Motivation

- Many applications require one-to-many communication
  - E.g., video/audio conferencing, news dissemination, file updates, etc.
- Using unicast to replicate packets not efficient → thus, IP multicast needed
  - What about the e2e arguments?

Semantics

- Open group semantics
  - A group is identified by a location-independent address
  - Any source (not necessary in the group) can multicast to all members in a group
- Advantages:
  - Query an object/service when its location is not known
- Disadvantage
  - Difficult to protect against unauthorized listeners

Problem

- Multicast delivery widely available on individual LANs
  - Example: Ethernet multicast
- But not across interconnection of LANs
  - I.e., can’t do Internet multicast
Three Approaches
[Deering & Cheriton ’89]
- Single spanning-tree (SST)
- Distance-vector multicast (DVM)
- Link-state multicast (LSM)
- Also: Sketches hierarchical multicast

Multicast Service Model
- Built around the notion of group of hosts:
  - Senders and receivers need not know about each other
  - Sender simply sends packets to "logical" group address
  - No restriction on number or location of receivers
      - Applications may impose limits
  - Normal, best-effort delivery semantics of IP
      - Some recovery mechanisms as unicast

Multicast Service Model (cont’d)
- Dynamic membership
  - Hosts can join/leave at will
- No synchronization or negotiation
  - Can be implemented a higher layer if desired

Key Design Goals
1. Delivery efficiency as good as unicast
2. Low join latency
3. Low leave latency
Network Model

- Interconnected LANs
- LANs support link-level multicast
- Map globally unique multicast address to LAN-based multicast address (LAN-specific algorithm)

Distance Vector Multicast Routing

- An elegant extension to DV routing
- Use shortest path DV routes to determine if link is on the source-rooted spanning tree

Reverse Path Flooding (RPF)

- A router forwards a broadcast packet from source (S) if it arrives via the shortest path from the router back to S
- Packet is replicated out all but the incoming interface
- Reverse shortest paths easy to compute \( \Rightarrow \) just use info in DV routing tables
  - DV gives shortest reverse paths
  - Works if costs are symmetric

Problem

- Flooding can cause a given packet to be sent multiple times over the same link

Solution: Reverse Path Broadcasting

Forward packets that arrive on shortest path from “t” to “S” (assume symmetric routes)
Reverse Path Broadcasting (RPB)

- Basic idea: forward a packet from \( S \) only on child links for \( S \)
- Child link of router \( R \) for source \( S \): link that has \( R \) as parent on the shortest path from the link to \( S \)

Identify Child Links

- Routing updates identify parent
- Since distances are known, each router can easily figure out if it’s the parent for a given link
- In case of tie, lower address wins

Problem

- This is still a broadcast algorithm - the traffic goes everywhere
- First order solution: Truncated RPB

Truncated RPB

- Don’t forward traffic onto network with no receivers
  - Identify leaves
  - Detect group membership in leaf
Reverse Path Multicast (RPM)

- Prune back transmission so that only absolutely necessary links carry traffic
- Use on-demand pruning so that router group state scales with number of active groups (not all groups)

Basic RPM Idea

- Prune (Source, Group) at leaf if no members
  - Send Non-Membership Report (NMR) up tree
- If all children of router R prune (5,6)
  - Propagate prune for (5,6) to parent R
- On timeout:
  - Prune dropped
  - Flow is reinstated
  - Downstream routers re-prune
- Note: again a soft-state approach

Details

- How to pick prune timers?
  - Too long \(\rightarrow\) large join time
  - Too short \(\rightarrow\) high control overhead
- What do you do when a member of a group (re)joins?
  - Issue prune-cancellation message (grafts)
- Both NMR and graft messages are positively acknowledged (why?)

RMP Scaling

- State requirements:
  - \(O(\text{Sources} \times \text{Groups})\) active state
- How to get better scaling?
  - Hierarchical Multicast
  - Core-based Trees
Core Based Trees (CBT)

- Ballardie, Francis, and Crowcroft,
  - "Core Based Trees (CBT): An Architecture for Scalable Inter-Domain Multicast Routing", SIGCOMM 93
- Similar to Deering's Single-Spanning Tree
- Unicast packet to core and bounce it back to multicast group
- Tree construction is receiver-based
  - One tree per group
  - Only nodes on tree involved
- Reduce routing table state from $O(S \times G)$ to $O(G)$

Example

- Group members: M1, M2, M3
- M1 sends data

Disadvantages

- Sub-optimal delay
- Single point of failure
  - Core goes out and everything lost until error recovery elects a new core
- Small, local groups with non-local core
  - Need good core selection
  - Optimal choice (computing topological center) is NP complete

Multicast Transport: Goal

- Transport protocol for multicast
  - Reliability
    - Apps: file distribution, non-interactive streaming
  - Low delay
    - Apps: conferencing, distributed gaming
  - Congestion control for multicast flows
    - Critical for all applications
Reliability: The Problems

- Assume reliability through retransmission
  - Even with FEC, may still have to deal with retransmission (why?)
- Sender can not keep state about each receiver
  - E.g., what receivers have received, RTT
  - Number of receivers unknown and possibly very large
- Sender can not retransmit every lost packet
  - Even if only one receiver misses packet, sender must retransmit, lowering throughput
- N(ACK) implosion
  - Described next

(N)ACK Implosion

- (Positive) acknowledgements
  - Ask every n received packets
  - What happens for multicast?
- Negative acknowledgements
  - Only ask when data is lost
  - Assume packet 2 is lost

NACK Implosion

- When a packet is lost all receivers in the sub-tree originated at the link where the packet is lost send NACKs

Scalable Reliable Multicast (SRM) [Floyd et al ’95]

- Receivers use timers to send NACKs and retransmissions
  - Randomized
  - Prevent implosion
  - Uses latency estimates
  - Short timer → cause duplicates when there is reordering
  - Long timer → causes excess delay
- Any node retransmits
  - Sender can use its bandwidth more efficiently
  - Overall group throughput is higher
- Duplicate NACK/retransmission suppression
Inter-node Latency Estimation

- Every node estimates latency to every other node
- Uses session reports (~5% of bandwidth)
- Assume symmetric latency

\[ d_{AB} = \frac{t_2 - t_1 - d}{2} \]

Repair Request Timer Randomization

- Chosen from the uniform distribution on
  \[ 2(C_1d_{SA} + C_2d_{SA}) \]
  - \( d \) - node that lost the packet
  - \( C_1, C_2 \) - algorithm parameters
  - \( d_{SA} \) - latency between \( S \) and \( A \)

Algorithm
- Detect loss \( \rightarrow \) set timer
- Receive request for same data \( \rightarrow \) cancel timer, set new timer, possibly with new iteration
- Timer expires \( \rightarrow \) send repair request

Timer Randomization

- Repair timer similar
  - Every node that receives repair request sets repair timer
- Latency estimate is between node and node requesting repair

- Timer properties - minimize probability of duplicate packets
  - Reduce likelihood of implosion (duplicates still possible)
  - Poor timer, randomized granularity
  - High latency between nodes
  - Reduce delay to repair
    - Nodes with low latency to sender will send repair request more quickly
    - Nodes with low latency to requester will send repair more quickly
  - When is this sub-optimal?

Chain Topology

- \( C_1 = D_1 = 1, C_2 = D_2 = 0 \)
- All link distances are 1
Star Topology
- \( C_1 = D_1 = 0 \),
- Tradeoff between (1) number of requests and (2) time to receive the repair
- \( C_2 \leq 1 \)
  - \( E(\text{# of requests}) = g - 1 \)
  - \( E(\text{time until first timer expires}) = 2C_2g \)
- \( C_2 > 1 \)
  - \( E(\text{# of requests}) = \sqrt{g} \)
  - \( E(\text{time until first timer expires}) = 1/\sqrt{g} \)

Bounded Degree Tree
- Use both
  - Deterministic suppression (chain topology)
  - Probabilistic suppression (star topology)
- Large \( C_2/C_1 \) → fewer duplicate requests, but larger repair time
- Large \( C_1 \) → fewer duplicate requests
- Small \( C_1 \) → smaller repair time

Adaptive Timers
- \( C \) and \( D \) parameters depend on topology and congestion
  → choose adaptively
- After sending a request:
  - Decrease start of request timer interval
- Before each new request timer is set:
  - If requests sent in previous rounds, and any dup requests were from further away:
    - Decrease request timer interval
  - Else if average dup requests high:
    - Increase request timer interval
  - Else if average dup requests low and average request delay too high:
    - Decrease request timer interval

Local Recovery
- Some groups are very large with low loss correlation between nodes
  - Multicasting requests and repairs to entire group wastes bandwidth
- Separate recovery multicast groups
  - e.g. hash sequence number to multicast group address
  - only nodes experiencing loss join group
  - recovery delay sensitive to join latency
- TTL-based scoping
  - send request/repair with a limited TTL
  - how to set TTL to get to a host that can retransmit
  - how to make sure retransmission reached every host that heard request