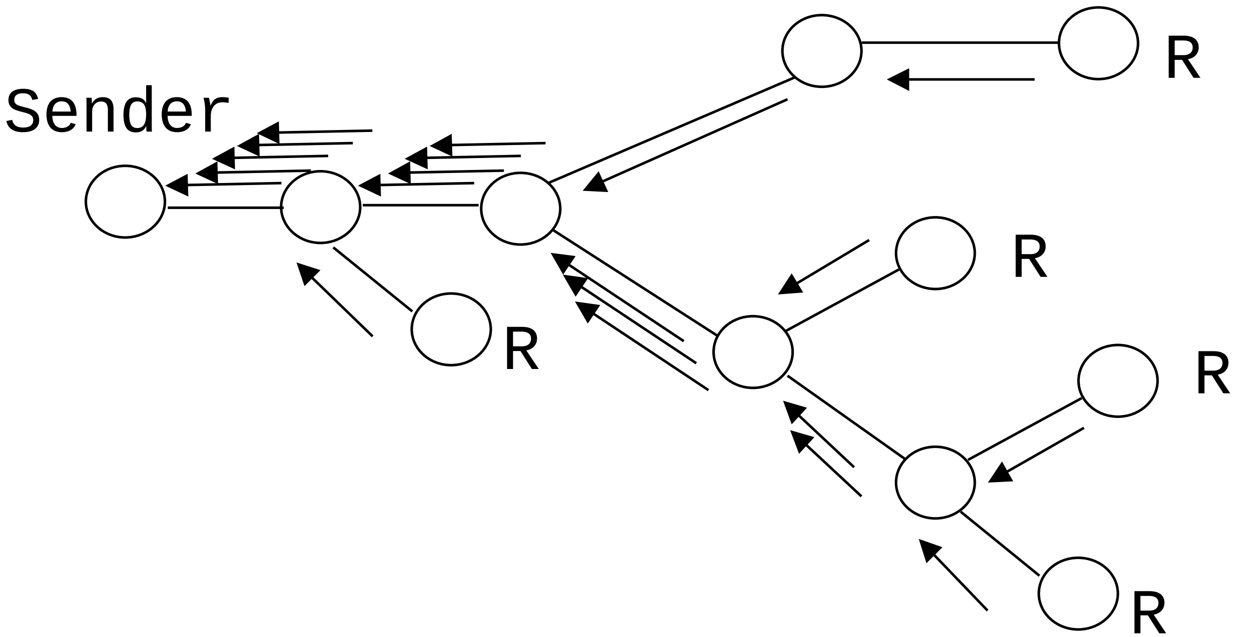
Reliable Multicast

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



Ack Implosion

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

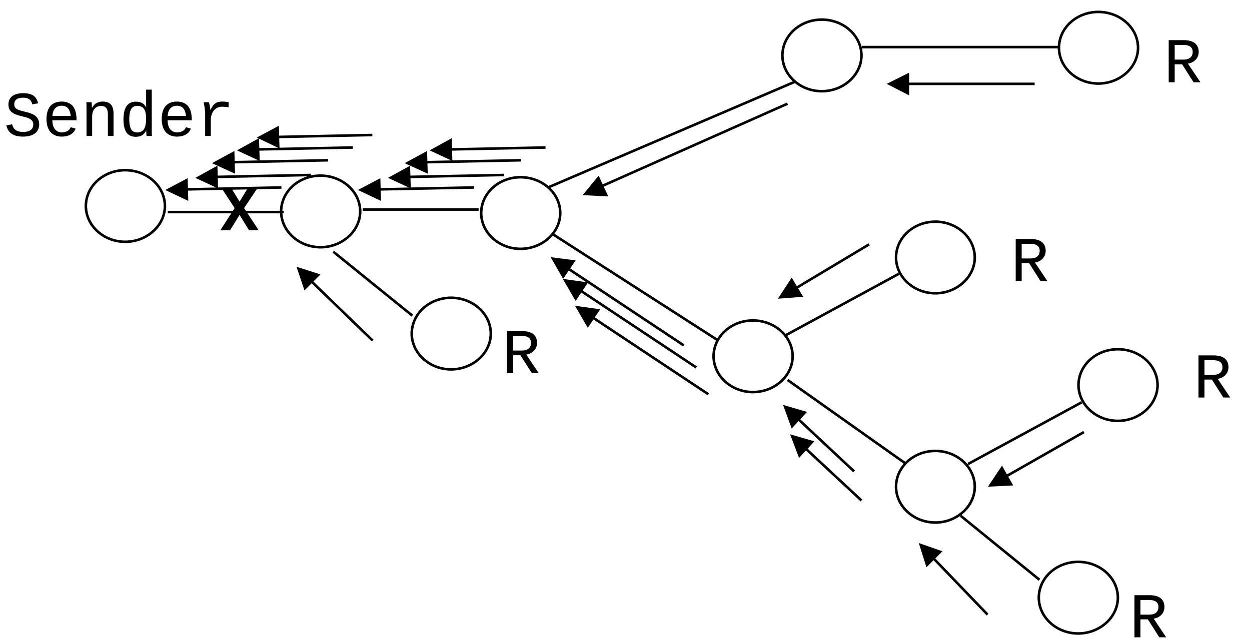


Negative Acks

- 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

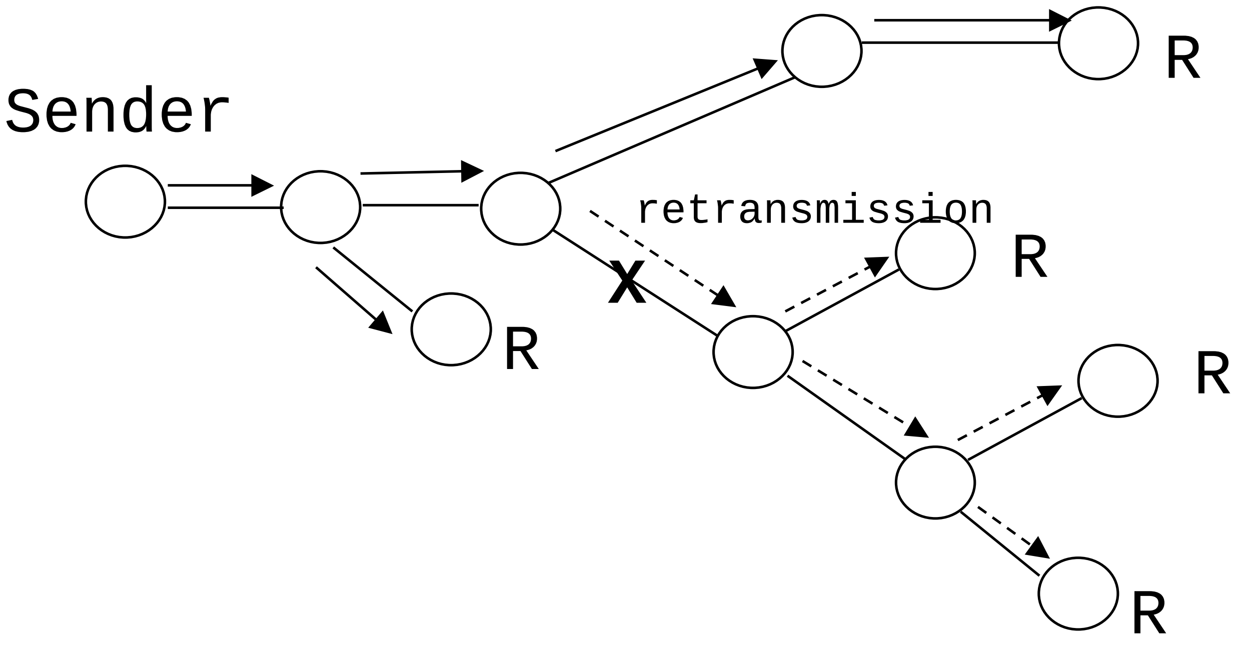
Nack Implosion

- 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 recover from packet losses!
  - If missing packet, multicast NACK
    - anyone get the packet?
    - Tell others to suppress NACK
  - Receivers with packet will multicast reply
    - anyone else missing the packet?
    - Tell others to suppress reply
- Assumes packets are signed by source

SRM Scalability?

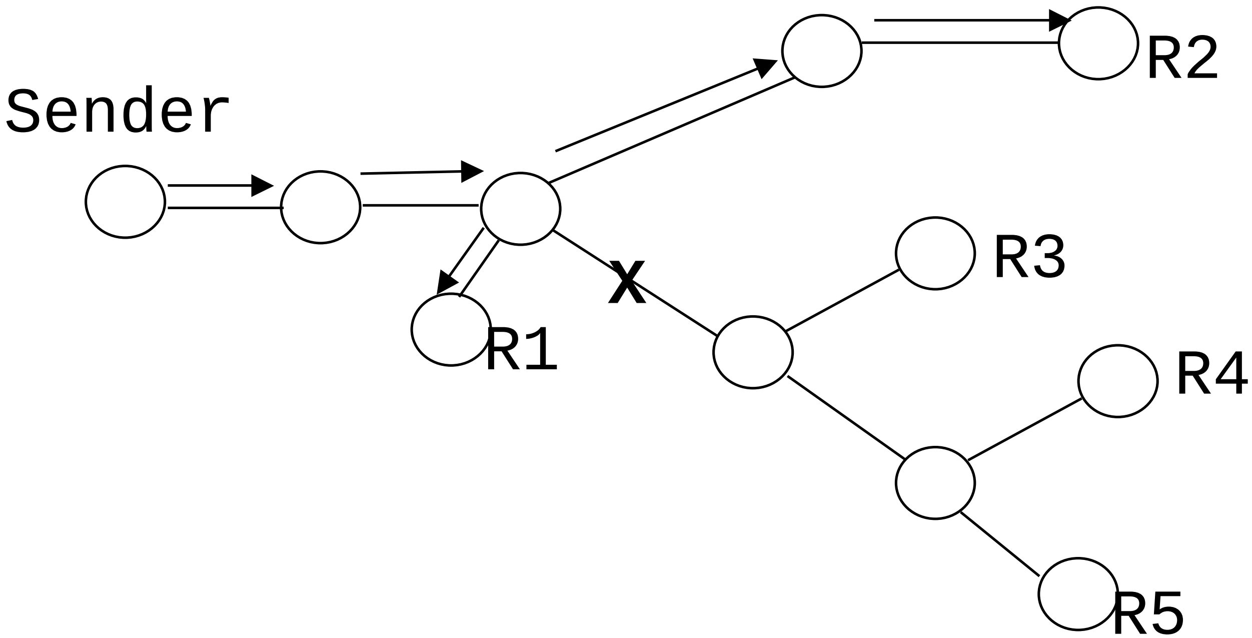
- If everyone multicasts NACK
  - NACK implosion everywhere!
- If everyone multicasts reply
  - data implosion everywhere!
- Goal: minimize simultaneous NACKs and replies

SRM Scalability

- Use random delay before sending NACK/reply
  - less likely for more than one to send at once
- Bias delay to reduce competition
  - NACK delay based on distance to source
  - Reply delay based on distance to NACK
  - distance est. using periodic session msgs
- Doesn't matter which host NACKs, replies

SRM Example

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

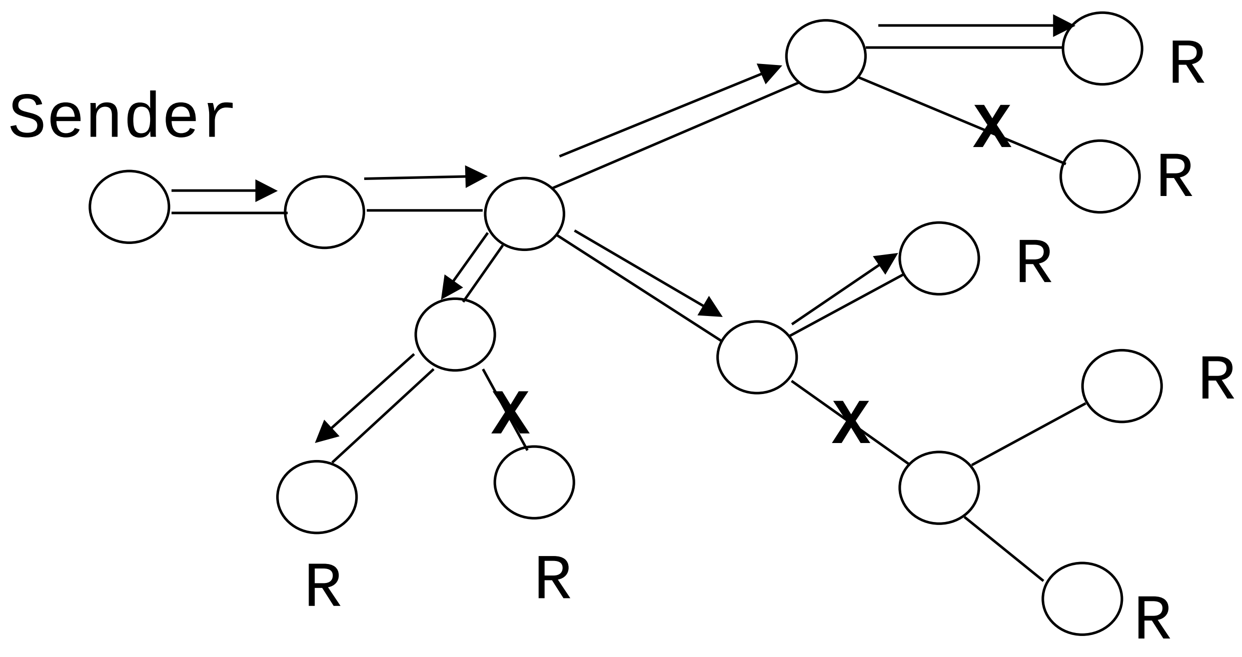


SRM Timer Adaptation

- 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



SRM Evaluation

- Scalability?
  - What happens as we increase # of hosts?
  - What about diversity of link bandwidths?
- Stability?
  - What happens as load increases?
- Security
  - reply with corrupted packet
  - denial of service => NACK everything!

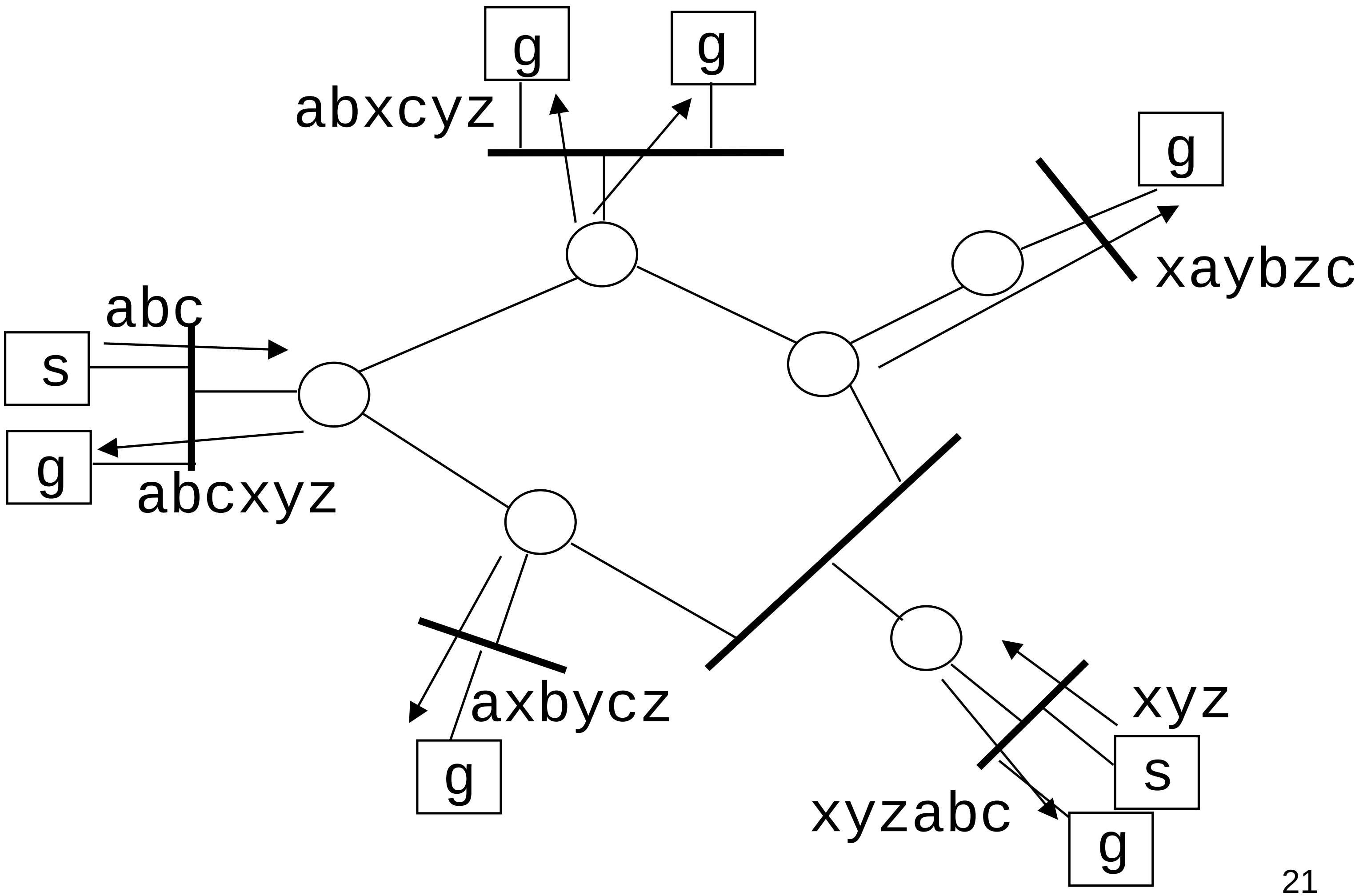
Application Level Framing

- Allow application to control how data is packetized on network
  - each RPC, object on packet boundaries
  - by contrast, TCP/IP transmits byte stream
- Advantages?
- Disadvantages?

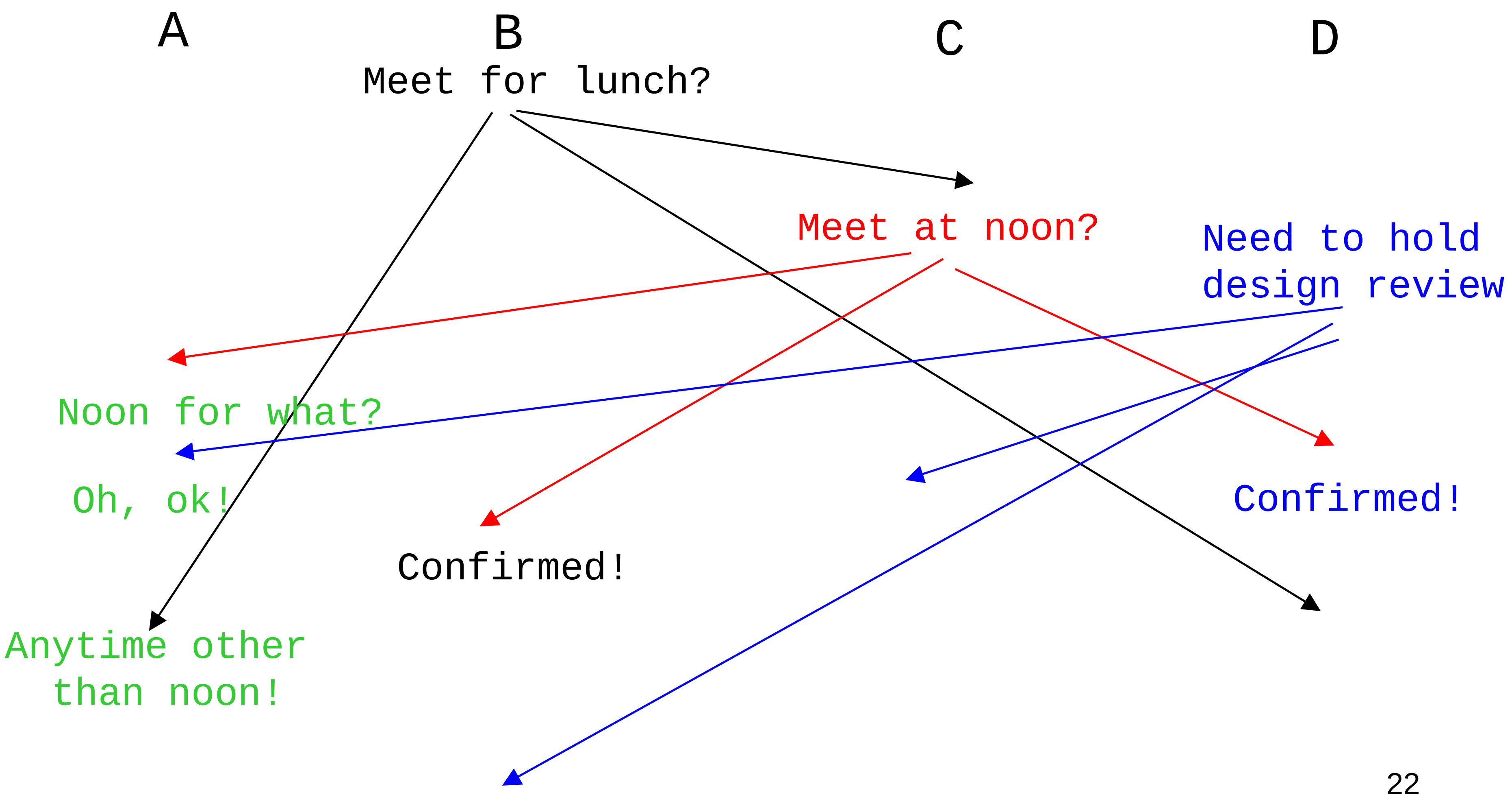
Multicast Packet Ordering

- 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



Example: Deterministic Replicas

- Replicate server for higher throughput, fault tolerance
  - Read from any replica
  - Write to all replicas
- How do we keep replicas consistent?
  - Provide all replicas exactly same sequence of messages
  - Each replica behaves deterministically, reaches same state as all other replicas

Multicast Total Ordering

- All packets are delivered in same order everywhere
- Single seq # for all packets to group
  - every source sends packets to arbiter
  - arbiter assigns sequence #
  - if arbiter fails, elect new one
  - receivers don’t process packets out of order

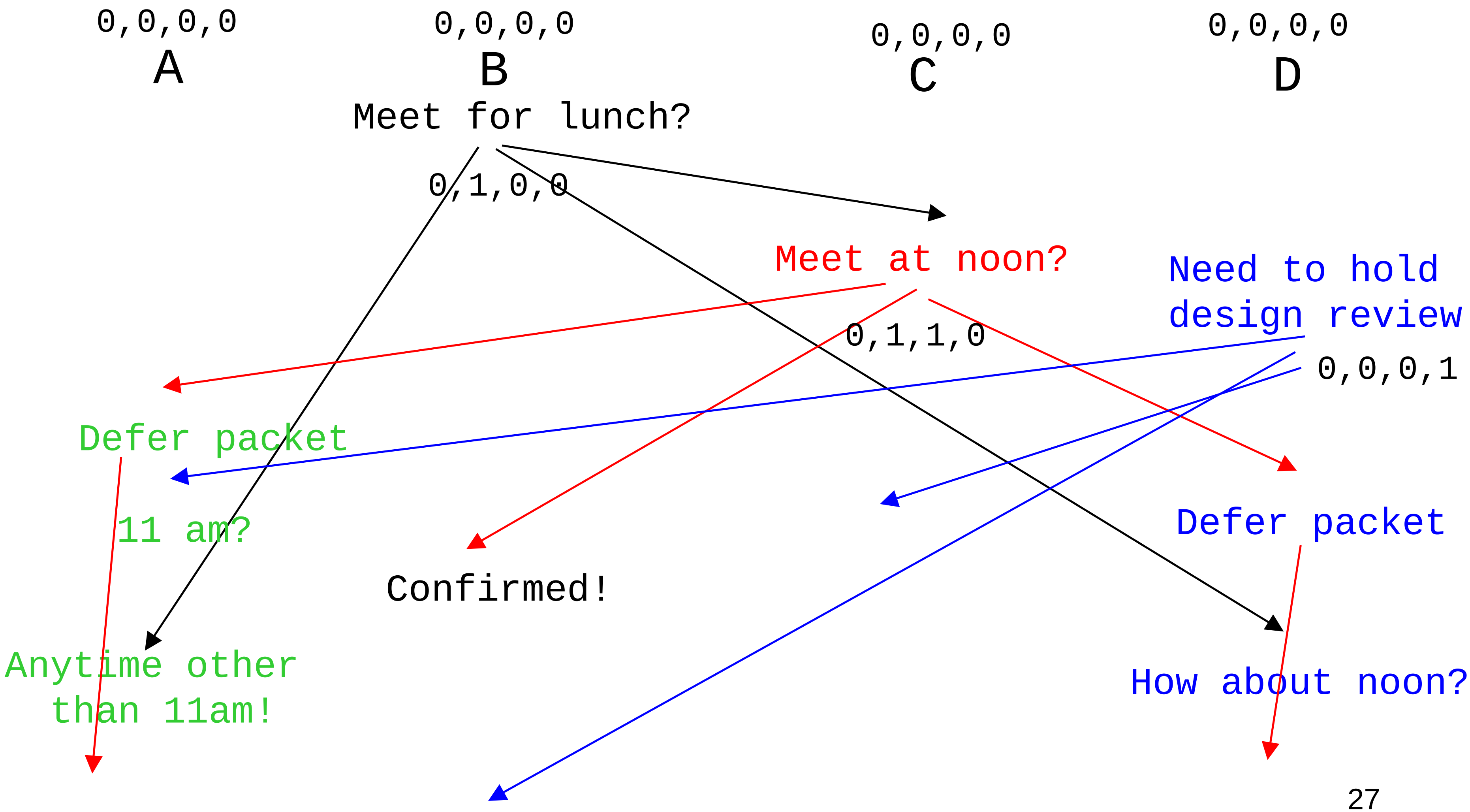
Multicast Causal Ordering

- Total ordering inefficient for subcasts
- Instead, causal ordering
  - packets are never delivered before packets that could have “caused” them
    - receiver must have gotten all the packets source has seen
  - packets that originate concurrently can be delivered in any order

Implementing Causal Ordering

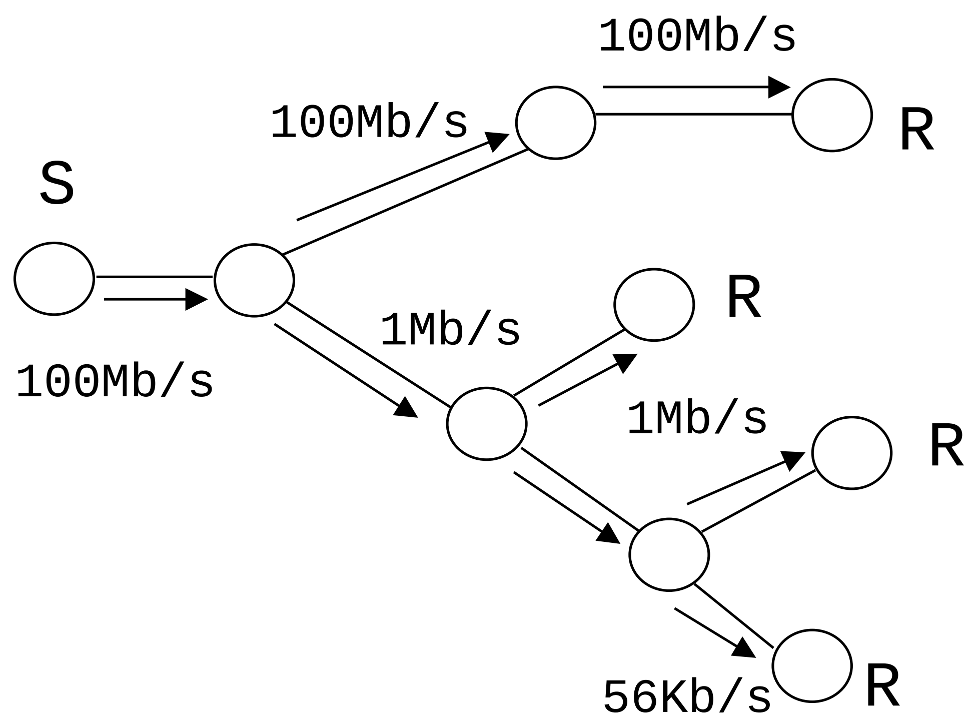
- Packets carry per-host sequence #
  - increment on each send
- Each host maintains a “version vector”
  - max seq #’s seen (in order) from each host
  - put version vector in each outgoing packet
- At receiver, delay packet until host vector > packet vector, for all sources

Causal Ordering Example



Multicast Congestion Control

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



Layered Multimedia

- Transmit signal at multiple granularities
  - 56Kb/s - voice only
  - 1Mb/s - choppy video
  - 100Mb/s - high quality video
- Layers can be
  - independent (redundant)
  - dependent (progressive refinement)

Drop Policies for Layered Multicast

- Priority
  - packets for low bandwidth layers are kept, drop queued packets for higher layers
  - requires router support (hard to deploy!)
  - wastes upstream resources
- Uniform (e.g., drop tail, RED)
  - packets arriving at congested router are dropped regardless of their layer

Receiver-Driven Layered Multicast

- Each layer a separate group
  - receiver subscribes to max group that will get through with minimal drops
- Dynamically adapt to available capacity
  - use packet losses as congestion signal
- Assume no special router support
  - packets dropped independently of layer

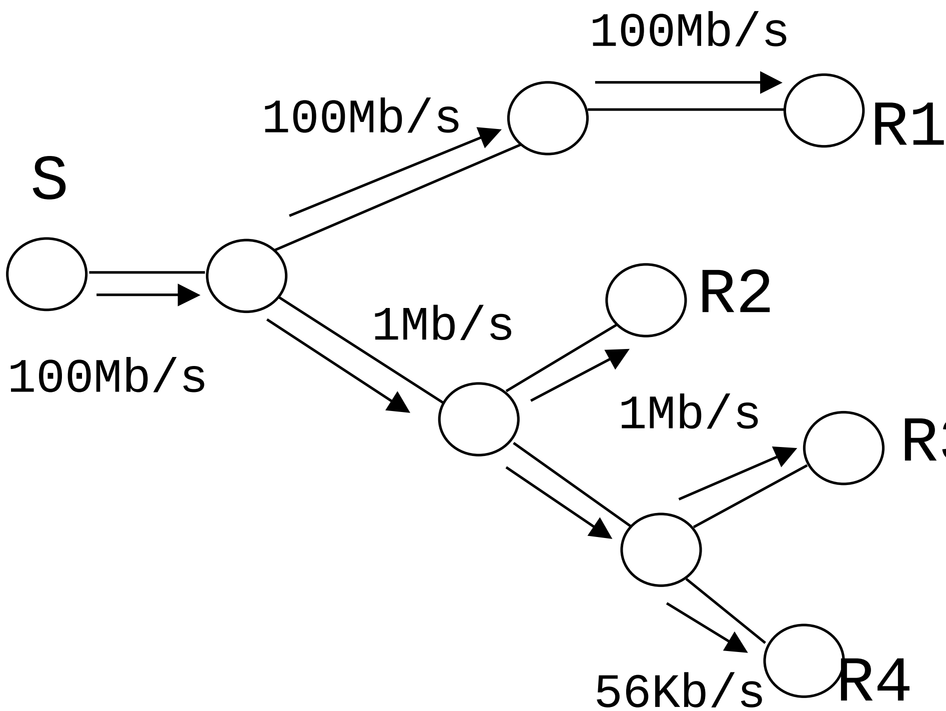
How does receiver know which layers to add?

- User decides based on observed quality?
  - Won't add layer if no benefit
- System dynamically adapts to available capacity
  - Use packet drops as congestion signal
  - No drops => try subscribing to higher layer
  - Drops => unsubscribe to layer

RLM Join Experiment

- Receivers periodically try subscribing to higher layer
- If enough capacity, no congestion, no drops => keep layer (& try next layer)
- If not enough capacity, congestion, drops => drop layer (& increase time to next retry)
  - what about impact on other receivers?

RLM Join Example



R1 joins layer 1,
joins layer 2
joins layer 3

R2,R3 join layer 1,
join layer 2
fails at layer 3

R4 joins layer 1,
fails at layer 2

RLM Scalability?

- What happens with more receivers?
- Increased frequency of experiments?
  - more likely to conflict (false signals)
  - network spends more time congested
- Reduce # of experiments per host?
  - Takes longer to converge

RLM Receiver Coordination

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

RLM Interactions

- With other multicast groups?
- With unicast TCP traffic?
- With RED?
- With fair queuing?
- With priority for lower layers?

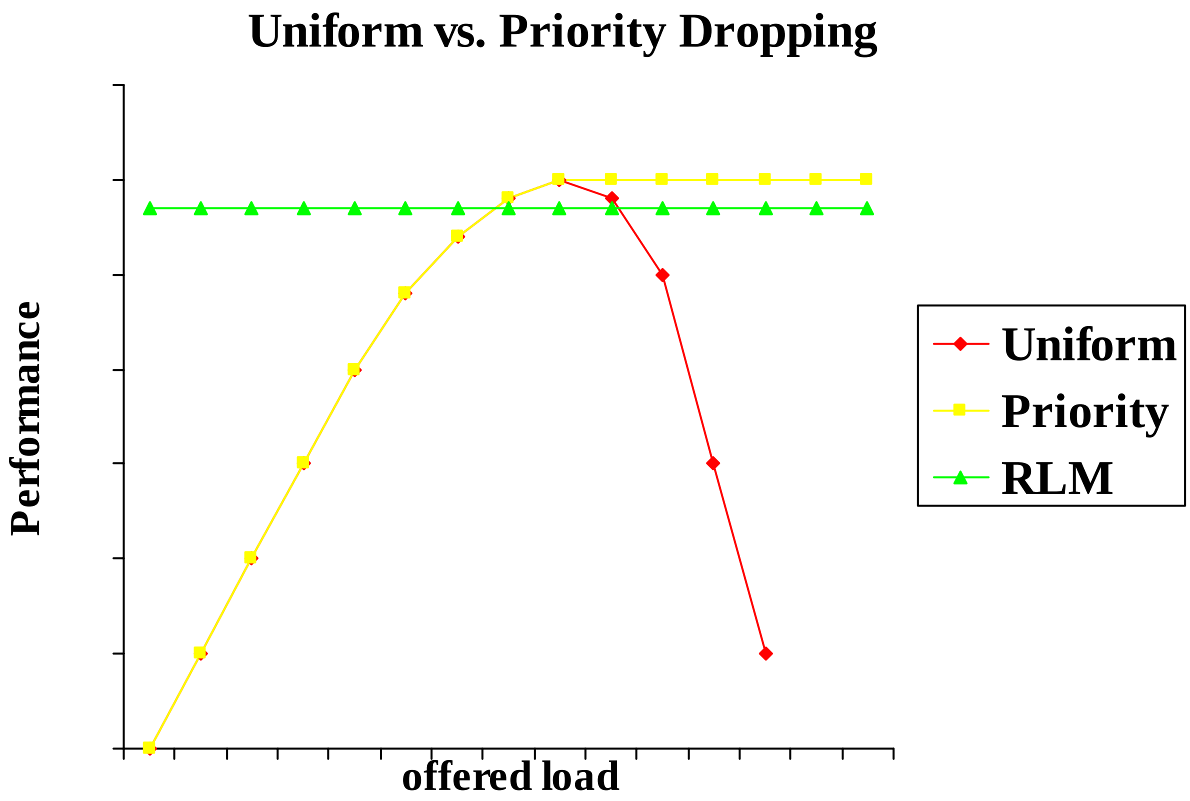
Drop Policies for Layered Multicast

- Priority
  - packets for low bandwidth layers are kept, drop queued packets for higher layers
- Uniform
  - packets arriving at congested router are dropped regardless of their layer
- Which is better?
  - Intuition vs. reality!

RLM Intuition

- Uniform offers better incentives to well-behaved users
  - if oversend, performance rapidly degrades
- Uniform offers clearer congestion signal
  - allows shared learning
- RLM approaches optimal operating point
  - uniform is already deployed

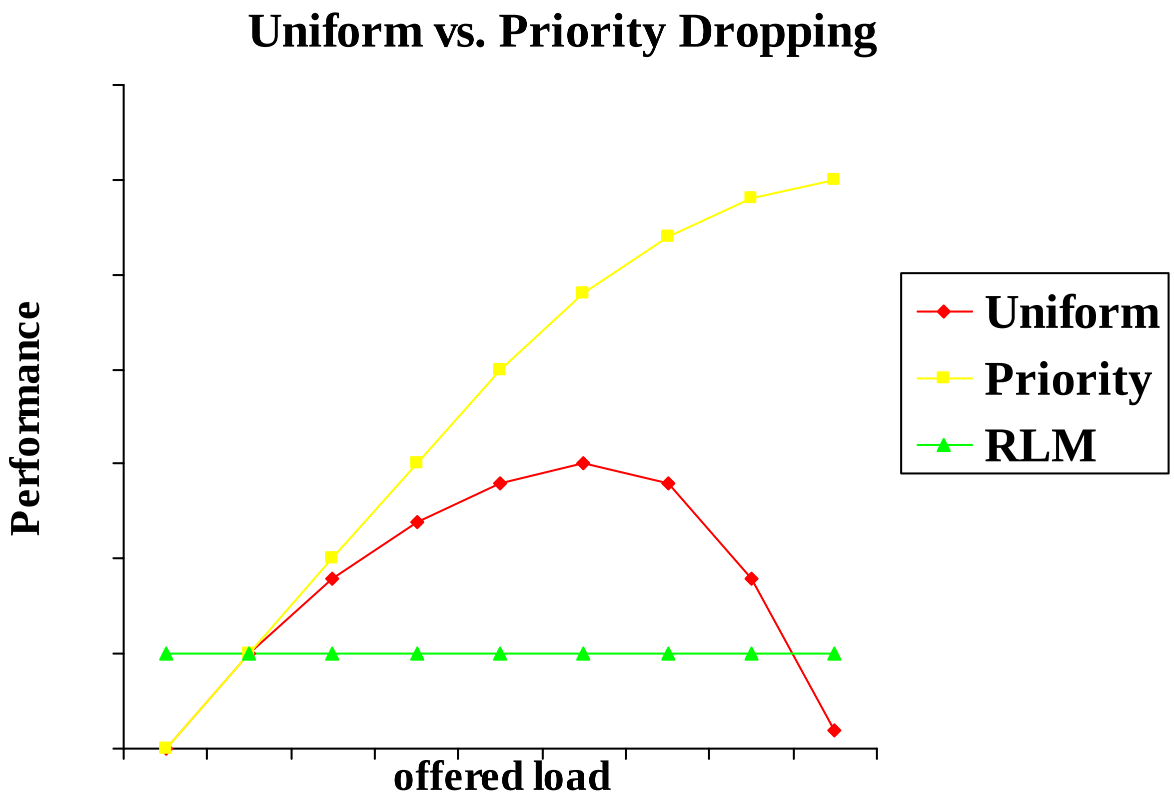
RLM Intuition



Bajaj et al. Intuition

- Priority offers much better performance
  - particularly with bursty traffic
- RLM will perform badly with bursty traffic
  - unable to adapt to congestion transients
- Uniform offers better incentives
  - but socially optimal != individually optimal

Bajaj et al. Intuition



Model Details

- Compare alternatives using total utility
- Utility of low layers > utility of high layers
- Utility of layer even if some drops
  - independent of drops on other layers
- Drop tail router
- Both simulation and analytical model

Model Results

- Smooth traffic: RLM is close to optimal
- Bursty traffic: RLM < uniform < priority
  - but not by much!
  - RLM worse if low layers not valuable
- Incentive to oversubscribe with both uniform and priority
- Slight disincentive to send/receive => individual = socially optimal

Multicast Summary

- Multicast needed for efficiency, group coordination
- Need to revisit all aspects of networking
  - Routing
  - Administration
  - Reliable delivery
  - Ordered delivery
  - Congestion control